



BD #2017/1

Big Data: Visualization Practical Work First Semester Course 2017/2018

Antonio LaTorre, Pablo Toharia

Contents

1	General description of the visualization practical work exercises	3
1.1	Exercises to be completed	3
1.2	Getting support	3
1.3	Notation	4
1.4	How to prepare the delivery of the results	4
1.5	Delivering the exercises	5
1.6	Deadlines	5
2	Description of the practical work	6
2.1	Installation and configuration of the tools needed to accomplish the practical work: R, RStudio, Shiny	6
2.2	Introduction to Shiny	7
2.3	Structure and GUI of apps developed with Shiny	8
2.4	Addition of new interactive controls	8
2.5	Interactive visualization of data	8
2.6	R scripting	9
2.7	Reactive expressions with Shiny	9
2.8	Sharing applications	9

2.9	Review of some examples programmed with Shiny	9
2.10	Design of a new interactive data analysis tool	9
History of versions		14
References		14

The objective of this practical work is to help students to put into practice the concepts learnt during the theory classes and to get proficiency in the use of some tools that will allow her to interactively analyze a set of data by means of the combination of data mining and visualization techniques. Furthermore, students will be able to develop their own tools to conduct this kind of analysis.

1 General description of the visualization practical work exercises

1.1 Exercises to be completed

The practical work consists in the following exercises:

- ① **Installation and configuration of the tools needed to accomplish the practical work: R, RStudio, Shiny**
- ② **Introduction to Shiny**
- ③ **Structure and GUI of apps developed with Shiny**
- ④ **Addition of new interactive controls**
- ⑤ **Interactive visualization of data**
- ⑥ **R scripting**
- ⑦ **Reactive expressions with Shiny**
- ⑧ **Review of some examples programmed with Shiny**
- ⑨ **Design of a new interactive data analysis tool**

1.2 Getting support

You can reach the lecturers in charge of this practical work on the following e-mail addresses:

- Antonio LaTorre <atorre@fi.upm.es>
- Pablo Toharia <ptoharia@fi.upm.es>

1.3 Notation

Through this document, the following notation will be used:

<code>text</code>	An input provided by the user.
<code>text</code>	An output generated by the program to the standard output .
<u><code>text</code></u>	An output generated by the program to the standard error .

1.4 How to prepare the delivery of the results

- ❑ The source code of the practical works, i.e., the programs developed to solve each exercise, needs to be stored in a directory that identifies the subject, exercise and course to which it belongs to: `bd2017_2018/<exercise>`.

The name for the folder `<exercise>` is the results of appending the corresponding exercise number to the root `exercise`: `bd2017_2018/exercise1`.

- ❑ You do not need to provide any printed document. All the exercises will be evaluated electronically.

Apart from the source code, the following items should be provided for each of the exercises:

- ① **authors.txt**. ASCII file with the personal data of the author of the exercise.

Each line will contain the following four strings separated by blanks (spaces or tabs): ID or passport number, surname, name and university registration number. Hyphenated names and surnames should be joined with the underline character (`_`). **Do not** use blanks in the middle of names or surnames.

Example: **P12456212F La_Torre Maria_Dolores 910347**

- ② **blog.txt** ASCII file with relevant comments on the fulfillment of the exercise, which should cover the following aspects: author, problems found during the exercise, personal comments, constructive review, suggestions, estimated time required to complete the exercise, etc.

DO NOT NEGLECT THE QUALITY OF THE REPORT: It will be evaluated in the same way the source code will be and needs to obtain a favorable evaluation in order for the practical work to pass. A low quality report may make you fail the practical work, even if the source code works properly.

1.5 Delivering the exercises

The exercises will be sent through the corresponding submission form in Moodle. All the files must be compressed in a single `.zip` file before submitting them. The exercise must be carried out in couples and only one of the members of the team needs to submit the practical work.

1.6 Deadlines

These exercises require continuous work from students. Except for the case of exercise ⑨, they should be completed in the laboratory and the results should be submitted before december the 10th 2017.

The deadline for exercise ⑨ is january the 21st 2018.

2 Description of the practical work

2.1 Installation and configuration of the tools needed to accomplish the practical work: R, RStudio, Shiny

R is a functional programming language with a large library of algorithms for data analysis [1]. It can be downloaded from [The Comprehensive R Archive Network](#).

RStudio is an IDE that provides a GUI to develop R applications [2]. It can be downloaded from [RStudio](#).

RStudio offers some nice features, such as input autocomplete (triggered with the tab key after writing the first part of the command), a graphical visualization of the variables currently stored in the environment (on the right pane) and integrated plots, help and history.

The **Esc** key deletes the line currently written at the **prompt** and can also be used to cancel an action that is currently running.

To obtain access the documentation of some function, you can use the **help** function, which receives the name of the function for which we need help as an argument, or the integrated help pane (bottom pane on the right side).

```
> help(help)
```

Autocomplete is also available within the help function, and an abbreviated form of this command is also available:

```
> ?help
```

Shiny provides a development environment that eases the design and implementation of interactive applications with R [3].

🔗 **NOTE:** A machine with Linux operating system is assumed for this practical work.

To proceed with the exercises, we must ensure that **Shiny** is present in our system. From within **RStudio** we can check if **Shiny** is installed by:

```
> installed.packages()
```

If **Shiny** does not appear in the list of installed packages, we should proceed to install it:

```
> install.packages("shiny")
```

2.2 Introduction to Shiny

We are going to follow the script proposed in [the first lesson of the Shiny tutorial](#).

This first exercise constitutes a brief introduction to RStudio and Shiny in which the students will be able to plot an interactive histogram.

This first example can be loaded from RStudio with the following commands:

```
> library(shiny)
> runExample("01_hello")
```

The first command imports the library and the second one runs the experiment, which should open the application in a web browser.

RStudio provides several views, each of them made up of several tabs:

- **Console:** it allows the execution of commands interactively.
- **Code:** it offers a full-featured editor in which we can code our scripts, run them (completely or even partially), search and replace for strings, etc.
- **Environment and history:** the first tab shows the variables currently stored in the environment and allows the visual inspection of complex structures such as data frames. It also offers the possibility of reading/writing those variables from/to files. The second one allows the inspection of recent commands run in the console.
- This view combines in several tabs a file explorer, a plotting area, the package center and a tool to navigate through the help pages of the loaded packages.

All these views can be minimized, maximized or resized.

The minimal structure of a Shiny application is made up of two components:

- One script for the GUI: it defines the structure and the look&feel of the interactive data analysis tool.
- One script to be ran server-side: it defines the context in which the application will be run.

These components can be stored in a directory named as the application that we are building and it can be run in the following way:

```
> library(shiny)
> runApp("my_app")
```

It should be noted that **Shiny** apps will be opened in the default web browser of the system and that the publication of the application is also possible from **RStudio**.

2.3 Structure and GUI of apps developed with Shiny

We are going to follow the script proposed in [the second lesson of the Shiny tutorial](#).

The objective of this exercise is to code a **Shiny** app from scratch. We will explore how to include text and visual information in the application.

2.4 Addition of new interactive controls

We are going to follow the script proposed in [the third lesson of the Shiny tutorial](#) and in the [application layout guide of Shiny](#).

In this section we will explore the set of interactive widgets available in **Shiny**, adding some of them to the application, and will describe with more details how a **Shiny** application is structured.

The available **Shiny** widgets come from [Twitter Bootstrap](#). There is a [widget gallery](#) that provides reference code to be able to program our own widgets with a lot of ease.

2.5 Interactive visualization of data

We are going to follow the script proposed in [the fourth lesson of the Shiny tutorial](#). This tutorial shows how the widgets in our application can be updated interactively. To summarize, this can be accomplished by linking those elements that are considered to be an input for users with the output. **Shiny** will take care of updating all the output elements that depend on the changes done by the users.

2.6 R scripting

We are going to follow the script proposed in [the fifth lesson of the Shiny tutorial](#). This lessons explains how to load datasets, new packages and run R scripts. Furthermore, it also reviews functions that allow to gather information on the workspace of the user. As an extra exercise, you can try to use the `xlsx` library, which provides the functionality to read and write from/to `xlsx` files. For this purpose, the aforementioned package must be installed, as well as its associated dependencies.

2.7 Reactive expressions with Shiny

We are going to follow the script proposed in [the sixth lesson of the Shiny tutorial](#). This lesson explains how to avoid global updates of components, as those presented in Section 2.5, by updating just those components that need to be updated.

2.8 Sharing applications

We are going to follow the script proposed in [the seventh lesson of the Shiny tutorial](#), which describes two alternative methods to share a Shiny application with other users: distributing the source code and installing all the dependencies on the new host or sharing the application through a web page.

2.9 Review of some examples programmed with Shiny

We are going to review some of the examples available in [the Shiny gallery of examples](#).

2.10 Design of a new interactive data analysis tool

This last exercise will consist in the design and implementation of a Visual Analytics tool. The design of this tool, like any other interactive analysis tool, must solve the particular problems that an “end user” or “analyst” has to deal with when facing some data that he wants to analyze in a work session. Students have two alternative regarding the datasets to work with:

1. Select one of the proposed datasets (see below).

2. Propose their own dataset. However, in order to make the level of complexity uniform for all the students, data should be approved by the lecturer responsible of the practical work during the lecture of december the 15th 2017 at the latest.

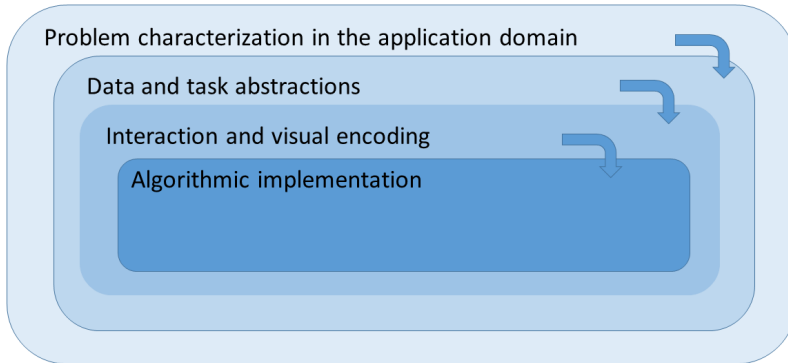


Figure 1: Design abstraction levels

Once the selection of the dataset has been validated by the lecturer, the students must start working on the design of the visual analytics tool by first selecting the questions the potential user of the tool (the “analyst”) could ask. These questions must comply, at least, with the following **minimum** constraints:

- They should involve the use of 3+ different types of plots (idioms) that accurately answer the considered questions.
- Filtering and/or aggregation should be possible in order for a user to be able to focus on the information he finds more interesting.

However, these are just minimum requirements to pass the practical work. The tool could be completed by incorporating other design options among those seen in class:

- Use of multiple (synchronized) views.

- Integration of the application with some data analysis algorithm (cluster analysis mainly).
- Use of specific arrangements for the data under consideration (mainly geographic arrangement for the datasets with that information).

For the practical work students must use **Shiny** and apply the concepts seen in the theory lectures of the course.

Datasets Students could propose their own dataset or select among the following selected ones.

- **Europe stats:** This dataset was already used in class in the practical work on clustering. It contains several statistics on european countries and can be effectively analyzed by means of hierarchical clustering.
- **USA arrests:** This dataset contains information on the crime rates for the states of the USA and can be effectively analyzed with EM clustering.
- **Markb2013 baseball dataset:** It contains a set of records on pitches for the 2013 season of the Major League Baseball. For each record, a large number of variables were recorded. One interesting analysis with this dataset would be to automatically detect to which type of pitch the record belongs. Normally, it is accepted that there are five types of pitches:
 - FT: Two-seam fastball
 - CH: Changeup
 - FC: cutter fastball
 - FF: Four-seam fastball
 - CU: Curveball
 - IN: Indefinite

More information on the different types of pitches can be obtained in <https://fastballs.wordpress.com/2007/08/02/glossary-of-the-gameday-pitch-fields/>.

The identification of the type of pitch can be done by clustering the data with the EM algorithm taking into account the following considerations:

- Data should be filtered from 2013-04-01 onwards to exclude pre-season matches.
- The most informative variables for this clustering are: `start_speed`, `break_y`, `break_angle` and `break_length`.
- When clustering with the EM algorithm, the VVV model should be selected, as it fits better data when no apriori knowledge on the shape of the clusters is provided.

The results of the clustering could be projected to ease their visualization and the plots with the evolution of the BIC measure could also be interesting to analyze.

- **Yelp Dataset Challenge:** This dataset is provided by Yelp, a service that allows users to review businesses and check other users reviews. They provide a subset of their data in a challenge (which is in its 10th round) to promote the development of innovative visual analytic tools. This dataset contains geolocalized information about businesses, users reviews and scores, etc. Many different analyses could be carried out on these data and they support many of the encoding options seen in class (cartographic arrangement, filtering of data, etc.).

The data can be downloaded from http://www.yelp.com/dataset_challenge and the challenge is now open, with ten awards of \$5,000 for the most innovative tools.

Report Apart from the recommendations provided in Sections 1.4 and 1.5, the student should provide a report describing the design decisions taken during the development of the application. This report should be structured according to the four design abstraction levels (Figure 1), and all the decisions made at every level must be discussed, highlighting those

dealing with the **interaction level and the visual encoding**. To this purpose, students should be aware that the data and tasks that the analyst wants to carry out are the base on top of which these decisions must be justified.

Revision History

This document has been published as a teaching report with code 2017/1 by the *Departamento de Arquitectura y Tecnología de Sistemas Informáticos* of the *Escuela Técnica Superior de Ingenieros Informáticos*, *Universidad Politécnica de Madrid* as part of the course *Big Data*. The history of changes made to this document is:

Revision	Date	Author(s)	Description
1.1	09/09/2015	JM, AR	Original version
1.2	30/09/2015	AR	New latex
2.1	30/11/2015	ALT, JM	New contents
2.2	17/11/2016	ALT	Updated statement

This document is currently maintained by: Antonio LaTorre, Pablo Toharia. © Departamento de Arquitectura y Tecnología de Sistemas Informáticos, Escuela Técnica Superior de Ingenieros Informáticos

References

- [1] GNU: *The R Project for Statistical Computing*. www.r-project.org 6
- [2] RStudio: *RStudio*. www.rstudio.com 6
- [3] Shiny: a web application framework for R: *RStudio*. www.shiny.rstudio.com 6