Supporting Open Science in Big Data Frameworks and Data Science Education

by

Michael E. Cotterell

(Under the Direction of John A. Miller)

Abstract

As the prevalence of data grows throughout the Big Data era, so does a need to provide and improve tools for the education and application of data-driven analytics and scientific investigation. The main contributions of this research can be summarized as follows: i) We provide an overview of the open source ScalaTion project, a big data framework that supports big data analytics, simulation modeling, and functional data analysis. ii) We outline some of the Functional Data support in ScalaTion, including a performance comparison for the evaluation of B-spline basis functions that shows that our method is faster than some other popular libraries. iii) To demonstrate how to provide lightweight big data framework integration in open notebooks, we present the open source ScalaTion Kernel project, a custom Jupyter kernel that enables ScalaTion support in Jupyter notebooks. iv) To demonstrate research using ScalaTion, we outline and evaluate a

tight clustering algorithm, written using ScalaTion, for the functional data analysis of time course omics data. v) To promote reproducibility in open science, we present the Applied Open Data Science (AODS) project, a collection of customized web applications for the hosting and sharing of open notebooks with ScalaTion support. This project also includes shareable, executable, and modifiable example notebooks that utilize ScalaTion to demonstrate various data science topics as well as detailed documentation on how to easily reproduce the environment in which the notebooks are hosted. Specifically, we propose and demonstrate, via readily accessible examples, methods to facilitate openness and reproducibility (both of results and infrastructure) in data science investigations using a big data framework.

INDEX WORDS: open science, open notebooks, big data frameworks, data science, data science education

Supporting Open Science in Big Data Frameworks and Data Science Education

by

Michael E. Cotterell

B.S., University of Georgia, 2011

A Dissertation Submitted to the Graduate Faculty
of The University of Georgia in Partial Fulfillment
of the
Requirements for the Degree

Doctor of Philosophy

Athens, Georgia

2017

Supporting Open Science in Big Data Frameworks and Data Science Education

by

Michael E. Cotterell

Approved:

Major Professor: John A. Miller

Committee: Maria Hybinette
Yi Hong
Thiab Taha

Electronic Version Approved:

Suzanne Barbour
Dean of the Graduate School
The University of Georgia
December 2017

Supporting Open Science in Big Data Frameworks and Data Science Education

Michael E. Cotterell

December 2017

This dissertation is dedicated to my parents, John and Emily, and to my Aunt and Uncle, Sheila and Jim.

Acknowledgments

I would like to thank all those who, over the years, have helped me progress towards the completion of my doctoral degree. This includes my lab mates, Mustafa, Hao, and Nick, as well as my colleague Xiaoxiao, who all contributed directly to different aspects of my research. While too numerous to enumerate, this also includes the other lab mates, close friends, family members, and lizard people (i.e., my Tuesday trivia group) who helped keep me motivated towards the completion of my goals. I would also like to thank Ping Ma for his guidance on the statistical aspects of my dissertation, including functional data analysis.

I would like to thank my committee members, Maria Hybinette, Yi Hong, and Thiab Taha for their guidance. Their wise and generous constructive criticism was extremely helpful. Without their feedback, this dissertation would not have been possible. I would like to especially thank my major professor, John A. Miller. He served as my faculty advisor during my undergrad and suggested that I take a directed study course under him. I did. That directed study was the catalyst that propelled me toward research in computer science. In my opinion, John's general approach to problem solving and teaching is a healthy balance between pragmatism

and practicality, both of which have been nothing less than inspirational. It has been an honor and a privilege working with him.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The recent proliferation of Big Data and open science initiatives brings with it a need for the provision and improvement of tools for the education and application of data-driven analytics and scientific investigation. This is evidenced by the recent growth in Data Science positions in major corporations as well as recent initiatives by the Computer Science Education (SIGCSE) community (e.g., "CS for All", etc.) and multiple programs supported by the National Science Foundation (NSF) under their "Big Data" initiative portfolio[1] (e.g., NRT, ITEST, S-STEM, STEM+C, etc.).

Some important open research questions in data science education include: How do you provide tools to support open science, big data, and reproducibility? What computing infrastructure is available and how do you use it? How do you make it easier for students and domain experts to do data science and big data analytics? For each of these questions, we discuss current roadblocks and potential

[1] https://www.nsf.gov/cise/bigdata/

solutions. Specifically, we describe and exemplify how big data frameworks and open science can be combined to help mitigate some of the problems related to these questions. The main contributions of this research can be summarized as follows:

- We provide an overview of the open source ScalaTion project, a big data framework that supports big data analytics and simulation modeling.
- To demonstrate how to provide lightweight big data framework integration in open notebooks, we present the open source ScalaTion Kernel project, a custom Jupyter kernel that enables ScalaTion support in Jupyter notebooks.
- To demonstrate applied research using ScalaTion, we outline and evaluate a tight clustering algorithm, written using ScalaTion, for the functional data analysis of time course omics data.
- To promote reproducibility in open science, we present the Applied Open Data Science (AODS) project, a collection of customized web applications for the hosting and sharing of open notebooks with ScalaTion support. This project also includes shareable, executable, and modifiable example notebooks that utilize ScalaTion to demonstrate various data science topics as well as detailed documentation on how to easily reproduce the environment in which the notebooks are hosted.

Document Features

Throughout this dissertation, various links to external online sources are included. These links are clickable in the Portable Document Format (PDF) version of this document. If the link URL is not displayed inline with the surrounding text, then it is provided in a footnote on the same page. Some of the links refer to example Jupyter notebooks that require the ScalaTion Kernel described in this dissertation. A general description of Jupyter notebooks is provided in Chapter 2, and an overview of the ScalaTion Kernel is provided in Chapter 4. If the reader wishes to try out some of the notebooks without installing anything on their local machine, then they should refer to Chapter 6 for information on how to use the applications and examples provided on the author's Applied Open Data Science (AODS) JupyterHub[2] website. Interested readers are also encouraged to see Chapter 4 for information on how to setup their own Jupyter installation with ScalaTion Kernel support, either locally or as a containerized application.

Roadmap

The rest of this dissertation is organized as follows: some background material is provided in Chapter 2 in order to motivate the research being presented and better lead into the individual manuscripts; overviews of the ScalaTion and ScalaTion Kernel projects are provided in Chapters 3 and 4, respectively; a tight clustering algorithm, implemented using ScalaTion, is presented and evaluated in Chapter 5; the Applied Open Data Science (AODS) project and example notebooks are pre-

[2] `http://hub.aods.io/`

sented in Chapter 6; and Chapter 7 provides a brief summary of the entire work.

Chapter 2

Background & Challenges

In this section we present some background material related to the research. Emphasis will be placed, as needed, on details pertaining closely to the manuscripts presented in this dissertation.

2.1 Big Data

The Oxford English Dictionary [2017] defines "big data" as "data of a very large size, typically to the extent that its manipulation and management present significant logistical challenges; (also) the branch of computing involving such data." Although originally coined by John Mashey in the 1990s [Steve Lohr, 2013], the term now generally refers to any kind of large-sized data used with different analytics techniques such as prediction, classification, clustering, etc. [Boyd and Crawford, 2011]. Recent surveys on the topic of big data are provided in Chen et al. [2014], Sri and Anusha [2016], Khan et al. [2014], and Ward and Barker [2013].

2.2 Data Science

Hayashi [1998] defines "data science" as, "not only a synthetic concept to unify statistics, data analysis, and their related methods but also comprises their results." The term generally refers to data-driven scientific investigation and its related activities, an endeavor that has become more and more popular in recent years due to the proliferation of Big Data [Provost and Fawcett, 2013; Dhar, 2013] and Analytics [Waller and Fawcett, 2013]. A good introduction to data science can be found in Hey et al. [2009]. Recent surveys in data science are provided in Cao [2017] and Blum et al. [2016].

2.3 Open Science

The term "open science" refers to efforts being made by the greater scientific community to make the activities involved in scientific investigations more accessible. This includes publicly accessible hosting of datasets, investigation procedures, and results in a manner that enables, encourages, and promotes reproducibility. Efforts include: i) making research articles publicly available for free via open access [Nicholas et al., 2005; Laakso et al., 2011]; ii) the use of open source software such as R [Ihaka and Gentleman, 1996], Spark [Zaharia et al., 2010], ScalaTion [Miller et al., 2010], and Python [Van Rossum and Drake, 2011]. iii) the use of services that help facilitate open development such as GitHub [Dabbish et al., 2012]; and iv) promoting reproducibility by making analyses available in open notebook formats such as Jupyter [Ragan-Kelley et al., 2014]. With respect

to (i), adoption of the open access model by journals has increased recently due to the National Science Foundation's Public Access Plan, which requires that journal articles produced in relation to NSF-funded research be made freely available for download, reading, and analysis within a year of publication [National Science Foundation, 2015]. In an effort to promote the same kind of openness in all academic research, regardless of funding, the Open Science Framework organization recently published, openly, their guidelines for transparency and openness promotion (TOP) [Alter et al., 2016; Nosek et al., 2016] in journals, which requires authors to properly cite and or make available the research data used in their articles. Elsevier, one of the journal publishers that has adopted these guidelines, defines research data as, "the results of observations or experimentation that validate research findings,"[Elsevier, 2017], including, "raw data, processed data, software and algorithms." Recently, there has also been an annual Open Science in Big Data (OSBD) workshop[1] at the IEEE Big Data Conference.

2.4 Domain-Specific Language

Deursen et al. [2000] defines "domain-specific languages" (DSLs) as, "programming languages or executable specification languages that offer, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain." The general idea behind DSLs is that they enable programmers to write code that more closely resembles domain literature instead of forcing then to fit a domain-specific problem or specification into a traditional

[1] `https://osbd.github.io/`

programming paradigm [Hofer and Ostermann, 2010; Hofer et al., 2008]. For example, consider the dot product operation on vectors from the domain of linear algebra. The domain literature might use $\mathbf{x} \cdot \mathbf{y}$ to express the dot product of the vectors $\mathbf{x}$ and $\mathbf{y}$. Using the DSL provided by ScalaTion, a big data framework for Scala described in Chapter 3, the code for the same dot product is written as `x dot y` or, more concisely, as `x · y` by making use of a Unicode character for the `dot` function. Unicode support in DSLs is a natural way to help express domain-specific notations in programming [Cotterell et al., 2011b].

2.5 Big Data Frameworks

Consistent with the definition by Tekiner and Keane [2013], "big data frameworks" are software libraries that enable applications to: i) aggregate and filter big data; ii) generate and fit analytics and simulation models; and iii) organize and interpret results. They usually have support for parallel and distributed operations, stream processing, and a battery of different modeling techniques for predictive analytics, machine learning, data mining, and/or simulation. As some of these terms are related and may be ambiguous to the reader, we provide a short definition for each as follows: i) predictive analytics generally refers to predictive model development (e.g., regression, time series, classification, clustering, etc.) and estimation, usually for business purposes [Finlay, 2014]; ii) machine learning is data-driven predictive analytics that can be either supervised (i.e., makes use of labeled data) or unsupervised (i.e., uses unlabeled data) [Kohavi and Provost, 1998]; iii) data

mining usually refers to unsupervised machine learning techniques where the intent produce inferences (e.g., patterns and relationships) about the data [Chakrabarti et al., 2006]; and simulation refers to continuous and/or discrete-event representations of processes that incorporate knowledge about data inputs and how they interact [Miller et al., 2013b]. Additionally, support for linear algebra operations, relational queries, and graph-based queries is usually provided by big data frameworks as well. A feature matrix for some popular open source big data frameworks is presented in Table 2.1.

Among the popular big data frameworks presented in Table 2.1, many are maintained by the American non-profit Apache Software Foundation[2], notably including Hadoop[3] and Spark[4]. Hadoop is a collection of projects founded by Doug Cutting and Mike Cafarella around 2006 that provide a MapReduce and distributed file system implementation [Wang et al., 2014] as well as a platform for cluster resource allocation and scheduling [Vavilapalli et al., 2013]. Spark, originally developed at UC Berkley, is a framework for providing implicit support for parallel and distributed operations regardless of paradigm [Zaharia et al., 2010]. Another interesting characteristic among the popular big data frameworks is programming language support. After Java, the second most prevalent language supported in Table 2.1 is Python. This is no surprise considering Python's recent growing popularity among data scientists [Puget, 2016]. The R programming language, while also popular among data scientists, does not appear much

[2] https://www.apache.org/

[3] http://hadoop.apache.org/

[4] http://spark.apache.org/

Table 2.1: Some Open Source Big Data Frameworks

For each open source big data framework, various support and features are included. Abbreviations: PAR = Parallel Operations; DOP = Distributed Operations; DDA = Distributed Data; STR = Stream Processing; RED = Dimensionality Reduction; REG = Regressions; CLA = Classification; CLU = Clustering; SQL = SQL and/or Relational Algebra; GRA = Graph Queries; TSF = Time Series; and SIM = Simulation. Languages: S = Scala; J = Java; P = Python; R = R; C = C and/or C++; G = Go; and O = Other language support. A framework must provide direct support and not merely facilitate implementation. Only currently maintained (as of November 2017) frameworks with open source distribution licenses approved by the Open Source Initiative (OSI) are included in this table.

Name	**Lang.**	PAR	DOP	DDA	STR	RED	REG	CLA	CLU	SQL	GRA	TSF	SIM
Accord.NET 3.8.0	O	Y	N	N	N	N	Y	Y	Y	N	N	N	N
Apache Drill 1.11	JCO	Y	Y	Y	Y	N	N	N	N	Y	N	N	N
Apache Flink 1.3.2	SJ	Y	Y	Y	Y	N	N	N	N	Y	N	N	N
Apache Giraph 1.2.0	J	Y	Y	Y	Y	N	N	N	N	N	Y	N	N
Apache Hadoop 2.8.2	J	Y	Y	Y	N	N	N	N	N	Y	N	N	N
Apache Hive 2.3.0	J	Y	Y	Y	Y	N	N	N	N	Y	N	N	N
Apache Impala 2.10	C	Y	Y	N	N	N	N	N	N	Y	N	N	N
Apache Mahout 0.13.0	SJO	Y	Y	Y	N	Y	N	Y	Y	N	N	N	N
Apache Pig 0.17.0	O	Y	N	N	Y	N	N	N	N	Y	N	N	N
Apache Samza 0.13	SJ	Y	Y	N	Y	N	N	N	N	N	N	N	N
Apache SINGA 1.1.0	JPC	Y	Y	N	N	N	Y	Y	Y	N	N	N	N
Apache Spark 2.1.2	SJPR	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N
Apache Storm 1.0.5	J	Y	Y	N	Y	N	N	N	N	Y	N	N	N
Caffe 1.0	PC	Y	Y	Y	N	N	N	Y	N	N	N	N	N
DistributedR 1.2.0	R	Y	Y	Y	N	Y	Y	Y	Y	N	Y	Y	N
Google TensorFlow 1.4	JPCG	Y	Y	Y	Y	Y	Y	Y	Y	N	N	Y	N
Microsoft CNTK 2.2	JPCO	Y	Y	N	N	N	Y	Y	Y	N	N	Y	N
Neo4j 3.4.0	J	Y	N	N	Y	N	N	N	N	N	Y	N	N
ScalaTion 1.4	S	Y	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
Theano 0.9	P	Y	N	N	Y	Y	N	N	N	N	N	N	N
Torch	CO	Y	N	N	Y	N	N	Y	Y	Y	N	N	N
PyTorch	P	Y	N	N	Y	N	N	Y	Y	Y	N	N	N

in Table 2.1. This suggests that there is either little developer interest in making a big data framework for R, with DistributedR being a notable exception, or that data scientists using R simply tend to make use of existing R packages.

2.6 Challenges in Data Science Education

In this section we present some challenges encountered in Data Science Education. Emphasis will be placed, as needed, on details pertaining closely to the manuscripts presented in this dissertation.

2.6.1 Storage and Access

The generation of big data, either by scientific investigation or by corporate data collection, brings with it two related challenges with regard to its use in data science curricula: i) How do you store the data?; and ii) How do you make the data accessible for analytics?

Storage is one of the most cited challenges when dealing with big data, as evidenced in [Chen and Zhang, 2014], [Katal et al., 2013], [Kaisler et al., 2013], [Siddiqa et al., 2017], and [Marx, 2013]. Where can data science students and instructors store their data? While recent advancements in nurturing the Nation's advanced cyberinfrastructure (discussed in Section 2.6.2) may eventually help mitigate the issue, this question still poses real-world logistical problems in the implementation of data science curricula. One potential solution to this problem is

via the adoption of container clusters using platforms like Docker[5] and Kubernetes[6] [Bernstein, 2014]. Containers are similar to lightweight virtual machines, except they usually virtualize at the operating system kernel's user-space level instead of at the hardware level [Merkel, 2014; Fink, 2014]. Container clusters are clusters that are setup to deploy containers across the cluster's infrastructure [Pahl and Lee, 2015]. With container clusters, it should be possible for instructors and researchers to deploy different configurations of containerized architectures at different scales. Consider a use case where a student or researcher wants to evaluate the performance of a distributed algorithm. They can setup a container for the evaluation experiment, test it locally, then deploy it to the container cluster in order to actually see the speedup associated with distributed algorithms. Furthermore, they can adjust their containerized architecture to see the speedup (or lack thereof) at different, reproducible, and readily available scales.

If instructors and students need to store their own big data, then one potential solution is the use of open tools that provide distributed file systems. For example, both the Dat Project[13] and Apache HDFS/YARN [Vavilapalli et al., 2013] are openly available, free for non-commercial use, and support a flexible array of configurations. However, the use of distributed file systems does require access to

[5] https://www.docker.com/
[6] https://kubernetes.io/
[7] https://catalog.data.gov/dataset
[8] https://www.kaggle.com/datasets
[9] https://archive.ics.uci.edu/ml/index.php
[10] http://www.internationalgenome.org/data
[11] http://proteomics.ucsd.edu/ProteoSAFe/datasets.jsp
[12] https://cloud.google.com/bigquery/public-data/
[13] https://datproject.org

Table 2.2: Some Examples of Open and/or Accessible Big Data

For each source, a short description, number of datasets, and general access URL is provided. The number of datasets indicated is based on statistics gathered in early November, 2017.

Source	**Description**	**Datasets**
Data.gov[7]	Federated U.S. government data.	197,993
Kaggle[8]	Open Data for Machine Learning.	1,261
UCI-ML Repository[9]	Machine Learning datasets.	394
1000 Genomes Proj.[10]	Human variation and genotype data.	2,504
Proteome eCommons[11]	Proteomics data.	8,037
Google BigQuery[12]	Varies.	Terabytes

multiple computers and storage devices. Such a setup may not be feasible in some situations. A better solution is to provide data science students and instructors easier access to existing datasets. Some examples of open and/or accessible big data are provided in Table 2.2. While all of the examples in this table constitute open data, some differ with respect to their big data characterization. Some are big in the quantity of datasets provided while others are big in actual file size. Data from the 1000 Genomes Project actually falls into both of these characterizations, coming in at more than 200 terabytes in size for 2,600 human genome datasets [Clarke et al., 2012].

2.6.2 Cyberinfrastructure

Over the past decade, the United States has transitioned from a funding model that encourages the creation and curation of individual computing resources for facilitating the pursuit of scientific investigations to a model that, instead, encourages the use of a sustainable, national advanced cyberinfrastructure (CI), broadly defined as the resources, tools, and services for advanced computation, data handling, networking and security. This includes technologies that support data science within a highly interoperable and collaborative ecosystem. The advancement of national, advanced CI is progressing, as evidenced by existing NSF CI and research projects such as XSEDE [Towns et al., 2014], NanoHub [Klimeck et al., 2008], CyVerse (formerly iPlant Collaborative [Goff et al., 2011]), LIGO [Althouse et al., 1992], and NHERI DesignSafe [Rathje et al., 2017]. An important challenges to address with regard to advanced CI how to incorporate it in data science curricula.

There is clear evidence that training and development programs are needed, as highlighted by the National Strategic Computing Initiative [Obama, 2015], the National Academies' report on Future Directions for NSF Advanced Computing Infrastructure to Support U.S. Science and Engineering in 2017-2020 [National Academies of Sciences, Engineering, and Medicine, 2016], and the Federal Big Data Research and Development Strategic Plan [Beninson et al., 2016]. Although infrastructure-specific training opportunities currently exist, they still suffer from two key problems that unintentionally limit collective impact: (i) considerable overlap exists in the training material for the different opportunities; and (ii)

while clearly defined and assessable, these training opportunities are often tailored specifically to their specific infrastructure environments. These problems are real, non-disjoint, and solvable in a way that complements these existing programs.

One potential solution to this challenge is the development of undergraduate CI courses and integrative community outreach programs. While course development would be primarily targeted at the undergraduate level, there would be some natural overlap to the graduate level (e.g., Dual BS/MS and aspects of MS programs) as well as to professionals seeking additional training. Each of these proposed programs are outlined below:

- **Course Development:** We propose the development and adoption of two courses, one undergraduate course and one graduate course, that focus on the application of advanced CI to applied data science. The first course emphasizes training aspect so that students understand how to apply data science investigations using advanced CI. The second course emphasizes algorithms, paradigms, and performance metrics for advanced CI. Both courses are to be developed with the idea of easily providing a partially or fully online component as well as make use of modern pedagogical approaches when taught in the classroom. A brief overview of each course, including details regarding pedagogy, is provided in Appendix A.

- **Integrative Community Outreach:** One aspect of this proposal includes two community outreach programs designed to raise awareness of data science and CI in STEM+C at the undergraduate, graduate, secondary school, and community levels. These programs will complement the other aspects of

this proposal by providing more avenues for experiential and service learning [Krusche et al., 2017; Derbinsky and Suresh, 2017]. The first program complements new and existing high school after school programs by providing organizers with the ability to demonstrate concepts more easily through the use of data science and CI. The second program emphasizes engagement by undergraduate students by providing the tools necessary to facilitate the integration of community service with applied data science projects. A brief overview of each community outreach program is provided in Appendix B.

The programs outlined above would increase net collective impact by complementing the existing arsenal of training and development opportunities. Additionally, they will equip trainees with the preliminary knowledge needed to be successful in the program complement as well as in the general application of advanced CI for applied data science.

2.6.3 Reproducibility

Perhaps the most fundamental concept in science is falsifiability, that is, the idea that results should always have a potential to be proven false. In order to facilitate this, the results and methods of a scientific investigation should be reproducible. One key challenge with regard to reproducibility in data science education is how to disseminate results in a way that is readable, reproducible, and easily shareable. While reproducibility has always been a critical aspect of scientific investigation, a shocking lack of reproducible results in the literature for various disciplines has surfaced in recent years [Aarts et al., 2015; Baker, 2016; Ioannidis, 2016; Aichner

et al., 2016; Camerer et al., 2016].

How do you make data science results and methods easy to share? One solution is via open notebook science and open notebook formats such as Jupyter [Ragan-Kelley et al., 2014]. Open notebook science has grown in popularity since the *Nature* interview by Sanderson [2008] and the recent open science movement. With open notebooks, students can easily modify and extend starter code, execute that code, and incorporate notes to produce "computational narratives" that, if shared, can be reproduced by others with the same notebook software. In the case of Jupyter notebooks, this is all done via a web interface with an option for users to download their notebooks locally if needed. Additionally, a local Jupyter installation can be used when the user does not have access to the instructional server.

These ideas can be easily extended to data science investigations using big data frameworks. With the current open access trend, research articles, while traditionally published online in PDF format, are readily available in other formats such as HTML. These articles could also be made available in open notebook formats. At the the very least, research data and source code to replicate the investigation should be included with research published scientific articles. As discussed in Section 2.3, a growing number of scientific journal publishers now support the open data initiatives by the National Science Foundation and the Open Science Framework organization. These initiatives require authors to provide their research data and source code in an effort to promote openness and reproducibility. The details on how to support the ScalaTion big data framework in Jupyter notebooks

is provided in Chapter 4. Example notebooks are provided in Chapter 6.

Chapter 3

ScalaTion

Michael E. Cotterell [1], Hao Peng [1], Nicholas Klepp [1], and John A. Miller [1]. "ScalaTion". 2017. [in preparation]

[1]Department of Computer Science, University of Georgia, Athens, GA, USA

3.1 Abstract

ScalaTion is a big data framework, written in Scala, that provides analytics techniques for prediction, classification, clustering, dimensionality reduction, functional data analysis, and simulation facilities for discrete-event simulation modeling. Major packages include support for serial and parallel execution of algorithms for linear algebra, analytics, simulation, and optimization. The software is free and open source under an MIT License.

3.2 Motivation and Significance

ScalaTion is motivated by the need to make big data analytics and simulation modeling more approachable to a cross-section of users, including data scientists and business analysts. Originally developed with expertise learned from the JSIM project [Miller et al., 1997], ScalaTion is constantly evolving via applied research in computer science and its use as an instructional tool for data science. To the developers of ScalaTion, approachability means making it easier for users to do the following: i) express and execute models via concise, readable, and well-documented source code; ii) relate models to domain knowledge via domain-specific language [Miller et al., 2010; Cotterell et al., 2011a]; and iii) combine modeling techniques along a continuum from analytics to simulation modeling [Miller et al., 2013a].

3.3 Software Description

3.3.1 Software Architecture

The module and top-level package structure for ScalaTion is presented in Figure 3.1. Section 3.3.2 describes the functionality provided by each module.

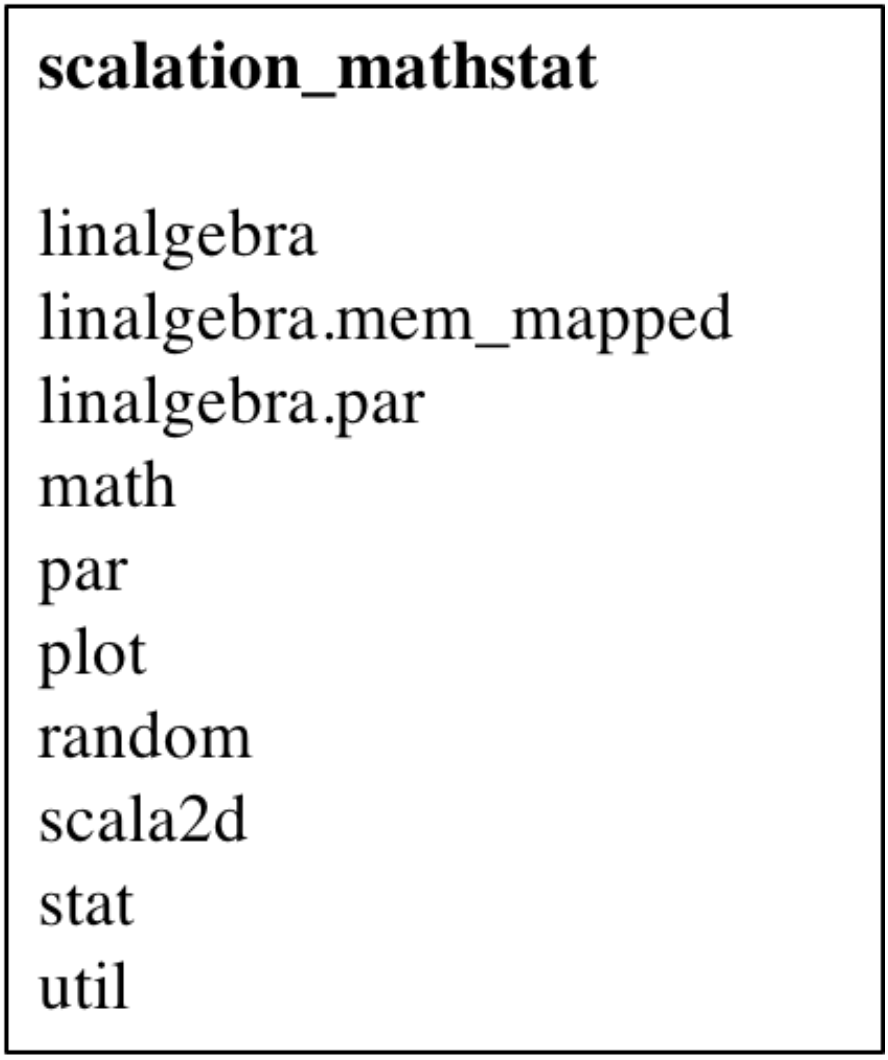


Figure 3.1: ScalaTion Module and Package Overview

3.3.2 Software Functionalities

As seen in Figure 3.1, ScalaTion consists of four modules (soon five), each containing a collection of related packages and sub-packages written in Scala. Overall functionality includes many techniques for prediction, classification, clustering, dimensionality reduction, functional data analysis, and simulation modeling. Major packages include support for serial and parallel execution of implemented algorithms. A description of each module, including major functionalities, follows: i) The `scalation_mathstat` module provides comprehensive mathematical and statistical capabilities, including support linear algebra operations, random number generation, and plotting. ii) The `scalation_database` module provides graph analytics and two NoSQL main memory databases: a graph database system and a columnar database system. iii) The `scalation_modeling` module provides support for the creation and analysis of analytics, optimization, and simulation models. iv) The soon to be added `scalation_automod` module will provide basic support for various automated modeling techniques. v) The `scalation_models` module provides a collection of sample models and applications facilitated by the other modules. A complete description of the packages in each module is presented in the developer documentation.

3.4 Illustrative Examples

3.4.1 Example 1: Multiple Linear Regression

In this example, the `AutoMPG_Regression` object performs multiple linear regression on the AutoMPG dataset[1] from the UCI Machine Learning Repository [Lichman, 2013]. The source code for this example is available in the `apps.analytics` package in `AutoMPG_Regression.scala`[2]. Example output is provided in Figure 3.2.

3.4.2 Example 2: Simple Biochemical Reaction Model in ScalaTion

In this example, the `Reaction` object models a simple biochemical reaction according some ordinary differential equations (ODEs). A glycan will pick up a new glycan residue to form another glycan. The reaction will be catalyzed by a protein enzyme. The source code for this example is available in the `apps.activity` package in `Reaction.scala`[3]. Example output is given in Figure 3.3.

[1] `http://archive.ics.uci.edu/ml/datasets/Auto+MPG`

[2] `http://cobweb.cs.uga.edu/~jam/scalation_1.3/scalation_models/src/main/scala/apps/analytics/AutoMPG_Regression.scala`

[3] `http://www.cs.uga.edu/~jam/scalation_1.4/scalation_models/src/main/scala/apps/activity/Reaction.scala`

Figure 3.2: Output of a Multiple Linear Regression Model in ScalaTion

In addition to individual parameter estimates, other model diagnostics are provided, including sum of squared error/residuals (`SSE`), standard error, coefficient of determination (`R-Squared`), etc.

```
> run-main apps.analytics.AutoMPG_Regression
[info] Running apps.analytics.AutoMPG_Regression
[info] model: y = b0 + b1*x1 + b2*x2 + b3*x3 + b4*x4 + b5*x5
                    + b6*x6 + b7*x7
[info] b = VectorD(   -17.2184, -0.493376, 0.0198956, -0.0169511,
                   -0.00647404, 0.0805758,  0.750773,    1.42614)
[info] Coefficients:
[info]        | Estimate   |   StdErr   |  t value |  Pr(>|t|)
[info]      x0 | -17.218435 |   4.644294 |  -3.7074 |   0.00021
[info]      x1 |  -0.493376 |   0.323282 |  -1.5261 |   0.12697
[info]      x2 |   0.019896 |   0.007515 |   2.6474 |   0.00811
[info]      x3 |  -0.016951 |   0.013787 |  -1.2295 |   0.21888
[info]      x4 |  -0.006474 |   0.000652 |  -9.9288 |   0.00000
[info]      x5 |   0.080576 |   0.098845 |   0.8152 |   0.41497
[info]      x6 |   0.750773 |   0.050973 |  14.7288 |       NaN
[info]      x7 |   1.426140 |   0.278136 |   5.1275 |   0.00000
[info]
[info] SSE:             4252.2125
[info] Residual stdErr: 3.3277 on 7 degrees of freedom
[info] R-Squared:       0.8215, Adjusted rSquared:  0.8182
[info] F-Statistic:     252.4280 on 7 and 384 DF
[info] AIC:             950.5017
[info] BIC:             982.2718
```

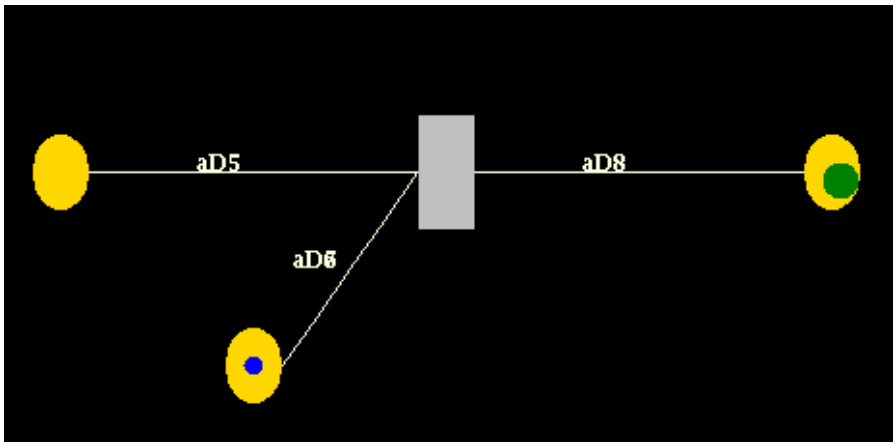
3.4.3 More Examples

More source code and executable examples are available in the `apps`[4] package in the `scalation_models` module.

[4] `http://www.cs.uga.edu/~jam/scalation_1.4/scalation_models/src/main/scala/apps/` ↗

Figure 3.3: Output of a Model for a Simple Biochemical Reaction

Each transition links to all of the incoming/outgoing places via true/false arcs. Additionally, the model establishes a back link to the containing Petri net.



3.5 Functional Data Analysis in ScalaTion

3.5.1 Introduction

Functional Data Analysis (FDA) is a rapidly growing area of analytics that aims to treat data as continuous functions instead of discrete, sampled observations in order to facilitate more elegant modeling. The appeal of this approach is captured in four main ideas [Ramsay and Dalzell, 1991; Ramsay and Silverman, 2005]: n

- **Frequency of Data:** With the advent of Big Data [Nature, 2008], data is being collected at a faster rate and in larger quantities than ever before. Instead of treating data as discrete observations, the rate of data acquisition enables analysts to collect enough data to realistically reproduce the

underlying process as a function even with only a finite number of actual observations available.

- **Modeling Simplicity:** Modeling can become simpler when the data is treated as functions. For example, suppose the data being sampled is known to correspond to some understood process. FDA allows the analyst to more elegantly aggregate and compose the data by taking advantage of its functional form.

- **Functional Analysis:** By treating data as functions, analysts can take advantage of existing methods from the field of Functional Analysis [Willem, 2013; Muscat, 2014], a branch of mathematics concerned with infinite-dimensional vector spaces and mappings between them. In particular, it facilitates operations on Hilbert spaces.

- **Structural Inference:** Analysts can use the structure of the functions in order to make inferences about the underlying process that produced the functions [Ramsay and Dalzell, 1991], e.g., via a clustering analysis [Tarpey and Kinateder, 2003]. This is incredibly applicable when the functions correspond to differential equations describing part of a process. Such equations can be combined with other equations to make understanding the process potentially easier.

In all, it enables analysts to obtain a more wholistic model that embodies the various relationships among the data. The seminal work in FDA is [Ramsay and Silverman, 1997], which has been updated to a second edition in [Ramsay and

Silverman, 2005].

Related Work

Below is a list of other available software tools and packages that are explicitly advertised as supporting FDA.

- **R 'fda' Package** [Ramsay et al., 2015]: This package and its included examples are provided by the authors of [Ramsay and Silverman, 2005]. In addition to the package manual, and introduction to the package is provided in [Ramsay et al., 2009]. MATLAB and S-PLUS versions of this package are available from http://www.psych.mcgill.ca/misc/fda/software.html.

- **R 'refund' Package** [Goldsmith et al., 2016]: This package includes methods for regression for functional data, including function-on-scalar, scalar-on-function, and function-on-function regression models. Some of the functions are applicable to image data.

- **R 'fdasrvf' Package** [Tucker, 2017]: This package performs alignment, PCA, and modeling of multidimensional and unidimensional functions using the square-root velocity framework [Srivastava et al., 2011] and [Tucker et al., 2013]. This framework allows for elastic analysis of functional data through phase and amplitude separation.

- **R 'fda.usc' Package** [Bande et al., 2016]: This package contains routines for exploratory and descriptive analysis of functional data such as depth

measurements, atypical curves detection, regression models, supervised classification, unsupervised classification and functional analysis of variance.

- **R 'funData' Package** [Happ, 2016]: This package provides classes for univariate and multivariate functional and image data and utility functions.

- **R 'fds' Package** [Shang and Hyndman, 2013]: This package contains a list of functional time series, sliced functional time series, and functional data sets.

- **R 'rainbow' Package** [Shang and Hyndman, 2016]: This package functions and data sets for functional data display and outlier detection.

- **R 'roahd' Package** [Tarabelloni et al., 2017]: This package a collection of methods for the robust analysis of univariate and multivariate functional data, possibly in high-dimensional cases, and hence with attention to computational efficiency and simplicity of use.

- **R 'FDboost' Package** [Brockhaus et al., 2016]: This package contains support for functional regression models (e.g., scalar-on-function, function-on-scalar and function-on-function regression models) that can be fitted by a component-wise gradient boosting algorithm.

- **R 'fdapace' Package** [Dai et al., 2017]: This package provides implementations of various methods for performing FDA, FPCA, and Empirical Dynamics. Its core contribution is an implementation of the Principal Analysis by

Conditional Estimation (PACE) algorithm for performing FPCA on sparsely or densely sampled random trajectories and time courses.

- **Python 'fdasrsf' Package** [Tucker, 2013]: This package is designed to implement the square root slope framework, a framework for separating the phase and the amplitude variability in functional data [Tucker et al., 2013], as well as functional principal component analysis (fPCA) [Srivastava et al., 2011]. The latest version of this package is available from https://pypi.python.org/pypi/fdasrsf/1.0.1.

ScalaTion

ScalaTion is a Scala-based library that serves a testbed for exploring a modeling continuum that includes Analytics, Simulation and Optimization. Recently, support has been added for facilitating smoothing, functional regression, and functional clustering. The modeling techniques supported include the following:

- Smoothing Spline Modeling
- Scalar-on-Function Regression Modeling

With respect to the estimation of Smoothing Spline models, ScalaTion supports various factorization techniques, including Cholesky and LU factorization, in addition to the standard inverse when using the least squares solution. Currently, the package supports the following basis functions for smoothing:

- B-Spline

- Polynomial
- Fourier
- Radial

Since the use of B-Spline basis functions is popular in FDA, we have provided a running time analysis and comparison with other packages in Section 3.5.2. Using some of the facilities described above, ScalaTion will soon support the following additional FDA techniques:

- Functional Clustering
- Functional Time Series

3.5.2 B-Spline Running Time Analysis and Evaluation

In this section, we describe two different ways to create the design matrix Φ and second derivative matrix $D^2\Phi$ for the estimation of a smoothing spline model. Specifically, given an order k (degree $k-1$) and non-decreasing input vector $\mathbf{t} = [t_0, t_1, \ldots, t_{n-1}]$ of length n, we construct the matrix $\Phi = \Phi(\mathbf{t})$. Each element of the matrix is computed using $\Phi_{i,j} = \phi_j(t_i)$ where $\phi_j(t)$ is an order-k B-Spline basis function parameterized with knot vector $(t_0^{(k-1)}, \mathbf{t}, t_{n-1}^{(k-1)})$ (i.e., a "clamped" version of $\mathbf{t}$). In a similar fashion, we construct the matrix $D^2\Phi = D^2\Phi(\mathbf{t})$ where $D^2\Phi(\mathbf{t})$ denotes taking the second derivative of each $\phi(t)$ with respect to t and evaluating it at each point provided in $\mathbf{t}$.

We constructed Φ and $D^2\Phi$ using four different methods: i) the `bsplineS` function provided by R's `fda` package; ii) the `Bspline.collmat` function provided

by Python's `bspline` module; ii) a recursive implementation written in Scala using ScalaTion; and iv) a dynamic programming implementation also written in Scala using ScalaTion.

A quick glance at R's `bsplineS` function in the `fda` package [Ramsay et al., 2015] shows that it is calling an S-PLUS function called `spline.des`, which calls some C programming language code. The source code is licensed under the GNU General Public License and contains little inline documentation describing the method being used. A cursory glance suggests that they may be using an implementation that is similar to ScalaTion's dynamic programming approach (explained later in this section), however, this is not clear from observations of their implementation's running time.

According to John T. Foster and Juha Jeronen, the author's of Python's `Bspline.collmat` function, their implementation is a memoized version of the recursive definition for B-Spline basis functions. The source code is licensed under an MIT license and available on GitHub[5].

The recursive implementation provided by ScalaTion simply follows the recursive definition for the j-th order k B-spline basis function $\phi_j(t)$ as described by [Patrikalakis and Maekawa, 2010]:

$$\phi_{j,k}(t) = \frac{t - \tau_j}{\tau_{j+k-1} - \tau_j}\phi_{j,k-1}(t) + \frac{\tau_{j+k} - t}{\tau_{j+k} - \tau_{j+1}}\phi_{j+1,k-1}(t) \tag{3.1}$$

where

[5] `https://github.com/johntfoster/bspline`

$$\phi_{j,1}(t) = \begin{cases} 1 & \tau_j \leq t < \tau_{j+1} \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

Similarly, the first derivative $D\phi_{j,m}(t)$ can be defined as [Patrikalakis and Maekawa, 2010; Piegl and Tiller, 1997]:

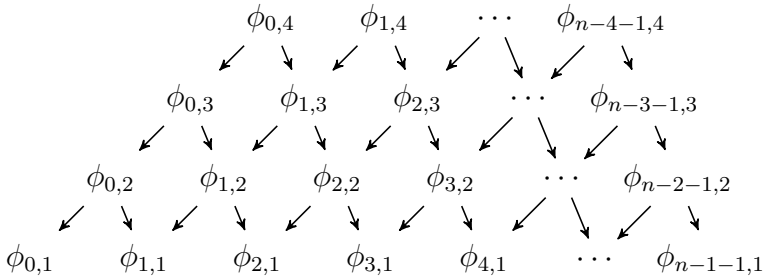
$$D\phi_{j,k}(t) = \frac{k-1}{\tau_{j+k-1} - \tau_j}\phi_{j,k-1}(t) - \frac{k-1}{\tau_{j+k} - \tau_{j+1}}\phi_{j+1,k-1}(t). \tag{3.3}$$

Other derivatives $D^n\phi_{j,m}(t)$ can be computed similarly by further differencing the coefficients:

$$D^n\phi_{j,k}(t) = \frac{k-1}{\tau_{j+k-1} - \tau_j}D^{n-1}\phi_{j,k-1}(t) - \frac{k-1}{\tau_{j+k} - \tau_{j+1}}D^{n-1}\phi_{j+1,k-1}(t). \tag{3.4}$$

A direct implementation of the recursive function is highly inefficient. As is seen in the recursion tree presented in Figure 3.4, there are many overlapping sub-problems that are re-evaluated in order to perform the desired evaluation. To

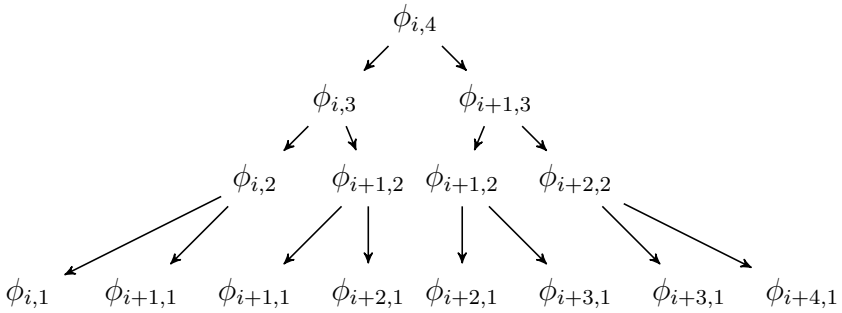
Figure 3.4: B-Spline Basis Function Overlapping Problem Decomposition



analyze the complexity of this approach, let's examine Figure 3.5, which shows the recursion tree with for a single order 4 B-spline basis function evaluation with its

sub-problems displayed disjoint (i.e., we explicitly show repeated sub-problems). Let $T(k)$ denote the number of floating point operations needed to evaluate a B-

Figure 3.5: B-Spline Basis Function Disjoint Problem Decomposition



spline basis function of order k. Clearly, by the recursive definition of $\phi_{j,k}(t)$, we have the following recurrence relation for $T(n)$:

$$T(k) = 2T(k-1) + \Theta(1) = \Theta(1) \cdot \sum_{i=1}^{k} 2^{i-1} = \Theta(2^k + 1) = \Theta(2^k) \tag{3.5}$$

Therefore, the number of floating point operations for evaluating all of the $(n - k + 1)$-many basis functions for a particular input point is:

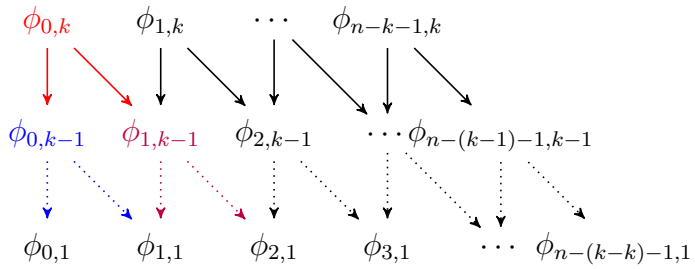
$$\begin{aligned} (n-k-1) \cdot \Theta(2^k) &= \Theta(n2^k - k2^k - 2^k) \\ &= \mathrm{O}(n2^k), \text{ assuming } k \leq n. \end{aligned} \tag{3.6}$$

As $\mathbf{t}$ often, and in our case precisely, contains $\Theta(n)$ input points, the construction of Φ where $\Phi_{i,j} = \phi_j(t_i)$ requires $\Theta(n) \cdot \mathrm{O}(n2^k) = \mathrm{O}(n^2 2^k)$ floating point operations using this recursive formulation.

A better implementation involves dynamic programming. Consider the prob-

lem decomposition described in Figure 3.6, where all order k B-spline basis functions are evaluated at an input point t [de Boor et al., 2001].. Here, a bottom-up

Figure 3.6: B-Spline Basis Function Optimal Substructure Decomposition



approach is observed where each level represents the evaluations of the basis functions for a particular order, starting at order 1. Evaluating the basis functions for a particular level/order depends only on the optimal solution for the level/order below. Assuming the first level/order is evaluated correctly, this constitutes an optimal substructure and satisfies the principle of optimality [Bellman, 1952]. This dynamic programming approach requires k levels to evaluate all B-spline basis functions of order k with $n-(k-i)-1$ evaluations at level $1 \leq i \leq k$. The total number of floating point operations for evaluating a particular input point is:

$$\begin{aligned}
\Theta(1)\sum_{i=1}^{k}(n-(k-i)-1) &= \Theta(1)\sum_{i=1}^{k}(n-k+i-1) \\
&= \Theta(1)\left[\sum_{i=1}^{k}n-\sum_{i=1}^{k}k+\sum_{i=1}^{k}i-\sum_{i=1}^{k}1\right] \\
&= \Theta(1)(kn-k^2+\frac{k(k+1)}{2}-k) \\
&= \mathrm{O}(nk), \text{ assuming } k \leq n.
\end{aligned} \tag{3.7}$$

As $\mathbf{t}$ often, and in our case precisely, contains $\Theta(n)$ input points, the construction of Φ where $\Phi_{i,j} = \phi_j(t_i)$ requires $\Theta(n) \cdot \mathrm{O}(nk) = \mathrm{O}(n^2 k)$ floating point operations using this dynamic programming formulation.

Both the recursive and dynamic programming formulations were implemented in ScalaTion. We observed their running times, along with R's `bsplineS` function (from the `fda` package) and Python's `Bspline.collmat` function (from the `bspline` module), for different sized input vectors over 100 replications. These observations were conducted on a single node in the University of Georgia's "sapelo" cluster, running a 64-bit CentOS 6.5 distribution of the Linux operation system with a 48 core AMD Opteron processor and 128GB of memory. Special care was taken to minimize and or exclude the impact of garbage collection on timings by observing each running each method separately for each knot vector configuration. The average evaluation times for $\Phi = \Phi(\mathbf{t})$ are provided in Figures 3.7 and 3.9 for orders 3 and 4 and Figures and Figures 3.8 and 3.10 for orders 5 and 6. The average evaluation times for $D^2\Phi = D^2\Phi(\mathbf{t})$ are provided in Figures 3.11 and 3.13 for orders 3 and 4 and Figures 3.12 and 3.14 for orders 5 and 6. As expected, the recursive implementations are inefficient compared to the others. Across all observations, ScalaTion's dynamic programming implementation performs the best.

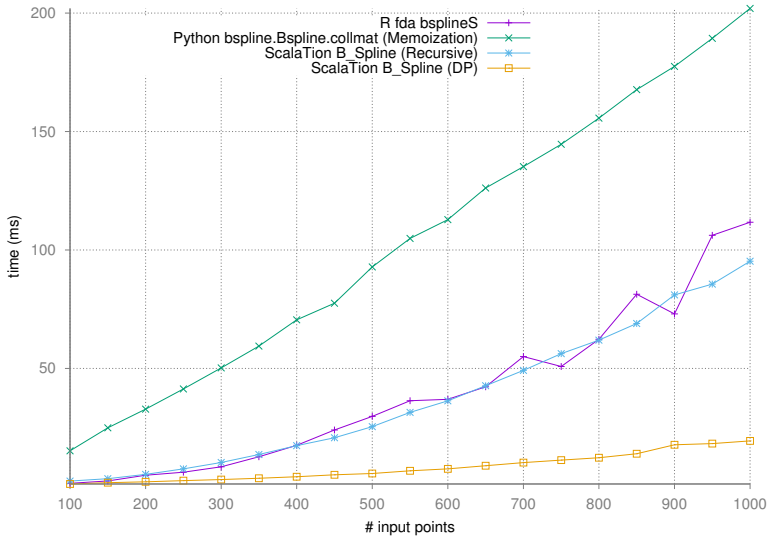
Acknowledgments

This study was supported in part by resources and technical expertise from the Georgia Advanced Computing Resource Center, a partnership between the University of Georgia's Office of the Vice President for Research and Office of the Vice

Figure 3.7: B-Spline Basis Evaluation Times for $\Phi(\mathbf{t})$ Orders 3 & 4

Average evaluation times in milliseconds, over 100 replications, for $\Phi(\mathbf{t})$ across different lengths of a uniformly-spaced, non-decreasing input vector $\mathbf{t}$, assuming the underlying knot vector is a clamped version of $\mathbf{t}$.

(a) Order 3



(b) Order 4

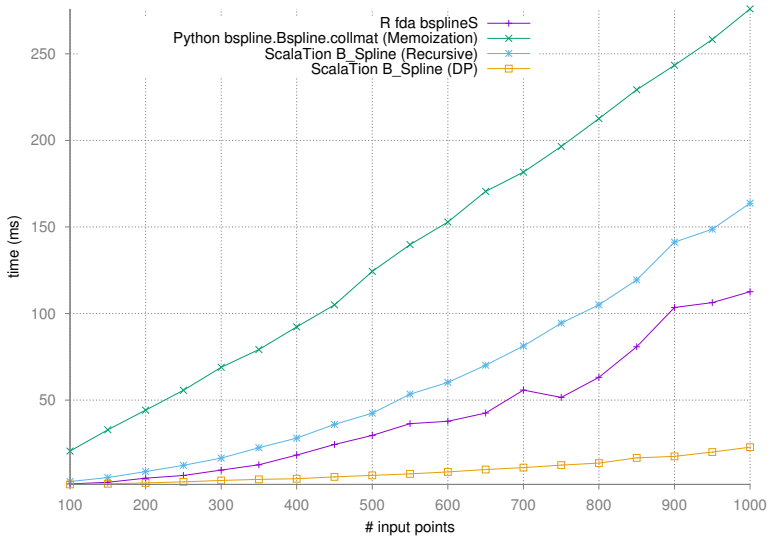
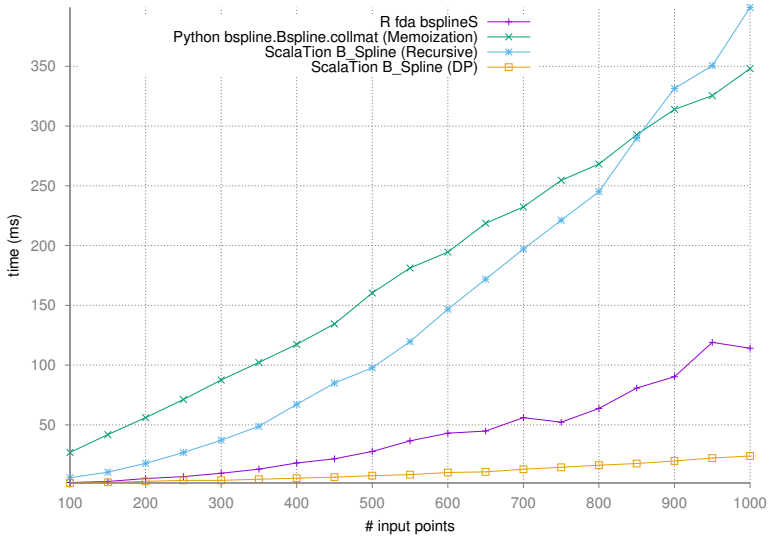


Figure 3.8: B-Spline Basis Evaluation Times for $\Phi(\mathbf{t})$ Orders 5 & 6

Average evaluation times in milliseconds, over 100 replications, for $\Phi(\mathbf{t})$ across different lengths of a uniformly-spaced, non-decreasing input vector $\mathbf{t}$, assuming the underlying knot vector is a clamped version of $\mathbf{t}$.

(a) Order 5



(b) Order 6

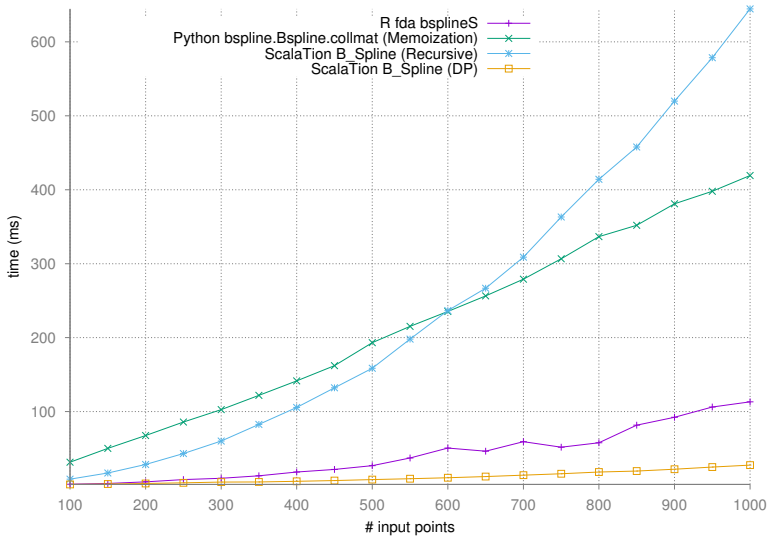
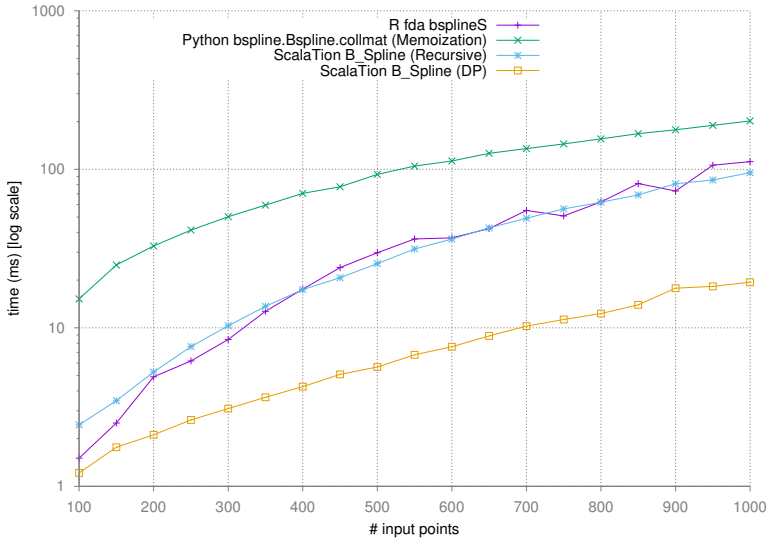


Figure 3.9: B-Spline Basis Evaluation Times (Log Scale) for $\Phi(\mathbf{t})$ Orders 3 & 4

Average evaluation times in milliseconds (presented in $\log_{10}$-scale), over 100 replications, for $\Phi(\mathbf{t})$ across different lengths of a uniformly-spaced, non-decreasing input vector $\mathbf{t}$, assuming the underlying knot vector is a clamped version of $\mathbf{t}$.

(a) Order 3



(b) Order 4

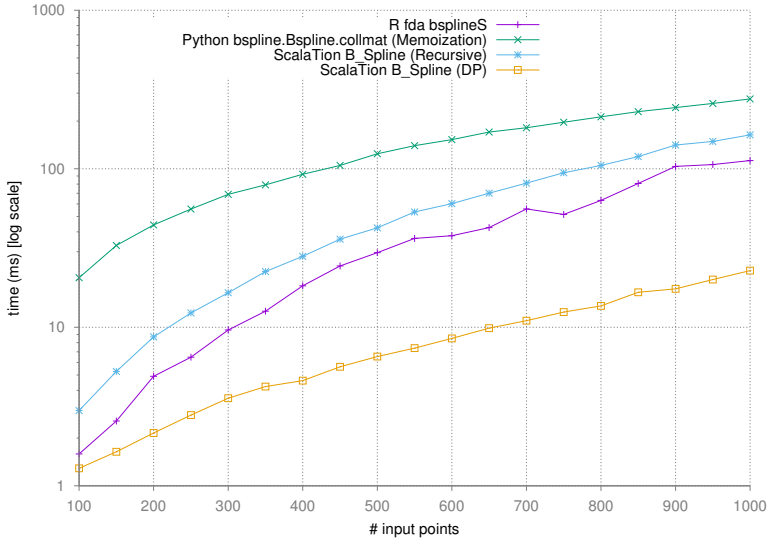
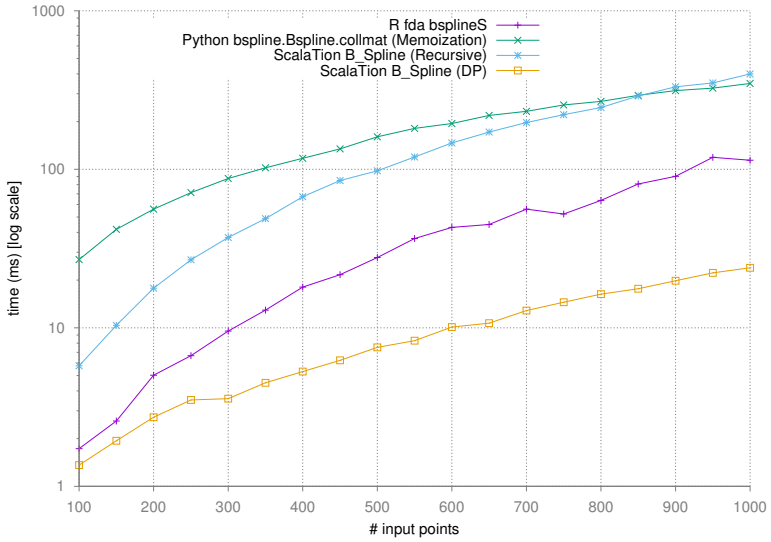


Figure 3.10: B-Spline Basis Evaluation Times (Log Scale) for $\Phi(\mathbf{t})$ Orders 5 & 6

Average evaluation times in milliseconds (presented in $\log_{10}$-scale), over 100 replications, for $\Phi(\mathbf{t})$ across different lengths of a uniformly-spaced, non-decreasing input vector $\mathbf{t}$, assuming the underlying knot vector is a clamped version of $\mathbf{t}$.

(a) Order 5



(b) Order 6

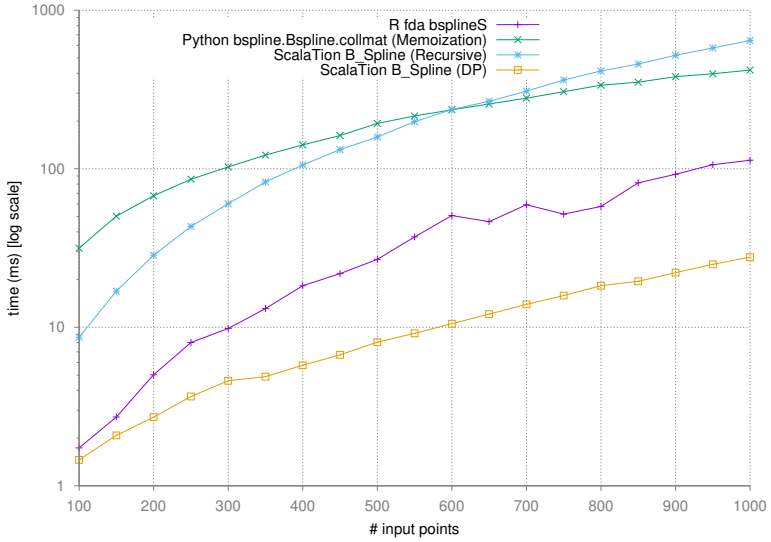
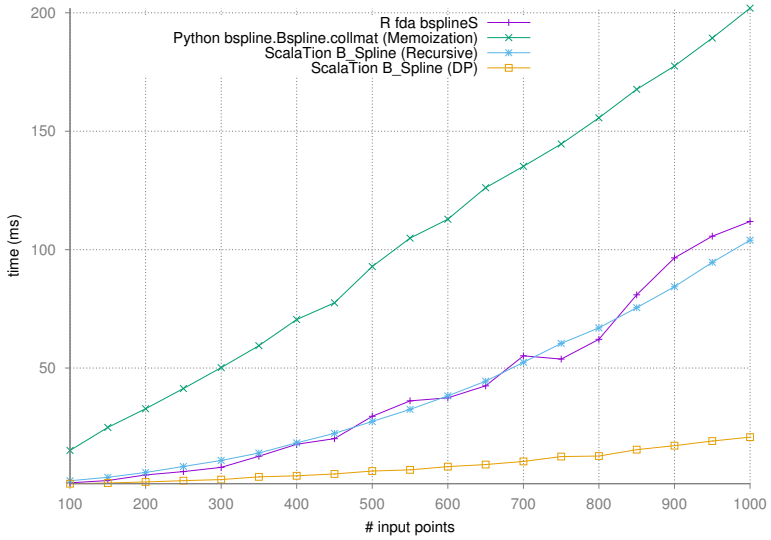


Figure 3.11: B-Spline Basis Evaluation Times for $D^2_{\mathbf{t}}\Phi(\mathbf{t})$ Orders 3 & 4

Average evaluation times in milliseconds, over 100 replications, for $D^2_{\mathbf{t}}\Phi(\mathbf{t})$ across different lengths of a uniformly-spaced, non-decreasing input vector $\mathbf{t}$, assuming the underlying knot vector is a clamped version of $\mathbf{t}$.

(a) Order 3



(b) Order 4

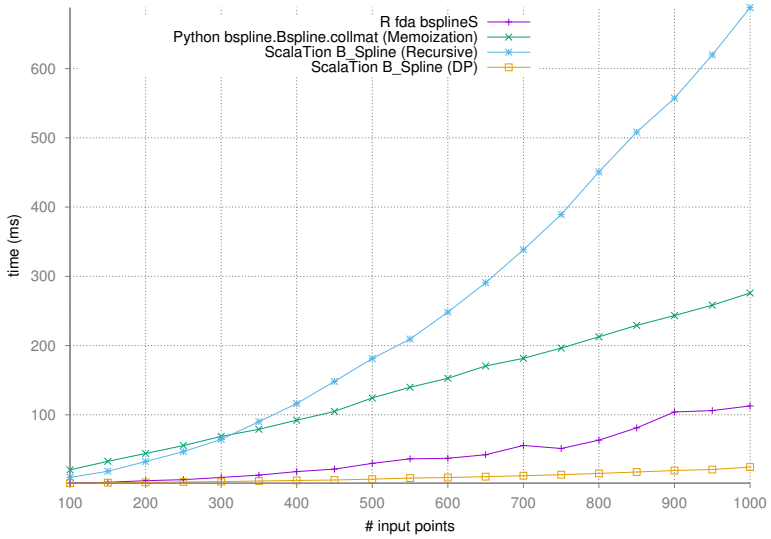
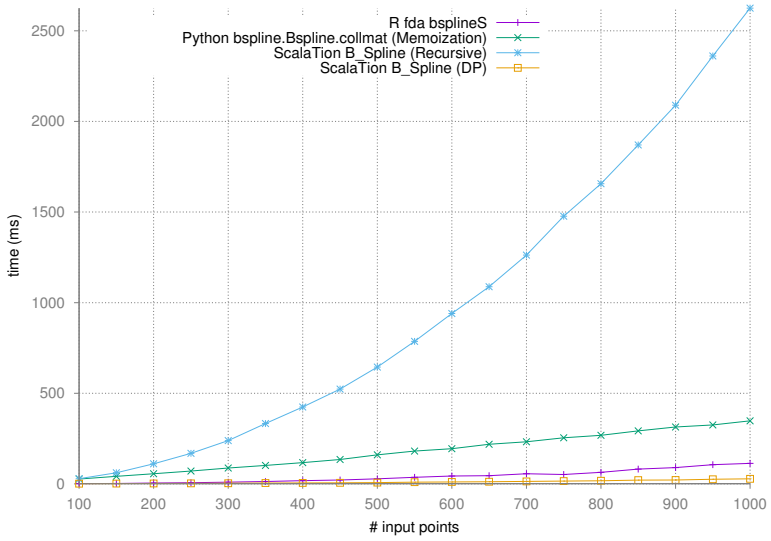


Figure 3.12: B-Spline Basis Evaluation Times for $D^2_{\mathbf{t}}\Phi(\mathbf{t})$ Orders 5 & 6

Average evaluation times in milliseconds, over 100 replications, for $D^2_{\mathbf{t}}\Phi(\mathbf{t})$ across different lengths of a uniformly-spaced, non-decreasing input vector $\mathbf{t}$, assuming the underlying knot vector is a clamped version of $\mathbf{t}$.

(a) Order 5



(b) Order 6

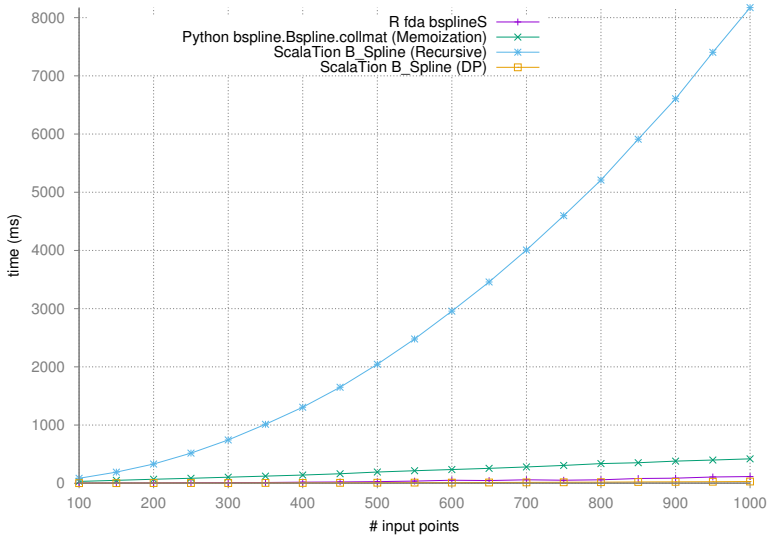
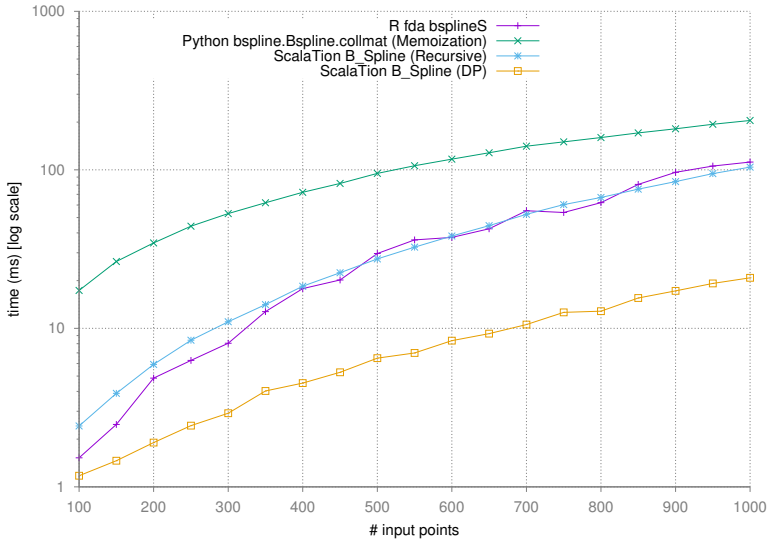


Figure 3.13: B-Spline Basis Evaluation Times (Log Scale) for $D^2_{\mathbf{t}}\Phi(\mathbf{t})$ Orders 3 & 4

Average evaluation times in milliseconds (presented in $\log_{10}$-scale), over 100 replications, for $D^2_{\mathbf{t}}\Phi(\mathbf{t})$ across different lengths of a uniformly-spaced, non-decreasing input vector $\mathbf{t}$, assuming the underlying knot vector is a clamped version of $\mathbf{t}$.

(a) Order 3



(b) Order 4

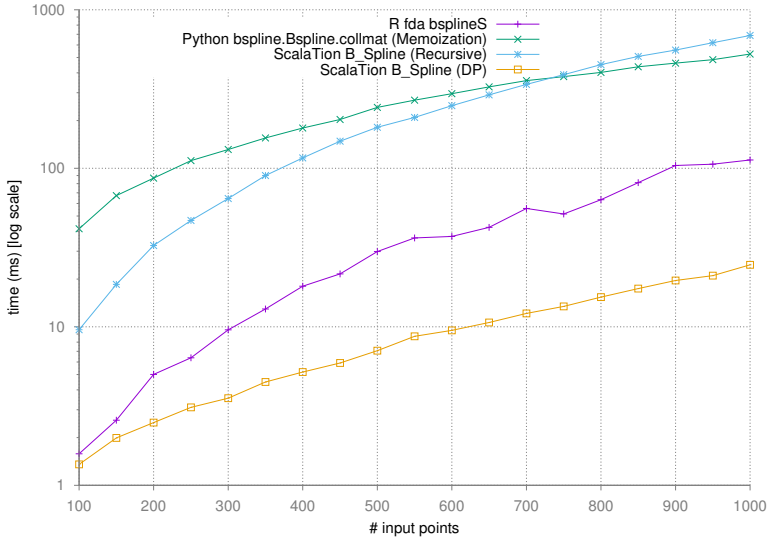
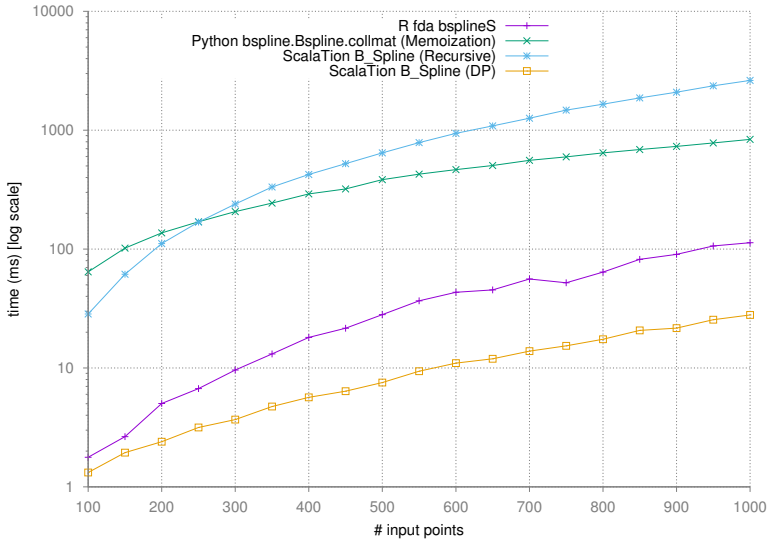


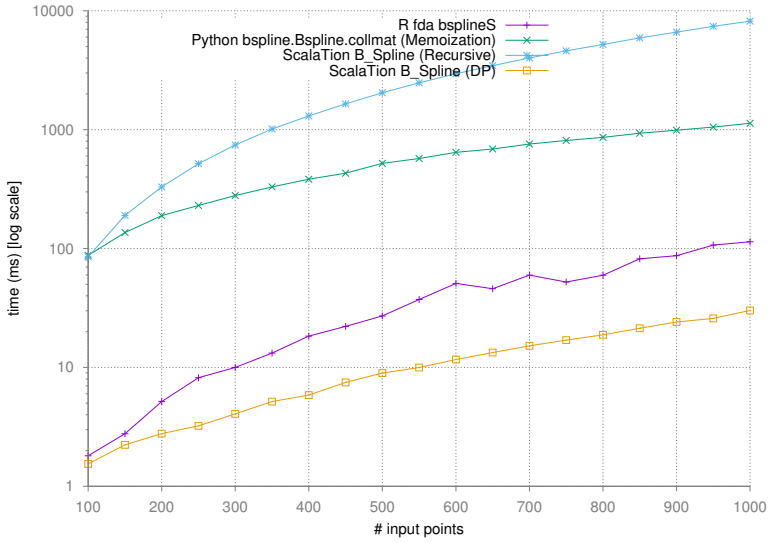
Figure 3.14: B-Spline Basis Evaluation Times (Log Scale) for $D^2_{\mathbf{t}}\Phi(\mathbf{t})$ Orders 5 & 6

Average evaluation times in milliseconds (presented in $\log_{10}$-scale), over 100 replications, for $D^2_{\mathbf{t}}\Phi(\mathbf{t})$ across different lengths of a uniformly-spaced, non-decreasing input vector $\mathbf{t}$, assuming the underlying knot vector is a clamped version of $\mathbf{t}$.

(a) Order 5



(b) Order 6



President for Information Technology.

3.6 Impact

ScalaTion represents an excellent tool for making big data analytics more approachable. It allows users to express, execute, and connect together models that are more concise and readable through well documented code and the use of domain-specific language. For example, each class in the framework is not only documented at the interface level with examples, it is also documented internally in order to provide insight into the implementation details. Implementations make use of domain-specific language, whenever possible, in order to promote domain recognition and validation by those familiar with the particular domain. A good example that illustrates both of these points can be seen in `Regression.scala`[6], where the source code and documentation for performing multiple linear regression resides. As seen in Figure 3.15, the `train` function, which is used to estimate a regression model's coefficient vector, provides illustrative documentation for both its function interface and implementation. Those familiar with matrix factorization in the domain of linear algebra will recognize the use of various factorization techniques for solving the regression's set of linear equations. Most source code files, including `Regression.scala`, also include test applications near the bottom of the file that provide illustrative usage examples for the classes and functions defined in the file.

[6] `http://www.cs.uga.edu/~jam/scalation_1.4/scalation_modeling/src/main/scala/scalation/analytics/Regression.scala`

Additionally, ScalaTion helps support previous and existing research in analytics. Some examples include: parallel, big data Bayesian Network classifiers with efficient cross-validation are implemented using ScalaTion in [Peng et al., 2017]; simulation models, modeled using ScalaTion, for traffic flow and travel times are discussed in [Bowman and Miller, 2016]; and support for semi-automated model selection using ontologies and meta-learning is presented in [Nural et al., 2015] and [Nural et al., 2017], respectively.

Furthermore, ScalaTion is also available for use in Jupyter [Ragan-Kelley et al., 2014] notebooks via the ScalaTion Kernel project presented in Chapter 4, making it appealing to data scientists and open science advocates due to Jupyter's presence in that space. As a result, users can create, share, and execute data science investigations performed using ScalaTion that are stored on Jupyter notebooks. Information on how to readily provide this support is described in Chapters 4 and 6.

3.7 Conclusions

In this paper, we presented ScalaTion, a big data framework that provides analytics techniques for prediction, classification, clustering, dimensionality reduction, functional data analysis, and simulation facilities for discrete-event simulation modeling. An overview of the framework's general functionality and architecture was presented, including illustrative examples. Furthermore, ScalaTion's impact in a variety of areas was also discussed.

Figure 3.15: Code Snippet from `Regression.scala`

This slightly modified (for space) code snippet from the `scalation.analytics` package's `Regression` class in ScalaTion's `modeling` module illustrates how the source code in ScalaTion is documented both at the interface level, with examples, and the implementation level. Furthermore, those familiar with matrix factorization in the domain of linear algebra will recognize the various factorization techniques used to estimate the model coefficients for a multiple linear regression, expressed concisely using domain-specific (i.e., linear-algebra-specific) language.

```
//::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
/** Train the predictor by fitting the parameter vector
 *  (b-vector) in the multiple regression equation
 *  <p>
 *      yy  =  b dot x + e
 *          =  [b_0, ... b_k] dot [1, x_1 , ... x_k] + e
 *  <p>
 *  using the ordinary least squares 'OLS' method.
 *  @param yy  the response vector
 */
def train (yy: VectoD)
{
    b = technique match {
        case QR       => fac.solve (yy)
        case Cholesky => fac.solve (x.t * yy)
        case SVD      => fac.solve (yy)
        case LU       => fac.solve (x.t * yy)
        case _        => fac.solve (x.t * yy)
    } // match
    e = yy - x * b       // compute residual/error vector e
    diagnose (yy)        // compute diagonostics
} // train
```

Chapter 4

ScalaTion Kernel: Towards Open Notebook Support

4.1 Introduction

In this chapter, we discuss how to provide light-weight big data framework support in open notebooks. Consider, for a moment, a data science investigation performed using a big data framework and published in a peer reviewed research article. Interested readers of the article may want to reproduce the authors' results. In the worst case scenario, the code, data, and infrastructure needed to reproduce the results are either generally unavailable or requires restricted access. In the best case scenario, everything is available from either the author's or publisher's website. However, facilitating the best case scenario is sometimes tedious and oftentimes impractical for investigators. To help mitigate this, we expand on

existing research that proposes the use of open notebooks for the dissemination of data science investigations.

As interest in the open science movement has grown in recent years, so has interest in interactive, open notebooks. [Shen, 2014] describes open notebooks as interactive lab notebooks for computational work. They combine notes and code in a format that permits sharing, modification, and easy execution. Users can run the code to generate and compare results. Different programming languages and frameworks are provided via "kernels", modules that usually must be installed and configured on the notebook platform being used.

The main contribution described in this chapter is providing support for the ScalaTion [Miller et al., 2010] big data framework in open notebooks running on Jupyter [Ragan-Kelley et al., 2014], an open source open notebook platform written in Python. Details of the ScalaTion framework are provided in Chapter 3. In this work, Scala language and ScalaTion framework support are provided via our ScalaTion Kernel project, a Jupyter kernel discussed later in this chapter.

Our integration approach is minimal in order to make it easier for investigators to adopt and adapt to their needs. While existing Jupyter kernels exist that provide Scala language support, most either require extensive configuration, are not maintained, or require configuration with the Spark[1] framework. A general list of available Jupyter kernels is provided here[2].

There is also existing research discussions on the use of open notebooks for research papers themselves. For example, in [Kluyver et al., 2016], the authors

[1] `https://spark.apache.org`

[2] `https://github.com/jupyter/jupyter/wiki/Jupyter-kernels`

discuss many of the use cases for open notebook platforms like Jupyter, including their use for academic papers. At the time their paper was written, they expressed this use case as an achievable goal pending solutions to certain roadblocks. The biggest problem they predicted is the integration of properly formatted academic citations into the notebooks. While this is not a problem addressed by our work, it is definitely an important aspect of open research articles that requires careful consideration. Also, in [Gil et al., 2016], the authors discuss the importance of computational provenance in research articles (i.e., the computational steps that were taken to achieve a result). They suggest the use of open notebooks to document and record the computational steps taken for part of an investigation or analysis as well as the use of standards such as W3C PROV [Missier et al., 2013] to facilitate integration of these records across different notebook platforms. While this is closely related to our work, we focus more on the lightweight integration of the ScalaTion big data framework.

The rest of this chapter is organized as follows: an overview of the ScalaTion Kernel project, including detailed installation instructions, is provided in Section 4.2; some examples are introduced in Section 4.3; impact is discussed in Section 4.4; and conclusions are given in Section 4.5.

4.2 ScalaTion Kernel for Jupyter

To help users incorporate ScalaTion into their Jupyter notebooks, we developed the lightweight ScalaTion Kernel. It is lightweight in the sense that it allows

users to harness ScalaTion in a Jupyter notebook with minimal dependencies. It uses the system or container's Scala installation for the underlying *read-evaluate-print-loop* (REPL), and it allows administrators to specify the local ScalaTion distribution to be used. The actual kernel code is written in Python 3 using the `pexpect` package, allowing it to interact with Scala's interactive REPL. Although the kernel is currently only written to interact with Scala's REPL, the authors anticipate that a similar approach could be taken with other REPLs (e.g., Java 9's JShell[3]) to provide big data framework support in other programming languages. A

The ScalaTion Kernel project includes free and open source code, and is available at `https://github.com/scalation/scalation_kernel` ↗ . General installation instructions for the development version of the kernel are available there. More detailed instructions are provided in the following subsection.

4.2.1 Installation Instructions

In this section, we describe three different ways to install and use ScalaTion Kernel: i) system-wide installation instructions are provided in Section 4.2.1; ii) a quick virtual environment installation option is described in Section 4.2.1; and iii) instructions on how to deploy an installation as a containerized application using Docker are provided in Section 4.2.2.

[3] `https://docs.oracle.com/javase/9/jshell/introduction-jshell.htm` ↗

Dependencies

Most of the installation instructions assume that the following dependencies are installed and available (i.e., they are on the executable path) on the system or container that will run the software:

- Java ≥ 8
- Python $\geq 3.6.3$
- Scala $\geq 2.12.4$

Older versions of these dependencies may work, but they are untested by the authors.

System-wide Installation

System-wide installation is currently available via the Python `pip` package. Privileged users can execute the following command to install ScalaTion Kernel from PyPi[4], the Python Package Index:

```
$ python3 -m pip install --upgrade pip
$ python3 -m pip install scalation_kernel
```

Then, assuming Jupyter is already installed, ScalaTion Kernel can be registered for use with Jupyter using the following command:

```
$ python3 -m scalation_kernel.install
```

[4] `https://pypi.python.org/pypi?:action=display&name=scalation-kernel`

Now, when users on the system launch the system-wide Jupyter installation, new and existing notebooks will have an option to use the ScalaTion Kernel, as seen in Figures 4.1 & 4.2.

Quick Setup

In order to facilitate rapid setup, a quick setup script is provided with the development version of ScalaTion Kernel that sets up an independent, virtual Jupyter installation with support for the kernel. In addition to the dependencies listed earlier, Git ($\geq$ 2.14.2) and the Python `virtualenv` package ($\geq$ 15.1.0) are required for these quick setup instructions. This script has been tested on MacOS, Linux, and Windows 10 (with Windows Subsystem for Linux) installations that satisfy the dependencies described in the previous section. Currently, only ScalaTion 1.4 is supported for quick setup. To download and run the quick setup script, the user can enter the following commands:

```
$ git clone https://github.com/scalation/scalation_kernel.git
$ cd scalation_kernel
$ bash quick_setup_1.4.sh
```

The first time the user executes this script, it may take some time as additional files are downloaded from the Internet. After the script executes correctly, a Jupyter installation will open in their default web browser with ScalaTion Kernel support. If the user's web browser does not open automatically, then the user should open the URL provided in the script output using the browser of their choosing to open the created Jupyter installation. If, at a later point in time, the user wishes to

reuse that same installation, then they need only run the quick setup script again. Now, when Jupyter is launched within the current virtual environment, new and existing notebooks will have an option to use the ScalaTion Kernel, as seen in Figures 4.1 & 4.2.

4.2.2 Docker Container

ScalaTion Kernel is also available for easy deployment as a containerized application using Docker[5]. The following instructions have been tested with the Docker Community Edition (CE)[6]. These instructions do not require any of the dependencies listed previously, since everything will be downloaded and setup inside of a container. However, users will still need to download the ScalaTion Kernel `Dockerfile`[7] in order to build and launch the container image. This method has been tested by users on MacOS, Linux, and Windows 10 (with Windows Subsystem for Linux) installations with Docker CE installed. Currently, only ScalaTion 1.4 is supported for quick setup. Assuming the user downloaded the `Dockerfile` and saved it in a directory called `scalation_kernel`, they can enter the following commands to build the container image:

```
$ cd /path/to/scalation_kernel
$ docker build -t scalation_kernel .
```

Building the container image for the first time may take some time as additional files are downloaded from the Internet. After the container image is built, the user

[5] `https://www.docker.com/`

[6] `https://store.docker.com/search?offering=community&type=edition`

[7] `https://raw.githubusercontent.com/scalation/scalation_kernel/master/docker/Dockerfile`

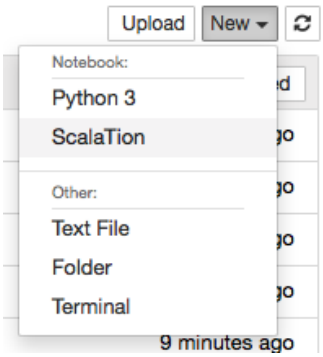
can run the image on the default Docker machine using the following command:

```
$ docker run -it --rm -p 8888:8888 scalation_kernel
```

Assuming everything went smoothly, the user should open the URL provided in the terminal output using the browser of their choosing to open the containerized Jupyter installation with ScalaTion Kernel support. Within the containerized application, new and existing notebooks will have an option to use the ScalaTion Kernel, as seen in Figures 4.1 & 4.2.

Figure 4.1: Screenshot of ScalaTion Kernel available in Jupyter Notebook

When the ScalaTion Kernel is installed, users can choose "ScalaTion" when creating a new notebook.

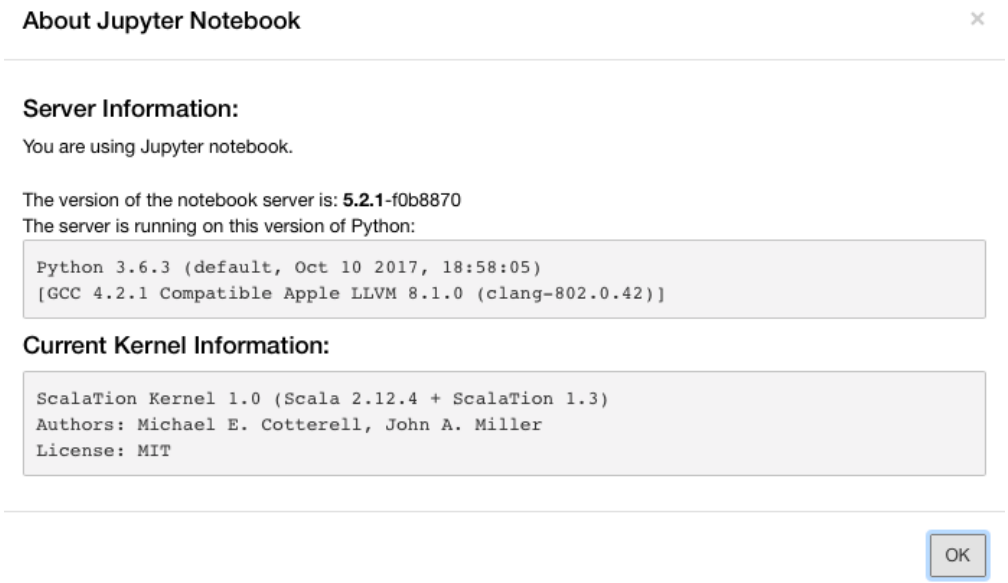


4.3 Usage and Example

An example that illustrates the exploration of the Longley macroeconomic dataset [Longley, 1967] using a multiple linear regression model is provided in Figures 4.3 & 4.4.

Figure 4.2: Screenshot of ScalaTion Kernel Info in Jupyter

When the ScalaTion Kernel is installed and being used by the current notebook, the kernel information is available on the "About" page.



This notebook is also available here[8]. A collection of more example notebooks is presented in Chapter 6. A user guide is available here[9] from the project's GitHub repository.

[8] https://github.com/scalation/scalation_kernel/blob/master/notebooks/regression.ipynb

[9] https://github.com/scalation/scalation_kernel/blob/master/docs/USER.md

4.4 Impact

ScalaTion Kernel represents an excellent complement to ScalaTion for making big data analytics more approachable. It allows users to express, execute, and share concise ScalaTion models using Jupyter notebooks. From a user perspective, the package is easy to deploy in a variety of ways, which should encourage adoption by both investigators wishing to disseminate their investigations and others wishing to reproduce the results of those investigations. Furthermore, it enables users to incorporate notes using Markdown[10] and LaTeX[11] syntax, supporting both rich formatted text and mathematical notations that complement the domain-specific language facilities provided by ScalaTion.

4.5 Conclusions

In this chapter, we presented ScalaTion Kernel, a Jupyter kernel that enables users to use Scala and ScalaTion in Jupyter notebooks. Installation instructions for system-wide installations, virtual environment installations, and Docker containers were provided. Detailed notebook examples are provided in Chapter 6.

Future work for this project includes providing installation support for other popular package managers such as Anaconda[12]. Additionally, the authors would like to provide basic plotting support for ScalaTion data types like vectors, matrices, relations, etc.

[10] `https://daringfireball.net/projects/markdown/syntax`
[11] `https://www.latex-project.org`
[12] `https://anaconda.org/`

Figure 4.3: ScalaTion Kernel Regression Example before Run

Here we see an example of a Jupyter notebook in which students use the ScalaTion Kernel to explore a dataset using a multiple linear regression model. This shows the notebook before the code cells are executed using the kernel.

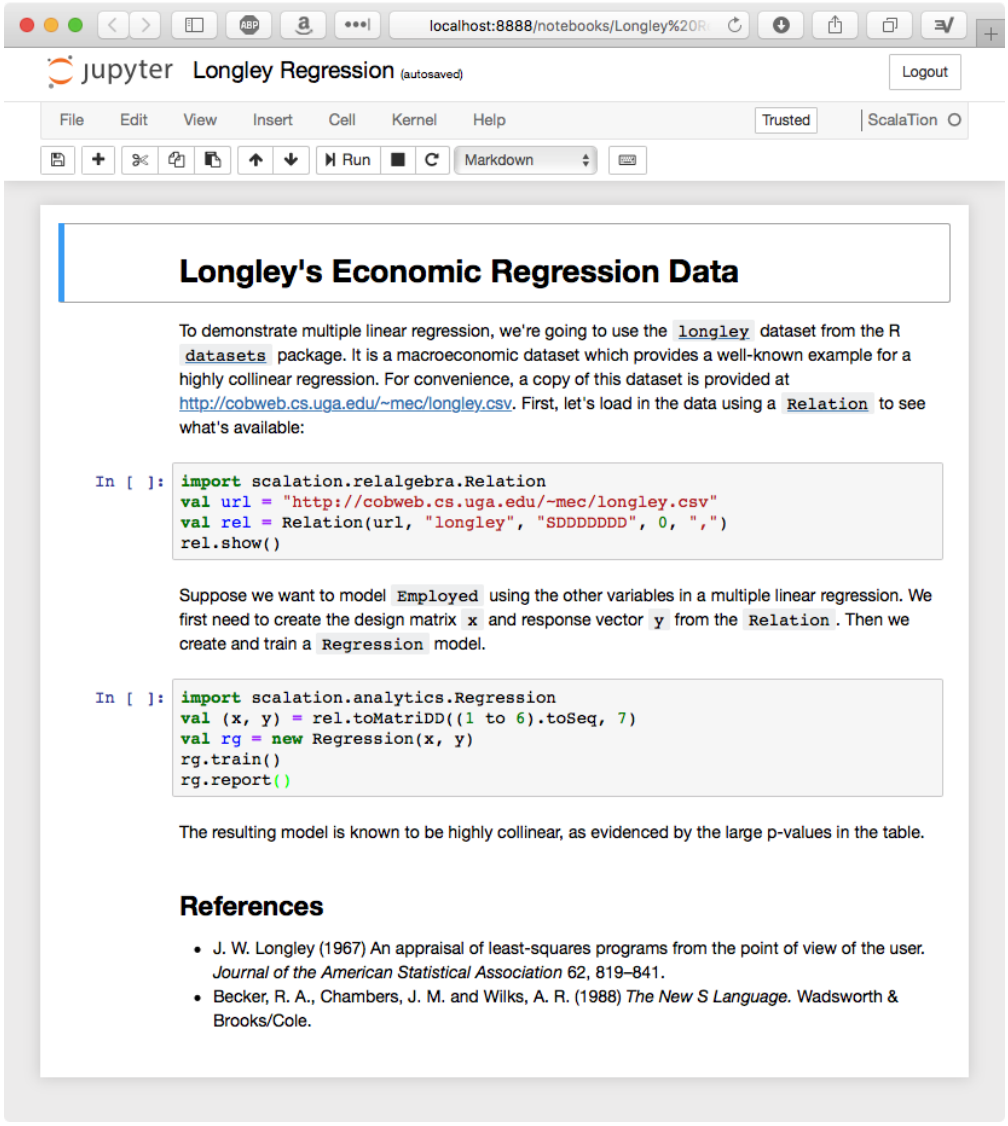
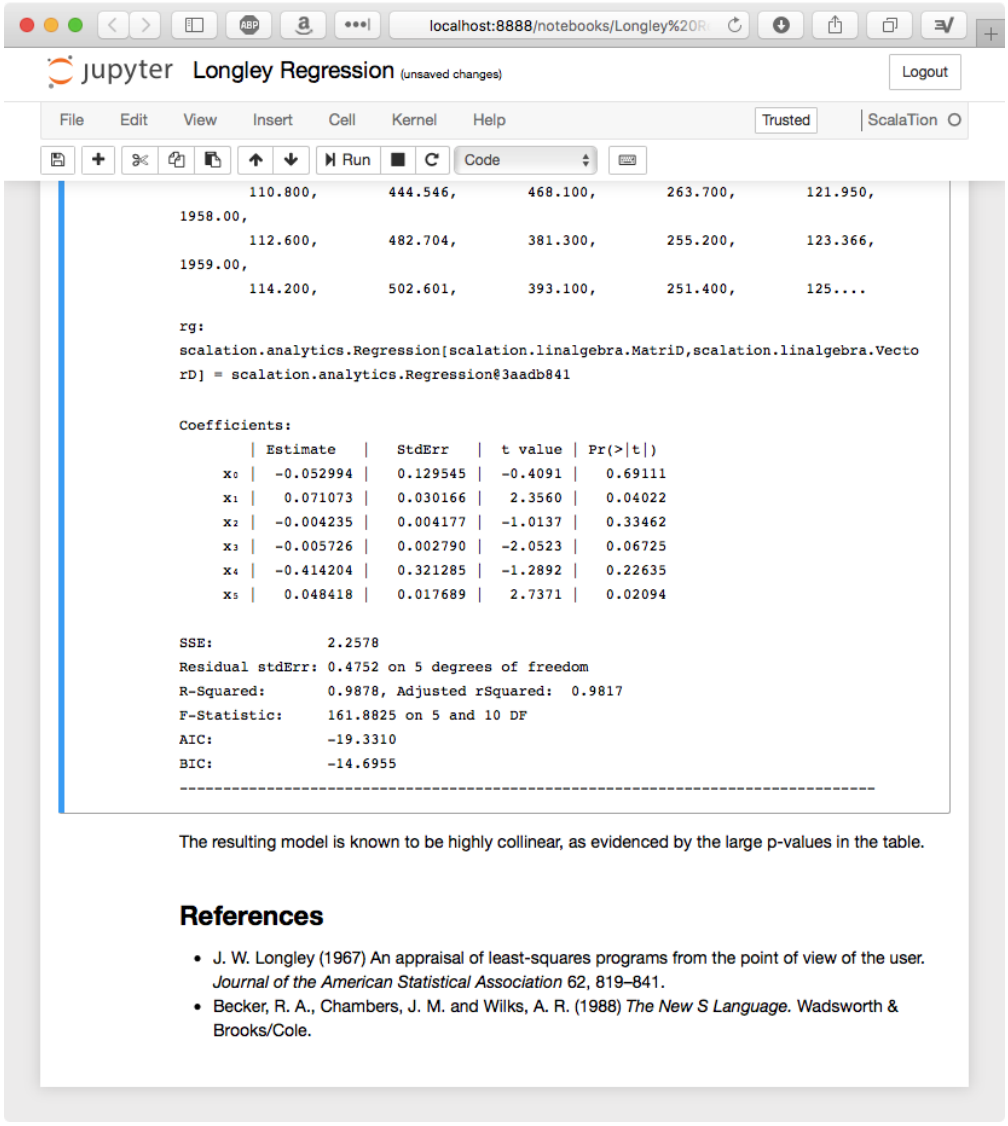


Figure 4.4: ScalaTion Kernel Regression Example after Run

Here we see an example of a Jupyter notebook in which students use the ScalaTion Kernel to explore a dataset using a multiple linear regression model. This shows the notebook after the code cells are executed using the kernel.



Chapter 5

ScalaTion Example: Functional Tight Clustering

Michael E. Cotterell [1,3], Xiaoxiao Sun [2,3], Nicholas Klepp [1], Hao Peng [1], Wenxuan Zhong [2], John A. Miller [1] and Ping Ma [2]. "A Functional Data Approach to Tight Clustering for Time Course Omics Data". 2017. [in preparation]

[1]Department of Computer Science, University of Georgia, Athens, GA, USA,

[2]Department of Statistics, University of Georgia, Athens, GA, USA, and

[3]These authors contributed equally to this work.

5.1 Abstract

In this article, we propose the functional tight clustering method, an efficient algorithm for the detection of most significant cluster patterns in time course omics data. Unlike other tight clustering methods, ours allows for significant pattern detection across all time points by modeling dependencies between the time points. The filtering method used in the algorithm provides higher insensitivity to the noise than the k-means based methods. In addition, the penalized method ameliorates issues with overfitting by imposing a complexity penalty on the model space. Simulation and real-data examples are presented to investigate the empirical performance of our functional data approach to tight clustering. Free and open source Scala code is available in the ScalaTion 'fda' package.

5.2 Introduction

Over the past 20 years, studies of temporal omics data, such as time course datasets generated by Chromatin ImmunoPrecipitation-sequencing (ChIP-Seq), Ribonucleic acid-sequencing (RNA-Seq) and Bisulfite sequencing, have improved the understanding of the structure and dynamics of biological systems [Berger et al., 2013]. Detecting a group of genes with similar expression patterns in temporal omics data accurately and efficiently is important in discovering novel regulation networks over the development of complex organisms [Graveley et al., 2011]. Clustering, one way to detect the groups, is an unsupervised learning technique that aims to group objects into "clusters" based on a similarity or distance metric. The

traditional clustering techniques, such as k-means, partitioning around medoids (PAM), self-organizing maps (SOM), and hierarchical clustering [MacQueen et al., 1967; Kaufman and Rousseeuw, 2009; Kohonen, 1990; Eisen et al., 1998], assign the genes into different clusters based on some correlation or distance measures. Although these methods have been applied in time course omics data successfully, they assume independence between different time points and thus omit the temporal dependence in the time course data [Ma and Zhong, 2008].

To better consider the correlation structure of temporal data, some functional methods treating the data as a path of a stochastic process have been proposed. Those methods can mainly be classified into two groups, filtering methods and adaptive methods. The filtering methods are two step approaches, which first approximate the data using some basis functions such as B-splines and then cluster the corresponding basis expansion coefficients [Abraham et al., 2003]. Instead of clustering coefficients directly, the adaptive methods consider the coefficients as random variables following a cluster-specific probability distribution [Jacques and Preda, 2014].

Some exceptions to explicitly handling the correlation structure of temporal data are reviewed below. In [Ma and Zhong, 2008], a mixed-effects model-based approach for clustering is presented by Ma et al. that uses a rejection-controlled expectation maximization (EM) algorithm with a smoothing spline-based penalized HendersonâĂŹs likelihood function to fit the data, estimate model parameters, and identify cluster membership. Their approach works particularly well when additional covariates are present in addition to fixed-effects functional data.

In [Futschik and Carlisle, 2005], a soft clustering technique is presented by Futschik and Carlisle that avoids the need for a priori pre-filtering of microarray data. Instead of providing concrete cluster assignments, soft clustering techniques (e.g., techniques usually implemented using fuzzy c-means [Gath and Geva, 1989]) provide a probabilistic sense of cluster membership measured between 0 and 1. Existing soft clustering techniques perform well in the identification of stable clusters at different levels of granularity, but can be improved when it comes to the identification of tight clusters.

None of the reviewed methods have been found to directly consider the presence of ubiquitously expressed and stimuli independent genes, commonly referred to as housekeeping genes [Eisenberg and Levanon, 2013], in the data. In [McLachlan et al., 2002], McLachlan et al. address the task of clustering microarray gene expression data on a very large number of genes from a much smaller number of tissue samples according to a model-based approach. The authors test the relevancy of each gene in the expression data using multi-component t mixture models, after which any genes that are determined irrelevant for clustering purposes are disregarded. In this way, housekeeping genes are discarded as irrelevant. However, this method relies on a probabilistic assumption of the data distribution, which is not satisfied for time course omics data in general except in the case of microarray gene expression data. In [Tseng and Wong, 2005], a bootstrap approach for identifying stable homogeneous clusters called "tight clustering" is presented by Tseng and Wong. This method produces tight and stable clusters without forcing all

points into clusters. The clusters are "tight" in the sense that they do not include outliers (i.e., they have minimal sum-of-squared-error); they are "stable" in the sense that the cluster membership tends to persist throughout multiple levels of subsample-based clustering. However, this method is not explicitly designed for time course data. Motivated by this method, in this article we propose an unsupervised clustering method, which incorporates functional data analysis approach, filtering methods, with the "tight clustering" algorithm proposed by Tseng and Wong [Tseng and Wong, 2005] in order to model the correlation structure between time points. We show that our functional tight clustering approach has three main benefits:

1. **De-noising; not sensitive to noise:** Repeated sub-sampling and filtering method based approach mitigate the potential affects of noisy data. In particular, this helps alleviate the impact of housekeeping genes in datasets that are analyzed for stimuli or regulatory reactions. In contrast with existing methods, our method works particularly well against housekeeping genes exhibiting a low degree of congruency.

2. **Low dimensional; more efficient:** By using a regression spline fit of the original time course data, our method can facilitate dimensionality reduction via lower order splines and knot vectors of reduced dimensionality. Such dimensionality reduction can produce speedup in the clustering process without sacrificing representation.

3. **Robust to outliers:** The use of repeated sub-sampling also minimizes the

effect outliers might have in the formation of clusters. This is because, by their very nature, the same outliers are unlikely to be present in subsequent subsamples of the dataset.

Both a simulation study and real-world use case are used to validate the identification of clusters via our method as well as highlight the benefits mentioned above. Overall, our results reveal that functional tight clustering is a suitable approach for rapidly and accurately identifying tight and stable clusters in time course omics data.

5.3 Materials and Methods

5.3.1 Data

Real Data

Understanding the mechanisms involved in the development of organisms is crucial in developmental biology. Most of the research has focused on some model organisms, such as *Drosophila melanogaster* (the common fruit fly) and *Caenorhabditis elegans* (a type of transparent roundworm). In this paper, we applied the methods, functional tight clustering (`ftclust`) and tight clustering (`tclust`), to two testing datasets from the modENCODE (Model Organism ENCyclopedia Of DNA Elements) project [Celniker et al., 2009], which provides comprehensive transcriptional profiling of *D. melanogaster* and *C. elegans* for different development stages and tissues. The first testing dataset contains the gene expression levels (Frag-

ments Per Kilobase of transcript per Million mapped reads; FPKM) of about 44,000 genes for *C. elegans*. The RNA-seq datasets under 24 embryonic developmental stages within 12 hours were used. The second testing dataset is from the developmental study of *D. melanogaster*. The time course RNA-seq datasets used in the paper include 12 embryonic samples collected every two hours within 24 hours. The expression levels (FPKM) of around 15,000 genes were profiled.

In real data, the true underlying cluster structure is unknown. To evaluate the performance of the cluster methods based on some criteria, we first defined six clusters for *C. elegans* and five clusters for *D. melanogaster* using the stage associated genes reported in [Li et al., 2014]. We treated these stage associated genes as clustered genes and the rest of the genes were scattered genes for *D. melanogaster*. To reduce the computing time, we only randomly selected half of all genes except the stage associated ones as the scattered genes for *C. elegans*.

Simulated Data

We simulated $R = 8$ clusters of gene expression values at 12 time points. The cluster sizes n_r were generated from $4p$ where p is distributed according to $P(\lambda)$, a Poisson distribution parameterized with $\lambda = 10$. In the i-th cluster, the gene expression values

$$X_i^r(t_j) = \sum_{h=1}^{H} (-1)^{(h+1)} h^{-1} Z_{ih} \vartheta_1(t_j, h), \tag{5.1}$$

where $j = 1, 2, \cdots, 16$, Z_{ih} is from the uniform distribution $U(-3, 3)$, and $\vartheta_1(t, h) = 1$ if $h = 1$ and $\sqrt{2}\cos((h-1)\pi t)$ otherwise. We added random noise with mean zero and variance σ^2 to the gene expression values $X_i^r(t_j)$. The value of σ was

adjusted to deliver six levels of signal-to-noise ratio (SNR = 0.5, 1, 1.5, 2, 2.5, and 3). In addition to the gene expression variability, we added the scattered genes to the clustered genes. The number of scattered genes was 25%, 50%, and 100% of the number of clustered genes. The parameter H in Equation (5.1) adjusts the roughness of the generated curves. We used $H = 3$ to generate 8 curves for the clustered genes and $H = 50$ to generate the expression values of scattered genes. Since the variation patterns over time were of primary interest and the input for tight clustering and our method were scaled data, i.e., z-scores, we did not transform the simulated data to positive values.

5.3.2 Smoothing Splines

Consider the expression levels in a column vector $\mathbf{y}$ for a gene at n time points $\mathbf{t} = (t_1, t_2, \ldots, t_n)$. The model can be written in vector notation, as described in [Ramsay and Silverman, 2005] and [Ma et al., 2006], as

$$\mathbf{y} = x(\mathbf{t}) + \epsilon, \tag{5.2}$$

where $\mathbf{y} \in \Re^n$ is a column vector containing the response (e.g., expression level), x is a "smoothed" function and $\epsilon \in \Re^n$ is a column vector containing the errors. In Figure 5.1, the "smoothed" function x is represented by the solid line and the observed data $\mathbf{y}$ are in green circles. The idea is that the smoothed function x is meant to reproduce the response vector $\mathbf{y}$ (before error) as closely as possible, assuming proper assumptions about the error distribution are made. This

is accomplished by representing $x(t)$ as a linear combination of K-many basis functions:

$$x(t) = \sum_{j=1}^{K} c_j \phi_j(t) = \mathbf{c} \cdot \boldsymbol{\phi}(t), \tag{5.3}$$

where $\mathbf{c} = (c_1, \ldots, c_K)$ is a vector of unknown coefficients to be estimated and $\boldsymbol{\phi}(t) = (\phi_1(t), \ldots, \phi_K(t))$ denotes a vector of K-many basis functions that are parameterized according to a non-decreasing "knot" vector $\boldsymbol{\tau} \subseteq [t_1, t_n]$. Similar to [Ramsay and Silverman, 2005], let us define $\boldsymbol{\Phi} \in \Re^{n \times K}$ as a design matrix where $\boldsymbol{\Phi}_{i,j} = \phi_j(t_i)$ for $1 \leq j \leq K$ and $t_i \in \mathbf{t}$. To estimate $\mathbf{c}$, one approach is via a penalized least squares criterion, expressed concisely in matrix form as

$$(\mathbf{y} - \boldsymbol{\Phi}\mathbf{c})' \boldsymbol{\Omega}^{-1} (\mathbf{y} - \boldsymbol{\Phi}\mathbf{c}) + n\lambda \mathbf{P}_p(x), \tag{5.4}$$

where $\boldsymbol{\Omega} = \text{Var}[\epsilon]$ is the variance of the error vector ϵ, λ is the penalty parameter, and $\mathbf{P}_p(x) = \int [D^p x(t)]^2 dt$ is the penalty imposed on the p-th derivative of the basis functions when estimating the coefficients of the spline function. The basis functions that are chosen for $x(\mathbf{t})$ depends on the nature of the response $\mathbf{y}$. If $\mathbf{y}$ is periodic, then a Fourier basis may be chosen [Ramsay and Silverman, 2005]. If $\mathbf{y}$ is not periodic, then it is common to choose a B-spline basis [Patrikalakis and Maekawa, 2010]. If the basis functions $\boldsymbol{\phi}$ are B-spline basis functions of order m (i.e., degree $m-1$) parameterized according to a non-decreasing augmented knot vector of the form $\boldsymbol{\tau} = ([t_1]^{m-1}, \mathbf{t}, [t_n]^{m-1})$ and $\mathbf{P}_2(x)$ is defined as a quadratic penalty on the second derivatives of the basis functions, then the function $x(t)$ is known as a smoothing spline [Ramsay and Silverman, 2005]. In such a scenario,

we would use $\mathbf{P}_2(x)$ for the penalty function, defined as

$$\mathbf{P}_2(x) = \int [D_t^2 x(t)]^2 dt = \mathbf{c}'\mathbf{Q}\mathbf{c} \tag{5.5}$$

where

$$\mathbf{Q} = \int [D_t^2 \phi(t)]^2 dt \tag{5.6}$$

and D_t^2 denotes the differential operator that takes the second derivative with respect to t. When it can be assumed that the residual vector $\epsilon \sim$ i.i.d. $\mathcal{N}(\mu, \sigma^2)$, then the conditional variance $\mathbf{\Omega} = \mathbf{I}$ (the identity matrix). Otherwise, $\mathbf{\Omega}$ must be estimated, usually in an iterative fashion, as is done in generalized least squares [Kariya and Kurata, 2004].

Estimating Coefficients

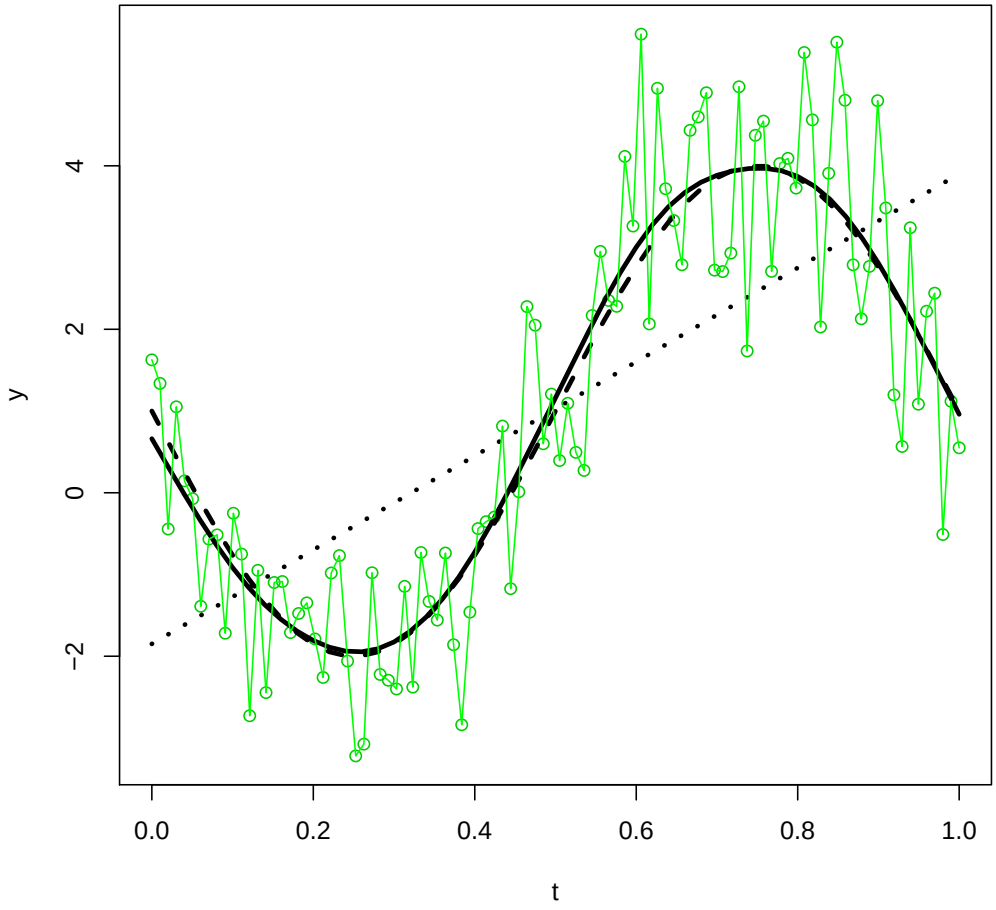
When $x(\mathbf{t})$ is a smoothing spline, the solution estimates $\hat{\mathbf{c}}$ that minimize the penalized least squares criterion in Equation (5.4) is

$$\hat{\mathbf{c}} = (\mathbf{\Phi}'\mathbf{\Omega}^{-1}\mathbf{\Phi} + \lambda\mathbf{Q})^{-1}\mathbf{\Phi}'\mathbf{\Omega}^{-1}\mathbf{y}. \tag{5.7}$$

For the equally spaced time points or for equal knot vectors (e.g., with B-splines), the basis functions are the same for each of the curves. Therefore, the smoothing function x in Equation (5.3) is summarized by the estimated coefficients $\hat{\mathbf{c}}$ of Equation (5.7).

Figure 5.1: Regression Spline Fit

A true function is shown in the solid line. Estimated fits for an arbitrary regression spline (green lines), a least squares fit (dotted lines), and a smoothing spline (dashed lines) are also shown. Raw data points are shown as green circles.



Estimating Penalty Parameter

An improper selection of λ may result in the over-fitting problem, see faded lines in Figure 5.1. We estimate the penalty parameter λ using the Generalized Cross-Validation (GCV) method, as described in [Ramsay and Silverman, 2005; Craven and Wahba, 1978], where the twice-discounted mean squared error criterion is expressed as

$$\mathrm{GCV}(\lambda) = \left(\frac{n}{n - df(\lambda)}\right)\left(\frac{\mathrm{SSE}}{n - df(\lambda)}\right) \tag{5.8}$$

where $\mathrm{SSE} = \mathbf{e} \cdot \mathbf{e}$ denotes the sum of squared error and

$$df(\lambda) = \mathrm{trace}\,[\mathbf{\Phi}(\mathbf{\Phi}'\mathbf{\Omega}^{-1}\mathbf{\Phi} + \lambda\mathbf{Q})^{-1}\mathbf{\Phi}'\mathbf{\Omega}^{-1}] \tag{5.9}$$

is the effective degrees of freedom computed from eigenvalues of the symmetric hat matrix (assuming $\mathbf{P}_2(x)$ in Equation (5.5) is used for the penalty function). An optimal estimate for λ can be obtained by minimizing Equation (5.8) using either a line search or numerical optimization technique. We used the golden section search method [Kiefer et al., 1953], which converges linearly toward an ϵ-accurate solution with $\mathrm{O}(\log 1/\epsilon)$ calls to the objective function. In Figure 5.1, the curve fitted by the smoothing splines with λ estimated by the GCV method is shown as dashed lines. If a smaller estimated variance for the fitted model is desired, then other potential criteria for choosing an optimal λ include the Generalized Information Criterion (GIC) [Konoshi and Kitagawa, 1996], Modified Akaike Information Criterion (mAIC) [Fujikoshi, 1997], and Generalized Bayesian Information Criterion (GBIC) [Konishi et al., 2004].

5.3.3 Functional Tight Clustering

In the previous section, we showed that the smoothing functions can be summarized by the estimated coefficients. Let $\hat{\mathbf{c}}^{(i)}$ denote the estimated coefficients for gene i where $1 \leq i \leq N$. Then, to cluster the fitted curves, we partition the N-many estimated coefficients $\hat{\mathbf{C}} = (\hat{\mathbf{c}}^{(1)}, \cdots, \hat{\mathbf{c}}^{(N)})'$. The unsupervised clustering method, k-means, is implemented to divide $\hat{\mathbf{C}} \in \Re^{N \times K}$ into k clusters. However, due to the random nature of the algorithm it is not guaranteed that k-means clustering will yield consistent, identical results each time the algorithm is run on the same dataset. Even worse, the algorithm falls in a local optimum with poor initial values [Hastie et al., 2009].

To address these problems, we propose using Tseng and Wong's "tight clustering" (`tclust`) [Tseng and Wong, 2005], a re-sampling-based algorithm that handles the aforementioned problems by only considering stable patterns that exist among subsets of the data throughout multiple levels of clustering. An implementation of their algorithm is available in the R `tightClust` package. The functional tight clustering algorithm (`ftclust`), an extension for use with estimated functional coefficient vectors, is outlined below.

In Figure 5.2, the outline of the functional tight clustering algorithm is presented. We take the estimated coefficient matrix $\hat{\mathbf{C}}$ as the input and then search the candidate tight clusters using the resampling method. The tight clusters are identified if they are repeatedly shown for different k (the number of clusters).

1: Take a random sub-sample $\hat{\mathbf{C}}^*$ from the original coefficients matrix $\hat{\mathbf{C}}$. Given the predefined k, calculate the cluster centers $\mathbf{Z}(\hat{\mathbf{C}}^*, k) = (Z_1, \cdots, Z_k)'$ using the k-means clustering algorithm.

2: Based on the estimated cluster centers calculated above, for each of the coefficient vectors $\hat{\mathbf{c}}^{(i)}$, where $i = 1, \cdots, N$, in the original matrix of coefficients, assign the coefficient vector to the cluster that minimizes the distance metric between the coefficient vector and the cluster center. The resulting clustering is represented by an N by N co-membership matrix $\mathbf{U}$, where $\mathbf{U}_{ij} = 1$ representing $\hat{\mathbf{c}}^{(i)}$ and $\hat{\mathbf{c}}^{(j)}$ are in the same cluster, and $\mathbf{U}_{ij} = 0$ otherwise [Tibshirani and Walther, 2005].

3: Repeat independent random sub-sampling B times to calculate $\mathbf{U}^{(b)}$, where $b = 1, \cdots, B$. Then the averaged co-membership matrix is $\bar{\mathbf{U}} = mean(\mathbf{U}^{(1)}, \cdots, \mathbf{U}^{(B)})$.

4: Search for a set of points $L = \{l_1, \cdots, l_m\} \subset \{1, \cdots, N\}$ such that $\bar{\mathbf{U}}_{l_i, l_j} \geq 1 - \delta$, where δ is a constant close to 0. Order sets with this property by size to obtain $L_{k1}, L_{k2}, \cdots$. These L sets are candidates of tight clusters.

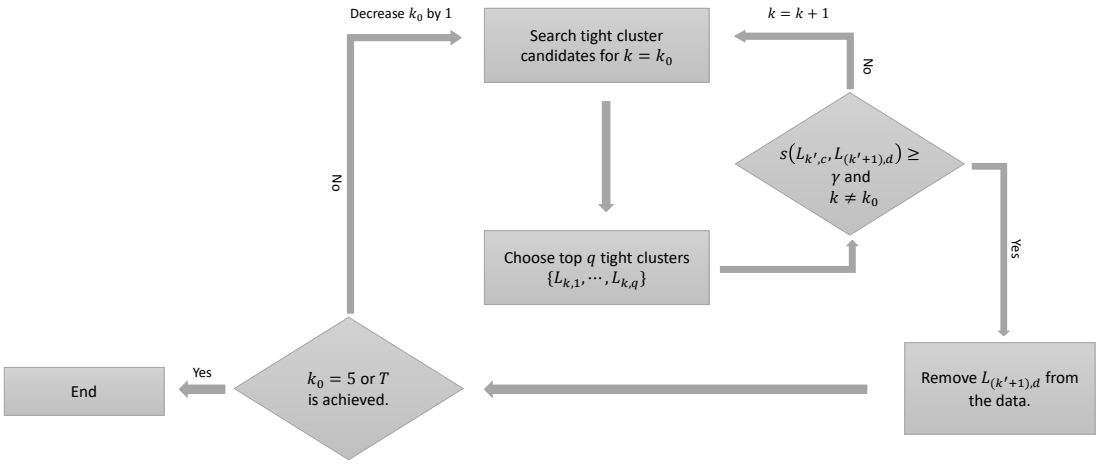
5: Apply steps 1–4 on the data $\hat{\mathbf{C}}$ for $k = k_0$, where k_0 is a suitable predefined starting value for the number of clusters. Choose the top q tight cluster candidates, namely $\{L_{k_0,1}, \cdots, L_{k_0,q}\}$. Let $k = k_0 + 1, k = k_0 + 2, \cdots$, choose the top q tight cluster candidates for each k. In this article, we set $q = 7$ as suggested in [Tseng and Wong, 2005].

6: Stop when $s(L_{k',c}, L_{(k'+1),d}) \geq \gamma$, where $s(L_i, L_j) = |L_i \cap L_j|/|L_i \cup L_j|$, and $|L|$ is the size of set L. $s(L_i, L_j) = 1$ if and only if sets L_i and L_j are identical. Here γ is a constant close to 1, $k' \geq k_0$, and $0 \leq c, d \leq q$. Identify $L_{(k'+1),d}$ as a tight and stable cluster. Remove it from the whole data.

7: Decrease k_0 by 1 and repeat steps 5 and 6 to identify the next tight cluster. The cluster selection terminates when k_0 is decreased to five or the target number of clusters, T, is achieved.

Algorithm 1: Functional Tight Clustering Algorithm

Figure 5.2: Outline of Functional Tight Clustering Algorithm



5.3.4 ScalaTion Implementation

We have implemented `ftclust`, the Functional Tight Clustering algorithm, in ScalaTion [Miller et al., 2010], a Scala framework for exploring a modeling continuum that includes analytics, simulation and optimization. The underlying algorithm used to cluster the subsample coefficient matrix is the Hartigan-Wong k-means algorithm [Hartigan and Wong, 1979]. Instead of the standard randomized seeding technique, our implementation includes *k-means++* initialization [Arthur and Vassilvitskii, 2007], a method that picks cluster centers with probability proportional to their contribution to the overall optimization problem.

Further information about ScalaTion can be found at `http://cobweb.cs.uga.edu/~jam/scalation.html` ↗ . The source code and instructions for our

functional tight clustering implementation can be found at `https://github.com/scalation/fda/tree/ftclust` ↗ . The software is free and open source under an MIT license.

5.4 Results

5.4.1 Simulation Results

In the simulation, we assessed the proposed method according to the weighted Rand index [Thalamuthu et al., 2006], R^*. The heatmaps of the simulated examples are shown in Figure 5.3. As the signal to noise ration (SNR) increased, it was much easier to extract signals from the noise. We applied the proposed method (`ftclust`), the Mfuzz method (`mfuzz`), and the original tight clustering (`tclust`) to the simulated datasets. We set the fuzzification parameter $m = 1.25$ for the method `mfuzz`. We had 100 runs for each combination of different percentages of scattered genes and SNRs. As seen in Figure 5.4, when the proportion of scattered genes is more than 50%, the `tclust` method outperforms `tclust` method in terms of R^*. The results also showed that regardless of the proportion of scattered genes, the proposed method performed better than `mfuzz` and `tclust` in terms of R^* under different SNRs.

The clustering results for the `ftclust` and `tclust` when $SNR = 1.5$ and the percentage of scattered genes is 50% are shown in Figure 5.5. Based on the cluster labels shown on the left side of the heatmap, the proposed `ftclust` method provides a clear advantage over the original `tclust` method with respect to recovering

Figure 5.3: Heatmaps of Simulated Data with Increasing SNR

Simulated data with increasing SNR under the same random seed number. The clustered genes in different clusters are divided by white horizontal lines.

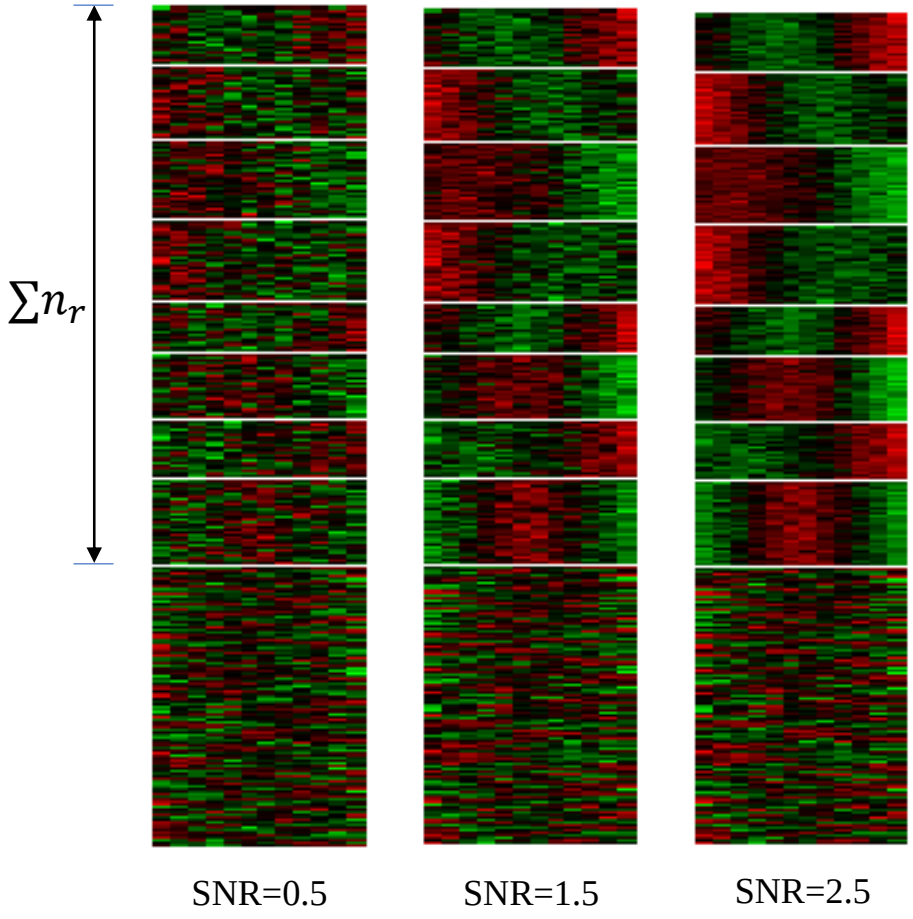
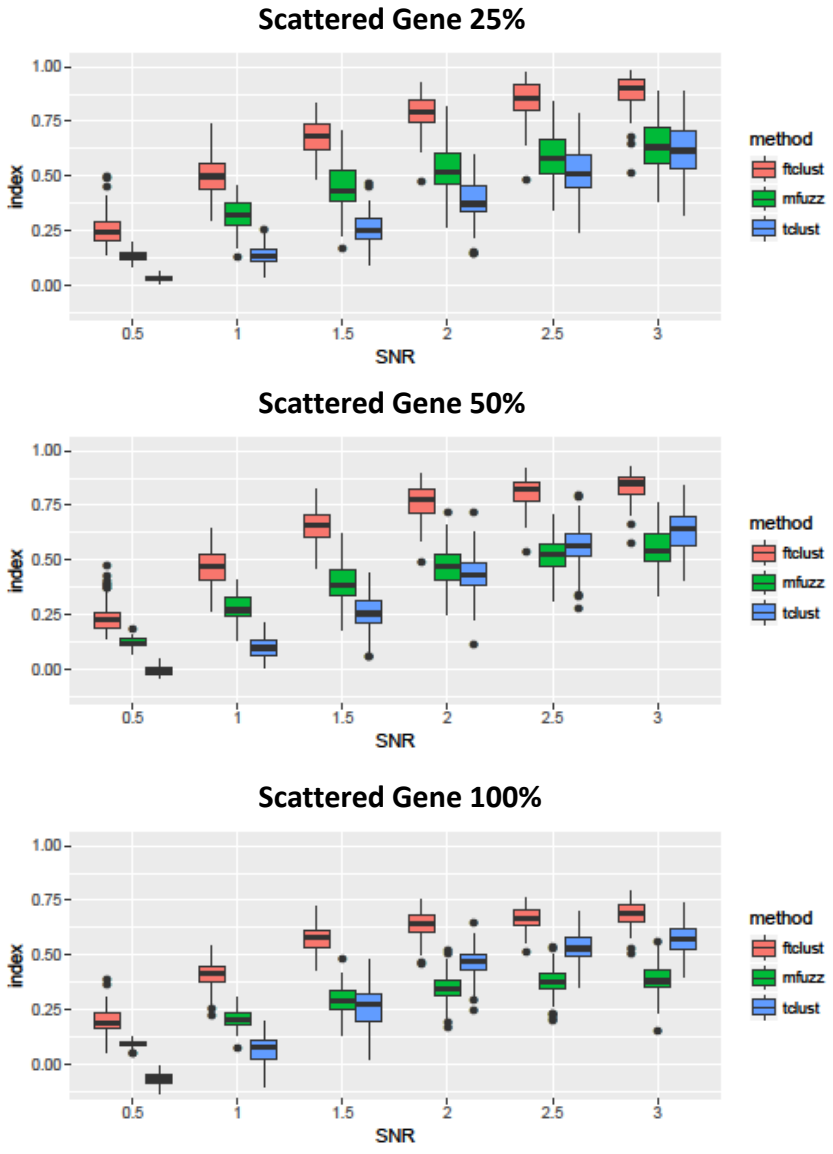


Figure 5.4: Box plots for Simulated Data

The box plots of R^* under six different SNRs, based on 100 simulation runs. The box plots from top to bottom panels respectively represent the R^*s for simulated datasets with different proportions of scattered genes. Box plots with different colors represent the results from different approaches.



the true clusters. In the right two panels of Figure 5.5, only the clusters with the percentages of overlapped genes between themselves and true clusters above 50% are labeled.

5.4.2 Real Data Results

The number of clusters for the `ftclust` and `tclust` was set to 6 and 5 for the data of *C. elegans* and *D. melanogaster*, respectively. The heatmaps of the stage associated genes for *C. elegans* and *D. melanogaster* are shown in the top left and bottom left panels of Figure 5.6. For *C. elegans* data, the R^* of `ftclust` is 0.19 whereas the R^* of `tclust` is 0.07. The cluster results for *ftclust* and *tclust* are shown in the top middle and top right panels of Figure 5.6. It was clear to see that the proposed method detected more stage associated genes. For *D. melanogaster* data, the R^*s of `ftclust` and `tclust` are 0.18 and 0.01. The proposed method identified 77.4% genes of the first stage associated cluster, which was the biggest stage associated genes cluster. However, only 22.6% of them were detected by the `tclust`. Such results are observed at the bottom of the middle and right panels of Figure 5.6.

To reduce the impact of randomness, we applied `ftclust` and `tclust` to both datasets 10 times. The comparison of the two methods is shown in Table 5.1. In the table, the large standard deviations (SD) of R^* was caused by two observations where the proposed method happened to have too large or small R^*s. For instance, the maximum value of R^* for `ftclust` in *C. elegans* was 0.23, which brought large variations in estimating standard deviation. In terms of all the other measures,

Figure 5.5: Heatmap Comparison

Heatmaps of the underlying true cluster structure, cluster results of the proposed method, and cluster results of the original tight clustering. The purple + symbols and yellow − symbols represent the cluster and scattered genes, respectively. The clustered (left panel) or estimated clustered (middle and right panels) genes in different clusters are divided by white horizontal lines. The true cluster labels and estimated ones are shown on the left of each heatmap.

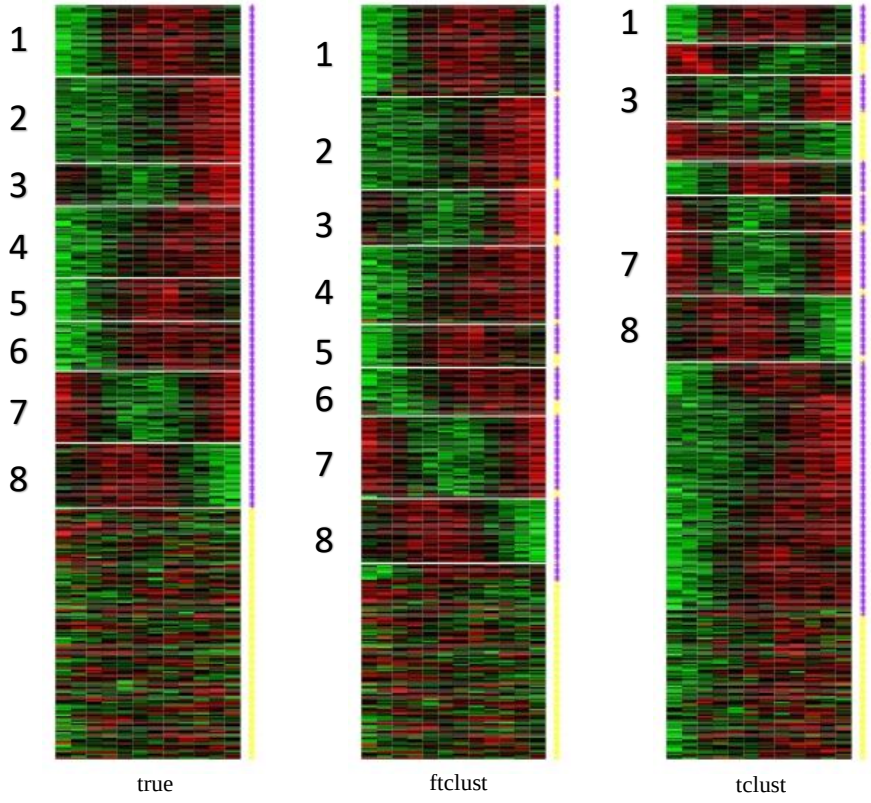


Table 5.1: Performance Comparison of Results

The median and mean R^* values over 10 runs are presented. The standard deviation (SD) is also presented for each run.

`ftclust`	Median	Mean	SD
C. elegans	**0.13**	**0.11**	0.08
D. melanogaster	**0.15**	**0.12**	0.06
`tclust`	Median	Mean	SD
C. elegans	0.09	0.10	**0.03**
D. melanogaster	0.03	0.05	**0.05**

the proposed method clearly showed the advantage in terms of R^*. Since there were true clustered genes in the pre-defined scattered genes, it was expected that the estimated weighted Rand index would not be high. However, the `ftclust` typically could have one estimated clusters including the predefined ones.

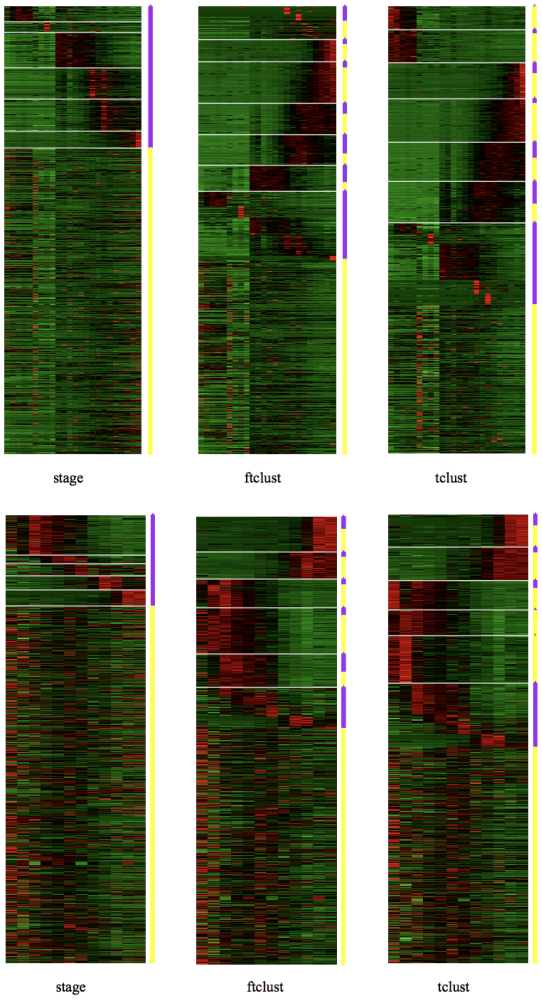
5.5 Discussion

We have presented the functional tight clustering method. Across different signal to noise ratios and proportions of scattered genes, the `ftclust` method had the best performance in terms of clustering accuracy.

The `ftclust` method can be applied to different kinds of time course omics data, such as ChIP-seq and RNA-seq data. It can also potentially be applied to the data over the spatial domain, such as genome and protein sequence coordinates.

Figure 5.6: Heatmaps of the Standardized Gene Expression Cluster Results

Top: heatmaps of the standardized gene expression levels, cluster results of `ftclust`, cluster results of `tclust` for *C. elegans* data. Bottom: heatmaps of the standardized gene expression levels, cluster results of `ftclust`, cluster results of `tclust` for *D. melanogaster* data. The purple + symbols and yellow − symbols represent the pre-defined stage associated cluster and scattered genes. The stage associated cluster (left panel) or estimated cluster (middle and right panels) genes in different clusters are divided by white horizontal lines.



For instance, in DNA methylation studies, the methylation levels observed across the whole genome can also be treated as the function values over the spatial domain.

When the time course omics data are collected sparsely and irregularly, the methods for sparsely observed functional data [Yao et al., 2005] should be implemented to estimate the coefficients matrix $\hat{C}$. One obvious consideration is when sampled functional data require different basis functions to fit each curve. Our algorithm currently assumes each curve is fitted using the same set of basis functions. In this case, working directly with the estimated coefficients is sufficient. However, different basis functions can be accommodated using a generalized functional distance metric in the underlying k-means routine. Such a metric between the n-th derivatives of two functions $f(t) = \mathbf{c}_f \cdot \boldsymbol{\phi}(t)$ and $g(t) = \mathbf{c}_g \cdot \boldsymbol{\psi}(t)$ can be expressed as

$$\delta_{n,p}(f, g) = \left(\int |D_t^n(\mathbf{c}_f \cdot \boldsymbol{\phi}(t)) - D_t^n(\mathbf{c}_g \cdot \boldsymbol{\psi}(t))|^p dt \right)^{1/p}$$

where $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ are vectors containing the basis functions for f and g, respectively. Given this formulation, $\delta_{0,1}$ and $\delta_{0,2}$ denote the standard $L1$ and $L2$ distance metrics, respectively. Further investigation in this direction is surely needed to accommodate time course omics datasets sampled at different rates.

5.6 Acknowledgments

Sun, Zhong, and Ma's research was supported by U.S. National Science Foundation under grants DMS-1440037, DMS-1440038, and DMS-1438957 and by U.S. National Institute of Health under grants R01GM122080 and R01GM113242.

Conflict of interest statement.

None declared.

Chapter 6

Applied Open Data Science: Website & Example Notebooks

6.1 Introduction

Two big problems in open science are accessibility and reproducibility. How do you make data science investigations and methods available to others? How do you make it easier for data scientists to share and verify results? While open notebook platforms such as Jupyter[1] and JupyterHub[2] provide a partial solution, there is still much room for improvement.

In this chapter, we propose the Applied Open Data Science (AODS) umbrella of applications and guidelines as a way to improve on existing, open methods for the dissemination of open notebooks. To demonstrate our methods, we provide

[1] https://jupyter.readthedocs.io/en/latest/

[2] https://jupyterhub.readthedocs.io/en/latest/

easily shareable, modifiable, and executable example notebooks with ScalaTion Kernel support. Readers are actively encouraged to try the examples using the JupyterHub installation provided on the author's AODS website[3]. As this site is provided primarily for the readers of this dissertation, only four users may be logged in to this website simultaneously. Readers are also encouraged to see Chapter 4 for information on how to setup their own Jupyter installation with ScalaTion Kernel support, either locally or as a containerized application.

There are some known setups that are similar to what we describe in this chapter. Fernández et al. [2016] outline the general infrastructure and benefits of their JupyterHub installation at the European Spallation Source (ESS), including an emphasis on their cloud storage integration and use of Docker containers for Jupyter instances. Our approach differs in the provision of detailed documentation that facilitates replication of our setup. Additionally, Milligan [2017] discusses the Minnesota Supercomputing Institute's initiative to provide an interactive high performance computing (HPC) service using JupyterHub. Of particular interest is the `BatchSpawner` module they implemented for batch job scheduling and job profile configuration, now available as an officially supported Jupyter component. Since our approach is minimal in the way it modifies JupyterHub, it should be possible to use the `BatchSpawner` module with one of our JupyterHub installations.

There are also some existing projects that attempt to address some of the research questions raised in this chapter. In particular, Jupyter's Nbviewer[4] project provides a notebook rendering service that users can use to share previews of their

[3] `http://aods.io/`

[4] `http://nbviewer.jupyter.org`

notebooks and facilitate easy downloading to their local machines. However, unlike our approach, this service does not currently facilitate directly uploading a notebook to an existing Jupyter or JupyterHub installation. There is also the hub share[5] project, which provides a directory sharing service for users of a JupyterHub installation. However, this service is currently only available as a REST-like service, which imposes a steep learning curve on many users.

The rest of this chapter is organized as follows: Section 6.2 provides a detailed overview of the AODS project, including its guidelines and sub-projects; a list of example notebooks using the ScalaTion big data framework that readers can view, modify, and run is provided in Section 6.3; impact is discussed in Section 6.4; and Section 6.5 presents conclusions and future work.

6.2 AODS Website & Projects

The AODS project website is currently composed of the AODS homepage as well as subdomains that host live instances of the AODS JupyterHub and AODS Upload projects. Each of these sub-projects are described in more detail in the following subsections.

6.2.1 AODS Homepage

The AODS homepage, located at `aods.io`[6], includes a short introduction to the AODS Guidelines and sub-projects. The AODS Guidelines are a set of charac-

[5] `https://github.com/jupyterhub/hubshare`
[6] `http://aods.io/`

teristics that AODS projects must strive to adhere to. These guidelines provide an expectation that materials related to each project be open, accessible, and reproducible. In an effort to fulfill these characteristics for the homepage itself, all of the homepage content hosted on `aods.io` is free and open source under a Creative Commons Attribution-ShareAlike (BY-SA) 4.0[7] license and available via the `aods-site`[8] repository on GitHub. For each of the AODS sub-projects, a brief description of the software is provided as well as a link to the project's repository where source code and documentation can be found. Additionally, links to live test installations of each project's associated web application are provided in order to demonstrate and stress the importance of openness and accessibility. The documentation provided for each project includes instructions on how to replicate the test setup. The AODS homepage and associated web applications are currently hosted using one or more Linode[9] virtual server instances running the Ubuntu 17.10 distribution of Linux 4.13, a convenient setup that provides extensive infrastructure monitoring support.

6.2.2 AODS JupyterHub Project

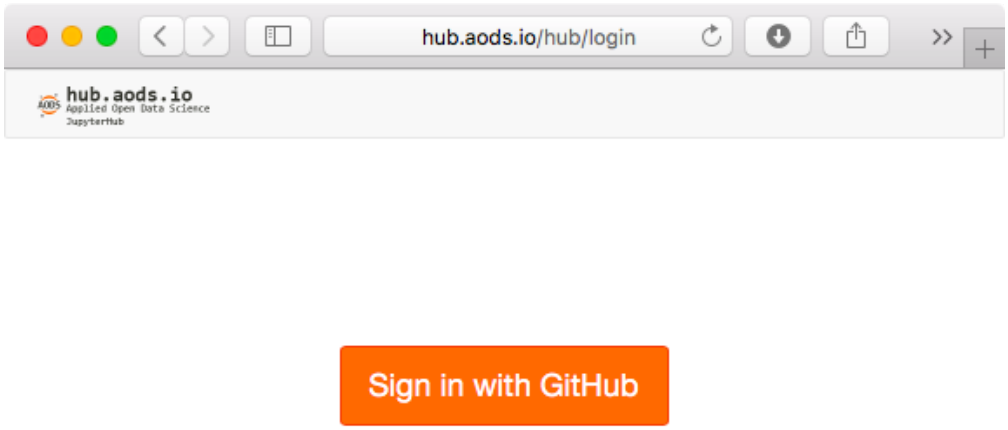
The AODS JupyterHub project, hosted here[10], is comprised primarily of a customized JupyterHub application that supports: i) social authentication; ii) the ScalaTion big data framework in Jupyter notebooks; and iii) providing new users with example notebooks to help them get started. This project currently does

[7] `https://creativecommons.org/licenses/by-sa/4.0/`
[8] `https://github.com/aods-io/aods-site`
[9] `https://www.linode.com/`
[10] `https://github.com/aods-io/aods-jupyterhub`

not modify the JupyterHub code base. Instead, it comprises a set of detailed instructions on how to configure a JupyterHub installation so that it provides the support described above and adheres to the AODS Guidelines. In the event that changes to JupyterHub's code base are needed in future to satisfy a project goal, the maintainers of the project have made a commitment to send any such changes back upstream to the main JupyterHub project.

Figure 6.1: Applied Open Data Science (AODS) JupyterHub

The `hub.aods.io` website is made available so that user can easily create and execute notebooks with ScalaTion Kernel support as well as try the example notebooks provided in Chapter 6.



As seen in Figure 6.1, a live, proof of concept test installation of the project's customized JupyterHub is available at `hub.aods.io`[11], primarily targeted at read-

[11] `http://hub.aods.io/`

ers of this dissertation. Users of the site can create, modify, and run notebooks with ScalaTion big data framework support. The project documentation includes detailed instructions on how to replicate the proof of concept installation. The two primary goals of this project are i) to make replicating the test installation as easy and as painless as possible; and ii) to not let modifications prevent the inclusion of existing contributions to the Jupyter and JupyterHub projects. Project documentation is free and open source under a Creative Commons BY-SA 4.0 license.

Social authentication, sometimes referred to as social logins, was chosen as the authentication method for AODS JupyterHub installations. Instead of forcing users to create a separate login account for your application, social authentication lets users authenticate using their existing personal accounts for various social websites. Although social authentication is a definite compromise between privacy and convenience [Gafni and Nissim, 2014; Scott et al., 2016], the choice to use the method was made primarily due to its prevalence among various data science related websites (e.g., Kaggle[12], Cross Validated[13], etc.) and its application in a recent research article related to big data [Madani et al., 2017]. Additionally, administrators are still free to provide different levels of authorization to their JupytHub installation, even if social authentication is configured. The recommended way of providing social authentication support in AODS JupyterHub installations is using the OAuth 2.0[14] protocol. This protocol is recommended

[12] https://www.kaggle.com

[13] https://stats.stackexchange.com

[14] https://oauth.net/2/

for two reasons: i) it is an industry standard authorization protocol supported by many popular social websites; and ii) it is cross-compatible with CAS[15], a protocol currently used by various academic institutions (e.g., the University of Georgia[16]).

In the AODS JupyterHub test installation, OAuth 2.0 support is provided using the `oauthenticator`[17] package. GitHub was chosen as the social website for authentication since that is where the AODS JupyterHub project is hosted. Administrators who wish to replicate the setup should install `oauthenticator` according its package documentation, register an OAuth application on GitHub by following the instructions found here[18], then add and configure the following settings in their JupyterHub installation's `jupyter_config.py` file:

```
from oauthenticator.github import LocalGitHubOAuthenticator
c.JupyterHub.authenticator_class = LocalGitHubOAuthenticator
c.LocalGitHubOAuthenticator.oauth_callback_url = ''
c.LocalGitHubOAuthenticator.client_id = ''
c.LocalGitHubOAuthenticator.client_secret = ''
c.LocalGitHubOAuthenticator.create_system_users = True
```

If configured correctly, users of the JupyterHub installation can now login using their GitHub account. The first time they attempt to login, they will be prompted by GitHub to authorize the application. In order to continue authenticating, authorization must be granted. This authorization can be revoked at a later time

[15] https://apereo.github.io/cas/4.1.x/protocol/CAS-Protocol.html

[16] https://eits.uga.edu/access_and_security/cas/

[17] https://github.com/jupyterhub/oauthenticator

[18] https://developer.github.com/apps/building-integrations/setting-up-and-registering-oauth-apps/registering-oauth-apps/

if the user desires.

Support for the ScalaTion big data framework is provided in the AODS JupyterHub test installation using the ScalaTion Kernel project. An overview of ScalaTion Kernel and a set of various installation instructions are provided in Chapter 4.

In order to help users get started exploring data science investigations, the AODS JupyterHub test installation also provides copies of the latest example notebooks (described in Section 6.3) from the ScalaTion Kernel project (described in Chapter 4). These notebooks are supplied during the user creation process via a custom `add_user.sh`[19] script, which is called the first time a user successfully authenticates with the JupyterHub installation. When a user logins, they can find the example notebooks in a directory called `default-notebooks`. Administrators who wish to replicate this setup should first make sure that `create_system_users` is set to `True` in `jupyterhub_config.py` for their chosen authenticator. Next, they should download `add_user.sh` to the server, perhaps placing it in the same directory as `jupyterhub_config.py`, and grant it execute permission (e.g., using the `chmod` command). Then, they should add and configure the following setting in their JupyterHub installation's `jupyterhub_config.py` file:

```
c.LocalAuthenticator.add_user_cmd = ['/path/to/add_user.sh']
```

Now, when new users successfully authenticate with the JupyterHub installation, the `add_user.sh` script gets executed and copies of the example notebooks are placed in `default-notebooks` under their home directory. By default, the script places new users into a group called, "aodshub." Administrators can edit the script

[19] `https://github.com/aods-io/aods-jupyterhub/blob/master/add_user.sh`

to change this behavior.

Lastly, the AODS JupyterHub test installation is served using the Nginx[20] web server configured to pass external requests for `hub.oads.io` on port 80 to the locally running installation running on port 8000 using a proxy. If an administrator wishes to replicate this setup, then they can use an Nginx site configuration similar to the following where `hostname.tld` refers to the external hostname that users will use to access the installation:

```
server {

  listen 80;
  server_name hostname.tld;

  location / {
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_pass http://127.0.0.1:8000;
  }

  location ~ /api/kernels/ {
    proxy_pass         http://127.0.0.1:8000;
    proxy_set_header   Host $host;
    proxy_set_header   X-Real-IP $remote_addr;
```

[20] `http://nginx.org` ↗

```
    proxy_http_version 1.1;
    proxy_set_header   Upgrade "websocket";
    proxy_set_header   Connection "Upgrade";
    proxy_read_timeout 86400;
  }

}
```

Although the AODS JupyterHub test installation does not use the Apache2 web server, the project maintainers anticipate adding documentation to support it since it is a popular option.

6.2.3 AODS Upload Project

The AODS Upload project, hosted here[21], is comprised primarily of a web application that facilitates the automatic uploading of notebooks to a JupyterHub workspace. The web application supports: i) social authentication; ii) rendering notebook previews; and iii) and linking a user directly to the uploaded notebook in their JupyterHub workspace. Written in Python using the Django library, the application is designed to perform a minimal set of operations while maintaining a small footprint on the system. Its functionality and implementation are directly motivated by and adhere to the AODS Guidelines. Project documentation is free and open source under a Creative Commons BY-SA 4.0 license. Source code is also free and open source under an MIT license.

[21] https://github.com/aods-io/aods-upload

A live, proof of concept test installation of the project's web application is available at `upload.aods.io`[22], primarily targeted at readers of this dissertation. When an authenticated user follows an AODS Upload link, they are presented with a rendered preview of the notebook and asked to confirm the upload. Once confirmed, the user is presented with the notebook's filename, as saved in their AODS workspace, as well as a convenient link to directly open the newly uploaded notebook on the AODS JupyterHub test installation site. Currently, only notebook links from locations in a configurable whitelist are supported. At the time of this writing, the whitelist includes the following link prefixes:

- `https://github.com/scalation/scalation_kernel/raw/`
- `http://www.cs.uga.edu/`

The project documentation includes detailed instructions on how to replicate the test installation. The primary goal of this project is provide users with an easier alternative to manually uploading notebooks to their JupyterHub workspace. To that end, using the AODS Upload test installation, authenticated users can use shareable AODS Upload links to preview and upload notebooks in a more automated fashion, allowing them to avoid downloading a notebook's `.ipynb` file to their local machine as an intermediate step.

In order to complement the AODS JupyterHub project described in Section 6.2.2, social authentication using OAuth 2.0 was chosen as the authentication method. This support is provided using the `social-auth-app-django`[23] package. GitHub

[22] `http://upload.aods.io/`

[23] `https://github.com/python-social-auth/social-app-django`

was chosen as the social website for authentication in order to match the service used for the AODS JupyterHub test installation. Administrators who wish to replicate the setup should install `social-auth-app-django` according its package documentation, register an OAuth application on GitHub by following the instructions found here[24], then add and configure the following settings in the project's `upload_site/secret_settings.py` file (which should be created if it does not exist):

```
SECRET_SETTINGS_SOCIAL_AUTH_URL_NAMESPACE = 'social'
SECRET_SETTINGS_SOCIAL_AUTH_GITHUB_KEY = ''
SECRET_SETTINGS_SOCIAL_AUTH_GITHUB_SECRET = ''
```

If configured correctly, users with an account on the corresponding JupyterHub installation can now login using their GitHub account. The default behavior of the AODS Upload test installation is to not authorize authenticated users who have not previously logged in to the corresponding AODS JupyterHub installation. This decision was made to ensure that account creation on the server is handled by the custom `add_user.sh` script provided by the AODS JupyterHub project.

Lastly, the AODS Upload test installation is deployed using the uWSGI[25] package and served using the Nginx[26] web server configured to pass external requests for `upload.oads.io` on port 80 to the local uWSGI running on port 8888 using a proxy. Below is an example that starts the web application locally on port 8888

[24] `https://developer.github.com/apps/building-integrations/setting-up-and-registering-oauth-apps/registering-oauth-apps/`

[25] `http://uwsgi-docs.readthedocs.io/en/latest/`

[26] `http://nginx.org`

using uWSGI:

```
$ uwsgi --http :8888 \
        --chdir /path/to/aods-upload/upload_site \
        --wsgi-file /path/to/aods-upload/upload_site/upload_site/wsgi.py \
        --virtualenv /path/to/aods-upload
```

If a user wishes to replicate the test setup, then they can use an Nginx site configuration similar to the following where `hostname.tld` refers to the external hostname that users will use to access the installation:

```
server {

  listen 80;
  server_name hostname.tld;

  location /static/ {
    alias /path/to/static/;
    gzip_static on;
    expires max;
    add_header Cache-Control public;
  }

  location / {
    proxy_set_header Host $host;
```

```
    proxy_set_header X-Real-IP $remote_addr;
    proxy_pass http://127.0.0.1:8000;
  }

}
```

Although the AODS Upload test installation does not use the Apache2 web server, the project maintainers anticipate adding documentation to support it since it is a popular option.

6.3 Example Notebooks

In this section, we provide various example Jupyter notebooks that use the ScalaTion big data framework to perform a data science investigation or demonstrate a topic related to data science. Readers are encouraged to use the AODS test installations to explore the example notebooks described in this section. For new and existing users of the AODS JupyterHub installation at `hub.aods.io`, this can be done in a few different ways, as outlined below:

1. **Default Notebooks:** Users can navigate to the `default-notebooks` directory in their AODS workspace to see a list of the example notebooks presented in this dissertation along with any additional example notebooks hosted by the ScalaTion Kernel project. The notebooks in this directory are personal copies of the example notebooks that were available during the user's account creation. As such, users should feel free to modify and execute

them as they see fit.

2. **AODS Upload Link:** Users can use AODS Upload links, described earlier, to upload notebooks directly to their AODS workspace. Each example notebook described later in this section has a corresponding "upload" link that is provided using the AODS Upload test installation at `upload.aods.io`. After following an AODS Upload link and confirming the upload, the site will provide the user with a link to directly access the uploaded notebook within their AODS JupyterHub workspace.

3. **Direct Download & Upload:** Users can directly download notebooks to their local machine, then upload it to their AODS workspace. Each example notebook described later in this section has a corresponding "download" link that is provided. Once a notebook is downloaded, users can upload it using the "upload" button that is available in their AODS JupyterHub workspace.

4. **Read Only:** Users can view a rendered, read only preview of a notebook using preview links. Each example notebook described later in this section has a corresponding "view" link that is provided using the preview feature of the the AODS Upload test installation at `aods.io/preview/`. While a preview link will properly render previously saved output, it cannot be used to run a notebook to generate new output.

Each of the methods described above demonstrates an open and accessible way to explore a disseminated data science investigation. If readers wish to run the notebooks on their own Jupyter installations, then should follow the instructions

provided in Chapter 4 to install ScalaTion Kernel. The list of example notebooks is provided below.

- **Longley's Economic Regression Data:** Illustrates the exploration of the Longley macroeconomic dataset [Longley, 1967] using a multiple linear regression model. The dataset is provided in an open data format (i.e., CSV) and downloaded directly from the Internet from within the notebook. The investigator notes that this dataset is known to be highly collinear. Using ScalaTion's `Regression` class, a multiple linear regression model is trained and model diagnostics are provided that support this claim. See Figure 6.2.
  (view ↗ , download ↗ , upload ↗)

- **Clustering of Edgar Anderson's Iris Data:** Illustrates how to use K-means clustering on Fisher's and Anderson's iris dataset [Becker et al., 1988] to investigate the relationship between iris petal length and species. The dataset is provided in an open data format (i.e., CSV) and downloaded directly from the Internet from within the notebook. The investigator wants to explore the relationship between iris petal size (i.e., width and height) and the three iris species provided in the dataset. Using ScalaTion's `KMeansPPClusterer` class, petal sizes are clustered into three clusters. The cluster assignments suggest to the investigator that a relationship between petal size and species does exist. See Figure 6.3.
  (view ↗ , download ↗ , upload ↗)

- **Deriving Multiple Linear Regression:** Illustrates how to derive and ap-

ply the least squares solution for multiple linear regression. The investigator begins by deriving the formula to estimate the coefficient vector in a multiple linear regression that minimizes the sum of squared error using Markdown and LaTeX. To explore this derivation, a response vector is simulated using a design matrix, a known coefficient vector, and random noise. Then, the coefficient vector for a multiple linear regression model is estimated directly using the derived method and used to produce a predicted response. The notebook concludes with a derivation of the model's residual sum of squared error (SSE) using a domain-specific language. See Figure 6.4.
(view ↗ , download ↗ , upload ↗)

- **Regression Splines:** Illustrates how to approximate a function for the number (in thousands) of Australian residents over time from the `austres` dataset [Brockwell and Davis, 2016] using a regression spline. The dataset is provided in an open data format (i.e., CSV) and downloaded directly from the Internet from within the notebook. First, the investigator gives a small introduction to regression splines, then proceeds to work through the steps to produce the approximated function using B-spline basis functions. This notebook also demonstrates how to easily incorporate additional source code into a notebook. See Figure 6.5.
(view ↗ , download ↗ , upload ↗)

Figure 6.2: Longley's Economic Regression Data Notebook

Here we see an example of a Jupyter ScalaTion notebook which illustrates the exploration of the Longley macroeconomic dataset [Longley, 1967] using a multiple linear regression model.

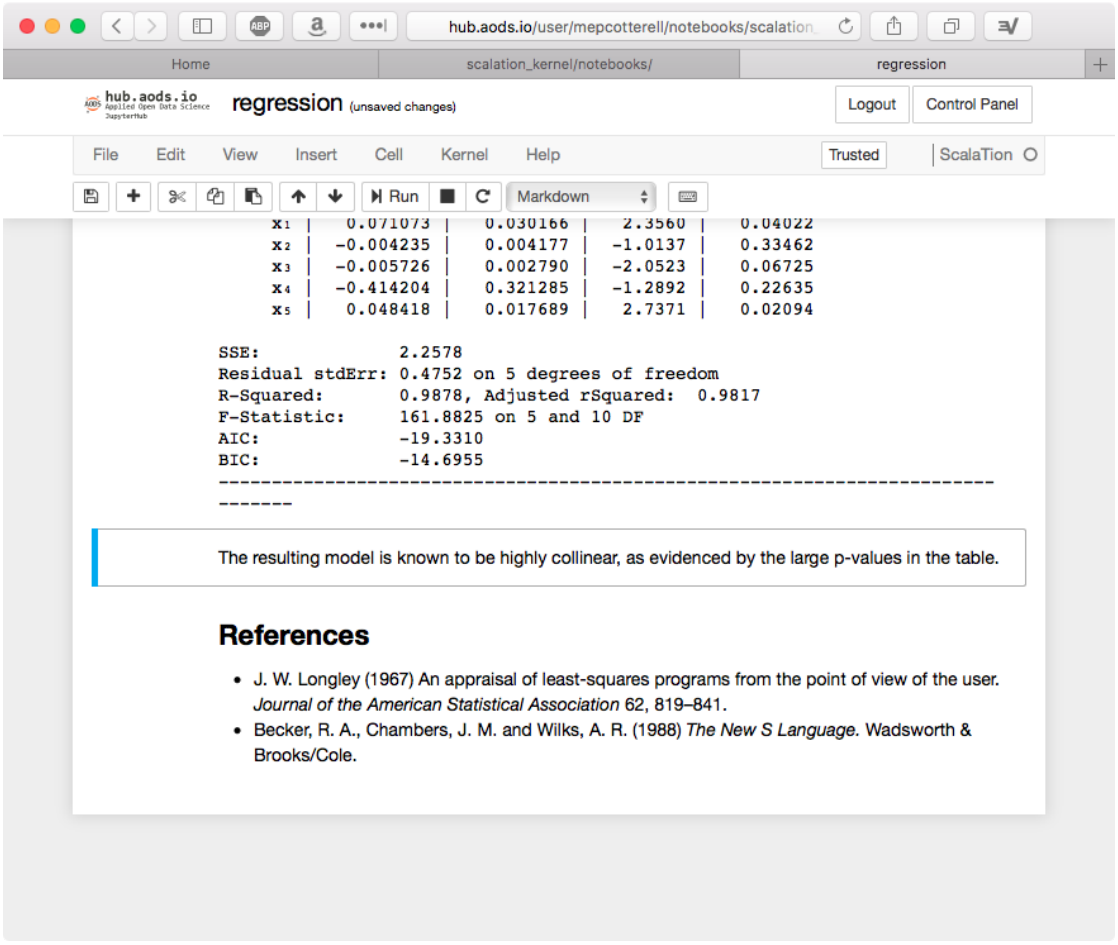
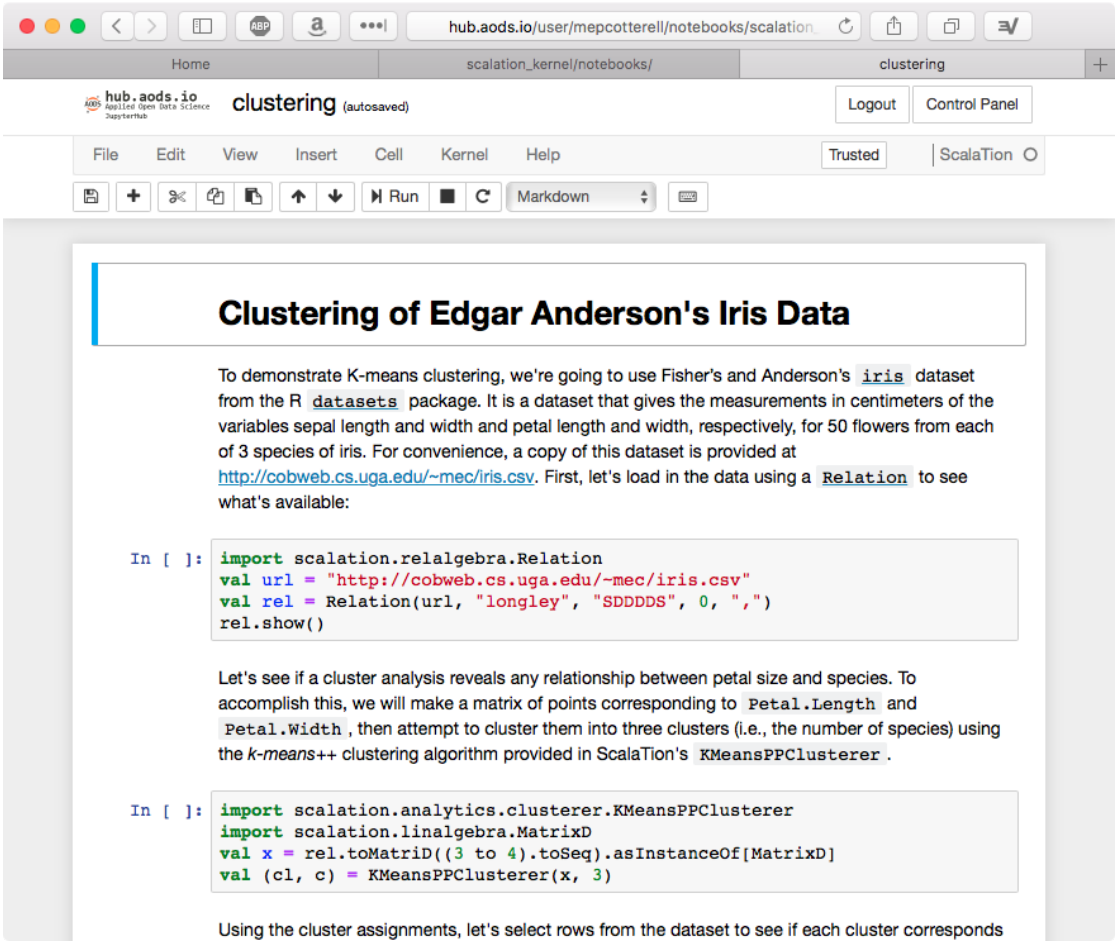


Figure 6.3: Clustering of Edgar Anderson's Iris Data Notebook

Here we see an example of a Jupyter ScalaTion notebook which illustrates how to use K-means clustering on Fisher's and Anderson's iris dataset [Becker et al., 1988] to investigate the relationship between iris petal length and species.

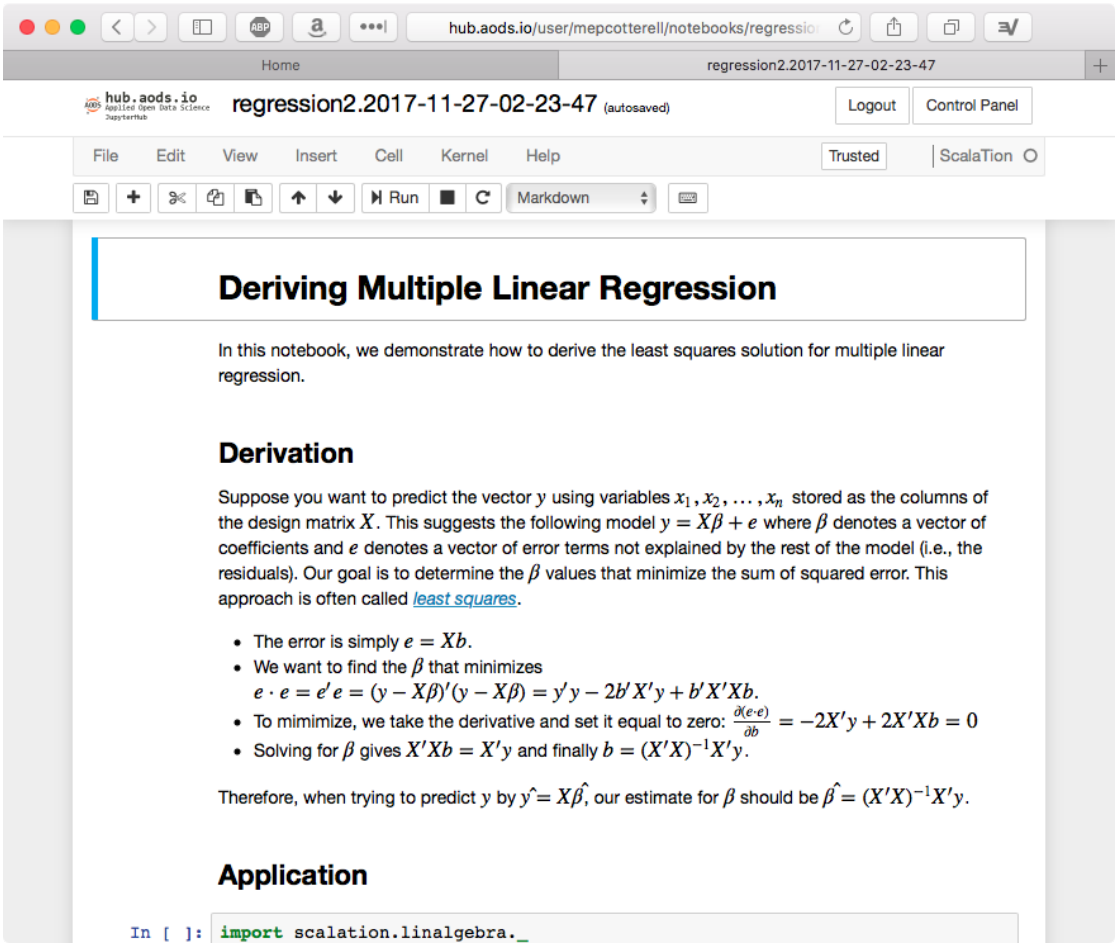


6.4 Impact

The AODS project and example notebooks showcase a method for disseminating data science investigations and lessons that make use of a big data framework in a

Figure 6.4: Deriving Multiple Linear Regression Notebook

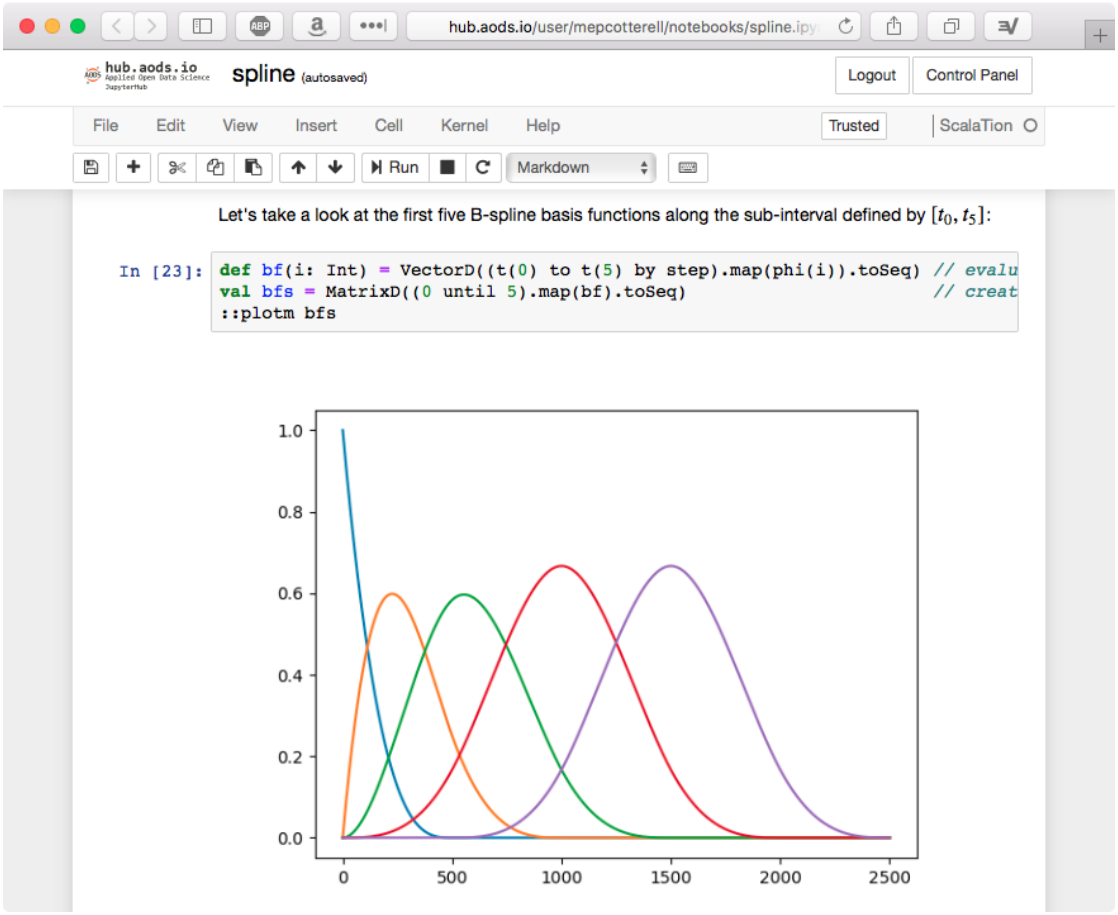
Here we see an example of a Jupyter ScalaTion notebook which illustrates how to derive and apply the least squares solution for multiple linear regression.



way that is both relatively easy for investigators to provide and convenient to users wishing to replicate investigation results or follow along with lessons. Specifically, the AODS JupyterHub project presents a documented, minimal approach, with

Figure 6.5: Regression Splines Notebook

Here we see an example of a Jupyter ScalaTion notebook which illustrates how to approximate a function for the number (in thousands) of Australian residents over time from the `austres` dataset [Brockwell and Davis, 2016] using a regression spline.



an accompanying example deployment, that allows data science investigators and educators to deploy a JupyterHub installation that supports a big data framework. Furthermore, the AODS Upload project presents a documented, minimal way to

make this setup even more appealing and easier to use for end users by providing an effective way for them to use links to automatically upload notebooks directly to a supported JupyterHub workspace. Readers can confirm this effectiveness using the "upload" links provided for each example notebook in Section 6.3.

6.5 Conclusions

In this chapter, we presented the Applied Open Data Science (AODS) project as well as a set of example notebooks that utilize the ScalaTion big data framework. Housed at `aods.io`, the AODS JupyterHub and AODS Upload projects demonstrated how open and accessible support for open notebooks that make use of big data frameworks can be provided. The implementations of these projects served as a live test-bed for readers of this dissertation to explore the various example notebooks that were provided.

Future work for the AODS project includes iterating on existing documentation so that it is easier to follow and supports different infrastructure scenarios. Additionally, more example notebooks will be created to further demonstrate the ideas conveyed in this dissertation.

Chapter 7

Summary

The work in this dissertation is motivated by the following open research questions in data science education: How do you provide tools to support open science, big data, and reproducibility? What computing infrastructure is available and how do you use it? How do you make it easier for students and domain experts to do data science and big data analytics? To that end, this work described and exemplified how big data frameworks and open science can be combined to help mitigate some of the problems related to these questions.

The main contributions of this research can be summarized as follows:

- We provided an overview of the open source ScalaTion project, a big data framework that supports big data analytics and simulation modeling. We present ScalaTion as an excellent tool for making big data analytics more approachable by allowing users to express, execute, and connect together models that are more concise and readable through well documented code

and the use of a domain-specific language. We also presented examples of how ScalaTion helps support previous and existing research in analytics.

- To demonstrate how to provide lightweight big data framework integration in open notebooks, we presented the open source ScalaTion Kernel project, a custom Jupyter kernel that enables ScalaTion support in Jupyter notebooks. We also discuss how the project's minimal integration approach makes it easier for others to reproduce Jupyter installations that include the kernel. This was demonstrated through the provision of installation instructions for multiple scenarios, including deployment in a Python virtual environment and as a containerized application.

- To demonstrate research ScalaTion is used in, we outlined and evaluated a tight clustering algorithm, written using ScalaTion, for the functional data analysis of time course omics data. Across different signal to noise ratios and proportions of simulated scattered genes, our method showed improvement in terms of the weighted Rand index metric.

- To promote reproducibility in open science, we presented the Applied Open Data Science (AODS) project, a collection of customized web applications for the hosting and sharing of open notebooks with ScalaTion support. We demonstrate the merit of this project by providing shareable, executable, and modifiable example notebooks, directly accessible to readers of this dissertation, that utilize ScalaTion to demonstrate various data science topics.

In conclusion, the research described and discussions presented in this dissertation

demonstrate potential solutions to some of the motivating questions posed above. As demonstrated and discussed through various, live examples, these methods would enable data science investigators and educators to provide a dissemination environment that is reproducible at the level of individual investigations and at the level of investigative infrastructure.

Future work related to this dissertation includes the submission of two grant proposals related to advanced cyberinfrastructure education (one for course and outreach development; the other for the development of a departmental container cluster) and the continued development of the ScalaTion Kernel and AODS projects.

Appendix A

Proposed Cyberinfrastructure Courses

This appendix includes detailed information on proposed data science courses that integrate the use of advanced cyberinfrastructure (CI). Much of the material here is from a grant proposal principally authored by the dissertation author. This appendix is organized as follows: i) an outline of the two proposed courses are provided in Sections A.1 and A.2; and ii) pedagogical details are provided in Section A.3

A.1 CI for Data Science I (CI1)

A.1.1 Course Description

CI includes technologies that support data science within a highly inter-operable and collaborative ecosystem. Usually utilizing a "flipped" pedagogical approach (described later), students in this course learn about the motivations and uses of CI for data science-driven investigations via tutorials and an applied term project.

- This course emphasizes "Computation and Data Science for All" and is, therefore, targeted at the broader STEM+C student community at a university. Students should have experience with pre-calculus and statistics. While some data science and programming experience is recommended, it is not required. The topical outline for this course (included below) includes a number of broad topics designed to better prepare students for infrastructure-specific training opportunities and should raise awareness of advanced CI and its applications to applied data science. Actual usage of advanced CI will be incorporated into coursework. Emphasis will also be placed on open data science.

A.1.2 Topical Outline (estimated 37.5 contact hours)

- Computation and Data Science for All
  - Introduction / History (1.00 hours)
  - Third Pillar: Computation (1.25 hours)

  - Fourth Pillar: Data-Driven Science (1.25 hours)
  - Applications / Examples (1.00 hours)
  - Ethics of Data Science (1.00 hours)
- Cyberinfrastructure (CI)
  - Introduction / History (1.00 hours)
  - Big Data (1.00 hours)
  - Hardware Survey (1.00 hours)
  - Software Survey (1.00 hours)
  - Community Engagement (1.00 hours)
  - Open Science Big Data (1.00 hours)
  - Privacy and Security Considerations (1.00 hours)
- Basic Command Scripting
  - Bash (1.25 hours)
  - Python (1.25 hours)
  - Scala (1.25 hours)
- Basic Command Scripting
  - Introduction / History (1.25 hours)
  - Map Reduce Paradigm (1.25 hours)
- Libraries for Distributed Computing

  - Hadoop (1.25 hours)
  - Spark (1.25 hours)
  - Flink (1.25 hours)
  - Others (1.25 hours)
- Collaboration
  - Version Control Basics (1.25 hours)
  - Team and Project Management (1.25 hours)
- Communication of Results
  - Data Visualization Methods (1.00 hours)
  - Technical Writing (1.00 hours)
- Term Project
  - Proposal (2.00 hours)
  - Milestone Demonstration [x2] (2.50 hours)
  - Demonstration (1.00 hours)

A.2 CI for Data Science II (CI2)

A.2.1 Course Description

Usually utilizing a "flipped" pedagogical approach (described later), students in this course learn about the motivation, implementation, and use of paradigms,

algorithms, and tools for advanced CI. Students will propose, implement, evaluate, and demonstrate an applied term project emphasizing open science and the use of advanced CI.

- This course is targeted at students who have taken CI1. Students should have experience with calculus and programming. The topical outline for this course (included below) includes a number of broad topics designed to better prepare students for the implementation, execution, debugging, and optimization of data science-related algorithms on advanced CI. Actual usage of advanced CI will be incorporated into coursework. Emphasis will also be placed on open data science.

A.2.2 Topical Outline (estimated 37.5 contact hours)

- Concepts for Cyberinfrastructure
  - Open Science Big Data (e.g., Notebooks) (3.33 hours)
  - Multicore CPU Programming (e.g., IPC, Threads, Actors) (3.33 hours)
  - Multicore GPU Programming (e.g., CUDA) (3.33 hours)
  - Distributed Storage and Memory (e.g., HDFS) (3.33 hours)
  - MapReduce (e.g., Hadoop) (3.33 hours)
  - Message Passing & Distributed Actors (e.g., MPI, Akka) (3.33 hours)
  - Domain-Specific Language (e.g., Sawzall, Apache Pig, ScalaTion) (3.33 hours)

  - Distributed Streams (e.g., Spark, Apache Flink) (3.33 hours)

- Performance and Profiling
  - Performance Metrics (1.11 hours)
  - Fault Tolerance (1.11 hours)
  - Profiling (1.11 hours)

- Communication of Results
  - Data Visualization Methods (1.00 hours)
  - Technical Writing (1.00 hours)

- Term Project
  - Proposal (2.00 hours)
  - Milestone Demonstration [x2] (2.50 hours)
  - Demonstration (1.00 hours)

A.3 Pedagogy and Additional Details

Both courses will be designed with the "flipped" pedagogical approach in mind. With this approach, the typical lecture and assignment elements of the course are reversed, emphasizing experiential and active learning Heyborne and Perrett [2016]. Required readings and short video lectures are viewed by students at home before the class session, while in-class time is devoted to active learning activities. The instructor serves a mentor for the students. Recently, many colleges

and universities have funded and developed programs involving flipped pedagogy. For example, the PIs' home institution, the University of Georgia, is currently engaged in multiple flipped pedagogy programs in the Departments of Biochemistry & Molecular Biology, Chemistry, Computer Science, Genetics, Physics and Astronomy, and Plant Biology. Preliminary evaluation suggests that students in the flipped versions of these courses outperform their non-flipped counterparts, on average, with respect to achieving expected learning outcomes. Examples of flipped pedagogy programs at other institutions include Tune et al. [2013]; Schultz et al. [2014]; Baepler et al. [2014]; Kong [2014]. Emphasis will also be placed on experiential and service learning, where instruction emphasizes practical experience in the material being taught instead of just theory Krusche et al. [2017]; Derbinsky and Suresh [2017]. The PIs will work closely with UGA's Office of the Vice President for Instruction in order to follow best practices and integrate experiential and service learning into the pilot courses. Experiential learning is a high priority objective at the institution level at UGA as well as in the Computer Science department. Specific avenues for experiential and service learning are outlined later in this proposal.

In order to help promote open data science and active, guided learning, most of the interactive tutorials in either course will be provided in an open notebook format such as Jupyter Ragan-Kelley et al. [2014]. For example, with Jupyter notebooks, the descriptions, methods, relevant code snippets, and student notes for an assignment or lecture are all integrated in an easy to read and easy to run computational narrative. With open notebooks, the course lectures, readings,

and activities will be provided in a way that is easily accessible and to the point, facilitating investigation and reproducibility. Students can follow along with each activity, modify and execute relevant code snippets, and incorporate their own notes and results all within a notebook. A sample activity for each course is briefly described below:

- **CI1 Sample Activity - Big Data Sample Statistics:** This activity guides students through the collection of sample statistics (e.g., mean, variance, mode, min, max, etc.) for some big data dataset (e.g., in the petabyte regime). An overview of the challenges faced in the collection of these sample statistics is presented, including specific examples that students can try out to see for themselves. Students are then asked to leverage advanced CI as well as some of the open frameworks discussed in class to collect and present the sample statistics.

- **CI2 Sample Activity - Gene Expression Clustering:** This activity guides students through the clustering of timecourse omics data for gene expression levels collected from an RNA-seq experiment. An overview of the different clustering techniques is presented, including starter code, so that students can rapidly explore the differences between those techniques. Students are then asked to optimize the running time for a clustering algorithm (e.g., simple k-means) by extending it for multiple cores and then by making it distributed. Use of advanced CI is required.

To make these activities more approachable, the use of domain-specific language and open frameworks for analytics will be used. For any particular activity, the

activity description, starter code, as well as student modifications, results, and notes will be collected in an open notebook format for easy access, assessment, and reflection. The use of open notebooks like Jupyter have been shown to be effective in Hu et al. [2017] as well as in the PIs' home institution in some of classes instructed by one of the Co-PIs.

When these courses are offered offline, active learning environments such as computer labs and SCALE-UP Beichner et al. [2007] will be be used. For example, with SCALE-UP the physical classroom environment is structured in a studio-like fashion in order to facilitate active, collaborative learning. Additionally, we plan to complement offline learning through the use of Peer Learning Assistants (PLAs). Depending on the level of the course offering, PLAs are undergraduate or graduate teaching assistants who receive explicit pedagogical training either prior to or during their assistantship. For example, the University of Georgia requires PLAs to take FCID 3100, a course that, according to the UGA Bulletin, "introduces current research findings on how people learn, reviews proven strategies for engaging undergraduates in active learning in introductory STEM courses, and offers opportunities to model effective teaching practices with in-class group activities." PLA programs have been shown to be successful at many universities Blackwell et al. [2017], including the University of Colorado Otero et al. [2010] and the University of Georgia (the PIs' home institution).

As the courses will be designed with the flipped pedagogy in mind, development of online versions should be achievable with minimal effort. The same materials provided for the offline version of these course will also be offered to the online

students, leaving online course development to the distribution and management of the active learning activities in an online environment. The PIs will work closely with UGA's Office of Online Learning to develop the online versions of these courses. Different strategies for effective online collaboration will be explored.

Both courses will involve term projects that are designed to increase net collective impact through experiential and service learning of the material. Students will propose, implement, evaluate, and demonstrate an applied term project emphasizing applied data science and the use of advanced CI. Only proposals that include a collaboration with a university research lab and or local community stakeholder will be accepted. Furthermore, projects should make use of of open science tools like Spark and ScalaTion. For example, ScalaTion is a free and open source framework for exploring a modeling continuum that includes analytics, simulation and optimization Miller et al. [2013b].

We anticipate that these courses will increase net collective impact by complementing the existing arsenal of training and development opportunities. Students who take these courses will be more prepared for infrastructure-specific training opportunities, allowing the trainers of those opportunities to effectively focus their efforts.

Appendix B

Proposed Community Outreach Programs

This appendix includes information on proposed community outreach programs that integrate the use of advanced cyberinfrastructure (CI) and complement the courses described in Appendix A. Much of the material here is from a grant proposal principally authored by the dissertation author. This appendix is organized as follows: i) an outline of the two proposed community outreach programs are provided in Sections B.1 and B.2; and ii) additional details are provided in Section B.3

B.1 Secondary School Program

In this program, we propose complementing new and existing high school after school, summer school, and summer camp programs by facilitating three key goals:

(i) raise awareness of data science and CI through exposure and engagement; (ii) provide means and support for organizers and participants to provide said engagement; and (iii) provide a means for assessing existing student exposure to computer science and data science.

B.2 Data Science as a Community Service Program

Here, we propose the preparation and continued development of a community service program called "DSaaCS", primarily designed to facilitate three key goals: (i) connect undergraduate students and faculty with community stakeholders who are in need; (ii) provide the tools and resources to help students aid community stakeholders through the use of applied data science and advanced CI; and (iii) raise community exposure to data science and advanced CI.

B.3 Additional Details

Specifically, this community outreach aspect of the proposal seeks to facilitate the experiential and service learning requirement for projects in the CI1 and CI2 courses (detailed earlier in this proposal). Connecting students with community stakeholders provides an effective means for experiential and service learning that often results in the increase of practical experience and collective impact for all parties involved; such connections also inform the instructors, ensuring that course

curriculum better suits the contemporary needs of stakeholders Bruhn and Camp [2004]. This aspect of the proposal also complements existing NSF programs such as STEM + Computing Partnerships (STEM+C) and Computer Science for All (CS for All), which, among other things, seek to raise awareness, interest, skill, and competency of data science in secondary schools.

How will we forge partnerships? In order to help better facilitate this community outreach aspect of the proposal, partnerships with local businesses and community leaders will be forged. Currently, we have letters of support from the following collaborators:

- **ElevateCS:** ElevateCS is an outreach program located in the Athens, GA area. Their mission is to promote, improve, and empower Computer Science/Technology education and those interested in it by providing resources, events, and workshops via immersion. The PIs will collaborate with ElevateCS for the proposed secondary school community outreach program.

- **FourAthens:** - Four Athens is a local Athens Technology Incubator that takes in young start-ups and entrepreneurs and helps them acquire everything they need to progress from inception stage to establishment of a viable company. The PIs will collaborate with Four Athens for the proposed DSaaCS program.

These partnerships are not explicitly for internship opportunities. Instead, it's a way for students to engage in experiential and service learning related to advanced CI and applied data science. Ideally, students working with these partners will

be taking one of the courses described elsewhere in this proposal concurrently. The role these partnerships will take is that of providing the connection between students and local businesses and community leaders. No management overhead would be required above facilitating these connections.

Opportunities also exist at the PIs' home institution for greater exposure of advanced CI and applied data science to high school students by allowing them to work side-by-side with the students and faculty involved in the course development and community outreach aspects of this proposal. For example, UGA's Young Dawgs program is a high school internship program designed to prepare high-achieving high school juniors and seniors for post-secondary education and future careers in their areas of interest. Students accepted into the program participate in unpaid internships on campus–in placements throughout the university–as well as in the community. In this case, students would be paired with other students and faculty engaged in the principal activities of this proposal. The program is open to students from public, private, and home schools. Students are required to have a GPA of at least 3.7 in order to participate.

We anticipate that these community outreach programs will increase net collective impact by facilitating public-private and public-public partnerships relevant to raising the awareness and education surrounding advanced CI, data science, STEM+C, and CS For All. Additionally, students who participate in these community outreach programs will be more prepared for infrastructure-specific training opportunities, allowing the trainers of those opportunities to effectively focus their efforts.

Bibliography

Alexander A Aarts et al. Estimating the reproducibility of psychological science. *Science*, 349(6251), 2015. ISSN 0036-8075. doi: 10.1126/science.aac4716. URL http://science.sciencemag.org/content/349/6251/aac4716.

Christophe Abraham, Pierre-André Cornillon, ERIC Matzner-Løber, and Nicolas Molinari. Unsupervised curve clustering using B-splines. *Scandinavian Journal of Statistics*, 30(3):581–595, 2003.

Thomas Aichner, Paolo Coletti, Cipriano Forza, Urban Perkmann, and Alessio Trentin. Effects of subcultural differences on country and product evaluations: A replication study. *Journal of Global Marketing*, 29(3):115–127, mar 2016. doi: 10.1080/08911762.2015.1138012. URL https://doi.org/10.1080/08911762.2015.1138012.

George Alter et al. Promoting an open research culture: The top guidelines for journals. 2016.

William E. Althouse, Michael E. Zucker, and et al. Ligo: The laser interferometer gravitational-wave observatory. *Science*, 256(5055):325, Apr 17

1992. URL http://proxy-remote.galib.uga.edu:80/login?url=https://search.proquest.com/docview/213538223?accountid=14537. Copyright - Copyright American Association for the Advancement of Science Apr 17, 1992; Last updated - 2014-05-30; CODEN - SCIEAS.

David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035, New Orleans, Louisiana, 2007. ISBN 9780898716245. URL http://dl.acm.org/citation.cfm?id=1283494{&}CFID=921150079{&}CFTOKEN=36982346.

Paul Baepler, JD Walker, and Michelle Driessen. It's not about seat time: Blending, flipping, and efficiency in active learning classrooms. *Computers & Education*, 78:227–236, 2014.

Monya Baker. 1,500 scientists lift the lid on reproducibility. *Nature*, 533(7604): 452–454, 2016.

Manuel Febrero Bande, Manuel Oviedo, Pedo Galeano, Alicia Nieto, and Eduardo Garcia-Portugues. *Package 'fda.usc'*. The Comprehensive R Archive Network (CRAN), 2016. URL https://stat.ethz.ch/CRAN/web/packages/fda.usc/fda.usc.pdf. https://stat.ethz.ch/CRAN/web/packages/fda.usc/fda.usc.pdf.

Richard A Becker, John M Chambers, and Allan R Wilks. The new s language. *Pacific Grove, Ca.: Wadsworth & Brooks, 1988*, 1988.

Robert J Beichner, Jeffery M Saul, David S Abbott, Jeanne J Morse, Duane Dear-

dorff, Rhett J Allain, Scott W Bonham, Melissa H Dancy, and John S Risley. The student-centered activities for large enrollment undergraduate programs (scale-up) project. *Research-based reform of university physics*, 1(1):2–39, 2007.

Richard Bellman. On the theory of dynamic programming. *Proceedings of the National Academy of Sciences*, 38(8):716–719, 1952.

Lida Beninson, Quincy Brown, Elizabeth Burrows, Dana Hunter, Craig Jolley, Meredith Lee, Nishal Mohan, Chloe Poston, Renata Rawlings-Goss, Carly Robinson, Alejandro Suarez, Martin Wiener, Fen Zhao, et al. The Federal Big Data Research and Development Strategic Plan. Technical report, Networking and Information Technology Research and Development (NITRD), 2016.

Bonnie Berger, Jian Peng, and Mona Singh. Computational solutions for omics data. *Nature Reviews Genetics*, 14(5):333–346, 2013.

David Bernstein. Containers and cloud: From lxc to docker to kubernetes. *IEEE Cloud Computing*, 1(3):81–84, 2014.

Stacey Blackwell, Sari Katzen, Nipa Patel, Sun Yan, and Mary Emenike. Developing the preparation in stem leadership programs for undergraduate academic peer leaders. *Learning Assistance Review (TLAR)*, 22(1):49 – 84, 2017. ISSN 10870059. URL `http://proxy-remote.galib.uga.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=ehh&AN=122687314&site=eds-live`.

Avrim Blum, John Hopcroft, and Ravindran Kannan. Foundations of Data Science. *Vorabversion eines Lehrbuchs*, 2016.

Casey N Bowman and John A Miller. Modeling traffic flow using simulation and big data analytics. In *Winter Simulation Conference (WSC), 2016*, pages 1206–1217. IEEE, 2016.

Danah Boyd and Kate Crawford. Six Provocations for Big Data. *SSRN Electronic Journal*, pages 1–17, sep 2011. ISSN 1556-5068. doi: 10.2139/ssrn.1926431. URL `http://www.ssrn.com/abstract=1926431`.

Sarah Brockhaus, David Ruegamer, and Torsten Hothorn. *Package 'FDboost'*. The Comprehensive R Archive Network (CRAN), 2016. URL `https://cran.r-project.org/web/packages/FDboost/FDboost.pdf`. https://cran.r-project.org/web/packages/FDboost/FDboost.pdf.

Peter J Brockwell and Richard A Davis. *Introduction to time series and forecasting*. springer, 2016.

Russel E. Bruhn and Judy Camp. Capstone course creates useful business products and corporate-ready students. *SIGCSE Bull.*, 36(2):87–92, June 2004. ISSN 0097-8418. doi: 10.1145/1024338.1024379. URL `http://doi.acm.org/10.1145/1024338.1024379`.

C. F. Camerer, A. Dreber, E. Forsell, T.-H. Ho, J. Huber, M. Johannesson, M. Kirchler, J. Almenberg, A. Altmejd, T. Chan, E. Heikensten, F. Holzmeister, T. Imai, S. Isaksson, G. Nave, T. Pfeiffer, M. Razen, and H. Wu. Evaluating

replicability of laboratory experiments in economics. *Science*, 351(6280):1433–1436, mar 2016. doi: 10.1126/science.aaf0918. URL https://doi.org/10.1126/science.aaf0918.

Longbing Cao. Data science: A comprehensive overview. *ACM Comput. Surv.*, 50(3):43:1–43:42, June 2017. ISSN 0360-0300. doi: 10.1145/3076253. URL http://doi.acm.org/10.1145/3076253.

Susan E Celniker, Laura AL Dillon, Mark B Gerstein, Kristin C Gunsalus, Steven Henikoff, Gary H Karpen, Manolis Kellis, Eric C Lai, Jason D Lieb, David M MacAlpine, et al. Unlocking the secrets of the genome. *Nature*, 459(7249): 927–930, 2009.

Soumen Chakrabarti, Martin Ester, Usama Fayyad, Johannes Gehrke, Jiawei Han, Shinichi Morishita, Gregory Piatetsky-Shapiro, and Wei Wang. Data mining curriculum: A proposal (version 1.0). *Intensive Working Group of ACM SIGKDD Curriculum Committee*, page 140, 2006.

CL Philip Chen and Chun-Yang Zhang. Data-intensive applications, challenges, techniques and technologies: A survey on big data. *Information Sciences*, 275: 314–347, 2014.

Min Chen, Shiwen Mao, and Yunhao Liu. Big data: A survey. *Mobile Networks and Applications*, 19(2):171–209, Apr 2014. ISSN 1572-8153. doi: 10.1007/s11036-013-0489-0. URL https://doi.org/10.1007/s11036-013-0489-0.

Laura Clarke, Xiangqun Zheng-Bradley, Richard Smith, Eugene Kulesha, Chunlin

Xiao, Iliana Toneva, Brendan Vaughan, Don Preuss, Rasko Leinonen, Martin Shumway, Stephen Sherry, and Paul Flicek. The 1000 genomes project: data management and community access. *Nature Methods*, 9(5):459–462, apr 2012. doi: 10.1038/nmeth.1974. URL `https://doi.org/10.1038/nmeth.1974`.

Michael E. Cotterell, John A. Miller, and Tom Horton. Unicode in Domain-Specific Programming Languages for Modeling & Simulation: ScalaTion as a Case Study, 2011a.

Michael E. Cotterell, John A. Miller, and Tom Horton. Unicode in Domain-Specific Programming Languages for Modeling & Simulation: ScalaTion as a Case Study. Technical report, Department of Computer Science, University of Georgia, Athens, Georgia, May 2011b. URL `http://arxiv.org/abs/1112.1751`.

Peter Craven and Grace Wahba. Smoothing noisy data with spline functions. *Numerische Mathematik*, 31(4):377–403, 1978. ISSN 0029-599X. doi: 10.1007/BF01404567. URL `http://link.springer.com/10.1007/BF01404567`.

Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. Social coding in github: Transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1277–1286. ACM, 2012.

Xiongtao Dai, Pantelis Z. Hadjipantelis, Hao Ji, Hans-Georg Mueller, and Jane-Ling Wang. *Package 'fdapace'*. The Comprehensive R Archive Network (CRAN),

2017. URL https://stat.ethz.ch/CRAN/web/packages/fdapace/fdapace.pdf. https://stat.ethz.ch/CRAN/web/packages/fdapace/fdapace.pdf.

Carl de Boor, J.E. Marsden, and L. Sirovich. *A Practical Guide to Splines (Revised Edition)*. Applied Mathematical Sciences 27. Springer-Verlag New York, 2001. ISBN 9780387953663. URL http://gen.lib.rus.ec/book/index.php?md5=C908D608964F2FDDB6F67FF1632DBF2F.

Nate Derbinsky and Durga Suresh. Sustainable methods for impactful service learning in computer science (abstract only). In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '17, pages 723–723, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4698-6. doi: 10.1145/3017680.3022346. URL http://doi.acm.org/10.1145/3017680.3022346.

Van A. Deursen, P. Klint, and J. Visser. Domain-Specific Languages: An Annotated Bibliography. *ACM Sigplan Notices*, 35(6):26–36, 2000. ISSN 0362-1340.

Vasant Dhar. Data Science and Prediction. *Communications of the ACM*, 56(12): 64–73, 2013.

Michael B Eisen, Paul T Spellman, Patrick O Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–14868, 1998.

Eli Eisenberg and Erez Y. Levanon. Human housekeeping genes, revisited. *Trends in Genetics*, 29(10):569–574, 2013. ISSN 01689525. doi: 10.1016/j.tig.

2013.05.010. URL http://www.sciencedirect.com/science/article/pii/S0168952513000899.

Elsevier. Research data faqs, 2017.

Leandro Fernández, Riccard Andersson, Hakan Hagenrud, Timo Korhonen, Emanuele Laface, and Bla Zupanc. Jupyterhub at the ESS. An Interactive Python Computing Environment for Scientists and Engineers. In *In Proceedings of 2016 International Particle Accelerator Conference (IPAC16)*, 2016.

John Fink. Docker: a software as a service, operating system-level virtualization framework. *Code4Lib Journal*, 25, 2014.

Steven Finlay. *Predictive analytics, data mining and big data: Myths, misconceptions and methods*. Springer, 2014.

Y Fujikoshi. Modified AIC and C_p in multivariate linear regression. *Biometrika*, 84(3):707–716, 1997. ISSN 0006-3444. doi: 10.1093/biomet/84.3.707. URL https://academic.oup.com/biomet/article-lookup/doi/10.1093/biomet/84.3.707.

Matthias E Futschik and Bronwyn Carlisle. Noise-robust soft clustering of gene expression time-course data. *Journal of Bioinformatics and Computational Biology*, 3(4):965–88, 2005. ISSN 0219-7200. URL http://www.ncbi.nlm.nih.gov/pubmed/16078370.

Ruti Gafni and Dudu Nissim. To social login or not login? exploring factors

affecting the decision. *Issues in Informing Science and Information Technology*, 11:57–72, 2014.

Isak Gath and Amir B. Geva. Unsupervised optimal fuzzy clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):773–780, 1989.

Yolanda Gil, Cédric H David, Ibrahim Demir, Bakinam T Essawy, Robinson W Fulweiler, Jonathan L Goodall, Leif Karlstrom, Huikyo Lee, Heath J Mills, Ji-Hyun Oh, et al. Toward the geoscience paper of the future: Best practices for documenting and sharing research from data to software to provenance. *Earth and Space Science*, 3(10):388–415, 2016.

Stephen Goff, Matthew Vaughn, Sheldon McKay, Eric Lyons, Ann Stapleton, Damian Gessler, Naim Matasci, Liya Wang, Matthew Hanlon, Andrew Lenards, Andy Muir, Nirav Merchant, Sonya Lowry, Stephen Mock, Matthew Helmke, Adam Kubach, Martha Narro, Nicole Hopkins, David Micklos, Uwe Hilgert, Michael Gonzales, Chris Jordan, Edwin Skidmore, Rion Dooley, John Cazes, Robert McLay, Zhenyuan Lu, Shiran Pasternak, Lars Koesterke, William Piel, Ruth Grene, Christos Noutsos, Karla Gendler, Xin Feng, Chunlao Tang, Monica Lent, Seung-jin Kim, Kristian Kvilekval, B.S. Manjunath, Val Tannen, Alexandros Stamatakis, Michael Sanderson, Stephen Welch, Karen Cranston, Pamela Soltis, Douglas Soltis, Brian O'Meara, Cecile Ane, Tom Brutnell, Daniel Kleibenstein, Jeffrey White, Jim Leebens-Mack, Michael Donoghue, Edgar Spalding, Todd Vision, Christopher Myers, David Lowenthal, Brian Enquist, Brad Boyle, Ali Akoglu, Greg Andrews, Sudha Ram, Doreen Ware, Lincoln

Stein, and Dan Stanzione. The iplant collaborative: Cyberinfrastructure for plant biology. *Frontiers in Plant Science*, 2:34, 2011. ISSN 1664-462X. doi: 10.3389/fpls.2011.00034. URL http://journal.frontiersin.org/article/10.3389/fpls.2011.00034.

Jeff Goldsmith, Fabian Scheipl, Lei Huang, Julia Wrobel, Jonathan Gellar, Jaroslaw Harezlak, Mathew W. McLean, Bruce Swihart, Luo Xiao, Ciprian Crainiceanu, Philip T. Reiss, Yakuan Chen, Sonja Greven, Lan Huo, Madan Gopal Kundu, So Young Park, David L. Miller, and Ana-Maria Staicu. *Package 'refund'*. The Comprehensive R Archive Network (CRAN), 2016. URL https://cran.r-project.org/web/packages/refund/refund.pdf. https://cran.r-project.org/web/packages/refund/refund.pdf.

Brenton R Graveley, Angela N Brooks, Joseph W Carlson, Michael O Duff, Jane M Landolin, Li Yang, Carlo G Artieri, Marijke J van Baren, Nathan Boley, Benjamin W Booth, et al. The developmental transcriptome of Drosophila melanogaster. *Nature*, 471(7339):473–479, 2011.

Clara Happ. *Package 'funData'*. The Comprehensive R Archive Network (CRAN), 2016. URL https://stat.ethz.ch/CRAN/web/packages/funData/funData.pdf. https://stat.ethz.ch/CRAN/web/packages/funData/funData.pdf.

J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979. ISSN 00359254, 14679876. URL http://www.jstor.org/stable/2346830.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York, New York, NY, 2009. ISBN 978-0-387-84857-0. doi: 10.1007/978-0-387-84858-7. URL http://link.springer.com/10.1007/978-0-387-84858-7.

Chikio Hayashi. What is data science? fundamental concepts and a heuristic example. In *Data Science, Classification, and Related Methods*, pages 40–51. Springer, 1998.

Tony Hey, Stewart Tansley, Kristin M Tolle, et al. *The Fourth Paradigm: Data-Intensive Scientific Discovery*, volume 1. Microsoft Research Redmond, WA, 2009.

William H. Heyborne and Jamis J. Perrett. To flip or not to flip? analysis of a flipped classroom pedagogy in a general biology course. *Journal of College Science Teaching*, 45(4):31–37, Mar 2016. URL http://proxy-remote.galib.uga.edu:80/login?url=https://search.proquest.com/docview/1769716240?accountid=14537. Copyright - Copyright National Science Teachers Association Mar/Apr 2016; Document feature - Tables; Graphs; ; Last updated - 2016-03-02; CODEN - JSCTBN.

C. Hofer, K. Ostermann, T. Rendel, and A. Moors. Polymorphic Embedding of DSLs. In *Proceedings of the 7th International Conference on Generative Programming and Component Engineering*, GPCE '08, pages 137–148. ACM, 2008.

Christian Hofer and Klaus Ostermann. Modular Domain-specific Language Components in Scala. In *Proceedings of the 9th International Conference on Generative Programming and Component Engineering*, GPCE '10, pages 83–92. ACM, 2010. ISBN 978-1-4503-0154-1. doi: http://doi.acm.org/10.1145/1868294.1868307. URL `http://doi.acm.org/10.1145/1868294.1868307`.

Helen H Hu, Douglas Blank, Albert Chan, and Travis Doom. Panel: Teaching to increase diversity and equity in stem. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, pages 665–666. ACM, 2017.

Ross Ihaka and Robert Gentleman. R: A Language for Data Analysis and Graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996.

John P. A. Ioannidis. Why most clinical research is not useful. *PLOS Medicine*, 13(6):e1002049, jun 2016. doi: 10.1371/journal.pmed.1002049. URL `https://doi.org/10.1371/journal.pmed.1002049`.

Julien Jacques and Cristian Preda. Functional data clustering: a survey. *Advances in Data Analysis and Classification*, 8(3):231–255, 2014.

Stephen Kaisler, Frank Armour, J Alberto Espinosa, and William Money. Big data: Issues and challenges moving forward. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, pages 995–1004. IEEE, 2013.

Takeaki Kariya and Hiroshi Kurata. *Generalized Least Squares*. John Wiley & Sons, 2004.

Avita Katal, Mohammad Wazid, and RH Goudar. Big data: issues, challenges, tools and good practices. In *Contemporary Computing (IC3), 2013 Sixth International Conference on*, pages 404–409. IEEE, 2013.

Leonard Kaufman and Peter J Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*, volume 344. John Wiley & Sons, 2009.

Nawsher Khan, Ibrar Yaqoob, Ibrahim Abaker Targio Hashem, Zakira Inayat, Waleed Kamaleldin Mahmoud Ali, Muhammad Alam, Muhammad Shiraz, and Abdullah Gani. Big data: survey, technologies, opportunities, and challenges. *The Scientific World Journal*, 2014, 2014.

J. Kiefer, J. Kiefer, J. Wolfowitz, Herbert Robbins, Sutton Monro, and J. Wolfowitz. Sequential minimax search for a maximum. *Proceedings of the American Mathematical Society*, 4(3):502–502, 1953. ISSN 0002-9939. doi: 10.1090/S0002-9939-1953-0055639-3. URL `http://www.ams.org/jourcgi/jour-getitem?pii=S0002-9939-1953-0055639-3`.

Gerhard Klimeck, Michael McLennan, Sean P. Brophy, George B. Adams III, and Mark S. Lundstrom. nanohub.org: Advancing education and research in nanotechnology. *Computing in Science & Engineering*, 10(5):17–23, 2008. doi: 10.1109/MCSE.2008.120. URL `http://aip.scitation.org/doi/abs/10.1109/MCSE.2008.120`.

Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian E Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica B Hamrick, Ja-

son Grout, Sylvain Corlay, et al. Jupyter notebooks-a publishing format for reproducible computational workflows. In *ELPUB*, pages 87–90, 2016.

Ron Kohavi and Foster Provost. Glossary of terms. *Machine Learning*, 30(2-3): 271–274, 1998.

Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

Siu Cheung Kong. Developing information literacy and critical thinking skills through domain knowledge learning in digital classrooms: An experience of practicing flipped classroom strategy. *Computers & Education*, 78:160–173, 2014.

Sadanori Konishi, Tomohiro Ando, and Seiya Imoto. Bayesian information criteria and smoothing parameter selection in radial basis function networks. *Biometrika*, 91(1):27–43, 2004. URL `http://www.jstor.org/stable/20441077?seq=1{#}page{_}scan{_}tab{_}contents`.

Sadanori Konoshi and Genshiro Kitagawa. Generalised information criteria in model selection. *Biometrika*, 83(4):875–890, 1996. URL `https://www.jstor.org/stable/2337290?seq=1{#}page{_}scan{_}tab{_}contents`.

Stephan Krusche, Andreas Seitz, Nadine von Frankenberg, and Bernd Bruegge. How to integrate interactive learning into large classes (abstract only). In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '17, pages 737–737, New York, NY, USA, 2017.

ACM. ISBN 978-1-4503-4698-6. doi: 10.1145/3017680.3017843. URL http://doi.acm.org/10.1145/3017680.3017843.

Mikael Laakso, Patrik Welling, Helena Bukvova, Linus Nyman, Bo-Christer Björk, and Turid Hedlund. The development of open access journal publishing from 1993 to 2009. *PloS one*, 6(6):e20961, 2011.

Jingyi Jessica Li, Haiyan Huang, Peter J Bickel, and Steven E Brenner. Comparison of D. melanogaster and C. elegans developmental stages, tissues, and cells by modENCODE RNA-seq data. *Genome Research*, 24(7):1086–1101, 2014.

A. Lichman. UCI Machine Learning Repository, 2013. URL http://archive.ics.uci.edu/ml. University of California, Irvine, School of Information and Computer Sciences.

James W Longley. An appraisal of least squares programs for the electronic computer from the point of view of the user. *Journal of the American Statistical association*, 62(319):819–841, 1967.

P. Ma, Cristian I. Castillo-Davis, Wenxuan Zhong, and Jun S. Liu. A data-driven clustering method for time course gene expression data. *Nucleic Acids Research*, 34(4):1261–1269, 2006. ISSN 0305-1048. doi: 10.1093/nar/gkl013. URL https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkl013.

Ping Ma and Wenxuan Zhong. Penalized clustering of large-scale functional data with multiple covariates. *Journal of the American Statistical Association*, 103 (482):625–636, 2008.

James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on Mathematical Statistics and Probability*, pages 281–297. Oakland, CA, USA., 1967.

Youness Madani, Jemaa Bengourram, and Mohammed Erritali. Social login and data storage in the big data file system hdfs. In *Proceedings of the International Conference on Compute and Data Analysis*, ICCDA '17, pages 91–97, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5241-3. doi: 10.1145/3093241.3093265. URL `http://doi.acm.org/10.1145/3093241.3093265`.

Vivien Marx. Biology: The big challenges of big data. *Nature*, 498(7453):255–260, 2013.

G J McLachlan, R W Bean, and D Peel. A mixture model-based approach to the clustering of microarray expression data. *Bioinformatics*, 18(3):413–422, 2002. ISSN 1367-4803. URL `http://dx.doi.org/10.1093/bioinformatics/18.3.413`.

Dirk Merkel. Docker: lightweight linux containers for consistent development and deployment. *Linux Journal*, 2014(239):2, 2014.

John A Miller, Rajesh S Nair, Zhiwei Zhang, and Hongwei Zhao. JSIM: A Java-based Simulation and Animation Environment. In *Proceedings of the 1997 Simulation Symposium*, pages 31–42. IEEE, 1997.

John A. Miller, Jun Han, and Maria Hybinette. Using domain specific language for modeling and simulation: ScalaTion as a case study. In *Proceedings of*

the 2010 Winter Simulation Conference, pages 741–752, Baltimore, MD, USA, 2010. ISBN 978-1-4244-9866-6. doi: 10.1109/WSC.2010.5679113. URL `http://ieeexplore.ieee.org/document/5679113/`.

John A. Miller, Michael E. Cotterell, and Stephen J. Buckley. Supporting a Modeling Continuum in ScalaTion: From Predictive Analytics to Simulation Modeling. In *2013 Winter Simulations Conference (WSC)*, pages 1191–1202, Washington, DC, dec 2013a. IEEE. ISBN 978-1-4799-3950-3. doi: 10.1109/WSC.2013.6721507. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6721507`.

John A Miller, Michael E Cotterell, and Stephen J Buckley. Supporting a modeling continuum in scalation: From predictive analytics to simulation modeling. In *2013 Winter Simulation Conference (WSC'13)*, pages 1191–1202. IEEE, 2013b.

Michael Milligan. Interactive hpc gateways with jupyter and jupyterhub. In *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact*, page 63. ACM, 2017.

Paolo Missier, Khalid Belhajjame, and James Cheney. The w3c prov family of specifications for modelling provenance metadata. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 773–776. ACM, 2013.

Joseph Muscat. *Functional Analysis: An Introduction to Metric Spaces, Hilbert Spaces, and Banach Algebras*. Springer International Publishing, Cham, 2014.

ISBN 978-3-319-06727-8. doi: 10.1007/978-3-319-06728-5. URL http://link.springer.com/10.1007/978-3-319-06728-5.

National Academies of Sciences, Engineering, and Medicine. *Future Directions for NSF Advanced Computing Infrastructure to Support U.S. Science and Engineering in 2017-2020*. National Academies Press, 2016.

National Science Foundation. *Public Access Plan: Today's Data, Tomorrow's Discoveries: Increasing Access to the Results of Research Funded by the National Science Foundation*. National Science Foundation, Mar 2015. URL https://www.nsf.gov/publications/pub_summ.jsp?ods_key=nsf15052.

Nature. *Science in the Petabyte Era*, volume 455. Nature Publishing Group, sep 2008. URL http://dx.doi.org/10.1038/455001a.

David Nicholas, Paul Huntington, and Ian Rowlands. Open access journal publishing: the views of some of the world's senior authors. *Journal of Documentation*, 61(4):497–519, 2005.

Brian A Nosek, George Alter, George Banks, Denny Borsboom, Sara Bowman, Steven Breckler, Stuart Buck, Colin Camerer, Chris Chambers, Gilbert Chin, et al. Transparency and openness promotion (top) guidelines. 2016.

Mustafa V Nural, Michael E Cotterell, and John A Miller. Using semantics in predictive big data analytics. In *Big Data (BigData Congress), 2015 IEEE International Congress on*, pages 254–261. IEEE, 2015.

Mustafa V Nural, Hao Peng, and John A Miller. Using meta-learning for model type selection in predictive big data analytic. In *Proceedings of the 2017 IEEE International Conference on Big Data, Special Session on Intelligent Data Mining*. IEEE, 2017.

Barack Obama. Executive orderâĂŤcreating a national strategic computing initiative. *The White House, US*, 29, 2015.

Valerie Otero, Steven Pollock, and Noah Finkelstein. A physics departmentâĂŹs role in preparing physics teachers: The colorado learning assistant model. *American Journal of Physics*, 78(11):1218–1224, 2010.

Oxford English Dictionary. *big data*. Oxford University Press, 2017. URL `http://www.oed.com/view/Entry/18833?redirectedFrom=big+data`.

Claus Pahl and Brian Lee. Containers and clusters for edge cloud architectures–a technology review. In *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*, pages 379–386. IEEE, 2015.

Nicholas M. Patrikalakis and Takashi Maekawa. B-spline Curves and Surfaces. In *Shape Interrogation for Computer Aided Design and Manufacturing*, chapter 1.4. Springer, Berlin, Heidelberg, Hyperbook edition, 2010. ISBN 978-3-642-04073-3. doi: 10.1007/978-3-642-04074-0. URL `http://link.springer.com/10.1007/978-3-642-04074-0http://web.mit.edu/hyperbook/Patrikalakis-Maekawa-Cho/node15.html`.

Hao Peng, Zhe Jin, and John A Miller. Bayesian networks with structural restric-

tions: Parallelization, performance, and efficient cross-validation. In *Big Data (BigData Congress), 2017 IEEE International Congress on*, pages 7–14. IEEE, 2017.

Les Piegl and Wayne Tiller. *The NURBS Book*. Monographs in Visual Communication. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997. ISBN 978-3-540-61545-3. doi: 10.1007/978-3-642-59223-2. URL http://link.springer.com/10.1007/978-3-642-59223-2.

Foster Provost and Tom Fawcett. Data science and its relationship to big data and data-driven decision making. *Big Data*, 1(1):51–59, 2013.

Jean Francois Puget. The Most Popular Language For Machine Learning Is ... *IBM developerWorks Blogs*, dec 2016. URL https://www.ibm.com/developerworks/community/blogs/jfp/entry/What_Language_Is_Best_For_Machine_Learning_And_Data_Science?lang=en.

M Ragan-Kelley, F Perez, B Granger, T Kluyver, P Ivanov, J Frederic, and M Bussonnier. The jupyter/ipython architecture: a unified view of computational research, from interactive exploration to communication and publication. In *AGU Fall Meeting Abstracts*, 2014.

J. O. Ramsay and C. J. Dalzell. Some Tools for Functional Data Analysis on JSTOR. *Journal of the Royal Statistical Society. Series B (Methodological)*, 53 (3):539–572, 1991. URL http://www.jstor.org/stable/2345586.

J. O. Ramsay and B. W. Silverman. *Functional Data Analysis*. Springer, 1997. ISBN 9781475771077.

J. O. Ramsay and B. W. Silverman. *Functional Data Analysis*. Springer Series in Statistics. Springer-Verlag, New York, 2nd edition, 2005. ISBN 0-387-40080-X. doi: 10.1007/b98888. URL http://link.springer.com/10.1007/b98888http://link.springer.com.proxy-remote.galib.uga.edu/book/10.1007/b98888.

J. O. Ramsay, Hadley Wickham, Spencer Graves, and Giles Hooker. *Package 'fda'*. The Comprehensive R Archive Network (CRAN), 2015. URL https://cran.r-project.org/web/packages/fda/fda.pdf. https://cran.r-project.org/web/packages/fda/fda.pdf.

James Ramsay, Giles Hooker, and Spencer Graves. *Functional Data Analysis with R and MATLAB*. Use R. Springer New York, New York, NY, 2009. ISBN 978-0-387-98184-0. doi: 10.1007/978-0-387-98185-7. URL http://link.springer.com/10.1007/978-0-387-98185-7.

Ellen M. Rathje, Clint Dawson, Jamie E. Padgett, Jean-Paul Pinelli, Dan Stanzione, Ashley Adair, Pedro Arduino, Scott J. Brandenberg, Tim Cockerill, Charlie Dey, Maria Esteva, Fred L. Haan, Matthew Hanlon, Ahsan Kareem, Laura Lowes, Stephen Mock, and Gilberto Mosqueda. Designsafe: New cyberinfrastructure for natural hazards engineering. *Natural Hazards Review*, 18 (3):06017001, 2017. doi: 10.1061/(ASCE)NH.1527-6996.0000246. URL http://ascelibrary.org/doi/abs/10.1061/%28ASCE%29NH.1527-6996.0000246.

Katherine Sanderson. Data on display. *Nature*, 455(7211):273–274, 2008.

David Schultz, Stacy Duffield, Seth C Rasmussen, and Justin Wageman. Effects of the flipped classroom model on student performance for advanced placement high school chemistry students. *Journal of chemical education*, 91(9):1334–1339, 2014.

Charles Scott, Devin Wynne, and Chutima Boonthum-Denecke. Examining the privacy of login credentials using web-based single sign-on - are we giving up security and privacy for convenience? In *2016 Cybersecurity Symposium (CYBERSEC)*. IEEE, apr 2016. doi: 10.1109/cybersec.2016.019. URL `https://doi.org/10.1109/cybersec.2016.019`.

Han Lin Shang and Rob J Hyndman. *Package 'fds'*. The Comprehensive R Archive Network (CRAN), 2013. URL `https://stat.ethz.ch/CRAN/web/packages/fds/fds.pdf`. https://stat.ethz.ch/CRAN/web/packages/fds/fds.pdf.

Han Lin Shang and Rob J Hyndman. *Package 'roahd'*. The Comprehensive R Archive Network (CRAN), 2016. URL `https://stat.ethz.ch/CRAN/web/packages/rainbow/rainbow.pdf`. https://stat.ethz.ch/CRAN/web/packages/rainbow/rainbow.pdf.

Helen Shen. Interactive notebooks: Sharing the code. *Nature*, 515(7525):151, 2014.

Aisha Siddiqa, Ahmad Karim, and Abdullah Gani. Big data storage technologies: a survey. *Frontiers of Information Technology & Electronic Engineering*, 18(8): 1040–1070, 2017.

PSG Aruna Sri and M Anusha. Big data-survey. *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, 4(1):74–80, 2016.

Anuj Srivastava, Wei Wu, Sebastian Kurtek, Eric Klassen, and J S Marron. Registration of functional data using Fisher-Rao metric. *arXiv preprint arXiv:1103.3817*, 2011.

Steve Lohr. Searching for Origins Of the Term 'Big Data', feb 2013. ISSN 03624331.

Nicholas Tarabelloni, Ana Arribas-Gil, Francesca Ieva, Anna Maria Paganoni, and Juan Romo. *Package 'rainbow'*. The Comprehensive R Archive Network (CRAN), 2017. URL `https://stat.ethz.ch/CRAN/web/packages/roahd/roahd.pdf`. https://stat.ethz.ch/CRAN/web/packages/roahd/roahd.pdf.

Thaddeus Tarpey and J Kimberly K Kinateder. Clustering Functional Data. *Journal of Classification*, 20(1):93–114, 2003. ISSN 1432-1343. doi: 10.1007/s00357-003-0007-3. URL `http://dx.doi.org/10.1007/s00357-003-0007-3`.

Firat Tekiner and John A Keane. Big data framework. In *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, pages 1494–1499. IEEE, 2013.

Anbupalam Thalamuthu, Indranil Mukhopadhyay, Xiaojing Zheng, and George C Tseng. Evaluation and comparison of gene clustering methods in microarray analysis. *Bioinformatics*, 22(19):2405–2412, 2006.

Robert Tibshirani and Guenther Walther. Cluster validation by prediction

strength. *Journal of Computational and Graphical Statistics*, 14(3):511–528, 2005.

John Towns, Timothy Cockerill, Maytal Dahan, Ian Foster, Kelly Gaither, Andrew Grimshaw, Victor Hazlewood, Scott Lathrop, Dave Lifka, Gregory D. Peterson, Ralph Roskies, J. Ray Scott, and Nancy Wilkins-Diehr. Xsede: Accelerating scientific discovery. *Computing in Science & Engineering*, 16(5):62–74, 2014. doi: 10.1109/MCSE.2014.80. URL `http://aip.scitation.org/doi/abs/10.1109/MCSE.2014.80`.

George C. Tseng and Wing H. Wong. Tight Clustering: A Resampling-Based Approach for Identifying Stable and Tight Patterns in Data. *Biometrics*, 61(1): 10–16, mar 2005. ISSN 0006-341X. doi: 10.1111/j.0006-341X.2005.031032.x. URL `http://doi.wiley.com/10.1111/j.0006-341X.2005.031032.x`.

J Derek Tucker. *Package 'fdasrsf'*. The Python Package Index (PyPI), 2013. https://pypi.python.org/pypi/fdasrsf/1.0.1.

J. Derek Tucker. *Package 'fdasrvf'*. The Comprehensive R Archive Network (CRAN), 2017. URL `https://stat.ethz.ch/CRAN/web/packages/fdasrvf/fdasrvf.pdf`. https://stat.ethz.ch/CRAN/web/packages/fdasrvf/fdasrvf.pdf.

J Derek Tucker, Wei Wu, and Anuj Srivastava. Generative models for functional data using phase and amplitude separation. *Computational Statistics & Data Analysis*, 61:50–66, 2013.

Johnathan D Tune, Michael Sturek, and David P Basile. Flipped classroom model

improves graduate student performance in cardiovascular, respiratory, and renal physiology. *Advances in physiology education*, 37(4):316–320, 2013.

Guido Van Rossum and Fred L Drake. *The python language reference manual*. Network Theory Ltd., 2011.

Vinod Kumar Vavilapalli, Arun C Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, et al. Apache hadoop yarn: Yet another resource negotiator. In *Proceedings of the 4th annual Symposium on Cloud Computing*, page 5. ACM, 2013.

Matthew A Waller and Stanley E Fawcett. Data science, predictive analytics, and big data: a revolution that will transform supply chain design and management. *Journal of Business Logistics*, 34(2):77–84, 2013.

Yandong Wang, Robin Goldstone, Weikuan Yu, and Teng Wang. Characterization and optimization of memory-resident MapReduce on HPC systems. In *2014 IEEE 28th International Parallel and Distributed Processing Symposium*. IEEE, may 2014. doi: 10.1109/ipdps.2014.87. URL `https://doi.org/10.1109/ipdps.2014.87`.

Jonathan Stuart Ward and Adam Barker. Undefined by data: a survey of big data definitions. *arXiv preprint arXiv:1309.5821*, 2013.

Michel Willem. *Functional Analysis: Fundamentals and Applications*. Cornerstones. Springer New York, New York, NY, 2013. ISBN 978-1-4614-7003-8.

doi: 10.1007/978-1-4614-7004-5. URL http://link.springer.com/10.1007/978-1-4614-7004-5.

Fang Yao, Hans-Georg Müller, and Jane-Ling Wang. Functional data analysis for sparse longitudinal data. *Journal of the American Statistical Association*, 100 (470):577–590, 2005.

Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. *HotCloud*, 10(10-10):95, 2010.