Interactive visualization for missing values, time series, and areal data

Xiaoyue Cheng
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/etd

Part of the Statistics and Probability Commons

Recommended Citation
Cheng, Xiaoyue, "Interactive visualization for missing values, time series, and areal data" (2015). Graduate Theses and Dissertations. 14661.
https://lib.dr.iastate.edu/etd/14661

This Dissertation is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
Interactive visualization for missing values, time series, and areal data

by

Xiaoyue Cheng

A dissertation submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Statistics

Program of Study Committee:
Dianne Cook, Co-Major Professor
Heike Hofmann, Co-Major Professor
Amy Froelich
Zhengyuan Zhu
Alan Wanamaker

Iowa State University
Ames, Iowa
2015

Copyright © Xiaoyue Cheng, 2015. All rights reserved.
DEDICATION

To my son Sawyer, who came to the world after the writing of this work.
TABLE OF CONTENTS

LIST OF TABLES ................................................................. vii
LIST OF FIGURES ................................................................. viii
ACKNOWLEDGEMENTS ......................................................... xi
ABSTRACT ................................................................. xii

CHAPTER 1. INTRODUCTION ...................................................... 1
  1.1 Interactive graphics .................................................. 2
  1.2 Missing data exploration ............................................. 2
  1.3 Time-related interactions ........................................... 3
  1.4 Cartogram evaluation ................................................. 4

CHAPTER 2. VISUALLY EXPLORING MISSING VALUES IN MULTI-VARIABLE DATA USING A GRAPHICAL USER INTERFACE ................. 5
  2.1 Introduction ............................................................ 6
  2.2 Functionality ............................................................. 8
    2.2.1 Overview of the missing data GUI .............................. 8
    2.2.2 Summary of missing values ................................... 9
    2.2.3 Imputation ....................................................... 12
    2.2.4 Plot types ........................................................ 19
    2.2.5 Design issues ................................................... 21
    2.2.6 Data input and output ......................................... 22
    2.2.7 Additional features of the GUI ............................... 24
  2.3 Example ................................................................. 25
    2.3.1 Data ............................................................ 25
2.3.2 Exploring missings .................................................. 26
2.3.3 Check assumptions ................................................. 27
2.4 Summary ................................................................. 29

CHAPTER 3. ENABLING INTERACTIVITY ON DISPLAYS OF MULTIVARIATE TIME SERIES AND LONGITUDINAL DATA ............... 32
3.1 Introduction .............................................................. 33
3.1.1 Characterizing time .................................................. 33
3.1.2 Visualizing time ..................................................... 34
3.1.3 Interactive graphics ................................................. 38
3.2 Layering to create a plot ............................................. 39
3.3 A taxonomy of interactions for temporal data displays .............. 40
3.4 Display Pipeline ......................................................... 46
3.4.1 Wrapping ............................................................... 47
3.4.2 Faceting ............................................................... 50
3.4.3 Mirroring .............................................................. 51
3.4.4 Shifting ............................................................... 51
3.4.5 Additivity of interactions ......................................... 52
3.4.6 Incremental vs baseline operations ......................... 53
3.5 Linking ................................................................. 55
3.5.1 Self-linking .......................................................... 55
3.5.2 Linking between plots .......................................... 58
3.5.3 Additional linking issues ........................................ 61
3.6 Querying ................................................................. 62
3.7 Conclusions and future work ....................................... 62

CHAPTER 4. CRANVASTIME: AN INTERACTIVE TOOL FOR TEMPORAL AND LONGITUDINAL DATA PLOTS .................... 64
4.1 Introduction ............................................................. 64
4.2 Usage ................................................................. 66
4.3 Data ................................................................. 67
4.4 Functionality ......................................................... 71
  4.4.1 Discover the seasonality ...................................... 71
  4.4.2 Explore the individual variation .......................... 74
  4.4.3 Focalize to the center of y ................................. 78
  4.4.4 Connect time series to other variables ............... 82
  4.4.5 Miscellaneous .............................................. 86
4.5 Conclusions and future work ................................. 89

CHAPTER 5. NEW APPROACHES TO EVALUATE CONTIGUOUS CARTOGRAMS FOR AREAL DATA ................. 90
5.1 Introduction ........................................................ 90
  5.1.1 Non-contiguous cartogram ................................. 90
  5.1.2 Contiguous cartogram ..................................... 92
  5.1.3 Software .................................................. 93
  5.1.4 Visual challenges .......................................... 94
5.2 Evaluating cartograms ............................................ 95
  5.2.1 Existing work .............................................. 95
  5.2.2 Problems ................................................ 96
  5.2.3 Proposed evaluation functions ......................... 97
5.3 Interactivity .................................................... 101
  5.3.1 Linear interpolation ..................................... 102
  5.3.2 Parameter tuning by graphical user interface ....... 103
5.4 Sensitivity analysis ................................................ 107
  5.4.1 Data and parameters .................................... 107
  5.4.2 Results .................................................. 108
5.5 Conclusions and future work .................................. 115
CHAPTER 6. SUMMARY AND FUTURE PLANS .......................... 117

6.1 Summary of the contributions .................................... 117

6.2 Future work .......................................................... 117

BIBLIOGRAPHY .......................................................... 120
**LIST OF TABLES**

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 2.1</td>
<td>Imputation methods included in the missing data GUI</td>
</tr>
<tr>
<td>Table 2.2</td>
<td>Comparison of three multiple imputation packages</td>
</tr>
<tr>
<td>Table 3.1</td>
<td>Examples from <code>cranvas</code> of layers used in constructing plots</td>
</tr>
<tr>
<td>Table 3.2</td>
<td>The combination of interactions modifies the $y$-coordinates</td>
</tr>
<tr>
<td>Table 3.3</td>
<td>Brushing the “wide data” format</td>
</tr>
<tr>
<td>Table 3.4</td>
<td>Brushing the “long data” format</td>
</tr>
<tr>
<td>Table 3.5</td>
<td>Linking between the long and wide data format may produce a problem</td>
</tr>
<tr>
<td>Table 5.1</td>
<td>Comparing two diffusion cartograms</td>
</tr>
<tr>
<td>Table 5.2</td>
<td>Parameter values in the sensitivity analysis</td>
</tr>
<tr>
<td>Table 5.3</td>
<td>Parameters to obtain the smallest and largest total errors</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Overview of the missing data GUI</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>A numerical summary of missing values</td>
<td>10</td>
</tr>
<tr>
<td>2.3</td>
<td>Missingness maps</td>
<td>11</td>
</tr>
<tr>
<td>2.4</td>
<td>Results of different univariate imputations</td>
<td>14</td>
</tr>
<tr>
<td>2.5</td>
<td>Results of the nearest neighbor imputation methods</td>
<td>15</td>
</tr>
<tr>
<td>2.6</td>
<td>Illustration of the ( k ) nearest neighbors imputation method</td>
<td>16</td>
</tr>
<tr>
<td>2.7</td>
<td>Results of the multiple imputations</td>
<td>17</td>
</tr>
<tr>
<td>2.8</td>
<td>Results of four imputing chains by the \texttt{mice} package</td>
<td>19</td>
</tr>
<tr>
<td>2.9</td>
<td>Effect of conditioning on imputed values</td>
<td>20</td>
</tr>
<tr>
<td>2.10</td>
<td>Four types of graphs available in the GUI</td>
<td>22</td>
</tr>
<tr>
<td>2.11</td>
<td>Subsidiary GUI tabs</td>
<td>23</td>
</tr>
<tr>
<td>2.12</td>
<td>Subsidiary GUI to import data</td>
<td>24</td>
</tr>
<tr>
<td>2.13</td>
<td>Subsidiary GUI to export data</td>
<td>24</td>
</tr>
<tr>
<td>2.14</td>
<td>Interactive attributes list for variable selection</td>
<td>25</td>
</tr>
<tr>
<td>2.15</td>
<td>Exploring the effect missingness of the tao data</td>
<td>27</td>
</tr>
<tr>
<td>2.16</td>
<td>Exploring the effect missingness conditionally</td>
<td>28</td>
</tr>
<tr>
<td>2.17</td>
<td>Checking assumptions by parallel coordinates plots</td>
<td>29</td>
</tr>
<tr>
<td>2.18</td>
<td>Finding the dependent variables by scatterplots and contour plots</td>
<td>30</td>
</tr>
<tr>
<td>3.1</td>
<td>Time series plots for regular and irregular time spacing</td>
<td>34</td>
</tr>
<tr>
<td>3.2</td>
<td>Six variations of horizontal axis time plots for multivariate time series</td>
<td>35</td>
</tr>
<tr>
<td>3.3</td>
<td>Three types of time series plots in polar coordinates</td>
<td>38</td>
</tr>
<tr>
<td>3.4</td>
<td>Interactions for the Lynx trappings data</td>
<td>42</td>
</tr>
</tbody>
</table>
Figure 3.5  Interactions for the pig production data .......................... 43
Figure 3.6  Order of interaction matters on faceting .......................... 44
Figure 3.7  Bi-directional faceting ............................................. 45
Figure 3.8  Illustration of shifting .............................................. 46
Figure 3.9  Cartoon illustrating the $x$-wrapping ............................... 50
Figure 3.10 Three modes of highlighting when a point is brushed .............. 57
Figure 3.11 Linking between different plots for the Google Flu Trends data . . . 59
Figure 3.12 Additional data created from the vertical faceting ..................... 62

Figure 4.1  Limitations of the static plots ....................................... 64
Figure 4.2  Basic time plots created by cranvastime ............................. 70
Figure 4.3  X-wrapping the time series of nasa data ........................... 71
Figure 4.4  X-wrapping the time series with a larger period .................. 72
Figure 4.5  Fully x-wrapping the time series .................................. 72
Figure 4.6  Calculation of R squares ............................................ 74
Figure 4.7  Univariate time series with a non-integer period .................. 75
Figure 4.8  Faceting the pig production data by variable ....................... 76
Figure 4.9  Faceting by longitudinal individual .................................. 77
Figure 4.10 Shifting a variable over time to check the dependence .............. 77
Figure 4.11 Mirroring the lynx data by the mean ................................ 79
Figure 4.12 Y-wrapping the lynx data before and after mirroring ............... 79
Figure 4.13 Y-wrapping the four variables of the pig production data .......... 80
Figure 4.14 Y-wrapping the Google Flu Trends data by state .................. 81
Figure 4.15 Faceting the Remifentanil data by four covariates ................. 83
Figure 4.16 Faceting the nasa data by grid ..................................... 84
Figure 4.17 Linking several plots for the Google Flu Trends data ............... 85
Figure 4.18 Switching to the area plots from line plots ............................. 86
Figure 4.19 Zooming in and zooming out ......................................... 87
Figure 4.20 Panning the local series ............................................ 87
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.21</td>
<td>Different sizes and transparency of the points</td>
<td>88</td>
</tr>
<tr>
<td>4.22</td>
<td>Labels in different directions of points</td>
<td>89</td>
</tr>
<tr>
<td>5.1</td>
<td>A choropleth map for 2012 US presidential election</td>
<td>91</td>
</tr>
<tr>
<td>5.2</td>
<td>Non-contiguous cartograms for 2012 US presidential election</td>
<td>91</td>
</tr>
<tr>
<td>5.3</td>
<td>Limitation of the Dorling cartogram</td>
<td>92</td>
</tr>
<tr>
<td>5.4</td>
<td>Diffusion cartograms for 2012 US presidential election</td>
<td>95</td>
</tr>
<tr>
<td>5.5</td>
<td>Angle difference</td>
<td>97</td>
</tr>
<tr>
<td>5.6</td>
<td>Edge difference</td>
<td>98</td>
</tr>
<tr>
<td>5.7</td>
<td>Angle and edge difference before and after the diffusion cartogram</td>
<td>100</td>
</tr>
<tr>
<td>5.8</td>
<td>Scatterplots between the shape error and area error</td>
<td>101</td>
</tr>
<tr>
<td>5.9</td>
<td>Linear interpolation between two polygons</td>
<td>102</td>
</tr>
<tr>
<td>5.10</td>
<td>Linear interpolation between the original map and diffusion cartogram</td>
<td>103</td>
</tr>
<tr>
<td>5.11</td>
<td>Errors of the linear interpolations with weights spaced by 0.05</td>
<td>104</td>
</tr>
<tr>
<td>5.12</td>
<td>Six parameters to create the density matrix for the diffusion method</td>
<td>105</td>
</tr>
<tr>
<td>5.13</td>
<td>The shiny GUI for diffusion cartograms</td>
<td>106</td>
</tr>
<tr>
<td>5.14</td>
<td>The vertices tab</td>
<td>107</td>
</tr>
<tr>
<td>5.15</td>
<td>The best and worst cartograms selected by the total error</td>
<td>109</td>
</tr>
<tr>
<td>5.16</td>
<td>Errors by the interpolation weight</td>
<td>110</td>
</tr>
<tr>
<td>5.17</td>
<td>Line plots between the interpolation weight and total error</td>
<td>111</td>
</tr>
<tr>
<td>5.18</td>
<td>Boxplots of errors by the number of rows and aspect ratio</td>
<td>112</td>
</tr>
<tr>
<td>5.19</td>
<td>Boxplots of errors by blank and sea weight</td>
<td>112</td>
</tr>
<tr>
<td>5.20</td>
<td>Boxplots of errors by blank weight and the weight difference</td>
<td>113</td>
</tr>
<tr>
<td>5.21</td>
<td>Boxplots of errors by blur and blank weight</td>
<td>114</td>
</tr>
<tr>
<td>5.22</td>
<td>Boxplots of errors by sea width and blank weight</td>
<td>114</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENTS

I am very grateful to my major professors Dr. Di Cook and Dr. Heike Hofmann for their guidance, patience and support throughout this research. From Di and Heike, I learned a lot on Statistical computing and graphics, as well as the way of scientific research. I could not have completed my graduation education or started an academic career without Di’s encouragement and direction.

I would like to thank my committee members: Amy Froelich, Zhengyuan Zhu, Alan Wannemaker, and Petrutza Caragea, for their time and suggestions on my work. Additionally, I thank Dr. Froelich for funding me as a graduate assistant in the Stat 101 report project for three years. It was a great experience which benefitted me a lot in both research and teaching.

I would also like to thank John McDanold and Russell Allgor for the opportunities they have provided me as a summer intern in 2012 and 2013. They encouraged me to implement my research to the complex real problems, and their feedback enlightened me to improve my work. The cartogram chapter of my dissertation was inspired by the internship experience.

Finally I thank my family and friends for their support during my study at Iowa State University. My husband Yihui always cooked for me and told dry jokes. I would like to name one of the friends, Wei Zhang, who pushed me to do much exercise in the recent five years, resulting in my first 5K race and Cyman triathlon.
ABSTRACT

Visualization is widely used to explore data, examine variation, reveal trends, and diagnose models. Furthermore, interactive plots can re-focus the view to features of interest, drill down into a fine resolution, query or lookup elements, look at data from various directions, and connect plots with model analysis. However, for specific data types and specific exploratory purposes, the general interactions like brushing, panning, zooming, and querying can be insufficient. The lack of a grammar for interactive graphics makes differences between the user interactions on data and on the view of data difficult to delineate. This thesis partially addresses these issues and fills gaps in methodology from three application areas: missing values, temporal/longitudinal data, and areal data.

Interactive graphics plays different roles in three areas. In missing data analysis, many imputation methods have been developed but little has been done for exploring the missing value structure to determine the missingness pattern, or to evaluate the imputations. This research addresses this gap, focusing on an interactive tool to explore missings, check the missingness assumptions, and compare imputation methods. For temporal and longitudinal data, using static plots is inadequate for exploring the trends, seasonality or unusual individuals, especially when the data set is large. This research develops special interactions and discusses the elements and pipeline in the interactivity construction. It is implemented in the R package, cranvastime, with details on how to use for a number of datasets. For the areal data, cartograms are widely used but there is no universally good algorithm for cartogram construction or evaluation. This research proposes an evaluation criterion and utilizes an interactive interface to optimize the visualization between the original shape-reserved map and area-reserved cartogram.
CHAPTER 1. INTRODUCTION

Visualization is important for data exploration, to examine variation, assess trends, and diagnose statistical models. Unlike static graphs, the strength of dynamic and interactive graphics is generating plots that can be refreshed smoothly and quickly, enabling the display of many more views of the data and model in rapid succession. With multiple linked windows different aspects of data can be explored simultaneously. Interactive plots emphasize the user control, so that users can re-focus the view to features of interest, drill down into a fine resolution, query or lookup elements, and look at data from various directions. One important focus of statistical graphics is the desire to have interactive graphics to be closely connected with analyzing and modeling, which can involve specific exploratory purposes and data types. Hence the general manipulations such as brushing, panning and zooming, querying, can be insufficient. With the design of novelty interactions, a lack of the grammar for interactive graphics makes differences between the user interactions on data and on the view of data difficult to delineate. This thesis addresses these issues and fills the gaps in methodology for three application areas: missing values, temporal/longitudinal data, and areal data.

The dissertation consists of four parts, including three independent papers and a vignette for the interactive graphics software that applies the proposed grammatical construction. Chapter 2 develops a graphical user interface (GUI) to interactively explore the missing patterns, summarizes and imputes missing values with various methods. Chapter 3 discusses the elements of the interactive graphics for temporal and longitudinal data, and proposes the constructive pipeline for data transformation during complex interactions. Chapter 4 implements the grammatical construction in Chapter 3 to a software package, and document usage with many data examples. Chapter 5 studies the errors in contiguous cartograms, and develops an application to evaluate diffusion cartograms, numerically and graphically.
1.1 Interactive graphics

Historically, research on interactive statistical graphics dates to the late 1960s. Seminal works include PRIM-9 Fisherkeller et al. (1988), Dataviewer Hurley (1987), XLispStat Tierney (2009), DataDesk Velleman (1989), XGobi Swayne et al. (1998), and MANET Unwin et al. (1996a). These tools provided ways to interactively explore various types of multivariate data, and to connect plots with model fitting. Contemporary software in use includes Mondrian Theus (2003), GGobi Swayne et al. (2003), and the R packages iplots Urbanek and Theus (2003), cranvas Xie et al. (2015), ggvis RStudio (2014). The R packages represent very current directions of research in terms of getting interactive graphics available directly integrated with the prominent contemporary data analytical tool, R. From the computer science and information visualization communities there have been parallel graphics developments over the same time frame. Several currently available interactive software systems are Tableau Hanrahan et al. (2007), d3.js Bostock (2012), and Processing Reas and Fry (2007).

There has also been substantial research on how to effectively construct a display of data. For the static graphics, Cleveland (1985, 1993a) and Cleveland and McGill (1987) studied human perception of data plots, reporting that position along a line is the mapping of data to plot element that people can read off most accurately. Wilkinson et al. (2006) and Wickham (2009) developed a grammatical construction for data plots, that helps to explain how different charts are similar or distinct from each other, and standardize plot construction. Underlying the construction of interactive graphics is a data pipeline discussed in Buja et al. (1988a), Wickham et al. (2009), Lawrence and Verzani (2012), Xie et al. (2014).

1.2 Missing data exploration

Missing values are a very common problem affecting data analysis. Many imputation methods have been developed but little has been done for exploring the missing value structure in order to determine the type of missing value pattern, and to evaluate the imputations. Many software packages handle missing values by simply removing the incomplete records with or without a warning, especially when the data are real-valued and multivariate. In order to
decide what to do with the missing values, before analyzing the data, we need to understand what the distribution of the missing values is, and how the missingness depends on the other collected variables. Also, for some model-based imputation methods, it is important to check the assumptions, like missing completely at random (MCAR) or missing at random (MAR), despite being difficult to prove.

Chapter 2 describes an interactive tool, the R package MissingDataGUI, to explore missing value structure, examining missingness assumptions, and comparing imputation results using static plots and numerical summaries. The GUI can handle the data set with multiple types of variables, summarize the missing values numerically and graphically, impute the missings by univariate, multivariate, and multiple imputation with or without conditioning factors, compare the results from different methods and chains, and suggest the dependent variables. Two real data studies were given to reveal the ability of comparing the imputation methods, and checking the assumptions. This paper is accepted by Journal of Statistical Software, and the package MissingDataGUI is available on CRAN.

1.3 Time-related interactions

For the temporal and longitudinal data, using the static plots is inadequate for exploring the trends, seasonality or unusual individuals, especially when the data set is large. With large longitudinal data plots are too congested by lines to reveal interesting details. Long time series make it difficult to explore periodicity and trend. Interactive graphics can help to tease these many components apart. Users should be able to facet multiple series, zoom in a long time series, query a specific individual, or wrapping the series to check the regularity of the period. Facilitating these interactions requires a new conceptual framework, and a proof-of-concept implementation for testing.

Chapter 3 elaborates the elements and pipeline of the interactive visualization for temporal and longitudinal data. Starting with an overview of the existing static visualization on time series and longitudinal data, the chapter introduces the basic elements and necessary interactions for time dependent data, followed by the pipeline of data transformation and linkage. This work is different from the existing interactive visualization grammar, because the new
interactions bring additional data that require additional storage and accessibility. And the pipeline could be expanded to other data types, like the areal data. This paper is submitted to *Journal of Computational and Graphical Statistics*.

Chapter 4 describes the implementation of Chapter 3: cranvastime. Cranvastime, as a part of the R package cranvas, produces interactive graphics for temporal and longitudinal data, with special design to explore the seasonality, individual variation, and to display a relatively large set of series. More concrete interactions with real examples are introduced in this chapter.

### 1.4 Cartogram evaluation

In the analysis of spatial data it is common to examine variables measured on areas. For example, to examine the US electoral trends, in electing a president, the basic unit is a state, because states votes are added as blocks to give a total count for each candidate. Among many types of spatial data, like points, lines, or areas, this work focuses on the areal data, which is displayed by polygons. This type of data is commonly represented by choropleth maps. However, in these displays attention focuses on the large regions, tending to diminish the small areas, which could be crucial areas. To emphasize the important areas, cartogram can be used. A cartogram reshapes the areas in proportion to the numerical value, while maintaining spatial integrity as much as possible. These have been effectively constructed manually, for specific purposes, but there is no universally best automatic algorithm for cartogram construction. In addition, evaluating the error in a cartogram is not broadly discussed and simply absent from the documentation of different cartogram methods. The result can be distorted cartograms, that are barely recognizable by a viewer, and inaccurate representation of the data.

Chapter 5 introduces the visualization for areal data focusing on diffusion cartograms. The cartogram interactions are described, and a new criterion to evaluate the visualization is proposed. The work also investigates the criterion with a simulation study. An R package cartogram is developed with an interactive application by shiny Chang et al. (2015). Four cartogram algorithms are provided in the package: Dorling and Dorling-like, non-contiguous, diffusion-based and grid-based cartograms.
Abstract

Missing values are common in data, and usually require attention in order to conduct the statistical analysis. One of the first steps is to explore the structure of the missing values, and how missingness relates to the other collected variables. This article describes an R package, that provides a graphical user interface (GUI) designed to help explore the missing data structure and to examine the results of different imputation methods. The GUI provides numerical and graphical summaries conditional on missingness, and includes imputations using fixed values, multiple imputations and nearest neighbors.

Keywords: missing values, imputation, exploratory data analysis, statistical graphics, data visualization, graphical user interface
2.1 Introduction

Missing values are a very common problem affecting data analysis. Many imputation methods have been developed but little has been done for exploring the missing value structure visually. Most plotting methods handle missing values by simply removing the incomplete records with or without a warning, especially when the data are continuous. Most statistical functions provide a limited list of handling missing values, such as, delete all cases with any missing values, delete pairwise or on single variables only.

The issue is, that in order to decide what to do with the missing values before analyzing the data, we need to understand what the distribution of the missing values is, and how the missingness depends on the other collected variables. A few \texttt{R} packages, like \texttt{Hmisc} (Harrell, 2013), \texttt{norm} (Novo and Schafer, 2013), and \texttt{mice} (van Buuren and Groothuis-Oudshoorn, 2011), have some routines for summarizing the number of missing by variable, and by case, in preparation for imputing the missing values. To understand the distribution of missings versus non-missings it is also important to make plots of the data.

For model-based imputation methods, it is important to check assumptions like missing completely at random (MCAR) or missing at random (MAR). These are not easy to verify. Little (1988) provided tests of the MCAR assumption, under normality conditions, and Jaeger (2006) proposed a test for MAR under some distributional conditions. Both tests employ inference based on likelihood ratios, and caution that the tests are sensitive to model misspecification (Little, 1988). Visual exploration of the missingness can help check the assumptions: it cannot prove any randomness assumption holds but visual checks can be used to reject MCAR assumptions, or suggest what dependencies exist, and should be incorporated into imputation for MAR data.

Some existing work describing visual exploration of missingness, and implementations, can be found in Unwin et al. (1996a), Swayne and Buja (1998), and Templ and Filzmoser (2008). \texttt{MANET} (Unwin et al., 1996a) implements the interactive methods to missing data. It presents the segmented barcharts of missing versus non-missing values for each variable, and with its many plot types like histograms, scatterplots, and mosaic plots, encourages the user to select cases
that are missing on any variable to highlight in other plots. This enables the user to explore
the missing status dependence in the distributions of the complete cases of other variables.

**XGobi** (Swayne et al., 1998), which implements the ideas described in Swayne and Buja (1998),
is similar to **MANET**, but focuses on interactive graphics for exploring missing values in real-valued data. It creates a shadow matrix of the original data where entries are 0 (complete) or
1 (missing value). This additional data structure allows the user to explore the multivariate
pattern of missing values, the dependence between missing value status and complete cases,
and compare imputation methods. These ideas were re-implemented in **GGobi** (Swayne et al.,
2003).

In the **R** community, the package **VIM** (Templ et al., 2013) provides a graphical user interface
via **VIMGUI** (Schopfhauser et al., 2013), to explore the structure of missing values and the
quality of several single imputation methods (kNN, hotdeck, irmi). Some packages for multiple
imputation have interfaces for easy manipulation, for example, migui, AmeliaView() and miP.
The **migui** (Lee and Su, 2011) is an interface for **mi** (Su et al., 2011), which implements
multiple imputation via Bayesian models and weakly informative prior distributions. The
function **AmeliaView()** in **Amelia** (Honaker et al., 2011), generates a graphical interface, to
implement its “EM with bootstrapping” algorithm. The package **miP** (Brix, 2012) adopts **VIM**
to visualize the imputation results from packages **mice**, **mi**, and **Amelia**.

This current work describes a new package for **R**, **MissingDataGUI**, which allows the ex-
ploration of missing value structure, and comparison of different imputations, using static
graphics and numerical summaries. The GUI makes these methods accessible for novice users.
This work builds on the ideas developed in Unwin et al. (1996a) and Swayne and Buja (1998).
The package utilizes routines in **Hmisc**, **norm**, **mice**, and **mi** for multiple imputation, and pro-
vides several other routines including kNN, random sampling and fixed values for the single
imputation. Section 2.2 explains the GUI design, functionality and rationale. Section 2.3 gives
a usage example.
2.2 Functionality

2.2.1 Overview of the missing data GUI

Figure 2.1 Overview of the missing data GUI. Region 1 contains the list of variables, variable type, and summary of missings on that variable. Region 2 has a list of the categorical variables that can be used for conditioning plots and imputations. Region 3 has a selection panel for selection of coloring by different types of missingness in the plots. Region 4 contains a radio button selection of imputation methods. Region 5 has several plot type selections, and region 6 allows selecting numeric or graphical summaries and some output routines. The summaries are displayed in the region 7.
The appearance of the missing data GUI is shown in Figure 2.1. (Section 2.3 describes the dataset.) All variables in the data along with the variable type and the percentages of NA’s are listed on the top left (region 1). The categorical variables (factor, ordinal factor, and character), auto-detected by their type, are shown on the bottom left as the potential conditioning variables (region 2). The variables having missing values are displayed under “Color.by.the.missing.of” on the top center (region 3). The graphical summary will distinguish the imputations from the observations by two colors, yellow (missing) versus blue (non-missing). This panel is used to choose what missing structure to color. Selecting the first row “Missing Any Variables” means that the color will depend on whether the case has missing values in any variables. The second row “Missing on Selected Variables” means the graph is colored by whether the case has missings in the selected variable. “Method” (region 4) and “Graph Type” (region 5) are two widgets illustrated in Sections 2.2.3 and 2.2.4. On the top right (region 6) there are five buttons: “Summary” can create a window as described in Section 2.2.2; “Plot” produces the plots in the graphics panel on the bottom right of GUI (region 7); “Export data” saves the imputed data into a file or to an R data frame; “Save plot” saves the plots in region 7 to png files; “Quit” destroys the main GUI window and the derived child windows.

2.2.2 Summary of missing values

2.2.2.1 Numerical summaries

To investigate missingness in a data set, start examining the numerical summaries of the missings. The “Summary” button will open a window with the overall missingness information (Figure 2.2 left panel) or conditional summary (Figure 2.2 right panel), depending on whether conditioning variables are chosen. Both summary windows present the percent of the values that are missing, the percent of variables that contain missing values, the percent of the cases that have at least one missing value, along with a tabulation of the number of values missing per case. The style of the table follows the summary provided by the package norm. In Figure 2.2 (left) it can be seen that the data has two observations have 3 missing values, another two have 2 missing values, 167 observations have one missing value and 565 are complete. By
percentages, 76.8% of the cases have no missings. Figure 2.2 (right) is conditioned on the variable “year”, which produced two boxes for 1993 and 1997 respectively. We can see that there are fewer missing values in 1997 than 1993, and all the observations having more than 1 missings appeared in 1993.

Figure 2.2 A numerical summary of missing values in the data is shown in a pop-up window. The left panel is the overall summary. The right panel shows the summary conditioned on “year”. The percentages of missings by total number of data values, by variables and by cases, is shown on the top. This dataset has 8 variables and the missing values by variable are summarized in the bottom table. No cases have more than 3 missing values, 76.8% of cases are complete, 22.7% of cases have one missing value, and only 4 cases have more than one missing values. The right panel shows that the missing pattern is different for each year.

2.2.2.2 Missingness map

The missingness map (Figure 2.3) provides a graphical summary of the missing patterns. Like the shadow matrix used in GGobi, the missingness map shows the position of missing values relative to variables and cases. The R packages Amelia (Honaker et al., 2011) and VIM (Templ et al., 2013) have versions of missingness maps. Organizing the missing values into blocks can be achieved by re-ordering variables and case ids, making it easier to see missing patterns, especially
for large data. Two re-ordered missingness maps are shown in Figure 2.3. One arranges the variables and cases by the number of missings, from the largest to the smallest; the other applies hierarchical clustering to both rows and columns. The strength of missingness map is to reveal whether the missings occur at some variables simultaneously. If so, then a similar missing pattern may indicate some association between the variables. If the missings happen at some observations synchronously, then it suggests dependence between those observations.

![Missingness maps](image)

**Figure 2.3** Missingness maps, same data but different ordering of variables (rows) and cases (columns): (left) raw data order, (middle) variables and cases sorted by decreasing number of missing values, (right) sorted by hierarchical clustering of missingness. From the raw data missingness map, the horizontal stripes indicate several variables have many missings, and the vertical stripes near the bottom indicate some structural missing cases. When variables and cases are sorted by missingness rate (middle), variables with missings often have missings on the same cases, and the additional few sporadic missing values can be easily spotted. Using the clustered missingness map (right) the blocks of missings on variables and cases is more easily seen.

**Figure 2.3** displays 245 observations and 34 variables for the dataset **brfss** (described in Section 2.3). From the missingness maps we can see that most of the missings occurred in seven variables. The missingness on some variables occur synchronously, indicating association. Users of the data should check the data collection procedures for these variables. For example, in
this data, questions about the drinking time and amount (ALCDAY4 and AVEDRNK2, the
top two variables in the right panel) were both skipped when the subject answered a previous
question with “did not drink in the past 30 days”.

2.2.3 Imputation

A number of imputation methods are available in the package. The purpose is two-fold: to
enable exploring dependence between missings or non-missings, and also to produce a complete
data set for later analysis. A few criteria were considered in the choices of methods to make
available and the design: (1) easy to understand and implement; (2) computing complexity is
medium or low; (3) adaptability to different situations, i.e., no strong model assumptions. Not
all of the imputation methods available in R are available in the package because (1) there are
too many methods and variations, so it is not practical to include all, and (2) users may use
their own method and import the result to missing data GUI for exploration.

The seven imputation methods provided are: “Below 10%”, “Simple”, “Neighbor”, “MI:areg”,
“MI:norm”, “MI:mice”, “MI:mi”. “Simple” and “Neighbor” contain more than one method.
Some methods (e.g., “Below 10%”) are only suitable for exploring the missingness patterns,
and are not suitable to use for producing a complete data set for analysis. Three tab labels
interface to the three methods provided by “Simple”, overall median, mean, and random value
(Figure 2.1, region 7). “Neighbor” interfaces to two methods, mean of the nearest neighbors,
and random nearest neighbor. The neighbor methods also allow the user to change the number
of neighbors. Table 2.1 summarizes and compares the imputation methods available in the
GUI.

2.2.3.1 Univariate imputations

The simplest start involves setting the missing values to 10% below the minimum on each
variable. The purpose of this is to place the missing values into the plot where they can be
distinguished from the non-missing values. In a scatterplot, all missing values will lie along a
Table 2.1  Imputation methods included in the missing data GUI. Strictly speaking, “Below 10%” is not an imputation method, but a way to put the missing values in the same graph with the observations. “Deterministic” indicates whether the method has a stochastic component or not. “Univariate” means whether the imputation only uses the individual variable where imputation is needed, or makes use of other variables as well. “Multiple imp.” indicates whether the methods is a type of multiple imputation that will provide multiple samples to impute the missings.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Deterministic</th>
<th>Univariate</th>
<th>Multiple imp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Below 10%</td>
<td>below 10% of the range</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Simple</td>
<td>overall median</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td></td>
<td>overall mean</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td></td>
<td>random value</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Neighbor</td>
<td>mean of the nearest neighbors</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td></td>
<td>random nearest neighbor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MI:areg</td>
<td>predictive mean matching</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>MI:norm</td>
<td>multivariate normal model</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>MI:mice</td>
<td>multivariate imp. by chained equations</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MI:mi</td>
<td>multiple iterative regression imputation</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

vertical line on the left or a horizontal line on the bottom of the display (Figure 2.4 (a)). This placement enables the distribution of missings to be compared with the distribution of non-missings. In the histogram, missing values will form a bar to the left of other data values. And in the parallel coordinates plot, the missing values are at the bottom of each axis.

Using the median, mean, or mode of the complete cases is a simple way to impute missing values. The software makes some automatic choices for the user: if the user selects median but the variable type is nominal, or selects mean but the variable is categorical, then the mode is returned. In the graph, points and bars are colored according to the missing status of the case. Figure 2.4 (b) and (c) show examples of the imputation by the median and mean for real-valued variables.

The “random value” method (Figure 2.4 (d)) randomly selects an existing value of the variable to impute the missing. When there is more than one missing value in an observation, then values are sampled independently from each related variable.

These imputation methods operate separately on each variable. Dependencies between variables are ignored, yielding covariance and correlation estimates that are potentially very
(a) Below 10%  (b) Overall median  

(c) Overall mean  (d) Random value

![Graphs](image)

Figure 2.4 Four panels of scatterplots displaying the results of different univariate imputations: (a) 10% below the minimum (not strictly an imputation method, it is used for displaying missings as part of a plot of complete cases); (b) median of each variable; (c) mean of each variable; (d) random selection from the existing values.

different from those of the complete cases. This could be a big problem for some analyses. These methods are not ideal from a statistical perspective. In some situations where the inadequate estimation of covariance does not affect results and conclusions they can provide a simple, few assumptions required, solution, but in most situations they are not advised. For the application here, we are primarily concerned about providing methods for analysts to explore the missing value structure, and the plots reveal quite clearly why these univariate imputation methods are inadequate. Figure 2.4 shows the “cross structure” (orange) induced on the pattern of points by mean and median imputation, and makes it quite clear that the covariance estimates for the imputed data would not well match that of the complete cases.
2.2.3.2 Neighbor imputations

The “Neighbor” methods replace a missing value with the mean of, or a random selection from, its $k$ nearest complete neighbors (Figure 2.5). The distance between two observations is calculated using Euclidean distance on the standardized variables that have no missings. Figure 2.6 illustrates the procedure. Ties are not considered, and only the first $k$ entries are used. This method requires at least one case in the dataset to be complete, and no categorical variables can be used. (Ordinal variables are treated as integers.) If there are less than $k$ complete cases, then all of them are used to generate the mean or a random value. If none of the cases are complete, then the mean or a random value of the entire data will be returned. By default $k = 5$, but this is the user’s choice.

![Scatterplots for nearest neighbor imputation methods](image)

**Figure 2.5** Scatterplots for nearest neighbor imputation methods: (a) mean of the 5 nearest neighbors, (b) a random value from the 5 nearest neighbors.

The neighbor methods in MissingDataGUI can be seen as two special cases of hot deck imputation (Andridge and Little, 2010). The neighbor mean method averages the weights on all chosen neighbors, and the random neighbor method places all the weight on one arbitrary neighbor. When $k = 1$, the methods are deterministic hot deck.
Figure 2.6 Illustration of the \( k \) nearest neighbors imputation method. The shaded entries are the complete observations to rank. The variables in red frames are used to compute the distance. After getting the rank of all complete observations, the first \( k \) are used as neighbors.

### 2.2.3.3 Multiple imputations

Multiple imputation, first proposed by Rubin (1978), is a method to get valid inferences by simulation. Multiple imputed datasets are generated based on the joint distribution, and serve a wide variety of analytical purposes. Functions from four R packages are utilized to implement multiple imputations in MissingDataGUI. Figure 2.7 demonstrates the results from different multiple imputations on the same data.

Among the four packages, \texttt{norm} is quite different from the other three. The ideas behind the package were introduced by Schafer and Olsen (1998). It assumes the observations are sampled from a multivariate normal distribution, and uses the EM algorithm to estimate the mean and variance-covariance matrix. It utilizes a data augmentation method to converge on distribution.

The other packages use a chained equation approach with similar steps but different settings. A comparison between the three packages is given in Table 2.2, based on Harrell (2013), van Buuren and Groothuis-Oudshoorn (2011), and Su et al. (2011). The main differences are that \texttt{Hmisc} provides three models with flexible drawing methods around the predicted values for quantitative variables, and applies bootstrap to obtain a sample for every iteration. The
package \texttt{mi} uses a convergence criterion to stop the iteration with some allowance for special situations. In between these two is \texttt{mice}: The models provided are more flexible than \texttt{Hmisc}, but not as bayesian as \texttt{mi}.

By default, $m = 3$ chains are imputed and users can choose the number of chains. Each chain will produce a result shown in a separate graphical panel. By switching between the panels, the user can compare the results and observe discrepancies between the results. Figure 2.8 shows the results of four different chains produced by \texttt{mice}. Three of the four produced results where a small clump of imputed values occurred.

![Figure 2.7](image)

**Figure 2.7** Scatterplots for the multiple imputations from four \texttt{R} packages: (a) predictive mean matching by \texttt{Hmisc}; (b) multivariate normal model by \texttt{norm}; (c) multivariate imputation using chained equations by \texttt{mice}; (d) multiple iterative regression imputation by \texttt{mi}. All the four imputations are conditioned on year.
Table 2.2  Comparison of the algorithm steps among three multiple imputation packages that use the chained equation approach.

<table>
<thead>
<tr>
<th>Algorithm steps</th>
<th>Hmisc</th>
<th>mice</th>
<th>mi</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Fill in the missing</td>
<td>at random</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Specify the model</td>
<td>pmm/regression/normpmm</td>
<td>selectable model or user-specific model</td>
<td></td>
</tr>
<tr>
<td>(Default model)</td>
<td>predictive mean matching</td>
<td>Baysian generalized linear models</td>
<td></td>
</tr>
<tr>
<td>3. Decide the data</td>
<td>a bootstrap sample</td>
<td>the entire dataset with the current imputed values</td>
<td></td>
</tr>
<tr>
<td>4. Iterate imputation</td>
<td>in every cycle, variables with missings are imputed sequentially</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Stop when</td>
<td>achieving the max # of iterations</td>
<td>difference of within and between variance is small</td>
<td></td>
</tr>
</tbody>
</table>

2.2.3.4 Conditional on the categorical variables

When the variables of interest have bimodal or multi-modal distributions, using center statistics like the mean or median for imputation, or simulating from an overall estimate like norm does, is inadequate because the center does not reflect the shape of distribution properly. In many situations, the modalities arise from the mixture of groups. Hence, a better imputation method is to condition by group, and then calculate the statistics.

This is available using the controls “categorical variables to condition on”. All categorical variables are listed with checkboxes. The variables checked will partition the data into blocks and then the imputation method is implemented in each block of the data. However, the condition is not used when the method is “Below 10%”, since the aim of “Below 10%” is simply to display the missings away from the non-missings. If the conditioning factor variable has missing values, then a “factor = NA” group will be generated to calculate the numeric summary or the imputed values. If the conditioning factor itself is one of the plotting variables, then a message box will emerge to ask the user to impute the missing values on the factor before other variables, and the plots are created without the condition.

The importance of conditioning in the imputation is illustrated in Figure 2.9. Without the condition, the imputed do not match the distribution of complete values (Figure 2.9 left). Calculating separately by group provides a better result (Figure 2.9 right).
Figure 2.8  Results of four imputing chains by mice, starting with the default random seed. Users can switch the panels by clicking the tabs, or close a panel by hitting the ‘x’ sign. Focusing on the imputed values when air temperatures around 22 degree, we see that the first, third and fourth chains cluster values in a small range of y-axis, but the second chain spread them very evenly in the y-direction.

2.2.4  Plot types

There are four types of graphs available in MissingDataGUI: histogram/barchart, spineogram/spineplot, pairwise plots, and parallel coordinates plot. Figure 2.10 displays all the graph types. Two color-blind friendly colors represent the observations and imputed values on any chosen variables. In Figure 2.10 the yellow color means that the value is originally missing in humidity.
Figure 2.9  Effect of conditioning on imputed values. The left panel is the imputation by median without condition and the right one is conditioned on year. In the left plot we can see that the imputed values (yellow) fall between the two clusters, at the overall median. But when the imputation is conditioned on year (right plot), the imputed values are now better placed into the two clusters in the data.

Separate histograms (continuous variables) and barcharts (categorical variables) are shown for each of the variables selected. When the missing values and the complete values share one bar, the bar is cut into two parts, and the ratio of the two heights is equal to the ratio of missing and non-missing values in that bar.

The spinogram (continuous variable) and spineplot (categorical variable), introduced by Hummel (1996) and Theus and Lauer (1999), use width of the rectangle to represent count. Height is the same for all bars. The focus is on proportion for each group. The bars in the spinogram or spineplot are partitioned into two colors for the missing and non-missing values.

A scatterplot matrix is used to display pairs of variables. Variable names and scales are placed on the diagonal. For the continuous variables, the pairwise scatterplots are placed in the lower triangle, and the contour plots are shown in the upper triangle. For the categorical variables, barcharts are displayed in both upper and lower triangles. Bars are colored in proportion to the missings. The combination of continuous and categorical variables is displayed as side-by-side boxplots of missing and non-missing values for each category on the upper
triangle and side-by-side histograms on the lower triangle. Limited space available to the
graphics device limits the number of variables that can be shown. The upper limit of the
number of variables is set to be 5 and the lower limit is 2.

The parallel coordinates plot by Inselberg (1985) and Wegman (1990) can be used to high-
dimensional data. Though many plot types, like the scatterplot or histogram, are helpful to
reveal the missing pattern, they are not convenient to display many variables simultaneously.
The parallel coordinates plot can give an overview of a relatively large quantity of variables. In
MissingDataGUI, the order of the variables can be chosen in one of two ways: the original order
in the data, or by sorting the variables from the best separator to the worst of missing values by
the $F$-statistic from ANOVA. In Figure 2.10, the best separating variable for the missingness
of humidity is humidity itself, because the “below 10%” method makes a big gap between the
missing and non-missing values. “Below 10%” is not an ideal method for the ordered parallel
coordinates plot. However, the plot is still useful: it reveals that the missingness on humidity
occurred in one year and one location, when sea.surface.temp and air.temp were low.

2.2.5 Design issues

The missing data GUI is organized as one window with three tabs. As shown in Figure 2.1,
the summary tab includes all the important widgets: list of variables, radio for imputation
methods, checkboxes for the conditional variables, the graphics device, etc. An appropriate
layout makes the widgets less crowded, and is easy to maintain. The other two tabs are not as
critical as the main tab, but also play important roles.

The help tab shown in Figure 2.11 (left) has the same layout as the summary tab. The only
difference is that the graphics device is replaced by the help document. The corresponding help
shows up when the user moves the mouse upon a widget.

The Settings tab shown in Figure 2.11 (right) allows the user to choose options for the
imputation methods in the package mice, as well as other settings for the multiple imputation,
neighbor selection, and the display of parallel coordinates plot. To change the imputation
Figure 2.10  The four types of graphs available: (top, from left to right) barchart, histogram, spineplot, and spinogram, and (bottom, left to right) pairwise plots, and two parallel coordinates plots. The order of variables in the parallel coordinate plot changed from the original (upper plot) to being ordered by difference between missings and non-missings. All the plots use “below 10%” imputation and are colored by the missingness on humidity.

models, users can double click a variable in the left table, and select any method provided in the pop-up window. The choices vary depending on the type of the variable.

2.2.6 Data input and output

Data can be entered as either a data frame or a comma separated file (csv). The preferred approach is to read an existing data frame in R because the type of variables (e.g., factor, numeric) are preserved. MissingDataGUI(data) is used to achieve this.

If reading from a csv file, MissingDataGUI() will trigger the data import GUI (Figure 2.12), from which to select a file. The “Open” button is for choosing files and the “Watch Missing Values” buttons will launch the missing data GUI. The file format must be csv, and only one
Figure 2.11 Subsidiary GUI tabs: (left) help tab, (right) settings tab. The layout of the help tab mirrors the actual functional GUI. Mousing over any part of it or clicking the radio/checkbox items will pop up text explanations in the summary region. All the widgets have a detailed introduction. The settings tab is used to make changes to the variable types and algorithm options. Users can modify the number of imputed sets to generate, the random number seed, the number of neighbors, and the jitter setting for parallel coordinates plot.

data set can be imported into the missing data GUI at a time, although several files can be opened in the data import GUI.

Once values are imputed, and a complete data set created, it can be saved using the “Export data” button (Figure 2.13). Only the selected variables will be imputed, but users could choose whether to export the selected columns or all the columns (with NA’s existing in the unselected variables). The shadow matrix is exported by default, so that analysts can always track back to find the locations of the real missings. Data can be saved in three ways: a csv file, an rda file, or a data frame. The multiple imputed sets from several chains will be saved as a list in rda format or data frame, or in separate csv files.

The exported data with its shadow matrix can be loaded back into the GUI, which implies the imputed data from other imputation methods (not provided by the missing data GUI) can also be imported. Users only need to provide a shadow matrix which indicates the locations of missings. In other words, the imported structure should be a data frame or a csv file with the first \( n \) columns being the imputed data and the next \( n \) columns being the shadow matrix.
2.2.7 Additional features of the GUI

- Change the variable attributes. Double clicking on any variables in the top left table of the summary tab will open an attribute window, as displayed in Figure 2.14. Users could edit the variable name, or assign another class to the variable. When the class of a variable is switched from numeric/integer to character/factor/ordinal, the variable will be automatically loaded into the checkbox group as the potential conditioning variable.
• Search a variable by text typing. The variable table, conditioning checkboxes, and color-by-variable selector allow text entry to find a variable. This feature is especially useful when there are many variables in the data.

• Save the plots. Plots can be saved to png formatted files by “Save plot” button. The imputation method and plot type will be auto-completed in the file name.

Figure 2.14 The attributes list for variable selection is interactive. The name can be edited, and the class could be changed to one of the five classes: integer, numeric, character, factor, or ordinal (factor). When a numeric variable is changed to a categorical variable, the widget for conditions will be updated.

2.3 Example

2.3.1 Data

Two data sets are provided with the package: tao, which is used as the example in this section, and brfss. The brfss data is a subset of the 2009 survey from the Behavioral Risk Factor Surveillance System, an ongoing data collection program designed to measure behavioral risk factors for the US adult population (18 years of age or older). The website for this program is http://www.cdc.gov/BRFSS/index.htm.
The data `tao` is from the Tropical Atmosphere Ocean project (TAO) (McPhaden, 2011). The TAO array consists of approximately 70 moorings in the Tropical Pacific Ocean, telemetering oceanographic and meteorological data to shore in real-time via the Argos satellite system. A subset of data from 6 moorings in 1993 and 1997 is used for the example. The data has 8 variables (year, latitude, longitude, sea surface temperature, air temperature, humidity, uwind and vwind) and 736 observations. The numeric summary of the 8 variables is shown in Figure 2.2. This subset is provided by Cook and Swayne (2007a). We can open the GUI by the following commands:

```r
library("MissingDataGUI")
MissingDataGUI(tao)
```

### 2.3.2 Exploring missings

Three of the 8 variables have missing values. First, let’s look at the distribution of missings on these variables. Figure 2.15 (left) shows the pairwise plots of three variables (sea.surface.temp, air.temp, and humidity) with missing values on any of the three variables colored in yellow, and shown as 10% below the minimum data value. Cases which are missing on humidity (string of points at bottom of bottom row of plots) have low values of sea and air temperature. This suggests the dependence between humidity missingness and the temperature variables. Imputation methods that incorporate this dependence may be preferable.

Figure 2.15 (right) shows the data imputed with median values. This imputation imposes a cross structure on the data, which does not match the shape of the complete cases. This would not be a recommended method for creating a complete data set.

Figure 2.16 (left) shows the data imputed with median values conditional by year. This better matches the distribution of complete cases, although the imputed values still form bands in the scatterplot. This might be a problem because the variance estimation will be affected.

For this data, the better ways to impute the data would take the strong association between the variables into account. This suggests that neighbor or multiple imputation might be the more desirable imputation methods. Figure 2.16 (right) shows the results for MI:areg, the
Figure 2.15 (Left) Exploring the effect missingness (yellow) on humidity, sea and air temperature. Missings on humidity (the bottom line of the third row) occur at the lower temperature values, suggesting a dependence relationship. Missing values are not missing completely at random. (Right) Imputation using the medians. Median imputation introduces a cross structure to the point scatter, and the imputed values don’t match the data well.

regression-based imputation, conditional on year. The imputed values match the distribution of complete cases reasonably well. There are a few slight concerns: some of the imputed values have lower air temperature values than any of the complete cases, the spread of the imputed values is a little greater than the complete cases. But overall, this is probably as good as it is going to get with imputing the missings for this data set. It would be reasonable to export the imputed data for further analysis at this point.

2.3.3 Check assumptions

In the statistical imputation literature, there are three types of missing data mechanisms: MCAR (missing completely at random), MAR (missing at random), and MNAR (missing not at random). Many imputation methods, including multiple imputation, assume MCAR or MAR. However, MCAR is the most difficult mechanism to substantiate, because it requires that missingness be independent of the observed or other missing values. MAR is less strict, because it allows for missings to be dependent on observed values, but it still expects the missingness
Figure 2.16 (Left) Imputation using the median, conditional on year. Imputed values better match the complete cases, with the exception of the banding due to a fixed median value. (Right) Imputation using the multiple imputation MI:areg conditional on year. The distribution of imputed values is fairly close to the distribution of complete cases.

to be independent from other missing values. To assess the adequacy of the assumptions, an important step is to review the data generation process. Beyond this we follow a process of elimination. The missing pattern is believed MCAR unless there is strong evidence against it. If MCAR is negated, then it is believed that the pattern is MAR, unless there are strong indications in the data generation process that render MAR implausible. If not MAR, then MNAR has to be assumed.

As an example, let us check whether the two incomplete variables (air.temp and humidity) in the data tao follows MCAR or MAR. Figure 2.17 gives two parallel coordinates plots, colored by the missingness on air.temp (left) and humidity (right). The yellow lines are the cases that are missing only on air.temp (or humidity). The missing values are represented by “Below 10%” method. Most of the missings on air.temp occurred in one year and one location, with higher sea.surface.temp, higher uwind, and lower humidity. Most of the missings on humidity happened in the same location as the missings on air.temp but in another year, with lower sea.surface.temp and lower air.temp. This is strong evidence against MCAR.
Figure 2.17 Parallel coordinates plots colored by whether missing on air.temp (left) or humidity (right). The variables are sorted by the $F$-statistic of ANOVA, i.e., the difference between the missing data and the observed data on a standard scale. Any missing values in the present variables are imputed by “Below 10%” method. Obviously the missingness on air.temp and humidity associates with other variables like year and location, so the MCAR assumption on these two variables are violated.

Rejecting MAR is very difficult generally, even when an obvious difference between missings and non-missings can be seen in the plots, because the real values of the missings are unknown. For example, in Figure 2.18, the distributions of uwind and vwind conditioned on the missingness of air.temp are different, since the missings on air.temp are higher in uwind and more scattered in vwind. The reasoning is complicated, but we cannot reject MAR. It is possible that, conditional on uwind and vwind, the distribution of true air.temp values of the missings is the same as that of the observed values. Thus, uwind and vwind remove any dependence of missing status for the air.temp variable. Generally, it is not possible to establish MNAR without actually knowing the true values of the missings. However, for this data, the plots suggest that the imputation of air.temp should involve uwind and vwind.

2.4 Summary

The MissingDataGUI package makes it possible to explore patterns of missingness in data and the impact of various imputation methods on the distribution of values in the data. Future work would add interaction to the plots so that it is possible to brush points to more completely explore missing structure, as can be done in GGobi and MANET.
Figure 2.18  Scatterplots (top row) and contour plots (bottom row) of uwind and vwind, two complete variables, colored by whether missing on air.temp (left column) or humidity (right column). The missings of air.temp is averagely higher on uwind and has a larger variation, than the non-missings of air.temp. In reverse, the joint distribution of uwind and vwind on the missings of humidity is closer to that of the non-missings.
Software

MissingDataGUI is written in R 3.1.1 (R Development Core Team, 2012) and based on the package gWidgets (Verzani et al., 2012) with the toolkit RGtk2. On different platforms (Windows, Linux, Mac) the appearance of the GUI will differ slightly, but the functionality will be the same.

The histogram/barchart, spinogram/spineplot, and missingness map are generated using ggplot2 (Wickham, 2009). The scatterplot matrix and parallel coordinates plot are produced by package GGally (Schloerke et al., 2013).

Acknowledgements

This work was partially supported by an unrestricted fellowship from Novartis, and National Science Research grant DMS0706949.
CHAPTER 3. ENABLING INTERACTIVITY ON DISPLAYS OF MULTIVARIATE TIME SERIES AND LONGITUDINAL DATA

A paper submitted to *Journal of Computational and Graphical Statistics*
Xiaoyue Cheng, Dianne Cook, Heike Hofmann

Abstract

Temporal data is information measured in the context of time. This contextual structure provides components that need to be explored to understand the data and that can form the basis of interactions applied to the plots. In multivariate time series we expect to see temporal dependence, long term and seasonal trends and cross-correlations. In longitudinal data we also expect within and between subject dependence. Time series and longitudinal data, although analyzed differently, are often plotted using similar displays. We provide a taxonomy of interactions on plots that can enable exploring temporal components of these data types, and describe how to build these interactions using data transformations. Because temporal data is often accompanied other types of data we also describe how to link the temporal plots with other displays of data. The ideas are conceptualized into a data pipeline for temporal data, and implemented into the R package *cranvas*. This package provides many different types of interactive graphics that can be used together to explore data or diagnose a model fit.

**Keywords:** interactive graphics, multivariate time series, longitudinal data, multiple linked windows, data visualization, statistical graphics
3.1 Introduction

Constructing interactive graphics for temporal data can be enabled by building upon static displays. Aspects of the graphical elements in the displays can be made accessible to modification by user actions, for the purpose of facilitating different exploration of the temporal components in the data. To explain how to do this we first need to understand how time might be structured, and the common types of temporal data displays.

3.1.1 Characterizing time

In data, time is coded in many different ways: as a date/time format ("Wed Oct 15 09:51:53 2014"), as discrete or continuous values, sporadic events or intervals. Recoding a time variable into other units like week in the year or days in the month, although convenient for some tasks, brings imprecision, as do events like leap years, and seconds. There may not be a well-defined absolute time line, and periodicity can be hard to quantify. Variables measured over time may be measured on different scales, e.g. atmospheric particulate matter and mortality might be drawn from different sources to study the effect of pollution on human health and measured at different resolutions.

The most common description of time is as a continuous or discrete ordered numerical variable. All data is technically discrete, but if measurements are recorded often enough, and long enough they are effectively continuous. For example, currency exchange rates change on a microsecond basis, blood pressure measurements made with a wearable device records at every minute. For these examples time could essentially be considered continuous. However, it may be not helpful to evaluate trends on this microscale, and aggregating at an hourly, daily or monthly value may be sufficient. For simplicity, the methods developed in this paper assume the time variable is measured on a discrete scale.

On a discrete scale, it may be possible to have regular or irregular time spacing. Regular time spacing means that the measurement is collected on constant time intervals, e.g. average monthly temperature in climate records. Irregular time spacing typically arise from events like measurements taken during visits to the doctor’s office. Figure 3.1 illustrates data collected at
regular, and irregular, intervals, respectively.

Figure 3.1 Time series plots for regular time spacing (left) and irregular time spacing (right). Tick marks at bottom indicate the time sampling.

When many measurements are made more complications ensue. In climate records we may have temperature measurements taken at many different locations. In longitudinal data, we may have records for many patients. Making comparisons between many time series is a challenge. This work addresses this for a moderate number of series.

3.1.2 Visualizing time

Longitudinal data and time series data, although analyzed very differently, have in common the context of time that is commonly plotted in similar ways. Here are examples of common types of temporal displays.

Time is most conventionally displayed on a horizontal axis of a plot. There are many different variations:

- A line graph, the basic building block for temporal data, displays the measured variable on the vertical axis, time horizontally, and consecutive time points are connected with line segments. Figure 3.1 gives two examples of line graphs, for a single measured variable: (left) a classical time series plot, (right) a profile plot of longitudinal data. When there are many series, for example time series of different stocks or different geographic locations, or many patients, the series may be overlaid on the same plot (Figure 3.2 row 1 left), or faceted in several blocks (Figure 3.2 row 1 right).
Figure 3.2 Six variations of horizontal axis time plots for multivariate time series: (From top left to bottom right) overlaid line plots, faceted line plots, stacked graph, faceted area chart, themeriver, horizon graph. Plots in the right column are examples of small multiples.
• Small multiples are used to display multiple series in separate plots (Figure 3.2 row 2 right). The terminology small multiples was introduced by Tufte and Graves-Morris (1983). A special modification was developed for multiple times series, called sparklines Tufte (2006). Small multiples can also be generated by subsetting based on categorical covariates (e.g. Cleveland (1993b)).

• Stacked graphs. Originated by Playfair in 1700’s and recently discussed by Byron and Wattenberg (2008); Javed et al. (2010); Heer et al. (2010), a stacked graph draws the time series sequentially, and uses the previous time series as the baseline for the current series (Figure 3.2 row 2 left). It is mostly used for the longitudinal data rather than multivariate time series since the individuals from the longitudinal data share the same scale.

• ThemERIVER and streamgraph. ThemERIVER is created by Havre et al. (2000), which is a special case of the stacked graphs, since it moves the starting baseline from the bottom to the center, and makes the plot symmetric vertically (Figure 3.2 row 3 left). Streamgraph is developed later by Byron and Wattenberg (2008). It changed the algorithm to avoid the symmetry which increases the internal distortion.

• Horizon graphs. The horizon graph is inspired by two-tone pseudo coloring (Saito et al., 2005) and formally developed at Panopticon Software (Reijner et al., 2008). Two-tone pseudo coloring is a technique to visualize the details of multiple time series precisely and effectively. However, the horizon graph became more popular after mirroring the lower part of the series and simplifying the color scheme (Figure 3.2 row 3 right). The horizon graphs were designed for visualizing the stock prices and economic/financial data, so the features fit the requirements very well: (1) The data have a baseline, which is usually the value at the starting time point. Then the baseline can be used to mirror the negative part to the positive, in order to save the graph space, where ‘negative/positive’ means smaller/greater than the baseline. (2) The positive and negative performance should be distinguished, so the horizon graph provides two hues. (3) The number of the color bands should be small, usually three color bands for the positive values and three for the
negative. Finding the band height is easy for the stock prices since they can use 10% of the initial value, and in most cases the price will not increase or decrease for more than 30%.

When time can be broken into two components it might be displayed on both horizontal and vertical axes:

- A high frequency time series often has hierarchic or nested period levels, like year, day, minute, etc. Those levels can be placed on horizontal and vertical axes to reveal the periodic dependency. For example, Keller and Keller (1993) used days and hours on two axes. In these graphs, the measurements of the time series are drawn in the grids via aesthetic settings like color or size.

- Calendar heat maps. Van Wijk and Van Selow (1999) proposed a colored calendar visualization (weeks and days on two axes). \texttt{d3.js} (Bostock et al., 2011) applies the calendar heat maps and makes it interactive.

Because time in some circumstances can be considered to be cyclical it is sometimes displayed in the polar coordinates:

- Nightingale’s coxcomb. Florence Nightingale might be the earliest author of a time series plot in polar coordinates. In the original plots, two unstacked barcharts were made in polar coordinates. Each diagram represents for one year. Later, people use the Nightingale’s coxcomb (Nightingale, 1858), also called circular histogram or rose diagram (Nemec, 1988), to plot the time series with a regular period like year or day.

- Spiral graphs. This approach is proposed by Weber et al. (2001). It can be seen as a temporal heatmap in polar coordinates. Figure 3.3 (right) shows an example. This approach is good for seeking the period, but the length for the same time unit changes over the loops. Besides, spiral graphs would be unhandy for the short period problems and multiple time series.
3.1.3 Interactive graphics

Interactive graphics emphasize the user manipulation of plot elements via input devices like the keyboard and mouse (Symanzik, 2012). Swayne and Klinke (1999) surveyed the use of the term “interactive graphics”, which revealed some differences in what people mean when they use the term. They found that most commonly people perceived interactive graphics to mean that a new plot can be recreated quickly from the command line, like base R plots. They suggested using a different term, direct manipulation, to mean directly changing elements of the plot using input devices. However, this term did not gain traction in the community, and we still use interactive graphics. To be clear, we use it here to indicate direct manipulation of plot elements through input actions of mouse or key strokes.

The work described here builds from a history of statistics software systems that support interactive graphics: e.g. PRIM-9 (FisherKeller et al., 1988), Data Desk (Velleman and Velleman, 1988), LISP-STAT (Tierney, 1990), XGobi (Swayne et al., 1998) and GGobi (Cook and Swayne, 2007b), MANET (Unwin et al., 1996b), Mondrian (Theus, 2002). The software Diamond
Fast (Unwin and Wills, 1988), XQz (McDougall and Cook, 1994), and Fortune (Kotter and Theus) provided tools specifically for exploring time series data. With the current popularity of R language (R Core Team, 2014), ideally interactive graphics can integrate closely and flexibly with statistical modeling. Packages that support this to varying extents are rggobi (Wickham et al., 2008), iplots (Urbanek and Wichtrey, 2013), rgl (Adler et al., 2003), cranvas (Xie et al., 2015), ggvis (RStudio, 2014), and animint (Hocking et al., 2014).

Of these software, cranvas, which evolved substantially from GGobi, is the vehicle for the ideas described in this paper. At its foundation is a data pipeline that channels data to plot elements, and provides interaction through reactive data elements, using plumbr (Lawrence and Wickham, 2014). The graphics are constructed using Qt (Qt Project, 2014) that enables flexible plot design and fast rendering for smooth interaction. cranvas has many different types of plots and possible interactions. The design of cranvas, Xie et al. (2014) provides single display interactions and the linking between different displays. Single display interactions include brushing, zooming, panning, and querying. Linked brushing can be done between different displays. To integrate temporal displays in this system requires integrating with this setup.

The next section describes the building blocks for temporal displays, which is followed by a taxonomy of interactive tasks that is desirable to use for exploration (Section 3.3). How to realize the interactions is described in Section 3.4. Linking between temporal data displays and other plots is described in Section 3.5.

3.2 Layering to create a plot

For interactive graphics, layering up the plot to enable different interactions can be useful, and efficient for large data. The base layer is typically the plot of all of the data. An overlay of a brush layer, where only elements actively being colored are displayed, can provide the efficiency of faster rendering. A brush layer is common to all displays because brushing is a basic function for interactive graphics. Background layers like an axis layer or grid layer are common too, but are not necessary in displays like maps. Table 3.1 lists the layers of common plot types in cranvas.
Table 3.1 Examples from *cranvas* of layers used in constructing plots. Some are common to all plots, and each plot has some layers that are unique. The “keys” layer listens for key strokes that change interaction modes. The “cue” layer on the histogram contains listeners and handles for dragging to change the binwidth interactively.

<table>
<thead>
<tr>
<th>display specific</th>
<th>scatterplot</th>
<th>point</th>
</tr>
</thead>
<tbody>
<tr>
<td>histogram</td>
<td>bar, cue</td>
<td></td>
</tr>
<tr>
<td>map</td>
<td>polygon, googlemaps, path, point</td>
<td></td>
</tr>
<tr>
<td>time plot</td>
<td>point, line, area, stats</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>common</th>
<th>required</th>
<th>brush, identify, keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>optional</td>
<td>grid, x-axis, y-axis, x-label, y-label, title</td>
<td></td>
</tr>
</tbody>
</table>

For temporal data displays, there are three basic layers: point, line, and area. The coordinates of the points are initially calculated from the data, and interactions may change the locations of the points in the display. The line layer connects the current positions of the points, so it requires the order, or, path information to know how to make the connections. The area layer shades the area under the line by constructing a baseline, matching the minimum data values, which enables closing the series to create a set of polygons. Each of these layers can take different interactions, and some care needs to be taken in realizing the effect on the different layers.

The point attributes, selected, color, size, or visibility, are generated for each observation when creating the plumbr mutaframe. The base element for the temporal plots are the points, and in *cranvas* brushing changes the attribute of each point in the mutaframe – essentially, points are brushed. The number of lines is one less than the number of points, and line color follows the first point in the defining pair. The number of polygons for the area display is the same as the number of lines, so color follows lines directly to polygons. The additional construction points of the area layer are only used in the area layer, and do not have independent attributes. This affects brushing behavior which is discussed later.

### 3.3 A taxonomy of interactions for temporal data displays

Wills (2012) summarizes the interactivity for temporal displays as changes to parameters or data. Data is mapped into coordinates in the plot. Parameters can be considered to be
attributes like color, labels, geometric elements, facet, or they can be considered to be aspects used to get the data into the plot like transformations, binning, dimension reductions or scales. Changes to the data or parameters provoke changes to the plots.

Parameters can often be attached to graphical user interface (GUI) items like sliders, that can generate the change in the plot. But more generally, interactions happen by direct action on the plot. For example, in brushing, the user selects elements like points in the plots. The software needs to locate these items in the data, update the attributes of these selected points, and broadcast these changes to other plots.

Some interactions, like brushing, selection, linking, zooming, panning, and querying, are universal for all plot types. Temporal and longitudinal data solicit special interactions to explore aspects of temporal dependence and trend. Buja et al. (1996) describe a taxonomy of interactive tasks for multivariate data. Here we describe a taxonomy of tasks for temporal data, that enable exploration of different components of time series and longitudinal data:

- Wrapping: In the $x$-direction explores seasonality and temporal dependence. In the $y$-direction it is done to compare magnitude of peaks and dips. It is easiest to explain $x$-wrapping: the series is cut at a fixed-length interval, the part of the series that extends beyond this interval is re-drawn from the initial point. This will change the $x$-coordinates of the data. The main purpose of $x$-wrapping is to explore the regularity of the periodicity. Some series that look to follow a regular period can be quickly revealed to have irregularities. The classical example is the lynx trappings data for 1821–1934 in the MacKenzie River District of North-West Canada (Campbell and Walker, 1977), which looks periodic (Figure 3.4). The wrapping shows that the period is not quite regular, matching one peak off-sets other peaks, and the period varies between 9-11 years. It is also possible to see that the increases are slower than the drops, that the population builds up and tends to plummet. To model this data well requires one also knows the snowshoe hare population. Figure 3.5 gives an example of the $y$-wrapping. It shows another classical example: quarterly pig production measured by four variables, herd size, production, profit and gilts, from 1967-1978 in United Kingdom (Andrews and Herzberg,
The $y$-wrapping induces something that might be considered a temporal boxplot, where density produced from overlaying the wrapped peaks emphasizes long runs of ups, or periodic ups.

Figure 3.4 Lynx trappings for 1821–1934: (a) Time series, (b – d) stages of $x$-wrapping, matching peaks, (e) faceted on the wrapped series, (f) area plot, (g) mirrored on the mean, blue indicating values above the mean, and yellow below the mean. (Video illustrating these interactions is available at https://vimeo.com/112431547 and https://vimeo.com/112432400.)
Figure 3.5 Quarterly pig production for 1967–1978 in UK measured by five variables: (a) faceted, (b) y-wrapped (see video at https://vimeo.com/112435889). Profit and gilts seem slightly lag-related. Herdsz and production may be related in a lag relationship also, but neither is seasonal. In the y-wrapped version density indicates the magnitude of values, and long periods of higher values, like in herds size are more visible.

- Faceting: This creates small multiples to organize and examine across structural data components. These components might be a period such as year, or month, or variables when multiple are measured at the same time points, or in longitudinal data these might be individuals. In the interactive setting these components can be used to slowly pull overlaid series apart, a process that might be more revealing that disjointly laying out each in a static plot. Figure 3.6 illustrates sequential faceting on two variables and three individuals. The order of these operations changes the final result, and changes which comparison is primary and which secondary. Figure 3.7 gives two more examples, by a grid of spatial locations, and by covariates, respectively. Watch the videos to see how these are achieved interactively. There is also another video at https://vimeo.com/112505175 shows faceting on period for a time series with a regular period of every 12 observations.
Figure 3.6  Order of interaction matters when faceting with two variables and three individuals: (a) all series overlaid, color indicates variable, (b) facet first on variable (A, B), (c) facet first by individual (1, 2, 3), (d) facets (b) by individual, and (e) facets (c) by variable. Both final configurations are useful: (d) supports the primary comparison of individuals, with variable comparisons secondary and (e) supports comparison of variables within individuals. (Video footage illustration is available at https://vimeo.com/112438919.)
Figure 3.7  Faceting can be conducted bi-directionally: (left) by spatial grid for spatiotemporal data (https://vimeo.com/112503285), (right) by two covariates, sex and age, in longitudinal data (https://vimeo.com/112509324).

- Mirroring: This splits the series vertically at a given value, and reflects the bottom half across this axis. With additional wrapping, the result is called a horizon graph, and it is used to compare the magnitude of peaks and troughs, particularly for binary phenomena like gains and losses. The $y$-coordinates of the data are modified by this interaction. The choice of split value are typically mean, median, midpoint of the range, or in economic data the initial series value. Figure 3.4 (f) and (g) shows mirroring of the lynx trappings data with the mean divider. We can see the peaks are sharp and irregular and the valleys are smooth and regular.

- Shifting: A series can be grabbed and shifted against another series. This is a more tangible operation than wrapping in order to compare periodicity and temporal dependence. Figure 3.8 shows three series that have been picked up and shifted together against the other three series to match peaks.
Figure 3.8  Illustration of shifting: three bottom series (variable A) are shifted horizontally to the right, to match the peak time with the top three series (variable B). (See also the video at https://vimeo.com/112439923.)

- Switching: At any time it should be possible to switch between line and area displays. Line plots are efficient but filling the area under the curve can give a stronger sense of the patterns in the series, especially when trying to compare multiple series. The video at https://vimeo.com/112530645 demonstrates switching.

3.4 Display Pipeline

Wilkinson et al. (2000, 2001, 2006) conceptualized and implemented a grammar of graphics that carefully details a mapping of data to plots. It was extended and implemented in ggplot2 by Wickham (2009). Data is parametrized into elements, and assigned to graphical elements, e.g. points, text, lines, polygons. Wills (2012) made a simple extension for interactive graphics: “allow the user to manipulate one of the two inputs, data or parameters, and show the changes in the chart”. Parameters, very generally, describe a very broad class of characteristics, e.g. display aesthetics like color or linetype, positional coordinates, statistics such as bins, scales like limits or color ladders, facets, and transformations. Much of what is needed to realize the taxonomy of tasks for interacting with time series plots can be considered to be data transformations. This section describes the transformations required to perform shifting, faceting, wrapping, and mirroring.
Let \((x, y)\) denote the positional coordinates for a temporal data set, where both are \(n\)-dimensional vectors. This notation is unconventional for time series, which typically uses \(x_t\) to represent the value records at time \(t\), but it is necessary for the graphical display because it allows us to think about horizontal and vertical positions and adjustments to these positions. Because, many sequential interactions can be made, and different types of interactions applied after each other, it is useful to incorporate notation specifying these into the equations. Let \(I\) be the temporal sequence of interactions, e.g. \{facet, wrap, facet, zoom, \ldots\}, and \(j \in J = \{1, 2, \ldots, J_i\}\) indicate the number of interactions made of type \(i \in I\). Let \(u_{ij} = (u_{ij1}, u_{ij2}, \ldots)\) denote the user’s input, e.g. key strokes, \(l_{ij} = (l_{ij1}, l_{ij2}, \ldots, l_{ijn})\) be a line group indicator for each point, since some interactions might force new sets of lines, \(p_i = (p_{i1}, p_{i2}, p_{i3}, \ldots)\) be a parameter vector, e.g. a wrapping stop value of 3 points in series, and \(m_{ij} = (\Delta x_{ij}, \Delta y_{ij})\) denote the movements in \(x\)- and \(y\)- directions, where \(i \in I\) is an interaction type, and \(j\) is the number conducted. The new data coordinates are given by

\[
(x, y)_{s + I_{ij}} = (x, y)_s + m_{ij},
\]

where \(s\) indicates the state before interaction. The movement \(m_{ij}\) can be written as a function on \(p_i, u_{ij}, l_{ij}, j,\) and the initial coordinates \((x, y)_0\),

\[
m_{ij} = f_i(p_i, u_{ij}, l_{ij}, j, (x, y)_0).
\]

These are the specific functional definitions used to generate the movement for the interactions available in cranvas.

### 3.4.1 Wrapping

Figure 3.4 (except bottom right plot) illustrates the horizontal wrapping of the lynx trapping data. The default wrapping interaction, by clicking a keystroke, induces the point at the end of the series, \(x_{(n)}\), to be cropped and moved to the very left side of the plot, at the same \(x\)-position as \(x_{(1)}\). With repeated keystrokes, the most recent elements of the series will be cropped and gradually wrapped onto the earliest elements. Only the \(x\)-coordinates are changed – the \(y\)-coordinates remain unchanged.
For simplicity, we assume that the difference between consecutive time values is 1. Let \( x_1, \ldots, x_n \) be the sorted \( x \)-coordinates of the \( n \) points in the series, that is the points in time order. The \( x \)-limits after \( j \) keystrokes for \( x \)-wrapping will be reset to \((x_1, x_{n-j})\), so the plot is rescaled accordingly. Let \( \Delta_{n-j} = x_{n-j} - x_1 + 1 \), then the new \( x \)-coordinate, \( x^* \) of \( x \) is

\[
x^* = \begin{cases} 
  x_{n-j} & \text{if } x - x_1 + 1 \mod \Delta_{n-j} = 0 \\
  (x - x_1 + 1) \mod \Delta_{n-j} + x_1 - 1 & \text{o.w.}
\end{cases}
\]

\[
x = \begin{cases} 
  x_{n-j} & \text{if } (x - x_1 + 1) \mod \Delta_{n-j} = 0 \\
  (x - x_1 + 1) - \left(\frac{x - x_1 + 1}{\Delta_{n-j}}\right) \times \Delta_{n-j} + x_1 - 1 & \text{o.w.}
\end{cases}
\]

\[
x = x - \left(\left\lceil\frac{x - x_1 + 1}{\Delta_{n-j}}\right\rceil - 1\right) \times \Delta_{n-j},
\]

enabling the movements for \( i = \text{wrap} \) to be described as

\[
m_{ij} = \begin{cases} 
  (-\left\lceil\frac{x - x_1 + 1}{\Delta_{n-j}}\right\rceil - 1) \times \Delta_{n-j}, & 1 \leq j \leq n - 3 \\
  (-\left\lceil\frac{x - x_1 + 1}{\Delta_3}\right\rceil - 1) \times \Delta_3, & j \geq n - 2
\end{cases}
\]

or equivalently in terms of line group indicators as well as points as

\[
m_{ij} = \begin{cases} 
  (-\left\lceil\frac{x - x_1 + 1}{\Delta_{n-j}}\right\rceil - 1) \times \Delta_{n-j}, & 1 \leq j \leq n - 3 \\
  (-\left\lceil\frac{x - x_1 + 1}{\Delta_3}\right\rceil - 1) \times \Delta_3, & j \geq n - 2
\end{cases}
\]

The line group indicator \( l_{ij} \) will depend on the number of interactions \( j \). The wrapping can be defined as an algorithm also:

1. Shift the data values up, usually by 1
2. Check the new \( x \)-limits, if a point has value large than upper limit, crop it using modulus arithmetic
3. Points that are cropped, have their line group indicator incremented
4. Connect the points that have the same line group indicator, in time order.
Sometimes it is useful to wrap the series faster. If you have a long time series, it might be useful to make full year jumps. This can be achieved with the above equations by setting the sequence of \( j \) to respect this period. In other instances it may be useful to have a multiplicative wrapping so that it looks like the series wraps faster and faster with each step. That means every keystroke will send a different number of points from the right to the left. The number of points wrapped by the \( j \)th step, can be represented by the user input parameter \( u_{ij} \). Then the \( x \)-range after \( j \) steps is 

\[
(x(1), x((n-\sum_{a=1}^{j} u_{ia})), \quad \Delta n-\sum_{a=1}^{j} u_{ia} = x((n-\sum_{a=1}^{j} u_{ia})) - x(1) + 1,
\]

and

\[
x^* = x - \left( \left\lfloor \frac{x - x(1) + 1}{\Delta n-\sum_{a=1}^{j} u_{ia}} \right\rfloor - 1 \right) \times \Delta n-\sum_{a=1}^{j} u_{ia},
\]

\[
m_{ij} = \begin{cases} 
(-1)_{ij} \times \Delta n-\sum_{a=1}^{j} u_{ia}, & 0 \leq \sum_{a=1}^{j} u_{ia} \leq n - 3 \\
(-1)_{ij} \times \Delta 3, & \sum_{a=1}^{j} u_{ia} \geq n - 2.
\end{cases}
\]

If the user wants to skip all intermediate positions and use only one jump to the fully wrapped position, then the new \( x \)-range will be \((x(1), x((p_{i}2)))\), where the parameter \( p_{i}2 \) is the length of period. Hence for \( j \geq 1 \),

\[
x^* = x - \left( \left\lfloor \frac{x - x(1) + 1}{\Delta p_{i}2} \right\rfloor - 1 \right) \times \Delta p_{i}2,
\]

\[
m_{ij} = \begin{cases} 
(-1)_{ij} \times \Delta p_{i}2, & 0\end{cases}
\]

To generalize our case to the irregular time series, we should specify a wrapping speed parameter \( p_{i}3 \), i.e., with every key stroke, the \( x \)-range is shortened by at least \( p_{i}3 \). The wrapping speed parameter will determine how many points are shifted every time, because if the difference between largest two points is greater than \( p_{i}3 \), then only one point is shifted; if the difference is smaller than \( p_{i}3 \), then more than one points are shifted. After \( j \) steps, the total number of points shifted is a function of \( p_{i}3 \) and \( x_{0} \). Denote the function as \( g_{j} \), so the new \( x \)-range is \((x(1), x((n-g_{j}(p_{i}3,x_{0})))\), and the movements can be calculated then.

Movements from the \( y \)-wrapping on the \( y \)-direction, as shown in Figure 3.5, could be obtained by similar formulas. It is messier to realize because the \( y \)-values are typically not in a sequential order which means that more structural components need to be added to the data to actually draw the wrapped series. Some of the issues are discussed later in this paper.
3.4.2 Faceting

When \( i = \text{facet by individual} \), an initial setting of the parameter \( p_{i1} = 0.05 \), which means that every hit on the key will lift the \( l \)th standardized line by \((l - 1) \times 0.05\). Hence for \( j \in \mathcal{J} \),

\[
    m_{ij} = \begin{cases} 
        (0, \ 0.05 \ (l - 1) \ j) & 1 \leq j < 20, \\
        (0, \ l_{i} - 1) & j \geq 20. 
    \end{cases}
\]

We can also generalize the equation above by

\[
    m_{ij} = \begin{cases} 
        (0, \ p_{i1} \ (l_{i} - 1) \ j) & 1 \leq j < \frac{1}{p_{i1}}, \\
        (0, \ l_{i} - 1) & j \geq \frac{1}{p_{i1}}, 
    \end{cases}
\]

where \( p_{i1} \in (0, 1) \).
The example shows that \( m_{ij} \) is a function of \( p_i, j \), and \( l_{ij} \), where \( l_{ij} = l_i \) in this example means that the line indicator for faceting is free from \( j \).

For \( i = \) facet by variable/period, one click will fully split the variables, so \( j \) does not matter. All lines should be standardized between \([0, 1]\) first, then the movement is given by

\[
 m_{ij} = (0, l_i - 1).
\]

Note that \( l_i \) in this case differs from \( l_i \) in faceting by individual.

### 3.4.3 Mirroring

To realize the interaction shown in Figure 3.4 (g), firstly we need to point the divider – mean in this example. Hence for \( i = \) mirroring, the divider parameter \( p = \frac{1}{n} \sum_{d=1}^{n} y_d \). Then by \( j \in J \) hits on some triggering key, the movements are

\[
 m_{ij} = \begin{cases} 
 (0, p + \max(p - y, y - p) - y) & j = 1, 3, 5, \ldots \\
 (0, 0) & j = 2, 4, 6, \ldots 
\end{cases}
\]

\[
 = \begin{cases} 
 (0, \max(2p - 2y, 0)) & j = 1, 3, 5, \ldots \\
 (0, 0) & j = 2, 4, 6, \ldots 
\end{cases}
\]

Note that if the mirroring is revisited after some other interactions, then the count of \( j \) should not be reset.

### 3.4.4 Shifting

Figure 3.8 illustrates shifting the series, which is used to compare one series against another. The user input uses \( u_{ij}, \) since the user can drag the series horizontally to any position. The starting point \( u_{ij1} \) and end point \( u_{ij2} \) of dragging on the \( x \)-axis, as well as the selected series \( u_{ij3} \) are the input from the user. The horizontally shifting will not change \( y \)-coordinates, so for \( i = x \)-shifting and \( j \in J \), we have

\[
 m_{ij} = ((u_{ij2} - u_{ij1}) \times I \{ l_{ij} = u_{ij3} \}, 0),
\]

where \( I \) is the indicator function.
### 3.4.5 Additivity of interactions

Most of the interactions could be considered to be additive. Figure 3.6 shows an example, where two different results are generated by different ordering of interactions. Faceting on individual is done after faceting on variable, with the process following panels (a) → (b) → (d). Faceting on variable after individual, as in the process (a) → (c) → (e), produces a different configuration of the time series. The additive application of interactions is not commutative. Both results are useful, because each facilitates a different type of comparison of the series, using proximity. Plot (d) enables the comparison of individuals, within variables, while plot (e) enables the comparison of series within individual. It is also interesting to note that wrapping vertically after mirroring, will result in a horizon graph, like Figure 3.2 (f).

The cumulative interactions could entirely change both $x$ and $y$ coordinates of data. For example, Figure 3.4 firstly directs 75 steps of $x$-wrapping, and then a faceting by period. That gives the eventual movement by

$$m = (-\left(\left\lfloor \frac{x - x(1) + 1}{\Delta_{39}} \right\rfloor - 1\right) \times \Delta_{39}, l_{facet} - 1)$$

$$= (-1_{\text{wrap,75}} - 1) \times \Delta_{39}, l_{facet} - 1)$$

$$= (-1_{\text{wrap,75}} - 1) \times \Delta_{39}, 1_{\text{wrap,75}} - 1).$$

Note that $1_{\text{facet}} = 1_{\text{wrap,75}}$ in this example, because the line group indicator for faceting by period is given by the wrapping steps.

In some other cases, a combination of interactions may only modify either $x$ or $y$ coordinates. Figure 3.6 (a → b → d) shows a combination of faceting first by variable then by individual. The final movement after the full split of individuals would be

$$m = (0, (1_{\text{facet by variable}} - 1) \times \max(1_{\text{facet by individual}}) + (1_{\text{facet by individual}} - 1)).$$

Table 3.2 takes the first point of each series as an example to show $y$ and the changes at the three stages.
Table 3.2  \( y \)-coordinates of the first point on each line, at Figure 3.6 (a) no faceting, (b) faceting by variable, (d) faceting by variable then individual.

<table>
<thead>
<tr>
<th>( l_{variable} )</th>
<th>( l_{individual} )</th>
<th>( y ) at (a)</th>
<th>( y ) at (b)</th>
<th>( y ) at (d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.16</td>
<td>0.16+(1-1)=0.16</td>
<td>0.16+(1-1)×3+(1-1)=0.16</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0.33</td>
<td>0.33+(1-1)=0.33</td>
<td>0.33+(1-1)×3+(2-1)=1.33</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0.26</td>
<td>0.26+(1-1)=0.26</td>
<td>0.26+(1-1)×3+(3-1)=2.26</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.84</td>
<td>0.84+(2-1)=1.84</td>
<td>0.84+(2-1)×3+(1-1)=3.84</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.84</td>
<td>0.84+(2-1)=1.84</td>
<td>0.84+(2-1)×3+(2-1)=4.84</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>0.90</td>
<td>0.90+(2-1)=1.90</td>
<td>0.90+(2-1)×3+(3-1)=5.90</td>
</tr>
</tbody>
</table>

3.4.6 Incremental vs baseline operations

Calculations can be made incrementally or with respect to a stable state (baseline), which, respectively, stores multiple copies of data, or a single storage of the data with storage of movement.

1. Incremental: Let

\[
s_0 = \text{initial status},
\]

\[
s_{t+1} = s_t + u_{i1}.
\]

Note that every new status is only one interaction after the previous status. The coordinates are given by

\[
(x, y)_{s_{t+1}} = (x, y)_{s_t} + m_{i1}
\]

\[
m_{i1} = f_i(p_i, u_{i1}, l_{i1}, (x, y)_{s_0})
\]

This procedure always computes the next position directly from the current position. The change only depends on the corresponding interaction parameters and the input. The current status is stored in the memory and ready to use for the next step. This method is an intuitive design for interactive graphs with few special interaction types. For example, scatterplots in cranvas can change the size and transparency of dots. The new size (or transparency) is always calculated by the multiplication of the current size (or
transparency) and a constant. The constant is greater than 1 if the aesthetic parameter is increasing, and less than 1 if the parameter is decreasing. The exponential growth of the parameter accelerates the change and reduces the times of repeated interaction.

The advantage of this procedure includes the straightforward design, the convenience of moving to the previous or next status, and the efficiency of avoiding the recomputation. However, when there are many special interactions that could transform the data, we need to record both the initial and current data positions of each interaction type in the stream $I$, because when moving backwards, we need to know when the initial state is reached and then stop. Then for an interaction stream $I$ of length $k$, at least $k + 1$ phases $((x, y)_{s_0}, (x, y)_{s_1}, (x, y)_{s_2}, \ldots, (x, y)_{s_k})$ should be saved, where $t_1, t_2, \ldots, t_k$ are the time of the end of interaction types 1, 2, $\ldots$, $k$. When the data set is large, those copies will occupy too much memory. Also, the storage and management of $u_i$'s and $l_i$'s is messy. Another drawback is that numerical errors could be introduced after the same number of forward and backward interactions, due to the floating-point arithmetic calculation.

2. Baseline: To store the $k + 1$ phases, we do not make $k + 1$ copies of the data set, instead, the movement item is traceable. The coordinates of any status can be computed by

\[
(x, y)_{s_t} = (x, y)_{s_0} + \sum_{i,j} m_{ij}
\]

\[
= (x, y)_{s_0} + \sum_{i,j} f_i(p_t, u_{ij}, l_{ij}, j, (x, y)_{s_0}).
\]

Note that when a new position is required, the calculation starts from the initial position instead of the previous status. The movements from the original to the current position are computed instantly.

The idea of baseline operation is not new. In the tour movement of XGobi and GGobi, the target position is calculated by the initial position and the projection parameters that come from the auto-oriented settings or user-oriented interactions (Cook et al., 1995; Cook and Buja, 1997). However, the interactivities that we discussed in this paper is more complex, because we need to consider the interaction stream $I$, but the tour movement
does not need to. When there is only one type of modification, the formulas in Section 3.4 will provide the target position easily. But when different types of modifications are mixed, the baseline method structures the computation well.

With this procedure we do not need to save the intermediate positions of the data, but we have to save the inputs for movements. Now the problem turns to: how to store the inputs including \( p_i, u_{ij}, l_{ij}, \) and \( j \)? The answer is, to save \( l_{iJ_i} \) with the data, where \( J_i \) is the largest \( j \) in each \( i \in I \). This is because \( p_i \) is fixed and \( j \) is known, \( u_{ij} \) is usually a short array, but \( l_{ij} \) is of the same length as the data, and depends on other parameters like \( u_{ij} \). So the data frame that we use to save the data includes not only the coordinates, point parameters like size and color, but also the line group indicators \( l_{iJ_i} \).

The advantage of this procedure is apparent: we do not have to save multiple copies of data, and it is a better way to manage the data and parameters during the interactions. However, it is not a comprehensive solution. We assumed that the movements are additive, but this is not always desirable. When changes in type of interaction make calculations better performed on a mid-way state, then it is better to stop, use this state as the baseline and then continue adding movements to this state.

3.5 Linking

Linking between plots is a critical component of using multiple linked windows (Stuetzle, 1987) to explore data. Xie et al. (2014) describes types of linking and how it is realized in cranvas. It is possible to both self-link, which is important for temporal data, and link on different data sources or aggregation levels, using categorical variables. For the temporal and longitudinal data, linking is complicated when there is the need for different forms of the dataset or additional data. Two situations are discussed in the following sections.

3.5.1 Self-linking

Self-linking is primarily used to highlight all of the points in a time series when any one is selected. It is the most common behavior that a user would use. When there are multiple time
series, it may also be useful to link to all points representing values recorded at a particular time.

Data underlying multiple time series, as for most of the other plots available in cranvas, are usually in “wide data” format (Table 3.3). One row contains the values recorded for a particular time, and aesthetic parameters are associated with each row. In this form if the display shows the multiple series, then when a user selects one point by brushing, all the points (values for V1, V2, V3) for this time are highlighted. This form is not conducive to selecting either a single point or an entire time series.

Table 3.3 Basic tabular form of data underlying plot (left), think of this as the “wide data” format. Each row contains values recorded at one time point. The aesthetic parameters are associated with one time point. Brushing on this form will highlight the points for a particular time (right), three points if all three series are drawn. It is probably more desirable for the behavior to be different: that selecting a single point will highlight all the values for that series, or only a single point, which can be achieved by a data re-structuring.

<table>
<thead>
<tr>
<th>Time</th>
<th>V1</th>
<th>V2</th>
<th>V3</th>
<th>.brushed</th>
<th>.color</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.1</td>
<td>27</td>
<td>11.9</td>
<td>FALSE</td>
<td>red</td>
</tr>
<tr>
<td>2</td>
<td>3.4</td>
<td>23</td>
<td>12.5</td>
<td>FALSE</td>
<td>blue</td>
</tr>
</tbody>
</table>

A more flexible format is provided by melting the data into the “long data” format (Table 3.4). In this format it is easy to realize brushing and self-linking in different ways: the user can select a single point, and (1) only this point is highlighted, (2) all points for that line (e.g. V1) are highlighted, or (3) all points for that time are highlighted (Figure 3.10). The latter two are achieved by treating the line group indicator or “Time” as a categorical linking variable, respectively.

The long data format is typically the basic format for longitudinal data, where there may be differing number time points per subject, and measured at different times. So this approach to implementing the brushing works here, too.
Figure 3.10  When a single point is brushed, there could be three modes of highlighting: (left) only a single point is highlighted; (center) all points for that series are highlighted, by treating the line group indicators as the categorical linking variables; (right) all points for that time are highlighted, by using the time as a linking variable.

Table 3.4  The “long data” format, which is called the melted form in Wickham (2007). This format allows a lot of flexibility. The “Variable” column can be used as a categorical linking variable, so that all points corresponding to “V1” are highlighted, when any one is selected, or a single point can be highlighted in the simplest brushing style, by changing the parameters of just that row. It would also be possible to use this form to use “Time” as a categorical variable for linking, and highlight all values recorded at a particular time.

<table>
<thead>
<tr>
<th>Time</th>
<th>Variable</th>
<th>Value</th>
<th>.brushed</th>
<th>.color</th>
<th>.brushed</th>
<th>.color</th>
<th>.brushed</th>
<th>.color</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>V1</td>
<td>3.1</td>
<td>TRUE</td>
<td>red</td>
<td>TRUE</td>
<td>yellow</td>
<td>FALSE</td>
<td>red</td>
</tr>
<tr>
<td>2</td>
<td>V1</td>
<td>3.4</td>
<td>TRUE</td>
<td>yellow</td>
<td>TRUE</td>
<td>yellow</td>
<td>TRUE</td>
<td>yellow</td>
</tr>
<tr>
<td>1</td>
<td>V2</td>
<td>27</td>
<td>FALSE</td>
<td>red</td>
<td>FALSE</td>
<td>red</td>
<td>FALSE</td>
<td>red</td>
</tr>
<tr>
<td>2</td>
<td>V2</td>
<td>23</td>
<td>FALSE</td>
<td>blue</td>
<td>FALSE</td>
<td>blue</td>
<td>TRUE</td>
<td>yellow</td>
</tr>
<tr>
<td>1</td>
<td>V3</td>
<td>11.9</td>
<td>FALSE</td>
<td>red</td>
<td>FALSE</td>
<td>red</td>
<td>FALSE</td>
<td>red</td>
</tr>
<tr>
<td>2</td>
<td>V3</td>
<td>12.5</td>
<td>FALSE</td>
<td>blue</td>
<td>FALSE</td>
<td>blue</td>
<td>TRUE</td>
<td>yellow</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
3.5.2 Linking between plots

Linking between plots builds a reactive brushing chain that when the data points on one plot are brushed, then they are highlighted on all the plots. In the normal cases of *cranvas*, data behind the plots is unique, so the brush interaction will modify the parameter attached to the data and trigger the listeners of plots to highlight the corresponding part. However, linking between a time series plot and other plots is different, because the data to create the time plot is in the long data format, while the data to create the other plots, like scatterplots or histograms, are in the wide data format. Hence, a link between the two data formats must be constructed.

Xie et al. (2014) delineated how to link two data objects. First a linking variable must be pointed out, then two listeners are attached on the two objects. If the `.brushed` parameter switches in one dataset, then the listener is triggered. And if any observations from the second dataset have the same value in the linking variable as the first dataset, then the corresponding `.brushed` parameter in the second dataset will be changed.

Note that the linking between wide data and long data is not a one-to-one linking. “Time” is the linking variable between two formats. In the direction from wide to long data, each entry in the wide data can project to multiple entries in the long data. In the opposite direction, an entry in the long data will map to one entry in the wide data. The unbalanced linking could produce a problem, as shown in Table 3.5.

This problem can be solved by cutting off the backward linking. To facilitate the cutoff, two signals are added respectively in the listeners of the two data objects. When one listener is triggered, the signal will be turned on until the listener finishes its work. During this period, the other listener cannot work. As in Table 3.5, the arrow from (b) to (c) will be cut off.

Figure 3.11 shows the linking between a longitudinal time plot, a map, and a histogram. The data come from the Google Flu Trends ([http://www.google.org/flutrends/](http://www.google.org/flutrends/)).
Figure 3.11  Linking between different plots for the Google Flu Trends data from November 24, 2013 – March 16, 2014. (Left) Time series faceted by state (see https://vimeo.com/112528131 for the transition to the facets), (top right) choropleth map grey scale indicating time of the peak in the series, light is earlier, (bottom right) histogram of the number of searches. Two states in the map are brushed, which highlights all flu searches from these states in the time series and histogram.
Table 3.5  Linking between the long and wide data format may produce a problem. Suppose we start from brushing a point (V1 at time 2) in the long data (a). Then the listener of the long data is triggered and changes the .brushed parameter for time 2 in the wide data (b). Then the listener of the wide data is triggered and switches the .brushed parameter for all the observations at time 2 in the long data (c), which will update (a) but make a conflict.

(a) Long data format, used in time plots

<table>
<thead>
<tr>
<th>Time</th>
<th>Variable</th>
<th>Value</th>
<th>.brushed</th>
<th>.color</th>
</tr>
</thead>
<tbody>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>2</td>
<td>V1</td>
<td>3.4</td>
<td>TRUE</td>
<td>blue</td>
</tr>
<tr>
<td>2</td>
<td>V2</td>
<td>23</td>
<td>FALSE</td>
<td>blue</td>
</tr>
<tr>
<td>2</td>
<td>V3</td>
<td>12.5</td>
<td>FALSE</td>
<td>blue</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>

(b) Wide data format, used in other plots

<table>
<thead>
<tr>
<th>Time</th>
<th>V1</th>
<th>V2</th>
<th>V3</th>
<th>.brushed</th>
<th>.color</th>
</tr>
</thead>
<tbody>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>2</td>
<td>3.4</td>
<td>23</td>
<td>12.5</td>
<td>TRUE</td>
<td>blue</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>

(c) Long data format, used in time plots

<table>
<thead>
<tr>
<th>Time</th>
<th>Variable</th>
<th>Value</th>
<th>.brushed</th>
<th>.color</th>
</tr>
</thead>
<tbody>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>2</td>
<td>V1</td>
<td>3.4</td>
<td>TRUE</td>
<td>blue</td>
</tr>
<tr>
<td>2</td>
<td>V2</td>
<td>23</td>
<td>TRUE</td>
<td>blue</td>
</tr>
<tr>
<td>2</td>
<td>V3</td>
<td>12.5</td>
<td>TRUE</td>
<td>blue</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>
3.5.3 Additional linking issues

Besides the issue of wide data and long data, there are other datasets created during the interactions that could produce some linking issues, such as the follows.

- The polygon data from the area layer. The area layer does not only need the values in the time series, but also the baseline to form many polygons. To draw the polygons, the data must be rearranged in an order of polygon vertexes. Each polygon is made of four vertexes: two from the time series and two from the baseline. Because the polygon layer should listen to the point and line layers, the link between the original data and the polygon data is one-way.

- Copies of the dataset during the incremental operation. When adopting the incremental procedure in Section 3.4.6, the variations of original dataset will be created with multiple stages of the interactions. However, these additional datasets are not required to get linked, because we only make the copies of the coordinates, not the properties. To use the coordinates, we combine them with the properties in the last minute, so there will not be any linking issues.

- Additional areas from the vertical faceting. The example of vertical faceting is shown in Figure 3.5 and Figure 3.12 explained how the interaction creates the additional areas. In Figure 3.12, the black dots are given by the time series, and the red dots are created during the cropping step of faceting, by the choice of cutting lines. The black and red dots are mixed in some order to form the shaded polygons. Whenever a point is brushed, one, two, or even more cropped polygons should be highlighted. Hence the two-direction linking between the cropped polygons and points should be constructed. Note that this is a one-to-n mapping, and the linking variable is the point ID, which should be assigned to the polygons when the red dots are generated.
3.6 Querying

In each of the examples, and plots shown the time axis is simply drawn using consecutive integers. This is necessary for convenience and generalizability. To know which actual time value requires querying, by mousing over the display, if labels have been set up in detail, the user can learn what day, month, year or individual identifier is under the cursor.

3.7 Conclusions and future work

This paper describes how interactions can be added to temporal displays by using sequences of affine transformations. This approach proves a rich variety of ways to slice and dice temporal data to explore seasonality, dependence, trends, anomalies and individual differences. These interactive temporal displays can be embedded in a large interactive graphics software system, enabling linking between plots to explore more general data where temporal components are just one aspect of many. Not everything is solved in terms of additivity, speed and interaction direction, but most common actions are possible for reasonably sized data.

Optimal aspect ratio for temporal displays is an unsolved problem, and is something that we have grappled with in the implementation. According to (Cleveland, 1993b) time series should be “banked” to 45°, which means that on average the angle of the lines, in comparison to the x- or y-axes is 45°. This is computed and used in the initial plots in cranvas, but as wrapping and faceting are conducted it probably should be re-calculated. Ideal integration of aspect ratio re-drawing with plot interactions could be examined in new work.
Future work would extend the interactions to include transforming between euclidean and polar coordinates, and between time and frequency domains. Earlier work in Dataviewer (Buja et al., 1988b) allowed users to interactively lag time series to generate lag plots to explore autocorrelation. This could be reasonably be accomplished similarly to the interactions described here.

Using data transformations to generate interactions is efficient but it assumes the components are ordinal. This is not necessarily true, for example, in the flu searches series (Figure 3.11) were faceted on the categorical variable state. This requires that states are first recoded to numerical values. In R, this is implicitly alphabetical order. However, because the implementation is created in R, the factor levels can be re-ordered easily, enabling the recoding to numerical value to be quite fluid. In the flu searches example, the states were re-ordered by earliest peak of searches.

This paper focused on interactive graphics for multivariate time series and longitudinal data, but the ideas should extend some to other temporal-context data.

**Acknowledgements**

This work was started with funding from Google Summer of Code, then partially supported by an unrestricted fellowship from Novartis, and by National Science Research grant DMS0706949.
CHAPTER 4. CRANVASTIME: AN INTERACTIVE TOOL FOR TEMPORAL AND LONGITUDINAL DATA PLOTS

4.1 Introduction

Figure 4.1 Examples of the static plots which have limitations in exhibiting the temporal data.

For the longitudinal and temporal data, using the static plots only may be insufficient to explore the trends, seasonality or unusual individuals, especially when the data set is large. Figure 4.1 gives three examples. In the top plot, the points are too crowded to reveal any
trend. The bottom-left makes the individuals of interest covered by others. In the bottom-right, the static plot does not help to find out the periodicity because it is hard to detect by eyes. In these situations, the interactive graphics may play an impressive role in exploring the data. For example, we can zoom in the long time series to see the details, or query a specific individual from the group, or wrap the series to check the period.

Cranvastime is such a tool that develops interactive plots for longitudinal and temporal data, using the design from the previous chapter. Longitudinal plots will show values for individuals, possibly at irregular time intervals. The purpose is to explore individual variation, differences and similarities. Time series plots tend to show the measurement on the time points, but there is implicitly an interest in studying seasonal trends, and temporal dependencies.

Cranvastime is a part of cranvas, an R package interfacing to the Qt libraries, to provide interactive and dynamic graphics for large data. It provides programmable interactive graphics in R, and encourages the users to bind the plots with analytical methods. cranvas has several basic plot types: scatterplot, histogram, barchart, density plot, mosaic plot, map, parallel coordinate plot and tours. The time plot can be linked to any other graph type, which supports the visualization of the longitudinal and temporal data even better, via a global or local view.

The installation instructions are provided on the homepage of cranvas: cranvas.org. More discussion about the package can be found at github.com/ggobi/cranvas/issues. Section 4.2 gives the R document of the function qtime(), which is the key function for cranvastime. In Section 4.3, all the data sets provided with cranvastime and several other data sets from the web and other R packages is listed, in order to make better demonstration in Section 4.4. Section 4.4 explains all the functionality of cranvastime by examples.
4.2 Usage

qtime  

Draw an interactive time plot

Description

Draw an interactive time plot.

Usage

qtime(time, y, data, vdiv=NULL, hdiv=NULL, shift=c(1,4,7,12,24),
   alpha=1, size=2, asp=NULL, fun.base=min,
   series.stats=ifelse(nrow(data)<1000,TRUE,FALSE),
   main=NULL, xlab=NULL, ylab=NULL, infolab=NULL, ...)

Arguments

time  the variable indicating time on the horizontal axis.
y  a vertor of all the variable names of interest.
data  Mutatframe data generated by qdata.
vdiv  a vector of variable names that will be used in faceting vertically.
hdiv  a vector of variable names that will be used in faceting horizontally.
shift  Wrapping speed selector. The default possible speeds are 1,7(for days a week),12(for months),24(for hours).
size  Point size, default to be 2.
alpha  Transparency level, 1=completely opaque, default to be 1.
asp  Ratio between width and height of the plot.
fun.base  a function to compute the baseline of the area plot.
series.stats  Whether to show the statistics which measure the similarity between series when wrapping. It gives the ACF, corr, and R square for one, two, and more series respectively.
main  main title for the plot.
xlab  label on horizontal axis, default is name of x variable.
ylab  label on vertical axis, default is name of y variable.
infolab  the variable(s) shown when identifying the points. Note that the x, y, and group information is already shown.
...

Details

Arrow up/down: in-/de-crease size of points.
Arrow left/right: wrap the time series in the initial mode, and drag the series in the series selection mode.
Shift + right: Time series will be folded directly to the width of maximal value in argument shift.
Shift + left: Time series will be backed to the original xaxis position.
Key ’+/−’: decrease/increase alpha level (starts at alpha=1 by default).
Key ’u’/’d’: separate/mix the series groups or the period by shifting them up and down.
Shift + ’u’/’d’: for multivariate y’s, separate/mix them by shifting up and down.
Key ’h’/’v’: horizontally or vertically faceting the series.
Shift + ’h’/’v’: horizontally or vertically mix the faceting panels.
Key ’g’: change the wrapping period circularly in the values of parameter ’shift’.
Key ’m’: Switch the mode for series selection. Default to be off. When there are more than one series in the plot, users can turn it on to hold a series and shift the series horizontally by dragging with the mouse.
Key ’y’: y-wrapping.
Shift + ’y’: y-wrapping backward.
Key ’r’: switch to the area plot.
Key ’f’: fold the series to a horizon plot.
Wheel: Zoom in/out. Then users can drag the series horizontally to see the details.

Value
A time plot.

4.3 Data

Seven data sets used in the examples of cranvastime are listed as follows. nasa, wages, and pigs are enclosed with cranvas. lynx, sunspots, Remifentanil are borrowed from other R packages. Google flu trends data is posted on the web. The code after the data set description show the usage of cranvastime and generate the graphs in Figure 4.2.


library(cranvas)
nasa2221 <- subset(nasa, Gridx == 22 & Gridy == 21)
nasa2221$Year <- factor(nasa2221$Year)
q.nasa <- qdata(nasa2221)
q.time(TimeIndx, ts, q.nasa)
**wages** Source: Singer and Willett (2003). It is a subset of data collected in the National Longitudinal Survey of Youth (NLSY) described at www.bls.gov/nls/nlsdata.htm. The data was collected to track the labor experiences of male high-school dropouts. The men were between 14 and 17 years old at the time of the first survey.

```
idtab <- table(wages$id)
idx <- names(idtab[idtab > 5])
qwages <- qdata(wages[wages$id %in% idx,])
qtime(exper, lnw, qwages, vdiv=c(black,id), hdiv=c(ged,hispanic))
```

**pigs** Source: Andrews and Herzberg (1985). It is a multivariate time series data on UK pig production during 1967 to 1978.

```
pig <- qdata(pigs)
qtime(TIME, c(GILTS,PROFIT,PRODUCTION,HERDSZ), pig, shift=c(1,4))
```

**lynx** Source: R package datasets. The lynx data gives annual numbers of lynx trappings for 1821–1934 in Canada. It was taken from Brockwell and Davis (1991) and originally from Campbell and Walker (1977).

```
lynx <- qdata(data.frame(Time=1:114, lynx))
qtime(Time, lynx, lynx, shift=1:13)
```

**sunspots** Source: R package datasets. The sunspots data gives monthly mean relative sunspot numbers from 1749 to 1983. It was collected at Swiss Federal Observatory, Zurich until 1960, then Tokyo Astronomical Observatory.

```
sun <- qdata(data.frame(Time=1:2820, sunspots))
qtime(Time, sunspots, sun, shift=c(1,c(1,6,7,13,26)*10))
```
Remifentanil Source: R package `nlme`. It studies pharmacokinetics and pharmacodynamics of remifentanil by recording the concentration at different time on 65 healthy volunteers after a continuous infusion. The demographics information of subjects is saved too.

```r
library(nlme)
Remi <- Remifentanil[complete.cases(Remifentanil) & Remifentanil$Time<=41,]
Remi$ID <- factor(Remi$ID)
qRemi <- qdata(Remi, color=Sex)
qtime(Time, conc, qRemi, vdiv=ID, infolab=c('Sex','Age','Ht','Wt'))
```

Google Flu Trends Source: [www.google.org/flutrends/us/data.txt](http://www.google.org/flutrends/us/data.txt). In the example we only used the US portion. The data records weekly flu search queries from September 28, 2003 to the most recent Sunday, aggregated by state.

```r
flu.data <- read.table("http://www.google.org/flutrends/us/data.txt", skip=11, sep="", header=TRUE, check.names=FALSE)[, c(1, 3:53)]
flu.melt <- reshape::melt.data.frame(flu.data, id.vars="Date")
flu.melt$Week <- as.integer(flu.melt$Date)
flu.melt$Date <- as.Date(flu.melt$Date)
colnames(flu.melt)[2:3] <- c("State", "FluSearches")
qflu <- qdata(flu.melt)
qtime(Week, FluSearches, data=qflu, vdiv=State, shift=c(1,52.2), infolab='Date')
```
Figure 4.2 Basic time plots created by cranvastime.
4.4 Functionality

A taxonomy of interactions and the corresponding pipeline for temporal data displays was introduced in the previous chapter. This section will focus on introducing the usage of the software and manipulation of the plots.

4.4.1 Discover the seasonality

4.4.1.1 Wrapping the series horizontally

For a time series with periodicity, an intuitive way to study the seasonality is shifting the observations from one period to match with another period. Then one can compare the patterns of two periods easily since any discrepancy will be conspicuous when lines are stacked.

In cranvastime, x-wrapping means moving the points at the end of a series (the right) to the beginning (the left) and forming a new series. The procedure is like to stretch the series, cut off the remainders that exceed the edge of the canvas, and align them to the left. In cranvastime, every hit on the right arrow will send one point to the left, and keeping pressing the key will wrap the series, until fully wrapped – three time points left. The x-wrapping procedure is invertible by hitting the left arrow. From left to right, Figure 4.3 shows the procedure.

Figure 4.3 Wrap the time series of nasa data. The period is quite regular, following a yearly cycle.
If wrapping one point each time is too slow for a long series, or if the seasonality period is confirmed, users can define the wrapping period. For example, the monthly temperature in the nasa data follows a period of 12. So we can set the function parameter `shift=c(1,12)` and switch the wrapping period between 1 and 12 on-the-fly by pressing key G. When the wrapping period is 12, then the procedure will be like Figure 4.4.

Figure 4.4  Wrap the time series with a larger period (12). The cycles fit well with each other.

Figure 4.5  Fully wrap the time series. The variation of temperatures in winter is larger than in summer.
When the time series have a fixed period, cranvastime allows the user to skip the wrapping steps and directly go to the fully wrapped plot (also called seasonal plot), by the keyboard manipulation Shift + right arrow (Figure 4.5). The period for the fully x-wrapping is set to be the maximal value in the argument \texttt{shift}. Resetting (Shift + left arrow) can be used at ANY status of x-wrapping.

### 4.4.1.2 Statistics to measure the similarity

The similarity of series in the current picture is defined by the autocorrelation function (ACF) when the series is not wrapped, and by correlation or R square when wrapped.

ACF is a useful statistic to measure the similarity between observations in terms of the time separation and to find the repeating patterns. In cranvastime, ACF is calculated for each time series, and the time lag is given by the wrapping period. Hence, in Figure 4.2 (1), ACF is 0.65 with lag 1, shown on the bottom-left; while ACF in Figure 4.4 (top-left) is 0.57 with lag 12. If we set more candidate periods in argument \texttt{shift}, then switching the wrapping periods will provide ACF with different lags.

When the the time series is wrapped into two lines – an original and a wrapped series, like Figure 4.3 (center) and Figure 4.4 (panel 2,3,4 by row), the correlation between two lines are computed. The number will be big if the wrapped series and original series match well with each other, and small if they do not fit well.

As wrapping goes on, more than two series will appear in the plot, like Figure 4.3 (right) and Figure 4.4 (panel 5,6). Then another statistic, the R square is present. Figure 4.6 shows the way we compute R square. Regardless to how many series there are, or whether the series are complete over time, we treat the time (x axis) as groups, and compute the sum of squares between-group (SSR), sum of squares in-group (SSE), and the total sum of squares (SST). Hence R-square is calculated by SSR/SST.
Figure 4.6  The calculation of R squares. The blue boxes separates the groups, and the thick blue line in the middle of each box is the mean of that group. The left panel is fully wrapped, with 6 series, the R-square (0.78) is closer to 1. The right panel is not wrapped well, with one of the series (the red line) not complete over time. Its R-square (0.43) is much smaller than the left plot.

4.4.2 Explore the individual variation

In many situations, users care about some individuals more than others, for instance, the patients under a special treatment, or the climate observations in a year when the earthquakes frequently happened. Various circumstances bring different definitions to “individual”. In cranvastime, an individual is a series, so it could be (1) observations in one period of a univariate time series, like the series from Figure 4.7 (center); (2) one variable in multivariate time series, like the series in Figure 4.2 (3). (3) one individual in a longitudinal dataset, like the series in Figure 4.2 (6).

4.4.2.1 Faceting the individuals vertically

A regular time plot usually mixes all the individuals together. But cranvastime could split them from each other by the keyboard manipulation. However, the plots need some special settings before the separation.
**Time periods** Faceting periods only works when there is a single time series and x-wrapping is applied. This is because when the plot has multiple time variables or longitudinal series, the faceting functionality aims to separate the variables or individuals. Faceting both on period and variable/individual makes the graph confused. We can use the Iowa state from Google Flu Trends as an example. The code below are written following Section 4.3, and will give Figure 4.7 (left). To compare the flu searches over years, we set the period length to be 52.2 in the argument `shift`. Figure 4.7 (center) is the fully wrapped series, and then pressing key U will give Figure 4.7 (right). Pressing key D could mix the periods back together.

```r
qIowa <- qdata(subset(flu.melt, State=='Iowa'))
qtime(Week, FluSearches, data=qIowa, shift=c(1,52.2), infolab='Date',
series.stats=FALSE)
```

Figure 4.7 Univariate time series with a non-integer period. After fully wrapping, the start of each series is the last Sunday in September. The gray scale from dark to light represents year 2003 to 2014. Faceting by period makes it easy to see the 6th and 7th lines from the bottom are quite irregular compared to other lines, which reflect the outbreak of swine flu.

**Multiple variables** If the dataset contains multiple variables on the same time line, we may have questions like whether the variables share the same period? Do the peaks and valleys match with each other? Cranvastime can handle multivariate time series, by putting all the variables in argument `y`. For example, Figure 4.2 (3) contains four variables from the pigs
data. All the variables are auto-scaled between 0 and 1 when drawing the plot. The ACF for all variables are present. Pressing Shift+U will facet the variables, and the result is in Figure 4.8. The labels for y-axis and the ACF indicators also shift with the lines.

![Time Plot of TIME And GILTS, PROFIT, PRODUCTION, HERDSZ](image)

Figure 4.8 Facet by variable for the pigs data. PRODUCTION shows a lag pattern compared to HERDSZ. The trend of GILTS is about two quarters later than the trend of PROFIT.

**Longitudinal individuals** To identify the longitudinal groups, at least one of the parameters `vdiv` and `hdiv` is needed. In the Remifentanil example, setting `vdiv=ID` will generate Figure 4.2 (6). Same as the manipulation to facet/mix the period groups, key U/D are used. Figure 4.9(left) is given by pressing U once, and keeping pressing U will get Figure 4.9(right), where all the groups are completely separated.

**4.4.2.2 Shifting the individual over time**

For some reasons, like, the starting time of groups are not the same, or a variable is leading the other, we may consider to shift one or more groups horizontally to match with other groups. In cranvastime, we can switch to “series mode”, where a series could be selected, and then shifted by pressing keys or dragging the cursor.
Figure 4.9  Facet by longitudinal individual. For most groups conc increases quickly then drops in a convex curve. A few groups have significantly different magnitudes.

Figure 4.10  Shift a variable over time for the pigs data to check the dependence. After shifting GILTS to the left by 2 quarters, the peaks and troughs of GILTS and PROFIT are matched much closer, which shows that the number of GILTS was heavily influenced by PROFIT.
In the pigs example, based on Figure 4.8, the trend of GILTS is about 2 quarters later than PROFIT. To check that, we first switch to the series mode by pressing key M, then move the cursor on top of GILTS to hook the series, as shown in 4.10 (left). By pressing the left arrow twice, we can shift GILTS to the left by 2 time points, as Figure 4.10 (center). Then moving the cursor away we will get the series un-highlighted, as Figure 4.10 (right).

### 4.4.3 Focalize to the center of y

When people looking at a time series plot, their attention could be easily attracted to the global or local extreme values, and to some special trends or sudden changes over time. The centers of y variables, like the mean, median, or mode, and the values around them, are packed between the extreme values and usually not connected over time, so sometimes they are unconsciously ignored. We use the following interactions to expand the space on y variables, and get the center of y more focalized.

#### 4.4.3.1 Mirroring

Mirroring means the observations under the series mean (or other center measurements) are mirrored up. Pressing key F will switch between the left and center panels of Figure 4.11. In the center panel, points above mean turn blue, and points below mean are mirrored and turn yellow. Figure 4.11 (right) resets the y-axis of the time series in the center panel, to reduce blank under the mean. Pressing Shift + F will switch between the left and right panels.

#### 4.4.3.2 Wrapping the series vertically

The vertical wrapping (or y-wrapping) slices the series by a horizontal line and shifts the big observations above the line to the bottom. After shifting, the overlapped area will have a deeper color, so the color saturation would reflect the values. In Figure 4.12, line 1 is generated from Figure 4.11 (left) by pressing key Y once. Keeping pressing Y will give line 2. Y-wrapping can also be applied after mirroring, which gives a plot called horizon graph. Figure 4.12 line 3 is generated from Figure 4.11 (right) by pressing key Y once, and line 4 will be obtained after pressing Y for a few times. The procedure is invertible by pressing Shift + Y.
Figure 4.11  Mirror the lynx data by the mean. We see the mean is closer to the bottom than the top, peaks are sharp and irregular, and the valleys are smooth and regular over years.

Figure 4.12  Vertically wrapping the lynx data before (first 2 rows) and after (last 2 rows) mirroring. The wrapped series make comparing values from different time easier, as the y-axis is zoomed in.
Y-wrapping could be more useful when there are multiple series, either from multivariate time series, or longitudinal data. By cranvastime, different series share the same wrapping parameter, so that values from different series at different wrapping line groups are comparable. Figures 4.13 and 4.14 show two examples for multivariate time series and longitudinal data.

![Figure 4.13](image)

Figure 4.13  Y-wrapping the four variables in the pigs data. The left panel is the area plot of Figure 4.8. Pressing key Y once will give the center panel, and keeping pressing Y will generate the right panel.

The code below continue from the previous code in the Google Flu Trends data to get a subset.

```r
flu.melt$State <- tolower(flu.melt$State)
flu2015 <- subset(flu.melt, Week>572)
flusearch <- sort(tapply(flu2015$FluSearches, flu2015$State, function(x) which(x>(max(x)/5*3))[1]))
ord <- names(flusearch)
flu1415 <- subset(flu.melt, Week>523)
flu1415$State <- factor(flu1415$State, levels=ord)
qflu1415 <- qdata(flu1415)
qtime(Week, FluSearches, data=qflu1415, vdiv=State, shift=c(1,52.2), infolab='Date')```
Figure 4.14  Y-wrapping the Google Flu Trends data by state. Two winters (Fall 2013 - Spring 2015) are picked. The states are ordered by the time when the flu searches in winter 2014-2015 arrived 3/5 of the maximum searches. The left panel fully separates the series by state, but makes the vertical space for each state very limited. The right panel after y-wrapping expands the vertical space, so we see more details between the two peaks. It is easier to compare values, for example, from the right panel, we can read that the maximal value of North Carolina is greater than that of West Virginia, because there is an obvious third wrapping line group for North Carolina, but only two wrapping line groups for West Virginia. However, from the left panel, it is very difficult to tell.
4.4.4 Connect time series to other variables

Time series are sometimes associated to important variables that are not timely measured, like the covariates in Pharmacokinetic/Pharmacodynamic data, or the spatial information in spatio-temporal data. Cranvastime provides two ways to connect time series with other variables: faceting and linking.

4.4.4.1 Faceting by covariates

Different from faceting the individuals, faceting by covariate could separate the series both horizontally and vertically, and it does not have any intermediate steps – pressing key H or V will split the series completely.

In the Remifentanil example, we can create another three categorical variables (age, height, and weight) by the code below. The argument \texttt{vdiv} lists the categorical variables that will be used to vertically separate the series, and \texttt{hdiv} to horizontally split the series. When there are more than one variable in \texttt{vdiv} or \texttt{hdiv}, multiple levels of faceting are applied. Pressing key V once will split the series by one variable (\texttt{age}, the first one in \texttt{vdiv}), as Figure 4.15 (1,1) to (2,1). Pressing key H once will split the series horizontally by one variable (\texttt{Sex}, the first one in \texttt{hdiv}), as Figure 4.15 (1,1) to (1,2). If users want to facet by other variable than the first one in \texttt{vdiv} or \texttt{hdiv}, they can press key P to rotate the variables in \texttt{vdiv}, make \texttt{vdiv=c(wt,ID,age)}, and use the current first variable \texttt{wt}, as Figure 4.15 (2,1) to (3,1); or press key O to rotate \texttt{hdiv} to \texttt{c(ht,Sex)}, as Figure 4.15 (1,2) to (1,3). Pressing key V for another time when \texttt{vdiv=c(age,wt,ID)} will make the vertical faceting depending on two variables (\texttt{age} and \texttt{wt}, the first two in \texttt{vdiv}), as Figure 4.15 (2,1) to (4,1). Pressing key H for another time when \texttt{hdiv=c(ht,Sex)} will make the horizontal faceting depending on two variables, as Figure 4.15 (1,3) to (1,4).

The faceting and rotating could be applied at any status, so we have other graphs in Figure 4.15. Shift + V and Shift + H are utilized to mix the series by removing one variable out of the current faceting variable(s).

\begin{verbatim}
Remi$age <- cut(Remi$Age, c(18,40,65,90))
\end{verbatim}
Remi$ht <- cut(Remi$Ht, c(153,163,173,183,193))
Remi$wt <- cut(Remi$Wt, c(40,70,80,110))
qRemi <- qdata(Remi, color=Sex)
qtime(Time, conc, qRemi, vdiv=c(age,wt,ID), hdiv=c(Sex,ht))

Figure 4.15  Faceting the Remifentanil data by four covariates: sex, age, height, and weight.

The nasa data have two variables indicating the grid locations. Faceting by the two variables will create a glyph map that displays the time series spatially. In Figure 4.16, the top-left panel is given by the following code. Then pressing key H will go to the bottom-left panel, pressing key V will go to the top-right panel, and pressing both H and V will give the bottom-right panel.
nasa$Gridx <- factor(nasa$Gridx)
nasa$Gridy <- factor(25-nasa$Gridy)
nasa$Year <- factor(nasa$Year)

qnasa <- qdata(nasa)

qtime(TimeIndx, ts, qnasa, hdiv=Gridx, vdiv=Gridy, shift=c(1,12), asp=1)

Figure 4.16  Faceting the nasa data by grid. The top-left panel shows different trends and variations of the series, and the bottom-right panel demonstrates that the big variation is from the land area and the flat trend is almost from the ocean area.
4.4.4.2 Linking with other plots

Linking the time plot with other plots could reveal the association as a solution to high-dimensional visualization, and check some observations of interest. Figure 4.17 links the longitudinal data with US map and a histogram. Any selection from one plot will be passed to all other plots.

![Figure 4.17](image)

The longitudinal plot of Google Flu Trends is linked with the US continental map, and a histogram of flu searches. The gray scale in the map matches with the order of states in the longitudinal plot. From the map we see the color (flu season) is lighter (earlier) in the middle south and southeast, and darker (later) on the west coast and northeast. Two states are selected: the two flu seasons in Louisiana were much earlier than Rhode Island, but the flu search trend had a long thick tail in Rhode Island in spring 2014.
qstate <- map_qdata('state')
qstate$.color <- paste0('gray', 20-flusearch[qstate$labels], '0')
qmap(qstate)
link_cat(qstate, 'labels', qflu1415, 'State')
qhist(FluSearches, data=qflu1415)

4.4.5 Miscellaneous

4.4.5.1 Switching between areas and lines

Line plots and area plots are two types of visualization on time. The line plots are concise when comparing the series in the same scales. But when the series are facetted, the area plots are better, because they make the comparison more accurate between values from different series. In cranvastime, we can use key R to switch between the two modes. Figure 4.18 (left) is the area mode of Figure 4.7 (right). Figure 4.18 (right) is the area version of Figure 4.8.

![Figure 4.18](image-url)  
Figure 4.18 Switching to the area plots from line plots. When the series are facetted, the area plots makes it easier to compare the y values from different series.
4.4.5.2 Zooming and panning

For the long series drawn in a small window, local features are easily covered by the intensive points. Zooming in is a way to solve the problem. By rolling the wheel of the mouse, users can zoom in/out from the cursor’s location, as shown in Figure 4.19.

Figure 4.19 Zooming in the time plot makes the change look gently, while zooming out makes the spikes look sharper.

To see the other part of the series at the zooming level, user can drag the series by mouse. Figure 4.20 displays three slices during panning. In this mode the icon will change to the hand-grabbing shape.

Figure 4.20 Panning the local series. The local features look similar when panning.
4.4.5.3 Resizing and transparentizing

The size and transparency of the points and lines can be changed by some keyboard manipulations. The Up/Down arrow will in/decrease the size of points and lines (Figure 4.21 left/center). The plus/minus key will make the points and lines darker/lighter (Figure 4.21 center/right).

Figure 4.21 Different sizes and transparency of the points. The left panel looks concise, but the center one marks and emphasizes the exact positions of the observations. When the plot is the transparent as the right panel, we are able to see the overlapped places.

4.4.5.4 Labeling the point of interest

The label, includes the values of x / y axes, and the indicators provided by the argument \texttt{infolab}, will show up when the cursor is close to the point of interest. Whether they are 'close' enough depends on the size of points. For Figure 4.21(left), it is hard to identify the points, but for the center/right panel, it is quite easy to get a label when moving the mouse. In the regular situation, the label tag is put at the right and below of the point. But when the point is near the boundary of the canvas, the position of label will change (Figure 4.22).
4.5 Conclusions and future work

In summary, cranvastime adds the interactive longitudinal and temporal plots to cranvas, that makes it possible to wrap, facet, mirror, shift, zoom, pan, brush, and identify the time series dynamically. It can also modify the wrapping period and rotate the faceting variables on-the-fly. These implementations are useful to analyze seasonality, compare individuals, and explore the time series. In addition, it is helpful for some statistics analytical goals, like periodicity diagnosis, periodic similarity measurement, leading time detection for multivariate series.

Current challenges for cranvastime include (1) how to choose the optimal aspect ratio, not only for the original time plots, but also for the wrapped and facetted plots, (2) how to order the individuals when facetting, (3) how to handle the missing data, which often exist in time series, (4) how to optimize the computation for y-wrapping, which currently employs too many loops when there are many series and the series are wrapped deep.

In future we plan to add the following functionality to cranvastime: (1) transformation between the euclidean and polar coordinates, (2) linking between the time and frequency domains, (3) the interactive tools for missing data imputation, (4) interactive lag plots to explore auto-correlation, (5) similarity computation for multivariate time series and longitudinal data.
CHAPTER 5. NEW APPROACHES TO EVALUATE CONTIGUOUS CARTOGRAMS FOR AREAL DATA

5.1 Introduction

Areal data, as a type of spatial data, is cumulatively or aggregately measured by region. It is commonly collected and analyzed in the fields of economics, politics, public health, social science, etc. For example, to examine the US electoral trends in electing a president, the basic unit is a state, because states votes are added as blocks to give a total count for each candidate. And when exploring the global economic growth, data are practically provided by country.

Geographical visualization will intuitively improve human’s understanding of spatial data. For areal data, choropleth maps is often utilized (Figure 5.1). However, those maps usually attract people’s attention on the large regions, and make people ignore the small but crucial areas. To emphasize the important areas, cartogram, a variation of the choropleth map, was developed. A cartogram reshapes the regions in proportion to a numerical thematic variable, while maintaining spatial integrity as much as possible. Three terms are important to measure a cartogram: area, shape, and topology. The areas of regions should represent the values from the thematic variable. The global and regional shapes should be preserved to identify regions. The topology, i.e., how to geometrically joint the regions to a map, should also be kept for readability.

5.1.1 Non-contiguous cartogram

Cartograms can be generally classified into two types: non-contiguous and contiguous cartograms. For non-contiguous cartogram, the shared edges between two adjacent regions are
Figure 5.1  A choropleth map for 2012 US presidential election on US continent. The blue states voted for the Democratic nominee Barack Obama, and the red states voted for the Republican nominee Mitt Romney. It is hard to predict who won the election based on the areas of two colors.

not kept. Regions are rescaled discretely and then moved to avoid overlapping (also called collision problem) as much as possible. During the movement, the original topology could be abandoned. Figure 5.2 shows two non-contiguous cartograms for the 2012 US presidential election. The left panel is a Dorling cartogram invented by Dorling (1996), and the right panel is the projector method proposed by Olson (1976). Several later non-contiguous methods, like RecMap (Heilmann et al., 2004) and Rectangular cartograms (van Kreveld and Speckmann, 2007), made much effort on re-arranging the region positions with some constraints.

Figure 5.2  Non-contiguous cartograms for 2012 US presidential election. (Left) Dorling cartogram, that uses circles or other simple shapes like squares or hexagons to represent areas. (Right) Projector method, that keeps the original shape of the states.
However, the non-contiguous cartograms have some limitations. The topology is disrupted and the neighborhood structure is broken by the gaps between polygons. The movement of polygons to eliminate overlapping increases the computational complexity and sometimes increases the topological error (Figure 5.3). The third problem is that in many applications the polygons maybe concave, and there is little research to detect or solve concave polygon collisions.

![Movement that switches the locations of two circles C and D: (left) original Dorling plot, (right) after moving according to attraction and repulsion forces, C and D are switched. Even after many iterations, they can not be switched back.](image)

5.1.2 Contiguous cartogram

Unlike the non-contiguous cartogram, contiguous cartograms maintain the spatial integrity, by keeping the common vertices and edges shared by the same regions before and after the transformation. The locations of vertices are changed to approach the new areas to the target thematic variable, but the neighborhood remains the same. There are quite a few algorithms that have been developed over a few decades. These can be grouped by the following principles.

- Lay a rectangular grid on top of the original map, assign densities to the grid intersections, then apply some methods to find the new locations of grid intersections, finally predict the vertices of regions by the quadrilateral. The algorithms include rubber map method (Tobler, 1973), pseudo-cartogram method (Tobler, 1986), diffusion-based method (Gastner and Newman, 2004), and carto-SOM (Henriques et al., 2009).
• Move the vertices by the sum of expansion forces. Depending on the radiator/direction of
the forces, there are some variations on this approach. Density equalized map projection
(Selvin et al., 1988) and rubber sheet distortion method (Dougenik et al., 1985) use
the centroids of regions to calculate the directions of forces. The line integral method
(Gusein-Zade and Tikunov, 1993) applies the radial transformation within a grid of small
cells. A constraint-based approach (Kocmoud, 1997) exerts the forces to the vertices on
simplified edges, and the direction bisects the angles. The CartoDraw (Keim et al., 2004)
and medial-axis-based method (Keim et al., 2005) adopt scanlines to find the rescaling
directions.

• Preserve the global shape of the map, and trade the internal areas between regions to
approach the optimal size. The cellular automate machine method (Dorling, 1996) sets up
the map on a grid and trades the areas cell by cell. The morphology-based method (Sagar,
2014) simulates a flood propagation process among centroids to obtain some balance of
area trading.

• The interactive polygon zipping method (Torguson, 1990) is different from other methods
because it rescales the regions independently and then fills the gap between adjacent
regions by “zipping up” the borders.

5.1.3 Software

Although there are a lot of methods for both non-contiguous and contiguous cartograms,
not many can be found available in software. This may be due to lack of detail in documenting
the computational steps of the algorithms or the computational complexity. The following is a
list of available methods in software.

**Diffusion cartogram** This algorithm is implemented in C, C++ and Java. The original al-
gorithm by Michael Gastner and Mark Newman, was implemented in C in the cart
library. This was ported to R in the package Rcartogram (Lang, 2008). The Cartogram
Geoprocessing Tool written in C++ applied the algorithm to support the ESRI and
ArcGIS software. ScapeToad is a standalone Java software for this algorithm.
Rubber sheet distortion method  This algorithm is implemented in JavaScript and Java. It is a relatively intuitive algorithm and clearly described in Dougenik et al. (1985) with pseudo code. cartogram.js is a JavaScript implementation, and MAPresso is a Java applet, although it is not currently supported.

Dorling cartogram  Dorling’s circle cartogram is JavaScript implemented in Protovis and d3.js. MAPresso also provided this method in Java. It is a very aesthetically appealing cartogram.

5.1.4 Visual challenges

The greatest challenge for cartograms is the trade-off between the area and shape. Making the area of a polygon proportional to the thematic variable is the first principle in cartogram construction. However, the cartogram transformation has difficulty preserving both shape and topology to match the new area. This limitation distinguishes the different types of cartograms. Non-contiguous cartograms abandon the topology, and the Dorling cartogram even drops the shape. Contiguous cartograms preserve the topology by zipping the adjacent regions together, and by result abandon the shape. Preserving shapes enables better readability because it builds on the viewer’s familiarity of shape of their country, from early childhood schooling. But preserving shape sacrifices the accuracy of the area representation.

Another challenge is the comparison of different cartograms. Even with the same algorithm, the result may change substantially with the use of different parameters. Figure 5.4 shows two diffusion cartograms, with different density parameter settings. Density is the value of thematic variable given to the blank space around the map. The left panel uses a small density resulting in a map that overflows the canvas and looks fat. The right panel has a larger density value, so the blank part squeezes the map and leaves several long “tails”. It is difficult to judge which one is a better visualization, since both have distorted the shape in order to make the area size close to the desired value.

This research provides ways to evaluate cartogram results, by examining error in shape and area. The diffusion cartogram is used for the analysis. Section 5.2 describes the evaluation
function that measures area and shape accuracy. Section 5.3 explores the interactive control of parameters that can be applied to the diffusion cartograms to enhance the readability. Section 5.4 shows the results from a simulation study of diffusion parameters in relation to shape and area accuracy, that can help determine optimal choices.

Figure 5.4 Diffusion cartograms for 2012 US presidential election on US continent. Different parameter settings will create quite different cartograms.

5.2 Evaluating cartograms

Comparing two cartograms needs to be both numerical, and perceptual. This section focuses on numerical assessment. A perceptual experiment would focus on how accurately a reader could compare areas or find geographic entities in the cartogram.

5.2.1 Existing work

The early research on cartograms focused more on the algorithms rather than evaluations. Kocmoud (1997) compared eight methods (including contiguous and non-contiguous cartograms) by seven characteristics. Other work includes iteration stopping rules (Kocmoud, 1997), objective functions (Keim et al., 2004), and result checking (Gastner and Newman, 2004; Sagar, 2014). Evaluation functions focused on area and shape error.

The area error had two forms. Kocmoud (1997) and Gastner and Newman (2004) used Equation 5.1, leading to a range of $(-1, +\infty)$. Keim et al. (2004) and Sagar (2014) used Equation 5.2, with a range of $[0, 1)$.

$$\text{area error}_j = \frac{A_{\text{desired}}^j - A_{\text{carto}}^j}{A_{\text{carto}}^j}$$  \hspace{1cm} (5.1)
area error \_j = \frac{|A\_j^{\text{desired}} - A\_j^{\text{carto}}|}{A\_j^{\text{desired}} + A\_j^{\text{carto}}} \tag{5.2}

where \(j\) is the index for a single region, and \(A\_j^{\text{desired}}\) and \(A\_j^{\text{carto}}\) are the scaled areas for region \(j\), given by

\[A\_j^{\text{desired}} = \frac{\text{Area}_{j}^{\text{desired}}}{\sum_i \text{Area}_i^{\text{desired}}},\]

and

\[A\_j^{\text{carto}} = \frac{\text{Area}_{j}^{\text{carto}}}{\sum_i \text{Area}_i^{\text{carto}}}.

The shape error can be defined in a number of different ways with different measurements. Kocmoud (1997) calculated the “average angle error” by the difference of angles before and after the transformation on the simplified polygon. Keim et al. (2004) utilized the Euclidean distance of a \(d\)-dimensional Fourier space as the shape error, while the Fourier coefficients are determined (without much difficulty) by approximating a squared wave function, which describes a smoothed curvature of the normalized polygon. Sagar (2014) used a similar function (Equation 5.3) as its area error function, where \(SI\) is a shapiness index function that measures the cardinality of a 0-1 matrix that allows shape operation mapped to the polygon.

\[\text{shape error}_j = \frac{|SI(\text{Region}_{\_j}^{\text{eap}}) - SI(\text{Region}_{\_j}^{\text{carto}})|}{SI(\text{Region}_{\_j}^{\text{eap}}) + SI(\text{Region}_{\_j}^{\text{carto}})} \tag{5.3}\]

5.2.2 Problems

The area error and the shape error can not be minimized simultaneously (Section 5.1.4). A combination of the two is desirable, but the current area and shape error functions proposed above are rooted in quite different ideas, making it difficult to combine them.

For shape error, the existing evaluation functions depend on a simplified polygon or matrix, to some extent, but the degree of polygon simplification and the effect of the simplification is not discussed.

Another issue is how to combine the errors from many regions on the map. The current methods compute the errors based on every single region, then simply sum up the errors to get a global area error or a global shape error. However, the distortion of big regions impacts the
readability differently from the distortion of small regions. Also, for different applications, the importance of regions might vary.

5.2.3 Proposed evaluation functions

This section proposes a new evaluation method for contiguous cartograms. The basic idea is similar: the shape error and area error are calculated respectively by region, then a mixed error is obtained. Below are details of the approach.

5.2.3.1 Shape error

The data of any independent region are the $x/y$ coordinates for vertices in the order of the connection. When a region has more than one polygon, such as the Michigan state, then the error should be computed on each polygon and summed up. Note that the diffusion cartogram transformation only moves the vertices and not changes the number of vertices or edges, so the comparison can be made on the vertices and edges one after another. The problem turns to how to measure the similarity of two polygons with the same number of vertices. Here is a list of things we should consider:

- Angle difference. No angled change is a necessary condition of claiming two similar shapes. For an arbitrary vertex, the angle difference can vary between $[0, 2\pi)$. Figure 5.5 gives two examples, the left panel shows two similar shapes with no angle difference, the right panel contains one vertex $A$ that the angle difference is close to $2\pi$.

![Figure 5.5](image)

Figure 5.5 The angle difference can vary between $[0, 2\pi)$. (Left) Two similar rectangles. (Right) Only one vertex locates differently, and the angle difference is nearly $2\pi$. 
• Edge difference. No angled change is not a complete condition to claim two similar shapes. Figure 5.6 shows an example with the same angles for each vertex, but the two shapes are not similar. Another condition is that the length ratio of any paired edges must remain the same. Hence for every vertex, the ratio between two adjacent edges can be used to measure the edge difference. For instance, in Figure 5.6, if $A_1B_1/A_1D_1 = A_2B_2/A_2D_2$, then there is no edge difference between vertices $A_1$ and $A_2$.

![Figure 5.6](image)

Figure 5.6 Two polygons may not be similar even if the angle difference is 0.

• A mixture of the angle and edge difference. Obviously we can not add an angle on a ratio. But the ratio can be projected to $[0, \frac{\pi}{2})$ by an $atan$ function. Hence the two parts are possible to be mixed.

• Weight of the vertices. For real geographical data, the vertices of a polygon could be dense when they are marked along a twisty river or coast. But the dense vertices are not as important as the sparse ones, because the sparse vertices govern longer border. Hence the weight of a vertex should be determined by the length of adjacent edges.

The shape error for a region $j$ is defined as follows. Let $P_j$ denote the number of polygons belonging to this region. For $i = \{1, 2, \ldots, P_j\}$, let $E_{ji}$ be the number of edges of polygon $i$ in region $j$. Since $\#$ edges = $\#$ vertices for any polygon, $E_{ji}$ is also the number of vertices of
polygon $i$ in region $j$. The regional shape error is the weighted sum of angle and edge difference, as shown in Equation 5.4.

$$\text{shape error}_j = \sum_{i=1}^{P_j} \sum_{k=1}^{E_{ji}} \left( \text{angle diff}_k + \text{edge diff}_k \right) \times \frac{L_{k-1}^{\text{map}} + L_k^{\text{map}}}{2},$$

(5.4)

where $L_k^{\text{map}}$ is the length of edge connected vertices $V_k$ and $V_{k+1}$ on the original map. Note that as a circle, $V_0 = V_{E_{ji}}$ and $V_{E_{ji}+1} = V_1$. The angle difference is given by

$$\text{angle diff}_k = |G_k^{\text{map}} - G_k^{\text{carto}}|$$

(5.5)

where $G_k^{\text{map}}$ is the angle of vertex $V_k$ on the original map, and $G_k^{\text{carto}}$ is the angle of $V_k$ on the cartogram. We have angle diff$_k \in [0, 2\pi)$. And the edge difference is given by

$$\text{edge diff}_k = \text{atan} \left( \max (\frac{L_k^{\text{map}}/L_k^{\text{carto}}}{L_{k+1}^{\text{carto}}/L_k^{\text{map}}}, \frac{L_k^{\text{carto}}/L_{k+1}^{\text{carto}}}{L_k^{\text{map}}/L_{k+1}^{\text{map}}}) - 1 \right)$$

(5.6)

where $L_k^{\text{carto}}$ is the length of edge connected vertices $V_k$ and $V_{k+1}$ on the cartogram. The range of an edge difference will be in $[0, \frac{\pi}{2})$.

An argument may arise that the range of angle difference is four times of edge difference. Actually, due to the algorithm for diffusion cartograms, it is very unlikely to obtain a transformed vertex with the angle difference greater than $\pi$. Figure 5.7 shows how the shape-influential vertices are captured by the shape error functions.

### 5.2.3.2 Area error

The area error for a region is easy to get by Equation 5.1 or 5.2, but those are not at the same scale as the shape error. To enable the additivity of shape and area errors, Equation 5.7 is employed, where $A_j^{\text{desired}}$ and $A_j^{\text{carto}}$ are defined as above.

$$\text{area error}_j = \sum_{i=1}^{P_j} \sum_{k=1}^{E_{ji}} \left( \text{atan} \left( \max (\frac{A_j^{\text{desired}}}{A_j^{\text{carto}}}, \frac{A_j^{\text{carto}}}{A_j^{\text{desired}}}) - 1 \right) \right) \times \text{weight}_k$$

(5.7)
Figure 5.7  Angle and edge difference for the vertices from the Iowa state before and after the diffusion cartogram transformation in Figure 5.4 (right). The first column shows the scatterplots between original angle and edge difference. The values are all less than $\frac{\pi}{2}$. The second column shows the difference weighted by edges. The colored points are the vertices that have large values either on the weighted angle difference (blue squares), or on the weighted edge difference (green triangles), or on both (red circles). The color thresholds for the three rows are 1%, 5%, 10% respectively. The right two columns show where are the greatly changed vertices.

In Equation 5.7, the ratio of the desired area and cartogram area is projected to $[0, \frac{\pi}{2}]$, then the average of two edge lengths of each vertex is used as weight, i.e., the scale. Since the ratio between areas does not depend on the edges or vertices, the formula can be simplified to

$$\text{area error}_j = \tan \left( \max \left( \frac{A^\text{desired}_j}{A^\text{carto}_j}, \frac{A^\text{carto}_j}{A^\text{desired}_j} \right) - 1 \right) \times \frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{P_j} \sum_{k=1}^{E_i} L^\text{map}_k$$

(5.8)

where $n$ is the number of regions on the map, and $\frac{1}{n} \sum_{j=1}^{n} \sum_{i=1}^{P_j} \sum_{k=1}^{E_i} L^\text{map}_k$ is the average perimeter of $n$ regions.
5.2.3.3 Total error of a cartogram

The association between the shape and area error can be found in Figure 5.8. The magnitudes of two errors matches well due to the similar design of the error functions. Hence Equation 5.9 is considered as the total error function for a diffusion cartogram. For Figure 5.4, the result is shown in Table 5.1, the total error of the left cartogram is larger than that of the right one, because the left plot distorted the shape more.

\[
total\ error = \sum_{j=1}^{n} (\text{shape\ error}_j + \text{area\ error}_j) \tag{5.9}
\]

Figure 5.8 Scatterplots between the shape error and area error, for the two diffusion cartograms in Figure 5.4. Overall the two errors are negatively associated. The regions in the top left corners are mostly the small states with large area errors. The regions in the bottom right corners are the large states with large shape errors.

5.3 Interactivity

Cartograms re-construct the map with complex steps, but do not result in any absolutely correct solution. People’s preference could strongly influence their opinion on whether the
Table 5.1  A comparison of the two diffusion cartograms in Figure 5.4.

<table>
<thead>
<tr>
<th>Error</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>934.96</td>
<td>904.71</td>
</tr>
<tr>
<td>Shape</td>
<td>590.86</td>
<td>411.51</td>
</tr>
<tr>
<td>Area</td>
<td>344.10</td>
<td>493.20</td>
</tr>
</tbody>
</table>

cartogram is informative or not. Hence providing some flexibility for users to manipulate a
cartogram and check the numerical summary is important. In this section we will discuss two
designs for the interactivity: linear interpolation and parameter tuning.

5.3.1 Linear interpolation

Figure 5.9 demonstrates the idea of linear interpolation between two polygons with the
same number of vertices. The corresponding vertices are linked and new vertices are chosen
along the connected lines by some weight.

Figure 5.9  Linear interpolation between two polygons.

In practice, $m$ equally spaced interpolations between the original map and the diffusion
cartogram are calculated. The user can conduct a smooth transformation either from the map
to cartogram or from the cartogram to map, and pick a preferred interpolation. This interaction
builds a bridge between the shape-preserved map and the area-motivated cartogram.
Figure 5.10 shows how the interpolations change with equally spaced weights. The first row does not look much different, but the last row changes the shape dramatically. Figure 5.11 presents the errors. The shape error linearly increases with the interpolation weights, and the area error decreases in most part but changes the direction at the end, which results in the curve of the total error.

This interaction is implemented in the R package cranvas (Xie et al., 2015).

5.3.2 Parameter tuning by graphical user interface

Six parameters should be identified for a diffusion cartogram: number of rows, number of columns, blank weight, sea weight, sea width, and blur. The number of rows and columns
are used to generate the lattice, since the grid data is required by the algorithm. Figure 5.12 (a) shows how the grid is constructed. Then a problem occurs that the dots within the map border could take some values, but how about the dots out of the map (Figure 5.12 b)? An interpolation by the range of the density will be assigned to the dots abroad, and the blank weight is used to calculate the interpolation (Figure 5.12 c). The blank weight could be negative, but the interpolation must be positive.

One characteristic of the diffusion method is the flowage of the grid dots. So the dots close to the border of the graphic device may flow outwards. Hence the algorithm adds a “sea” around the grid (Figure 5.12 d). After the transformation, the core grid dots could move to the sea area. The sea weight is used to compute the initial density for the dots in the sea, similar as the blank weight (Figure 5.12 e). The sea width is a multiplier on the number of rows and columns that determines the width of the surrounding area.

The last parameter blur is utilized as the bandwidth of a 2-dimensional Gaussian kernel smoothing on the density (Figure 5.12 f). When blur= 0, no smoothing is directed. The density matrix after smoothing will be used as the input of the diffusion algorithm.
Figure 5.12  How the six parameters create the density matrix for the diffusion method: (a) the numbers of rows and columns determine the resolution of the grid; (b) the density values calculated by the thematic variable are assigned to the intersections within the map, but those out of the map (green dots) need a positive density too; (c) the density values for the green intersections are computed by the blank weight; (d) a sea of grid is created by the sea width; (e) the density values for the intersections in the sea are computed by the sea weight; (f) the blur is used to smooth the density.
In R community, not much work has been done for cartograms. The package \texttt{Rcartogram} employs the diffusion method and becomes the only “active” R package to create cartograms. It allows very strict input data format (positive values in a square matrix as the geographical density) and two parameters (sea width and blur). The package has not been updated for at least 6 years. Started from the release of R-3.1.0, the predict function in the package did not work. Hence a new package \texttt{cartogram} is developed to create cartograms through different methods. The diffusion algorithm still depends on \texttt{Rcartogram}, but fixes the bugs and is more friendly. Since the graphical user interface (GUI) will improve the user experience in tuning parameters and visually checking the results, several \texttt{shiny} (Chang et al., 2015) applets were built within the package in order to enhance the interactivity. Figure 5.13 and 5.14 show the layout of the GUI for diffusion cartograms.

![Figure 5.13](image)

Figure 5.13  (Left) The cartogram tab lists all the parameters with sliders, display the diffusion cartogram, and presents the overall shape, area, and total errors. The regional shape and area errors are also given by a scatterplot. (Right) The evaluation tab directs a further study on the errors by region. By selecting a region, its errors versus the interpolation weight are highlighted. The corresponding map and cartogram with problematic vertices are shown on the right.
5.4 Sensitivity analysis

An interesting question is how the parameters will impact the cartogram result, visually and numerically. In this research a sensitivity analysis was conducted for the diffusion method. The parameters have been introduced in Section 5.3.2.

5.4.1 Data and parameters

The data is from the 2012 US presidential election. Two data components were obtained separately. The first component has three variables: the state name (region), number of electors (desired area), and the candidate they voted to (the color on the map). The second component is the geographical information, including the vertex coordinates for every state. 48 states on the US continent and Washington D.C. are used to plot the map.

The parameter setting is given by Table 5.2. Here are the reasons for the choice: for the number of rows and columns, if the values are too small, say less than 50, then some small regions like D.C. or Rhode Islands could be missed in the lattice. And a small-scale pre-stage test showed that the numbers greater than 150 will dramatically increase the errors.
The number of columns is less than the number of rows, because the regions spread more horizontally than vertically. The differential change caused by the blank weight and sea weight is not linear, so instead of using the equally spaced values between 0 and 1, values close to 0 are preferred. Sea width can greatly influence the computation time since it increases the number of grid points by squared geometric progression. The pre-stage test showed that the difference between width 3 and 5 is small, so the upper bound of the sea width is set to be 3. For blur, $0 - 3$ are employed to check the statement from Gastner and Newman (2004) that as blur increases, the readability of the cartogram goes better. So we assume the shape error would decrease as blur increases.

The total number of combinations from 6 parameters is 10,800. For each combination, 9 additional interpolations with weights $0.1, 0.2, \ldots, 0.9$ are computed. Hence there are 108,000 plots in total, and the evaluation functions should be applied on each of them.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td># rows</td>
<td>50, 75, 100, 125, 150</td>
</tr>
<tr>
<td># columns</td>
<td>50, 75, 100, 125, 150 &amp; (#\text{col} \leq #\text{row})</td>
</tr>
<tr>
<td>blank weight</td>
<td>0, 0.05, 0.1, 0.2, 0.4, 0.8</td>
</tr>
<tr>
<td>sea weight</td>
<td>0, 0.05, 0.1, 0.2, 0.4, 0.8</td>
</tr>
<tr>
<td>sea width</td>
<td>0.1, 0.5, 1, 2, 3</td>
</tr>
<tr>
<td>blur</td>
<td>0, 1, 2, 3</td>
</tr>
</tbody>
</table>

### 5.4.2 Results

#### 5.4.2.1 Best and worst choices

The total errors range from 538.2 to 2008.5 for the 108,000 cartograms, while the baseline – total error for the original map, is 858.6. Table 5.3 lists the top 5 smallest and largest total errors. The best cartograms come with an appropriate aspect ratio, smaller blank weight and sea weight that gently keep the global shape from distortion, and blur=1. Figure 5.15 shows the best and worst cartograms chosen by the the total error or area error.
Table 5.3  Parameters to obtain the smallest (top 5 rows) and largest (bottom 5 rows) total errors.

<table>
<thead>
<tr>
<th></th>
<th>total error</th>
<th># rows</th>
<th># cols</th>
<th>blank weight</th>
<th>sea weight</th>
<th>sea width</th>
<th>blur</th>
</tr>
</thead>
<tbody>
<tr>
<td>smallest errors</td>
<td>538.20</td>
<td>150</td>
<td>75</td>
<td>0.05</td>
<td>0.05</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>538.44</td>
<td>150</td>
<td>75</td>
<td>0.05</td>
<td>0.05</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>538.63</td>
<td>150</td>
<td>75</td>
<td>0.05</td>
<td>0.05</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>539.66</td>
<td>150</td>
<td>75</td>
<td>0.05</td>
<td>0.05</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>545.62</td>
<td>150</td>
<td>75</td>
<td>0.05</td>
<td>0.1</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>largest errors</td>
<td>2008.47</td>
<td>125</td>
<td>125</td>
<td>0.8</td>
<td>0.1</td>
<td>0.5</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1884.72</td>
<td>150</td>
<td>125</td>
<td>0.8</td>
<td>0.4</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1861.62</td>
<td>150</td>
<td>125</td>
<td>0.8</td>
<td>0.4</td>
<td>0.1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1835.69</td>
<td>150</td>
<td>125</td>
<td>0.8</td>
<td>0.4</td>
<td>0.1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1806.07</td>
<td>150</td>
<td>125</td>
<td>0.8</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 5.15  (Left) The best cartogram by total error. (Center) The best cartogram by area error. (Right) The worst cartogram by either the total error or area error.

5.4.2.2  Interpolation

Can the interpolation find a balance between the shape and area by minimizing the total error? The answer depends on other parameters. As shown in Figure 5.16, as the interpolation weight goes from 0 to 1, the shape error increases linearly, while the area error decreases on a concave (downward) curve when the weight < 0.75. But the area error deviates from the curve and increases dramatically when the weight > 0.75 in some cases. Figure 5.17 shows the line plots of interpolation weight and total error, faceted by the grid resolution and sea weight, colored by the blank weight. Obviously the blank weight groups the lines, and the number of rows and columns controls the spread of total errors.
Figure 5.16 Errors by the interpolation weight. When the weight is 0, all plots are just the original map, so the errors all start from the same point. Then depending on the parameter setting, lines go to a variation of directions. The shape errors increase linearly with different slopes. The area errors decrease until at least weight = 0.75, then different situations should be discussed.

However, another truth is that if the parameters are chosen appropriately, then the total error is minimized when the interpolation weight is 1, i.e., the interpolation is not necessary, unless another evaluation function for the shape error is designed with a convex curve. Hence all the plots and comparison below will only use the evaluation data with the interpolation weight = 1.

5.4.2.3 Parameters

- Grid resolution and aspect ratio. Resolution means the number of rows or columns. Aspect ratio is the ratio between number of rows and number of columns. Figure 5.18 shows how the resolution and aspect ratio impact the errors. Generally speaking, the errors spread wider when the resolution goes higher. And as the aspect ratio increases
Figure 5.17  Line plots between the interpolation weight and total error. The lines are colored by the blank weight, and facetted by sea weight on rows and the number of rows/columns on columns. The main finding is that blank weights can separate lines into groups.

from 1 to 2, the errors (especially the shape error) decreases. In this research, we are more interested in the minimal value of each boxplot than the interquartile. It turns out that for this data, cartograms with a higher resolution and aspect ratio around 2 are preferred.

• Blank weight. According to this study, blank weight is the most important parameter that impacts the cartogram evaluation. As shown in Figure 5.17, blank weight can classify the interpolation error lines. It is also clear to see from Figure 5.19 that the total error achieves the smallest when the blank weight is 0.05, i.e., the density in the blank area should be a little greater than the minimal density on the map.
Figure 5.18  Boxplots of errors by the number of rows and aspect ratio. The range of area errors is wider when the resolution is higher. The shape error decreases when the aspect ratio increases.

Figure 5.19  Boxplots of errors by blank and sea weight. All the three errors show the U shape as the blank weight increases from 0 to 1. The shape error is lower when the blank weight is 0.05, and the area error is lower when the blank weight is 0.05 or 0.1.
• Sea weight. Sea weight determines the density of the area around the lattice. In most cases, the area is adjacent to the blank area. So it is interesting to study if the sea weight should be the same as or different from the blank weight. Figure 5.19 shows that given a fixed blank weight, the minimized total error first decreases then increases as the sea weight grows, and the optimal value could occur anywhere when the sea weight is 0.05, 0.1, or 0.2. Figure 5.20 reveals that according to the area error, the sea weight should be smaller than the blank weight.

![Figure 5.20](image)

Figure 5.20  Boxplots of errors by blank weight and the difference between two weights. In terms of the area error, the sea weight should be smaller than the blank weight if the blank weight is fixed. But to get the lower shape error or total error, sea weight should be moderate given the blank weight.

• Blur. Gastner and Newman (2004) suggested that the shape error would decrease as blur increases. This is confirmed by Figure 5.21. However, the area error positively associates with blur, which results in a U shape of the total error given the blank weight.

• Sea width. Figure 5.22 shows that sea width does not influence the area error. The total error is also very stable as long as the sea width is not too small.
Figure 5.21  Boxplots of errors by blur and blank weight. Give the blank weight, as the increases, the shape error decreases while the area error increases, and total error has a U shape.

Figure 5.22  Boxplots of errors by sea width and blank weight. Sea width is not an interesting parameter.
5.4.2.4 U shape of the total error

As a summary of the study, the blank weight is the key parameter to create a diffusion cartogram. Looking at the boxplots of total error given blank weights, it is common to find a U shape over other parameters, like the sea weight, blur, and grid resolution. Hence the optimization of the parameters can be achieved by the proposed cartogram evaluation functions, and the visual check can be supplemented with the shiny GUI.

5.5 Conclusions and future work

A cartogram is a variation of a map that better represents areal data. It reshapens the polygon mapping regions to produce polygons with area in proportion to a thematic variable. There are two types: non-contiguous and contiguous cartograms. Non-contiguous cartograms strictly follow the desired area, but abandon the adjacency between contiguous neighbors, and sometimes break the topology. Contiguous cartograms keep the adjacency and topology, but sacrifice a small part of the area and a big part of the regional/global shape. This research developed an evaluation method that can balance the shape and area accuracy in contiguous cartograms, and implemented the ideas in an interactive visualization tool that can allow analysts to examine the effect of parameter choices on the cartogram.

Further testing of the evaluation function is needed. The sensitivity analysis we have conducted indicates that our evaluation function can help in tuning parameters for diffusion cartograms. However, whether the resulting cartogram is aesthetically pleasing and can be accurately interpreted by the user should be tested with a human subject experiment. Theoretically, this evaluation method can be applied with any contiguous cartogram methods that replace the vertices, but it is yet to be tested on other algorithms.

Our software and visualization package, cartogram, could be extended in several ways. The current package includes functions to structure the areal data, tackle the geometric problems, create cartograms by various algorithms, plot shapes and different types of cartograms, and examine the algorithm results with a GUI. Those many functionalities make it difficult to balance all the possible ways that cartograms can be created and examined, and a better
strategy is needed for evaluating all the different pieces or even separating the functionality into smaller, more focused packages.
CHAPTER 6. SUMMARY AND FUTURE PLANS

6.1 Summary of the contributions

The research presented in the preceding chapters emphasizes the importance of interactive graphics in exploring data, examining variation, revealing trends, and diagnosing models.

Chapter 2 provides a new tool that makes it possible to explore patterns of missing data and the impact of various imputation methods on the data distribution. Chapter 3 studies a taxonomy of interactivities and extracts the pipeline to execute the additive interactions. The taxonomy and pipeline have been implemented in cranvastime, as described in Chapter 4. Chapter 5 develops the numeric and visual approaches to evaluate contiguous cartograms and balance the trade-off between desired areas and readable shapes.

The thesis also illustrates developing interactive graphics tools that assist specific research problems and purpose: missing data imputation, contiguous cartogram evaluation, and time series exploration. In the first two cases graphical user interfaces are adopted to connect visualization with statistical models and analysis. Because these provide graphical summaries, that update interactively, they can be used to help researchers develop and evaluate new methodology. The chapters on time series and longitudinal data illustrate interactive graphics software designed for exploring the characteristics of time dependent data.

6.2 Future work

Below is the outline for the next steps in this research.

- Missing values

  - A better integration of the missing data imputation and interactive graphics. The
current GUI can be seen as the first stage of human-computer interaction. Integrating these ideas into an interactive graphics software like cranvas will speed up the interaction and enable the imputation on-the-fly. It will also enhance the examination of missing patterns by linking with other graph types, and handle the imputed values more flexibly.

- More imputation methods can be studied and added to the current work.

- Time series data

  - Interactive time series plots in the frequency domain. A link between the time domain and frequency domain could intuitively explain models in application, and better demonstrate the Fourier transformation to learners.

  - Interactive graphics in the polar coordinates. Polar coordinates are commonly used in visualizing time. When a regular period exists, polar coordinates make it easy to compare periods by wrapping the series in loops. The current interactive graphics system only provides Euclidean coordinates, so the polar coordinates will be an interesting extension.

  - Interactive lag plot to explore the auto-correlation. This functionality links the time series plot with a lag plot, and when the lag time is changed on-the-fly, the lag plot will be updated to reveal the level of auto-correlation.

  - Similarity evaluation for multivariate time series and longitudinal data. When there are multiple time series on one graph, which of them have a similar pattern will be a question of interest. During facetting (by individual or variable), if no order is preferred, then it is better to put the similar series together by default. Hence to compute a distance matrix for series is important.

  - Auto update of the aspect ratio after interactions. Cranvastime can optimize the aspect ratio for the first window, but later after wrapping or facetting, the user will need to manually adjust the window border to get a nice aspect ratio. An automatic update could make the exploration more efficient.
– Test on different human-computer interaction designs. In Chapter 3 there are many possible design choices. An experiment on testing the differences will help determine the best choice.

– More effort on the GUI development for interactive graphics. Currently the cranvas package does not have a GUI, it is hard for the user to remember all the interaction commands for all types of graphs.

• Areal data and cartograms

– Further tests on the evaluation function and user preference. Whether the cartogram chosen by the evaluation function can express the information efficiently should be tested by human subject experiment. The preference between contiguous and non-contiguous cartograms is also an interesting question that can be investigated by experiment.

– A refined structure of the software. The current cartogram package includes functions to structure the areal data, tackle the geometrical problems, create cartograms by various algorithms, plot shapes and different types of cartograms, and make GUIs. More work is needed to completely develop this package. And later the package should be split into several smaller ones, so that the functionality for each purpose will be refined and fitted for more research questions of areal data.

– Visualization for the confidence intervals in areal data estimations. The areal data uses x- and y- axes to lock the geographical coordinates, so the estimates or predictions by region can only be displayed by an aesthetic element like color (heatmap) or size (cartogram). Hence the confidence intervals are not able to be displayed with the estimates. The accuracy of reading the estimates and confidence intervals is even harder to achieve. Interactive graphics that links the map with other plots could be a solution for visualizing the areal confidence intervals.
BIBLIOGRAPHY


Bostock, M. (2012). Data-driven documents (d3.js), a visualization framework for internet browsers running javascript. 2


Kotter, T. and Theus, M. Fortune a system for interactive statistical graphics for time series. home.vrweb.de/~martin.theus/Fortune_JCGS.pdf. 39


Lawrence, M. and Verzani, J. (2012). *Programming Graphical User Interfaces in R*. CRC Press. 2


Lee, D. and Su, Y.-S. (2011). *migui: Graphical User Interface of the mi Package*. R package version 0.00-09. 7


Nightingale, F. (1858). *Notes on Matters Affecting the Health, Efficiency, and Hospital Administration of the British Army: Founded Chiefly on the Experience of the Late War. By Florence Nightingale. Presented by Request to the Secretary of State for War*. Harrison and Sons, St. Martin’s Lane, WC. 37


Verzani, J., Urbanek, S., Grosjean, P., and Lawrence, M. (2012). gWidgets: gWidgets API for Building Toolkit-independent, Interactive GUIs. R package version 0.0-52. 31


