BD/DV
2021/2022

Big Data / Data Visualization: Visualization Practical Work

Antonio LaTorre, Pablo Toharia

Contents

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 them to interactively analyze a set of data. 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

Before starting the exercises please install and configure the environment and tools we are going to use: RStudio and Shiny.

The practical work consists in the following exercises:

① **Shiny Tutorial**

- **Introduction to Shiny**
- **Structure and GUI of apps developed with Shiny**
- **Addition of new interactive controls**
- **Interactive visualization of data**
- **R scripting**
- **Reactive expressions with Shiny**
- **Review of some examples programmed with Shiny**

② **Design of a new interactive data analysis tool**

1.2 Getting support

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

- Antonio LaTorre <a.latorre@upm.es>
- Pablo Toharia <pablo.toharia@upm.es>

1.3 Notation

Through this document, the following notation will be used:

`text`	An input provided by the user.
text	An output generated by the program to the **standard output**.
text	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 and should be uploaded to Moodle in a compressed file.

❒ 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:

☞ **authors.txt**. ASCII file with the personal data of the authors 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 joined with the underline character (_). **Do not** use blanks in the middle of names or surnames.
Example: **P12456212F La_Torre Maria_Dolores 910347**

☞ **report.pdf** For exercise ② A PDF file with a report of the work done including: data abstraction, task abstraction, idioms used, justification of the decisions made, 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 and some of the work will be done during class time but students have to also dedicate extra time in order to complete them.

The deadlines are the following:

- Exercise ① deadline is Dec 13^{th} 2021.
- Approval of the chosen data-set: Dec 13^{th} 2021.
- Exercise ② deadline is Jan 24^{th} 2022.

2 First exercise: Shiny Tutorial

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].

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 Lesson 1. 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: if 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 Lesson 2. 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 objetive 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 Lesson 3. 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 Lesson 4. 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 acommplished 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 `xls*` files. For this purpose, the aforementioned package must be installed, as well as its associated dependencies.

2.7 Lesson 5. 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 Lesson 6. Sharing applications

We are going to follow the script proposed in the seventh lesson of the `Shiny` tutorial, which describes two alternative mehods 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 Lesson 7. Review of some examples programmed with Shiny

We are going to review some of the examples available in the `Shiny` gallery of examples.

3 Second exercise: Design of a new interactive data analysis tool

This exercise will consist in the design and implementation of a interactive visualization 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. The analysis and design should follow the steps depicted in Figure 1 as seen in theory classes.

The first step would be to characterize the problem in the application domain. For that, students should select a data-set and pose some questions the analysts would like to answer with the visual analytics tool.

Regarding the datasets students have two alternatives:

1. Choose one of the proposed datasets (which will mean the highest grade will not be achievable, therefore not recommended). See the proposed ones below.
2. Propose their own dataset (highly recommended!). However, in order to make the level of complexity uniform for all the students, the selected data-set should be approved (in class time or by email/Teams) by the lecturer responsible of the practical work (check deadline in Section 1.6).

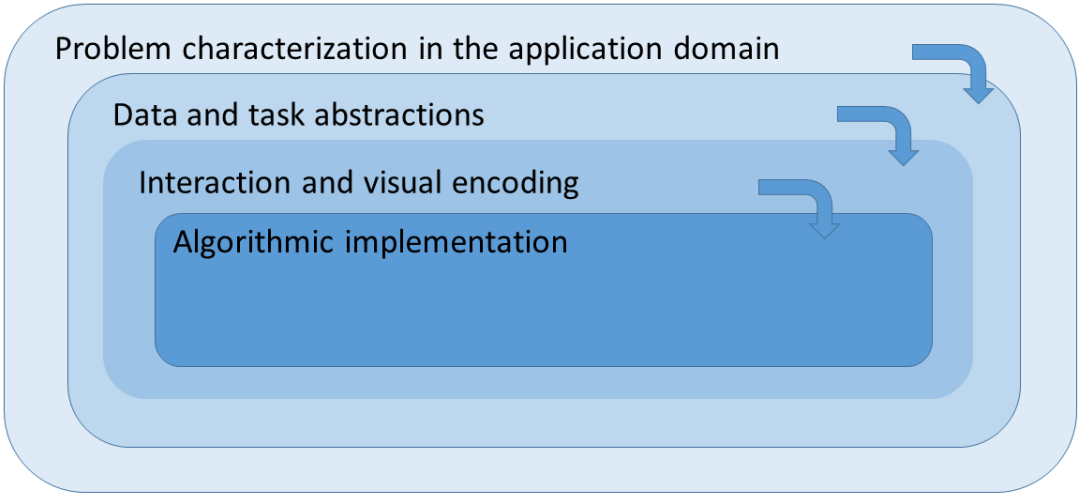

Figure 1: Design abstraction levels

Once the selection of the dataset has been validated by the lecturer, the students must pose the questions (tasks) the potential user of the tool (the "analyst") could ask. It should be taken into account that the questions posed must be well suited for answering by means of visualization (avoid questions that can be answered automatically in a more effective way, for example, by querying a table).

Once the data and tasks are defined in the application domain, the next step would be to do the data and task abstractions (as seen in theory classes).

When the data and task abstractions have been defined it is time to start the design of the tool by choosing idioms (visualization+interaction) that are suited and effective for the tasks and data.

Finally, the design has to be implemented in `Shiny`.

These questions and design must comply, at least, with the following **minimum** constraints:

- The tool should use 3+ different types of plots that accurately answer the considered questions.
- Some kind of 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, but only if it is well justified!).
- Publish the app

3.1 Datasets

Students can propose their own dataset (highly recommended as it will be taken into account for the grades!) or choose one among the following.

- **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 used to provide a subset of their data in a challenge (which had 12 rounds) 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 https://www.yelp.com/dataset.

3.2 Report

Apart from the recommendations provided in Sections 1.4 and 1.5, the delivered report should meet the following requirements:

- Structure the report 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.

- Include proper instructions on how to run the tool (important for evaluation!) and the dependencies needed.
- In case the tool is published in https://www.shinyapps.io/, please write the link in the report.
- Include screenshots of the different idioms that illustrate how they help answering the questions posed by the analysts.

Revision History

This document has been published as a teaching report 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 *Information Visualization*. 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
3.1	6/2/2019	PT	Changed structure
3.2	6/11/2019	PT	Dates and subject names changed
3.3	14/11/2020	PT	Updated dates and other minor changes
3.4	11/11/2021	PT	Updated dates and rewritten part of the second exercise.

References

[1] GNU: *The R Project for Statistical Computing*. www.r-project.org 3

[2] RStudio: *RStudio*. www.rstudio.com 3

[3] Shiny: a web application framework for R: *RStudio*. www.shiny.rstudio.com 3