Time Series Scenario Composition Framework in Hydroinformatics Systems

Von der Fakultät für Umweltwissenschaften und
Verfahrenstechnik der Brandenburgischen
Technischen Universität Cottbus-Senftenberg
zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs genehmigte Dissertation

vorgelegt von

Master of Science

Chi-Yu LI

aus

Taipeh, TAIWAN

Gutachter:	apl. Prof. Dr.-Ing. Frank Molkenthin
Gutachter:	Prof. Dr.-Ing. Reinhard Hinkelmann
Tag der mündliche Prüfung:	04. Dezember 2014

Cottbus 2014

Declaration

I, Chi-Yu Li, hereby declare that this thesis entitled "Time Series Composition Framework in Hydroinformatics Systems" was carried out and written independently, unless where clearly stated otherwise. I have used only the sources, the figures and the data that are clearly stated. This thesis has not been published elsewhere.

Cottbus, June 25, 2014

Chi-Yu Li

Abstract

Since Z3, the first automatic, programmable and operational computer, emerged in 1941, computers have become an unshakable tool in varieties of engineering researches, studies and applications. In the field of hydroinformatics, there exist a number of tools focusing on data collection and management, data analysis, numerical simulations, model coupling, post-processing, etc. in different time and space scales. However, one crucial process is still missing — filling the gap between available mass raw data and simulation tools.

In this research work, a general software framework for time series scenario composition is proposed to improve this issue. The design of this framework is aimed at facilitating simulation tasks by providing input data sets, e.g. Boundary Conditions (BCs), generated for user-specified what-if scenarios. These scenarios are based on the available raw data of different sources, such as field and laboratory measurements and simulation results. In addition, the framework also monitors the workflow by keeping track of the related metadata to ensure its traceability.

This framework is data-driven and semi-automatic. It contains four basic modules: data pre-processing, event identification, process identification, and scenario composition. These modules mainly involve Time Series Knowledge Mining (TSKM), fuzzy logic and Multivariate Adaptive Regression Splines (MARS) to extract features from the collected data and interconnect themselves. The extracted features together with other statistical information form the most fundamental elements, MetaEvents, for scenario composition and further time series generation. The MetaEvents are extracted through semi-automatic steps forming Aspects, Primitive Patterns, Successions, and Events from a set of time series raw data. Furthermore, different state variables are interconnected by the physical relationships derived from process identification. These MetaEvents represent the complementary features and consider identified physical relationships among different state variables from the available time series data of different sources rather than the isolated ones. The composed scenarios can be

further converted into a set of time series data as, for example, BCs, to facilitate numerical simulations.

A software prototype of this framework was designed and implemented on top of the Java and R software technologies. The prototype together with four prototype application examples containing mathematical function-generated data, artificial model-synthetic hydrological data, and measured hydrological and hydrodynamic data, are used to demonstrate the concept. The results from the application examples present the capability of reproducing similar time series patterns from specific scenarios compared to the original ones as well as the capability of generating artificial time series data from composed scenarios based on the interest of users, such as numerical modelers. In this respect, it demonstrates the concept's capability of answering the impacts from what-if scenarios together with simulation tools. The semi-automatic concept of the prototype also prevents from inappropriate black-box applications and allows the consideration of the knowledge and experiences of domain experts. Overall, the framework is a valuable and progressive step towards holistic hydroinformatics systems in reducing the gap between raw data and simulation tools in an engineering suitable manner.

Zusammenfassung

Seit der erste automatische, programmierbare und betriebsfähige Computer, Z3, im Jahr 1941 entwickelt wurde, sind Computer ein unverzichtbares Werkzeug für die vielfältigen Aufgaben in der ingenieurwissenschaftlichen Forschung und Praxis geworden. Auf dem Gebiet der Hydroinformatik gibt es eine Reihe von Werkzeugen, die den Fokus u. a. auf Datenerfassung und -management, Datenanalyse, numerische Simulationen, Modellkoppelung sowie Ergebnisauswertung in unterschiedlichen Raum- und Zeitskalen legen. Ein wesentlicher Arbeitsschritt wird jedoch nur unzureichend unterstützt: die Aufbereitung von Rohdaten zur Spezifikation von Szenarien als Eingabegrößen für Simulationswerkzeuge.

In dieser Forschungsarbeit wird ein generelles Konzept für die ingenieurgerechte Erstellung von Zeitreihen zur Szenarienspezifikation vorgeschlagen. Das Ziel des Konzepts ist die Bereitstellung von Zeitreihen als Eingangsdatensätze, z. B. Randbedingungen, für Simulationsaufgaben zur Analyse von benutzerspezifizierten Was-Wäre-Wenn-Szenarien. Die Szenarien werden aus verfügbaren Rohdaten unterschiedlicher Quellen, z. B. Feld- und Labormessungen und Simulationsergebnissen, erstellt. Das Konzept protokolliert zudem den Arbeitsablauf durch zugehörige Metadaten, um die Nachvollziehbarkeit der Arbeitsschritte sicherzustellen.

Das Konzept ist datengesteuert und halbautomatisch. Es enthält vier wesentliche Module: Datenvorbereitung, Eventidentifizierung, Prozessidentifizierung und Szenariokomposition. Diese Module verwenden als theoretische Grundlagen vor allem Time Series Knowledge Mining (TSKM), Fuzzylogik und Multivariate Adaptive Regression Splines (MARS), um Merkmale verschiedener Zustandsgrößen aus den gesammelten Daten zu extrahieren und miteinander zu verbinden. Die gesammelten Merkmale samt anderen statistischen Daten gestalten die grundsätzlichsten Komponenten, sog. MetaEvents, für die Szenariokomposition und die weitere Generierung der resultierenden Zeitreihen für die Simulation der Szenarien. Die MetaEvents werden halbautomatisch mit Hilfe von Aspects, Primitive Patterns, Successions und Events gebildet. Zusätzlich werden durch

Prozessidentifizierung funktionale Beziehungen zwischen den verschiedenen Zustandsvariablen abgeleitet. Die MetaEvents stellen komplementäre Merkmale dar und berücksichtigen die identifizierten physikalischen Beziehungen zwischen den verschiedenen Zustandsvariablen aus den verfügbaren Zeitreihendaten anstatt der traditionellen getrennten Verarbeitung. Die mit den MetaEvents zusammengestellten/komponierten Szenarien ermöglichen die Generierung von resultierenden Zeitreihen von Randbedingungen für numerische Simulationen.

Ein Software-Prototyp dieses Konzepts wurde auf Basis von Java- und R-Software-Technologien entworfen und implementiert. Der Prototyp zusammen mit vier Prototyp-Anwendungsbeispielen – ein mathematisch-analytischer Datensatz, ein künstlicher hydrologischer Datensatz, ein real gemessener hydrologischer Datensatz und ein hydrodynamischer Datensatz – werden benutzt, um die Funktionsfähigkeit des Prototyps und die Eigenschaften des Konzepts zu demonstrieren und nachzuweisen. Die Anwendungsbeispiele weisen nach, das Zeitreihenmuster aus spezifischen Originalszenarien reproduziert werden können und zeigen die Fähigkeit auf, für den Anwender, z. B. numerische Modellierer, Zeitreihen für relevante, interessante Szenarien zu generieren. In dieser Hinsicht demonstriert es die Fähigkeit, die Auswirkungen von Was-Wäre-Wenn-Szenarien mit Simulationswerkzeugen effizient vorzubereiten. Das halbautomatische Konzept des Prototyps verhindert auch eine Black-Box Anwendung und berücksichtigt Kenntnisse und Erfahrungen der Anwender als Fachexperten. Damit stellt das Konzept einen wertvollen, innovativen Schritt zu ganzheitlichen Hydroinformationssystemen dar, um eine ingenieurgerechte, effiziente Datenaufbereitung von Zeitreihen aus Rohdaten als Eingabedatensätze für Simulationswerkzeuge bereitzustellen.

摘要

自從全球第一部自動化、可程式化以及可操作性的電腦Z3於西元1994年問世，電腦在各式各樣的工程研究及應用上，已成為不可或缺的工具。在傳統水資訊（Hydrdoinformatics）領域中，針對不同的目的，例如：資料收集和管理、資料分析、數值模擬、模型耦合（Model Coupling）、後處理…等等，以及不同的時間和空間尺度，已存在許多工具可處理。然而，在嘗試填補介於大量原始資料及眾多模擬工具間缺口方面，仍然缺乏一關鍵工具。

為了彌平此般闕漏，本研究提出一個通用軟體架構，藉由已收集的時間序列資料來分析並組成（Compose）時間序列的情境（Scenarios），並藉由提供可自定的假設情境（What-if Scenarios）來評估在此情境下的影響。此外，亦透過實做此一架構之原型以及四個例子來評估此一架構。該架構可視為一資料管理／生成的工具，而非數值模擬的工具。

本軟體架構的設計目標，是讓使用者可基於各種不同來源的原始資料，例如：實驗室和現地量測、數值模擬結果…等等，而自訂出所需要的假設情境，再將其假設情境轉換並產生輸入資料，例如：邊界條件（Boundary Conditions, BCs）等，來進行模擬任務。而決策者可利用在不同假設情境下的結果，來做進一步的評估以及決策。另外，此架構亦透過相關元數據（Metadata）的記錄，來監控整個模擬流程以確保其可追溯性（Traceability）。

該架構是一種由資料驅動（Data-driven）的半自動架構，其包含四個基本模組：資料預處理、事件定義（Event Identification）、程序定義（Process Identification）以及情境組成。這些模組主要利用時間序列知識探勘（Time Series Knowledge Mining, TSKM）、模糊邏輯（Fuzzy Logic）以及MARS（Multivariate Adaptive Regression Splines）來針對已蒐整的資料擷取其特徵，再進一步將其相互連結。所擷取到的資料特徵，以及其相對應的統計資訊，便成為情境組成及更進一步生成時間序列的最基本單位：元事件（MetaEvent）。這些元事件是利用一系列半自動的步驟，透過自原始資料一步步自形成相（Aspects）、基本式樣（Primitive Patterns）、序列（Successions）以及事件（Events）的過程而導出。此外，不同狀態變數間的關係亦透過由程序定義模組所導出的物理關係來描述。元事件本身並非一孤立事件，它不僅僅描述時間序列各事件間時序上相互的關係，它亦包含不同狀態變數間的關係。該組成的情境可進一步的轉換成時間序列資

料，例如：邊界條件等，以作為數值模擬的需要。

此通用軟體架構的原型，是以Java和R程式語言為基礎設計以及實做。於本論文中，利用此原型與四個不同例子，展示本架構的概念。這四例包含了由數學函數生成的資料、模式合成的資料以及量測的水文、水理資料。此通用軟體架構除了可單獨使用，亦可與其他水資訊模式整合應用。

結果顯示，此架構原型可藉由情境組成來重製出類似原始時間序列式樣（Patterns），並且並且針對使用者需要所組成的情境轉換成對應的時間序列輸入資料，以提供進一步數值模擬上的需求。因此，此架構原型已展示了本研究所提出的軟體架構，如何與其他水資訊模式併行應用，以解決並評估在自訂假設情境下的影響及未來的可能性。此外，透過其半自動的手段亦可避免不適當的黑箱（Black box）結果並且允許考量領域專家的知識經驗。總而言之，該架構在工程面以及在整合水資訊系統層面上，進一步提供一填補介於大量原始資料及眾多模擬工具之間缺口的工具。

Acknowledgement

This is a long journey, and it can not be achieved without much help from many people.

Firstly, I would like to express my deep gratitude to apl. Prof. Dr.-Ing. F. Molkenthin from BTU Cottbus-Senftenberg and Prof. Dr.-Ing. R. Hinkelmann from TU Berlin, my two supervisors, especially to apl. Prof. Dr.-Ing. F. Molkenthin, for their guidance, patience, supports and many others. My grateful thanks are also extended to Prof. Dr.-Ing. P. Holz and Prof. Dr. G. Bader. It is always helpful to discuss all kinds of problems with them.

Besides, I would also like to thank many colleagues during the stay in Germany, and they are Mikro Filetti, Anne Gädeke, Gunnar Jenet, Frank Merting, Islam Shafi Noor, Kunwar Vikramjeet Notay, and Uta Warstatt. Thanks for their support in administration, technological issues, and, of course, scientific problems.

Also, I appreciate the financial support from DAAD for my first three years' work.

I also thank many Cottbusers for their warm hospitality, especially the Farnsworths, the Hirschs, Mrs. Löschner, and the Schröders. Finally, I appreciate the tolerance from my family.

Cottbus, June 2014

Chi-Yu Li

Related Publications

[1] C.-Y. Li and F. Molkenthin. Time Series Scenario Composition Framework in Supporting Environmental Simulation Tasks. In J. M. Gómez, M. Sonnenschein, U. Vogel, A. Winter, B. Rapp, and N. Giesen, editors, *Proceedings of the 28th Conference on Environmental Informatics - Informatics for Environmental Protection, Sustainable Development and Risk Management*, pp. 247–254. BIS-Verlag, September 2014.

[2] F. Molkenthin, C.-Y. Li, and K. V. Notay. Information Handling in Interdisciplinary, Hydroenvironment Engineering Projects. In P. Gourbesville, J. Cunge, and G. Caignaert, editors, *Advances in Hydroinformatics*, Springer Hydrogeology, pp. 65–76. Springer Singapore, January 2014.

[3] C.-Y. Li, K. V. Notay, and F. Molkenthin. Time Series Scenario Composition Framework In Hydroinformatics Systems. In R. Hinkelmann, M. H. Nasermoaddeli, S. Y. Liong, D. Savic, P. Fröhle, and K. F. Daemrich, editors, *Proceedings of 10th International Conference on Hydroinformatics, HIC 2012, Understanding changing climate and environment and finding solutions, Hamburg, Germany, July 14 - 18, 2012*, Hamburg, Germany, 2012. TuTech Innovation.

[4] C.-Y. Li and K. V. Notay. Information Mining Framework Concept for Time-Series Data in Hydroinformatics Systems. In *Proceedings of Forum Bauinformatik 2010*, Volume 9 of *Heftreihe des Instituts für Bauingenieurwesen*, pp. 153–160, Berlin, Germany, 2010. Shaker Verlag GmbH, Germany.

[5] K. V. Notay, C.-Y. Li, and S. Franz. Model Coupling by the Use of Autonomous Tensor Objects. In *Proceedings of Forum Bauinformatik 2010*, Volume 9 of *Heftreihe des Instituts für Bauingenieurwesen*, pp. 161–168, Berlin, 2010. Shaker Verlag GmbH, Germany.

[6] V. Notay, C.-Y. Li, F. Molkenthin, F. Simons, and R. Hinkelmann. Model integration and coupling in a hydroinformatics system. In *Proceedings of the Ninth International Conference on Hydro-Science and Engineering*, Chennai, India, 2010.

[7] F. Molkenthin, K. V. Notay, C.-Y. Li, R. Hinkelmann, and L. Stadler. Model Integration and Coupling in a Hydroinformatics System. In *European Group for Intelligent Computing in Engineering EG-ICE Workshop*, Volume 01, pp. 218–225, Berlin, Germany, 2009. Shaker Verlag GmbH, Germany.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In retrospect, the development of human civilizations in the human history is based on the acquiring of natural resources and the battle with natural disasters. According to the concept of cause and effect in causality, this has been a dilemma in the development of human history. Natural resources are limited and the more resources are endlessly acquired for human purposes, the higher risk nature will revenge. The problem of water resources is one example. In terms of the problem of water scarcity, only three percent of the water body in the world is fresh water, and the rest of the 97% is salt water. Among this three-percent fresh water, around two-thirds are in the form of ice, e.g. glaciers, and the rest one-third is made up of groundwater and surface water. In the end, only about 0.0002% of total water (about 2120 km^3) is in rivers and about 0.007% of total water (about 91000 km^3) is in freshwater lakes as fresh surface water for the consumption of the world population [Gleick, 1993]. In addition to the problem of water scarcity, some other problems such as water pollution and hydrological disasters, especially those caused by increasing extreme events, e.g. floods, are also examples of how human beings strive for the balance between their own development of civilizations and nature.

Civil engineering, as its name suggests, is a discipline which involves the development of civilizations by engineering activities. It does not only contain the design, construction, operation and maintenance of artificial

infrastructures, but also is an art of seeking the balance between human needs and natural resources. It holds characteristics of both natural science and engineering. As science it involves the awareness and the understanding of the principles hidden behind problems, e.g. physics, mathematics, statistics, chemistry, ecology, economics, etc.; as engineering it includes the consideration and analysis of budget, time, efficiency, optimization techniques, technology, etc. Independent from which aspect of looking at civil engineering, its tasks nowadays all greatly involve dealing with a mass of data, especially by measurements, calculations and computations. Hence, how to calculate and compute accurately and efficiently has been one of the most important topics in civil engineering. A revolutionary invention tremendously improved the process of these tasks — the automatic, programmable and fully operational computer, Z3, was built by Konrad Zuse in Berlin in 1941 [Zuse, 1993]. Since then, the way and the process of dealing with engineering tasks entered a new era.

1.1 Hydroinformatics

Civil engineering encompasses a wide range of sub-disciplines, such as structural engineering, transportation engineering, hydraulic engineering, and so forth. Hydroinformatics is regarded as a sub-discipline of civil engineering which emerges from the field of computational hydraulics [Gautam, 2000] and supports hydro science and engineering together with related environmental issues by the use of modern Information and Communication Technology (ICT) [Molkenthin, 2000]. Since the beginning of the 90s, many experts have tried to define the term “hydroinformatics” [Abbott, 1991, 1994; Molkenthin, 2000; Price et al., 1998], and with the rapid development of ICT, the definition of the term “hydroinformatics” also evolved with time.

One of the first appearance of the term “hydroinformatics” appeared in the definition of a hydroinformatics system by M. B. Abbott [Abbott, 1991] as:

> A hydroinformatics system is the bringing together of compu-

> tational hydraulic modelling and information systems, including knowledge-based systems and artificial intelligence. It is an electronic knowledge encapsulator that models part of the real world.

Later, another adapted definition of hydroinformatics was given by F. Molkenthin [Molkenthin, 2000] as:

> Hydroinformatics is a basic discipline of hydroengineering. It supports the sustainable development of the aquatic environment by the use of computers and nets. In its core hydroinformatics concerns the modelling of information related to hydroengineering as well as ICT supported distributed project platforms and working processes in engineering for design, construction, management, consultation and administration.

While observing the development of the definition of hydroinformatics, the evolution path of hydroinformatics can be noticed. At the beginning, the elements of Computer Intelligence (CI) was considered as another important jigsaw puzzle of hydroinformatics in addition to traditional computational hydraulics. Afterwards, as the development of ICT continues, especially in the pervasiveness of the Internet and World Wide Web (WWW), the precision of digital data collectors, the huge storage capacity, etc., the discipline of hydroinformatics started to confront the benefits and issues brought by such improvement — mass data, which also corresponds to the term “Big Data” in the era of Internet [Snijders et al., 2012]. At this time, it deals no more equation-solving alone but also other topics, such as data preprocessing, storage, retrieval, management, etc. The computer itself is no more mere a number-crunching machine for simulation and analysis but also a tool which supported hydro science and engineering in the tasks of collaborative communication, design, construction, management, consultation and administration. Based on these definitions, hydroinformatics is a continuously evolving discipline based on four basic pillars: engineering science, mathematics, physics, and ICT [Molkenthin, 2007], as shown in Fig. 1.1.

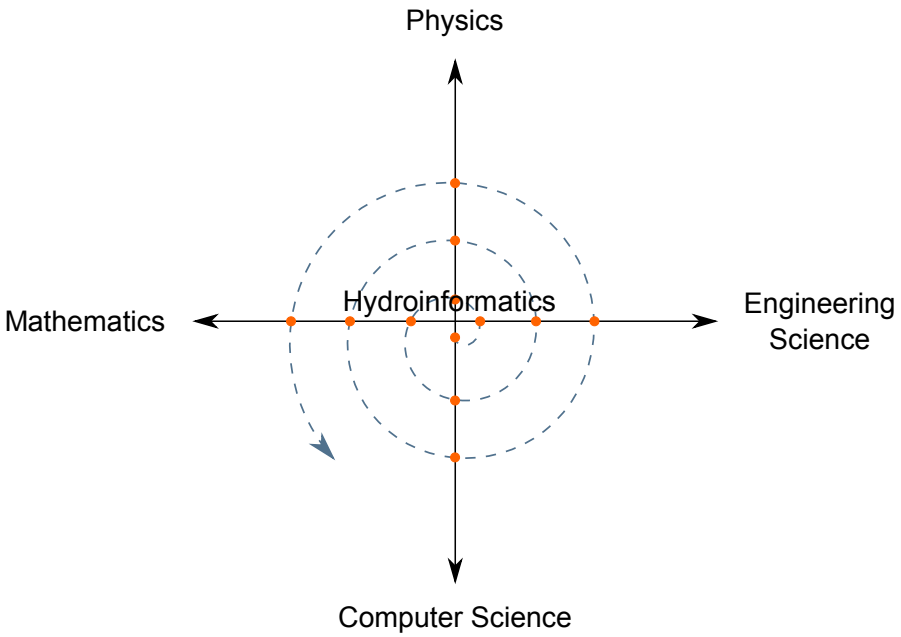


Figure 1.1: Evolution of hydroinformatics (after [Molkenthin, 2007])

Therefore, the subjects of hydroinformatics contain the fields of data acquisition, data analysis, information management, system optimization, CI, software development, etc., in addition to numerical simulation based on the development and the level of maturity of these four basic pillars. Here is a list of aims and scope of the current Journal of Hydroinformatics [IWA Publishing] which also indicates the areas the community of hydroinformatics is targeting at:

- Physically-based simulation modeling
- Numerical methods
- Data-driven modelling and management
- Artificial neural networks
- Evolutionary methods
- Cellular automata
- Modeling systems

- Geographic Information Systems (GISs) and virtual imaging
- Ecology and water quality modeling
- Environmental impact assessment
- Knowledge engineering and management
- Socio-economic framework
- Intelligent decision support, negotiation and management
- Education and training
- Internet-based applications
- Optimization and control
- Risk analysis, fuzzy logic and management of uncertainty
- Tools, environments and languages

Although this list will adapt itself to correspond the development of hydroinformatics with time, it shows the interdisciplinary coverage of itself. Moreover, it also demonstrates that it covers not only the traditional engineering-oriented computation, but also considers the issues in social science, such as the impact of water scarcity mentioned at the beginning.

1.2 Motivation

Based on the definition mentioned above, the discipline of hydroinformatics can be regarded as a amalgamation of ICT and traditional water-related disciplines such as hydrology, hydraulic engineering, hydrogeology, water supply, etc., targeting at solving hydro science and engineering problems.

Due to the pressing scientific and engineering demand, there have been a variety of studies, researches, and software implementations targeting different interests in the discipline of hydroinformatics. Some mainly focus on numerical solutions, such as MIKE by Danish Hydraulic Institute

(DHI) [DHI], ISIS by CH2M HILL [CH2M HILL], Deltares systems by Deltares [Deltares], Kalypso by Björnsen Consulting Engineers (BCE) and the Institute of River and Coastal Engineering at Hamburg University of Technology [BCE], open TELEMAC-MASCARET by a consortium of Artelia, Bundesanstalt für Wasserbau (BAW), Centre d'Etudes Techniques Maritimes et Fluviales (CETMEF), Daresbury Laboratory, R&D group of Electricité de France (EDF), and HR Hallingford [open TELEMAC-MASCARET], HEC-RAS by US Army Corps of Engineers [HEC-RAS], MODFLOW by United States Geological Survey (USGS) [MODFLOW], OpenGeoSys by Helmholtz-Zentrum für Umweltforschung (UFZ) [OpenGeoSys], etc.; some specialize in GISs, including ArcGIS by Esri [Esri], GRASS GIS by the GRASS Development Team [GRASS Development Team, 2012], QGIS by the QGIS Development Team [QGIS Development Team, 2014], SAGA by the SAGA Development Team [SAGA], etc.; some aim at model coupling, for instance, Open Modeling Interface (OpenMI) by OpenMI Association [Gregersen et al., 2007], etc.; some make efforts in geospatial data infrastructure and sharing, such as, INSPIRE [INSPIRE]; some target at system-wise management of data and metadata, like WISKI by KISTERS [KISTERS], CUAHSI-HIS by CUAHSI [CUAHSI], etc.; some address the standardization of data access, for example, Geography Markup Language (GML) [Open Geospatial Consortium], SensorML [Open Geospatial Consortium], WaterML2 [WaterML2]; some direct towards visualization, like data Processing, Analysis and Visualization (datPAV) [datPAV] as an example.

Among these outstanding studies, researches and implementations, one of the key challenges still exists, especially in interdisciplinary research and engineering projects — filling the gap between available mass raw data and simulation tools. With the help of the rapid improvement of ICT and sensor technology, the quantity and complicatedness of acquired data from field measurements, numerical simulations and laboratory experiments in different time and space scales also grow with the increase of complexity. Nevertheless, the storage is no more the major issue among researchers and engineers. Instead, turning the mass raw data into useful information for the purposes of problem analysis and simulation tasks becomes a vexing

problem.

In this PhD research work, a semi-automatic, data-driven based time series scenario composition framework based on the existing implementation of Turtle [Molkenthin et al., 2009] and its software prototype are presented. This framework intends to assist answering the impacts of user-specified what-if scenarios by generating corresponding Initial Conditions (ICs), Boundary Conditions (BCs) and Parameter Sets (PSs) as time series data sets from the collected data, such as field measurements, laboratory experiments and simulation results of simulation tools. In addition, this framework also keeps track of the metadata of the data, for instance, the operations which are applied on the data. In this respect, it describes how the data are processed and guarantees the reproducibility of the entire workflow. Hence, the framework can be regarded as a data management/generation tool generating inputs for simulation tools.

1.3 Problem Identification and Objective

Gap between Mass Data and Simulation Inputs

While recalling the workflow of ritual research or engineering activities regarding simulation tasks, they typically involve five parts in an iterative manner:

- data acquisition
- pre-processing
- model simulation
- post-processing
- decision making

Although with such a number of contributions in the discipline of hydroinformatics as mentioned in Section 1.2, researchers and engineers still suffer from the lack of information, especially if they want to research

the impacts from certain what-if scenarios for further decision makings. Some examples are the National Groundwater Modelling System (NGMS) [Whiteman et al., 2012] and the Intergovernmental Panel on Climate Change (IPCC) Data Distribution Center [IPCC], which collect different predefined scenarios for investigations of groundwater problems using MODFLOW and climate change issues respectively. Other examples are described in different studies, as in [Kalyanapu et al., 2012; Nuswantoro et al., 2014], which use Monte Carlo based approaches to generate reasonable inputs for further deterministic models to assess flood risk. Under the status of modern ICT, the capacity of storage can easily reach the unit of terabyte, and researchers and engineers are frequently buried under large numbers of raw data. It leads to the core of this research work — turning the mass raw data into needed input information for simulation tasks.

Proposed Solution

Therefore, a general framework of time series composition [Li and Molkenthin, 2014; Molkenthin et al., 2014] and its software prototype are presented here. It targets at facilitating the process of hydro science and engineering simulation tasks by providing time series data sets as ICs, BCs and PSs based on user-specified what-if scenarios. Unlike weather generators [Racsko et al., 1991; Richardson, 1981; Wilks and Wilby, 1999], which are widely used as downscaling techniques to obtain regional-scale information from the outputs of Global Climate Models (GCMs), “features” of time series data sets are extracted as LEGO® bricks with the help of a set of tools and Application Programming Interfaces (APIs) for possible extensions of techniques in the prototype. Researchers and engineers can further compose the scenarios of interest based on the extracted feature bricks and the information of possible combinations of bricks proposed by the framework. Afterward, these user-specified scenarios can be converted into time series data sets as inputs, such as ICs, BCs and PSs for further simulation tasks.

The framework is, in principle, data-driven based and the data are

nonspatial time series[1]. In addition, the core of this framework is mainly based on Time Series Knowledge Mining (TSKM), which is a temporal reasoning framework proposed by [Mörchen, 2006a,b; Mörchen et al., 2005], and fuzzy logic. It contains a sequence of manipulations and operations on data, and users have full control on both the results of feature extraction and scenario composition. Hence, it is a semi-automatic process and users must have enough knowledge in the domain of interest to decide the operations in different steps and further to evaluate the final results. Furthermore, it also offers an opportunity to uncover and identify unknown physical phenomena as well as the tracking of every single manipulation and operation applied. The framework itself can also be regarded as a toolbox for both scientific and engineering purposes. To achieve this, it is designed to provide proper APIs for further necessary extensions together with a basic set of algorithms and tools. In this way, it can also serve as an add-on to other hydroinformatics tools.

1.4 Research Approaches and Methods

This framework is mainly based on TSKM and fuzzy logic, and the entire framework consists four modules, as shown in Fig. 1.2, corresponding to the pre-processing activity as mentioned in Section 1.3:

- data pre-processing
- event identification
- process identification
- scenario composition

The module of data pre-processing can be regarded as a decisive step. This is because the raw data are usually incomplete and faulty, and the quality of data is essential for any data-driven based technique [Pyle, 1999; Witten and Frank, 2005], upon which the framework itself grounds. Without

[1]The extension to possible spatial data applications will be further discussed in Chapter 8.

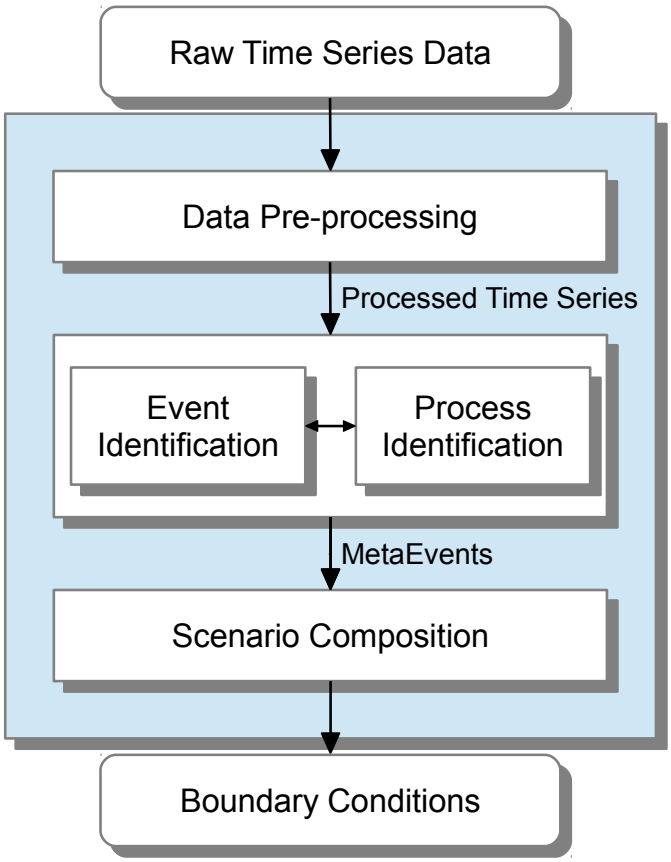


Figure 1.2: Illustration of four modules in the scenario composition framework

proper handling of data pre-processing, it is easily prone to a “Garbage In, Garbage Out” situation. The purpose of this module is to turn the mass raw data into necessary time series data sets, and the methods can be vastly domain- and data-specific, e.g. gaps filling, trends removing, domain transform, dimension reduction, etc.

The event identification is based on TSKM and transforms the time series data sets into pieces of human-readable and human-interpretable information. These pieces of information can be further adapted as features of the entire time series data sets. The event identification contains a sequence of steps and methods and users have full control over methods of choice and results. In addition, expert knowledge and experience can also be brought into the steps of identifying events if required. Besides, it

is a semi-automatic manner and users can gain a generalized and human-understandable descriptive overview of how phenomena develop with the help of TSKM.

Due to the results derived from event identification being quantitative and loosely connected, the process identification provides a way to identify and describe the physical relationships among variables in a stronger manner. The process identifcation is built upon the Mamdani-type fuzzy inference system together with Multivariate Adaptive Regression Splines (MARS) [Friedman, 1991], which serves as a second mapping function to ensure better descriptions of the phenomena of interest.

The results from both event identification and process identification will be aggregated and formed as "MetaEvent" for the purpose of scenario composition. The MetaEvent serves as the most basic unit of information in time series, which describes the characteristics of each extracted feature, in scenario composition. Their realizations, MetaEventEntities, can be seen as LEGO® bricks, as aforesaid, and users can compose the scenarios of interest by assembling these "bricks". In the prototype of this framework, it provides an user interface, which offers users additional information of each Event, such as duration, expected values, next possible "brick", etc., to assist users composing scenarios. Later, these composed scenarios can be converted into a set of time series data with user-specified properties, e.g. time step (Δt), for further needs, for example, simulation tasks, additional post-processing, etc.

The prototype implementation of this framework is based on R [R Core Team, 2012] and Java platforms [Java], and the data exchange between these two software environments is through the R package `rJava` [Urbanek, 2011], which uses Java Native Interface (JNI) [Liang, 1999] as a communication channel.

1.5 Thesis Structure

Chapter 2 provides an overview of state of the art of hydroinformatics systems used in either scientific or professional fields. This overview focuses on an introduction of different hydroinformatics systems, data management in hydroinformatics, and time series data mining in hydroinformatics.

Chapter 3 introduces the fundamental background knowledge applied in the framework. It contains the descriptions of Time Series Knowledge Representation (TSKR), TSKM, fuzzy logic, MARS and suffix tree.

Chapter 4 elaborates the terminology and theories used in this framework of scenario composition. It delineates the structure of the framework, how TSKM is applied in event identification, how fuzzy logic and MARS are utilized in process identification, and how scenarios can be composed in scenario composition.

Chapter 5 presents the implementation of the software prototype of this scenario composition framework. It contains the used software technologies, the integration with the Hydroinformatics system Turtle, and the Graphical User Interface (GUI). Finally, it also includes an overview discussion of this implementation.

Chapter 6 offers several application examples, including synthetic academic and real-measured application examples. It covers detailed descriptions of these application examples and the results based on the implemented prototype. In this chapter, it shows the possibilities this framework can achieve.

Chapter 7 discusses the results from chapter 6, and gives an evaluation based on criteria offered. Recommendations based on the discussions are also presented here.

Chapter 8 summarizes the conclusions of this research work. In addition, it also gives an assessment of this framework, showing its capabilities and possible recommendations, and finally gives an outlook for further studies.

At the end of this thesis, a glossary and a list of acronyms are also affixed, apart from the bibliography and appendices for the clarification of the terms mentioned in the context.

Chapter 2

Hydroinformatics Systems

2.1 Definition

One of the first definitions of the term “hydroinformatics system” is defined in 1991 by M. B. Abbott [Abbott, 1991] as mentioned in Section 1.1:

> A hydroinformatics system is the bringing together of computational hydraulic modelling and information systems, including knowledge-based systems and artificial intelligence. It is an electronic knowledge encapsulator that models part of the real world.

However, with the evolution of the definition of hydroinformatics, the definition of the term “hydroinformatics system” has also to be adapted. Based on the definition in Merriam-Webster online dictionary (2013) [“system”], the definition of a system is defined as:

> a regularly interacting or interdependent group of items forming a unified whole

Hence, a hydroinformatics system, based on this definition and the definition of the term “hydroinformatics” stated in Section 1.1, is defined here as:

> *a set of modularized and firmly integrated Information and Communication Technology (ICT) tools together with the application of knowledge in mathematics, physics, engineering, etc., targeting at systematically investigating and solving water-related problems with a complete concept*

2.2 Components and Systems

Components

According to the definition of the term “hydroinformatics system” given earlier, a hydroinformatics system should contain functionalities such as pre-/post-processing, data management, data exchange and application programming interfaces, simulation, analysis, visualization, network communication, etc. A basic illustration of components in a hydroinformatics system based on a general workflow is shown in Fig. 2.1. There, a hydroinformatics system is decomposed into three different components based on a general engineering workflow and wrapped by the User Interface (UI):

- Leading Component: Preparations for the core component, such as grid generator, data editor, data pre-processing, data management, etc.
- Core Component: Processing the data/information from the leading component to needed information, for instance, numerical analysis, statistical analysis, machine learning, etc.
- Trailing Component: Representations and summaries of the information derived from the core component, for example, visualizer, report generator, etc.

Fig. 2.1 also indicates where this research work is located: between the leading and the core components which turn the raw data into needed information for further simulation tasks. It first extracts features from the existing time series data and modellers can further compose scenarios of

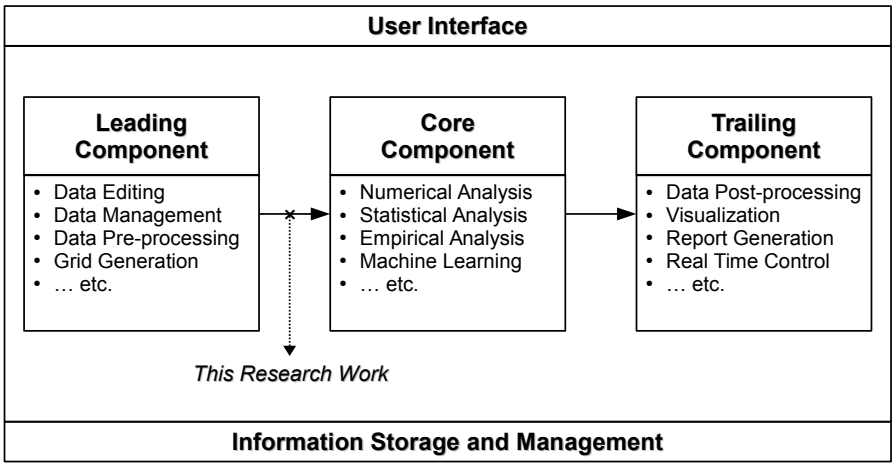


Figure 2.1: Components of a hydroinformatics system and the current research work's location

interest and generate corresponding data based on these extracted features, unlike other methods, such as:

- weather generators [Racsko et al., 1991; Richardson, 1981; Wilks and Wilby, 1999] downscaling results from Global Climate Models (GCMs)
- Monte Carlo based approaches [Kalyanapu et al., 2012; Nuswantoro et al., 2014] stochastically generating desired results
- National Groundwater Modelling System (NGMS) [Whiteman et al., 2012] and Intergovernmental Panel on Climate Change (IPCC) Data Distribution Center [IPCC] with predefined scenarios

In addition, the current prototype implementation works as a stand-alone application as well as offers Application Programming Interfaces (APIs) for further integration with other hydroinformatics tools or systems.

Systems

Several reviews regarding different 2D hydroinformatics systems can be found in [Néelz and Pender, 2009, 2013]. Here, a comparison of several well-known scientific or engineering hydroinformatics tools based on the

dimensions, with or without Graphical User Interface (GUI), major application fields, type of engine, etc., with respect to these components can be seen in Table 2.1. Although FluidEarth [FluidEarth] can not be regarded as a "system" based on those three components, it offers a software environment to couple different simulation tools. Thus, the entire coupled tools together with itself can be regarded as a system. SWIM [Krysanova et al., 1998] and TELEMAC [open TELEMAC-MASCARET] on the other hand, mainly offer the core engine component, and the rest has to be supplemented by third party applications. Some applications, such as HEC-RAS [US Army Corps of Engineers] and ISIS 1D [CH2M HILL], are restricted mainly for single purpose and are difficult to be considered as a system. However, if they are integrated with other series of applications, such as ISIS [CH2M HILL], the entire application can be regarded as a system.

There are several different criteria to categorize the systems, such as the simulation method, the theory of analysis, the application field, and so on. Independent of which criterion is used for the categorization, these systems usually serve as Decision Support Systems (DSSs) operated by modelers with the knowledge of their own fields to support further decision makings on environmental and socioeconomic issues for management and operational tasks. There are numerous hydroinformatics tools or systems developed for water-related scientific and engineering environments, as illustrated in Table 2.1. Apart from it, the descriptions for some selected hydroinformatics tools or systems based on license type (open source/proprietary), application field, etc./space with different objectives, approaches, etc. are as follows:

- FluidEarth [FluidEarth]: An open source project initialized by HR Wallingford provides a software environment to couple different simulation tools through the help of OpenMI [Gregersen et al., 2007]. In addition to a Software Development Kit (SDK) for developers to conform to the requirements for their simulation tools, it also provides modelers a GUI tool, pipistrelle, to link different simulation components, set up a model and execute it in a visual programming fashion. Although it can not be regarded as a system as mentioned earlier, with the help of

Table 2.1: Comparison of typical hydroinformatics tools

Tools	Dimension(s)	GUI	Major App. Field(s)	Data Editor	Major Engine	Visualization	Extension(s)
FluidEarth[a]	-	Yes	Flood modeling, river modeling, surface water modeling	-	-	-	OpenMI, SDK
HEC-RAS	1D	Yes	River modeling	Yes	Numerical	Yes	OpenMI, GIS, etc.
ISIS 1D	1D	Yes	River modeling	Yes	Numerical	Yes	OpenMI, GIS, etc.
Kalypso	1D/2D	Yes	Flood modeling, surface water modeling	Yes	Numerical	Yes	GIS
MIKE URBAN	1D/2D	Yes	Urban water modeling	Yes	Numerical	Yes	OpenMI, GIS, etc.
SOBEK	1D/2D	Yes	Flood modeling, drainage systems, sewer design, water quality	Yes	Numerical	Yes	OpenMI, GIS, etc.
SWAP-GIS	2D	Yes	Groundwater contamination	Yes	Numerical	Yes	GIS
SWIM[b]	2D	No	Climate and land use change impacts	No	GIS	No	GIS
TELEMAC[b]	2D/3D	No	Flood modeling, surface water modeling	No	Numerical	No	NA

[a] Comparing to other tools listed here, it is a software environment coupling different simulation tools to solve problems instead of solving problems by its own.

[b] It only offers the core/engine part, and the rest has to be done through 3rd party applications for visualization, data editing, etc.

this software environment, a customized hydroinformatics system can be formed.

- Kalypso [BCE]: A Java-based open source project developed by Björnsen Consulting Engineers (BCE) and the Department for River and Coastal Engineering at Hamburg University of Technology. It provides modelers a set of modularized tools to investigate surface water problems towards flooding analysis, and further towards report generation and risk management. It conforms to several open standards, such as Open Geospatial Consortium (OCG) standards, OCG Web Processing Service (WPS) standards, Open Document Format for Office Applications (ODF) standards, and so forth. It also offers a complete tool set for pre-processing, simulation, and post-processing.

- MIKE URBAN: It belongs to the MIKE family developed by Danish Hydraulic Institute (DHI) [DHI], and it targets at urban water problems, such as sewers, water distribution, and so on. It has its own simulation models for rainfall-runoff, pipe flow, etc., tools for automatic calibration, weir control, etc., and integrates with ArcGIS by ESRI [Esri].

- SWAP-GIS [SWAP-GIS]: A complete set of SWAP-GIS was firstly developed by the Environmental Resources Research Institute at Penn State University and was originally targeted at a project of potential groundwater contamination problems from Pennsylvania Department of Environmental Protection. One of the features is that it provides a batched operation of analysis of data sets. In the end, the analyzed results can be viewed visually by a ArcView-based tool and a concise report can be also generated within the system.

Although the definition of the term “hydroinformatics system” is given here and several examples are illustrated, the boundaries among different systems are getting less distinct with the rapid development of ICT, especially if the systems are targeting at the problem solving and decision making as complete DSSs in the same discipline as their final goal. The following

sections will be centered on data management and time series data mining in the field of hydroinformatics due to the focus of this research work.

2.3 Data Management in Hydroinformatics

Data are qualitative or quantitative descriptions of information and can be stored in digital or nondigital carriers. In the fields of natural science and engineering, these data are mostly quantitative because of the need for quantitative description in physical laws and number crunching. Besides, they can come from different resources with different temporal and spatial scales, such as field measurements, computer simulations, laboratory experiments, etc. According to different recording approaches, these data can be stored in nondigital carriers, such as paper notebooks, analog recorders, etc., or digital carriers, such as hard disks, Solid-State Drives (SSDs), Network-Attached Storage (NAS) devices, Redundant Array of Independent Disks (RAID) devices, etc. Whereas, with the rapid improvement of ICT, sensor technologies and data measuring devices, e.g. digital data collectors, remote sensing devices, together with the huge requirement of working with computers, records are now stored in the digital carriers and historical data are also converted from analog carriers to digital ones.

Data Problems

In many hydro science and engineering projects, numerous and various data have been collected for further calculation, e.g. numerical simulation, or analysis. Furthermore, the data types, scales, duration, and so on, have to be carefully decided before the collection activities are carried out. These data usually serve as Initial Conditions (ICs), Boundary Conditions (BCs), calibration references, validation references, etc., for different purposes. Besides, these data, especially raw data, collected from different measurement devices, simulation programs, and so forth often suffer from many problems of:

- unclear/incomplete information[1]: This problem contains different aspects, such as unrecognizable handwriting, unclear description, etc. and mainly happens when the data are collected manually without the help of modern data logging technology. It can be regarded as a result of the history of data collection technology. In such cases, each record has to be written by hand on paper. Depending on the circumstances of how the record is kept, the final data might be undecipherable for others in need. In addition, the metadata, such as unit, time zone, etc., might be also missing due to carelessness or ignorance. This also affects the data maintenance and the further usage of the data.

- errors: Any measurement contains errors and these errors are not avoidable but can only be controlled. In addition, these errors cause the measurements to be inaccurate or imprecise. These errors can come from misreading of values by people or wrong calibration of devices or inappropriate measuring devices or instability of voltage from low power batteries, to many other different possible sources.

- data formats: The data format defines how information is stored in a digital file. Due to impact of the more and more computerized working environment, the information exchange among different specifications of data formats becomes a must in our routine work. Moreover, the exchange of information, e.g. through conversions of data formats, often goes along with the loss of information. This need indicates the importance of the standardization of data formats.

- inconsistent description: In addition to the requirement of converting among heterogeneous file formats as mentioned earlier, extracting information from the homogeneous file formats can be also an issue. This is usually because the data are commonly collected from different authorities and devices. The content of the collected data can have different formats, such as:

[1] Information is data together with the semantics derived from data themselves which describes certain correlations, such as patterns, associations, relationships, etc

  - date and time: The format of date and time is one of the most common encountered issues while extracting data from a file. Depending on the standards applied in different countries and languages, the date “01 May 2001” can be denoted in the form of “2001-05-01”, “05-01-2001”, “01-05-2001”, etc. Moreover, these descriptions usually come without proper descriptions of information, for instance, the corresponding time zones, summer/winter time, etc. Without this information, problems, such as records with the same time but in different time zones, emerge.
  - column order and number: Since the data are collected from different sources, the format of the content may differ. A table with headers is one of the most common used content formats to store and display data. It might happen that two files from different sources storing the same activity have different column orders or column numbers. This increases the difficulty to automatize the working process.
  - units: A unit is crucial for any measurement. However, different devices may have different default unit settings and standards for the same measurement. The conversions in length (mm, cm, m and km), in temperature (°C and °F), and in time and date (sec., min., hr., day, month and year) are common tasks. However, the conversions between different dates have to be specially taken into consideration if different time zones are encountered. For example, the same date values but within different time zones will cause great impacts on large scale models if they are not correctly specified. Moreover, some issues, such as accuracy, resolution, etc. also have to be taken into consideration.

- gaps: Independent of how data are collected, gaps are difficult to avoid, especially for field measurements. Several different situations might happen, for instance:

  - If the data are collected by manpower, the gaps might happen

because of the weather conditions or the laziness of collectors;

- If the data collection devices are installed in the open fields, the devices are highly vulnerable to wild animals and natural conditions;
- If the data are collected by electronic data loggers, these loggers stop collecting activities when the electricity goes off;
- Different devices may have different intervals while collecting data, and these differences have to be harmonized before applications' running;
- Different devices have different mechanisms, e.g. event-triggered mechanisms in tipping bucket rain gauge, regular measurement in thermometer, etc., for data collection.

In addition, it is inevitable to avoid the loss of information during any operation on data as mentioned earlier. Hence, it is a challenge in a project to accurately and efficiently extract necessary information from the collected data.

One typical example is the interdisciplinary research project "Großhang — Natural Slope" [Hinkelmann et al.; Molkenthin et al., 2014; Zehe and Hinkelmann, 2013] which deals with "Coupling of Flow and Deformation Processes for Modelling the Movement of Natural Slopes". This project is divided into 5+1 different sub-projects which deal with their own specialities, and these sub-projects are listed below:

- Sub-project 1: Hydrology and applied seismics
- Sub-project 2: Subsurface hydraulics
- Sub-project 3: Continuum mechanics
- Sub-project 4: Technical scale experiments
- Sub-project 5: Geophysics
- Central sub-project: Project and information management

More than five years of data in different space and time scales were collected. These data come from different field measurements, simulation results, and laboratory experiments, including soil moisture, discharge, wind speed, wind direction, snow height, topography, slope deformation, etc. One key factor to the success of the project was to effectively handle such heterogeneous data in terms of exchange, retrieval, conversion, etc. among different sub-projects.

Different Solutions

To resolve the issue of heterogeneous descriptions of data among different files, the use of metadata can be applied. The term “metadata” means “data about data”. In a practical sense, the metadata of the data describes where, when, what, who, why and how the data are recorded and some standards, such as ISO 19115 [ISO 19115, 2003] for geographic data, which defines how the geographical information is described. Metadata ensure the accessibility of the information of the data in the future. The description of data and their metadata in a file are represented in the format of Extensible Markup Language (XML). With the help of XML and its schema, they ensure the interpretability of data. For instance, the project Earth Observing System (EOS) Clearing House (ECHO) by National Aeronautics and Space Administration (NASA) now adopts the ISO 19115 standard for its metadata description [Earthdata Collaboration Environment (ECE)]. In addition, the INSPIRE directive [INSPIRE], as an European Spatial Data Infrastructure (SDI), is also devoted to establishing an European Union (EU)-wide infrastructure for sharing spatial information in support of decision making across boundaries.

In addition to traditional archives which accumulate physical records, such as documents, digital information is usually stored either in files or in databases depending on the requirements. Several software systems are designed and implemented to manage the stored data. For the file-based data storage, some solutions, such as Document Management Systems (DMSs), provide not only the basic file storage, but also other functionalities, such as

metadata description, versioning, indexing, collaboration, publishing, etc. A concept of applying one of such systems, WWW-based document management system — DCMS [Brüggemann et al., 2001] for flood management can be seen in [Holz et al., 2006]. There, the DCMS system, with web-based server components and a front-end, manages the metadata of file-based media information from different heterogeneous resources over the Internet. It provides a flexibility to define task-related attributes by semantic markup language and a rapid means to retrieve, access, exchange, and share information through XML.

Similarly, Database Management Systems (DBMSs) are usually applied to manage the information stored in databases. DBMSs provide different operations to manage the stored data, for instance: creation, update, and deletion. There exist several general-purpose DBMSs, such as MySQL [MySQL], SQLite [SQLite], Microsoft SQL Server [Microsoft SQL Server], Oracle database [Oracle Database], PostgreSQL [PostgreSQL], and so on, for this purpose. Among these DBMSs, two major categories are:

- Relational Database Management Systems (RDBMSs): MySQL, SQLite, Microsoft SQL Server, etc.
- Object-Relational Database Management Systems (ORDBMSs): Oracle database, PostgreSQL, etc.

Comparing to RDBMSs, ORDBMSs offer direct support of objects, classes and inheritance in their schemas and query languages. For instance, PostgreSQL supports not only the common data types, such as numeric, string, date/time, etc., it supports a wider range of data types, including geometric, network address, JavaScript Object Notation (JSON), etc. The geometric data types can represent two-dimensional spatial objects and some basic data types are already available by default, e.g. `point`, `line`, `box`, `polygon`, `circle`, etc. With such supports, ORDBMSs can more efficiently operate and manage the hydrogeological data in the database.

Several applications are available, such as:

- The marine environmental database, Meeresumwelt-Datenbank (MUDAB) [MUDAB], storing data of 700 different variables collected

for the protection of the North Sea and the Baltic Sea uses Oracle database as its back-end.

- PostGIS [PostGIS], as an extension for PostgreSQL, offers an additional support for dealing with geographic objects and can be made use of as a database back-end in many well-known Geographic Information Systems (GISs), such as GRASS GIS [GRASS Development Team, 2012], QGIS GIS [QGIS Development Team, 2014], and ArcGIS [Esri].

Data have to be conveyed, either being exchanged among different software programs for various number-crunching purposes or being delivered to human beings as information. Several different standards are established for the data exchange among different software programs, such as WaterML2 [WaterML2] defining file exchange format based on Geography Markup Language (GML) as standards to represent hydrological time series structures, and Open Modeling Interface (OpenMI) [Gregersen et al., 2007] providing an interface for run-time data exchange. Unlike the data exchange among different software programs, the "data exchange" between software programs and human beings is to deliver the information hidden behind the data themselves. There are two different types of data representations — static and dynamic. The media for static data representation are usually reports, books, maps, static web pages, etc., and these can be generated by different tools, like a word processor. However, once these documents are produced, the data which the information derives from are often omitted and the processes of reproduction and update become laborious. Tools for literate programming and reproducible research, such as Sweave [Leisch, 2002] and Org-mode [Schulte et al., 2012], can be used to avoid the missing links to the original data and increase the efficiency of reproduction and update. Dissimilar to static data representations, the dynamic way provides additional functionalities to access the original data or their metadata, and interactive web pages are common media for this purpose. Apart from the academic applications, as in [Brüggemann and Holz, 2000; Molkenthin, 2000], some operational public services, such as Flusshydrologische Software (FLYS) [BfG] and Elektronischer Wasserstraßen-Informationsservice

(ELWIS) [ELWIS] providing the information of water levels and waterways of German major rivers, are available online. Besides, several general key principles and issues regarding interactive graphical representation of data, like over-plotting, are addressed in [Theus and Urbanek, 2008].

2.4 Time Series Data Mining in Hydroinformatics

As stated in Section 1.2, with the rapid improvement of ICT and the sensor technology, the quality and the quantity of the collected data, overall are better and richer comparing to those collected decades ago. Due to these improvements in data acquisition as well as the data analyzing power, they encourage scientists and engineers to uncover and solve problems which were difficult or even impossible before.

Data Mining

Data mining is one of many approaches which utilize the advantage of these improvements. It refers to the process of searching through and analyzing large data sets in order to acquire useful or meaningful information out of them and it is considered as a step of Knowledge Discovery in Databases (KDD) [Fayyad et al., 1996]. The term KDD was first coined at the first KDD workshop in 1989, and data mining is a means of KDD to reach its goal — knowledge discovery [Fayyad et al., 1996].

Till now, three different terms, data, information, and knowledge, have been brought up. In addition to the definitions of data and information described in Section 2.3, the term knowledge in the scope of this research work means something human beings have learned or acquired through the exposure of data or information, and it can be applied repeatedly in an empirical or a theoretical fashion.

The coverage of application fields in data mining is large, and it comprises business, surveillance, medicine, biological engineering, etc. One of the most

famous data mining examples is that Walmart, an American retailer, found out the statistically significant positive correlation between the purchases of beer and the purchases of diapers on male customers while analyzing customers' shopping habits. After that, Walmart decided to place diaper products next to beers, and it led to the significance growth in the sales of both.

There are many different techniques or approaches in the fields of statistics, Machine Learning (ML), and Artificial Intelligence (AI) used in data mining, such as clustering, classification, and regression to help "digging out" information from mass data. Some common information data mining seeks out includes:

- anomaly
- category
- cluster
- association
- sequential pattern

Besides, some standards regarding data mining in software technology, e.g. Predictive Model Markup Language (PMML), are established and some software applications, e.g. MATLAB® [MATLAB], STATISTICA® [STATISTICA], IBM SPSS software [SPSS], R [R Core Team, 2012], and Waikato Environment for Knowledge Analysis (WEKA) [Hall et al., 2009], are available to assist the activities of data mining.

In the field of hydroinformatics, data collected for data mining are mostly either time series or spatial variables describing physical behavior, material properties, and the like. However, the data collected for the usage in hydroinformatics, especially in the field measurements, comparing to those in other disciplines, are suffering problems of poor representation of real world, systematic measurement errors, and the like, for instance [Spate et al., 2003]:

- The definition of “daily” varies from variable to variable depending on criteria. For example, the daily rainfall records are usually collected from 9 a.m. to 9 a.m., yet the streamflow records are usually collected at 12 a.m.
- The records measured from rainfall gauges may suffer losses from evaporation, or even splashing.
- The measurement of flow velocity converted from the weir function may not be adequate to represent the whole velocity profile at the measurement point.

These collected data usually serve as ICs, BCs, Parameter Sets (PSs), and calibration and validation references for simulation purposes, in addition to the sources for mining purposes. The motives for time series and spatial data mining in hydroinformatics are usually different because of the properties of data types, although with the same ultimate objectives — problem solving and decision making. Looking into several studies in time series and spatial data mining in the field of hydroinformatics, such as in [ASCE Task Committee on Application of Artificial Neural Networks in Hydrology, 2000; Babovic, 2005; Bárdossy and Disse, 1993; Bárdossy and Duckstein, 1995; Bárdossy et al., 1995; Hall and Minns, 1999; Mennis and Guo, 2009; Miller and Han, 2009; Nayak et al., 2004], the focus of time series data mining, in general, is more toward re-representing rules of physical behavior for prediction purposes instead of the geographic knowledge discovery in spatial data mining. In addition, it is usually computationally more expensive to mine spatial data than time series data nowadays due to the richness of spatial data [Goodchild, 2007; Mennis and Guo, 2009; Miller and Han, 2009].

Time Series Data Mining

Several general studies and discussions on the topic of time series data mining can be found, for example, in [Antunes and Oliveira, 2001; Mörchen, 2006b; Ratanamahatana et al., 2005], and both types of time series data,

numeric and symbolic, are discussed. In addition, several major tasks in time series mining are considered as [Ratanamahatana et al., 2005]:

- indexing
- clustering
- classification
- prediction
- summarization
- anomaly detection
- segmentation

However, in the field of hydroinformatics, the tasks which time series mining focuses on are mainly prediction-related [ASCE Task Committee on Application of Artificial Neural Networks in Hydrology, 2000; Babovic, 2005], apart from other types of applications, such as reconstruction of missing values [Abebe et al., 2000], classification [Hall and Minns, 1999], etc., and one of the most common applications is to describe the rainfall-runoff relationship. Several different methods and approaches, independent of the unit hydrograph theory and statistical models, are proposed, such as Artificial Neural Network (ANN) [ASCE Task Committee on Application of Artificial Neural Networks in Hydrology, 2000], Genetic Programming (GP) [Babovic, 2005], fuzzy logic [Özelkan and Duckstein, 2001], and some other ML-based methods [Dimitri P. Solomatine, 2008], etc. One big advantage of these models is that these models require less extensive data comparing to other deterministic models once they are properly set up [Babovic, 2005]. It is especially usefully for those areas with less or insufficient data collections.

The major concept and applications of this research work are different from those major hydroinformatics applications mentioned earlier. Instead of focusing on representing physical behavior rules, “features” of collected data are extracted together with the information of their temporal order. With such information, the user-specified scenarios can be constructed and their

corresponding time series data can then be generated for further simulation tasks as ICs, BCs and PSs.

Chapter 3

Framework Fundamentals

3.1 Overview

In this chapter, the fundamentals applied in the framework of scenario composition are presented here. They are utilized in different modules inside the framework, and are categorized by the usage as:

- Event Identification:
  - Time Series Knowledge Representation (TSKR)
  - Time Series Knowledge Mining (TSKM)
- Process Identification:
  - Fuzzy logic
  - Multivariate Adaptive Regression Splines (MARS)
- Scenario Composition:
  - Suffix tree

3.2 Time Series Knowledge Representation (TSKR)

The TSKR is proposed by Mörchen [Mörchen, 2006a,b; Mörchen et al., 2005] describing the relation between two symbolic time intervals in replace of the widely adopted Allen's temporal relations.

A symbolic time interval, which represents an activity during a time period, is an important format in discovering temporal knowledge. It can be defined by three components: *start time*, *end time*, and *symbolic value*, as a triple [Mörchen and Ultsch, 2007]. Among these three components, the symbolic value is any description which expresses the state of the time interval. For instance, a rainfall in the range of 10 to 50 mm/hr starting at 7 pm and stopping at 8 pm can be denoted as [7 pm, 8 pm, heavy rain][1]. Furthermore, Allen's temporal relations, also known as Allen's relations, describe the relation between any two time intervals. They are widely used for unsupervised temporal knowledge mining, such as in [Cohen, 2001; Höppner, 2001; Kam and Fu, 2000].

Allen's relations to express time intervals contain 13 relations[2]: *before*, *meets*, *overlaps*, *starts*, *during*, *finishes*, and *equals* [Allen, 1983], yet Mörchen criticizes Allen's relations have three severe defects regarding: robustness, expressivity, and interpretability[3] [Mörchen, 2006a,b]. Here are the explanations:

Robustness: An example given by Mörchen as shown in Fig. 3.1 illustrates three different patterns of two time intervals *A* and *B*, which have Allen's relations of *overlaps*, *during* and *finishes*, but are actually very much alike. This is because Allen's relations need at least two or more endpoints and this will cause problems to distinguish the relations of

[1] Based on the classification in [Met Office, National Meteorological Library and Archive, 2005], the precipitation in the range of 10 to 50 mm/hr is categorized as a heavy rainfall.

[2] Each relation, except equals, contains a corresponding inverse. For instance, two time intervals *A* and *B*, and *A* takes place before *B*. In this case, the relation can be denoted as *A before B* and its inverse is *B after A*. That is the reason why Allen's relations consist of 13 relations.

[3] The definitions of these three terms can be seen in the Glossary.

two time intervals while noise, for instance, from measurements, are involved. While noise is difficult to avoid during measuring, these tiny differences caused by noise can lead to totally different Allen's relations of two time intervals as shown in Fig. 3.1.

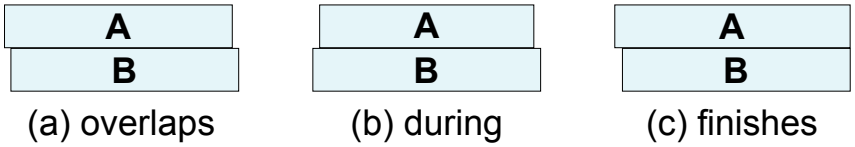


Figure 3.1: Unrobustness of Allen's relations. (after [Mörchen, 2006a])

Expressivity: Another example given by Mörchen as shown in Fig. 3.2 exemplifies the inexpressivity of Allen's relations. The examples in Fig. 3.2 (a), (b), and (c), present the same *overlaps* relation yet result in very different meanings. The Fig. 3.2 (d) furthermore gives an example of one exact pattern, which can be described by three different rules.

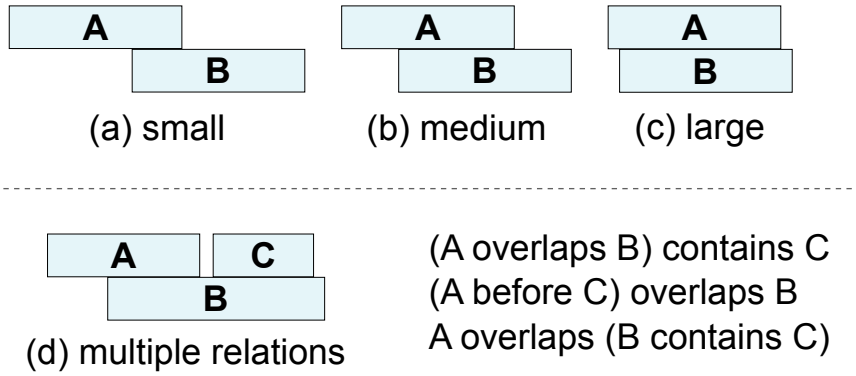


Figure 3.2: Inexpressivity of Allen's relations. (after [Mörchen, 2006a])

Interpretability: According to the example given in [Höppner, 2001], the patterns described by Allen's relations with all intervals require additional pairwise relations. Besides, the number of potential candidates

of rules increases exponentially with the number of the patterns. These cause that the relations of patterns, which reason the targeting process, described by Allen's relations are lengthy and not easy to be interpreted. This also hinders the process of temporal reasoning.

Due to these disadvantages in robustness, expressivity, and interpretability as mentioned above, Mörchen proposed another hierarchical language for temporal pattern discovery — TSKR [Mörchen, 2006a,b; Mörchen et al., 2005]. TSKR is based on a hierarchical rule language — Unification-based Temporal Grammar (UTG), proposed by [Ultsch, 2004] especially for the description of patterns in multivariate time series.

Unlike Allen's relations which define relations of random two time intervals strictly based on the starting and ending time stamps, UTG introduces the relation of *more or less simultaneousness* which can be regarded as an approximated version of *equals* in Allen's relations [Mörchen, 2006b]. With the introduction of *more or less simultaneousness* in UTG by considering thresholds in comparing time intervals, it solves the problem of unrobustness in Allen's relations. Moreover, with the concept of hierarchy, it creates shorter and more abstract patterns which lead to earlier pruning and details on demand [Mörchen, 2006b]. In this manner, UTG also resolves the interpretability problems of Allen's relations. However, not all patterns described by UTG can be expressly described as by Allen's according to [Mörchen, 2006b]. One example given by [Mörchen, 2006b] is as shown in Fig. 3.3. There, an example pattern of Allen's *overlaps* relation with long prefix (l_1) and suffix (l_3) is presented, and, unfortunately, it can be described by neither *more or less simultaneous* relation nor the concept of hierarchy in UTG. To be able to describe it using UTG, l_1 and l_3 must be much shorter than l_2, and this limits UTG's expressivity.

To overcome the drawbacks of Allen's relations and UTG, TSKR is proposed as a solution and will be introduced later in this section. In summary, the basic comparison of Allen's relations, UTG and TSKR is described in Table 3.1. There, it shows that Allen's relations have drawbacks in robustness and interpretability, and UTG has difficulties in expressivity

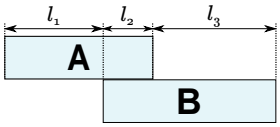


Figure 3.3: Inexpressivity of UTG. (after [Mörchen, 2006b])

as mentioned earlier. Furthermore, TSKR has much higher expressivity and much more robustness compared to Allen's relations and UTG without their drawbacks.

Table 3.1: Comparison of Allen, UTG, and TSKR for time interval representation (after [Mörchen, 2006b])

Comparison	Allen	UTG	TSKR
Robustness	-	+	++
Expressivity	+	-	++
Interpretability	-	+	+

Instead of 13 relations in Allen's relations, TSKR uses *duration*, *coincidence*, and *partial order* to describe the temporal relations of intervals, and consists of three types of components:

Tones: A Tone is the most basic component in TSKR, which describes a specific property or state within a certain time interval of a time series data. It represents the concept of duration, and is defined as a triple, $[s, e, \alpha]$ with $[s, e] \in \mathbb{T}$, $s \leq e$, and $\alpha \in \Sigma$, where s, e and α represent start time, end time, and symbolic value respectively [Mörchen, 2006a]. For instance, a rainfall log time series can be described as a sequential combination of descriptive rainfall information, such as heavy (> 10 mm/hr and ≤ 50 mm/hr), moderate (> 2 mm/hr and ≤ 10 mm/hr) and slight (≤ 2 mm/hr) according to [Met Office, National Meteorological Library and Archive, 2005]. Fig. 3.4 shows how Tones are derived from time series data. In Fig. 3.4, a set of time series data is grouped into three groups, and each group represents a Tone with the definition

above — start time, end time, and symbolic value. In the example of Fig.3.4, these groups are labeled as *Heavy*, *Moderate*, and *Slight* as depicted. After the derivation, the Tones contain no information regarding the value of each time series point but only the information of Tones as shown in the last step in Fig. 3.4.

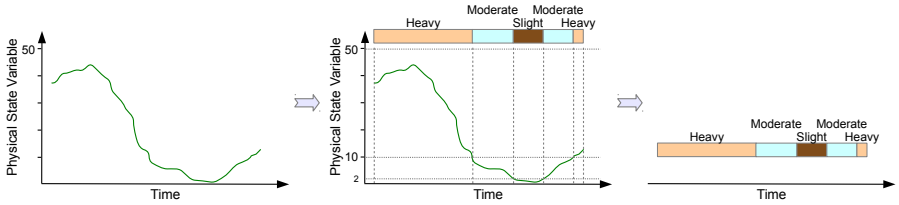


Figure 3.4: Derivation of Tones from time series data

Chords: A Chord represents coincidence, and it consists of distinct Tones occurring simultaneously within a certain time interval. Based on this definition, the size of a Chord, k, means the number of Tones it consists of and the Chord is denoted as k-Chord. The steps of the derivation of 3-Chords from three sets of time series data is shown in Fig. 3.5. Three sets of time series data, as shown in Fig. 3.5, are grouped into two groups, three groups and three groups respectively, to determine Tones, as the same process shown in Fig. 3.4. After Tones are derived, Chords are discovered through simultaneity among Tones. It groups, in general, the involved Tones based on the longest common length of interval. In the end, six different 3-Chords are derived in this example as shown in the last step of Fig. 3.5. Also, what is inside the dashed circle in Fig. 3.6 demonstrates how the first 3-Chord in the last step is composed. Another schematic example of the composition of a 3-Chord in hydrology can also be seen in Fig. 3.6 which will be discussed shortly.

A trivial Chord, 1-Chord, is a special case which is simply a copy of a Tone [Mörchen, 2006b]. If two Chords, c_i and c_j, where c_i contains the descriptions of subsets (Tones) from c_j, c_i is the sub-Chord of c_j and c_j is the super-Chord of c_i, denoted as $c_i \subset c_j$. Sub-Chords generally have longer duration compared to super-Chords because they are less restricted. Besides, larger Chords, which consist of more Tones, are

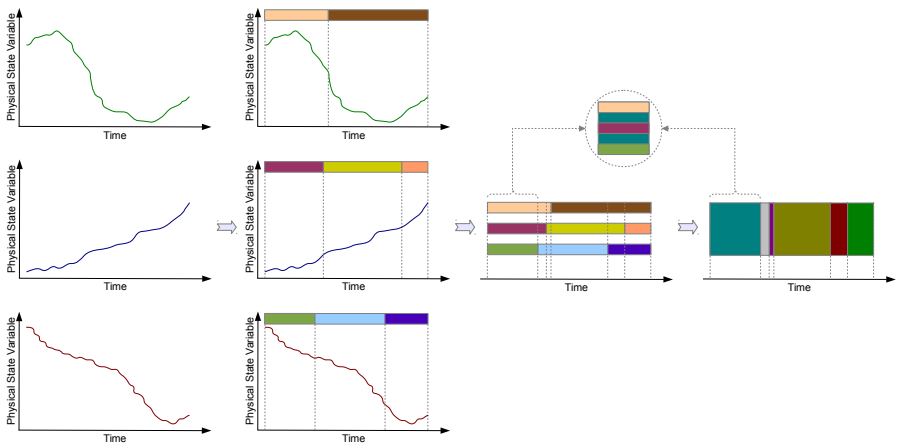


Figure 3.5: Derivation of Chords from time series data

normally more interesting because they are more specific [Mörchen, 2006a]. An example in hydrology can be a 3-Chord, as shown in Fig. 3.6, labeled as *aridity*, which consists of 3 Tones which are labeled separately as high air temperature, low precipitation, and low soil moisture as the aforementioned example of Event. In other words, these three simultaneous hydrological facts describe the phenomenon of aridity. Inside the Chord *aridity* exists another 2-Chord, named *drying*, which describes the process of getting dried. In this case, the Chord *aridity* is the super-Chord of the Chord *drying*, and the Chord *drying* is the sub-Chord of the Chord *aridity*.

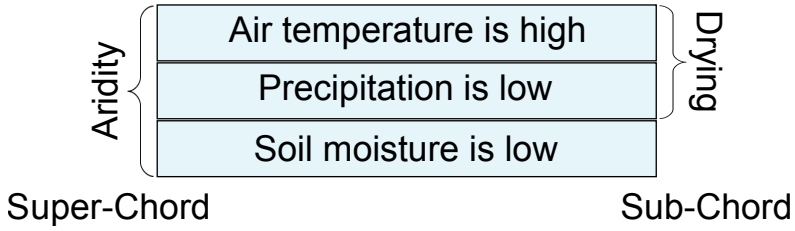


Figure 3.6: Representation of a Chord, a super-Chord and a sub-Chord

Phrases: A Phrase represents the concept of partial order and is constructed by a sequence of Chords without overlaps. Fig. 3.7 shows,

in general, how a Phrase is derived from available Tones. Three Tones A, B, and C are first used to derive three Chords — 2-Chord AB, 3-Chord ABC, and 2-Chord BC. Since overlapping Chords are not allowed in Phrases, these several two Chords, 2-Chord AB and 2-Chord BC, have to be "truncated" in order to form a Phrase. A Phrase indicates the process of events and also helps to identify how a phenomenon develops, e.g. how a drought happens.

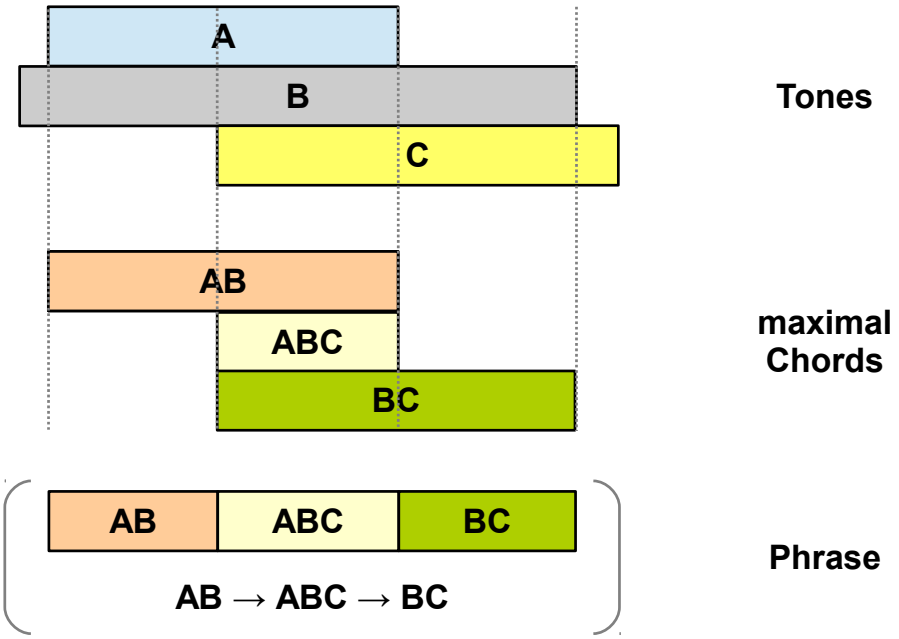


Figure 3.7: Derivation of a Phrase (after [Mörchen, 2006b])

These aforementioned components in TSKR form the basic foundation of TSKM. Further descriptions of how these components are derived and how knowledge is discovered will be illustrated in the coming Section 3.3.

3.3 Time Series Knowledge Mining (TSKM)

TSKM is a framework proposed by [Mörchen, 2006a,b; Mörchen et al., 2005], aiming to find and describe the temporal relations of multiple time series data sets for the purpose of temporal reasoning. The rules to describe

these relations derived by TSKM are based on the definition of TSKR as mentioned in Section 3.2. The entire workflow of TSKM contains five steps — pre-processing, finding Aspects, finding Tones, finding Chords, and finding Phrases, as shown in Fig. 3.8.

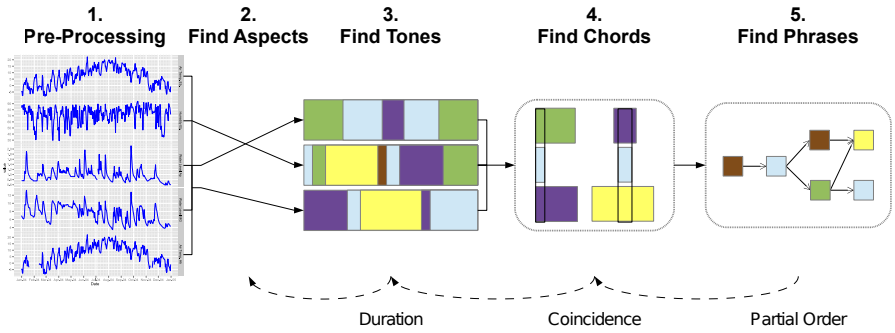


Figure 3.8: Pprocesses of the framework TSKM (after [Mörchen, 2006b])

Unlike Tones, Chords, and Phrases which are already defined in TSKR, Aspects have to be defined here. An Aspect is a group of time series data sets sharing similar semantics. The semantic here does not mean only physical property, such as rainfall, soil moisture, etc. Even with the same physical property, two different time series data sets can have different semantics. For example, two rainfall stations are separated by a hill and the rainfall time series data sets may have two different patterns. In this way, these two rainfall time series data sets can be seen as two Aspects.

The relations of TSKR, *duration*, *coincidence*, and *partial order*, are corresponding to the steps of finding Tones, finding Chords, and finding Phrases, as depicted in Fig. 3.8. Besides, as a rule of thumb in knowledge discovery, it is always recommended and sometimes necessary to have iterations between steps in order to acquire justifiable knowledge [Mörchen, 2006b]. These iterations are depicted as dashed lines in Fig. 3.8. This also implies that the entire process is not an one-way process but iterations of alternating between former and latter steps. The results of these iterations are usually difficult to be justified only by some indexes of algorithms and also have to be determined with the help of expert's knowledge. Due to this reason, the process of TSKM usually involves manual interactions. In this way, to achieve the final objective — temporal reasoning from existing

available data, the entire process is considered as a semi-automatic process instead of a full automatic one.

The following are the descriptions of each step in TSKM:

Pre-Processing: Data pre-processing is a crucial step not only in TSKM framework but also in data mining. Collected time series data are often incomplete, noisy, inconsistent, and so on, in many applications and projects, such as [Hinkelmann et al.; Molkenthin et al., 2014; Zehe and Hinkelmann, 2013]. For instances, due to all kinds of happenings occurred during data collection, the data can contain missing values, biased values, outliers, etc. as described in Section 2.3. In addition, different facilities and tools for data collection also have different design and specifications, and these also cause the inconsistency of the collected data. Furthermore, some issues of different scales in time and space, transformations to proper domain, enhancement for further analysis process, etc. also have to be taken into account. A proper data pre-processing helps not only to facilitate subsequent analyses, but also to give an overview of what kind of a problem to deal with. Some hydroinformatics tools, such as MIKE [DHI], also have some built-in functionalities for data pre-processing.

However, the techniques for data pre-processing are domain- and problem-specific. One typical example can be the missing values due to the malfunction of a collecting device. In this way, these values can be replaced by mean values, interpolation values, representative values compared to historical data, or special flags depending on the type of data, problem, etc. The techniques for data pre-processing can include digital filters for noise removal [Smith, 1997] , interpolation functions for gap filling, box plots and Grubbs' Test [Grubbs, 1969] for outlier detection, Principal Component Analysis (PCA) [Jolliffe, 2002], Independent Component Analysis (ICA) [Hyvärinen, 1999], and Self-Organizing Map (SOM) [Kantola, 2012] for feature extraction and further dimension reduction, and Fourier transformation and Hilbert-Huang Transform (HHT) [Huang and Wu, 2008] for domain transfor-

mation etc.

Finding Aspects: An Aspect is a group of time series data sets sharing similar semantics. In other words, this step is actually a continuation of pre-processing or a subset of pre-processing, and it tries to reduce the dimension of input data sets to be analyzed. For instance, d time series data sets (d-dimensional inputs) can be reduced into k different semantic blocks (k Aspects), and these k Aspects are the subset of original d-dimensional inputs. To achieve this reduction, PCA can be used for highly correlated variables and ICA can be used to reveal independent influences [Mörchen, 2006b]. The results of grouping, i.e. Aspects, should be consistent with the domain knowledge, which the problem of interest belongs to. In an extreme case, one Aspect per time series set can be used [Mörchen et al., 2005]. Additionally, [Mörchen, 2006b; Ratanamahatana et al., 2005] also strongly recommend performing normalization, e.g. by means of mean and standard deviation, to avoid unwanted biases happening during further analyses.

Finding Tones: The process of finding Tones is a process of converting each Aspect into a sequence of symbolic Tones with their maximum occurrences. Through this process, the numeric data types of time series data are converted into human-readable and meaningful descriptions. For example, the intensity of rainfall can be categorized into heavy, moderate, and slight as mentioned earlier according to [Met Office, National Meteorological Library and Archive, 2005]. Based on this category, a collected rainfall time series data can be converted into a series combination of descriptions, e.g. [slight, moderate, slight, moderate, heavy, moderate, . . .], depending on the content of the data.

Except with the help of expert's knowledge, the methods of clustering, segmentation, rule generation, etc. are also available to find Tones depending on the problem of interest. Nevertheless, these methods should be carefully applied to obtain meaningful bins which can be approved by experts. In [Mörchen, 2006b], the algorithm PERSIST [Mörchen and Ultsch, 2005] is recommended and is applied in skating

data sets. In [Gronz et al., 2008], the algorithm PERSIST is found well-performed in soil moisture data sets however not suitable for other hydrological data sets, e.g. precipitation. The Emergent Self-Organizing Map (ESOM) with U-Matrix [Ultsch, 2003] is used for the data sets of sleep related breathing disorders [Ultsch, 1999] and skating data sets [Mörchen et al., 2005]. In [Moskovitch et al., 2007], the algorithm PERSIST is compared with a human expert and Symbolic Aggregate approXimation (SAX) [Lin et al., 2003], an algorithm based on Piecewise Aggregate Approximation (PAA) [Keogh and Pazzani, 2000] featuring in dimension reduction and indexing with lower bounding measurement, in finding proper bins. The algorithm PERSIST receives a comment that it is not suitable for domains like medicine due to its assumption of uniform sampling of time series data. However, it is claimed to outperform other methods, like k-means clustering, Hidden Markov Model (HMM), Gaussian Mixture Model (GMM), etc. in [Mörchen and Ultsch, 2005]. On the basis of these studies, the methods to find Tones are also problem-dependent and the results have to be further validated by experts. In addition, more investments are also needed to decide which techniques are suitable for different types of hydro science and engineering data.

Additionally, the discovered Tone patterns may be interrupted by noisy data, which lead to breaking down a long-interval Tone into several Tones with different intervals. To prevent this issue, it is also recommended to filter out the small interval gaps caused by noises through proper filters [Mörchen et al., 2005], e.g. filtering by proper relative interval ratio and maximum interval value of a gap.

Finding Chords: As mentioned in Section 3.2, chords represent the concept of coincidence, and the process of finding Chords is to consider all Aspects concurrently and to mine the simultaneous occurrences among Tones. In this manner, the inputs are the multiple sequences of Tones, and the number of these sequences is based on the step of “Finding Aspects”. The output is a single sequence of labeled intervals — Chords.

To find Chords, a linear depth-first approach based on the algorithm for Closed Association Rule Mining (CHARM) [Zaki and Hsiao, 2002] is proposed in [Mörchen, 2006a,b] because the similarity of a Chord and an itemset in association rules mining. In addition, similar to the concept of *more or less simultaneous* relation in UTG as mentioned in Section 3.2, some Chords might be similar and can be seen as an equivalent to each other. In order to take this into account, the concept of margin-closeness is also introduced into the algorithm. A Chord c_i can be viewed as a margin-closed Chord when no super-Chord having almost the same support exists, and the definition is:

$$\frac{sup_\delta(c_j)}{sup_\delta(c_i)} < 1-\alpha, \qquad c_i \subset \forall c_j \tag{3.1}$$

In this definition, α is a threshold to determine margin-closeness and the default value of α is set to be 0.1, which means 10% of differences among intervals of Chords can be accepted. $sup_\delta()$ represents the support of a Chord. The support of a Chord, as shown in Fig. 3.9, is the interval of all maximum occurrences with at least minimum length δ, which depends on the study of interest. With the introduction of $sup_\delta()$ as well as α, it determines the degree of the concept of *more or less simultaneous* relation in UTG. In addition, the minimum size of the Chord is set at least to be two according to the algorithm.

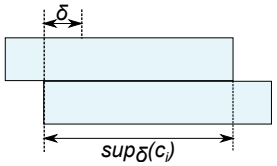


Figure 3.9: Representation of the support of the Chord c_i

Finding Phrases: A Phrase represents the concept of a partial order as mentioned in Section 3.2, and Finding Phrases is the last step towards temporal reasoning according to the framework of TSKM. Before this

step, all time series data have been converted into series of temporal intervals. In order to find Phrases, these series of temporal intervals, Chords, are taken as inputs and the outputs are a set of Phrases representing temporal developments of phenomena described by TSKR.

The concept of finding Phrases actually contains two phases. The first phase is to find sequences and many techniques in sequence mining are available since the outputs, Phrases, have similarity to itemsets. The second phase is similar to the concept of pruning, which tries to find generalized sequences by grouping similar ones because sequences are often overlapping. In [Mörchen, 2006b], the combination of CLOsed Sequential PAtterN mining (CloSpan) [Yan et al., 2003] and modified CHARM algorithm is proposed to find proper Phrases inside the TSKM framework. However, similar to finding Chords, other available techniques exist for this purpose, especially those for sequence mining, due to the property of the inputs [Mörchen et al., 2005]. For instance, the suffix trie [Vilo, 1998] is used for moderate-sized data sets and the mining techniques from [Yang et al., 2002] are proposed for large-sized data sets in [Mörchen et al., 2005]. A suffix tree is one of the most common ways to describe the order of the series, and algorithms of creating suffix trees are widely applied in fields, such as string mining, gene mining, and so on. An advantage of using suffix trie is that the results can be easily queried, e.g. the frequency, to find interesting patterns [Mörchen et al., 2005]. In [Mörchen and Ultsch, 2004], the algorithm Sequitur [Nevill-Manning and Witten, 1997] is applied, and several unsupervised techniques for temporal rules mining are compared, e.g. suffix trie [Vilo, 1998], association rules [Das et al., 1998], Multi-Stream Dependency Detection (MSDD) [Oates et al., 1997], etc., in [Mörchen, 2006b].

TSKM has been applied in different disciplines, especially in temporal reasoning — mining knowledge from temporal data, using the representation of TSKR. Some of these applications are related to sport medicine with skating data [Mörchen, 2006a,b; Mörchen et al., 2005], artificial intelligence with

video data for visual recognition [Mörchen, 2006b], software engineering for software specification discovery [Lo and Khoo, 2008], medicine [Moskovitch et al., 2007], and hydrology [Gronz et al., 2008].

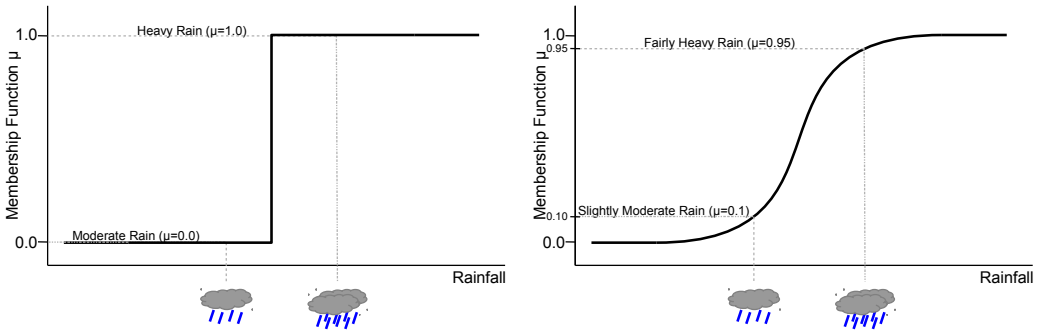
3.4 Fuzzy Logic

Since the logic was introduced by the Greek philosopher and polymath Aristotle in 300 BC, it has been the fundamental principle of mathematics, and the later influenced Boolean logic, developed by [Boole, 1854], is the basic of modern computer science. These two are the groundwork of modern science and engineering, and hydroinformatics is no exception. However, as Albert Einstein addressed in his lecture in 1921 in Berlin [Einstein et al., 1922], it reveals a room of vagueness in the traditional concept of bivalence developed since Aristotle's time:

> So far as the laws of mathematics refer to reality, they are not certain. And so far as they are certain, they do not refer to reality.

Fuzzy logic is based on the fuzzy set theory introduced by [Zadeh, 1965]. Unlike conventional set theory, which any item either belongs, or does not belong, to a set of items without ambiguity, fuzzy set theory introduces the concept of degrees of truth. Based on the classification given in [Met Office, National Meteorological Library and Archive, 2005], four different intensities of rainfall, violent, heavy, moderate, and slight, are defined separately as greater than 50 mm/h, 10 to 50 mm/h, 2 to 10 mm/hr, and less than 2 mm/hr. Suppose two measured rainfall intensity records are given, 47.6 mm/hr and 6.8 mm/hr, and they will be classified as heavy rain and moderate rain respectively based on the classification given above, as shown in Fig. 3.10a. Based on the conventional set theory, there is no ambiguity to allocate the record of 47.6 mm/hr to the category of heavy rain even though it almost reaches its upper boundary. On the other hand, the membership function (μ) is introduced into the fuzzy set theory, which maps the degrees of truth into the interval $[0,1]$. In this case, the record of 47.6 mm/hr is assigned

to the value of $\mu = 0.95$, which denotes fairly heavy rain, and the record of 6.8 mm/hr is given the value of $\mu = 0.1$, which represents the slightly moderate rain as shown in Fig. 3.10b. In other words, with the help of membership function, this provides the capability to describe the ambiguity caused by the more-or-less truth.



(a) Conventional set (b) Fuzzy set

Figure 3.10: Illustration of different set theories

As mentioned earlier, fuzzy logic is capable of dealing with uncertainties. However, unlike stochastic uncertainty, which is caused by the occurrence of the event itself, what fuzzy logic deals with is the so-called lexical uncertainty, which depends on the definition of the event itself. These kinds of uncertainties are subjective and the nature of the human languages. Like the previous example, the record of 47.6 mm/hr event may be assigned to the value of $\mu = 0.2$ of violent rain depending on the choice of membership function.

In addition to the capability of dealing with uncertainties, fuzzy logic is also able to embody the contributions of human logic. Thus, the knowledge and the experience of experts can be considered as well. On the whole, fuzzy logic is able to work with linguistic variables, tolerate imprecise data, and to implement human logic into scientific or engineering solutions.

In the field of hydroinformatics, many applications also apply fuzzy logic theories. In addition to the applications mentioned above, there are also other applications, such as water demand forecasting [Bárdossy and Duckstein, 1995], groundwater infiltration process description [Bárdossy and

Disse, 1993; Bárdossy and Duckstein, 1995], habitat modeling [Lange et al., 2013], reconstruction missing precipitation events [Abebe et al., 2000], classification of atmospheric circulation patterns [Bárdossy and Duckstein, 1995; Bárdossy et al., 1995], classification of hydrologically homogeneous gauged regions [Hall and Minns, 1999], reservoir operation rules derivation [Shrestha et al., 1996], rainfall-runoff simulation [Özelkan and Duckstein, 2001], hybrid deterministic fuzzy rule based model for nitrate transportation [Shrestha et al., 2007], hybrid neuro-fuzzy model for hydrological time series simulation [Nayak et al., 2004], and the like.

In order to better delineate fuzzy logic, several definitions have to be clarified:

Fuzzy Sets: The term "fuzzy set" is the core concept fuzzy logic theories built upon, and has been brought up several times earlier without a clear definition. Based on the description by Lofti A. Zadeh himself, the definition of fuzzy set is [Zadeh, 1965]:

> Let X be a space of points (objects), with a generic element of X denoted by x. Thus $X = \{x\}$.
>
> A fuzzy set (class) A in X is characterized by a *membership (characteristic) function* $f_A(x)$ which associates with each point in X a real number in the interval $[0,1]$, with the values of $f_A(x)$ at x representing the "grade of membership" of x in A. Thus, the nearer the value of $f_A(x)$ to unity, the higher the grade of membership of x in A.

Based on this definition, the definition of a fuzzy set can be written as:

$$A = (x, \mu_A(x)), \qquad x \in X, \mu_A(x) \in [0,1] \tag{3.2}$$

where A is the fuzzy subset of X, and μ_A is the membership function of A, which describes to which degree x belongs to A, and the value of membership function is limited in the interval $[0,1]$.

Membership Functions: The membership function, as mentioned earlier,

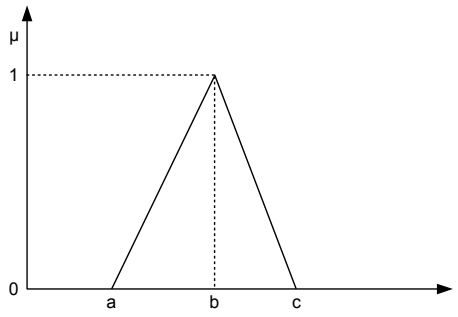
represents the degree of membership of x in A. There is no other special condition a membership function must fulfill, except the value of it must inside the interval $[0,1]$. Thus, a special membership function is:

$$\mu_A(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases} \tag{3.3}$$

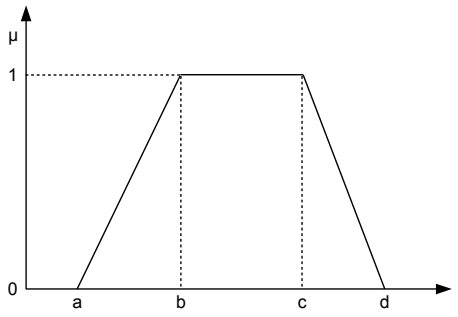
It is a step function which describes the conventional set in the way of fuzzy logic. In this manner, the conventional set can be regarded as the subset of the fuzzy set.

Other than this special membership function, here are several common membership functions [Gautam, 2000; IEC 1131, 1997]:

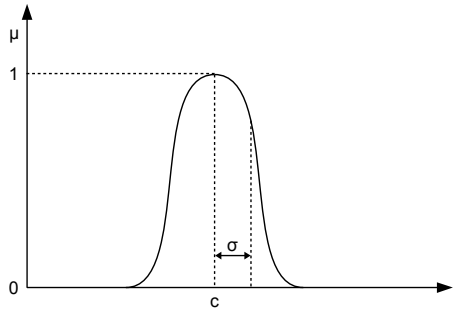
- Triangular Membership Function
  The triangular membership function, as shown in Fig. 3.11a, is given by:
  $$\mu_A(x) = \begin{cases} 0 & x \le a \\ \frac{x-a}{b-a} & a \le x \le b \\ \frac{c-x}{c-b} & b \le x \le c \\ 0 & c \le x \end{cases} \tag{3.4}$$
  which contains three parameters, a, b, and c. The parameter b is the peak and the other two parameters, a and c, are the base points of the triangle.
- Trapezoidal Membership Function
  The trapezoidal membership function, as shown in Fig. 3.11b, is given by:
  $$\mu_A(x) = \begin{cases} 0 & x \le a \\ \frac{x-a}{b-a} & a \le x \le b \\ 1 & b \le x \le c \\ \frac{d-x}{d-c} & c \le x \le d \\ 0 & d \le x \end{cases} \tag{3.5}$$
  which contains four parameters, a, b, c and d. The parameter b and c are two upper points of the trapezoid and the parameters a



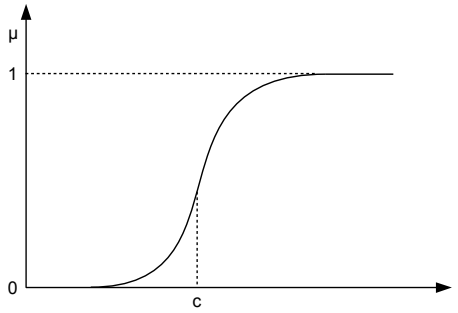
(a) Triangular membership function



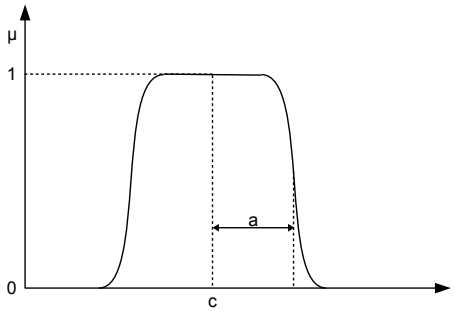
(b) Trapezoidal membership function



(c) Gaussian membership function



(d) Sigmoidal membership function



(e) Generalized bell-shaped membership function

Figure 3.11: Illustration of different membership functions

and d are the two bases of the trapezoid. Besides, the trapezoidal membership function can be regarded as the extension of the triangular membership function. In the case when b equals to c, the trapezoidal membership function is actually the triangular membership function.

- Gaussian Membership Function
  The Gaussian membership function, as shown in Fig. 3.11c, is given by:
  $$\mu_A(x) = e^{\frac{-(x-c)^2}{2\sigma^2}} \tag{3.6}$$
  which contains two parameters, c and σ. The parameter c determines the center of the function in the x-axis and the parameter σ is the standard deviation by definition which describes the span of the function.

- Sigmoidal Membership Function
  The sigmoidal membership function with the "S" shape, as shown in Fig. 3.11d, is given by:
  $$\mu_A(x) = \frac{1}{1 + e^{-a(x-c)}} \tag{3.7}$$
  which contains two parameters, a and c. The parameter c is the center of the function in the x-axis, and the parameter a determines the direction of the function and the steepness of the "S" shape.

- Generalized Bell-Shaped Membership Function
  The generalized bell-shaped membership function, as shown in Fig. 3.11e, is given by:
  $$\mu_A(x) = \frac{1}{1 + \left|\frac{x-c}{a}\right|^{2b}} \tag{3.8}$$
  which contains three parameters, a, b and c. The parameter a decides the "width" of the function, the parameter b determines the steepness of the curve at both sides, and the parameter c is where the center of the function locates.

- Miscellaneous Membership Functions
  There are some other type of membership functions, such as the spline-based membership function, the piece-wise linear membership function, singleton membership function, etc. The combina-

tion of the functions mentioned earlier also servers the purpose as a membership function, as long as it satisfies the only criterion — the value of μ_A must be inside the interval $[0, 1]$.

Fuzzy Rules: Fuzzy rules are a set of linguistic rules, which serves as the "brain" of the fuzzy logic system. The set of rules helps the system to draw inferences based on the descriptions of rules. The illustration of the composition of rules and the process of inference is shown in Fig. 3.12. The rules are in the form of IF-THEN statements, and they describe which action is performed under which condition. The process of inference contains three steps [IEC 1131, 1997]:

- Aggregation: This step is to aggregate all the conditions through logic operators, *AND*, *OR* and *NOT*, and it determines the overall degree of accomplishment.
- Activation: This step decides the action based on the overall IF condition with consideration of the weighting factor of each rule.
- Accumulation: This step is to derive the overall result by combining the action result of each rule.

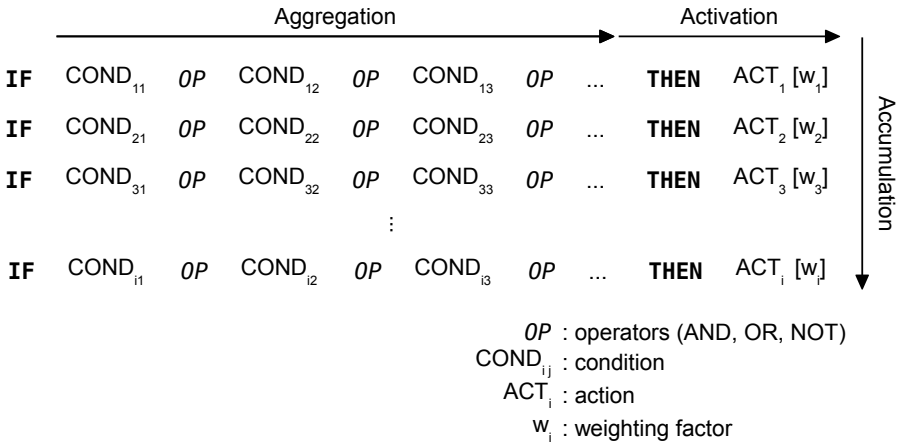


Figure 3.12: Illustration of fuzzy rules

The acquisition of fuzzy rules is a delicate part of applying fuzzy logic. In general, these rules can be acquired by expert knowledge,

derived from existing data, or the combination of the previous two approaches. Several different approaches were suggested or applied to extract these rules from existing data, such as clustering method [Chiu, 1994; Yager and Filev, 1994] and TSKM [Gronz et al., 2008]. Besides, optimization techniques can also be applied to adjust the weighting factors of rules or membership functions, and one of the examples is Adaptive Neuro Fuzzy Inference System (ANFIS) [Jang, 1993] which uses Artificial Neural Network (ANN) as a means of optimization.

Fuzzy Operations: Fuzzy operations express the "logic" of fuzzy logic, and they are also essential to the fuzzy logic reasoning. Like conventional set theory, some rules, such as the de Morgan's law[4], associativity, commutativity, and distributivity, also apply to fuzzy set theory. As mentioned earlier in fuzzy rules, these operations can be categorized into [IEC 1131, 1997]:

- Aggregation: In this section, the common operations are *intersection*, *union* and *complement*, and their operators are denoted as *AND*, *OR* and *NOT* respectively. In order to fulfill de Morgan's law, these operators should appear paired-wise and their mathematical definitions are shown in Table 3.2.

Table 3.2: Definitions of operations in aggregation

Intersection (AND)	Union (OR)	Complement (NOT)
$\text{Min}(\mu_A(x), \mu_B(x))$	$\text{Max}(\mu_A(x), \mu_B(x))$	$1 - \mu(x)$
$\mu_A(x)\mu_B(x)$	$\mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x)$	$1 - \mu(x)$
$\text{Max}(0, \mu_A(x) + \mu_B(x) - 1)$	$\text{Min}(1, \mu_A(x) + \mu_B(x))$	$1 - \mu(x)$

[4] De Morgan's law:

$$(A \cup B)' = A' \cap B'$$
$$(A \cap B)' = A' \cup B'$$

where A and B are any two subsets of set X, $\cup$ represents union operator (OR), $\cap$ represents intersection operator (AND), and $'$ represents negation (NOT).

- Activation: The common operations to convert IF-THEN results are *product* and *minimum*, and the mathematical definitions are described in Table 3.3.

  Table 3.3: Definitions of operations in activation

Product	Minimum
$\mu_A(x)\mu_B(x)$	$\text{Min}(\mu_A(x), \mu_B(x))$

- Accumulation: In order to combine the result of each rule into one single result, the ordinary operations are *maximum*, *bounded sum* and *normalized sum*, and the definitions in mathematics are shown in Table 3.4.

  Table 3.4: Definitions of operations in accumulation

Maximum	Bounded Sum	Normalized Sum
$\text{Max}(\mu_A(x), \mu_B(x))$	$\text{Min}(1, \mu_A(x) + \mu_B(x))$	$\frac{\mu_A(x)+\mu_B(x)}{\text{Max}(1,\mu_A(x)+\mu_B(x))}$

Fuzzy Inference System: Fuzzy inference is to formulate the mapping between a given input and an output using fuzzy logic with linguistic rules. The system which applies fuzzy inference to map between a set of inputs and a set of outputs is called Fuzzy Inference System (FIS). Fig. 3.13 shows the process of a FIS, which contains:

- Fuzzification: The process of fuzzification turns real values into degrees of the membership function for linguistic terms of fuzzy sets. For instance, a temperature of 15°C can be represented by 50% of Cold ($\mu_{\text{Cold}} = 0.5$) together with 50% of Warm ($\mu_{\text{Warm}} = 0.5$), as shown in Fig. 3.14.
- Fuzzy Inference: As mentioned in the section of Fuzzy Rules, fuzzy inference uses linguistic variables to construct rules for reasoning,

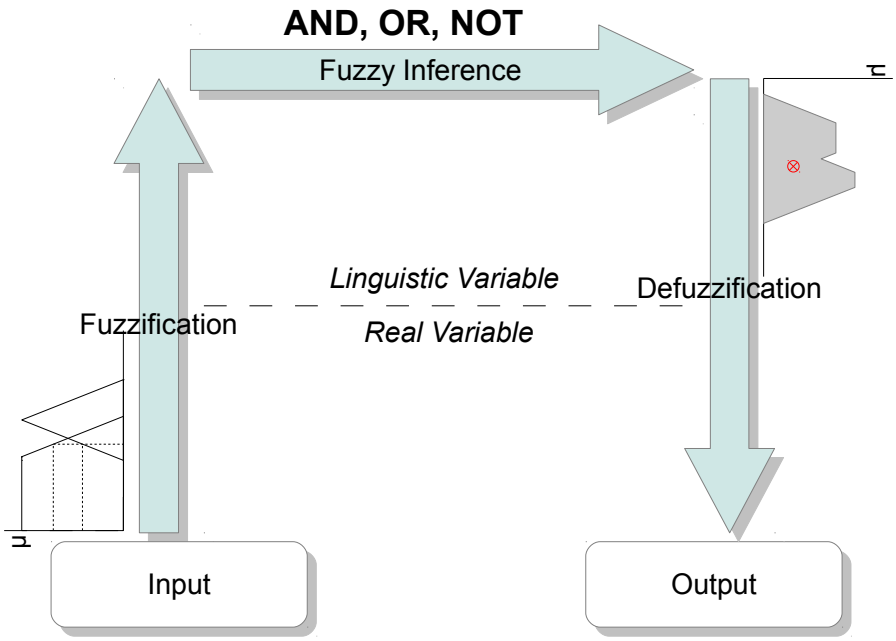


Figure 3.13: Fuzzy inference system process

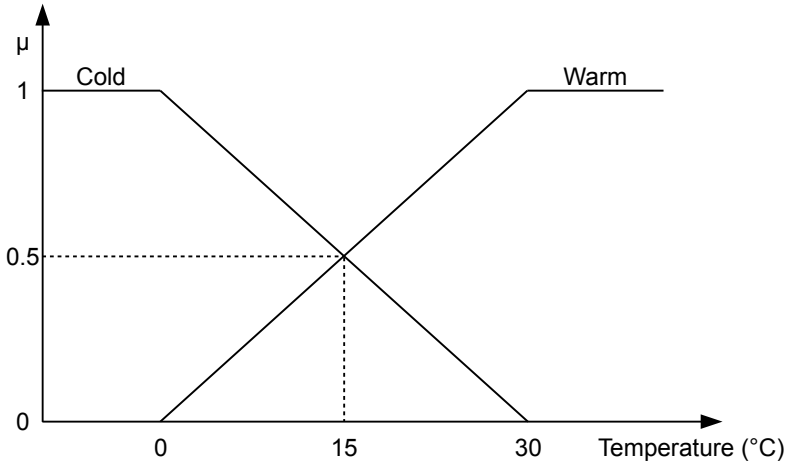


Figure 3.14: Illustration of fuzzification

and serves as the “brain” of the fuzzy system. An example of how fuzzy inference describing rules based on the IF-THEN statement is as follows:

IF
precipitation is HIGH **AND** discharge is HIGH
THEN
reservoir water level is HIGH

The description above illustrates a general example how the water level of a reservoir is influenced by precipitation and discharge. Although the definitions of HIGH in all physical variables are not defined here, this description can be cognitively comprehended. The definitions of these descriptions of different physical variables have to be done in the process of defining fuzzy sets and can be problem-specific.

- Defuzzification: Defuzzification is a process of converting the accumulated fuzzy result (as shown in Fig. 3.12) into a specific value. In other words, the accumulated linguistic consequence is normally interpreted into a single real value. Several methods exist to defuzzify this accumulated membership function, such as Center of Area (CoA), Center of Gravity (CoG), Middle of Maximum (MoM), Last of Maximum (LoM), First of Maximum (FoM), Fuzzy Mean (FM), etc. [IEC 1131, 1997; Leekwijck and Kerre, 1999]. Some of these common methods are described in the form of a formula to find the defuzzified value x':
  - Center of Area (CoA): CoA represents the center of area method, and it finds the value x' which divides the area under the membership function evenly. Therefore, it has to satisfy:

$$\int_{\text{Min}}^{x'} \mu(x)dx = \int_{x'}^{\text{Max}} \mu(x)dx \tag{3.9}$$

    where Min and Max are the lower and upper boundaries for defuzzification. An illustration of CoA is shown in Fig. 3.15.
  - Center of Gravity (CoG): CoG is the center of gravity method which calculates where the center of gravity is located. The

formula can be written as:

$$x' = \frac{\int_{\text{Min}}^{\text{Max}} x\mu(x)dx}{\int_{\text{Min}}^{\text{Max}} \mu(x)dx} \tag{3.10}$$

where Min and Max are the lower and upper boundaries for defuzzification. However, one special case is that the membership function is singleton, this CoG turns into Center of Gravity for Singleton (CoGS) and the integral form becomes discrete as:

$$x' = \frac{\sum_{i=1}^{N} x_i \mu_i}{\sum_{i=1}^{N} \mu_i} \tag{3.11}$$

where N is the number of singletons. An illustration is shown in Fig. 3.15.

- Last of Maximum (LoM): LoM seeks the location where the nearest maximum membership function value locates, as shown in Fig. 3.15.
- Middle of Maximum (MoM): Like LoM, MoM attempts to find where the middle of the maximum membership function value is, as shown in Fig. 3.15.
- First of Maximum (FoM): Similar to LoM and MoM described above, FoM finds the farthest point of which the maximum membership function stands, as shown in Fig. 3.15.

There are two types of FISs: Mamdani-type [Mamdani and Assilian, 1975] and Sugeno-type [Sugeno, 1985]. The Mamdani-type FIS is the most common one seen as discussed so far. The main difference between these two types of FISs lies in the output membership function. Different from the Mamdani-type FIS, the output membership function in Sugeno-type FIS is either linear or constant, and sometimes is known as singleton. Hence, the CoGS is used as the defuzzification method.

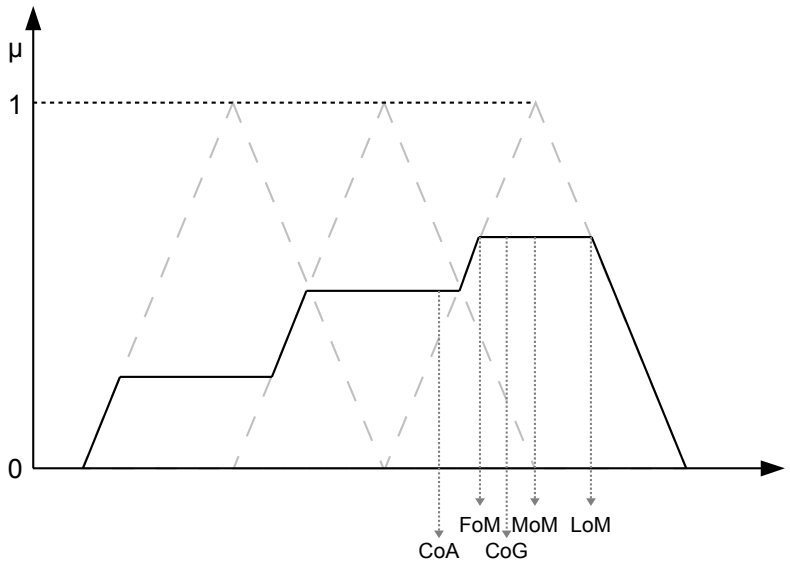


Figure 3.15: Illustration of defuzzification

3.5 Multivariate Adaptive Regression Splines (MARS)

In the module of process identification (see Fig. 4.1), an additional/optional step is introduced at the end of the process identification — regression. Although fuzzy rules can be derived from events that are the outputs in the module of event identification, many factors still affect the results of process identification, and one crucial factor is the completeness of these rules [Bárdossy and Duckstein, 1995]. Moreover, there are also many parameters and choices introduced during this semi-automatic event identification process including subjective ones, such as expert knowledge, and objective ones, such as limitations of algorithms. Due to these uncertainties, the derived relationships among variables may not be sufficient enough to precisely describe themselves. Instead, only trends among different variables are captured. Hence, a mathematical mapping is proposed to resolve this issue — Multivariate Adaptive Regression Splines (MARS), and several applications and discussions of using MARS in the field of hydroinformatics can be found in [Coulibaly and Baldwin, 2005; Herrera et al., 2010; Sharda et al., 2008].

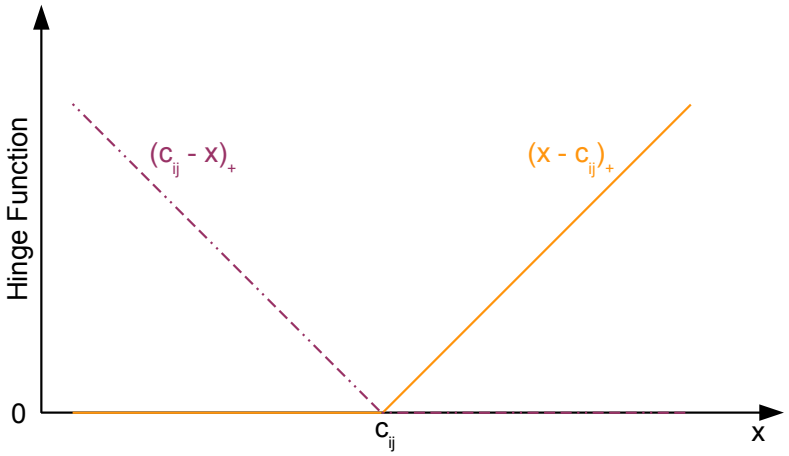


Figure 3.16: Illustration of hinge functions

MARS is a type of nonparametric regression and can be regarded as a linear combination of multiple basis functions. Compared to other regression techniques, such as linear regression, nonlinear regression, regression trees, etc., MARS does not require any a priori assumption, and the general form is written as:

$$\hat{f}(x) = \beta_0 + \sum_{j=1}^{M} \beta_j H_j(x) \tag{3.12}$$

where β_j represents a constant coefficient for the corresponding basis function, $H_j(x)$, as mentioned earlier. The j is the index of basis function and the number M is the total number of basis functions which formulate this nonparametric regression model. The basis function H_j is made up of the product of hinge functions as:

$$H_j(x) = \prod_{i=1}^{N} h_{ij}(x) \tag{3.13}$$

where i is the index of hinge function, $h_{ij}(x)$, and the number N is the total number of hinge functions which construct the basis function. The form of the hinge function is written as (see Fig. 3.16):

$$h_{ij}(x) \equiv (x - c_{ij})_+ = \begin{cases} x - c_{ij} & x > c_{ij} \\ 0 & \text{otherwise} \end{cases} \tag{3.14}$$

or

$$h_{ij}(x) \equiv (c_{ij} - x)_+ = \begin{cases} c_{ij} - x & c_{ij} > x \\ 0 & \text{otherwise} \end{cases} \tag{3.15}$$

where c_{ij} is any arbitrary constant representing the *knot* of the basis functions, and the + sign means that only the positive results are considered. These two hinge functions are considered as a pair and are directly derived from data. In this way, MARS describes the nonlinearity of the problem of interest through the concept of piecewiseness.

The process of constructing MARS models involves two phases:

Forward Phase: In this phase, the algorithm greedily searches and adds pairs of hinge functions which lead to the maximum reduction of the Residual Sum of Squares (RSS) until the termination criteria are reached. The termination criteria can be either the change of the RSS small enough or the maximum number of pairs reached.

Backward Phase: Like many algorithms in Machine Learning (ML), especially the nonparametric ones, the derived model from the forward phase is probable that the model is overfitting due to its adaptivity to data. To avoid overfitting, pruning the number of hinge functions is used to reach the generalization of the model. In this phase, hinge functions are dropped one by one until the *best* model is reached. The criterion to choose the *best* model in the algorithm of MARS is the Generalized Cross Validation (GCV):

$$\text{GCV} = \frac{\sum_{i=1}^{N} \left(y_i - \hat{f}(x_i)\right)^2}{\left(1 - \frac{r + cK}{N}\right)^2} \tag{3.16}$$

where N is the number of observations, c is the penalty, which is in the range $2 < c < 3$, r is the number of independent basis functions, and K is the number of knots [Hastie et al., 2009].

3.6 Suffix Tree

Suffix tree is a type of data structure in computer science. As its name suggests, this data structure is tree-like and is able to provide the suffix based on a given string and it is widely used in string operations. The structure of suffix tree is shown in Fig. 3.17 with a given string “mississippi” as an example. A special character, e.g. “$”, denoting the end of the string is attached to the given string. The edges of the suffix tree structure are labeled with substrings of the given string, and the nodes denote the beginning, the end, and the string split of the data structure. Three types of nodes exist in the structure:

- Root node: There is only one root node in the entire structure and it is the beginning of the data structure.
- Leaf nodes: Leaf nodes stand for the end of suffix and usually come together with the special termination symbol. In the schematic example Fig. 3.17, they appear in the colored circle.
- Internal nodes: The rest of nodes, which are neither root nor leaf nodes, are called internal nodes. These nodes create the branches of the tree structure and separate the given string into substrings.

With this given example (see Fig. 3.17), the string “mississippi” is restructured into the combination of different nodes and edges, and the suffix(es) can be easily identified by a given string. In this example, two suffixes, “i” and “pi”, are identified if the given string is “p” which is the rightmost branch in Fig. 3.17. In other words, once “p” is identified in the data structure, the following strings, “i” and “pi”, can be easily found and the sequences of these strings can also be known. Another example, which is the extreme case, is that the string “mississippi” itself is its own suffix shown at the leftmost branch in Fig. 3.17. With the help of the suffix tree, a long string can be compressed and its suffixes can be easily identified, which is especially useful for some applications, such as the Human Genome Project [Collins et al., 2003].

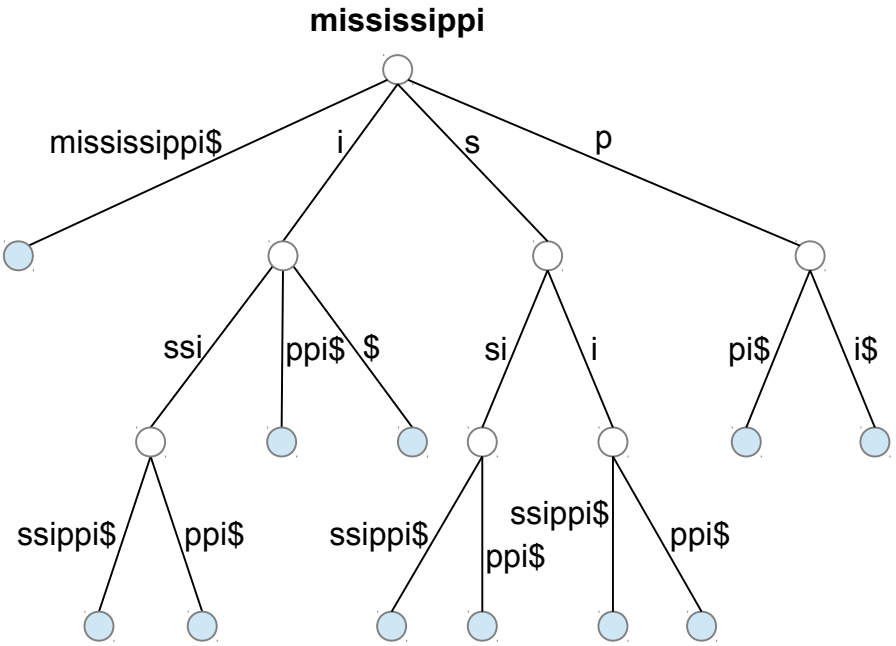


Figure 3.17: Schematic representation of the suffix tree data structure

Constructing a suffix tree for a given string usually takes computational memory space and time depending on the implementation. The basic algorithm requires $O(n^2)$ to $O(n^3)$ memory space and Central Processing Unit (CPU) time, where n is the length of the given string. Several improved algorithms provide better ways to reduce the resources needed. For example, a real-time online algorithm for constructing suffix trees proposed by [Ukkonen, 1995], known as Ukkonen's algorithm, reduces the dimension to $O(n\log n)$ on average by starting with an empty tree containing the first character and updating this tree structure with the help of the suffix pointer[5] till it is completed.

Several application fields using suffix trees are as follows:

- String search
- DNA or protein patterns identification
- Data compression

[5]The suffix tree pointer indicates where to break the substring and insert a new edge with a corresponding substring in the tree data structure.

and an application using suffix tree mining hydrological periodic pattern can be found in [Zhu et al., 2012].

Chapter 4

Framework Concepts

4.1 Problem Analysis

A general workflow of research or professional projects/applications in the field of hydroinformatics dealing with real data can be divided into three parts as shown in Fig. 2.1. These three parts represent the concepts of:

- Preparation: A preparation of everything for the purpose of simulation, including system setup, data collection, data pre-processing, etc., which is included in the leading component in the hydroinformatics system as shown in Fig. 2.1.
- Simulation: As shown in the core component in Fig. 2.1, this part serves as an engine of the problem-solving for the targeting problems. Depending on the type of the problems, it can be related to numerical analysis, statistical analysis, optimization, etc.
- Finalization: During simulation, the results, usually in the form of numbers, are generated. In this part, shown in the trailing component in Fig. 2.1, tools of different purposes, e.g. visualization tools, are usually applied to summarize the simulation results into more expressive representations, such as graphs, tables, etc. In this way, together with the help of experts' experience, conclusions, solutions, further proposals can be drawn.

Based on the workflow, several problems still exist, especially in the projects dealing with multi-disciplinary fields, even if the tools for these three parts are well-selected and appropriate for such projects. Concerning the objective of this research work, these problems are:

- Scenario Sparseness: Suppose the simulation tools are able to well-represent the targeting problem and the tasks in both preparation and finalization are satisfactory. The investigation of the same problem with different what-if scenarios often troubles modelers. The reason is that the data sets for these scenarios are often not existing. Without these corresponding data sets, the impacts of such scenarios are difficult to determine.

- Mass Data: In the projects/applications dealing with real data, the data are usually collected in a great amount. With the current technology, it is usually not an issue to collect and store these data. Instead, how to parse the data efficiently and to extract necessary information without being buried in the nowhere of mass data are usually the major concerns.

- Workflow Monitoring: In the process of workflow, the major attention usually focuses on the operations of each step, the techniques applied, the parameters used, the simulation results, etc. However, little attention is paid to the process of workflow. The negligence of the process of workflow often leads to the difficulties of reproducing results, tracing mistakes, and so on.

To resolve these problems, a framework [Li and Molkenthin, 2014; Molkenthin et al., 2014] is proposed in this research work which locates in between the leading component and the core component as shown in Fig. 2.1. This framework is targeting at parsing the data from the leading component and further analyzing them in order to extract necessary information. This information represents the most basic unit describing the characteristics of the collected data. This unit is named MetaEvent as mentioned in Section 1.4. With these pieces of extracted information, they

can serve as the foundation of scenario composition. Once the scenarios are composed, they can be further converted into corresponding time series data for the simulation tasks in the core component. Apart from the capability for scenario composition, this framework also serves as a concept of information extraction from mass data. Besides, the framework also keeps track of every operation applied in the process as metadata for the purpose of workflow monitoring.

4.2 Concept Overview

As described earlier, the framework provides a way of generating time series data as inputs for simulation tools based on users' interests and available data. These inputs can be used for studying impacts under different scenarios of interest. In addition, the framework also monitors and stores the records of each operation applied. To describe the concept of this framework, it can be basically divided into four parts:

1. breaking time series data into representative blocks, and each block manifesting a specific characteristic of phenomenon
2. providing each block with meaningful information
3. describing relationships among time series variables
4. supporting users composing scenarios of interest

Fig. 4.1 extending from Fig. 1.2 in Section 1.4 delineates steps in each module and also the procedure of the framework. The solid lines with arrows describe how data flow and the dashed lines with arrows are optional steps where users have to examine how good the results are and take necessary actions, e.g. following the solid or dashed line.

The necessary background knowledge for the framework is already described in Chapter 3, which contains Time Series Knowledge Representation (TSKR), Time Series Knowledge Mining (TSKM), fuzzy logic and Multivariate Adaptive Regression Splines (MARS). The workflow can be divided into

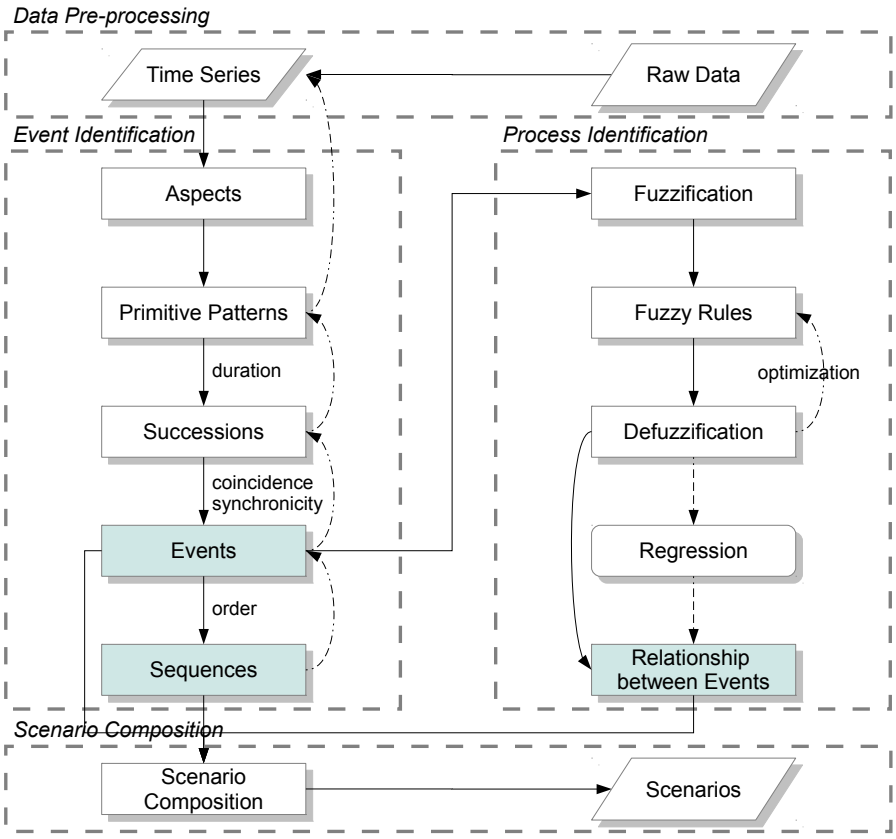


Figure 4.1: General framework concept of the scenario composition

four different modules and following the solid arrows, the workflow can be divided according to the modules into:

- Data Pre-processing: Here, the raw data are processed depending on the requirements, type of problems, etc. After these preliminary processing procedures, such as outliers removing, gaps filling, trends eliminating, etc., these processed data are then organized, depending on the data format, into required time series.

- Event Identification: The acquired time series are further transmitted for the purpose of event identification. As shown in Fig. 4.1, it includes different steps to achieve different intermediate results: Aspects, Primitive Patterns, Successions, Events, and Sequences. The terminology used in each intermediate result corresponds to the terminology used

in TSKM for better understanding of engineers, which is explained further in Section 4.3. Each solid arrow represents the sequence flow in event identification and each dashed arrow indicates an “optional” path. This means the process in event identification is not strict sequential and each intermediate result has to be checked for its validity. Once the results do not satisfy the requirements, the process has to be restarted one step, or even more steps back, due to the concept of TSKM. The final two colored intermediate results, Events and Sequences, serve as inputs for scenario composition, and Events also serve as inputs for process identification.

- Process Identification: The derived Events from event identification are passed as inputs for process identification as shown in Fig. 4.1. Like the workflow in event identification, solid and dashed arrows represent the general and optional paths in process identification together with different intermediate results/steps: Fuzzification, Fuzzy Rules, Defuzzification, Regression, and Relationship between Events. The first three illustrate the general process in a fuzzy inference system. In addition, an optional step, as shown in rounded rectangle in Fig. 4.1, Regression, is added in process identification to provide an additional optimization to improve results when information is not sufficient. At the end, the results of Relationship between Events, denoted in the colored rectangle, describing relationships among different physical state variables are derived and further transferred as inputs for scenario composition to build MetaEvents.

- Scenario Composition: To compose scenarios of interest, three different inputs are required:

  - Events and Sequences from event identification
  - Relationship between Events from process identification

  With these, MetaEvents can be built with Events as the core along with Sequences, Relationship between Events and other statistics as metadata. Once MetaEvents are built, they serve as the basic elements

for scenario composition. Once the scenarios of interest are composed, they can be further converted into corresponding time series for further tasks.

From the description above, it can be seen that the entire workflow of this framework is not an automatic process which generates results as natural consequences when necessary inputs are provided. Instead, it is a semi-automatic process which requires manual interventions on each step to determine which next action is required, as denoted in dashed arrows and the rounded rectangle. Apart from these manual interventions, the rest is automatically achieved. On the other hand, the workflow of this framework provides opportunities to improve the results with the consideration of external information, such as experts' knowledge, if needed. The following sections will bring out how the background knowledge mentioned in Chapter 3 is applied in the concepts.

4.3 Event Identification

The event identification in the framework is to identify Events, which describe representative features in a certain time interval among the entire time series data set. For instance, a hydrological data set, which contains the information of air temperature, soil moisture and precipitation, can be identified an Event as *aridity* when the air temperature is high, the soil moisture is low and the precipitation is low if the dry season occurs.

The event identification is adapted to the needs of hydroinformatics systems, and based on the concept of TSKR and TSKM proposed by [Mörchen, 2006b; Mörchen et al., 2005]. TSKM is a method designed for temporal reasoning with the representation of TSKR, and especially to overcome the shortcomings of Allen's temporal relations [Allen, 1983] for temporal knowledge mining.

However, the intention of the event identification is to identify so-called Events among the entire time series data sets which represent features of hydrological/hydrodynamical facts, as mentioned at the beginning of

this section, and TSKR and TSKM are adapted here to serve this purpose instead of temporal reasoning. The Events derived from TSKM are described linguistically and these human-readable descriptions provide users a great advantage while composing scenarios due to the transparency of the meaning. Besides, in order to prove and to demonstrate the framework concept, a prototype which carries out its idea was designed and implemented. Within the implementation of the prototype, only the necessities of TSKM which are useful to the framework, are adopted in the prototype instead of implementing every element of TSKM. The more detailed description regarding the implementation of the scenario composition framework will be further discussed in Chapter 5. Comparing the general framework concept of scenario composition (Fig. 4.1) with the framework of TSKM (Fig. 3.8), the part of pre-processing in TSKM is moved to the data pre-processing module in the framework of scenario composition, and the rest parts of the TSKM belong to the module of event identification in the framework of scenario composition. In this event identification, Aspects are the Aspects in TSKM, Primitive Patterns and Successions are the Tones in TSKM, Events are the Chords in TSKM, and Sequences are the Phrases in TSKM, as organized in Table 4.1.

Table 4.1: The mapping of the terms used in this framework onto the ones of TSKM

TSKM	The Framework
Aspect	Aspect
Tone	Primitive Pattern + Succession
Chord	Event
Phrase	Sequence

The terminology used in this framework of scenario composition, as mentioned earlier, is based on [Mörchen et al., 2005] and is the predecessor of the terminology used in TSKM. The main reason of using this terminology instead of the terminology in TSKM inside this framework of

scenario composition is its comprehensibility for engineers, hydrologists, etc. In addition, the Sequences in the framework of scenario composition are implemented differently from the Phrases in TSKM and serve different purposes. Unlike the Phrases in TSKM to reason temporal phenomena in the study of interest, the Sequences in the framework of scenario composition provide additional information of temporal relationships of Events to support users' composition of scenarios. Moreover, the concept of a MetaEvent which aggregates the information of each corresponding Event (Chord), such as statistics, is used as the basic unit for scenario composition. These will be further explained in Section 4.5.

Due to the characteristic of the nonlinear process of TSKM, the process of event identification in the framework of scenario composition is also nonlinear. The derived Events (Chords) will further append additional information to create MetaEvents as basic elements for scenario composition. However, the information they represent is the linguistic description of either itself or the relationship between themselves. This information is good for temporal reasoning, yet it does not accord with the purpose of this study — providing information for further computational simulations. Hence, these linguistic descriptions of themselves have to be converted into numbers and one approach is to use temporal average values among the intervals to represent the states of themselves. As for the conversion of the linguistic relationship between themselves, the module of process identification is introduced into the scenario composition framework for any necessary need.

4.4 Process Identification

The characteristics of the results of event identification, Events, are, as mentioned in Section 4.3, quantitative and loosely connected and contain mainly temporal-averaged information. Such information is readable and comprehensible for users and is useful for composing scenarios. However, it would be better to have a way to strongly and physically describe the relationship between variables, such as the function in mathematics $f : X \rightarrow$

Y. This function does not only provide additional information to the system, but also *facilitates the investigation of the problem*. In addition, it may be able to uncover and identify unknown physical phenomena with the help of Data-Driven Modeling (DDM) approaches, especially for large, multidisciplinary and complex problems.

In this framework concept of scenario composition, the approach of fuzzy logic is chosen for the process identification. There are several reasons for this choice, and they are mainly:

- The results of event identification are formed and described by “crisp” linguistic variables, which correspond to fuzzy variables.
- The derived Events from event identification, which are Chords in TSKM, can also be considered as fuzzy rules in fuzzy logic [Gronz and Casper, 2008; Gronz et al., 2008].
- Comparing to some black-box approaches in DDM, the rules, which describe the system behavior, are more explicit and more interpretable. In this way, this plain description of the system behavior also affords users to better understand the system and probably to identify unknown phenomena.

In addition, here the Mamdani-type Fuzzy Inference System (FIS) is chosen for the purpose of process identification instead of the Sugeno-type one. Besides the fitting of the structure of Chords into fuzzy rules, Mamdani-type fuzzy inference system also offers better interpretation as mentioned in [Gorzałczany, 2002]. The rules, which describe the physical phenomena, are derived from the events in the event identification module. In addition, expert knowledge and rules of thumb can also be added into the set of rules if needed. Furthermore, a mechanism of optimization on membership functions or weighting factors of rules to ensure the reliability of the derived parameters inside fuzzy rules is also introduced in the framework, if necessary.

However, there still exist many situations which cause the unreliability of the derived rules, such as the parameters introduced in the semi-automatic

event identification process. To resolve this issue for a better description of this system behavior, the technique of MARS [Friedman, 1991] is added in this module as a second mapping function.

The regression technique MARS is selected to act as an optional fine-tuning of the results of FIS a step earlier. Although there is no regression technique best-fitted for all situations, the main two reasons why MARS is chosen in this framework are:

- Applicability: Unlike some other regression techniques, such as statistical regressions, applying the nonparametric MARS does not need any assumption beforehand as mentioned earlier. This feature fits in the need of the framework, since the characteristics of the results of FIS are unknown in advance. Hence, no statistical hypothesis testing is needed.

- Interpretability: The representation of MARS model consists of the linear combination of basis functions, and it is easier to interpret comparing to other techniques, such as Artificial Neural Network (ANN).

In this section, the concepts and reasons of how and why two different approaches, fuzzy logic and MARS, are applied in the module of process identification in the framework, are described. Mamdani-type FIS bridges the crisp linguistic results of event identification by providing meaningful descriptions among variables. MARS, on the other hand, fine-tunes the results from the FIS to ensure better descriptions among variables. The results of this process identification serve as additional information to supplement the results of the event identification.

4.5 Scenario Composition

The aim of scenario composition is to provide a manner to compose synthetic scenarios of interest and, then, to generate time series data based on these user-defined scenarios. The generated time series data can be later used

for different tasks, such as Boundary Conditions (BCs) inputs for different simulation tools to investigate the impacts under these scenarios. Moreover, the scenario composition module together with other modules can also be integrated with other hydroinformatics systems towards a holistic modeling approach.

To achieve the objectives mentioned above, two major elements have to be offered in scenario composition for composing and generating the corresponding time series data:

- information
- interface

The information comes from the results of previous modules. These results will be aggregated and delivered to form the most elementary element, MetaEvent, of scenario composition. As previously stated in Section 1.4, the role of a MetaEvent in scenario composition is as a LEGO® brick — the most basic entity that a scenario can be built upon. A MetaEvent is composed of one corresponding Event from event identification along with some additional information, such as statistics for each time series which makes up this Event. Hence, a MetaEvent can be considered as its corresponding Event with related metadata.

At this stage, a number of MetaEvents are available for users to compose their scenarios of interest, and the number of MetaEvents depends on the choices of methods, parameters, personal decisions, etc. in previous steps. With these MetaEvents at hand, users need a “guide” to know *what* the MetaEvents represent and *how* they can be used, and metadata suffice these demands. For instance, the statistics of a MetaEvent provide information, such as frequency, maximum duration, minimum duration, average value, etc., of each time series based on the collected data for the request of *what*, and the suffix tree data structure offers the information of the next coming MetaEvents to answer the demand of *how*. Here, there is no restriction on the composition of scenarios; instead, it provides a “guide” with supplementary information. On the other hand, the “guide” provides

semantics of MetaEvents illustrating the corresponding natural events, and helps users to compose a reasonable scenario of interest based on the collected data instead of any random, arbitrary one.

The interface to compose scenarios can be arbitrary as long as it serves the purpose. The interface should, a least, offer some basic functionalities, such as:

- reviewing the composed scenario
- displaying necessary information of metadata
- generating corresponding time series data

In addition, the generated time series data can be further processed, for instance, by downscaling for higher temporal resolution requirements, if necessary. In the prototype design and implementation, a simple Graphical User Interface (GUI), illustrated further in Section 5.6, is created to provide these basic functionalities.

Chapter 5

Framework Prototype Design and Implementation

5.1 Implementation Environment Options

In this chapter, the prototype design and implementation of the framework is introduced. However, at least four aspects have to be taken into consideration beforehand:

Operating Systems (OSs): An OS, in general, is a set of software programs which bridge users and hardware. From the users' side, it provides a user interface, either a Graphical User Interface (GUI) or a Text-based User Interface (TUI), for users to have access to or control over hardware devices without explicit knowledge of how tasks are performed. From the side of hardware, the OS takes care of every single command given by users which involves information exchange among software programs themselves and between software and hardware, such as controlling inputs and outputs, memory management, accessing disks, etc.

Current OSs can be categorized into three major groups:

- Microsoft Windows OSs: The family of Microsoft Windows is a proprietary OS developed by Microsoft Corporation. It currently

dominates the market of the world's personal computers and the number of the share is 91.82% in February, 2013 according to [Net Applications].

- Unix-like OSs: The development of Unix-like OSs can date back to the project of developing the Multiplexed Information and Computing Service (Multics) for the GE-645 mainframe under the collaboration between Massachusetts Institute of Technology (MIT), General Electric (GE) and AT&T Bell Laboratories in 1964 [Multics; Stuart, 2008]. Due to the original design targeting at the mainframe, the Unix-like OSs have some features, such as multitasking, multi-user, etc., by nature. A variety of OSs belong to Unix-like OSs, like the BSD family (FreeBSD, NetBSD, Sun OS, Mac OS X, iOS, etc.), Minix, Linux, Solaris, HP/UX, Google Chromium OS, WebOS, Android, Firefox OS, Sailfish, etc., and they are all compatible with the Single UNIX Specification (SUS) standard, including Portable Operating System Interface (POSIX).
- Other OSs: Apart from the above two categories, some OSs still exist, which do not belong to neither of them. They are usually more platform-specific or for embedded systems, such as Palm OS, BeOS, Mac OS, OS/2, z/OS, etc.

The choice of an OS also depends on the type of hardware platforms, which will also be discussed later. For instance, some OSs are targeting at mobile/handheld devices, such as iOS, WebOS, Android, Firefox OS, Sailfish, Palm OS, etc. They are usually more lightweight and use web technologies, e.g. HyperText Markup Language (HTML) and JavaScript, for the development of the applications. Although traditional Personal Computers (PCs), such as desktops and laptops, still have their places in the consumer market, the current trend is going to the direction of mobile/handheld devices in combination with servers, e.g. cloud technologies. According to the report in Oct. 2013 from Gartner [Gartner Inc.], the shipments of smartphones and tablets are expected to grow 3.7% and 53.4% in year 2013 respectively, but

the shipments of traditional PCs are expected to drop 11.2% in year 2013. From the trend observed above, the OSs designed for such mobile/handheld devices start to play a more important role in the consumer market.

Programming Languages: A programming language is an artificial language which humans communicate with machines and instruct machines what to do. To categorize different programming languages, there are many aspects, such as the level, the purpose, the paradigm, etc. In the following, there are some definitions and categories regarding programming languages:

- High-level and low-level programming languages: The difference between high-level and low-level programming languages lies in the abstraction from the computer's instruction set. A high-level programming language uses language patterns closer to natural languages with semantics and it is more understandable by humans. Besides, it is also independent of the architecture of hardware. Instead, a low-level programming language is hardware-specific and the written codes can be executed directly by the machine. In other words, to execute the codes written in high-level programming languages, a "translation", e.g. interpreting, compiling, etc., to low-level programming languages is necessary. Here are some examples of high-level and low-level programming languages:
  - High-level programming languages: C, C++, Java, Fortran, R, MATLAB, Python, etc.
  - Low-level programming languages: machine languages and assembly languages with different assemblers, e.g. GNU Assembler (GAS) for multiple platforms, Microsoft Macro Assembler (MASM) for Microsoft Windows and MS-DOS, etc.
- General-purpose and domain-specific programming languages: As their names stand, the major difference between the General-

Purpose Language (GPL) and the Domain-Specific Language (DSL) is the purpose they are designed for. A GPL is designed for dealing with all types of problems; a DSL, on the contrary, is intended to solve problems inside specific domains. Here are some examples of programming languages belonging to GPL and DSL:

  - General-purpose programming languages: C, C++, Java, Fortran, Lisp, Python, etc.
  - Domain-specific programming languages: MATLAB, Octave, R, SAS, Mathematica, Structured Query Language (SQL), Object Query Language (OQL), etc.

- Compiled and scripting programming languages: One way to distinguish between compiled and scripting programming languages is the scriptability. Scripting programming languages reduce the traditional edit-compile-link-run process of compiled programming languages and they are interpreted line by line instead of precompiled in advance. Hence, they gain the advantage of flexibility and convenience over compiled programming languages, but less performance compared to their counterparts. In the following, there are some compiled and scripting programming languages:
  - Compiled programming languages: C, Java, C++, Pascal, Fortran, Visual Basic, etc.
  - Scripting programming languages: Python, Ruby, Perl, PHP, R, Octave, MATLAB, JavaScript, etc.
- Procedural, functional, and object-oriented programming languages: The terms of “procedural”, “functional”, and “object-oriented” describe three different programming paradigms among other paradigms and a programming paradigm defines how a program abstracts and describes problems. Procedural programming defines the sequence of steps computers have to perform. In procedural programming, a procedure is usually known as a *subroutine* or a *function* and can be called at anytime needed. Functional programming, on the other hand, sees the computation

as the evaluation of mathematical functions and is based on lambda calculus. Unlike procedural programming, which changes the state during the execution, functional programming avoids using mutable data. As for Object-Oriented Programming (OOP), it focuses on mapping the real world into the digital one, and each item in the real world can be represented by an *object*, which encapsulates all necessary information, e.g. the state and the behavior, describing itself. Since a type of programming paradigms is only a way to describe problems, most of the programming languages support multi-paradigm design. For example, C++ and lisp can be procedural, functional, or object-oriented; Java also supports multi-paradigm design, and can be either procedural or object-oriented.

A brief summary of programming languages mentioned above is shown in Table 5.1. Looking back into the history of hydroinformatics, the programming languages traditionally used for the implementation of these tools were mainly high-level, compiled and procedural languages, such as Fortran in the core parts of HEC-RAS [HEC-RAS], TELEMAC-MASCARET [open TELEMAC-MASCARET], MIKE family [DHI], etc. The main reason can be attributed to the history of Information and Communication Technology (ICT), for instance, the motive of the invention of computers, the performance of hardware, etc. However, due to the development of programming languages and the improvement of hardware, modern programming languages offer more versatility and convenience in design and implementation without suffering the performance issue in most descent-sized projects. Particularly, scripting languages offer more flexible and nimble supports, such as metaprogramming, in design and implementation. These do not only help the implementation in new projects but also facilitate the extension of existing projects. For example, ArcGIS [Esri] supports both Python and R scripting languages if external requirements are needed.

Hardware Platforms: A hardware platform is a physical carrier which

Table 5.1: Brief summary of programming languages

Languages	High-level	Low-level	General-purpose	Domain-specific	Compiled	Scripting	Procedural	Functional	Object-oriented
assembly	-	+	+	-	-	-	-	-	-
C	+	-	+	-	+	-	+	-	-
C++	+	-	+	-	+	-	+	+	+
Fortran	+	-	+	-	+	-	+	-	+[a]
Java	+	-	+	-	+[b]	-	-	-	+
JavaScript	+	-	+	-	-	+	-	+	+
Lisp	+	-	+	-	+	+[c]	+	+	+
Mathematica	+	-	-	+	-	+	+	+	+
MATLAB	+	-	-	+	-	+	+	+	+
Octave	+	-	-	+	-	+	+	+	+
OQL	+	-	-	+	-	+	-	-	-
Pascal	+	-	+	-	+	+	+	-	+[d]
Perl	+	-	+	-	-	+	+	+	+
PHP	+	-	+	-	-	+	+	+	+
Python	+	-	+	-	-	+	+	+	+
R	+	-	-	+	-	+	+	+	+
Ruby	+	-	+	-	-	+	+	+	+
SAS	+	-	-	+	-	+	+	-	+
SQL	+	-	-	+	-	+	-	-	-
Visual Basic	+	-	+	-	+	-	+	-	+[e]

[a] Fortran 2003 supports object-oriented programming paradigm.
[b] Java codes are compiled into bytecode.
[c] Some dialects, such as Emacs Lisp, are considered as scripting languages.
[d] Object Pascal supports object-oriented programming paradigm.
[e] Visual Basic .Net supports object-oriented programming paradigm.

accepts commands from users through input devices, performs computational tasks, and displays results with the help of output devices. As mentioned earlier, the choice of OSs also depends on the choice of hardware platforms. In addition, this choice also further affects the choice of the design and the implementation, such as programming languages, technologies, etc. Here, the types of hardware platforms are generally grouped depending on their physical sizes and computational performance. The groups are mainframe computers, supercomputers, workstations, personal computers, mobile devices, and others.

- Mainframe computers: Mainframe computers usually refer to the "big" computers located in a cabinet and are accessed through terminals by multiple users at the same time. Compared to normal commercial computers, they are more powerful, more reliable, more secure, having better input and output capacity, and taking more care in the compatibility with order software, etc. Mainframe computers are usually used to undertake the tasks, such as bulk data processing, which ordinary computers are less appropriate for. In addition, mainframe computers are usually running on platform specific OSs, such as z/OS, and Unix-like OSs (72% of z/OS and 28% of Unix-like OSs running for IBM System z[1] in Dec. 2008 [Wikipedia, the free encyclopedia]).
- Supercomputers: Compared to mainframe computers which focus more on stability, supercomputers center more on the performance and deal with the tasks which computational speed is of great importance, such as high performance computing. Due to the high performance in calculation, they are widely used in scientific or engineering fields, especially for large scale problems, such as weather forecasting. Besides, Unix-like OSs are usually their OSs of choice (98% of the market share among the top 500 supercomputers in Nov. 2012 [TOP500]).

[1] IBM System z is a family name of IBM's mainframes and this series is chosen by nearly 95% of Fortune 1000 companies as their mainframes [Computer & Communications Industry Association, 2008].

- Workstations: A workstation can be regarded as a high-end personal computer and is usually equipped with faster Central Processing Unit (CPU), larger capacity of Random-Access Memory (RAM), and better and bigger graphic display when compared to a standard PC. Moreover, the quality of hardware in workstations is usually more reliable than the one in personal computers. Thus, a workstation is usually used for the tasks which require a relative amount of computing power and a higher resolution display, such as the applications in Geographic Information System (GIS), Computer-Aided Design (CAD), Computer-Aided Manufacturing (CAM), numerical simulations, Graphics Processing Unit (GPU), 3D and Virtual Reality (VR) visualization. In addition, a workstation is also sometimes used as a mini server. As for OSs deployed in workstations, they depend on the purposes of the workstations, and Microsoft Windows and Unix-like OSs are usually the choices.
- Personal computers: Compared to the computers mentioned above, PCs are less powerful, more affordable, and designed for use by an individual user at a time. Depending on the size and the performance, PCs can be desktops, laptops, netbooks, etc., and most of them are running with Microsoft Windows as OSs (91.82% of global market share in Feb. 2013 [Net Applications]).
- Mobile devices: Mobile devices are smaller computing devices compared to normal standard-sized laptops, and target mainly at portability and mobility. They are also equipped with sufficient computing power for everyday tasks, such as documents editing/viewing, Internet browsing, etc. These devices include Personal Digital Assistant (PDA), smartphones, tablets, etc., and are mainly running with Unix-like OSs (81.53% of global market share in Feb. 2013 [Net Applications]), such as Android, iOS, etc.
- Others: Some other hardware platforms still exist, such as embedded systems, Single-Board Computers (SBCs), etc. They are mostly designed less powerful compared to PCs, e.g. with a less

powerful CPU, slower memory, smaller capacity of storage, etc., and are usually cheaper than PCs. They generally serve to perform specific tasks with limited functionalities, such as, Network-Attached Storages (NASs), private web servers, ticket vending machines, etc., or act as experimental prototypes, for instance, the Raspberry Pi. The OSs running on these kinds of platforms are usually either platform-specific or Unix-like-based. These types of hardware platforms are outside the scope of discussion in this research work.

Yet, the boundaries among such categories are getting obscurer due to the rapid development of ICT. A current high-end mobile device can be more powerful than a descent 10-year-old laptop. A powerful NAS can not only serve as a file server, but also, for instance, as a personal cloud service, a private web server, a media streaming service, etc. In addition, the current trend is in the direction towards mobile computing, and it can be observed from the report of Gartner [Gartner Inc.], as pointed out earlier.

Networks: The invention of the network can trace back to the development of the Advanced Research Projects Agency Network (ARPANET) in the 1960s. Afterwards, with the standardization of the Internet protocol suite, TCP/IP, defining layers and protocols of how different computers exchange information through networks in the 1980s, it became the cornerstone of today's network. Later, the Internet and World Wide Web (WWW) started to develop and blossom after the first web server was set up by Tim Berners-Lee and others at the European Organization for Nuclear Research (CERN) in the beginning of 1990s.

The network can be generally categorized into two types depending on the scope computers connect to each other:

- Local area network: The scope of a Local Area Network (LAN) is usually limited to a specific range, such as offices, schools, universities, etc., and the communication standards are defined in

the IEEE 802 family. For instance, the current two most common standards for connecting computers are:

  - Ethernet for the wired environment with a speed from 10 Mbit/s to 100 Gbit/s
  - Wi-Fi for the wireless environment with a speed from 1Mbit/s to 6.75 Gbit/s

  They are defined in IEEE 802.3 and IEEE 802.11 respectively.

- Wide area network: In contrast to the LAN, the scope of the Wide Area Network (WAN) is broader, and Internet can be regarded as a type of WAN connecting different LANs in the world through the TCP/IP protocol suite as mentioned earlier.

Apart from these standards of the IEEE 802 family, there exist some other different standards, such as Universal Mobile Telecommunications System (UMTS) and Long-Term Evolution (LTE), which provide protocols for different mobile devices to communicate in the network. There are many services which can run over networks, such as WWW, e-mail, file sharing, Instant Messaging (IM), Voice over Internet Protocol (VoIP), etc. To enable these services, the application layer of TCP/IP defines different protocols for them. Here are some major protocols for different common services:

- WWW: Hypertext Transfer Protocol (HTTP), Hypertext Transfer Protocol Secure (HTTPS)
- File sharing: File Transfer Protocol (FTP), Server Message Block (SMB)
- E-mail: Simple Mail Transfer Protocol (SMTP), Post Office Protocol (POP), Internet Message Access Protocol (IMAP)
- Instant messaging: Extensible Messaging and Presence Protocol (XMPP), Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions (SIMPLE)
- VoIP: H.323, Session Initiation Protocol (SIP)

- Security over the Internet: Transport Layer Security (TLS), Secure Sockets Layer (SSL), Secure Shell (SSH)

The abovementioned four options — OSs, programming languages, hardware platforms, and networks, will determine how the prototype is designed and implemented. In the following section, the criteria and the decisions for the design and implementation of the framework prototype will be discussed.

5.2 Prototype Implementation Criteria and Environments

As discussed in Section 5.1, the boundaries among the choices of OSs, programming languages, and hardware platforms, are becoming less clear due to the rapid development of technology. For instance, programming languages can support multi-paradigm designs, and some personal computers have almost equal computational performance as workstations. Hence, the choices of implementation environments are most of the time neutral and based on personal preferences as well as the current working environment. However, some criteria still affect the choice of how the prototype is implemented. In this research work, the criteria for the prototype implementation environments are based on:

- the current working environments, including the OSs, the working programming languages, and the hardware platforms
- the support of adequate resources, such as documents, libraries, etc. for programming languages
- the simplicity of the working environments in terms of software and hardware technology

In addition to the implementation environments, some principles of the prototype design philosophy are:

- the idea of “a bird in the hand is worth two in the bush”, which emphasizes the realization of the basic concept of the framework instead of a complete software product
- the priority of applicability and transparency coming before that of performance
- the possibility for further extensions
- the adaption with other software technologies

According to the criteria above, the prototype is implemented under such environments and conditions:

- on Linux OS
- in Java and R programming languages
- on a single PC-based computer with an AMD Athlon 64 x2 4800+ CPU and 4 GB RAM
- without network connection, but offering opportunities for further extensions, such as, cloud computing through TCP/IP protocol with the number crunching service on the server, etc.
- the information exchange mostly through well-structured plain text file format

The chosen environments for the prototype implementation come from several circumstances:

- the current working environments: Linux OS, Java programming language, and the single PC-based computer
- the support of adequate resources, such as documents, libraries, etc., for data processing: R programming language
- the simplicity of the developing environments: no network connection

However, Java and R are two different programming languages with different design philosophies and programming environments. In order to exchange information between these two programming environments, additional settings, such as, environment parameters, are necessary. The following is the brief introduction of these two programming languages and how they are applied in the prototype implementation:

Java programming language: Java is a general-purpose programming language under GPL and targeting mainly at the object-oriented paradigm. It was originally developed by James Gosling, Mike Sheridan, and Patrick Naughton at Sun Microsystems in 1991 [Byous, 1998]. One of its features is “write once, run anywhere”, which is done by compiling Java codes into bytecodes running on top of the Java Virtual Machine (JVM). JVM provides an abstract layer between Java applications and the latent platform, and interprets the compiled bytecodes. In this manner, the applications written in Java are platform independent. Therefore, the choice of OS and hardware platform becomes trivial to consider. In addition, Java has a wide range of support in all kinds of application fields due to its maturity.

In the prototype implementation, the Java programming language is applied in the implementation of the event identification, the process identification, and the scenario composition. The developing activities are carried out in the Eclipse Integrated Development Environment (IDE) [Eclipse Foundation] based on Java 5, and several major applied external libraries include:

- Apache Commons FileUpload [Apache Software Foundation: Commons FileUpload]
- Apache Commons Math [Apache Software Foundation: Commons Math]
- Apache POI [Apache Software Foundation: POI]
- Apache Tomcat [Apache Software Foundation: Tomcat]
- Guava [Google Guava]

- JFreeChart [JFree.org]
- jFuzzyLogic [Cingolani and Alcala-Fdez, 2012]
- Joda-Time [Joda.org]
- Java/R Interface (JRI) [RForge.net]
- Suffix Tree Implementation [Havsiyevych]
- Turtle [Molkenthin et al., 2009]

R programming language: Unlike Java, R is a domain-specific scripting language and particularly focusing on procedural and functional paradigms with a certain support of OOP paradigm. R was created by Ross Ihaka and Robert Gentleman in 1993 [Ihaka and Gentleman, 1996]. Besides being a computer programming language, R also offers a Read-Eval-Print Loop (REPL) software environment for interactive operations. In addition, R is mainly designed for statistical computing and visualization, which is different from Java being a generalized programming language, and is popular among statisticians and data miners with the support of abundant packages. Moreover, being a free software environment under GPL, it can also run on different OSs.

The parts of the prototype implementation using R is under the Emacs text editor together with the package Emacs Speaks Statistics (ESS) [ESS]. Several major R packages used in the prototype implementation, apart from other optional ones which depend on the types of the problems, involve:

- `ggplot2` [Wickham, 2009]
- `gridExtra` [Auguie, 2012]
- `mclust` [Fraley and Raftery, 2007]
- `reshape` [Wickham, 2007]
- `rJava` [Urbanek, 2011]
- `scales` [Wickham, 2012]
- `zoo` [Zeileis and Grothendieck, 2005]

The implementation of the prototype is carried out in either a single programming language or in mixed programming languages (Java and R):

- Data Pre-processing: R
- Event Identification: Java and R
- Process Identification: Java
- Scenario Composition: Java

The choice of being implemented in a single programming language or in mixed programming languages depends on the requirements of each module, and the implementation tries to take advantages from both sides if mixed-language programming style is adopted.

Since the implementations of different modules are accomplished in different languages depending on the feature of the language itself and the requirements of the module, the information exchange between different modules has to be decided. As mentioned earlier in the principles of the prototype design philosophy, transparency comes before the performance. Therefore, the exchange of information among different steps and modules is carried out through files in this implementation prototype. These files are mainly in the form of plain text files, for instance, spreadsheet-like format for time series data, and detailed formats will be illustrated in the later descriptions of different modules. However, a part of information exchange is done through the binary file format, which is about the complete results of the Event Identification. This is because of the complex hierarchy of the data structure and it does not help the transparency of the process. Although information communication through files is not computationally efficient and may not be favorable to some situations, it provides a pause to review the results and further has an opportunity to ensure the quality of the results before stepping to the next step. Furthermore, the framework itself is a semi-automatic process, where the results of each step are of great consequence to the subsequent steps, and a mechanism to assure the quality of results is necessary.

The prototype is implemented, as previously stated, in two heterogeneous programming languages — Java and R. In order to exchange information between Java and R environments at runtime, especially for the module implemented in both programming languages, Java Native Interface (JNI) comes to the rescue. JNI, as its name stands, is a programming interface which enables Java codes running on JVM to communicate with native applications implemented in other programming languages than Java. The schematic illustration of the concept of JNI is shown in Fig. 5.1. There, two applications, a Java-based and a non-Java based, are exchanging information through the help of JNI. First, the Java data inputs are converted to non-Java data inputs, being the native data structure of the non-Java environment, through JNI. Second, the calculated results in the non-Java environment, non-Java data outputs, are converted back to Java data outputs over JNI. In the prototype, the calculated statistical or data mining results performed in the R environment can be passed to the Java environment for further operations, and the choice of algorithms or the arguments needed for the calculation in the R environment can be assigned directly in the Java code.

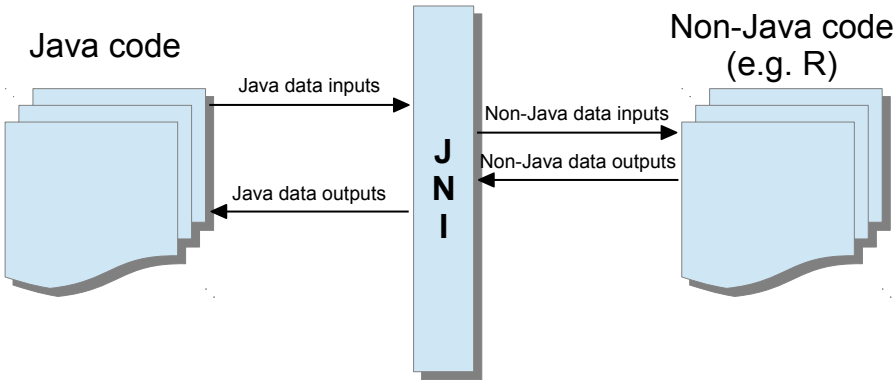


Figure 5.1: Schematic concept of the JNI

To connect both R and Java platforms, several approaches exist, such as:

- `SJava` package [Temple Lang and Chambers, 2013]: A contribution from the Omegahat project [Omegahat] using JNI to communicate between R and Java environments.

- `rJava` package [Urbanek, 2011]: Like the `SJava` package, it also exchanges information between R and Java through JNI. Unlike the `SJava` package supported by a third-party project, the `rJava` package is maintained by the R project directly and can be found in the official package repository, Comprehensive R Archive Network (CRAN). In addition to being widely deployed by different projects, it can be easily extended to a client-server model by using the `RServe` package described below. This is because both `rJava` and `RServe` packages share identical methods in exchanging information from outside the JVM environment. This becomes an advantage in the implementation because the extension of the application can be achieved without much effort.

- `RServe` package [Urbanek, 2012]: Unlike the previous two packages that must exchange information locally, this package communicates information through TCP/IP protocol. Therefore, a client-server model can be deployed and users can run the calculation in the cloud.

In this implementation, the package `rJava` is chosen simply because of:

- its number of available applications which implies an available amount of documentation
- the simplification of the prototype implementation within a single PC
- the extensibility of future growth to online services, for instance, cloud computing

In addition, several environmental parameters have to be set up to be able to execute it. In the current implementation environment, these environmental parameters are as follows:

```
R_HOME=/usr/lib64/R
R_SHARE_DIR=/usr/share/R
R_INCLUDE_DIR=/usr/include/R
R_DOC_DIR=/usr/share/doc/R
LD_LIBRARY_PATH=/usr/lib:/usr/lib64/R/lib:/usr/lib64/R/bin
```

Besides, the JVM argument `java.library.path` has to be set to where the `libjri.so` (for Unix-like OS) or the `libjri.dll` (for Windows OS) locates.

5.3 Data Pre-processing Design and Implementation

As stated in Section 3.3, the tasks of data pre-processing are problem- and domain-specific. Hence, rather than implementing a limited number of methods for it, it is more appropriate to have a thorough solution for this purpose. In addition, one principle of the prototype design philosohpy is to realize the basic concept of the framework as mentioned in Section 5.2. For these reasons, the full-featured R software environment is adopted in the prototype, instead of "reinventing the wheel". Besides, the R REPL environment and the preferred editing environment, Emacs and the ESS package, are used as interfaces for operations. The main reason is that the R software environment has already supported a wide variety of features, from basic data handling to advanced time series analysis.

The entire R software environment offers a wide range of tools for data manipulation and analysis. Apart from what has been mentioned in Section 5.2 for general implementation in this prototype, there exist some other packages specific for data pre-processing depending on type of problems, such as:

- missing data handling: R standard packages (e.g. `base`, `stats`, etc.), R package `mitools` [Lumley, 2012], etc.
- seasonal decomposition: R standard packages (e.g. `stats`), etc.
- Autoregressive-Moving-Average (ARMA) model: R standard packages (e.g. `stats`), etc.
- Principal Component Analysis (PCA): R standard packages (e.g. `stats`), etc.

- Support Vector Machine (SVM): R package `kernlab` [Karatzoglou et al., 2004], R package `e1071` [Meyer et al., 2012], etc.
- Self-Organizing Map (SOM): R package `som` [Yan, 2010], R package `kohnen` [Wehrens and Buydens, 2007], etc.

The R software environment provides a general environment to read from and write to different data sources, from plain text to binary files and even to different Relational Database Management Systems (RDBMSs), such as Comma-Separated Values (CSV) file format, Microsoft Excel Spreadsheet (XLS) file format, MySQL, Oracle database, etc. However, importing from and exporting to the CSV file format is the most common approach also used here, and the file format looks like:

```
Date, Rainfall (mm), Temperature (Celsius), Discharge (m^3/s)
1974-05-18,  2.1, 19.0, 0.0
1974-05-19,  0.0, 19.3, 0.0
1974-05-20,  0.0, 17.5, 0.0
1974-05-21,  0.0, 18.9, 0.0
1974-05-22,  0.0, 20.1, 0.0
1974-05-23,  0.0, 19.5, 0.0
1974-05-24, 28.5, 20.0, 0.1
1974-05-25, 18.1, 19.7, 0.7
```

In the example above, it contains some basic information describing data, which are the date, the type of data, and the unit of data. The reasons why the CSV file format is adopted here are mainly its versatility to be read by different software applications, including plotting applications, and the readability by human beings. In this manner, it is easier to have an overview of the raw data. After an adequate pre-processing, the input data sets are grouped into a pack of distinct Aspects which have different semantic representations, contain variant characteristics, etc., and these Aspects are saved into a CSV file with the same format described above for the following steps.

5.4 Event Identification Design and Implementation

As mentioned in previous chapters, the purpose of event identification is to identify the so-called Event, which is made up of the coincidence of several time states, representing a certain feature of the phenomenon of interest. Although the event identification is mainly based on Time Series Knowledge Mining (TSKM), the framework of TSKM is not completely implemented in this prototype. This is mainly because the purpose of this module of event identification is to extract features of time series sets, as the concept of itemsets, instead of the temporal reasoning as TSKM does. However, the two basic elements, Tone and Chord, corresponding to the naming convention in Time Series Knowledge Representation (TSKR) as described in Section 3.2, are taken out from the TSKM and redesigned in an OOP way in Java as shown in both Fig. 5.2 and Fig. 5.3 respectively.

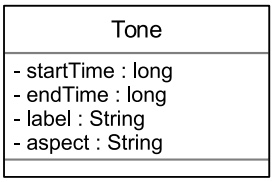


Figure 5.2: Design of the class Tone in the event identification

The design of class Tone (see Fig. 5.2) describes the concept of Tones in TSKR, as mentioned in Section 3.2. A Tone comprises three components: the start time, the end time, and the symbolic value, as described in Section 3.2. As shown in Fig. 5.2, it contains attributes defining the start time (`startTime`), the end time (`endTime`), and the symbolic value (`label`) to fulfill the definition of a Tone. In addition, it adds additional information (`aspect`) to specify which time series it belongs to, and that is the Aspect according to TSKM.

As described in Section 3.2, a Chord represents coincidence, and it is built by one Tone of each Aspect. Fig. 5.3 illustrates the basic design of

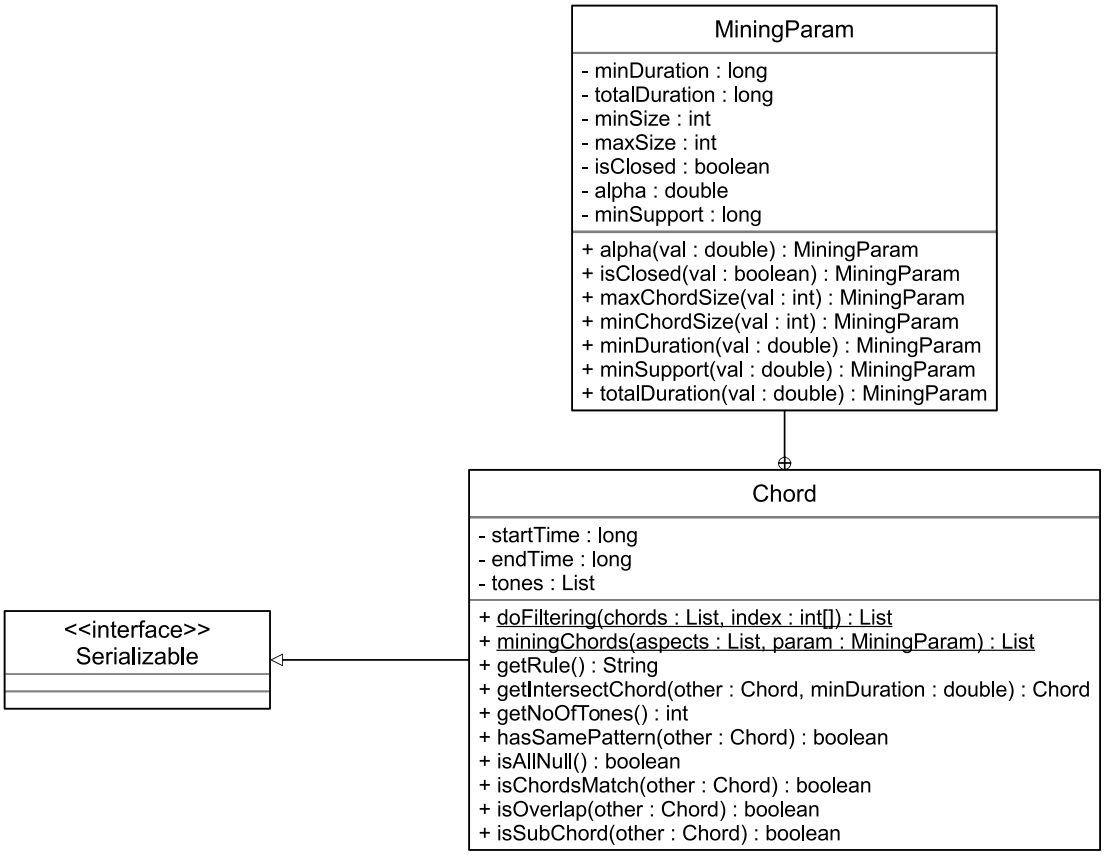


Figure 5.3: Design of the class Chord in the event identification

the Chord defined in TSKR. Since a Chord is made of Tones, it contains the information of Tones which belong to it (`tones`) together with the start time (`startTime`) and the end time (`endTime`) of itself. To derive a list of Chords from a set of Aspects, the static method `miningChords(aspects:List, param:MiningParam)`, which will be described below, is utilized.

Apart from these attributes, it also has some methods, such as:

- `doFiltering(chords:List,index:int[])`: It is a static method, which filters out the Chords with the same pattern. In addition, once the index is given, the Chords will be removed if the Tone with given index is empty. This is useful in generating fuzzy rules in process identification discussed in Section 5.5.

- `miningChords(aspects:List,param:MiningParam)`: It is a static

method, which mines Chords based on the modified Closed Association Rule Mining (CHARM) algorithm according to TSKM, and the parameters for mining are defined in the inner class MiningParam.

- `getRule()`: It is an instance method describing the composition of the Chord.
- `isOverlap(other:Chord)`: It is an instance method, which determines if two Chords are overlapping.
- `isSubChord(other:Chord)`: It is an instance method checking if the Chord is the sub-Chord of the other one.

However, the flexibility of providing arbitrary semantic descriptions by users for each Tone and Chord are not implemented here. Instead, these descriptions are defined by the given name of the Aspect and the results of the algorithm with a fixed and consistent format throughout the entire workflow.

In addition, in order to facilitate setting parameters for the mining algorithm, the builder pattern[2] is adopted here. As seen in Fig. 5.3, the inner class MiningParam is used to provide parameters for the mining algorithm. With this design, users can assign parameters for the mining algorithm with ease. An example of assigning mining parameters is given as:

[2]The builder pattern is one of the design patterns in software engineering. The design patterns are optimized and reusable solutions for the problems of software design encountered in the daily basis and are often employed in OOP.

```
/*
 * variables:
 *  data        : a set of Aspects composed of Tones to
 *                be mined
 *  miningResult: a list of mined Chords
 */
private static final long MIN = 60000L; // 1 min = 60000 ms
miningResult = Chord.miningChords(data, new MiningParam()
                                        .alpha(0.0)
                                        .minChordSize(3)
                                        .minDuration(30*MIN));
```

where the mining parameter alpha[3] is set to 0.0, the minimum size of a Chord is set to 3, the minimum duration is set to 30 minutes. As for the rest unmodified parameters remain the same with default settings.

As for the approach of finding Tones, the design of the architecture is as shown in Fig. 5.4. There, four classes are shown, which are class AbsBinGen, class AbsRBin, class EM, and class RKMeans. class AbsBinGen and class AbsRBin are abstract classes, and class EM and class RKMeans are concrete classes which implement different algorithms for finding Tones. The class AbsBinGen defines the basic method needed for finding Tones. The class AbsRBin defines an abstract method which describes how to utilize the R environment to implement algorithms. Finally, two inherited classes, class EM and class RKMeans, implement specified algorithms, the Expectation Maximization (EM) clustering algorithm in the R package `mclust` [Fraley and Raftery, 2007] and the k-means clustering algorithm, respectively, and those abstract methods. With this design, the approach of finding Tones can be further extended to other R-based algorithms, Java-based algorithms or other algorithms on different computing platforms if required.

Since these Java classes use the algorithms implemented in the R environment, there exists a way to exchange information between the JVM and the R environments. As described in Section 5.2, the `rJava` package is adopted in

[3]alpha is the α in Eq. 3.1 which is the threshold determining margin-closeness

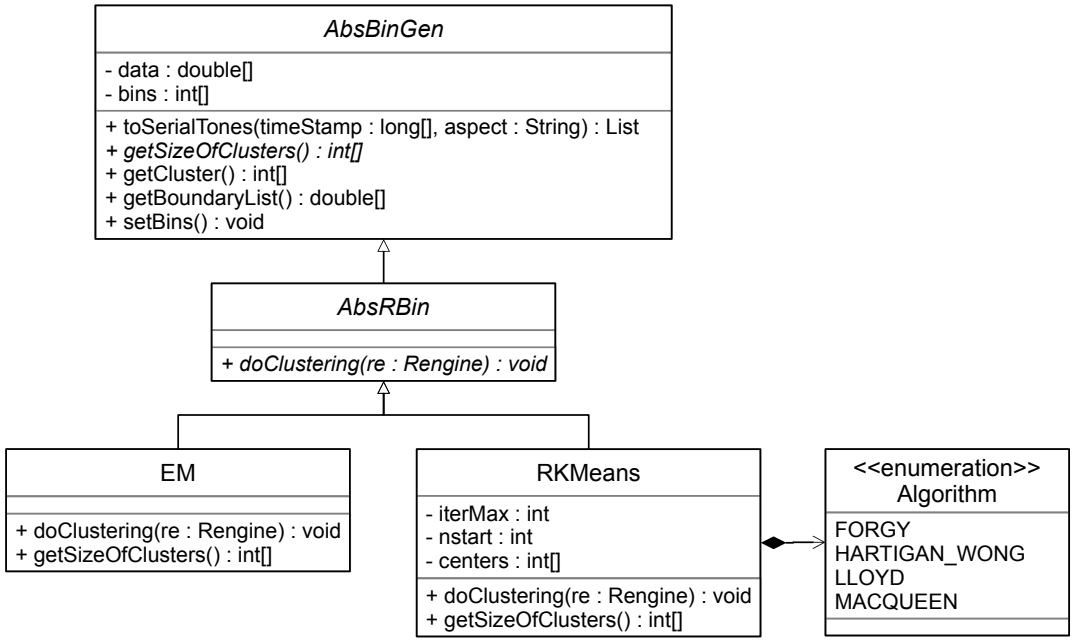


Figure 5.4: Design of finding Tones in the event identification

this implementation. Fig. 5.5 illustrates how R codes communicate with Java codes with the help of the `rJava` package. The `rJava` package contains two components, rJava and JRI. The component rJava allows to do operations in JVM from R, such as accessing fields of Java objects. JRI, on the contrary, allows R code to run in JVM as a single thread. With the help of both components, information can be exchanged bidirectionally.

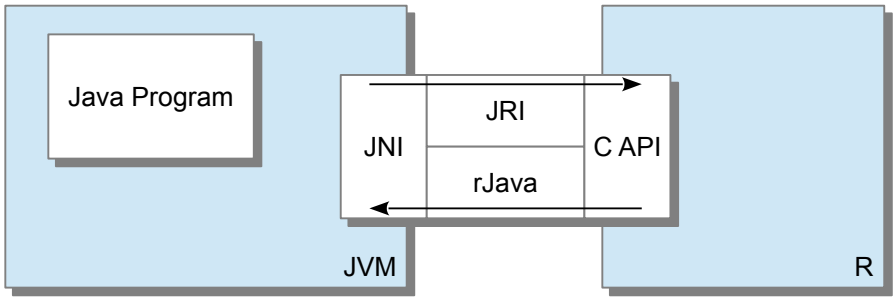


Figure 5.5: Interface layers of the R package `rJava` (after [Urbanek, 2009])

Here, an example of code snippet using the EM algorithm demonstrating how methods in R are applied in the Java environment is shown below:

```
/*
 * variables:
 *  aspect: an array of time series data points
 *  emAlg : an object of the class EM
 *  re    : an object of the class Rengine
 */
EM emAlg = new EM(aspect);
emAlg.doClustering(re);
emAlg.getCluster();
```

where the R scripts for finding Tones are wrapped in the method `doClustering(re:Rengine)`, and the class Rengine is a class defined in JRI. The method `getCluster()` will further return an array of integers indicating the cluster number which each time series data point belongs to.

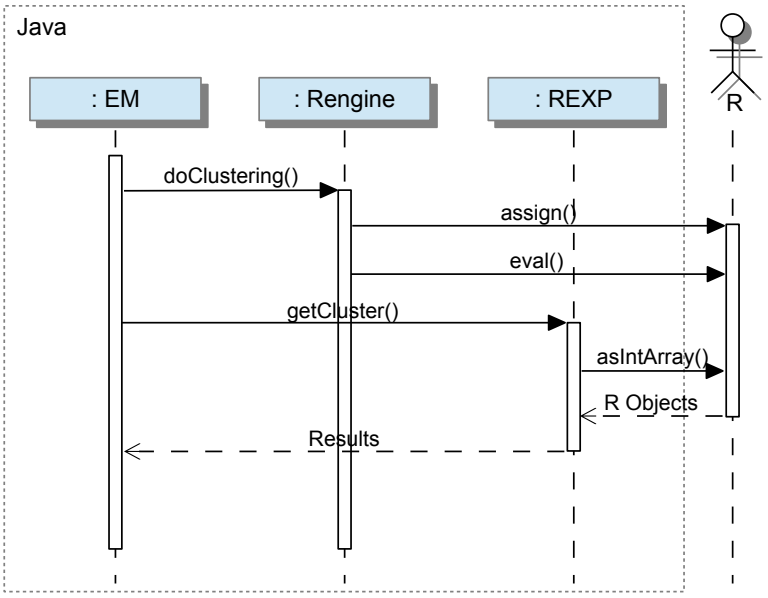


Figure 5.6: Example of information exchange between R and Java

The example above involves the information exchange between R and Java through the help of the `rJava` as mentioned earlier. To further illustrate

how information is exchanged between R and Java, it is shown in Fig. 5.6, which reflects the code snippet above. In Fig. 5.6, an object of class EM is trying to perform the task of finding Tones. In order to accomplish the method `doClustering(re:Rengine)`, an object of the class Rengine is created to assign and to evaluate the statements given to R. Finally, an object of the class REXP is created to retrieve computed R objects back to the JVM and these R objects are also converted to Java primitive data types for further usage, for instance, the method `getCluster()`.

All performed actions will be recorded as a part of the workflow monitoring mentioned in Section 4.1. In the current implementation of the prototype, the actions above are logged in the plain text file format as:

```
This file is created on Wed Jan 29 18:05:09 CET

[...]

Date/Time  : Wed Jan 29 18:10:51 CET 2014
Action     : Clustering
  Function : Mclust in the package mclust ver. 4.2
             R version 3.0.2 (2013-09-25)
  Algorithm: Model-Based
  Parameter: default
             please check the package manual for more
             detailed information
```

In this log file, some basic information, such as date/time, algorithm, method, package version, etc., is recorded.

5.5 Process Identification Design and Implementation

The idea of process identification, as discussed in previous chapters, is to formulate the relationships among physical state variables with the help of

Mamdani-type Fuzzy Inference System (FIS) as well as Multivariate Adaptive Regression Splines (MARS).

In the first part of the process identification, the open source fuzzy logic library "jFuzzyLogic" [Cingolani and Alcala-Fdez, 2012] is adopted as the engine of the Mamdani-type FIS. In order to carry out this part using fuzzy logic, an input file containing information, such as variables, membership functions, fuzzy rules, etc., has to be generated. The format of the file is in the form of the Fuzzy Control Language (FCL) specification [IEC 1131, 1997] and the jFuzzyLogic library also implements it. An example of the file format containing the settings of:

- three input variables of `var1`, `var2` and `var3`
- one output variable of `var4`
- membership functions defined by points
- aggregation defined by $\text{Min}(\mu_A(x), \mu_B(x))$
- activation defined by $\text{Min}(\mu_A(x), \mu_B(x))$
- accumulation defined by $\text{Max}(\mu_A(x), \mu_B(x))$
- defuzzification with the Center of Gravity (CoG) method

is shown as follows:

```
FUNCTION_BLOCK function

// define input variables and their types
VAR_INPUT
  var1 : REAL;
  var2 : REAL;
  var3 : REAL;
END_VAR

// define output variables and their types
VAR_OUTPUT
```

```
  var4 : REAL;
END_VAR

// fuzzification of input variable 'var1'
FUZZIFY var1
  TERM var1_desc1 := (0,1) (1.0,0);
  TERM var1_desc2 := (0,0) (1.0,1) (2.00,0);
  TERM var1_desc3 := (1.0,0) (2.0,1);
END_FUZZIFY

// fuzzification input variable 'var2'
FUZZIFY var2
  TERM var2_desc1 := (0.03,1) (1.06,0);
  TERM var2_desc2 := (0.03,0) (1.06,1) (2.00,0);
  TERM var2_desc3 := (1.06,0) (2.00,1);
END_FUZZIFY

// fuzzification input variable 'var3'
...

// defuzzification of output variable 'var4'
DEFUZZIFY var4
  TERM var4_desc1 := (0.07,1) (1.355,0);
  TERM var4_desc2 := (0.07,0) (1.355, 1) (3.41,0);
  TERM var4_desc3 := (1.355,0) (3.41,1);

  // use COG for defuzzification
  METHOD : COG;
  // default value set to '0'
  DEFAULT := 0;
END_DEFUZZIFY

RULEBLOCK rule
  AND : MIN;    // 'minimum' for aggregation
  ACT : MIN;    // 'minimum' for activation
```

```
  ACCU : MAX;    // 'maximum' for accumulation

  RULE 1 : IF var1 IS var1_desc2 AND var2 IS var2_desc3
           THEN var4 IS var4_desc2;
  ...
END_RULEBLOCK

END_FUNCTION_BLOCK
```

Some parameters are already known beforehand, such as input and output variables, and some can be defined by users, like the aggregation and defuzzification methods. However, the generation of rules is an intractable task. Here, the rules can be generated by the method `doFiltering(chords:List,index:int[])` in the class Chord as described in Section 5.4, in addition to rules of thumb or experts' experience. With proper arguments given, different sets of rules can be generated and converted to meet the users' requirements.

What is more, this library also provides the funtionality of optimization of membership functions and weighting factors of rules. There are also several optimization algorithms available, like gradient method, to fit the needs of the problem.

As for the second part of the process identification, the R package `earth` [Milborrow, 2011] is chosen for the task of carrying out the regression technique MARS in the environment of REPL. As for the data exchange, it is simply done through plain text file format as mentioned in Section 5.2. The file format is a simple CSV file format as described in Section 5.3. In this way, the results from the Mamdani-type FIS can be inspected publicly, and then it can be decided if the further process is necessary.

5.6 Scenario Composition Design and Implementation

In this section, the design and the implementation of the scenario composition are introduced. Unlike the modules mentioned in the previous two sections, event identification and process identification, which are based on the framework or the theory, the module of scenario composition uses the results from the aforementioned modules to compose scenarios of interest. In addition to compose scenarios of interest, the scenario composition also supports users to generate time series data sets based on these user-created scenarios. In this sense, scenario composition has to at least provide two capabilities of:

- providing intuitive and sufficient information for users to choose from
- supporting users to compose scenarios of interest with descent interfaces/tools

In order to suffice these two criteria, two parts of scenario composition are introduced as follows:

MetaEvent: A MetaEvent, as its name represents, is the composition of an Event, which is denoted as Chord in TSKM, and the metadata of its own. Besides, a MetaEvent is an aggregation of different Events with the same patterns, which means Events sharing the same composition of Tones will be regarded as one MetaEvent. In this case, a MetaEvent contains no more specific duration information but the information of a maximum and a minimum duration among all Events. Also, the naming in the design of scenario composition, as shown in Fig. 5.7, starts to deviate from that in TSKM for the reasons of:

- a clearer representation in the context of the framework
- an implication of departing from the main concept of TSKM

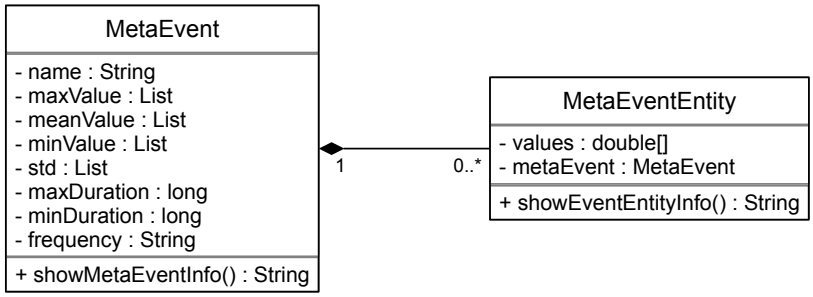


Figure 5.7: Class MetaEvent and class MetaEventEntity in the scenario composition

As shown in Fig. 5.7, a MetaEvent contains not only the basic information of an Event, but also other information derived from the existing time series data set. In the current implementation, the metadata contain the information of:

- the maximum value of each existing Aspect object
- the minimum value of each existing Aspect object
- the mean value of each existing Aspect object
- the maximum duration of the Event object
- the minimum duration of the Event object
- the frequency of the Event object

and they can be extended if necessary.

Apart from the MetaEvent, the concept of MetaEventEntity is also introduced in this design as shown in Fig. 5.7. With respect to the MetaEventEntity, it can be regarded as the realization of a MetaEvent — the real *entity* represents the corresponding MetaEvent in creating scenarios. Therefore, it contains the information of the corresponding MetaEvent and the value of each existing Aspect as shown in Fig. 5.7.

In addition to the MetaEvent and the MetaEventEntity, there exists another design which facilitates the scenario composition — Sequence, as shown in Fig. 5.8. The class Sequence does not play an explicit role

in the scenario composition, but it provides necessary information, the sequence of Chords, while composing scenarios as requested in the first criterion. The class Sequence can be regarded as a wrapper of the algorithm generating the suffix tree data structure discussed in Section 3.6. As shown in Fig. 5.8, it contains some basic methods, such as:

- `findNextEvent(val:Chord)`: It is an instance method which finds the next MetaEvent(s) of a given Chord/MetaEvent.
- `ifEnds(val:Chord)`: It is an instance method which checks if the given Chord/MetaEvent is the last one in the suffix tree data structure.

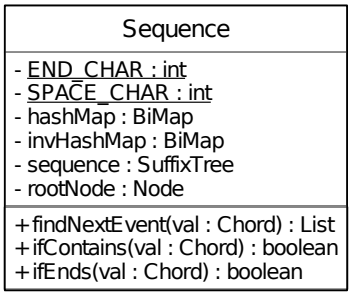


Figure 5.8: Class Sequence in the scenario composition

On the road to implementation, a Java package implementing Ukkonen's suffix tree algorithm by [Havsiyevych] is utilized to support the creation of a suffix tree based on a list of MetaEvents. The list of MetaEvents is derived from the event identification once the Events (Chords) are determined together with the original time series data sets as shown in Fig. 4.1. Depending on the parameters chosen in the process of mining Chords, the derived Chords can overlap in time in order to describe different possible phenomena as shown in Fig. 3.7.

Since the idea of scenario composition is to find the most fundamental elements which work as LEGO® bricks first, and then use them to compose scenarios with other supplementary information. Therefore, the temporal overlapping in Chords is not allowed and Chords have to

be “chopped” to come to a list of nonoverlapping Events, which is the Phrase in TSKR as shown in Fig. 3.7.

In the implementation of this prototype, the criteria to break temporal overlapping Chords into individual Events are mainly based on the maximum size of the Chord. As Fig. 3.7 shows, three Chords, 2-Chord AB, 3-Chord ABC, and 2-Chord BC, overlap temporally. Due to the characteristics of Chords, the duration of 3-Chord is less than that of 2-Chord. In this case, the entire 3-Chord will be preserved and turned into an Event, and the rest of the other overlapping parts will be converted to the corresponding Events. Finally, these Events in addition to the original time series data sets will be merged to create MetaEvents. The naming of each individual MetaEvent is assigned automatically with a predefined rule instead of determined by users, and all the metadata are stored as fields in each MetaEvent object.

After a list of nonoverlapping MetaEvents is generated, it can be viewed as an analogy to a string, and each MetaEvent is like a character. None the less, the data structure of the suffix tree is targeting at strings/characters. Due to this reason, the mapping between each MetaEvent and a character has to be set up first as illustrated in Fig. 5.8. Since the native character encoding in Java is UTF-16, it offers enough character candidates to map onto available MetaEvents.

Once the data structure of the suffix tree based on MetaEvents is constructed, it offers a foundation:

- to provide information regarding each available MetaEvent for users to choose from
- to create corresponding MetaEventEntities to compose scenarios of interest

GUI: The design and implementation of a GUI is to suffice for the second criterion mentioned earlier — supporting the composition of scenarios of interest. The design of the GUI, which is currently desktop-oriented and considers future extensions, comprises:

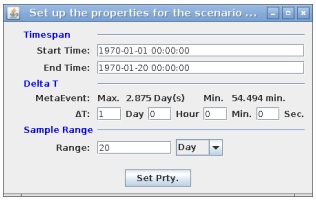
- components:
  - two tabbed panes for scenario composition and time series generation
  - two panels for each pane displaying the composed scenario or generated time series graphically and logging the operations
- operations:
  - scenario composition pane:
    - setting general properties, such as start time, end time, time difference Δt, etc.
    - composing events
    - generating time series data based on the composed scenario
    - exporting the composed scenario
  - time series generation pane:
    - changing how time series data are represented
    - showing information of selected point
    - exporting generated time series

A simple GUI, based on the design above, was implemented to assist in composing scenarios of interest and further generating the desired time series data as shown in Fig. 5.9 and Fig. 5.10 respectively.

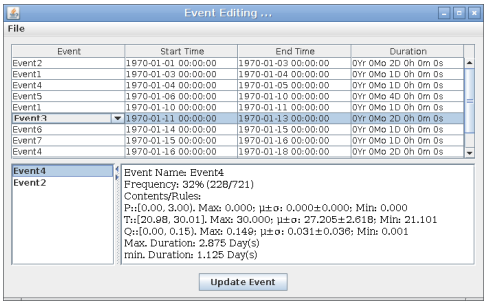
Fig. 5.9 shows a simple GUI to help users composing scenarios of interest. It contains three windows:

- the scenario composition window
- the scenario property window
- the event editing window

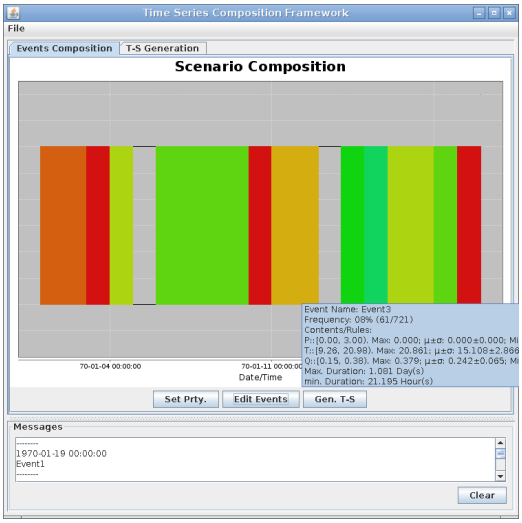
The scenario composition window displays the information of the composed scenario once it is updated. A horizontal bar plot exists in the center of the window and its bottom side is the time axis indicating the start time and the end time of each MetaEvent and the entire scenario.



Scenario Property Window



Event Editing Window



Scenario Composition Window

Figure 5.9: Windows of event composition in the scenario composition

Inside the bar plot, each rectangle represents a MetaEventEntity and different colors mean different MetaEvents. Once the cursor is hovered on top of a rectangle, the metadata of this MetaEventEntity will show up as shown in Fig. 5.9. In addition, the scroll pane at the bottom shows the messages of the operations.

In order to compose scenarios for the purpose of generating discrete time series data, several parameters have to be given, such as the time where the scenario begins, the time where the scenario ends, the time difference Δt, and the period. These parameters can be assigned in the scenario property window (Fig. 5.9).

The event editing window (Fig. 5.9) is a three-panel window assisting users to compose scenarios. The upper panel contains a spreadsheet-like table which defines the starting time and the end time of each

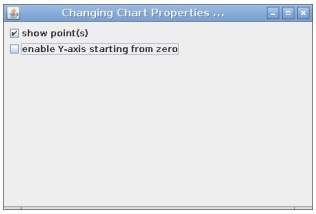
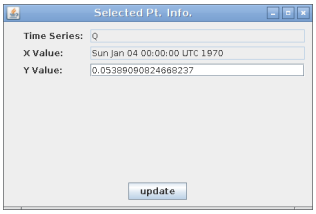
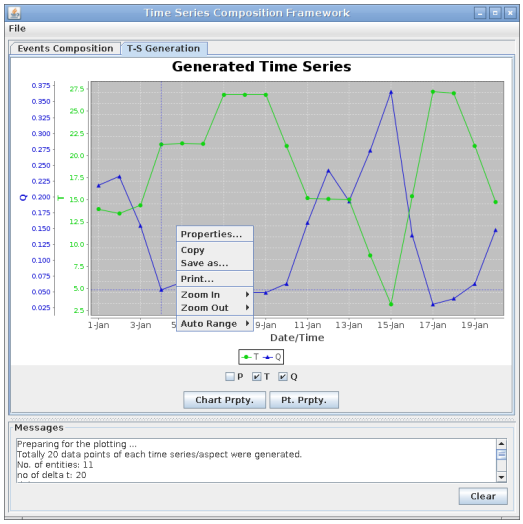


Chart Property Window



Selected Point Information Window



Time Series Generation Window

Figure 5.10: Windows of time series generation in the scenario composition

MetaEventEntity chosen from the list of MetaEvents in the drop-down menu and the available MetaEvents can be found inside the drop-down menu. As soon as the starting time and the end time are given, the duration of the MetaEventEntity will be calculated automatically. Once the MetaEvent is selected in the upper panel, the lower-left panel will display the possible following MetaEvent or MetaEvents based on the generated suffix tree data structure. The lower-right panel shows the metadata of the selected MetaEvent once it is picked up in the lower-left panel. In addition, the composed scenario can be exported from the `File` menu to a CSV file, and the file format is as:

```
MetaEvent,Start Time,End Time
Event21,1980-03-01 00:00:00,1980-04-01 00:00:00
Event22,1980-04-01 00:00:00,1980-04-02 00:00:00
Event23,1980-04-02 00:00:00,1980-04-04 00:00:00
Event28,1980-04-04 00:00:00,1980-04-05 00:00:00
```

When the composition of a scenario is complete, time series data sets can be generated through the button “Gen. T-S” in the scenario composition window shown in Fig. 5.9. The operation of generating time series data sets is based on the settings given in the scenario property window (Fig. 5.9). If there is a gap between two MetaEventEntities, the values of the generated time series data sets in the range of the gap are filled by linear interpolated values derived from the collected data as the background values. The generated time series data sets will appear in the time series generation window as shown in Fig. 5.10.

In a nutshell, to compose the scenario of interest and to generate the time series data based on the scenario, the following steps are performed:

1. Read the binary file containing information of Events generated from event identification.
2. Define parameters for the scenario, such as time span, time difference, and the period in the scenario property window.
3. Compose the scenario of interest with the given information in the event editing window.
4. Generate the time series data through the button “Gen. T-S” in the scenario composition window.

The time series generation window has the same layout as that of the scenario composition window in Fig. 5.9, and the only difference is that the middle panel shows the corresponding time series data sets instead of the bar plot of the scenario. This time series plot supports some basic functions to view the results, for example, zooming in, zooming out, showing selected time series data sets only, etc. In addition, the generated time series data sets can be exported to three different basic file formats in this prototype implementation:

- Portable Network Graphics (PNG) for raster images
- Portable Document Format (PDF) for vector images

- Comma-Separated Values (CSV) for raw data

Other file formats, such as XLS, WaterML2 [WaterML2], etc., can be added in the future if necessary.

Apart from the time series generation window, there are two other windows:

- the chart property window
- the selected point information window

The chart property window is to change the display style of the chart, e.g. highlighting discrete values by displaying points. The selected point information window shows the value of the selected point on the time series generation window. In addition to show the value of a selected point, the value can also be changed by the users. Once the value is updated, the change will be displayed right away on the time series generation window.

In addition, four application examples will be used to demonstrate how the prototype illustrates the concept, capabilities, etc., of the framework with different data sets in Chapter 6.

5.7 Hydroinformatics Systems Integration

As mentioned in Section 1.3, the objective of this work is mainly to assist simulation tasks with synthetic time series data sets based on collected historical information as inputs, such as Boundary Conditions (BCs). Therefore, an integration with available hydroinformatics systems is necessary, and a schematic illustration of the integration with hydroinfomratics systems is shown in Fig. 5.11.

As illustrated in Fig. 5.11, this framework plays as a role of a piece of jigsaw puzzles in simulation tasks. The external inputs can be from field measurements, laboratory experiments, or other simulation models. Based on the users' needs and experience, this framework can generate different

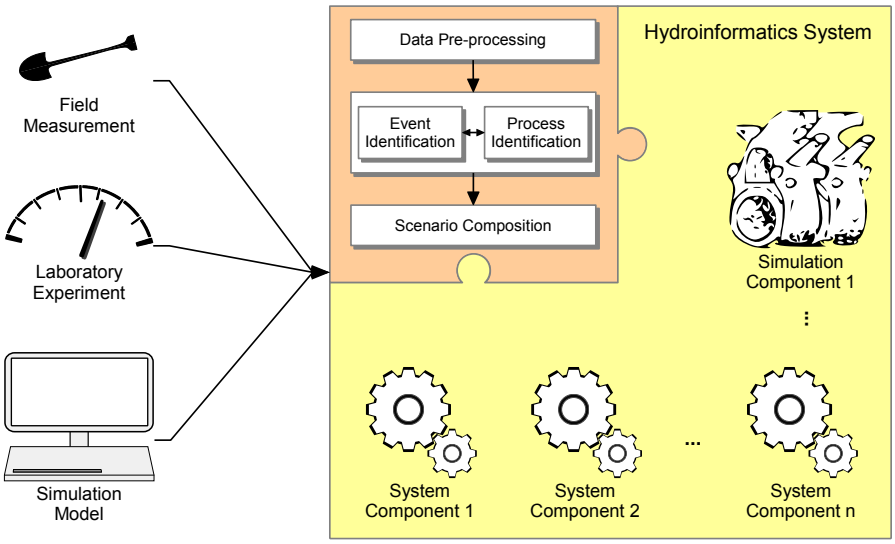


Figure 5.11: Schematic illustration of the integration with hydroinformatics systems

time series data sets based on the user-specified scenarios for further simulation tasks. The methods of integration with other hydroinformatics systems can be carried out through:

- file exchange, either through well-structured plain text files, such as CSV, WaterML2 [WaterML2], etc., or binary files if the specification is known or the tool is available, e.g. `.dfs0` time series data format for MIKE [DHI], as illustrated in this prototype
- Java Application Programming Interface (API) with other Java-based tools, such as Kalypso [BCE], Turtle [Molkenthin et al., 2009]
- Internet-based protocals, such as XML-RPC [XML-RPC.com] or Simple Object Access Protocol (SOAP) [W3C – World Wide Web Consortium] exchanging information through HTTP, if the server-client architecture, e.g. through `RServe` package [Urbanek, 2012] as mentioned in Section 5.2, is set up

A simple example can be considered as a hydroinformatics system receiving external inputs through file exchange, e.g. CSV, or Database Management

Systems (DBMSs), e.g. MySQL, and communicating with other system components through Java API assuming the system is Java-based. In this scenario, these external inputs have to be pre-processed, e.g. outliers detection, gap filling, scaling, etc., before applying them and this can be performed in the data pre-processing module in this framework once the data are loaded into the system. Afterwards, users can compose the scenarios of their interests and further generate the required time series data for further simulation tasks. Once these required time series data are generated, they are stored as objects inside the system. Through the help of API, the system components can access these data completely and can even extract the necessary parts of the data or manipulate the data based on their requirements. Once these data are applied for simulation tasks, further decisions can be made based on the investigation of the simulation results.

Chapter 6

Concept and Prototype Applications

6.1 Preliminary Remarks

In this chapter, two types of categorical applications, concept and prototype applications, are presented. The concept application illustrates which type of applications the framework is targeting at. However, due to the implementation of the prototype, it is difficult to demonstrate and further to judge the concept of this framework only with the prototype. Instead, artificial or smaller data sets are adopted in the prototype applications detailing the workflow, the capabilities, etc., of the framework.

The prototype applications contain four different application examples detailing how the framework works, inspecting its feasibility for scenario composition, and further demonstrating how the framework works together with other simulation tools. The first application example is an artificial data set generated by four different mathematical equations for the purpose of testing different functions/modules in the framework prototype. In the second application example, an artificial data set from the R package `hydromad` [Andrews and Guillaume, 2012] is considered. This data set is generated and used to test the empirical hydrological modeling framework in the package itself. In the third application example, a hydrological data set measured at Ernies Catchment, Western Australia is adopted and used to demonstrate the capability of scenario composition. In the end, the data

set describing the 1997 Oder Flood in the area of German-Polish border are chosen to illustrate how the framework works with other simulation tools in the fourth application example.

In these applications, the main focuses lie on event identification, process identification, and scenario composition. The discussion of data preprocessing is not the main focus of this chapter and also less described because it is very domain- and problem-specific as mentioned earlier. Besides, the data sets used in the chapter are mostly complete and validated.

6.2 Concept Application

As the motivation and the objective mentioned in Section 1.2 and Section 1.3, applications or projects dealing with the complexity and an abundance of data are ideal to demonstrate the concept of this framework, and one typical example is the Großhang project [Hinkelmann et al.; Molkenthin et al., 2014; Zehe and Hinkelmann, 2013]. This project deals with the complexity of different time and space scales and different physical state variables of years of daily records.

The interdisciplinary research project “Großhang — Natural Slope” [Hinkelmann et al.; Molkenthin et al., 2014; Zehe and Hinkelmann, 2013], as briefly indicated in Section 2.3, is targeting at investigating and understanding the movement of large hillslopes until failure and further simulating this phenomenon, and the study area is situated at Ebnit, Austria. This phenomenon covers complicated interactions among different processes in rainfall, runoff, infiltration, subsurface hydraulics, soil deformation, etc., and no suitable simulation tool was available to model the interacting processes. In the period of this project, varieties of data in different temporal and spatial scales, such as rainfall, discharge, soil moisture, seismic events, etc., were collected, and the project is divided into 5+1 sub-projects:

- Sub-project 1 “Hydrology and Applied Seismics” identifying the structures, parameters, and processes of the hillslope hydrology and the slope deformation of the study area

- Sub-project 2 “Subsurface Hydraulics” developing and simulating the air-water two-phase subsurface flow in the macroporous soils and the rainfall-runoff
- Sub-project 3 “Continuum Mechanics” formulating a continuum mechanical model for the coupled flow and deformation processes in the unsaturated soils
- Sub-project 4 “Technical Scale Experiments” exploring the infiltration and the soil deformation processes, identifying parameters, and verifying models by laboratory-based experiments
- Sub-project 5 “Geophysics” investigating the slope movements with different approaches, such as (nano-)seismic monitoring techniques, direct-current resistivity measurements, etc., and the soil moisture dynamics with electromagnetic induction
- Central Sub-project “Project and Information Management” managing and integrating information from all sub-projects

Although such projects show the need and are suitable for this framework concept, several details have to be further examined and implemented, for instance, the algorithms to determine Tones, Chords, etc., before being applied in such complex projects. The current design and implementation of the prototype, as discussed in Chapter 5, offers the software framework for further extensions, such as algorithms, functionalities, etc., with basic and general methods to perform tasks needed in the framework. Due to the reason that these methods are not implemented and optimized specifically for a specific project, a more well-designed and appropriate data set is needed. Therefore, to demonstrate the concept of this framework, artificial or smaller data sets are adopted in the prototype as examples in the coming sections. Although the implemented methods are not optimized for any specific problem as mentioned earlier, these demonstrations still exhibit the workflow of the framework, the expected results, the capabilities, etc. with artificial or smaller data sets.

6.3 Academic Test Case

In this academic test case, an one-year artificial daily time series data from 1970-01-01 to 1970-12-31 are generated by four mathematical functions with x from 1 to 365:

$$f1 = sin(\pi/90 \cdot x) + 1 \tag{6.1}$$

$$f2 = [1 + \exp(-10/365 \cdot (x-1) + 4)]^{-2} \tag{6.2}$$

$$f3 = 2^{[1-4/364(x-1)]} \tag{6.3}$$

$$f4 = 0.5 \times f1 + \exp f2 + \log f3 \tag{6.4}$$

where $f1$ is a sine function representing a periodic phenomena in nature; $f2$ is a sigmoid function describing a natural process with a slow start, then a rapid acceleration during the process, and a slowdown till saturation; $f3$ is an exponentially decreasing function indicating a degrading development; $f4$ is a random nonlinear combination of previous three functions, $f1$, $f2$, and $f3$, defining an arbitrary cause-effect relevance. These functions are shown in Fig. 6.1, and the basic statistical information is shown in Table 6.1.

Table 6.1: Description of the data set in the academic test case

	no.	mean	sd	median	min	max	range
$f1$	365	1.00	0.70	1.03	0.00	2.00	2.00
$f2$	365	1.19	0.76	1.46	0.04	1.99	1.96
$f3$	365	0.68	0.51	0.50	0.12	2.00	1.88
$f4$	365	4.42	2.15	4.54	1.82	7.04	5.21

The purpose of this test case focuses mainly on the event identification and the process identification, while the scenario composition is neglected in this test case simply because it has no definite physical meaning.

Based on the definition of Aspect, each series can be viewed as an Aspect and directly used in the framework. For the purpose of simplification and demonstration, each Aspect is divided into three categories, denoted as bins,

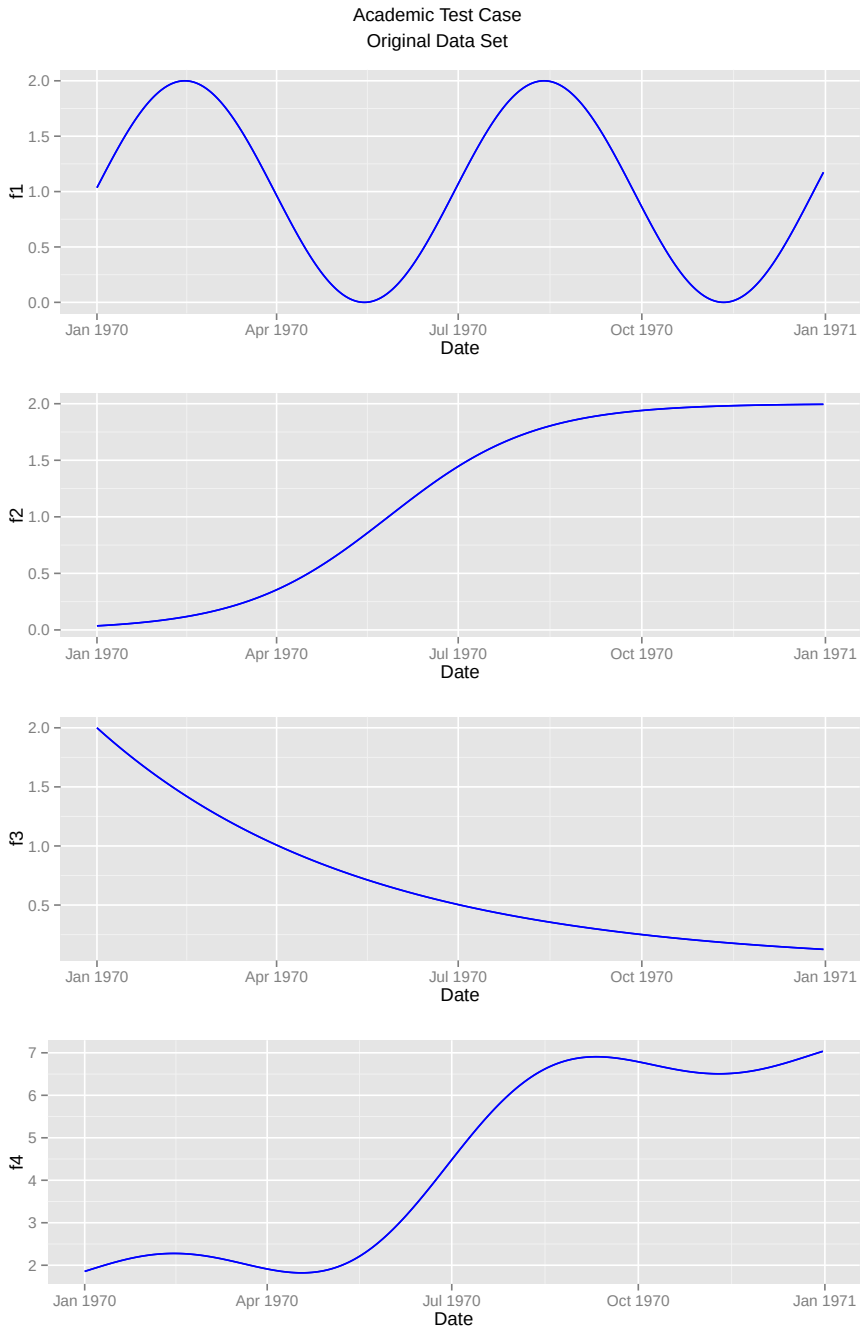


Figure 6.1: Original data set in the academic test case

by the method of the k-means clustering algorithm with $k = 3$ and the results (Tones) are displayed in Fig. 6.2. These bins are as follows:

- $f1$
  - $0.00 \leq x < 0.61$
  - $0.61 \leq x < 1.39$
  - $1.39 \leq x \leq 2.01$
- $f2$
  - $0.04 \leq x < 0.65$
  - $0.65 \leq x < 1.47$
  - $1.47 \leq x \leq 2.00$
- $f3$
  - $0.12 \leq x < 0.58$
  - $0.58 \leq x < 1.22$
  - $1.22 \leq x \leq 2.01$
- $f4$
  - $1.82 \leq x < 3.28$
  - $3.28 \leq x < 5.49$
  - $5.49 \leq x \leq 7.05$

where x represents any dependent value in the functions, $f1$, $f2$, $f3$, and $f4$.

The Fig. 6.2 illustrates how different time series are categorized and how Events are formed. As shown in the previous results, every function is categorized into three different categories (bins) simply representing the basic common sense of classifying a value as high, median, or low. In Fig. 6.2, three different colors represent these three different bins of different function. For example, the horizontal fine dashed rectangle appearing on top of the $f3$ function plot depicts the category $1.22 \leq x \leq 2.01$ of the function $f3$.

Once the Tones are decided, the next step is to find Chords which represent the concept of coincidence by the modified Closed Association

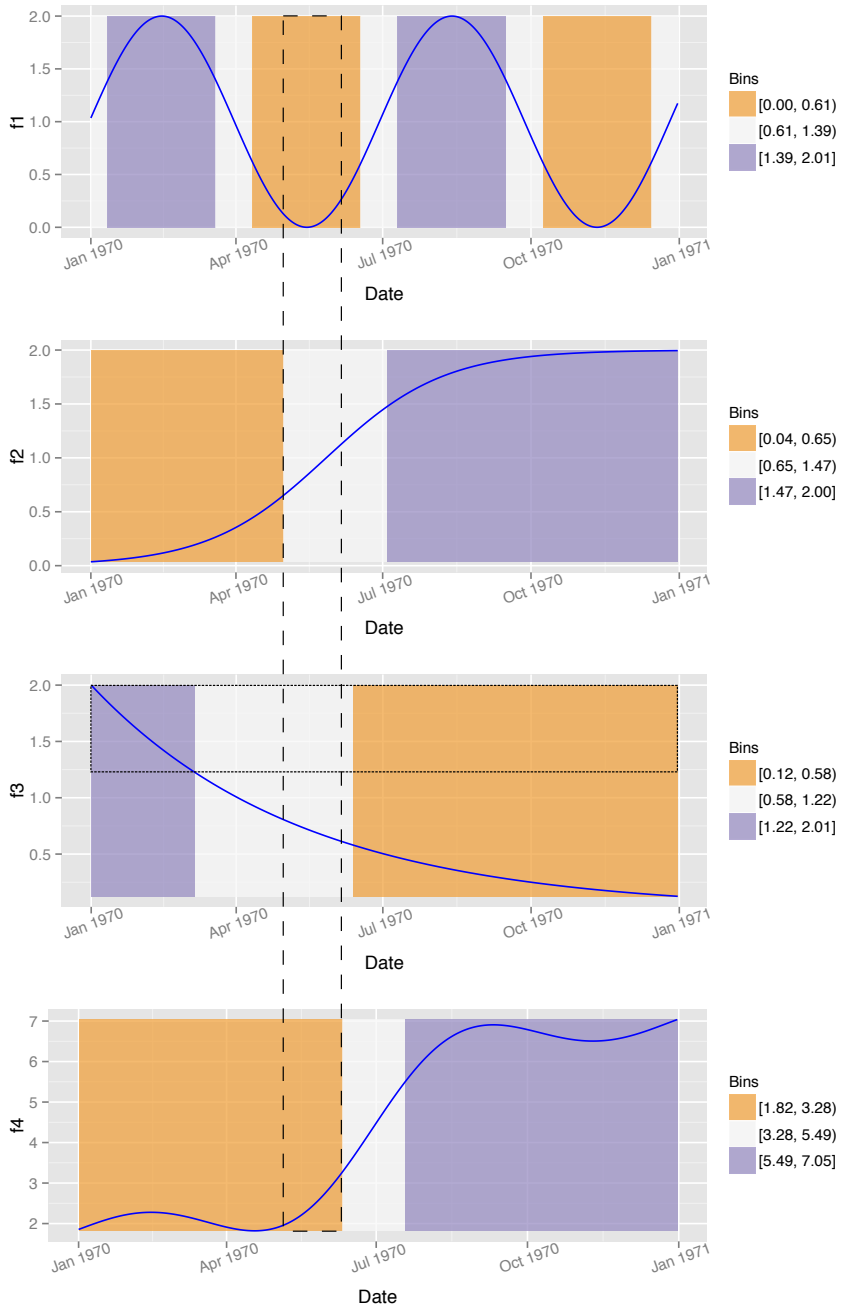


Figure 6.2: Derived Tones in the academic test case

Rule Mining (CHARM) algorithm. This concept of a Chord is also represented in Fig. 6.2. There, the vertical coarse dashed rectangle span-

ning across all four plots forms the concept of Event, which can further become the MetaEvent Event6 shown in Table 6.2 if more information is added. Here, the process of finding Tones is carried out by the static method `miningChords(aspects:List, param:MiningParam)` in the class Chord, as mentioned in Section 5.4 with the default settings except setting the minimum size of Chords to four to avoid overlaps in the derived Chords and these derived Chords can be further used to form MetaEvents. Finally, the total number of 12 MetaEvents are generated as shown in Table 6.2. Each MetaEvent in Table 6.2 contains statistics derived from the existing data set:

- the frequency (Feq.) of each MetaEvent
- the maximum and minimum duration of each MetaEvent
- the rule of each variable
- the maximum, mean (μ), standard deviation (σ), and minimum values of each variable

Table 6.2: List of 12 generated MetaEvents based on the data set in academic test case

		Feq.	Max. Duration	Min. Duration
		03% (11/365)	1.500 Week(s)	1.500 Week(s)
Event1	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	f1::[0.61, 1.39)	1.375	1.207 ± 0.113	1.035
	f2::[0.04, 0.65)	0.047	0.041 ± 0.004	0.036
	f3::[1.22, 2.01]	2.000	1.926 ± 0.049	1.853
	f4::[1.82, 3.28)	2.003	1.930 ± 0.049	1.855

		Feq.	Max. Duration	Min. Duration
		15% (54/365)	7.714 Week(s)	7.714 Week(s)
Event2	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	f1::[1.39, 2.01]	2.000	1.836 ± 0.166	1.407
	f2::[0.04, 0.65)	0.191	0.105 ± 0.042	0.048
	f3::[1.22, 2.01]	1.839	1.514 ± 0.181	1.228
	f4::[1.82, 3.28)	2.278	2.207 ± 0.073	2.018
		Feq.	Max. Duration	Min. Duration
		04% (13/365)	1.857 Week(s)	1.857 Week(s)
Event3	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	f1::[1.39, 2.01]	1.743	1.583 ± 0.110	1.407
	f2::[0.04, 0.65)	0.262	0.228 ± 0.022	0.196
	f3::[0.58, 1.22)	1.219	1.165 ± 0.035	1.113
	f4::[1.82, 3.28)	2.174	2.114 ± 0.041	2.050
		Feq.	Max. Duration	Min. Duration
		06% (23/365)	3.286 Week(s)	3.286 Week(s)
Event4	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	f1::[0.61, 1.39)	1.375	1.000 ± 0.233	0.625
	f2::[0.04, 0.65)	0.442	0.350 ± 0.053	0.269
	f3::[0.58, 1.22)	1.104	1.017 ± 0.053	0.934
	f4::[1.82, 3.28)	2.039	1.927 ± 0.063	1.839
		Feq.	Max. Duration	Min. Duration
		05% (19/365)	2.714 Week(s)	2.714 Week(s)
Event5	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	f1::[0.00, 0.61)	0.593	0.343 ± 0.145	0.134
	f2::[0.04, 0.65)	0.646	0.545 ± 0.061	0.451
	f3::[0.58, 1.22)	0.927	0.866 ± 0.037	0.808
	f4::[1.82, 3.28)	1.883	1.837 ± 0.019	1.821

		Feq.	Max. Duration	Min. Duration
		11% (41/365)	5.857 Week(s)	5.857 Week(s)
Event6	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	f1::[0.00, 0.61)	0.384	0.103 ± 0.110	0.000
	f2::[0.65, 1.47)	1.189	0.920 ± 0.161	0.658
	f3::[0.58, 1.22)	0.802	0.691 ± 0.063	0.591
	f4::[1.82, 3.28)	3.249	2.431 ± 0.414	1.894
		Feq.	Max. Duration	Min. Duration
		04% (16/365)	2.286 Week(s)	2.286 Week(s)
Event7	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	f1::[0.61, 1.39)	1.139	0.880 ± 0.164	0.625
	f2::[0.65, 1.47)	1.467	1.382 ± 0.056	1.293
	f3::[0.12, 0.58)	0.556	0.526 ± 0.019	0.496
	f4::[3.28, 5.49)	4.604	4.150 ± 0.287	3.700
		Feq.	Max. Duration	Min. Duration
		02% (7/365)	7.000 Day(s)	7.000 Day(s)
Event8	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	f1::[0.61, 1.39)	1.375	1.275 ± 0.072	1.174
	f2::[1.47, 2.00]	1.539	1.509 ± 0.022	1.478
	f3::[0.12, 0.58)	0.492	0.481 ± 0.008	0.470
	f4::[3.28, 5.49)	5.020	4.843 ± 0.128	4.664
		Feq.	Max. Duration	Min. Duration
		02% (8/365)	1.143 Week(s)	1.143 Week(s)
Event9	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	f1::[1.39, 2.01]	1.616	1.513 ± 0.073	1.407
	f2::[1.47, 2.00]	1.612	1.581 ± 0.022	1.549
	f3::[0.12, 0.58)	0.467	0.455 ± 0.008	0.443
	f4::[3.28, 5.49)	5.467	5.275 ± 0.136	5.078

		Feq.	Max. Duration	Min. Duration
		16% (59/365)	8.429 Week(s)	8.429 Week(s)
Event10	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	f1::[1.39, 2.01]	2.000	1.824 ± 0.164	1.407
	f2::[1.47, 2.00]	1.909	1.794 ± 0.085	1.621
	f3::[0.12, 0.58)	0.439	0.355 ± 0.046	0.282
	f4::[5.49, 7.05]	6.907	6.491 ± 0.427	5.520
		Feq.	Max. Duration	Min. Duration
		11% (40/365)	3.286 Week(s)	3.286 Week(s)
Event11	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	f1::[0.61, 1.39)	1.375	0.956 ± 0.214	0.625
	f2::[1.47, 2.00]	1.995	1.958 ± 0.032	1.911
	f3::[0.12, 0.58)	0.280	0.205 ± 0.064	0.125
	f4::[5.49, 7.05]	7.035	6.859 ± 0.080	6.719
		Feq.	Max. Duration	Min. Duration
		18% (67/365)	9.571 Week(s)	9.571 Week(s)
Event12	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	f1::[0.00, 0.61)	0.593	0.213 ± 0.186	0.000
	f2::[1.47, 2.00]	1.992	1.977 ± 0.012	1.952
	f3::[0.12, 0.58)	0.235	0.185 ± 0.027	0.142
	f4::[5.49, 7.05]	6.779	6.593 ± 0.077	6.506

In Table 6.2, the rules of variables are denoted by the mathematical interval notations as follows:

- $[a, b) = \{a \leq x < b, \forall x \in \mathbb{R}\}$
- $[a, b] = \{a \leq x \leq b, \forall x \in \mathbb{R}\}$

Besides, this list also contains information of frequency, maximum and minimum duration of each MetaEvent. The frequency is decided by the occurrences of historical data which fit into the type of MetaEvent. The default value assigned to each MetaEvent, as stated earlier in Section 4.5, is the averaged property of the affiliated MetaEvent. The comparison between

default values and original data is shown in Fig. 6.3, and at least three observations are recognized:

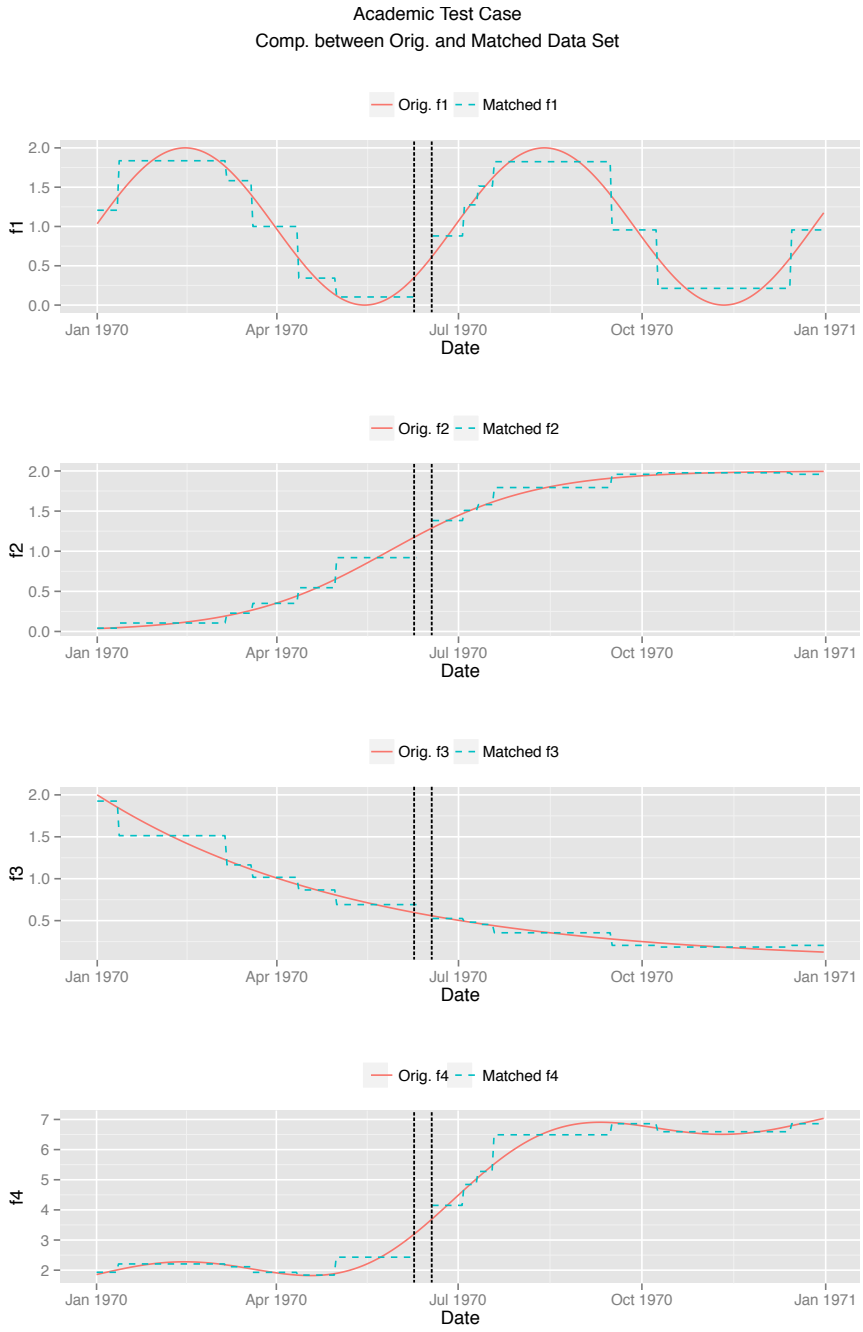


Figure 6.3: Comparing original data set with the matched MetaEvent default values (academic test case)

- Obvious stepwise patterns are perceived due to the default values coming from the averaged occurrences in each corresponding MetaEvent.
- The default values are not able to match extreme numbers for the same earlier reason.
- A gap starting from 1970-06-11 to 1970-06-17 in each time series is emphasized with a dashed column in Fig. 6.3. The reason why the gap exists in each time series is that the MetaEvents which match the values in the gap are considered less significant and dropped by the algorithm. While investigating this gap in Fig. 6.3, it can be noticed that these MetaEvents locate in the range where several Tones interact and have relatively short duration to the degree of few days compared to the duration of some weeks in Table 6.2. Of course, since the entire process is semi-automatic, these dropped MetaEvents can be revived by tuning parameters in the mining algorithm.

Although Table 6.2 is not able to represent the results of the derived suffix tree, it offers an overview of the features of the MetaEvents. Furthermore, a test of the process identification is carried out. Due to the fact that $f4$ is the function of $f1$, $f2$, and $f3$, as defined earlier, the $f4$ can be viewed as the target function to approach. Under this concept, a file containing 22 rules in the format for the process identification, as described in Section 5.5, was generated based on the existing MetaEvents derived from the event identification mentioned earlier. Within this file, the default settings of methods and parameters, such as triangular membership functions derived from the range of the bins of each function, Center of Gravity (CoG) for defuzzification, etc., were prepared for the process identification. The derived results (Sim. f4) compared with the values of original mathematical function (Orig. f4) are shown in Fig. 6.4. The Root-Mean-Square Error (RMSE) is equivalent to 0.114 and is able to represent the original mathematical function well.

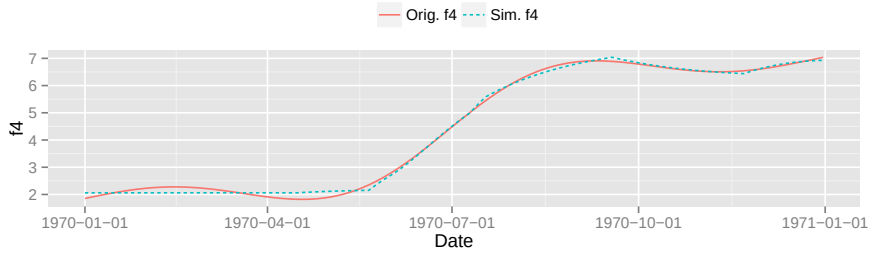


Figure 6.4: Results of the process identification in the academic test case

6.4 HydroTestData Data Set

Similar to the purpose of the academic test case earlier, an artificial hydrological data set from the R package `hydromad` [Andrews and Guillaume, 2012] is adopted here to test the event identification and process identification. This data set is used to test the hydrological modeling framework in the package `hydromad`, and contains three different physical state variables as shown in Fig. 6.5: rainfall (mm/day), temperature (°C), and streamflow (mm/day).

The package `hydromad` offers a spatially-lumped and empirical approach to simulate hydrological processes, such as the rainfall-runoff process. The approach contains two steps:

1. It generates effective rainfall by a soil moisture accounting model with inputs, such as rainfall, temperature, etc.

2. The effective rainfall from the previous step will be used in a routing model, which is available for different options inside the package `hydromad`, to generate streamflow.

As shown in Fig. 6.5, these data are for the duration of a three-month period with the time step of three hours ($\Delta t = 3$ hr). The rainfall data have a value of either 0 mm/day or 6 mm/day, except one record has the value of 24 mm/day. The temperature data are in the form of a sine function. Among these temperature data, the lowest value is 0 °C and the highest value is

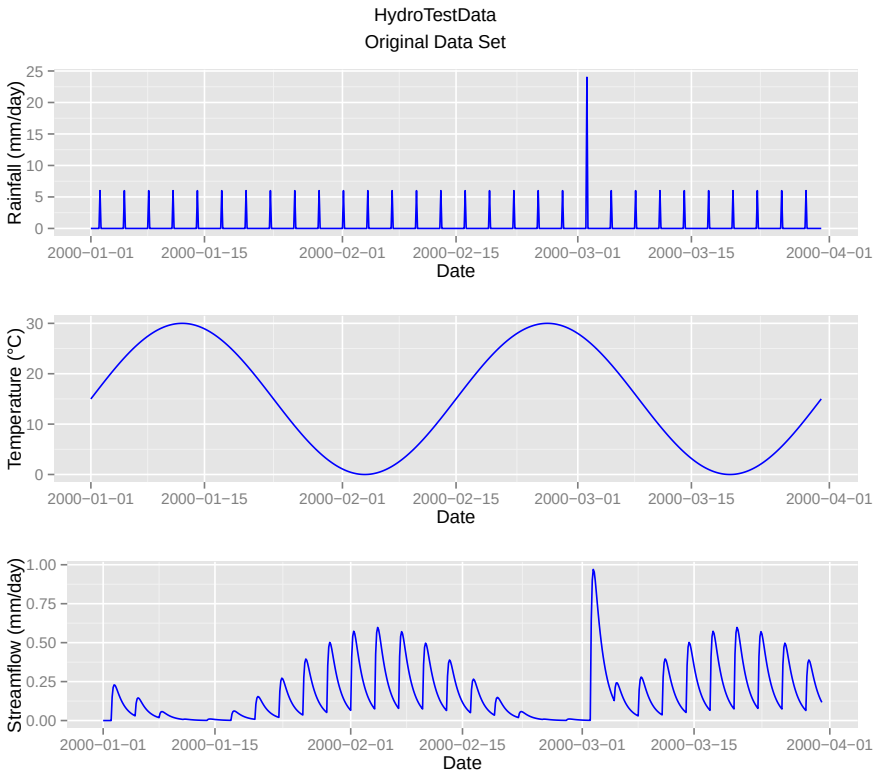


Figure 6.5: Original HydroTestData data set from the R package `hydromad` [Andrews and Guillaume, 2012]

30 °C. The streamflow data are generated from the hydrological model in the package `hydromad`. These data sets contain no missing value, and the general statistics are shown in Table 6.3.

Table 6.3: Description of the HydroTestData data set

	no.	mean	sd	median	min	max	range
Rainfall	721	0.27	1.47	0.00	0.00	24.00	24.00
Temperature	721	15.00	10.61	15.00	0.00	30.00	30.00
Streamflow	721	0.17	0.17	0.11	0.00	0.97	0.97

Since there are no missing values in the data set and each physical variable has only one series, each series can be viewed as an Aspect

according to the definition of the Aspect and can be used in the framework directly, leading to the same situation as in the academic test case.

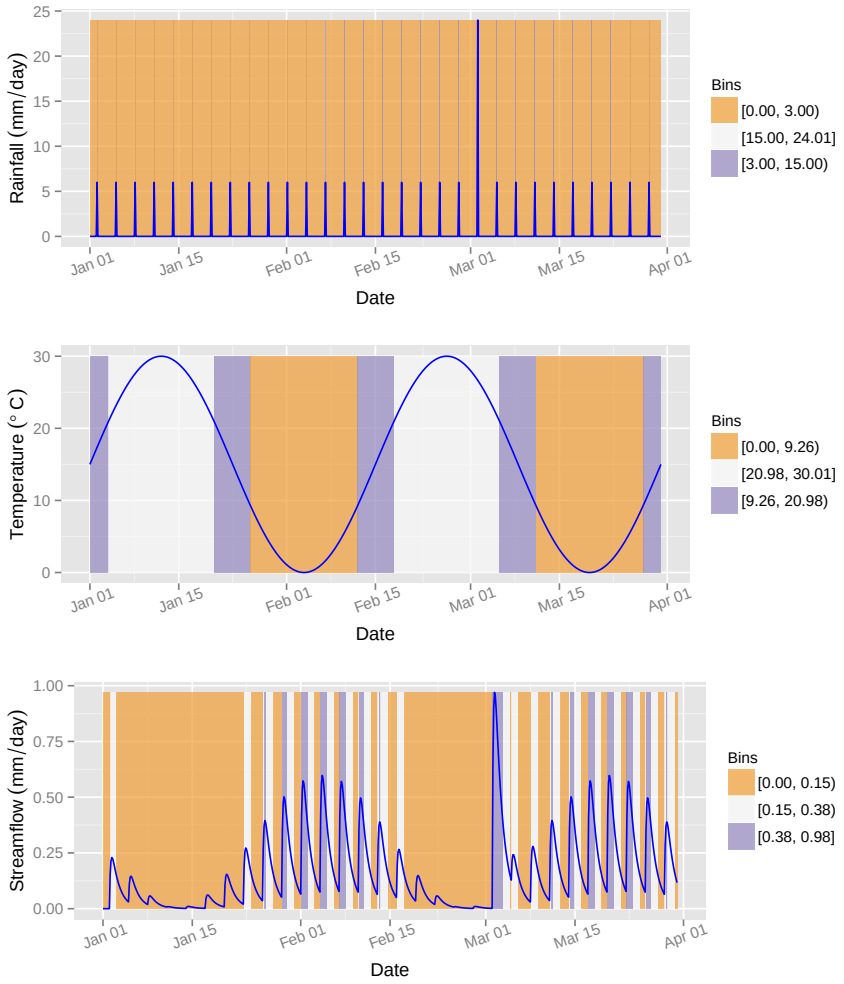


Figure 6.6: Derived Tones from the HydroTestData data set

Also, for the same reason to simplify the problem for the demonstration, each Aspect is divided into three categories, by the method of k-means clustering algorithm with $k = 3$. The results of the Tones for each Aspect are shown in Fig. 6.6, and the bins are as follows:

- Rainfall (mm/day)
  - $0.00 \leq x < 3.00$

  - $3.00 \leq x < 15.00$
  - $15.00 \leq x \leq 24.01$

- Temperature (°C)
  - $0.00 \leq x < 9.26$
  - $9.26 \leq x < 20.98$
  - $20.98 \leq x \leq 30.01$

- Streamflow (mm/day)
  - $0.00 \leq x < 0.15$
  - $0.15 \leq x < 0.38$
  - $0.38 \leq x \leq 0.98$

where x represents any physical state variable used here.

To derive MetaEvents based on the derived Tones, the default parameters, except setting the minimum size of Chords to three, are used in the modified CHARM algorithm implemented in the framework. In addition to the exisiting data set, a list of MetaEvents can be derived as shown in Table 6.4, and it contains 15 MetaEvents based on the conditions given above. In the description of the rules, the symbols, P, T, and Q, represent rainfall, temperature, and streamflow respectively.

Table 6.4: List of 15 generated MetaEvents based on the HydroTestData data set

		Feq.	Max. Duration	Min. Duration
		16% (114/721)	2.500 Day(s)	1.062 Day(s)
Event1	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[9.26, 20.98)	20.861	15.315 ± 3.677	9.381
	Q::[0.00, 0.15)	0.150	0.077 ± 0.040	0.000

		Feq.	Max. Duration	Min. Duration
		01% (8/721)	3.000 Hour(s)	1.437 Hour(s)
Event2	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[3.00, 15.00)	6.000	6.000 ± 0.000	6.000
	T::[9.26, 20.98)	17.347	13.468 ± 3.410	9.624
	Q::[0.15, 0.38)	0.285	0.232 ± 0.054	0.152
		Feq.	Max. Duration	Min. Duration
		08% (61/721)	1.081 Day(s)	21.195 Hour(s)
Event3	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[9.26, 20.98)	20.861	15.108 ± 2.866	9.381
	Q::[0.15, 0.38)	0.379	0.242 ± 0.065	0.155
		Feq.	Max. Duration	Min. Duration
		32% (228/721)	2.875 Day(s)	1.125 Day(s)
Event4	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[20.98, 30.01]	30.000	27.205 ± 2.618	21.101
	Q::[0.00, 0.15)	0.149	0.031 ± 0.036	0.001
		Feq.	Max. Duration	Min. Duration
		01% (10/721)	3.000 Hour(s)	3.000 Hour(s)
Event5	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[3.00, 15.00)	6.000	6.000 ± 0.000	6.000
	T::[20.98, 30.01]	29.815	26.857 ± 3.075	21.810
	Q::[0.00, 0.15)	0.113	0.049 ± 0.044	0.007
		Feq.	Max. Duration	Min. Duration
		10% (70/721)	21.231 Hour(s)	5.679 Hour(s)
Event6	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[0.00, 9.26)	9.139	2.828 ± 2.656	0.037
	Q::[0.38, 0.98]	0.598	0.483 ± 0.064	0.380

		Feq.	Max. Duration	Min. Duration
		15% (106/721)	1.162 Day(s)	1.162 Day(s)
Event7	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[0.00, 9.26)	8.661	3.629 ± 3.104	0.057
	Q::[0.15, 0.38)	0.376	0.248 ± 0.065	0.152
		Feq.	Max. Duration	Min. Duration
		12% (84/721)	1.331 Day(s)	1.331 Day(s)
Event8	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[0.00, 9.26)	9.139	3.168 ± 2.581	0.000
	Q::[0.00, 0.15)	0.150	0.102 ± 0.027	0.052
		Feq.	Max. Duration	Min. Duration
		01% (4/721)	2.092 Hour(s)	2.092 Hour(s)
Event9	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[3.00, 15.00)	6.000	6.000 ± 0.000	6.000
	T::[0.00, 9.26)	5.560	4.977 ± 0.674	4.393
	Q::[0.15, 0.38)	0.360	0.357 ± 0.004	0.354
		Feq.	Max. Duration	Min. Duration
		01% (6/721)	3.000 Hour(s)	54.494 min.
Event10	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[3.00, 15.00)	6.000	6.000 ± 0.000	6.000
	T::[0.00, 9.26)	1.635	0.884 ± 0.727	0.021
	Q::[0.38, 0.98]	0.427	0.415 ± 0.010	0.407
		Feq.	Max. Duration	Min. Duration
		00% (2/721)	5.017 Hour(s)	5.017 Hour(s)
Event11	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[9.26, 20.98)	11.626	11.626 ± 0.000	11.626
	Q::[0.38, 0.98]	0.389	0.389 ± 0.000	0.389

		Feq.	Max. Duration	Min. Duration
		00% (1/721)	3.415 Hour(s)	3.415 Hour(s)
Event12	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[15.00, 24.01]	24.000	24.000 ± 0.000	24.000
	T::[20.98, 30.01]	26.657	26.657 ± 0.000	26.657
	Q::[0.38, 0.98]	0.643	0.643 ± 0.000	0.643
		Feq.	Max. Duration	Min. Duration
		02% (12/721)	1.548 Day(s)	1.548 Day(s)
Event13	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[20.98, 30.01]	26.491	25.495 ± 0.673	24.440
	Q::[0.38, 0.98]	0.970	0.711 ± 0.201	0.408
		Feq.	Max. Duration	Min. Duration
		02% (14/721)	1.070 Day(s)	1.070 Day(s)
Event14	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[20.98, 30.01]	24.235	22.634 ± 1.143	21.101
	Q::[0.15, 0.38)	0.367	0.233 ± 0.066	0.151
		Feq.	Max. Duration	Min. Duration
		00% (1/721)	3.000 Hour(s)	3.000 Hour(s)
Event15	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[3.00, 15.00)	6.000	6.000 ± 0.000	6.000
	T::[20.98, 30.01]	21.810	21.810 ± 0.000	21.810
	Q::[0.15, 0.38)	0.214	0.214 ± 0.000	0.214

As stated earlier, this list is not able to represent the results of the derived suffix tree. However, it offers an overview of the features of MetaEvents. For instance, Event12 plays relative unimportant role due to only one occurrence in the history. The components of Event12 are:

- rainfall is high (15.00 mm/day $\leq x <$ 24.01 mm/day)
- temperature is high (20.98 °C $\leq x \leq$ 30.01 °C)

- streamflow is high (0.38 mm/day $\leq x \leq$ 0.98 mm/day)

which is the occurrence at 3:00 AM, 2nd March, 2000. Due to the effect of hysteresis in the rainfall-runoff process, the response should occur afterwards. In searching of the results of the derived suffix tree, the MetaEvent coming after Event12 is Event13 which has components:

- rainfall is low (0.00 mm/day $\leq x <$ 3.00 mm/day)
- temperature is high (20.98 °C $\leq x \leq$ 30.01 °C)
- streamflow is high (0.38 mm/day $\leq x \leq$ 0.98 mm/day)

with duration of 1.548 days in accordance with the development in the data set of HydroTestData.

The default values given to each MetaEvent in comparison with the original data set are shown in Fig. 6.7. An obvious stepwise pattern is noticed, especially in the temperature plot, due to the reason that these values are derived from the averaged occurrences in each corresponding MetaEvent as mentioned earlier. Also, due to the same reason, the default values are not able to match extreme numbers. These values can be later changed with the help of the Graphical User Interface (GUI) mentioned in Section 5.6, if necessary.

The same action is taken for the process identification here. Assuming streamflow is the effect of both rainfall and temperature, there are 15 rules derived including some default settings, e.g. triangular membership functions, CoG for defuzzification, etc., for the purpose of process identification, and the results in comparison with the streamflow data are shown in Fig. 6.8. The results shown in Fig. 6.8 are only showing the process identification in this case which catches the major trend, however, without adequate accuracy (RMSE = 0.137 mm/day).

In this application example, another test with an increased number of categories was carried out to investigate the impacts on the results. In this further test, all parameters, settings, and algorithms are exactly the same as the previous one except the number of categories of the temperature

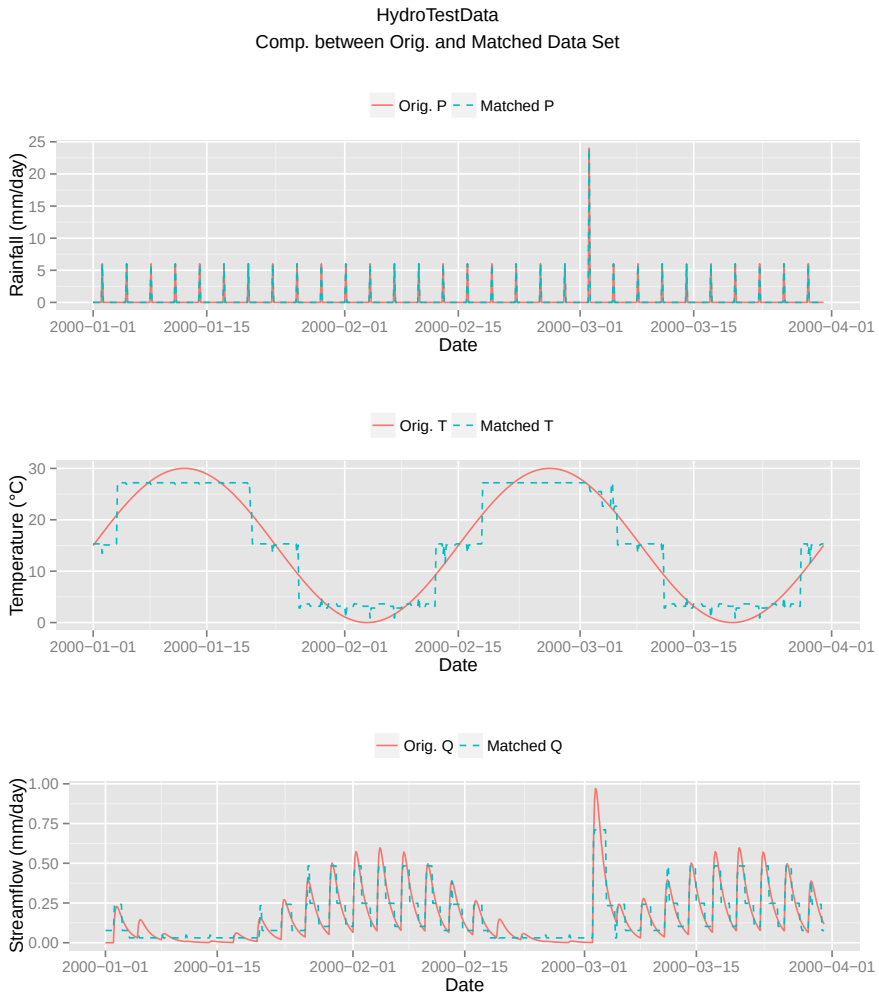


Figure 6.7: Comparing original data set with the matched MetaEvent default values (HydroTestData)

and streamflow data. The number of categories of the temperature and streamflow data is increased to five, and the number of categories of the rainfall data is kept the same at three, due to the reason that the data contain only three different values and can not be further categorized. The categorized results are as:

- Rainfall (mm/day)
  - $0.00 \leq x < 3.00$
  - $3.00 \leq x < 15.00$

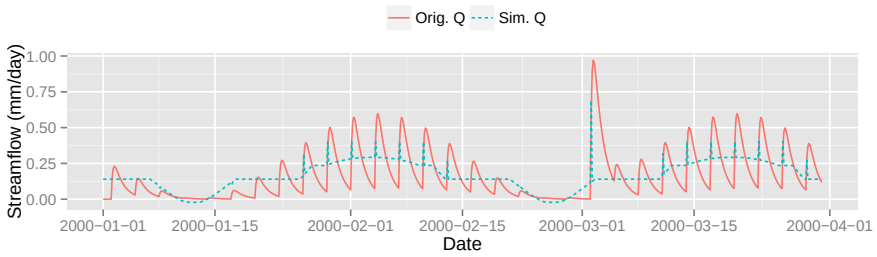


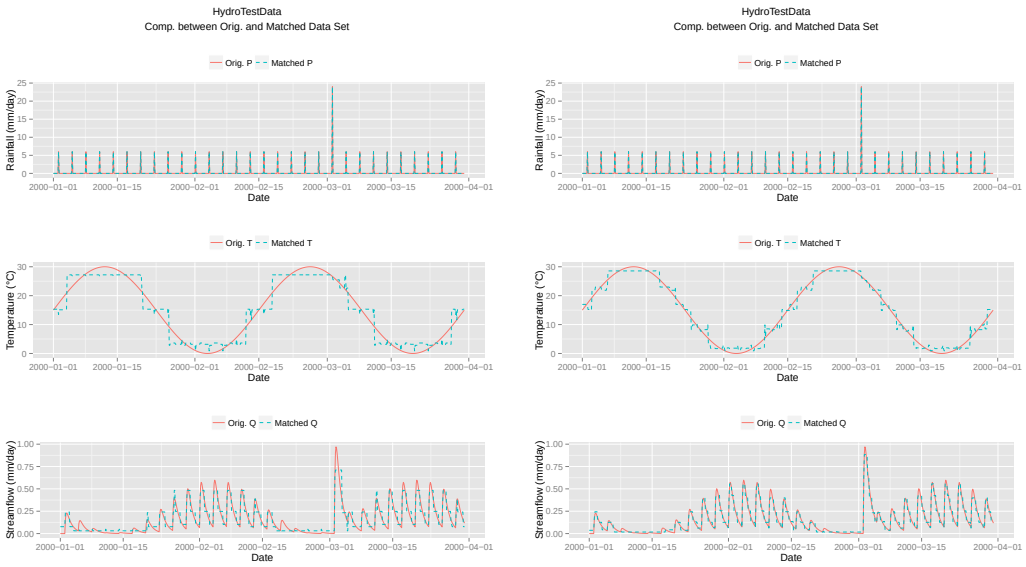
Figure 6.8: Results of the process identification in the data set HydroTest-Data

  - $15.00 \leq x \leq 24.01$

- Temperature (°C)
  - $0.00 \leq x < 5.26$
  - $5.26 \leq x < 12.27$
  - $12.27 \leq x < 19.26$
  - $19.26 \leq x < 25.51$
  - $25.51 \leq x \leq 30.01$
- Streamflow (mm/day)
  - $0.00 \leq x < 0.08$
  - $0.08 \leq x < 0.19$
  - $0.19 \leq x < 0.33$
  - $0.33 \leq x < 0.51$
  - $0.51 \leq x \leq 0.98$

The plots of these results can be also seen in Fig. A.1 in the Appendix A. With the same settings as the previous one, 29 MetaEvents were derived and are described in Table A.1 in the Appendix A.

In order to illustrate the impacts of the number of categories, the comparison of the plots, which compare the original data set with the



(a) Comparison between the original data set with the matched MetaEvent default values (coarse case)

(b) Comparison between the original data set with the matched MetaEvent default values (refined case)

Figure 6.9: The impacts of the number of categories on the MetaEvent default values (HydroTestData)

matched MetaEvent default values of these two test cases, are shown in Fig. 6.9. Where shows two plots of comparing the original data set with the matched MetaEvent default values of these two tests. Similar to the concept of discretization, the default MetaEvent values are closer to the original ones as shown in Fig. 6.9.

(a) Results of process identification with less number of categories

(b) Results of process identification with more number of categories

Figure 6.10: The impacts of the number of categories on the process identification (HydroTestData)

Another comparison of these two tests on the process identification

is shown in Fig. 6.10. With the same settings as the previous ones, the calculated RMSE in this test leads to 0.135 mm/day with the derived 29 rules, and this value does not differ much from the previous one (0.137 mm/day), which catches the trend, however, without adequate accuracy.

Several reasons to explain the results of the process identification can be concluded as:

- Although the data set contains three different physical state variables, the temperature variable plays no importance in the rainfall-runoff relationship.
- The nature of this prototype is not capable of handling the hysteresis effect, since the relationship is built upon the “current” event.
- In the first test, 15 rules are derived but three among them have no impact after optimization; in the second one, five out of 29 rules also show no impact after optimization. These numbers do include other trivial rules yet. In this sense, the number of real effective rules is only a few and can not describe the physical phenomena apart from the reasons above. Although it is possible to further increase the number of rules by increasing the number of categories, it also increases the complexity of the scenario composition. Therefore, it will cause a dilemma between the usability and precision. In this situation, the process identification can be replaced by other methods specifically describing the required relationship, such as Artificial Neural Networks (ANNs), boosting, empirical functions or even physically-based models.

A more generalized discussion will be addressed in Chapter 7.

6.5 BinghamTrib Data Set

In this application example, the BinghamTrib data set from the R package `hydromad` [Andrews and Guillaume, 2012] is used to proceed the same process mentioned earlier. This data set also contains three different physical state variables: rainfall (mm/day), temperature (°C), and streamflow

(mm/day). Among them, rainfall and streamflow data are collected for the Bingham River Trib at Ernies Catchment (2.68 km^2) by the Department of Water, Water Information Provision section, Perth, Western Australia. Temperature data are collected by the Bureau of Meteorology, Australia. This data set contains daily records from 1974-05-18 to 2008-11-02, and the collecting gauge and stations are located at the following positions based on the World Geodetic System (WGS) standard (WGS 84) [Andrews and Guillaume, 2012]:

- (-33.2921, 116.4451) for the rain gauge station
- (-33.2939, 116.4449) for the stream gauge
- (-33.57, 115.82) for the meteorological station of 63 m height

as shown in the Fig. 6.11.

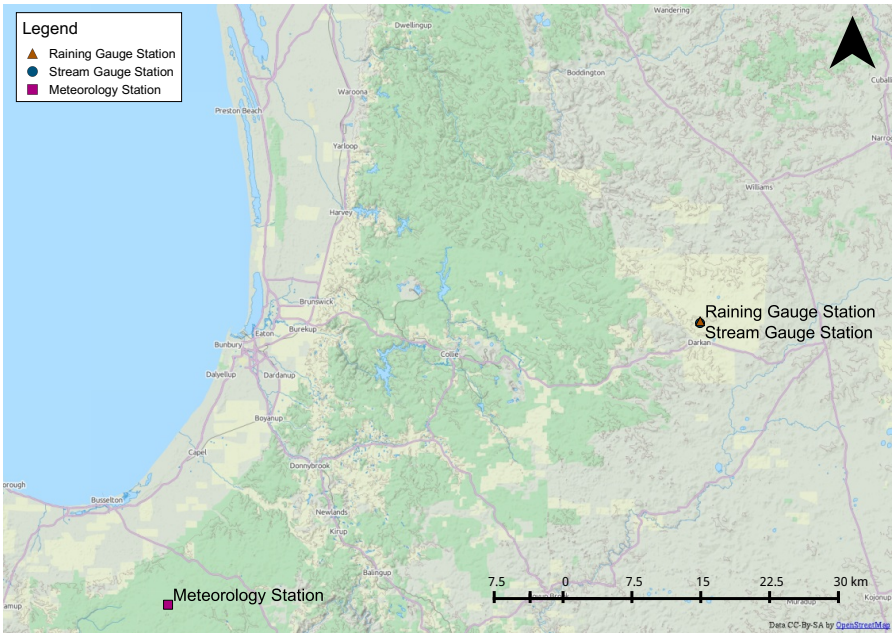


Figure 6.11: Study area and gauging stations of the BinghamTrib data set

The time series plots for the data set are shown in Fig. 6.12 and the summary of the data set is described in Table 6.5. As shown in Fig. 6.12 and Table 6.5, the rainfall data have a mean value of 1.97 mm/day, a maximum

value of 142.60 mm/day, a minimum value of 0.00 mm/day and 154 missing records; the temperature data have a mean value of 23.27 °C, a maximum value of 34.30 °C, and a minimum value of 15.50 °C without missing records; the streamflow data have a mean value of 0.02 mm/day, a maximum value of 4.81 mm/day, and a minimum value of 0.00 mm/day without missing records.

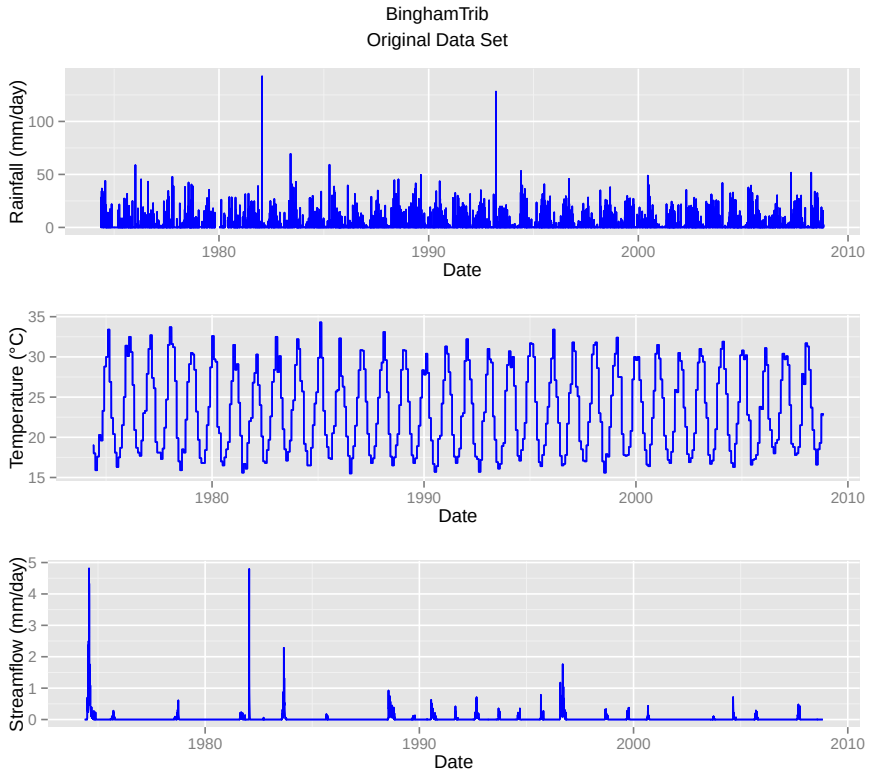


Figure 6.12: Original BinghamTrib data set from the R package `hydromad` [Andrews and Guillaume, 2012]

Table 6.5: Description of the BinghamTrib data set

	no.	mean	sd	median	min	max	range
Rainfall	12434	1.97	5.53	0.00	0.00	142.60	142.60
Temperature	12588	23.27	5.19	22.40	15.50	34.30	18.80
Streamflow	12588	0.02	0.13	0.00	0.00	4.81	4.81

Before performing the event identification, a simple data pre-processing is carried out to deal with the missing records which appear in the rainfall data. These gaps lie in the intervals of 1979-02-06 – 1979-02-20, 1979-10-30 – 1980-01-10, 1980-04-20 – 1980-05-30, 1980-07-08 – 1980-07-10, 1980-08-19 – 1980-09-03, and 1981-01-14 – 1981-01-19. By inspecting the history and the neighboring values next to these gaps, these missing records are replaced by the value of zero because most historical records and most neighboring values are also low values.

For the same reasons as in the previous application examples, these three physical time series are viewed as three individual Aspects. Furthermore, different categories are decided in the purpose of properly and evenly representing different properties of Aspects by giving different k values in the k-means clustering algorithm, and the results are shown as follows:

- Rainfall (mm/day)
  - $0.00 \leq x < 3.85$
  - $3.85 \leq x < 12.55$
  - $12.55 \leq x < 26.30$
  - $26.30 \leq x < 103.10$
  - $103.10 \leq x \leq 142.61$
- Temperature (°C)
  - $15.50 \leq x < 18.35$
  - $18.35 \leq x < 21.35$
  - $21.35 \leq x < 25.15$
  - $25.15 \leq x < 28.95$
  - $28.95 \leq x \leq 34.31$
- Streamflow (mm/day)
  - $0.00 \leq x < 0.34$

- $0.34 \leq x < 2.01$
- $2.01 \leq x \leq 4.82$

where x represents any physical state variable used here, and the results are shown in Fig. 6.13.

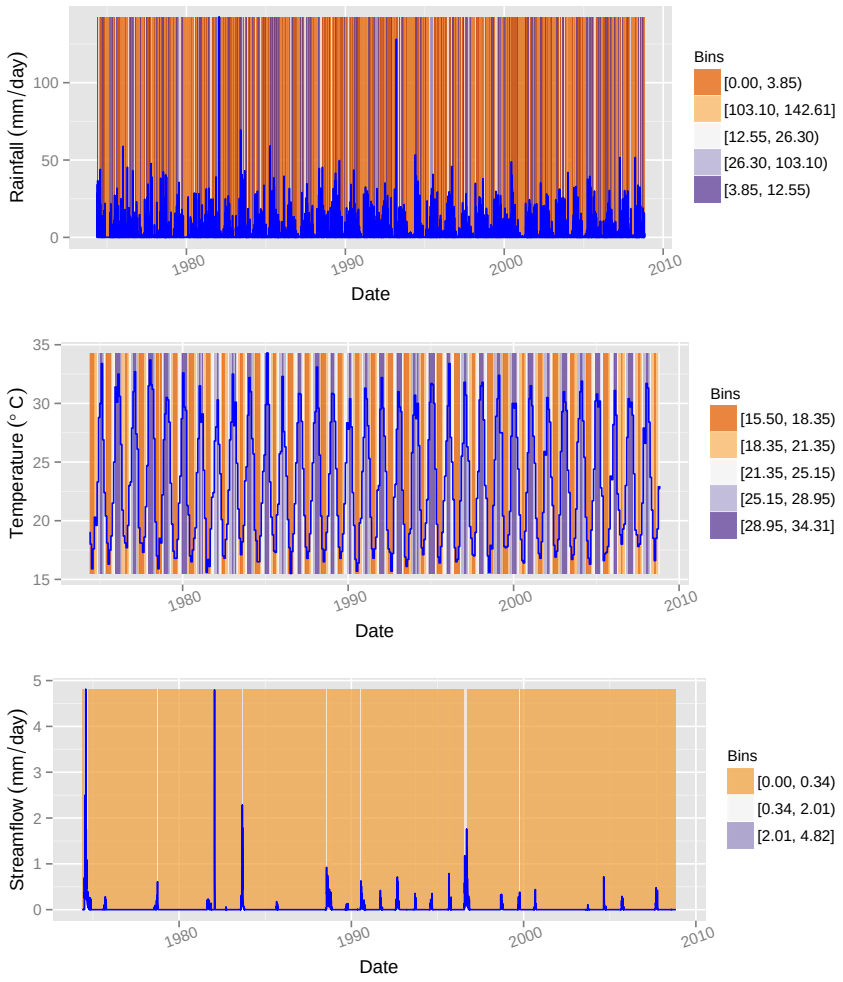


Figure 6.13: Derived Tones from the BinghamTrib data set

The MetaEvent are derived based on these Tones with the same settings as used in the previous application examples. In total, there are 42 MetaEvents. An extraction of these MetaEvents is shown in Table 6.6 and the complete list is found in Table A.2 in the Appendix A. The symbols, P, T, and

Q, in the list represent rainfall, temperature, and streamflow accordingly. In this case, the number of MetaEvents increases due to the more complicated data distribution as well as more Tones involved in the process.

Table 6.6: Extraction of the generated MetaEvents based on the BinghamTrib data set

		Feq.	Max. Duration	Min. Duration
		17% (2157/12588)	5.985 Week(s)	5.135 Day(s)
Event1	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.85)	3.800	0.443 ± 0.854	0.000
	T::[18.35, 21.35)	21.300	19.626 ± 0.871	18.400
	Q::[0.00, 0.34)	0.320	0.013 ± 0.036	0.000
		⋮		
		Feq.	Max. Duration	Min. Duration
		01% (115/12588)	3.018 Day(s)	1.745 Day(s)
Event3	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[12.55, 26.30)	25.800	17.364 ± 3.510	12.600
	T::[18.35, 21.35)	21.300	19.489 ± 0.872	18.400
	Q::[0.00, 0.34)	0.310	0.035 ± 0.075	0.000
		Feq.	Max. Duration	Min. Duration
		02% (305/12588)	4.266 Day(s)	17.795 Hour(s)
Event4	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[3.85, 12.55)	12.500	7.171 ± 2.475	3.900
	T::[18.35, 21.35)	21.300	19.491 ± 0.837	18.400
	Q::[0.00, 0.34)	0.330	0.033 ± 0.066	0.000
		⋮		

		Feq.	Max. Duration	Min. Duration
		00% (0/12588)	17.130 Hour(s)	17.130 Hour(s)
Event16	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.85)	NaN	NaN ± NaN	NaN
	T::[15.50, 18.35)	NaN	NaN ± NaN	NaN
	Q::[2.01, 4.82]	NaN	NaN ± NaN	NaN
		⋮		
		Feq.	Max. Duration	Min. Duration
		16% (2022/12588)	8.633 Week(s)	4.398 Week(s)
Event20	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.85)	3.800	0.156 ± 0.532	0.000
	T::[25.15, 28.95)	28.900	27.218 ± 1.058	25.200
	Q::[0.00, 0.34)	0.331	0.000 ± 0.010	0.000
		Feq.	Max. Duration	Min. Duration
		20% (2579/12588)	12.962 Week(s)	8.509 Week(s)
Event21	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.85)	3.800	0.094 ± 0.406	0.000
	T::[28.95, 34.31]	34.300	30.732 ± 1.183	29.000
	Q::[0.00, 0.34)	0.039	0.000 ± 0.001	0.000
		Feq.	Max. Duration	Min. Duration
		00% (23/12588)	1.485 Day(s)	1.485 Day(s)
Event22	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[12.55, 26.30)	23.800	17.452 ± 3.272	12.600
	T::[25.15, 28.95)	28.900	27.028 ± 1.196	25.200
	Q::[0.00, 0.34)	0.222	0.010 ± 0.046	0.000

		Feq.	Max. Duration	Min. Duration
		01% (76/12588)	2.485 Day(s)	22.796 Hour(s)
Event23	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[3.85, 12.55)	12.500	7.334 ± 2.527	3.900
	T::[25.15, 28.95)	28.600	26.918 ± 1.016	25.200
	Q::[0.00, 0.34)	0.000	0.000 ± 0.000	0.000
		⋮		
		Feq.	Max. Duration	Min. Duration
		00% (8/12588)	1.533 Day(s)	1.533 Day(s)
Event28	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[26.30, 103.10)	78.000	42.188 ± 16.730	27.400
	T::[25.15, 28.95)	28.600	26.963 ± 1.345	25.300
	Q::[0.00, 0.34)	0.000	0.000 ± 0.000	0.000
		⋮		
		Feq.	Max. Duration	Min. Duration
		00% (0/12588)	6.382 Hour(s)	6.382 Hour(s)
Event32	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[26.30, 103.10)	NaN	NaN ± NaN	NaN
	T::[25.15, 28.95)	NaN	NaN ± NaN	NaN
	Q::[2.01, 4.82]	NaN	NaN ± NaN	NaN
		⋮		
		Feq.	Max. Duration	Min. Duration
		00% (0/12588)	13.764 Hour(s)	13.764 Hour(s)
Event35	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.85)	NaN	NaN ± NaN	NaN
	T::[25.15, 28.95)	NaN	NaN ± NaN	NaN
	Q::[2.01, 4.82]	NaN	NaN ± NaN	NaN
		⋮		

		Feq.	Max. Duration	Min. Duration
		00% (0/12588)	3.988 Hour(s)	3.988 Hour(s)
Event42	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.85)	NaN	NaN ± NaN	NaN
	T::[21.35, 25.15)	NaN	NaN ± NaN	NaN
	Q::[0.34, 2.01)	NaN	NaN ± NaN	NaN

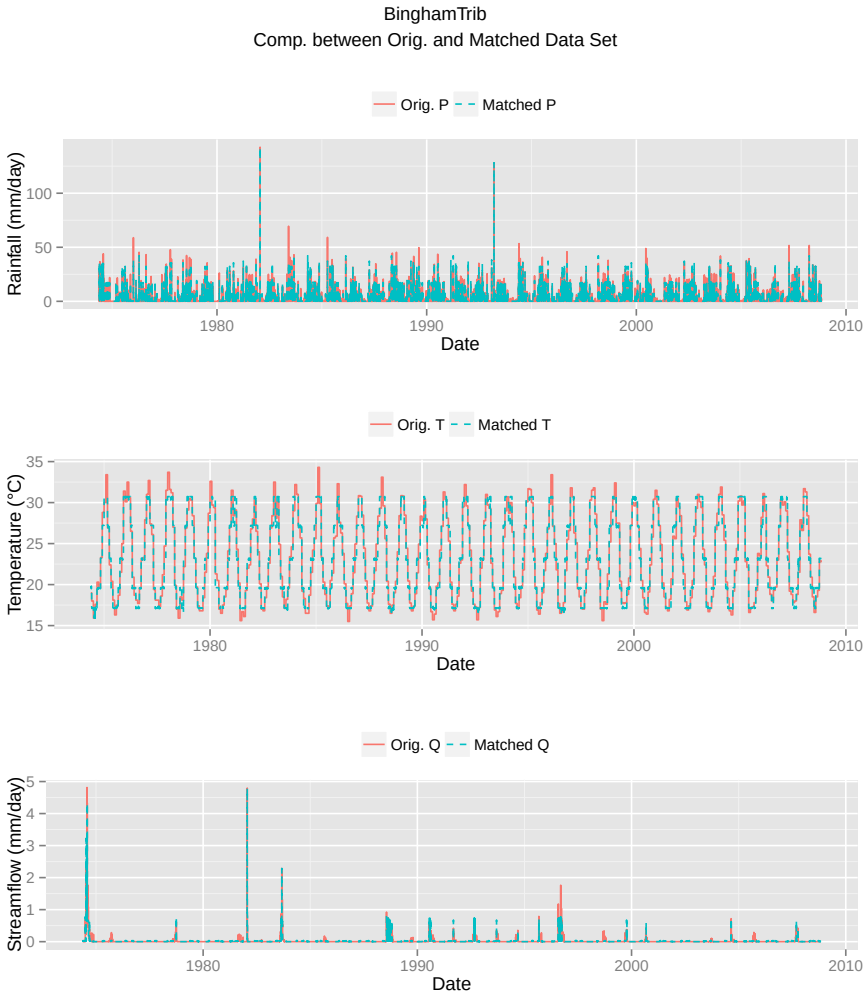


Figure 6.14: Comparing original data set with the matched MetaEvent default values (BinghamTrib)

While reviewing Table 6.6, there are MetaEvents, e.g. Event16, containing no information of statistics and these types of information are expressed by the representation of Not A Number (NaN). These MetaEvents generally have a duration of a smaller period, the unit of hours in this case, compared to the majority of MetaEvents in Table A.2. No record in history is found to match these MetaEvents. Apart from this, this list of generated MetaEvents also contains all the properties mentioned earlier. In Fig. 6.14, the comparison between the default assigned values of MetaEvents and the original data are shown. In this case, the original data set are better represented since more MetaEvents are generated, hence less stepwise is displayed.

Similarly, an operation of process identification is carried out in this application example. In this case, the assumption that streamflow is the effect of both rainfall and temperature is still valid. The same default settings used in previous example applications together with 42 generated fuzzy rules are applied. Under this condition, the comparison between the generated results in the process identification and the original streamflow data is shown in Fig. 6.15. The accuracy (RMSE = 0.130 mm/day) in this case is still not considered sufficient and the possible reasons are the same as discussed earlier.

Figure 6.15: Results of the process identification in the BinghamTrib data set

Unlike as in the previous application examples, a demonstration of the scenario composition is given here. Since the composition of scenarios can

be done arbitrarily, here a scenario of a three-month period (March to May) is composed. Before composing a scenario, a basic investigation of the history is carried out to have an overview of MetaEvents composition within each month which is not availabe in the Table A.2 nor in the GUI. According to the historical data, the list of MetaEvents within each month in the lexicographical order is as follows:

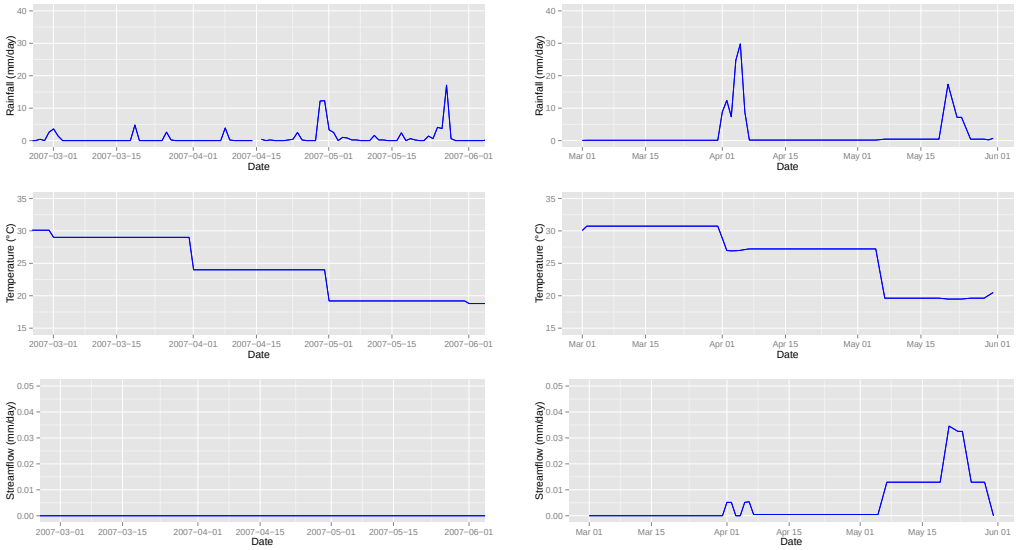
- March: Event20, Event21, Event22, Event23, Event26, Event27, Event28, Event41
- April: Event1, Event3, Event4, Event18, Event19, Event20, Event21, Event22, Event23, Event24, Event28, Event31
- May: Event1, Event2, Event3, Event4, Event18, Event19, Event24, Event31

In the process of scenario composition, not only the information in Table A.2 and the results of the suffix tree are referred to, but additional results of a regular expression query of history are also applied as supplementary information. The reason why another additional query is also applied in this application example is due to the limitation of the implementation dealing with overlapping Chords in the scenario composition mentioned in Section 5.6. It depends on the complexity of the Tones composition in each Aspect. Some orderings might be neglected due to the “chopping” process implemented in the scenario composition. In this case, an additional query provides more detailed information if needed.

Based on the information provided from different sources mentioned above, an example of a three-month scenario composition, from 1st March to 30th May, is composed as shown in Table 6.7 with information of the MetaEvent, the start time, the end time, and the duration. In this example, an attempt to keep the decreasing trend in temperature similar to history is addressed. Besides, some additional rainfall events are deliberately inserted following the suggestion of the MetaEvent sequence from the system and query results. Under this circumstances, a generated time series based on

Table 6.7: Composition of the three-month scenario

MetaEvent	Start Time	End Time	Duration (Day)
Event21	03-01	04-01	31
Event22	04-01	04-02	1
Event23	04-02	04-04	2
Event28	04-04	04-05	1
Event22	04-05	04-06	1
Event20	04-06	05-06	30
Event1	05-06	05-20	14
Event3	05-20	05-22	2
Event4	05-22	05-25	3
Event1	05-25	05-30	5



(a) Three-month extraction (March to May) in year 2007 from the BinghamTrib data set

(b) Generated three-month (March to May) time series based on Table 6.7

Figure 6.16: Time series plots of the original data set extraction and the composed scenario

the default values in each MetaEvent is shown in Fig. 6.16b and a three-month temporal extraction, from 2007-03-01 to 2007-05-30, is shown in Fig.

6.16a as a reference. Although the appropriateness of the composed scenario has to be further examined by the domain experts, the generated time series plot does represent a reasonable physical response of the rainfall-runoff effect, and further modifications can be made to improve the generated time series.

6.6 Oder River Data

The purpose of this application example is to demonstrate how the framework can be used with other simulation tools. For this reason, the focus of this application example lies in generating time series data of a specific user-defined scenario from the framework, and applying these generated data as boundary conditions in the selected simulation tool. The data set chosen for this example is a part of the data which describe the 1997 Oder Flood. This flood occurred in July 1997 and was caused by two successive extreme rainfall events. According to [Landesumweltamt Brandenburg (LUA), 1997], these two extreme events occurred during two time periods: from 1997-07-03 to 1997-07-09 and from 1997-07-18 to 1997-07-22. During the first extreme rainfall event, the highest precipitation was measured at the station in Lysá hora, Czech Republic with the value of 586 mm. This flood caused great damages in Poland, Germany, and the Czech Republic. Apart from [Landesumweltamt Brandenburg (LUA), 1997], there are several studies and discussions referring to this flood event, such as [European Communities, 1999; Grünewald, 1998; Kundzewicz, 2007; Kundzewicz et al., 1999; Plate, 2002].

The study area of this example is the part of this Oder river, starting from the town of Eisenhüttenstadt to the city of Frankfurt (Oder), on the border between Germany and Poland, as shown in Fig. 6.17. The distance of this river section is about 30 km long and it covers an area of around 85 km^2. During this flood event, a dam breach occurred and further caused the flooding in the area of the Ziltendorf lowlands.

For the purpose of demonstration, this flood event is simplified into only

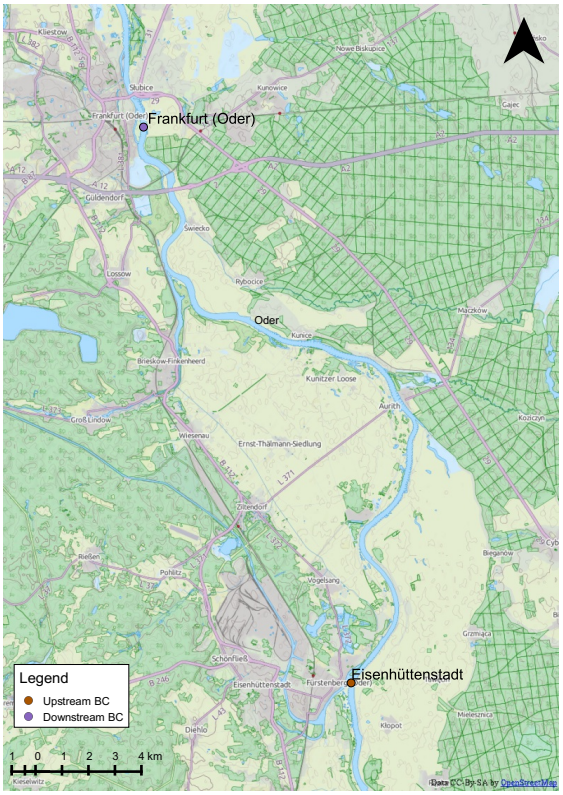


Figure 6.17: Study area and boundary condition locations for the Oder river data

considering the flow condition of the river from the town of Eisenhüttenstadt to the city of Frankfurt (Oder) without the dam breach. Hence, the Mike 11 is chosen as the 1-D simulation tool, which works with the framework, for this demonstration. The time series data collected for this example contain:

- discharge data (m^3/s) at daily intervals from 1996-01-11 to 1997-11-01 at Eisenhüttenstadt
- water level data (m) at 15-minute intervals from 1996-11-01 to 1997-11-02 at Eisenhüttenstadt
- water level (m) at 15-minute intervals from 1996-11-01 to 1997-11-02 at Frankfurt (Oder)

where the daily discharge data at Eisenhüttenstadt were processed by the Wasser- und Schifffahrtsamt Eberswalde with its own rating curve. In

addition, the cross section data at 2500 m intervals for the simulation task were derived from the 25 m × 25 m bathymetry data also provided by Wasser- und Schifffahrtsamt Eberswalde.

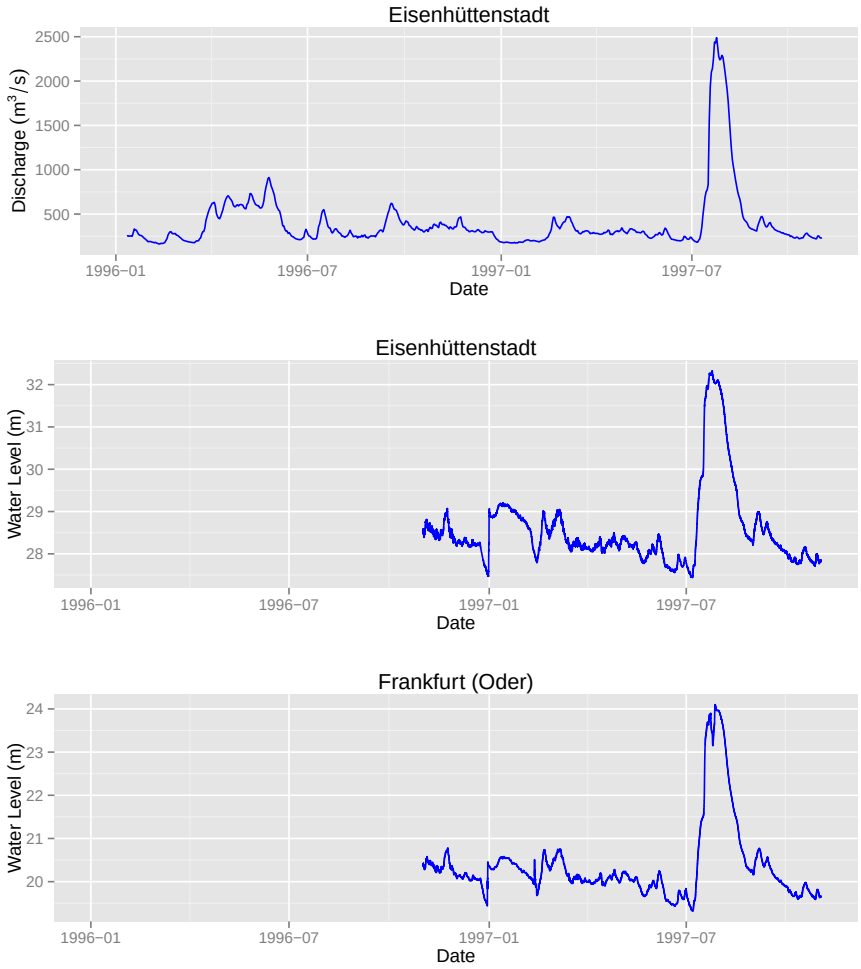


Figure 6.18: Measured time series data at Eisenhüttenstadt and Frankfurt (Oder)

The time series plots for each data set are shown in Fig. 6.18, and a basic statistical summary is described in Table 6.8. From Fig. 6.18 and Table 6.8, two major structural differences can be observed: time span and resolution. The discharge data set at Eisenhüttenstadt has longer time span than the water level data sets measured at Eisenhüttenstadt and Frankfurt (Oder). In addition, the resolution of the discharge data set at Eisenhüttenstadt

is coarser than the water level data set at either Eisenhüttenstadt or Frankfurt (Oder). Apart from these, the discharge data at Eisenhüttenstadt have a mean value of 397.05 m^3/s, maximum value of 2490.00 m^3/s, and a minimum value of 162.00 m^3/s without missing records; the water level data at Eisenhüttenstadt have a mean value of 28.57 m, a maximum value of 32.32 m, and a minimum value of 27.44 m without missing records; the water level data at Frankfurt (Oder) have a mean value of 20.37 m, a maximum value of 24.10 m, and a minimum value of 19.32 m without missing records. Moreover, the peaks happened in July, 1997 also indicate the event of the 1997 Oder Flood. Also, it has to be kept in mind that these data were measured during the flood event. Hence, the water level at Frankfurt (Oder) after the flood event may not be well-simulated without considering the dam breach between Eisenhüttenstadt and Frankfurt (Oder).

Table 6.8: Description of the Oder River data

	no.	mean	sd	median	min	max	range
Q at Ei[a]	661	397.05	348.55	301.00	162.00	2490.00	2328.00
H at Ei[b]	35136	28.57	0.96	28.29	27.44	32.32	4.88
H at Fr[c]	35136	20.37	0.92	20.14	19.32	24.10	4.78

[a] Discharge at Eisenhüttenstadt [b] Water level at Eisenhüttenstadt
[c] Water level at Frankfurt (Oder)

Before carrying out this application example, these data have to be processed to resolve the issues mentioned earlier: time span and resolution. Firstly, these time series data sets were truncated to fit the time span from 1996-11-01 12:00:00 to 1997-11-01 12:00:00. In this case, they all now have the same time span. Then, the daily discharge data at Eisenhüttenstadt were interpolated into the ones at 15-minute intervals by the `spline` function in R with default settings and methods. Since only two boundary conditions are needed for the one-dimensional shallow water equation, only two time series data sets, the discharge at Eisenhüttenstadt as upstream boundary condition and the water level at Frankfurt (Oder) as downstream boundary condition,

are chosen for categorization. Like previous application examples, the k-means clustering algorithm with default system settings is chosen for the task. As shown in the following list, the discharge at Eisenhüttenstadt was categorized into five categories and the water level at Frankfurt (Oder) was categorized into three categories. The reason behind these categorizations is mainly to have a more categories for the abrupt increase in the discharge at Eisenhüttenstadt and the odd number fits the convention how engineers categorize properties. The categorized results are as follows:

- Discharge at Eisenhüttenstadt (m^3/s)
  - $173.54 \leq x < 295.52$
  - $295.52 \leq x < 588.60$
  - $588.60 \leq x < 1207.85$
  - $1207.85 \leq x < 1917.07$
  - $1917.07 \leq x \leq 2490.01$
- Water level at Frankfurt (Oder) (m)
  - $19.32 \leq x < 20.24$
  - $20.24 \leq x < 21.96$
  - $21.96 \leq x \leq 24.11$

where the variable x represents any physical state variable here, and these categorization results are shown in Fig. 6.19. Unlike the water level time series at Frankfurt (Oder) with three bins indicating high, median, and low, the discharge time series at Eisenhüttenstadt were categorized into five bins. The main reason is to differentiate the differences during the normal flow condition. In addition, only two data sets are involved in this categorization. Hence, the number of the derived MetaEvents is less compared to those in the previous application examples due to the finite combination, and the derived nine MetaEvents are shown in Table 6.9. The symbols, Q_Ei and H_Fr, used in the description of the rules denote the discharge at Eisenhüttenstadt and the water level at Frankfurt (Oder)

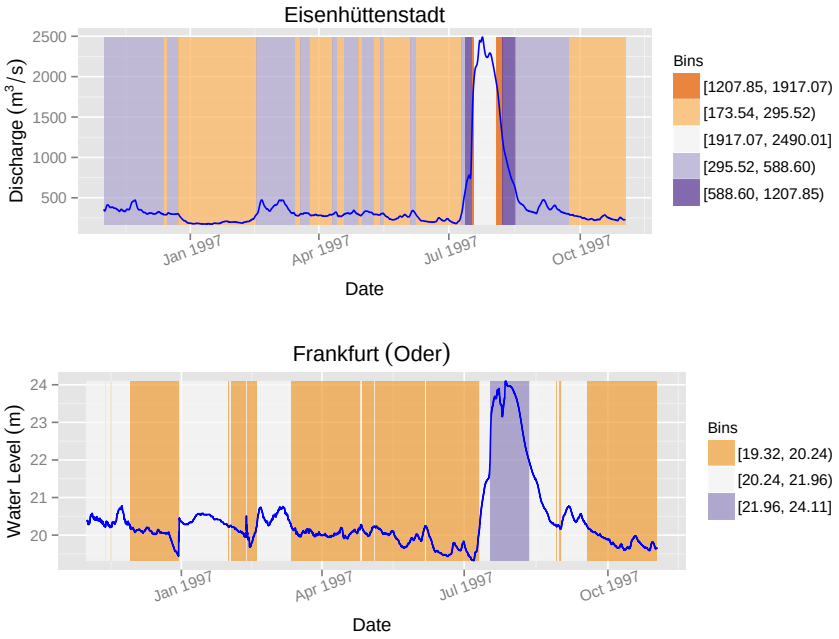


Figure 6.19: Derived tones from Oder river data

accordingly. While reviewing Table 6.9, it can be noticed that no NaN can be found when compared to the results in the application example of the BinghamTrib data set. The reason is that the minimum duration of these MetaEvents is about 8.25 hours and the interval of the data set is 15 minutes. For this reason, inside the interval of each MetaEvent, there must be at least one record for the purpose of statistical description in Table 6.9.

Table 6.9: List of 9 generated MetaEvents based on the Oder river data

		Feq.	Max. Duration	Min. Duration
Event1		22% (7864/35041)	3.164 Week(s)	1.660 Week(s)
	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	Q_Ei::[295.52, 588.60)	588.177	391.104 ± 51.326	308.231
	H_Fr::[20.24, 21.96)	21.370	20.489 ± 0.199	20.240

Event2		Feq.	Max. Duration	Min. Duration
		19% (6579/35041)	1.980 Week(s)	8.250 Hour(s)
	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	Q_Ei::[295.52, 588.60)	404.946	311.448 ± 13.359	295.533
	H_Fr::[19.32, 20.24)	20.230	20.121 ± 0.060	19.820
Event3		Feq.	Max. Duration	Min. Duration
		40% (14069/35041)	5.607 Week(s)	2.637 Day(s)
	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	Q_Ei::[173.54, 295.52)	295.507	247.268 ± 32.935	180.006
	H_Fr::[19.32, 20.24)	20.230	19.827 ± 0.203	19.320
Event4		Feq.	Max. Duration	Min. Duration
		09% (3137/35041)	4.469 Week(s)	4.469 Week(s)
	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	Q_Ei::[173.54, 295.52)	215.353	186.649 ± 10.929	173.545
	H_Fr::[20.24, 21.96)	20.580	20.415 ± 0.106	20.240
Event5		Feq.	Max. Duration	Min. Duration
		03% (932/35041)	5.219 Day(s)	4.552 Day(s)
	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	Q_Ei::[588.60, 1207.85)	1203.170	740.848 ± 95.157	589.030
	H_Fr::[20.24, 21.96)	21.950	21.481 ± 0.243	20.860
Event6		Feq.	Max. Duration	Min. Duration
		00% (46/35041)	11.737 Hour(s)	11.737 Hour(s)
	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	Q_Ei::[1207.85, 1917.07)	1570.484	1397.636 ± 108.051	1211.647
	H_Fr::[20.24, 21.96)	21.940	21.737 ± 0.107	21.590
Event7		Feq.	Max. Duration	Min. Duration
		01% (503/35041)	4.375 Day(s)	20.495 Hour(s)
	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	Q_Ei::[1207.85, 1917.07)	1916.527	1612.250 ± 213.989	1208.606
	H_Fr::[21.96, 24.11]	23.650	23.135 ± 0.364	21.960

		Feq.	Max. Duration	Min. Duration
Event8		04% (1530/35041)	2.277 Week(s)	2.277 Week(s)
	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	Q_Ei::[1917.07, 2490.01]	2490.000	2239.400 ± 148.566	1917.607
	H_Fr::[21.96, 24.11]	24.100	23.723 ± 0.235	23.150
		Feq.	Max. Duration	Min. Duration
Event9		01% (381/35041)	3.906 Day(s)	3.906 Day(s)
	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	Q_Ei::[588.60, 1207.85)	1207.090	1020.706 ± 91.616	870.710
	H_Fr::[21.96, 24.11]	22.690	22.281 ± 0.210	21.960

To demonstrate the usage of the framework together with a simulation tool, a calibrated and validated 1-D Mike 11 model has to be prepared in advance. For the purpose of simplicity and demonstration, the dam breach is neglected in this demonstration, and only one branch starting from Eisenhüttenstadt to Frankfurt (Oder) in the study area was simulated as described earlier. Moreover, the derived Manning's roughness coefficient is about 0.030. In order to show the framework's capability of creating any user-specified scenario, a one-year scenario of two successive peaks, starting from 1996-11-01 12:00:00 to 1997-11-01 12:00:00, compared to the existing one-peak event (Fig. 6.18) was designed and composed based on Table 6.10. Moreover, two time series sets of 15-minute intervals serving as upstream and downstream boundary conditions were generated and are shown in Fig. 6.20. Here, no additional modification, such as artificially setting up specific values, curve smoothing, etc., was carried out in terms of the time series generation. Therefore, a stepwise pattern can be observed in Fig. 6.20, which can also be observed in previous application examples mentioned earlier. While observing these two boundary conditions in Fig. 6.20, the positive correlation between the upstream discharge and the downstream water level can be observed — upstream high flow discharge causes downstream high water level. This is because both involved time series data sets are holistically taken into consideration during the derivation of MetaEvents.

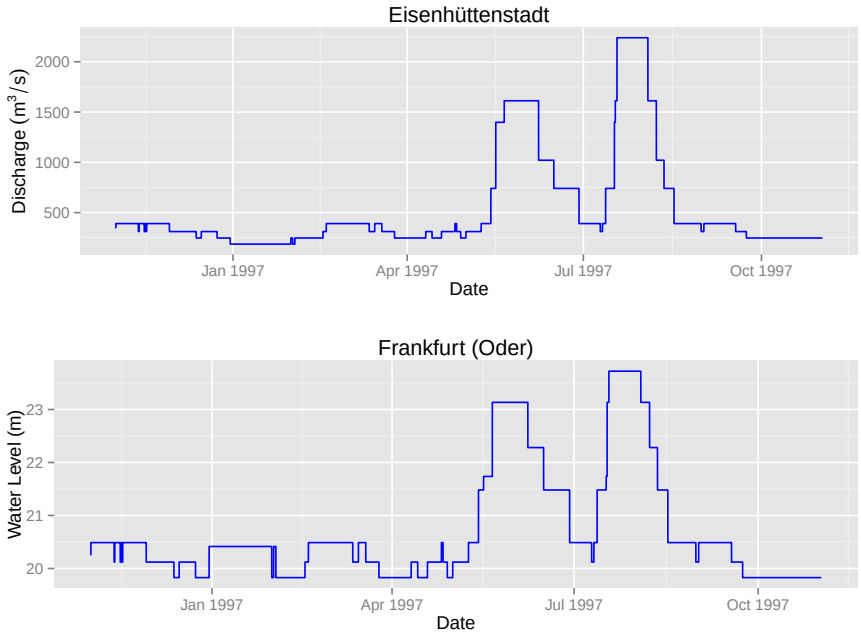


Figure 6.20: Generated two-peak time series based on Table 6.10

Table 6.10: Composition of the two-peak scenario

MetaEvent	Start Time	End Time
Event1	1996-11-01 12:00:00	1996-11-13 03:00:00
Event2	1996-11-13 03:00:00	1996-11-13 11:00:00
Event1	1996-11-13 11:00:00	1996-11-16 04:00:00
Event2	1996-11-16 04:00:00	1996-11-16 17:00:00
Event1	1996-11-16 17:00:00	1996-11-17 03:00:00
Event2	1996-11-17 03:00:00	1996-11-17 10:00:00
Event1	1996-11-17 10:00:00	1996-11-29 04:00:00
Event2	1996-11-29 04:00:00	1996-12-13 00:00:00
Event3	1996-12-13 00:00:00	1996-12-15 15:00:00
Event2	1996-12-15 15:00:00	1996-12-23 19:00:00
Event3	1996-12-23 19:00:00	1996-12-30 14:00:00
Event4	1996-12-30 14:00:00	1997-01-30 21:00:00

Event3	1997-01-30 21:00:00	1997-01-31 20:00:00
Event4	1997-01-31 20:00:00	1997-02-01 23:00:00
Event3	1997-02-01 23:00:00	1997-02-16 12:00:00
Event2	1997-02-16 12:00:00	1997-02-18 05:00:00
Event1	1997-02-18 05:00:00	1997-03-12 09:00:00
Event2	1997-03-12 09:00:00	1997-03-15 05:00:00
Event1	1997-03-15 05:00:00	1997-03-18 21:00:00
Event2	1997-03-18 21:00:00	1997-03-25 11:00:00
Event3	1997-03-25 11:00:00	1997-04-10 14:00:00
Event2	1997-04-10 14:00:00	1997-04-13 20:00:00
Event3	1997-04-13 20:00:00	1997-04-18 17:00:00
Event2	1997-04-18 17:00:00	1997-04-25 16:00:00
Event1	1997-04-25 16:00:00	1997-04-26 12:00:00
Event2	1997-04-26 12:00:00	1997-04-28 15:00:00
Event3	1997-04-28 15:00:00	1997-05-01 08:00:00
Event2	1997-05-01 08:00:00	1997-05-09 05:00:00
Event1	1997-05-09 05:00:00	1997-05-14 05:00:00
Event5	1997-05-14 05:00:00	1997-05-16 18:00:00
Event6	1997-05-16 18:00:00	1997-05-21 03:00:00
Event7	1997-05-21 03:00:00	1997-06-07 21:00:00
Event9	1997-06-07 21:00:00	1997-06-15 18:00:00
Event5	1997-06-15 18:00:00	1997-06-28 18:00:00
Event1	1997-06-28 18:00:00	1997-07-09 18:00:00
Event2	1997-07-09 18:00:00	1997-07-10 21:00:00
Event1	1997-07-10 21:00:00	1997-07-12 12:00:00
Event5	1997-07-12 12:00:00	1997-07-17 01:00:00
Event6	1997-07-17 01:00:00	1997-07-17 12:00:00
Event7	1997-07-17 12:00:00	1997-07-18 09:00:00
Event8	1997-07-18 09:00:00	1997-08-03 08:00:00
Event7	1997-08-03 08:00:00	1997-08-07 17:00:00
Event9	1997-08-07 17:00:00	1997-08-11 16:00:00
Event5	1997-08-11 16:00:00	1997-08-16 20:00:00
Event1	1997-08-16 20:00:00	1997-08-30 20:00:00

Event2	1997-08-30 20:00:00	1997-09-01 06:00:00
Event1	1997-09-01 06:00:00	1997-09-17 16:00:00
Event2	1997-09-17 16:00:00	1997-09-23 05:00:00
Event3	1997-09-23 05:00:00	1997-11-01 12:00:00

These generated time series data, shown in Fig. 6.20, served as boundary conditions for the simulation task conducted by Mike 11. With the predefined parameters, such as Manning's roughness coefficient, cross section data, etc., a simulation was carried out and the plots of 15-minute intervals of the simulated water level at Eisenhüttenstadt and the discharge at Frankfurt (Oder) are shown in Fig. 6.21 and Fig. 6.22 respectively. The simulated results in Fig. 6.21 and Fig. 6.22 also appear in a stepwise pattern, and this is caused by the default characteristics of the boundary conditions generated from the framework.

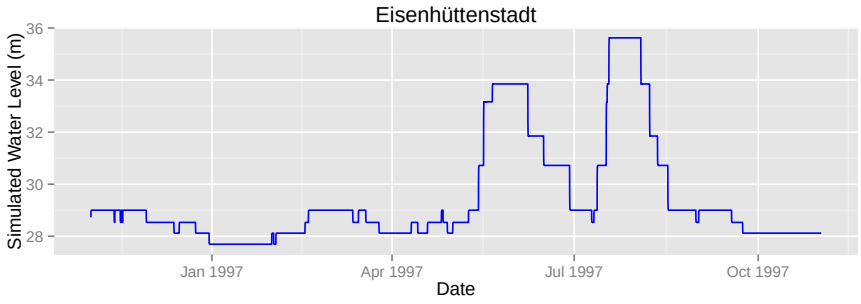


Figure 6.21: Simulated water level at Eisenhüttenstadt

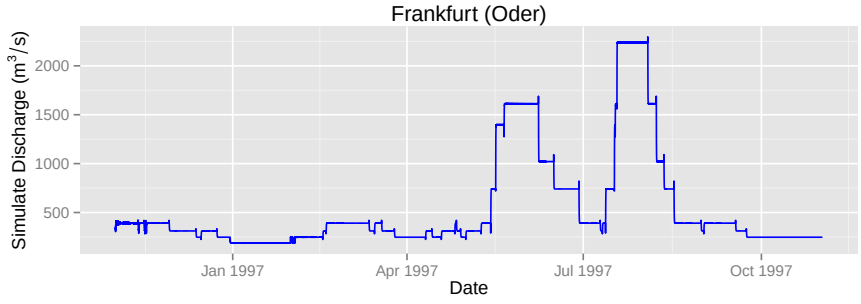


Figure 6.22: Simulated discharge at Frankfurt (Oder)

As mentioned at the beginning of this section, the focus of this application example is to demonstrate how this framework can be used with other simulation tools. As the results presented in this section, the prototype demonstrates the capability of the framework. Further discussions regarding the evaluation of this prototype for future improvements will be addressed in the coming Chapter 7.

Chapter 7

Framework Prototype Evaluation

7.1 General Statement

This framework, as already mentioned in Section 1.3, is aiming to assist simulation tasks mainly in hydro science and engineering disciplines with the support of the information from user-composed scenarios. This information contains synthetic time series data sets generated from user-specified scenarios as inputs, such as, Boundary Conditions (BCs), for simulation tasks. This framework is formed by four modules, data pre-processing, event identification, process identification and scenario composition, as shown in Fig. 1.2. Each module is based on a different concept and has a different objective.

Hence, the following discussion on the framework evaluation will be module-based and on the basis of the results of the application examples carried out from the prototype of this framework described in Chapter 6. In the following discussion, three modules, event identification, process identification, and scenario composition, will be evaluated mainly on the concepts, implementation, limitations, and the results of the application examples in addition to possible suggestions for improvement. Although the module of data pre-processing plays an important role in the framework, it is left out in the following discussion because it has less to do with the main concepts of the framework described in Chapter 4.

7.2 Evaluation on Event Identification

The purpose of event identification is to identify Events from gathered time series data, and these Events together with historical data can form MetaEvents for scenario composition. In this sense, the results of event identification, Events, represent groups of the facts and features in the study case. The concept of event identification is based on the framework of Time Series Knowledge Mining (TSKM).

Before evaluating event identification, criteria have to be defined first. However, it would be difficult to quantify results since this module is based on the framework of TSKM, which is designed for temporal reasoning, as stated in [Moskovitch et al., 2007]:

> Evaluating knowledge discovered from a mining process is challenging since it is hard to estimate the quality of the discovered knowledge in quantitative terms, such as accuracy in classification.

In addition, one key process in TSKM is to find Tones, and finding Tones requires methods grouping similar objects together, such as clustering analysis. Although some criteria exist, e.g. Davies–Bouldin index, Fowlkes–Mallows index, etc., which evaluate the performance of algorithms, or methods, such as comparing with known artificial data [Mörchen and Ultsch, 2005], the results still have to be “validated” by experts. For instance, the proposed algorithm PERSIST in the framework of TSKM [Mörchen, 2006b; Mörchen and Ultsch, 2005] can only produce reasonable results for soil moisture data in the application example described in [Gronz et al., 2008].

Moreover, a relative diverse data set or a data set with multiple different Aspects is more appropriate for the framework of TSKM. Yet, the data sets in the application examples in Chapter 6 are not very diverse, and this also restricts the evaluation of event identification.

Apart from the above objective restraints, the current implementation of the prototype also does not implement the filtering mechanism in the process

of finding Tones as described in the framework of TSKM [Mörchen, 2006b; Mörchen et al., 2005]. This filtering mechanism is used to remove short Tones in each Aspect, and these short Tones may come from noise and do not represent a state well. This is especially important for temporal reasoning to focus on main features to gain the overall understanding of the process of interest. However, this might also cause uncertainties to identify short-term extreme events in natural environment, such as a short-term thundershower, and further affects the possibilities in composing scenarios. This is still an open question.

With the properties of TSKM itself and under the current implementation of the prototype, the module of event identification shows the capability to determine different features based on the users' choice of algorithms, grouping numbers, etc., within a flexible, modularized software implementation as shown in Chapter 6. Further discussions regarding, for instance, the possible improvements, will be discussed in Chapter 8.

7.3 Evaluation on Process Identification

As also mentioned in Section 4.4, the purpose of process identification is to describe the relationship among different physical variables. However, the results in Chapter 6, except the academic test case, are not considered competent compared to other data-driven based approaches as listed in Section 4.4. Some possible reasons are mentioned in Chapter 6, such as:

- improper composition of input and output variables
- no hysteresis considered
- insufficient number of effective rules

Although it is possible to adjust the steps in the module of event identification to gain sufficient number of rules and study further possibilities to consider hysteresis, one core limitation for the process identification still exists — the composition of input and output variables. This limitation is derived from the different concepts between event identification and

process identification. For process identification, a process to determine the correlation among variables is required that focuses on building the descriptions of the relationships. Event identification, on the other hand, also contains a process to reduce dimensions of variables, but it considers more on how to extract features of phenomena. Due to this basic difference, the base of the rule sets derived from the event and process identifications will differ. In this sense, the rules derived from event identification can be regarded as “global” rules which consider overall phenomena as one system and it is a top-down approach. The rules derived from process identification can be viewed as “local” rules which describe the relationships among physical variables themselves and it is a bottom-up approach.

Under the current development of the prototype, the rules for process identification are acquired from the event identification. Under this condition, the rules for process identification may not be suitable and include less relevant variables, as the cases in Chapter 6 including temperature variable in the rainfall-runoff analysis.

To improve the results of the process identification, it can be redesigned to be an independent module which focuses on the process identification only and the results can be later merged into the framework. With this design, the framework will be more flexible and the inputs will be independent of the outputs from event identification. In this case, this module will be more specialized in the description of the process based on the collected data directly. Furthermore, the choices of process identification methods can have more options, such as Artificial Neural Networks (ANNs), Genetic Algorithms (GAs), empirical functions, or physically-based models, etc., apart from fuzzy logic.

Although the results of the process identification are not considered feasible, the current research work is mainly focusing on the investigation and development of the scenario composition framework and the module of process identification can be replaced by other approaches without much effort within this framework.

7.4 Evaluation on Scenario Composition

Scenario composition aggregates information from previous modules and provides a Graphical User Interface (GUI) to assist users' composition of scenarios and to realize results by generating corresponding time series data sets. Examples can be seen in both Fig. 6.16b and Fig. 6.20. Also, these generated time series data can be further applied in simulation models as BCs for the investigation of the impacts under certain pre-defined scenarios as the application example illustrated in Section 6.6.

Since users have all kinds of possibilities to compose any scenario of interest and the current prototype implementation has no constraints on scenario composition, any time series data set can be generated, even an absurd one. To avoid any abuse of scenario composition, sufficient information has to be given to users while composing scenarios. At the current stage, the basic statistical information for each physical variable in every MetaEvent and the results of the suffix-tree indicating the order of MetaEvents are given. However, more information is still required for scenario composition. In the examples given in Section 6.5 and Section 6.6, a top-down approach is taken to compose a three-month and one-year scenarios individually. Besides, an overview of data composition in these certain periods is also necessary. Several suggestions can be proposed to provide the additional information in the current prototype, such as:

- a user-specified time window to display the historical data and the composition of existing MetaEvents
- a way to group MetaEvents into subgroups based on their properties, such as time, period, data range, etc., to reduce the complexity of composition by screening out unnecessary MetaEvents

Besides, the provided information does not contain *when* MetaEvents appear in the Event Editing Window (shown in Fig. 5.9), and that is the reason why regular expression comes in for help while composing scenarios as described in Section 6.5. In addition, due to the properties of Chords

derived from the TSKM which might overlap with each other in the direction of the time axis, the current internal mechanism which breaks down the overlapping parts in the scenario composition module may also cause some orderings in the history not disclosed in the data structure of the suffix tree.

Apart from previous discussions, some other recommendations and suggestions are worth trying to facilitate the scenario composition and time series generation, such as:

- other possible techniques in finding phrases in the framework of TSKM, e.g. modified Closed Association Rule Mining (CHARM), to compare with the contribution of the current suffix tree version
- feedback from process identification to optimize the generation of time series data
- downscaling techniques applied on the generated time series data to satisfy more detailed requirements if necessary, e.g. applications of Global Climate Model (GCM)

7.5 Summary

In this chapter, major modules in the framework of time series composition are evaluated, and the foundation of the evaluations is mainly based on the implemented prototype. Although the evaluations are not quantified and the results still have room for improvement, the prototype does serve the purpose of proving the concept and achieving the objectives of the time series composition framework stated in Section 1.3.

In addition, with the background knowledge used behind the framework, e.g. TSKM, the results are suggested to be evaluated by regional or domain experts as a part of a semi-automatic process in the framework instead of being judged by certain fixed indexes. Further suggestions to improve the current prototype will be then discussed in the coming Chapter 8.

Chapter 8

Conclusions

8.1 Summary

This research work was carried out to design a general framework of scenario composition within hydroinformatics systems. A software prototype was also implemented to demonstrate and validate the concept of the design. This framework can be used to compose scenarios of interest and these user-specified scenarios can be further converted into a set of time series data as inputs, e.g. Boundary Conditions (BCs), to support simulation tasks in the disciplines of hydro science and engineering.

The concept of this framework is based on the fundamental needs in answering the impacts of what-if scenarios and to take advantage of the collected time series raw data. What the framework does, in general, is to fill the gap between the available mass time series data and the simulation tools by providing input data sets generated from these what-if scenarios with the help of modern software technology, as stated in Chapter 1. The operational process of the framework is semi-automatic which means the process is usually iterative due to different types of data sets, algorithms, etc., and the results are supposed to be reviewed and confirmed by domain experts to ensure the feasibility of the generated time series data set.

The concept of the framework can be divided into four modules, as described in Chapter 4:

- data pre-processing
- event identification
- process identification
- scenario composition

These modules are aiming to break the time series data of different sources representing different hydrological or hydrodynamic processes into representative blocks, Events, to present specific characteristics of phenomena. Further, these Events will be extended with additional information, e.g. statistics, describing themselves, and the relationships among state variables inside each Event are also described. This turns each Event into a MetaEvent which forms a basic element for the scenario composition. At the end, scenarios of interest can be composed by users with the help of information offered from the system, and sets of time series data can be generated from these specified scenarios for further investigations of problems.

Since the framework contains four modules and each module performs differently, there are different possible theories, methods and techniques, etc., to achieve the desired objective. The background knowledge used for these modules, except data pre-processing, is described in Chapter 3 and contains:

- Time Series Knowledge Representation (TSKR) and Time Series Knowledge Mining (TSKM)
- fuzzy logic and Multivariate Adaptive Regression Splines (MARS)
- suffix tree

The prototype is aimed to demonstrate the capability and usability of the framework concept. Due to this reason, neither specialized nor problem-specific but general-purpose algorithms and functions were implemented. The implementation of the prototype of the framework was carried out within the scope of available software technology and under varied considerations

and criteria, as discussed in Chapter 5. In the end, the prototype implementation was carried out:

- on Linux Operating System (OS)
- in Java and R programming languages
- on single Personal Computer (PC) based computer

In addition to the implementation of the core functionalities of the framework, a simple Graphical User Interface (GUI) is also provided as a tool for normal users in the hydroinformatics system to assist the composition of scenarios of interest and the generation of time series data. This implementation can be used as a stand-alone application or be integrated into different hydroinformatics systems as illustrated in Section 5.7.

Moreover, four application examples to demonstrate the framework concept and an evaluation on the framework based on these examples, designs and implementations are discussed in Chapter 6 and Chapter 7 respectively. The time series data used for application examples contain from mathematical function-generated ones to measured hydrological and hydrodynamic ones. The results of these four application examples demonstrate the capability and future possibilities of this framework. It is achieved by reproducing similar time series patterns from specific scenarios compared to the original ones together with providing simulation tools with time series data as BCs generated from the scenarios of interest. With these results, the objective of this framework, filling the gap between available raw data and simulation tools, is considered being accomplished. Besides, compared to other approaches, such as weather generators, this framework offers a different approach, which composes scenarios semi-automatically from the collected time series raw data of different sources, e.g. measurements, simulation results, etc., to assist in answering the impacts under what-if scenarios.

8.2 Outlook

From the results and evaluations described in Chapter 6 and Chapter 7, at least two different aspects can be viewed as directions for further investigations, and these two aspects are:

Concept: Although the prototype is able to prove the framework's concept, several facets are still worth investigating to improve the framework's applicability in real tasks, and they are:

- Data: Even though different application examples, as described in Chapter 6, are used to inspect and demonstrate how the framework works based on the current prototype implementation, the contents of data used are limited from three to four variables and some variables are too monotonic. Since the framework is targeting at filling the gap between available mass data and the simulation tools in real application projects, more diverse data sets, which contain more variables and different data features, such as the research project "Großhang — Natural Slope" [Hinkelmann et al.; Molkenthin et al., 2014; Zehe and Hinkelmann, 2013] as mentioned earlier, are needed for further investigation. Moreover, the point-source-based time series outputs can be further extended to spatial ones by broadening the concept of temporal patterns to the concept of spatial patterns. In addition, the scalar quantity can be also expanded to the vector one to open up more application fields.
- Relationships: The application examples in Chapter 6 and the discussion in Section 7.3 show that the current design and implementation still have room for improvement in describing relationships among different variables due to some reasons, e.g. insufficient number of effective rules. Although describing the relationships among different variables is not the major focus in the current implementation, this functionality can be further improved by the methods or approaches suggested in Section 7.3, such as

replacing the method of the process identification module with other problem-specific options. Later, these relationships can be further investigated to optimize generated results.

- Algorithms: Since the current implementation is general-purpose, the functionalities in the prototype are not chosen or designed for any specific data type or problem. As described in Chapter 3, different possibilities exist in the choice of algorithms. In addition, different types of physical variables may also need different algorithms during the steps inside the framework. In order to improve the applicability, different algorithms have to be further implemented and tested on different types of variables based on their characteristics, e.g. temperature, soil moisture, rainfall, etc., to find suitable algorithms for specific variable types.
- Guidelines: In the current prototype implementation, it offers an interface to compose scenarios and a top-down approach of composing scenarios is also described in Section 6.5 and Section 6.6. A general guideline for the purpose of scenario composition is required for practical applications. Besides, more detailed guidelines depending on the type of, e.g. applications, are also needed. These guidelines should not only describe steps of how scenarios are composed but also suggest which algorithm is a better choice under which condition. With such guidelines, users can have better control over the creation of scenarios. Besides, the generated results are suggested to be examined by local or domain experts. It can be an improvement of the framework to offer some possible indexes to judge the quality of the results.

Information and Process Handling: In terms of information and process handling, it is targeted at providing more precise information and a more convenient environment for users to efficiently compose and generate required time series data of their needs. To achieve these, two facets can be considered:

- More precise information: As discussed in Chapter 7, more precise information, e.g. subgroups of MetaEvents depending on the seasons, is required while composing scenarios even with the information provided by the default GUI settings. That is the reason why the regular expression was used in the application example in Section 6.5. To improve the quality of the provided information, the information can be provided strategically based on the guidelines of the scenario composition. For instance, if the composition is a top-down process, at least, a way to reduce the number of available MetaEvents and a basic query functionality to extract matched MetaEvents are necessary. Apart from this, a possibility for users to add proper descriptions to MetaEvents can be added instead of the original pre-defined strings. Since MetaEvents work as LEGO® bricks while composing user-specified scenarios, it will be of great help with properer and more meaningful semantics.
- Integrated environment: The current prototype implementation takes in the information from other modules in the form of serialized or unserialized files and provides two major windows for scenario composition and time series generation as shown in Fig. 5.9 and Fig. 5.10. In this way, users have to generate these files separately, and this procedure can be improved by integrating other functionalities into the current implementation as a whole system and exchanging information through open standards. For example, a domain-specific Read-Eval-Print Loop (REPL) environment designed and implemented by scripting languages of a homogeneous computing environment, such as Groovy [Groovy], Clojure [Clojure], etc., can be adapted for pre-processing and the tasks of other modules to improve the overall workflow. Moreover, WaterML2 [WaterML2] can be applied for the information exchange among different modules. In addition, some functionalities, such as investigating available time series data by interactive windows, some basic downscaling operations, etc., can also be extended to facilitate scenario composition and further time series

generation.

In general, a concept [Li and Molkenthin, 2014; Molkenthin et al., 2014] is proposed in this research work to fill the gap of available mass time series data and simulation tools by providing a semi-automatic approach. It provides a step forward and a valuable tool to holistic hydroinformatics sytesms, e.g. Integrated Water Resources Management (IWRM), and the challenge of the human society.

Appendices

Appendix A

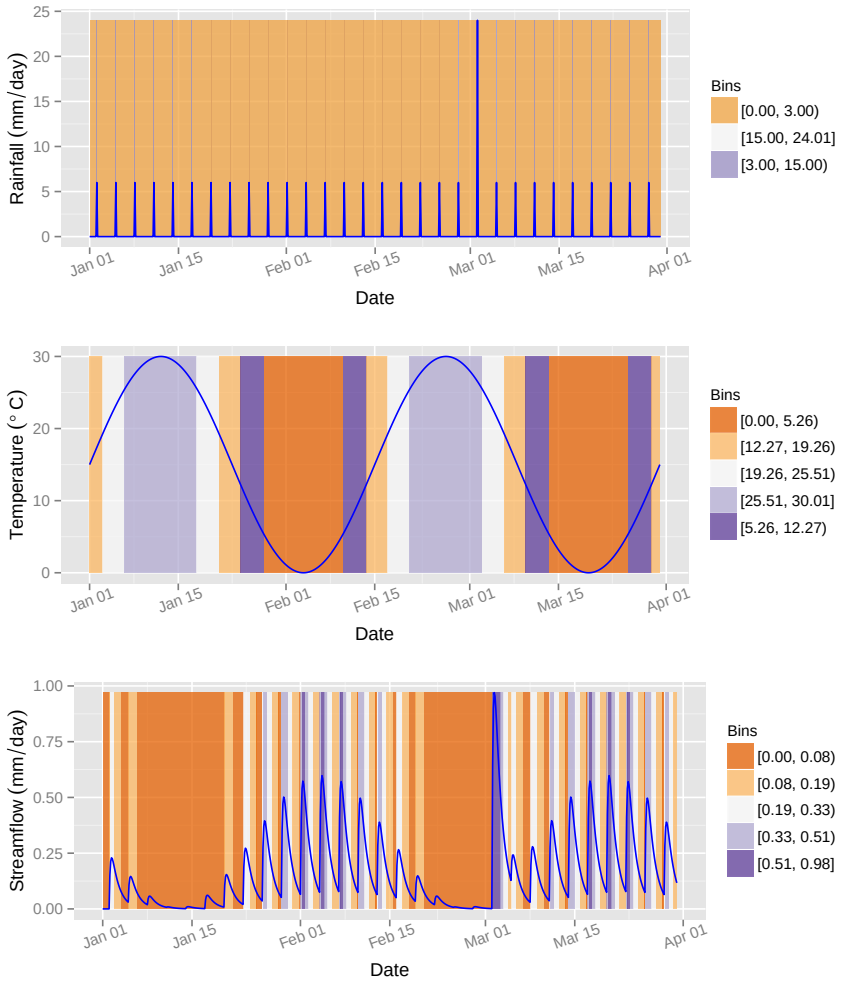


Figure A.1: Derived Tones from the HydroTestData data set with the settings of more categories

Table A.1: List of 29 generated MetaEvents based on the HydroTestData data set with the settings of more categories

		Feq.	Max. Duration	Min. Duration
		05% (37/721)	1.603 Day(s)	1.062 Day(s)
Event1	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[12.27, 19.26)	19.135	16.978 ± 1.008	15.000
	Q::[0.00, 0.08)	0.077	0.036 ± 0.026	0.000
		Feq.	Max. Duration	Min. Duration
		00% (2/721)	3.000 Hour(s)	3.000 Hour(s)
Event2	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[3.00, 15.00)	6.000	6.000 ± 0.000	6.000
	T::[12.27, 19.26)	17.347	16.566 ± 1.104	15.785
	Q::[0.08, 0.19)	0.189	0.170 ± 0.026	0.152
		Feq.	Max. Duration	Min. Duration
		05% (37/721)	20.243 Hour(s)	15.008 Hour(s)
Event3	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[12.27, 19.26)	18.882	15.214 ± 2.141	12.395
	Q::[0.19, 0.33)	0.326	0.244 ± 0.036	0.191
		Feq.	Max. Duration	Min. Duration
		04% (31/721)	1.070 Day(s)	5.802 Hour(s)
Event4	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[12.27, 19.26)	19.135	14.892 ± 2.218	12.395
	Q::[0.08, 0.19)	0.184	0.135 ± 0.030	0.085

		Feq.	Max. Duration	Min. Duration
		07% (49/721)	1.159 Day(s)	20.506 Hour(s)
Event5	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[19.26, 25.51)	25.037	21.866 ± 1.986	19.386
	Q::[0.08, 0.19)	0.177	0.126 ± 0.025	0.082
		Feq.	Max. Duration	Min. Duration
		06% (41/721)	2.125 Day(s)	1.146 Day(s)
Event6	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[19.26, 25.51)	25.420	22.988 ± 1.510	19.386
	Q::[0.00, 0.08)	0.080	0.042 ± 0.023	0.008
		Feq.	Max. Duration	Min. Duration
		00% (3/721)	2.780 Hour(s)	2.780 Hour(s)
Event7	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[3.00, 15.00)	6.000	6.000 ± 0.000	6.000
	T::[19.26, 25.51)	23.170	22.716 ± 0.785	21.810
	Q::[0.08, 0.19)	0.113	0.109 ± 0.004	0.106
		Feq.	Max. Duration	Min. Duration
		23% (168/721)	2.875 Day(s)	1.500 Day(s)
Event8	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[25.51, 30.01]	30.000	28.563 ± 1.294	25.607
	Q::[0.00, 0.08)	0.062	0.017 ± 0.018	0.001
		Feq.	Max. Duration	Min. Duration
		01% (7/721)	3.000 Hour(s)	3.000 Hour(s)
Event9	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[3.00, 15.00)	6.000	6.000 ± 0.000	6.000
	T::[25.51, 30.01]	29.815	28.632 ± 1.315	26.657
	Q::[0.00, 0.08)	0.046	0.024 ± 0.020	0.007

		Feq.	Max. Duration	Min. Duration
		00% (0/721)	34.511 min.	34.511 min.
Event10	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[3.00, 15.00)	NaN	$NaN \pm NaN$	NaN
	T::[19.26, 25.51)	NaN	$NaN \pm NaN$	NaN
	Q::[0.00, 0.08)	NaN	$NaN \pm NaN$	NaN
		Feq.	Max. Duration	Min. Duration
		00% (2/721)	3.000 Hour(s)	3.000 Hour(s)
Event11	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[3.00, 15.00)	6.000	6.000 ± 0.000	6.000
	T::[12.27, 19.26)	17.347	16.566 ± 1.104	15.785
	Q::[0.19, 0.33)	0.197	0.197 ± 0.000	0.197
		Feq.	Max. Duration	Min. Duration
		05% (36/721)	1.071 Day(s)	7.125 Hour(s)
Event12	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[5.26, 12.27)	12.138	8.250 ± 2.020	5.560
	Q::[0.08, 0.19)	0.185	0.126 ± 0.033	0.083
		Feq.	Max. Duration	Min. Duration
		04% (26/721)	21.872 Hour(s)	21.872 Hour(s)
Event13	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[5.26, 12.27)	11.626	9.885 ± 2.025	5.358
	Q::[0.00, 0.08)	0.079	0.061 ± 0.015	0.036
		Feq.	Max. Duration	Min. Duration
		03% (20/721)	16.896 Hour(s)	1.003 Hour(s)
Event14	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[5.26, 12.27)	8.190	7.730 ± 0.332	7.274
	Q::[0.19, 0.33)	0.311	0.254 ± 0.038	0.203

		Feq.	Max. Duration	Min. Duration
		01% (4/721)	3.000 Hour(s)	3.000 Hour(s)
Event15	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[3.00, 15.00)	6.000	6.000 ± 0.000	6.000
	T::[5.26, 12.27)	11.118	10.371 ± 0.862	9.624
	Q::[0.19, 0.33)	0.285	0.281 ± 0.004	0.277
		Feq.	Max. Duration	Min. Duration
		04% (32/721)	20.953 Hour(s)	13.935 Hour(s)
Event16	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[5.26, 12.27)	12.138	8.521 ± 2.217	5.765
	Q::[0.33, 0.51)	0.497	0.399 ± 0.050	0.334
		Feq.	Max. Duration	Min. Duration
		02% (16/721)	10.738 Hour(s)	10.738 Hour(s)
Event17	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[0.00, 5.26)	5.159	2.911 ± 2.070	0.009
	Q::[0.00, 0.08)	0.078	0.067 ± 0.009	0.052
		Feq.	Max. Duration	Min. Duration
		06% (40/721)	21.400 Hour(s)	1.262 Hour(s)
Event18	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[0.00, 5.26)	4.210	1.943 ± 1.547	0.146
	Q::[0.33, 0.51)	0.504	0.427 ± 0.053	0.339
		Feq.	Max. Duration	Min. Duration
		01% (8/721)	3.000 Hour(s)	3.000 Hour(s)
Event19	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[3.00, 15.00)	6.000	6.000 ± 0.000	6.000
	T::[0.00, 5.26)	4.393	1.761 ± 1.737	0.021
	Q::[0.33, 0.51)	0.427	0.399 ± 0.029	0.354

		Feq.	Max. Duration	Min. Duration
		06% (44/721)	16.330 Hour(s)	16.330 Hour(s)
Event20	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[0.00, 5.26)	3.679	1.683 ± 1.380	0.112
	Q::[0.19, 0.33)	0.325	0.254 ± 0.043	0.191
		Feq.	Max. Duration	Min. Duration
		09% (66/721)	1.071 Day(s)	1.071 Day(s)
Event21	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[0.00, 5.26)	5.159	1.834 ± 1.668	0.000
	Q::[0.08, 0.19)	0.188	0.126 ± 0.032	0.081
		Feq.	Max. Duration	Min. Duration
		03% (24/721)	12.497 Hour(s)	10.524 Hour(s)
Event22	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[0.00, 5.26)	2.142	0.932 ± 0.797	0.037
	Q::[0.51, 0.98]	0.598	0.556 ± 0.022	0.522
		Feq.	Max. Duration	Min. Duration
		00% (2/721)	3.000 Hour(s)	3.000 Hour(s)
Event23	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[3.00, 15.00)	6.000	6.000 ± 0.000	6.000
	T::[5.26, 12.27)	5.560	5.560 ± 0.000	5.560
	Q::[0.33, 0.51)	0.360	0.360 ± 0.000	0.360
		Feq.	Max. Duration	Min. Duration
		00% (1/721)	3.750 Hour(s)	3.750 Hour(s)
Event24	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[15.00, 24.01]	24.000	24.000 ± 0.000	24.000
	T::[25.51, 30.01]	26.657	26.657 ± 0.000	26.657
	Q::[0.51, 0.98]	0.643	0.643 ± 0.000	0.643

		Feq.	Max. Duration	Min. Duration
		01% (6/721)	18.375 Hour(s)	18.375 Hour(s)
Event25	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[25.51, 30.01]	26.491	26.054 ± 0.331	25.607
	Q::[0.51, 0.98]	0.970	0.884 ± 0.080	0.756
		Feq.	Max. Duration	Min. Duration
		00% (3/721)	9.983 Hour(s)	9.983 Hour(s)
Event26	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[19.26, 25.51)	25.420	25.229 ± 0.191	25.037
	Q::[0.51, 0.98]	0.685	0.621 ± 0.063	0.558
		Feq.	Max. Duration	Min. Duration
		01% (5/721)	12.706 Hour(s)	12.706 Hour(s)
Event27	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[19.26, 25.51)	24.841	24.437 ± 0.322	24.027
	Q::[0.33, 0.51)	0.503	0.412 ± 0.068	0.331
		Feq.	Max. Duration	Min. Duration
		01% (10/721)	16.220 Hour(s)	16.220 Hour(s)
Event28	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.00)	0.000	0.000 ± 0.000	0.000
	T::[19.26, 25.51)	23.817	22.242 ± 1.252	20.619
	Q::[0.19, 0.33)	0.298	0.234 ± 0.032	0.195
		Feq.	Max. Duration	Min. Duration
		00% (1/721)	2.465 Hour(s)	2.465 Hour(s)
Event29	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[3.00, 15.00)	6.000	6.000 ± 0.000	6.000
	T::[19.26, 25.51)	21.810	21.810 ± 0.000	21.810
	Q::[0.19, 0.33)	0.214	0.214 ± 0.000	0.214

Table A.2: List of 42 generated MetaEvents based on the BinghamTrib data set

		Feq.	Max. Duration	Min. Duration
		17% (2157/12588)	5.985 Week(s)	5.135 Day(s)
Event1	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.85)	3.800	0.443 ± 0.854	0.000
	T::[18.35, 21.35)	21.300	19.626 ± 0.871	18.400
	Q::[0.00, 0.34)	0.320	0.013 ± 0.036	0.000
		Feq.	Max. Duration	Min. Duration
		00% (33/12588)	2.325 Day(s)	1.076 Day(s)
Event2	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[26.30, 103.10)	69.500	35.721 ± 9.281	26.700
	T::[18.35, 21.35)	21.200	19.785 ± 0.938	18.400
	Q::[0.00, 0.34)	0.305	0.011 ± 0.053	0.000
		Feq.	Max. Duration	Min. Duration
		01% (115/12588)	3.018 Day(s)	1.745 Day(s)
Event3	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[12.55, 26.30)	25.800	17.364 ± 3.510	12.600
	T::[18.35, 21.35)	21.300	19.489 ± 0.872	18.400
	Q::[0.00, 0.34)	0.310	0.035 ± 0.075	0.000
		Feq.	Max. Duration	Min. Duration
		02% (305/12588)	4.266 Day(s)	17.795 Hour(s)
Event4	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[3.85, 12.55)	12.500	7.171 ± 2.475	3.900
	T::[18.35, 21.35)	21.300	19.491 ± 0.837	18.400
	Q::[0.00, 0.34)	0.330	0.033 ± 0.066	0.000

		Feq.	Max. Duration	Min. Duration
		00% (55/12588)	2.153 Day(s)	15.333 Hour(s)
Event5	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[26.30, 103.10)	49.700	32.387 ± 5.240	26.400
	T::[15.50, 18.35)	18.300	17.358 ± 0.627	15.700
	Q::[0.00, 0.34)	0.199	0.010 ± 0.039	0.000
		Feq.	Max. Duration	Min. Duration
		17% (2098/12588)	3.629 Week(s)	23.584 Hour(s)
Event6	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.85)	3.800	0.598 ± 0.948	0.000
	T::[15.50, 18.35)	18.300	17.168 ± 0.668	15.500
	Q::[0.00, 0.34)	0.321	0.014 ± 0.045	0.000
		Feq.	Max. Duration	Min. Duration
		04% (459/12588)	4.483 Day(s)	1.565 Day(s)
Event7	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[3.85, 12.55)	12.500	7.535 ± 2.427	3.900
	T::[15.50, 18.35)	18.300	17.085 ± 0.699	15.500
	Q::[0.00, 0.34)	0.330	0.016 ± 0.052	0.000
		Feq.	Max. Duration	Min. Duration
		02% (234/12588)	4.306 Day(s)	1.076 Day(s)
Event8	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[12.55, 26.30)	26.200	17.903 ± 3.765	12.600
	T::[15.50, 18.35)	18.300	17.107 ± 0.708	15.500
	Q::[0.00, 0.34)	0.336	0.018 ± 0.062	0.000
		Feq.	Max. Duration	Min. Duration
		00% (7/12588)	1.214 Day(s)	16.134 Hour(s)
Event9	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[26.30, 103.10)	45.900	34.771 ± 8.518	27.000
	T::[15.50, 18.35)	18.100	16.600 ± 0.985	15.700
	Q::[0.34, 2.01)	1.764	0.780 ± 0.443	0.502

		Feq.	Max. Duration	Min. Duration
		00% (30/12588)	1.889 Day(s)	1.359 Day(s)
Event10	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[3.85, 12.55)	12.400	7.500 ± 3.055	3.900
	T::[15.50, 18.35)	18.200	16.903 ± 0.893	15.700
	Q::[0.34, 2.01)	1.757	0.662 ± 0.356	0.341
		Feq.	Max. Duration	Min. Duration
		00% (59/12588)	4.142 Day(s)	2.172 Day(s)
Event11	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.85)	3.700	0.937 ± 1.112	0.000
	T::[15.50, 18.35)	18.200	17.044 ± 0.757	15.700
	Q::[0.34, 2.01)	1.693	0.638 ± 0.341	0.341
		Feq.	Max. Duration	Min. Duration
		00% (24/12588)	2.566 Day(s)	20.702 Hour(s)
Event12	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[12.55, 26.30)	23.800	17.575 ± 3.260	13.300
	T::[15.50, 18.35)	18.200	16.900 ± 0.740	15.700
	Q::[0.34, 2.01)	1.473	0.735 ± 0.272	0.352
		Feq.	Max. Duration	Min. Duration
		00% (3/12588)	1.178 Day(s)	11.440 Hour(s)
Event13	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[26.30, 103.10)	44.000	35.067 ± 8.491	27.100
	T::[15.50, 18.35)	15.900	15.900 ± 0.000	15.900
	Q::[2.01, 4.82]	4.814	3.229 ± 1.374	2.369
		Feq.	Max. Duration	Min. Duration
		00% (5/12588)	22.685 Hour(s)	22.685 Hour(s)
Event14	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[3.85, 12.55)	10.100	7.460 ± 2.185	4.300
	T::[15.50, 18.35)	17.600	16.920 ± 0.931	15.900
	Q::[2.01, 4.82]	3.651	2.813 ± 0.688	2.222

		Feq.	Max. Duration	Min. Duration
		00% (1/12588)	1.416 Day(s)	4.314 Hour(s)
Event15	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[12.55, 26.30)	21.700	21.700 ± 0.000	21.700
	T::[15.50, 18.35)	17.600	17.600 ± 0.000	17.600
	Q::[2.01, 4.82]	4.334	4.334 ± 0.000	4.334
		Feq.	Max. Duration	Min. Duration
		00% (0/12588)	17.130 Hour(s)	17.130 Hour(s)
Event16	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.85)	NaN	NaN ± NaN	NaN
	T::[15.50, 18.35)	NaN	NaN ± NaN	NaN
	Q::[2.01, 4.82]	NaN	NaN ± NaN	NaN
		Feq.	Max. Duration	Min. Duration
		00% (13/12588)	1.214 Day(s)	7.094 Hour(s)
Event17	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[12.55, 26.30)	24.700	16.977 ± 3.603	12.800
	T::[18.35, 21.35)	20.800	19.431 ± 0.693	18.600
	Q::[0.34, 2.01)	1.792	0.676 ± 0.439	0.356
		Feq.	Max. Duration	Min. Duration
		16% (1972/12588)	5.881 Week(s)	2.245 Week(s)
Event18	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.85)	3.800	0.268 ± 0.656	0.000
	T::[21.35, 25.15)	25.100	23.216 ± 1.158	21.400
	Q::[0.00, 0.34)	0.258	0.003 ± 0.016	0.000
		Feq.	Max. Duration	Min. Duration
		00% (46/12588)	2.292 Day(s)	2.292 Day(s)
Event19	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[12.55, 26.30)	25.600	17.293 ± 3.705	12.600
	T::[21.35, 25.15)	25.100	22.980 ± 1.080	21.400
	Q::[0.00, 0.34)	0.298	0.022 ± 0.068	0.000

		Feq.	Max. Duration	Min. Duration
		16% (2022/12588)	8.633 Week(s)	4.398 Week(s)
Event20	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.85)	3.800	0.156 ± 0.532	0.000
	T::[25.15, 28.95)	28.900	27.218 ± 1.058	25.200
	Q::[0.00, 0.34)	0.331	0.000 ± 0.010	0.000
		Feq.	Max. Duration	Min. Duration
		20% (2579/12588)	12.962 Week(s)	8.509 Week(s)
Event21	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.85)	3.800	0.094 ± 0.406	0.000
	T::[28.95, 34.31]	34.300	30.732 ± 1.183	29.000
	Q::[0.00, 0.34)	0.039	0.000 ± 0.001	0.000
		Feq.	Max. Duration	Min. Duration
		00% (23/12588)	1.485 Day(s)	1.485 Day(s)
Event22	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[12.55, 26.30)	23.800	17.452 ± 3.272	12.600
	T::[25.15, 28.95)	28.900	27.028 ± 1.196	25.200
	Q::[0.00, 0.34)	0.222	0.010 ± 0.046	0.000
		Feq.	Max. Duration	Min. Duration
		01% (76/12588)	2.485 Day(s)	22.796 Hour(s)
Event23	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[3.85, 12.55)	12.500	7.334 ± 2.527	3.900
	T::[25.15, 28.95)	28.600	26.918 ± 1.016	25.200
	Q::[0.00, 0.34)	0.000	0.000 ± 0.000	0.000
		Feq.	Max. Duration	Min. Duration
		01% (157/12588)	2.928 Day(s)	1.083 Day(s)
Event24	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[3.85, 12.55)	12.500	7.260 ± 2.439	3.900
	T::[21.35, 25.15)	25.100	22.968 ± 1.104	21.400
	Q::[0.00, 0.34)	0.303	0.015 ± 0.045	0.000

		Feq.	Max. Duration	Min. Duration
		00% (5/12588)	1.646 Day(s)	1.646 Day(s)
Event25	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[26.30, 103.10)	58.900	38.340 ± 12.643	28.000
	T::[28.95, 34.31]	31.000	30.140 ± 0.594	29.500
	Q::[0.00, 0.34)	0.000	0.000 ± 0.000	0.000
		Feq.	Max. Duration	Min. Duration
		00% (19/12588)	2.381 Day(s)	7.925 Hour(s)
Event26	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[12.55, 26.30)	26.200	19.211 ± 3.779	13.100
	T::[28.95, 34.31]	32.500	30.395 ± 0.878	29.100
	Q::[0.00, 0.34)	0.000	0.000 ± 0.000	0.000
		Feq.	Max. Duration	Min. Duration
		00% (43/12588)	3.016 Day(s)	1.791 Day(s)
Event27	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[3.85, 12.55)	12.200	6.977 ± 2.482	4.100
	T::[28.95, 34.31]	32.700	30.308 ± 1.052	29.000
	Q::[0.00, 0.34)	0.000	0.000 ± 0.000	0.000
		Feq.	Max. Duration	Min. Duration
		00% (8/12588)	1.533 Day(s)	1.533 Day(s)
Event28	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[26.30, 103.10)	78.000	42.188 ± 16.730	27.400
	T::[25.15, 28.95)	28.600	26.963 ± 1.345	25.300
	Q::[0.00, 0.34)	0.000	0.000 ± 0.000	0.000
		Feq.	Max. Duration	Min. Duration
		00% (1/12588)	9.995 Hour(s)	9.995 Hour(s)
Event29	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[26.30, 103.10)	30.100	30.100 ± 0.000	30.100
	T::[21.35, 25.15)	21.500	21.500 ± 0.000	21.500
	Q::[0.34, 2.01)	0.424	0.424 ± 0.000	0.424

		Feq.	Max. Duration	Min. Duration
		00% (1/12588)	1.185 Day(s)	1.185 Day(s)
Event30	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[3.85, 12.55)	12.400	12.400 ± 0.000	12.400
	T::[21.35, 25.15)	22.200	22.200 ± 0.000	22.200
	Q::[0.34, 2.01)	0.612	0.612 ± 0.000	0.612
		Feq.	Max. Duration	Min. Duration
		00% (13/12588)	1.294 Day(s)	1.046 Day(s)
Event31	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[26.30, 103.10)	59.200	37.223 ± 8.964	29.000
	T::[21.35, 25.15)	24.436	23.198 ± 1.035	21.700
	Q::[0.00, 0.34)	0.276	0.022 ± 0.076	0.000
		Feq.	Max. Duration	Min. Duration
		00% (0/12588)	6.382 Hour(s)	6.382 Hour(s)
Event32	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[26.30, 103.10)	NaN	NaN ± NaN	NaN
	T::[25.15, 28.95)	NaN	NaN ± NaN	NaN
	Q::[2.01, 4.82]	NaN	NaN ± NaN	NaN
		Feq.	Max. Duration	Min. Duration
		00% (1/12588)	22.175 Hour(s)	22.175 Hour(s)
Event33	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[103.10, 142.61]	142.600	142.600 ± 0.000	142.600
	T::[25.15, 28.95)	28.000	28.000 ± 0.000	28.000
	Q::[2.01, 4.82]	2.758	2.758 ± 0.000	2.758
		Feq.	Max. Duration	Min. Duration
		00% (1/12588)	22.044 Hour(s)	22.044 Hour(s)
Event34	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[12.55, 26.30)	16.200	16.200 ± 0.000	16.200
	T::[25.15, 28.95)	28.000	28.000 ± 0.000	28.000
	Q::[2.01, 4.82]	4.797	4.797 ± 0.000	4.797

		Feq.	Max. Duration	Min. Duration
		00% (0/12588)	13.764 Hour(s)	13.764 Hour(s)
Event35	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.85)	NaN	NaN ± NaN	NaN
	T::[25.15, 28.95)	NaN	NaN ± NaN	NaN
	Q::[2.01, 4.82]	NaN	NaN ± NaN	NaN
		Feq.	Max. Duration	Min. Duration
		00% (4/12588)	4.130 Day(s)	4.130 Day(s)
Event36	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.85)	0.400	0.100 ± 0.200	0.000
	T::[25.15, 28.95)	28.000	28.000 ± 0.000	28.000
	Q::[0.34, 2.01)	1.329	0.806 ± 0.388	0.442
		Feq.	Max. Duration	Min. Duration
		00% (9/12588)	4.472 Day(s)	3.429 Hour(s)
Event37	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.85)	2.500	1.033 ± 1.119	0.000
	T::[18.35, 21.35)	19.600	19.600 ± 0.000	19.600
	Q::[0.34, 2.01)	1.043	0.644 ± 0.240	0.429
		Feq.	Max. Duration	Min. Duration
		00% (1/12588)	19.467 Hour(s)	19.467 Hour(s)
Event38	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[26.30, 103.10)	27.200	27.200 ± 0.000	27.200
	T::[18.35, 21.35)	19.800	19.800 ± 0.000	19.800
	Q::[0.34, 2.01)	0.355	0.355 ± 0.000	0.355
		Feq.	Max. Duration	Min. Duration
		00% (1/12588)	13.670 Hour(s)	13.670 Hour(s)
Event39	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[26.30, 103.10)	43.100	43.100 ± 0.000	43.100
	T::[18.35, 21.35)	19.600	19.600 ± 0.000	19.600
	Q::[2.01, 4.82]	2.288	2.288 ± 0.000	2.288

		Feq.	Max. Duration	Min. Duration
		00% (7/12588)	1.039 Day(s)	1.039 Day(s)
Event40	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[3.85, 12.55)	10.300	7.129 ± 2.198	4.900
	T::[18.35, 21.35)	20.800	19.729 ± 0.655	19.000
	Q::[0.34, 2.01)	1.610	0.603 ± 0.450	0.343
		Feq.	Max. Duration	Min. Duration
		00% (1/12588)	1.187 Day(s)	1.187 Day(s)
Event41	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[103.10, 142.61]	128.200	128.200 ± 0.000	128.200
	T::[28.95, 34.31]	29.300	29.300 ± 0.000	29.300
	Q::[0.00, 0.34)	0.004	0.004 ± 0.000	0.004
		Feq.	Max. Duration	Min. Duration
		00% (0/12588)	3.988 Hour(s)	3.988 Hour(s)
Event42	Rules	Max. Value	$\mu \pm \sigma$	Min. Value
	P::[0.00, 3.85)	NaN	NaN ± NaN	NaN
	T::[21.35, 25.15)	NaN	NaN ± NaN	NaN
	Q::[0.34, 2.01)	NaN	NaN ± NaN	NaN

Bibliography

M. B. Abbott. *Hydroinformatics: Information Technology and the Aquatic Environment*. Avebury, Sept. 1991. ISBN 1856288323.

M. B. Abbott. Hydroinformatics: A Copernican revolution in hydraulics. *Journal of Hydraulic Research*, 32(sup1):3–13, 1994. ISSN 0022-1686. doi: 10.1080/00221689409498800.

A. J. Abebe, D. P. Solomatine, and R. G. W. Venneker. Application of adaptive fuzzy rule-based models for reconstruction of missing precipitation events. *Hydrological Sciences Journal*, 45(3):425–436, 2000. doi: 10.1080/02626660009492339.

J. F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, Nov. 1983. ISSN 0001-0782. doi: 10.1145/182.358434.

F. Andrews and J. Guillaume. *hydromad: Hydrological Model Assessment and Development*, 2012. URL `http://hydromad.catchment.org/`. R package version 0.9-16.

C. M. Antunes and A. L. Oliveira. Temporal data mining: an overview. In *KDD Workshop on Temporal Data Mining*, pp. 1–13, 2001.

Apache Software Foundation: Commons FileUpload. Apache Commons FileUpload. URL `https://commons.apache.org/proper/commons-fileupload/`. [Accessed: 30-December-2013].

Apache Software Foundation: Commons Math. Math - Commons Math: The Apache Commons Mathematics Library. URL https://commons.apache.org/proper/commons-math/. [Accessed: 30-December-2013].

Apache Software Foundation: POI. Apache POI - the Java API for Microsoft Documents. URL https://poi.apache.org/. [Accessed: 30-December-2013].

Apache Software Foundation: Tomcat. Apache tomcat. URL https://tomcat.apache.org/. [Accessed: 30-December-2013].

ASCE Task Committee on Application of Artificial Neural Networks in Hydrology. Artificial Neural Networks in Hydrology. II: Hydrologic Applications The ASCE Task Committee on Application of Artificial Neural Networks in Hydrology. *Journal of Hydrologic Engineering, ASCE*, 5:124–137, 2000. ISSN 1084-0699.

B. Auguie. *gridExtra: functions in Grid graphics*, 2012. URL http://CRAN.R-project.org/package=gridExtra. R package version 0.9.1.

V. Babovic. Data mining in hydrology. *Hydrological Processes*, 19(7):1511–1515, 2005. doi: 10.1002/hyp.5862.

BCE. Kalypso von BCE. URL http://kalypso.bjoernsen.de/. [Accessed: 22-January-2013].

BfG. BfG - FLYS – Flusshydrologische Software. URL http://www.bafg.de/DE/08_Ref/M2/03_Fliessgewmod/01_FLYS/flys_node.html. [Accessed: 12-May-2014].

G. Boole. *An Investigation of the Laws of Thought: On which are Founded the Mathematical Theories of Logic and Probabilities*. Walton and Maberly, 1854.

B. M. Brüggemann and K. P. Holz. Integration of hydroinformatics tools in dynamic interactive documents. In *4 th International Conference on Hydroinformatics*, 2000.

B. M. Brüggemann, G. Hildebrandt, and K.-P. Holz. Distributed Engineering Environments based on Semantic information Modeling. In *Forum der Forschung*, Volume 13, Germany, 2001. BTU Cottbus.

J. Byous. JAVA TECHNOLOGY: THE EARLY YEARS, May 1998. URL http://web.archive.org/web/20050420081440/http://java.sun.com/features/1998/05/birthday.html. [Accessed: 29-December-2013].

A. Bárdossy and M. Disse. Fuzzy rule-based models for infiltration. *Water Resources Research*, 29(2):373–382, 1993. ISSN 1944-7973. doi: 10.1029/92WR02330.

A. Bárdossy and L. Duckstein. *Fuzzy rule-based modeling with applications to geophysical, biological, and engineering systems*. CRC Press, Apr. 1995. ISBN 9780849378331.

A. Bárdossy, L. Duckstein, and I. Bogardi. Fuzzy rule-based classification of atmospheric circulation patterns. *International Journal of Climatology*, 15 (10):1087–1097, 1995. ISSN 1097-0088. doi: 10.1002/joc.3370151003.

CH2M HILL. Flood Modeller Suite - ISIS. URL https://www.floodmodeller.com/global/isis/. [Accessed: 19-February-2015].

S. L. Chiu. Fuzzy model identification based on cluster estimation. *Journal of intelligent and Fuzzy systems*, 2(3):267–278, 1994.

P. Cingolani and J. Alcala-Fdez. jFuzzyLogic: a robust and flexible Fuzzy-Logic inference system language implementation. In *2012 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 1–8, June 2012. doi: 10.1109/FUZZ-IEEE.2012.6251215.

Clojure. Clojure. URL http://clojure.org/. [Accessed: 27-June-2014].

P. R. Cohen. Fluent learning: Elucidating the structure of episodes. In *Proceedings of Fourth Symposium on Intelligent Data Analysis*. Springer, 2001.

F. S. Collins, M. Morgan, and A. Patrinos. The Human Genome Project: Lessons from Large-Scale Biology. *Science*, 300(5617):286–290, Apr. 2003. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.1084564. URL http://www.sciencemag.org/content/300/5617/286.

Computer & Communications Industry Association. IBM Tightens Stranglehold Over Mainframe Market; Gets Hit with Antitrust Complaint in Europe, July 2008. URL http://www.ccianet.org/index.asp?sid=5&artid=62&evtflg=False. [Accessed: 27-March-2013].

P. Coulibaly and C. K. Baldwin. Nonstationary hydrological time series forecasting using nonlinear dynamic methods. *Journal of Hydrology*, 307 (1–4):164–174, June 2005. ISSN 0022-1694. doi: 10.1016/j.jhydrol.2004.10.008.

CUAHSI. CUAHSI Hydrologic Information System (CUAHSI-HIS). URL http://his.cuahsi.org/. [Accessed: 07-May-2014].

G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and P. Smyth. Rule Discovery From Time Series. pp. 16–22. AAAI Press, 1998.

datPAV. datPAV — Data Processing, Analysis and Visualization. URL http://smbl.nus.edu.sg/DATPAV/. [Accessed: 15-January-2013].

Deltares. Deltares systems by Deltares. URL http://www.deltaressystems.com/. [Accessed: 15-January-2013].

DHI. MIKE by DHI. URL http://mikebydhi.com/. [Accessed: 15-January-2013].

M. M. Dimitri P. Solomatine. Instance-based learning compared to other data-driven methods in hydrological forecasting. *Hydrological Processes*, 22(2):275–287, 2008.

Earthdata Collaboration Environment (ECE). ECHO and ISO 19115. URL https://wiki.earthdata.nasa.gov/display/echo/ECHO+and+ISO+19115. [Accessed: 23-October-2013].

Eclipse Foundation. Eclipse - The Eclipse Foundation open source community website. URL http://www.eclipse.org/. [Accessed: 01-January-2014].

A. Einstein, G. B. Jeffery, W. Perrett, and PIMS - University of Toronto. Geometry and experience. In *Sidelights on relativity*, pp. 25–56. London, Methuen & co. ltd, 1922.

ELWIS. Elektronischer Wasserstraßen-Informationsservice. URL https://www.elwis.de/Aktuelles/index.html. [Accessed: 25-October-2013].

Esri. Esri - GIS Mapping Software, Solutions, Services, Map Apps, and Data. URL http://www.esri.com/. [Accessed: 22-January-2013].

ESS. ESS - Emacs Speaks Statistics. URL http://ess.r-project.org/. [Accessed: 01-January-2014].

European Communities. *Proceedings of the European Expert Meeting on the Oder Flood 1997 : 18 May 1998, Potsdam, Germany : Ribamod concerted action*. Luxembourg : Office for Official Publications of the European Communities, 1999. ISBN 9789282860731.

U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From Data Mining to Knowledge Discovery in Databases. *AI magazine*, 17(3):37, 1996. URL http://www.aaai.org/ojs/index.php/aimagazine/article/viewArticle/1230.

FluidEarth. FluidEarth Portal. URL http://fluidearth.net. [Accessed: 28-May-2013].

C. Fraley and A. Raftery. MCLUST Version 3 for R: Normal Mixture Modeling and Model-Based Clustering. Technical Report 504, Department of Statistics, University of Washington, 2007. URL http://www.stat.washington.edu/mclust/.

J. H. Friedman. Multivariate Adaptive Regression Splines. *The Annals of Statistics*, 19(1):1–67, Mar. 1991. ISSN 0090-5364. doi: 10.1214/

aos/1176347963. URL http://projecteuclid.org/euclid.aos/1176347963. Mathematical Reviews number (MathSciNet): MR1091842; Zentralblatt MATH identifier: 0765.62064.

Gartner Inc. Gartner Says Worldwide PC, Tablet and Mobile Phone Shipments to Grow 4.5 Percent in 2013 as Lower-Priced Devices Drive Growth. URL http://www.gartner.com/newsroom/id/2610015. [Accessed: 07-December-2013].

D. K. Gautam. *Neural Networks and Fuzzy Logic based System Identification in Hydroinformatics*. Brandenburgische Technische Universität Cottbus Lehrstuhl Bauinformatik, Aug. 2000. ISBN 3934934056.

P. H. Gleick, editor. *Water in Crisis: A Guide to the World's Fresh Water Resources*. Oxford University Press, USA, 1st edition, Aug. 1993. ISBN 0195076281.

M. F. Goodchild. Citizens as sensors: the world of volunteered geography. *GeoJournal*, 69(4):211–221, 2007. URL http://link.springer.com/article/10.1007/s10708-007-9111-y.

Google Guava. Guava: Google Core Libraries for Java 1.6+. URL http://code.google.com/p/guava-libraries/. [Accessed: 30-December-2013].

M. B. Gorzałczany. *Computational Intelligence Systems and Applications: Neuro-Fuzzy and Fuzzy Neural Synergisms*. Springer, Feb. 2002. ISBN 9783790814392.

GRASS Development Team. *Geographic Resources Analysis Support System (GRASS GIS) Software*. Open Source Geospatial Foundation, USA, 2012. URL http://grass.osgeo.org.

J. B. Gregersen, P. J. A. Gijsbers, and S. J. P. Westen. OpenMI: Open modelling interface. *Journal of Hydroinformatics*, 9(3):175, July 2007. ISSN 14647141. doi: 10.2166/hydro.2007.023. URL http://www.iwaponline.com/jh/009/jh0090175.htm.

O. Gronz and M. Casper. Automatic analysis of hydrologic time series using time series knowledge mining to identify system states. 2008.

O. Gronz, M. Casper, and P. Gemmar. Identifying State Variables in Multivariate Hydrological Time Series Using Time Series Knowledge Mining. *Proceedings of the iEMSs Fourth Biennial Meeting: International Congress on Environmental Modelling and Software (iEMSs 2008)*, 3:1813–1820, July 2008.

Groovy. Groovy: A dynamic language for the java platform. URL `http://groovy.codehaus.org/`. [Accessed: 27-June-2014].

F. Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21, 1969.

U. Grünewald. *The Causes, Progression and Consequences of the River Oder Floods in Summer 1997 Including Remarks on the Existence of Risk Potential: An Interdisciplinary Study - Abridged Version*. Executive Board of the German IDNDR Committee, Bonn, 10th edition, 1998. ISBN 9783933181060.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11 (1):10–18, 2009. doi: 10.1145/1656274.1656278.

M. J. Hall and A. W. Minns. The classification of hydrologically homogeneous regions. *Hydrological Sciences Journal*, 44(5):693–704, 1999. doi: 10.1080/02626669909492268.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer New York, 2009. ISBN 978-0-387-84857-0.

I. Havsiyevych. Keep Learning: Suffix Trees: Java Ukkonen's Algorithm. URL `http://illya-keeplearning.blogspot.de/2009/04/suffix-trees-java-ukkonens-algorithm.html`. [Accessed: 19-March-2013].

HEC-RAS. HEC-RAS by US army corps of engineers. URL http://www.hec.usace.army.mil/software/hec-ras/. [Accessed: 22-January-2013].

M. Herrera, L. Torgo, J. Izquierdo, and R. Pérez-García. Predictive models for forecasting hourly urban water demand. *Journal of Hydrology*, 387(1–2): 141–150, June 2010. ISSN 0022-1694. doi: 10.1016/j.jhydrol.2010.04.005.

R. Hinkelmann, E. Zehe, W. Ehlers, J. Braun, M. Joswig, R. Helmig, A. Bárdossy, F. Molkenthin, J. Tronicke, and P. Dietrich. DFG Research Unit FG 581 "Natural Slope (Großhang)". URL http://www.grosshang.de. [Accessed: 26-February-2013].

K.-P. Holz, G. Hildebrandt, and L. Weber. Concept for a Web-based Information System for Flood Management. *Natural Hazards*, 38(1-2):121–140, May 2006. ISSN 0921-030X, 1573-0840. doi: 10.1007/s11069-005-8605-5.

N. E. Huang and Z. Wu. A review on Hilbert-Huang transform: Method and its applications to geophysical studies. *Reviews of Geophysics*, 46(2): 1–23, 2008. URL http://www.msri.org/people/members/2008cc/Projects/Project_5B_Ice_Core_EMD/HuangWu_EMD_RevGeo_2008.pdf.

A. Hyvärinen. Survey on independent component analysis. *Neural Computing Surveys*, 2(4):94–128, 1999. URL http://www.stat.rutgers.edu/~rebecka/Stat687/NCS99.pdf.

F. Höppner. Discovery of Temporal Patterns - Learning Rules about the Qualitative Behaviour of Time Series. In *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 192–203, Freiburg, Germany, 2001. Springer.

IEC 1131. Programmable controllers, part 7: Fuzzy control programming, fuzzy control language. Technical report, Jan. 1997. URL http://www.fuzzytech.com/binaries/ieccd1.pdf. [Accessed: 11-January-2013].

R. Ihaka and R. Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996.

INSPIRE. INSPIRE — Infrastructure for Spatial Information in the European Community. URL http://inspire.jrc.ec.europa.eu/. [Accessed: 15-January-2013].

IPCC. IPCC Data Distribution Center. URL http://www.ipcc-data.org/. [Accessed: 08-May-2014].

ISO 19115. *Geographic Information - Metadata*. Number ISO 19115. ISO, Geneva, Switzerland, 2003.

IWA Publishing. IWA Publishing — Journal of Hydroinformatics. URL http://www.iwaponline.com/jh/default.htm. [Accessed: 21-January-2013].

J.-S. Jang. ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man and Cybernetics*, 23(3):665–685, 1993.

Java. java.com: Java + You. URL http://www.java.com/. [Accessed: 08-May-2014].

JFree.org. JFreeChart. URL http://www.jfree.org/jfreechart/. [Accessed: 30-December-2013].

Joda.org. Joda-Time - Java date and time API. URL http://www.joda.org/joda-time/. [Accessed: 30-December-2013].

I. T. Jolliffe. *Principal Component Analysis*. Springer, Oct. 2002. ISBN 9780387954424.

A. J. Kalyanapu, D. R. Judi, T. N. McPherson, and S. J. Burian. Monte Carlo-based flood modelling framework for estimating probability weighted flood risk. *Journal of Flood Risk Management*, 5(1):37–48, 2012. ISSN 1753-318X. doi: 10.1111/j.1753-318X.2011.01123.x.

P.-S. Kam and A. W.-C. Fu. Discovering Temporal Patterns for Interval-based Events. In *Proceedings of the 2nd International Conference on Data Warehousing and Knowledge Discovery (DaWaK'00)*, pp. 317–326. Springer, 2000.

J. Kantola. *Knowledge Service Engineering Handbook*. CRC Press, May 2012. ISBN 9781439852941.

A. Karatzoglou, A. Smola, K. Hornik, and A. Zeileis. kernlab – An S4 Package for Kernel Methods in R. *Journal of Statistical Software*, 11(9):1–20, 2004. URL http://www.jstatsoft.org/v11/i09/.

E. J. Keogh and M. J. Pazzani. Scaling up dynamic time warping for datamining applications. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '00, pp. 285–289, New York, NY, USA, 2000. ACM. ISBN 1-58113-233-6. doi: 10.1145/347090.347153.

KISTERS. WISKI - Das Wasserwirtschaftliche Informationssystem KISTERS, Jan. 2013. URL http://www.kisters.de/wasser/software/wiski-messdatenmanagement.html. [Accessed: 22-January-2013].

V. Krysanova, D.-I. Müller-Wohlfeil, and A. Becker. Development and test of a spatially distributed hydrological/water quality model for mesoscale watersheds. *Ecological modelling*, 106(2):261–289, 1998.

Z. W. Kundzewicz. SUMMER 1997 FLOOD IN POLAND IN PERSPECTIVE. In O. F. Vasiliev, P. H. A. J. M. v. Gelder, E. J. Plate, and M. V. Bolgov, editors, *Extreme Hydrological Events: New Concepts for Security*, number 78 in NATO Science Series, pp. 97–110. Springer Netherlands, Jan. 2007. ISBN 978-1-4020-5739-7, 978-1-4020-5741-0. URL http://link.springer.com/chapter/10.1007/978-1-4020-5741-0_8.

Z. W. Kundzewicz, K. Szamalek, and P. Kowalczak. The Great Flood of 1997 in Poland. *Hydrological Sciences Journal*, 44(6):855–870, 1999. ISSN 0262-6667. doi: 10.1080/02626669909492285. URL http://www.tandfonline.com/doi/abs/10.1080/02626669909492285.

Landesumweltamt Brandenburg (LUA), editor. *Das Sommerhochwasser an der Oder 1997 - Fachbeiträge anläßlich der Brandenburger Ökologietage II*, Volume 16. Potsdam, Mar. 1997.

C. Lange, M. Schneider, M. Mutz, M. Haustein, M. Halle, M. Seidel, H. Sieker, and R. Hinkelmann. Model-based design for restoration of a small urban river. In *35th IAHR Congress, Chengdu, China*. Tsinghua University Press, 2013. ISBN 978-7-89414-588-8.

W. V. Leekwijck and E. E. Kerre. Defuzzification: criteria and classification. *Fuzzy Sets and Systems*, 108(2):159–178, Dec. 1999. ISSN 01650114. doi: 10.1016/S0165-0114(97)00337-0.

F. Leisch. Sweave: Dynamic generation of statistical reports using literate data analysis. In *Compstat*, pp. 575–580, 2002. URL `http://link.springer.com/chapter/10.1007/978-3-642-57489-4_89`. [Accessed: 04-June-2013].

C.-Y. Li and F. Molkenthin. Time Series Scenario Composition Framework in Supporting Environmental Simulation Tasks. In J. M. Gómez, M. Sonnenschein, U. Vogel, A. Winter, B. Rapp, and N. Giesen, editors, *Proceedings of the 28th Conference on Environmental Informatics - Informatics for Environmental Protection, Sustainable Development and Risk Management*, pp. 247–254. BIS-Verlag, Sept. 2014. ISBN 978-3-8142-2317-9.

S. Liang. *The Java Native Interface: Programmer's Guide and Specification*. Addison-Wesley Professional, Reading, Mass, 1st edition, June 1999. ISBN 9780201325775.

J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, DMKD '03, pp. 2–11, New York, NY, USA, 2003. ACM. doi: 10.1145/882082.882086.

D. Lo and S.-C. Khoo. Mining patterns and rules for software specification discovery. *Proc. VLDB Endow.*, 1(2):1609–1616, Aug. 2008. ISSN 2150-8097.

T. Lumley. *mitools: Tools for multiple imputation of missing data*, 2012. URL http://CRAN.R-project.org/package=mitools. R package version 2.2.

E. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7(1): 1–13, Jan. 1975. ISSN 0020-7373. doi: 10.1016/S0020-7373(75)80002-2.

MATLAB. MATLAB - The Language of Technical Computing - MathWorks. URL http://www.mathworks.de/products/matlab/. [Accessed: 26-October-2013].

J. Mennis and D. Guo. Spatial data mining and geographic knowledge discovery—An introduction. *Computers, Environment and Urban Systems*, 33 (6):403–408, 2009. URL http://www.sciencedirect.com/science/article/pii/S0198971509000817.

Met Office, National Meteorological Library and Archive. *Water in the Atmosphere*. Number NO. 3 in Fact Sheet (Met Office). Met Office, Exeter, 2005. URL http://www.metoffice.gov.uk/learning/library/publications/factsheets. [Accessed: 06-February-2013].

D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, and F. Leisch. *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*, 2012. URL http://CRAN.R-project.org/package=e1071. R package version 1.6-1.

Microsoft SQL Server. Microsoft SQL Server. URL http://www.microsoft.com/sqlserver. [Accessed: 25-October-2013].

S. Milborrow. *earth: Multivariate Adaptive Regression Spline Models*, 2011. URL http://CRAN.R-project.org/package=earth. Derived from mda:mars by Trevor Hastie and Rob Tibshirani. R package version 4.2.0.

H. J. Miller and J. Han, editors. *Geographic Data Mining and Knowledge Discovery, Second Edition*. CRC Press, 2nd edition, May 2009. ISBN 1420073974.

MODFLOW. MODFLOW and related programs. URL `http://water.usgs.gov/nrp/gwsoftware/modflow.html`. [Accessed: 22-January-2013].

F. Molkenthin. *WWW-based Hydroinformatics Systems*. Brandenburgische Technische Universität Cottbus Lehrstuhl Bauinformatik, Dec. 2000. ISBN 3934934048.

F. Molkenthin. Hydroinformatics — An Overview. Lecture Notes in IAHR-EGW Short Course Modeling of Hydrosystems, Fachgebiet Wasserwirtschaft und Hydrosystemmodellierung, TU Berlin, 2007.

F. Molkenthin, V. Notay, and C.-Y. Li. Hybrid Hydroinformatics System for an Interdisciplinary Research Project. In *33rd IAHR Congress, Vancouver, Canada*, 2009.

F. Molkenthin, C.-Y. Li, and K. V. Notay. Information Handling in Interdisciplinary, Hydroenvironment Engineering Projects. In P. Gourbesville, J. Cunge, and G. Caignaert, editors, *Advances in Hydroinformatics*, Springer Hydrogeology, pp. 65–76. Springer Singapore, Jan. 2014. ISBN 978-981-4451-41-3, 978-981-4451-42-0.

R. Moskovitch, D. Stopel, M. Verduijn, N. Peek, E. D. Jonge, and Y. Shahar. Analysis of ICU Patients Using the Time Series Knowledge Mining Method. In *IDAMAP 2007 Workshop*, Amsterdam, 2007. doi: 10.1.1.135.6057.

MUDAB. MUDAB database. URL `http://www.bsh.de/en/Marine_data/Environmental_protection/MUDAB_database/index.jsp`. [Accessed: 04-June-2013].

Multics. Multics. URL `http://www.multicians.org/`. [Accessed: 23-March-2013].

MySQL. MySQL :: The world's most popular open source database. URL `https://www.mysql.com/`. [Accessed: 03-June-2013].

F. Mörchen. Algorithms for time series knowledge mining. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pp. 668–673, New York, NY, USA, 2006a. ACM. ISBN 1-59593-339-5. doi: 10.1145/1150402.1150485.

F. Mörchen. *Time Series Knowledge Mining*. PhD thesis, Dept. of Mathematics and Computer Science, University of Marburg, Germany, 2006b. ISBN 3-89703-670-3.

F. Mörchen and A. Ultsch. Mining hierarchical temporal patterns in multivariate time series. *Proceedings of the 27th Annual German Conference in Artificial Intelligence (KI'04)*, pp. 127–140, 2004. doi: 10.1.1.88.603.

F. Mörchen and A. Ultsch. Optimizing time series discretization for knowledge discovery. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, KDD '05, pp. 660–665, New York, NY, USA, 2005. ACM. ISBN 1-59593-135-X. doi: 10.1145/1081870.1081953.

F. Mörchen and A. Ultsch. Efficient mining of understandable patterns from multivariate interval time series. *Data Mining and Knowledge Discovery*, 15(2):181–215, Oct. 2007. ISSN 1384-5810, 1573-756X. doi: 10.1007/s10618-007-0070-1.

F. Mörchen, A. Ultsch, and O. Hoos. Extracting interpretable muscle activation patterns with time series knowledge mining. *International Journal of Knowledge-Based & Intelligent Engineering Systems*, 9(3):197–208, 2005.

P. C. Nayak, K. P. Sudheer, D. M. Rangan, and K. S. Ramasastri. A neuro-fuzzy computing technique for modeling hydrological time series. *Journal of Hydrology*, 291(1-2):52–66, May 2004. ISSN 0022-1694. doi: 10.1016/j.jhydrol.2003.12.010.

Net Applications. Market share for mobile, browsers, operating systems and search engines | NetMarketShare. URL `http://www.netmarketshare.com/`. [Accessed: 22-March-2013].

C. G. Nevill-Manning and I. H. Witten. Identifying hierarchical structure in sequences: A linear-time algorithm. *Journal of Artificial Intelligence Research*, 7:67–82, Aug. 1997.

R. Nuswantoro, F. Diermanse, and F. Molkenthin. Probabilistic flood hazard maps for jakarta derived from a stochastic rain-storm generator. *Journal of Flood Risk Management*, June 2014. ISSN 1753-318X. doi: 10.1111/jfr3.12114.

S. Néelz and G. Pender. *Desktop review of 2D hydraulic modelling packages*. Environment Agency, Bristol, United Kingdom, July 2009. ISBN 978-1-84911-079-2.

S. Néelz and G. Pender. *Benchmarking the latest generation of 2D hydraulic modelling packages*. Environment Agency, Bristol, United Kingdom, Aug. 2013. ISBN 978-1-84911-306-9.

T. Oates, M. D. Schmill, and P. R. Cohen. Parallel and distributed search for structure in multivariate time series. In M. v. Someren and G. Widmer, editors, *Machine Learning: ECML-97*, number 1224 in Lecture Notes in Computer Science, pp. 191–198. Springer Berlin Heidelberg, Jan. 1997. ISBN 978-3-540-62858-3, 978-3-540-68708-5. URL http://link.springer.com/chapter/10.1007/3-540-62858-4_84.

Omegahat. The Omega Project for Statistical Computing. URL http://www.omegahat.org/. [Accessed: 12-March-2013].

Open Geospatial Consortium. Open Geospatial Consortium. URL http://www.opengeospatial.org/. [Accessed: 12-October-2013].

open TELEMAC-MASCARET. open TELEMAC-MASCARET. URL http://www.opentelemac.org/. [Accessed: 22-January-2013].

OpenGeoSys. OpenGeoSys. URL http://www.ufz.de/index.php?en=18345. [Accessed: 15-January-2013].

Oracle Database. Database 11g | Oracle Database 11g | Oracle. URL http://www.oracle.com/us/products/database/overview/index.html. [Accessed: 03-June-2013].

E. J. Plate. Flood risk and flood management. *Journal of Hydrology*, 267 (1–2):2–11, Oct. 2002. ISSN 0022-1694. doi: 10.1016/S0022-1694(02)00135-X. URL http://www.sciencedirect.com/science/article/pii/S002216940200135X.

PostGIS. PostGIS — Spatial and Geographic Objects for PostgreSQL. URL http://postgis.net/. [Accessed: 03-June-2013].

PostgreSQL. PostgreSQL: the world's most advanced open source database. URL http://www.postgresql.org/. [Accessed: 03-June-2013].

R. K. Price, K. Ahmad, and P. Holz. Hydroinformatics Concepts. In *Hydroinformatics Tools for Planning, Design, Operation and Rehabilitation of Sewer Systems*. Springer, June 1998. ISBN 9780792350972.

D. Pyle. *Data Preparation for Data Mining*. Morgan Kaufmann, 1999. ISBN 9781558605299.

QGIS Development Team. *QGIS Geographic Information System*. Open Source Geospatial Foundation, 2014. URL http://qgis.osgeo.org.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. URL http://www.R-project.org. ISBN 3-900051-07-0.

P. Racsko, L. Szeidl, and M. Semenov. A serial approach to local stochastic weather models. *Ecological Modelling*, 57(1–2):27–41, Oct. 1991. ISSN 0304-3800. doi: 10.1016/0304-3800(91)90053-4. URL http://www.sciencedirect.com/science/article/pii/0304380091900534.

C. A. Ratanamahatana, J. Lin, D. Gunopulos, E. Keogh, M. Vlachos, and G. Das. Mining Time Series Data. In O. Maimon and L. Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, pp. 1069–1103. Springer US, Jan. 2005. ISBN 978-0-387-24435-8, 978-0-387-25465-4.

RForge.net. JRI - Java/R Interface. URL https://rforge.net/JRI/index.html. [Accessed: 30-December-2013].

C. W. Richardson. Stochastic simulation of daily precipitation, temperature, and solar radiation. *Water Resources Research*, 17(1):182–190, 1981. ISSN 1944-7973. doi: 10.1029/WR017i001p00182. URL http://onlinelibrary.wiley.com/doi/10.1029/WR017i001p00182/abstract.

SAGA. SAGA - system for Automated Geoscientific Analyses. URL http://www.saga-gis.org/. [Accessed: 22-January-2013].

E. Schulte, D. Davison, T. Dye, and C. Dominik. A Multi-Language Computing Environment for Literate Programming and Reproducible Research. *Journal of Statistical Software*, 46(3):1–24, 2012. ISSN 1548-7660. URL http://www.jstatsoft.org/v46/i03. [Accessed: 04-June-2013].

V. N. Sharda, S. O. Prasher, R. M. Patel, P. R. Ojasvi, and C. Prakash. Performance of Multivariate Adaptive Regression Splines (MARS) in predicting runoff in mid-Himalayan micro-watersheds with limited data / Performances de régressions par splines multiples et adaptives (MARS) pour la prévision d'écoulement au sein de micro-bassins versants Himalayens d'altitudes intermédiaires avec peu de données. *Hydrological Sciences Journal*, 53(6):1165–1175, 2008. ISSN 0262-6667. doi: 10.1623/hysj.53.6.1165.

B. Shrestha, L. Duckstein, and E. Stakhiv. Fuzzy rule-based modeling of reservoir operation. *Journal of Water Resources Planning and Management*, 122(4):262–269, 1996. ISSN 0733-9496. doi: 10.1061/(ASCE)0733-9496(1996)122:4(262).

R. R. Shrestha, A. Bárdossy, and M. Rode. A hybrid deterministic-fuzzy rule based model for catchment scale nitrate dynamics. *Journal of Hydrology*, 342(1-2):143–156, Aug. 2007. ISSN 0022-1694. doi: 10.1016/j.jhydrol.2007.05.020.

S. W. Smith. *The scientist and engineer's guide to digital signal processing*. California Technical Publishing, San Diego, CA, USA, 1997. ISBN 0-9660176-3-3.

C. Snijders, U. Matzat, and U.-D. Reips. Big Data: Big Gaps of Knowledge in the Field of Internet Science. *International Journal of Internet Science*, 7 (1):1–5, Jan. 2012.

J. M. Spate, B. Croke, and A. J. Jakeman. Data Mining in Hydrology. In *Proceedings modelling and simulation society of Australia and New Zealand. MODSIM*, Volume 4, pp. 422–427, 2003. URL http://www.mssanz.org.au/MODSIM03/Volume_01/A06/04_Spate.pdf. [Accessed: 05-June-2013].

SPSS. IBM SPSS software. URL http://www-01.ibm.com/software/analytics/spss/. [Accessed: 26-October-2013].

SQLite. SQLite Home Page. URL https://sqlite.org/. [Accessed: 25-October-2013].

STATISTICA. StatSoft® Making the World More Productive®. URL http://www.statsoft.com/. [Accessed: 26-October-2013].

B. L. Stuart. *Principles of Operating Systems: Design & Applications*. Cengage Learning EMEA, 1st edition, Jan. 2008. ISBN 9781418837693.

M. Sugeno. *Industrial Applications of Fuzzy Control*. Elsevier Science Inc., New York, NY, USA, 1985. ISBN 0444878297.

SWAP-GIS. SWAP-GIS. URL http://www.orser.psu.edu/GISSupport/swapgis.htm. [Accessed: 28-May-2013].

"system". Merriam-Webster Online, May 2013. URL http://www.merriam-webster.com/dictionary/system. [Accessed: 23-May-2013].

D. Temple Lang and J. Chambers. *SJava: The Omegahat interface for R and Java*, Mar. 2013. URL http://www.bioconductor.org/packages/2.12/bioc/html/SJava.html. Bioconductor package version 0.85.2.

M. Theus and S. Urbanek. *Interactive Graphics for Data Analysis: Principles and Examples*. Chapman and Hall/CRC, 1st edition, Oct. 2008. ISBN 1584885947.

TOP500. Project | TOP500 Supercomputer Sites. URL http://www.top500.org/project/. [Accessed: 26-March-2013].

E. Ukkonen. On-Line Construction of Suffix Trees. *Algorithmica*, 14(3):249–260, 1995. doi: 10.1.1.10.751.

A. Ultsch. Data Mining and Knowledge Discovery with Emergent Self-Organizing Feature Maps for Multivariate Time Series. In *Kohonen Maps*, pp. 33–46. New York, NY: Elsevier Science, 1999.

A. Ultsch. *U*-matrix: a tool to visualize clusters in high dimensional data*. Fachbereich Mathematik und Informatik, 2003. URL http://www.informatik.uni-marburg.de/~databionics/papers/Ultsch09U-Matrix.pdf. [Accessed: 28-November-2012].

A. Ultsch. Unification-based temporal grammar. Technical Report 37, Philipps-Universität Marburg, Germany, 2004.

S. Urbanek. How to talk to strangers: ways to leverage connectivity between R, Java and Objective C. *Computational Statistics*, 24(2):303–311, May 2009. ISSN 0943-4062, 1613-9658. doi: 10.1007/s00180-008-0132-x.

S. Urbanek. *rJava: Low-level R to Java interface*, 2011. URL http://CRAN.R-project.org/package=rJava. R package version 0.9-3.

S. Urbanek. *Rserve: Binary R server*, 2012. URL http://CRAN.R-project.org/package=Rserve. R package version 0.6-8.

US Army Corps of Engineers. HEC-RAS by US Army Corps of Engineers. URL http://www.hec.usace.army.mil/software/hec-ras/. [Accessed: 12-January-2013].

J. Vilo. Discovering Frequent Patterns from Strings. Technical report, Department of Computer Science, University of Helsinki, 1998.

W3C – World Wide Web Consortium. SOAP specifications. URL http://www.w3.org/TR/soap/. [Accessed: 08-April-2013].

WaterML2. WaterML2 - A Global Standard for Hydrological Time Series. URL http://www.waterml2.org/. [Accessed: 01-June-2013].

R. Wehrens and L. M. C. Buydens. Self- and Super-organising Maps in R: the kohonen package. *J. Stat. Softw.*, 21(5), 2007. URL http://www.jstatsoft.org/v21/i05.

M. I. Whiteman, C. H. Maginness, R. P. Farrell, P. J. A. Gijsbers, and M. Ververs. The National Groundwater Modelling System: providing wider access to groundwater models. *Geological Society, London, Special Publications*, 364(1):49–63, Jan. 2012. ISSN 0305-8719, 2041-4927. doi: 10.1144/SP364.5.

H. Wickham. Reshaping data with the reshape package. *Journal of Statistical Software*, 21(12), 2007. URL http://www.jstatsoft.org/v21/i12/paper.

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer New York, 2009. ISBN 978-0-387-98140-6. URL http://had.co.nz/ggplot2/book.

H. Wickham. *scales: Scale functions for graphics.*, 2012. URL http://CRAN.R-project.org/package=scales. R package version 0.2.3.

Wikipedia, the free encyclopedia. Usage share of operating systems, Mar. 2013. URL http://en.wikipedia.org/w/index.php?title=Usage_share_of_operating_systems&oldid=546944439. Page Version ID: 546944439.

D. S. Wilks and R. L. Wilby. The weather generation game: a review of stochastic weather models. *Progress in Physical Geography*, 23 (3):329–357, Sept. 1999. ISSN 0309-1333, 1477-0296. doi: 10.1177/030913339902300302. URL http://ppg.sagepub.com/content/23/3/329.

I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2nd edition, June 2005. ISBN 0120884070.

XML-RPC.com. XML-RPC Homepage. URL http://xmlrpc.scripting.com. [Accessed: 08-April-2013].

R. Yager and D. Filev. Generation of fuzzy rules by mountain clustering. *Journal of Intelligent and Fuzzy Systems*, 2(3):209–219, 1994.

J. Yan. *som: Self-Organizing Map*, 2010. URL http://CRAN.R-project.org/package=som. R package version 0.3-5.

X. Yan, J. Han, and R. Afshar. CloSpan: Mining Closed Sequential Patterns in Large Datasets. In *In SDM*, pp. 166–177, 2003.

J. Yang, W. Wang, P. S. Yu, and J. Han. Mining long sequential patterns in a noisy environment. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, SIGMOD '02, pp. 406–417, New York, NY, USA, 2002. ACM. ISBN 1-58113-497-5. doi: 10.1145/564691.564738.

L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, June 1965. ISSN 0019-9958. doi: 10.1016/S0019-9958(65)90241-X. URL http://www.sciencedirect.com/science/article/pii/S001999586590241X.

M. J. Zaki and C.-j. Hsiao. CHARM: An efficient algorithm for closed itemset mining. In *Proceedings of 2nd SIAM International Conference on Data Mining*, pp. 457–473, 2002.

E. Zehe and R. Hinkelmann. DFGForschergruppe 581 "Großhang": Kopplung von Strömungs- und Deformationsprozessen für die Modellierung von Großhangbewegungen. Abschlussbericht, 2013.

A. Zeileis and G. Grothendieck. zoo: S3 Infrastructure for Regular and Irregular Time Series. *Journal of Statistical Software*, 14(6):1–27, 2005. URL `http://www.jstatsoft.org/v14/i06/`.

Y. L. Zhu, S. J. Li, N. N. Bao, and D. S. Wan. Mining approximate periodic pattern in hydrological time series. In *EGU General Assembly Conference Abstracts*, Volume 14, p. 515, 2012.

K. Zuse. *The Computer - My Life*. Springer, 1993 edition, Nov. 1993. ISBN 3540564535.

E. C. Özelkan and L. Duckstein. Fuzzy conceptual rainfall-runoff models. *Journal of Hydrology*, 253(1-4):41–68, Nov. 2001. ISSN 0022-1694. doi: 10.1016/S0022-1694(01)00430-9.

Glossary

Aspect

In TSKR, an Aspect is a group of time series data sets sharing similar semantics.

Chord

In TSKR, a Chord represents the coincidence of Tones.

coincidence

In TSKR, the term "coincidence" is used to describe the temporal overlapping time period of Tones or Chords.

data

Data are qualitative or quantitative descriptions of information or facts and can be stored in digital or nondigital carriers.

duration

In TSKR, the term "duration" is used to describe how long the three components in TSKR, Tone, Chord, and Phrase, last.

Event

In the framework of time series scenario composition, an Event is the same to a Chord in TSKM which represents the coincidence of Tones

but with different semantic meaning describing a certain feature of a scenario.

expressivity

In TSKR, it is one of three major criticisms on Allen's relations that the same relations in Allen's relations may have very different manifestations.

information

Information is data together with the semantics derived from data themselves which describes certain correlations, such as patterns, associations, relationships, etc.

interpretability

In TSKR, it is one of three major criticisms on Allen's relations which indicates that the relations described by Allen's relations are ambiguous and difficult to comprehend due to additional information needed and the complexity of description caused by it.

knowledge

Knowledge is something human beings have learned or acquired through exposure of data or information, and it can be applied repeatedly in an empirical or a theoretical manner.

MetaEvent

In the framework of time series scenario composition, a MetaEvent is combination of an Event and its metadata based on the history, e.g. min. and Max. values, describing the characteristics of the extracted features, and it serves as the most basic unit of a measurement for scenario composition.

MetaEventEntity

In the framework of the time series scenario composition, a MetaEventEntity is a realization of the MetaEvent it belongs to. It serves as the basic element for composing scenarios.

partial order

In TSKR, the term "partial order" describes the ordering of Chords with binary relation inside a phrase by analogy with the one in the order theory.

Phrase

In TSKR, a Phrase represents the concept of partial order of nonoverlapping Chords which describes how a phenomenon develops.

robustness

In TSKR, one of three major criticisms that Allen’s relations suffer is the lack of robustness. This is because they need at least two endpoints to determine relations. While taking into account of measurement errors at the boundaries, it is difficult to determine relations of intervals.

semi-automatic

A semi-automatic process in the context of this research means that users have rights to and usually have to take part in the steps inside the process. This involvement may contain a judging of the outputs from one step which are the inputs of another step, and users have the right to decide if these outputs can be sent to the next step as inputs. If not, the step may be repeated again until the acceptable results are derived. This process causes nonlinear workflow inside the process.

support of a Chord

The support of a Chord used in TSKR is a property of a Chord which describes the longest common interval of all Tones in a Chord.

Tone

In TSKR, a Tone describes a specific property or state of an Aspect within a given time duration.

Acronyms

A

AI Artificial Intelligence
ANFIS Adaptive Neuro Fuzzy Inference System
ANN Artificial Neural Network
API Application Programming Interface
ARMA Autoregressive-Moving-Average
ARPANET Advanced Research Projects Agency Network

B

BAW Bundesanstalt für Wasserbau
BC Boundary Condition
BCE Björnsen Consulting Engineers

C

CAD Computer-Aided Design
CAM Computer-Aided Manufacturing
CERN European Organization for Nuclear Research
CETMEF Centre d'Etudes Techniques Maritimes et Fluviales
CHARM Closed Association Rule Mining
CI Computer Intelligence
CloSpan CLOsed Sequential PAtterN mining
CoA Center of Area

CoG Center of Gravity
CoGS Center of Gravity for Singleton
CPU Central Processing Unit
CRAN Comprehensive R Archive Network
CSV Comma-Separated Values

D

datPAV data Processing, Analysis and Visualization
DBMS Database Management System
DDM Data-Driven Modeling
DHI Danish Hydraulic Institute
DMS Document Management System
DSL Domain-Specific Language
DSS Decision Support System

E

ECHO Earth Observing System (EOS) Clearing House
EDF Electricité de France
ELWIS Elektronischer Wasserstraßen-Informationsservice
EM Expectation Maximization
EOS Earth Observing System
ESOM Emergent Self-Organizing Map
ESS Emacs Speaks Statistics
EU European Union

F

FCL Fuzzy Control Language
FIS Fuzzy Inference System
FLYS Flusshydrologische Software
FM Fuzzy Mean
FoM First of Maximum
FTP File Transfer Protocol

G

GA Genetic Algorithm
GAS GNU Assembler
GCM Global Climate Model
GCV Generalized Cross Validation
GE General Electric
GIS Geographic Information System
GML Geography Markup Language
GMM Gaussian Mixture Model
GP Genetic Programming
GPL General-Purpose Language
GPU Graphics Processing Unit
GUI Graphical User Interface

H

HHT Hilbert-Huang Transform
HMM Hidden Markov Model
HTML HyperText Markup Language
HTTP Hypertext Transfer Protocol
HTTPS Hypertext Transfer Protocol Secure

I

IC Initial Condition
ICA Independent Component Analysis
ICT Information and Communication Technology
IDE Integrated Development Environment
IM Instant Messaging
IMAP Internet Message Access Protocol
IPCC Intergovernmental Panel on Climate Change
IWRM Integrated Water Resources Management

J

JNI Java Native Interface
JRI Java/R Interface
JSON JavaScript Object Notation
JVM Java Virtual Machine

K

KDD Knowledge Discovery in Databases

L

LAN Local Area Network
LoM Last of Maximum
LTE Long-Term Evolution

M

MARS Multivariate Adaptive Regression Splines
MASM Microsoft Macro Assembler
MIT Massachusetts Institute of Technology
ML Machine Learning
MoM Middle of Maximum
MSDD Multi-Stream Dependency Detection
MUDAB Meeresumwelt-Datenbank
Multics Multiplexed Information and Computing Service

N

NaN Not A Number
NAS Network-Attached Storage
NASA National Aeronautics and Space Administration
NGMS National Groundwater Modelling System

O

OCG Open Geospatial Consortium
ODF Open Document Format for Office Applications
OOP Object-Oriented Programming

OpenMI Open Modeling Interface
OQL Object Query Language
ORDBMS Object-Relational Database Management System
OS Operating System

P

PAA Piecewise Aggregate Approximation
PC Personal Computer
PCA Principal Component Analysis
PDA Personal Digital Assistant
PDF Portable Document Format
PMML Predictive Model Markup Language
PNG Portable Network Graphics
POP Post Office Protocol
POSIX Portable Operating System Interface
PS Parameter Set

R

RAID Redundant Array of Independent Disks
RAM Random-Access Memory
RDBMS Relational Database Management System
REPL Read-Eval-Print Loop
RMSE Root-Mean-Square Error
RSS Residual Sum of Squares

S

SAX Symbolic Aggregate approXimation
SBC Single-Board Computer
SDI Spatial Data Infrastructure
SDK Software Development Kit
SIMPLE Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions
SIP Session Initiation Protocol

SMB Server Message Block
SMTP Simple Mail Transfer Protocol
SOAP Simple Object Access Protocol
SOM Self-Organizing Map
SQL Structured Query Language
SSD Solid-State Drive
SSH Secure Shell
SSL Secure Sockets Layer
SUS Single UNIX Specification
SVM Support Vector Machine

T

TLS Transport Layer Security
TSKM Time Series Knowledge Mining
TSKR Time Series Knowledge Representation
TUI Text-based User Interface

U

UFZ Helmholtz-Zentrum für Umweltforschung
UI User Interface
UMTS Universal Mobile Telecommunications System
USGS United States Geological Survey
UTG Unification-based Temporal Grammar

V

VoIP Voice over Internet Protocol
VR Virtual Reality

W

WAN Wide Area Network
WEKA Waikato Environment for Knowledge Analysis
WGS World Geodetic System
WPS Web Processing Service

WWW World Wide Web

X

XLS Microsoft Excel Spreadsheet
XML Extensible Markup Language
XMPP Extensible Messaging and Presence Protocol