

# Introduction to Quantitative Analysis in Support of Evidence-Based Policy Making for Social Assistance Programmes

United Nations

Economic and Social Commission for Western Asia



## Content table

Introduction .....	3
Social Protection Programmes Rapid Assessment Framework (SPP-RAF).....	3
Manual structure and final remarks .....	5
Chapter 1: Introduction to R .....	8
What is R?.....	8
Installing R and R Studio.....	9
RStudio Interface .....	13
Packages and libraries.....	16
Installing packages .....	17
Elements in R .....	18
Variables .....	18
Vectors .....	19
Matrices .....	19
Factors .....	21
Data frames.....	21
Function.....	22
Conditional statements.....	24
Loops .....	25
While loop.....	25
For loop.....	26
Chapter 2: Data management .....	27
Uploading data .....	27
Managing the data .....	28
Chapter 3: Graphs.....	41
Basics .....	41
Formatting .....	43
Labels .....	43
Position, gridlines, and colours.....	43
Axes and spaces .....	44
Other formatting criteria.....	45
Clustering graphs.....	45
Graph types .....	46
Chapter 4: Excel synchronization .....	49
Main commands .....	49
Dictionaries.....	52

Chapter 5: Automatizing of large processes.....	57
The project and the master file.....	57
Producing descriptive statistics.....	59
Analysing the results.....	61
Conclusions .....	67
Chapter 6: Profiling beneficiaries .....	69
Purpose of profiling beneficiaries.....	69
Clustering.....	69
Step 1: Distances .....	70
Step 2: Clustering .....	71
Classification trees.....	73
Profiling analysis.....	76
Final remarks .....	84
Chapter 7: Targeting characteristics .....	85
Purpose of targeting characteristics .....	85
Targeting strategy .....	85
Variance decomposition technique .....	85
Lasso regression.....	86
Targeting analysis .....	87
Chapter 8: Coverage evaluation.....	92
Purpose of the coverage evaluation.....	92
Purpose of the beneficiary evaluation .....	92
The missing data challenge.....	93
The outlier challenge .....	94
Coverage analysis.....	95
Some notes on the beneficiary evaluation .....	102
Final remarks.....	104
References .....	104

## Introduction

Due to the current social, economic, and environmental crisis, the citizens of the world are facing greater challenges, and every year millions of people are falling into vulnerability and poverty statuses. For that reason, social protection programmes are becoming a core part of the development planning of the countries. These programmes are designed to support those who, due to unexpected events such as natural events and sudden economic crisis, have seen their livelihood affected, and if they do not receive support, then the consequences of the shock can last for many years. Similarly, the programmes also aim to target those groups that are facing structural challenges in society (e.g. single mothers, people with disabilities, and elderly population) so that their problems can be reduced or at least attenuated and accordingly, they will have enough tools to develop their personal and professional careers. In this way, social protection programmes currently involve a wide spectrum of strategies to prevent poverty or reduce its impact, tackling both the structural problems that are causing it, as well as its consequences..

In a period where resources are scarce, and people's needs are raising, the governments need to develop strategies that allow them to benefit the people who need them the most. With these strategies, policymakers can improve their chances in choosing the right programmes to use money efficiently, and in that way cover, as much as possible, the needs of the population. Unfortunately, there is no magic recipe for the success of this aspect, since the local reality of each country, its culture, environment, institutions, and history make each case unique in many ways. For that reason, it is not reasonable to claim that there is a solution that fits all the countries in the same way. Nevertheless, it's important to recognize that some structural elements of these programmes have similar patterns. For example, given the limited resource, any social assistance programme needs to have a mechanism that collects information from the individuals and decides if they have the characteristics needed in order for them to become beneficiaries. Therefore, while it is not realistic to create a manual that provides the right guidelines on how to structure social security programmes, it is possible to develop a manual that can guide the policymakers on how to organize and analyse the data that is currently generated by the system in a way that supports the main elements of the programmes, and identify possible areas for improvement. After doing these first-level evaluations, the policymakers will have enough information on some of the points where the system needs improvement and proceed accordingly.

### Social Protection Programmes Rapid Assessment Framework (SPP-RAF)

Social protection assessment tools are common in the international literature and present (based on different perspectives), the key points that the social protection system needs to be aware of. Some of these frameworks are highly conceptual and are tailored to guide the reader on identifying the overall strategy of the system (UNICEF, 2019), other frameworks are more focused on the continuous monitoring of the goals of the programme in order to identify their efficiency and efficacy (OECD, 2019), other frameworks place their attention in the advocacy and socialization strategy, and how to approach the different stakeholders to guarantee their commitment to the implementation of the plans (ILO, 2016), and finally other frameworks develop logical analytical tools to identify the capacity and challenges that a programme can face (GIZ, 2017).

Taking important lessons from the available frameworks and tools, the SPP-RAF aims to provide practical low cost assessments that, without expectations of being exhaustive, can be regularly implemented as part of the programmes and can inform the policymakers on key points affecting the success of the programmes. For that reason, the framework is built on four steps aligned with the diagram displayed in Figure 1.

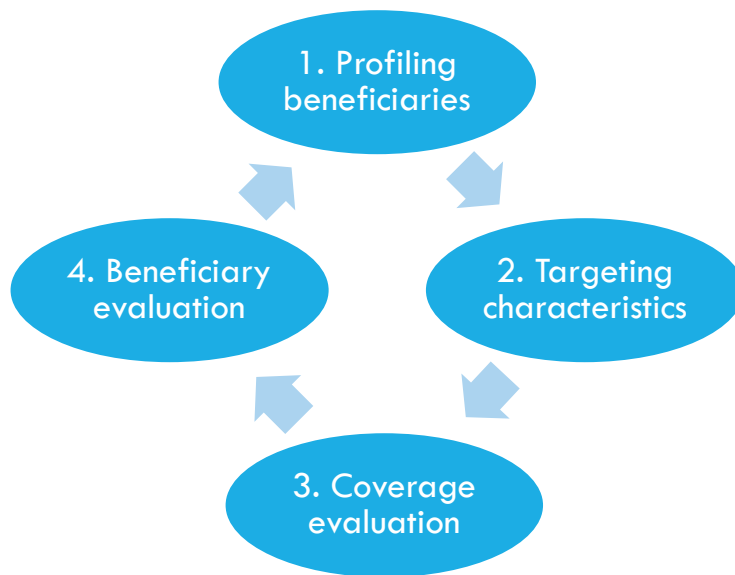


Figure 1: SPP-RAF

While each step is explained in detailed in the following chapters, the overall logical path goes as follows:

1. **Profiling beneficiaries:** The first part is based on the data collected from the beneficiaries. In this stage there is a set of statistical tools that groups them according to their traits. These profiles allow the policymakers to understand who are benefitting from the program, what are their needs, and if possible to improve the targeting of their benefits to guarantee their prompt recovery or efficient alleviation of their hardship conditions.
2. **Targeting characteristics:** The second part is based on the selection process of beneficiaries. By reviewing the official selection criteria and contrasting them with the traits of the current beneficiaries, this step identifies what are the characteristics that the selection process is choosing. In addition, it provides inputs for a high level discussion where decision makers can reflect on the current variables that are defining who is part of the programme, discusses their relevancy given the current and future of the system, and checks if additional variables are needed and if some of the current ones need to be reframed or removed.
3. **Coverage evaluation:** The third part is based on the information of the current beneficiaries and the reduced information (collected by other administrative sources) on non-beneficiaries. By identifying outliers within the current beneficiary population, and estimating the missing characteristics of the non-beneficiaries, this section estimates potential individuals that deserve to be selected as beneficiaries but are not currently in the system. It also identifies those beneficiaries whose combination of selection characteristics is atypical, and it might be the case that some of these variables are not accurately recorded. Although this information is not enough to identify people that are misallocated, it guides the decision-makers on flagging key socio-demographic traits where these issues can be taking place and proceed accordingly. For example, this step can guide the policymakers in the identification of governorates that can have a systematic sub registry of beneficiaries and allows them to discuss better ways to improve their outreach strategy in this area.
4. **Beneficiary evaluation:** The last step is based on the capacity of the system to trace the individuals while they are participating in the programmes and understand how their traits change. Social security programmes are created to improve the conditions of their

beneficiaries; therefore, one ordinary question can be asked about the capacity of the programme to improve these conditions. Thus, by identifying key tracking variables and following individuals as they evolve through the programme, it is possible to check if the individuals are engaging with the system and using it to improve their livelihoods. Caveat: In contrast to the other steps, this last one requires a detailed analysis on the identification of trackers, trend evaluation (being able to distinguish the benefits linked to the programme from those general benefits associated to the overall improvement of the society), and the capacity to know if the individual is getting the benefits of the programme or is just nominally participating. As these elements require a large set of techniques that are distant from the other three steps, the manual develops the first three steps in details and this final step is left for a subsequent manual.

To maximize the practicality of the SPP-RAF, it has been built upon four pillars:

- **Systemization:** The statistical techniques used to implement the SPP-RAF: are replicable (therefore the results are independent of the individual calculating them), can be scaled (the same tools can be used even if the programmes increase their number of beneficiaries or their coverage package), are standardized (the same analytical procedure can be applied to different programmes), and adaptable (as priorities of a programme evolve, the analytical tools reflect these changes and inform policy makers accordingly).
- **Quantitative Analysis:** The process uses administrative generated data, allowing low-cost data collection and regular updates to be done to the system status. E.g. it uses data from the questionnaires that people need to provide in order to access the program, and additional data that they fill when they report their taxes or payment for public services. Moreover, it is designed to begin with low data requirements and increase as the social protection system matures. Finally, conclusions from analysis reflect the local situation, as it is seen from the data gathered for the programs. Thus, these tools can both help to guide decision-makers on doing policies as well as the data managers to continuously improve the information system.
- **Evidence Base Decisions:** Conclusions from the process are supported by the data evidence. Thus, the process is transparent and strengthens the accountability and legitimacy of the organization. For example, policies that are derived from these conclusions can help policy makers to argue in public debates, as they can refer to the data that supports the policy.
- **Status and Evolution:** By regularly monitoring the system, the process helps identifying:
  - a. Populations that have improved by the programs. E.g. Programmes on unemployment that help people to get jobs.
  - b. Population with difficulties accessing the programs. E.g. People having problems to reach public institutions to apply to the program.
  - c. Programmes whose selection methods are not as relevant as before. E.g. Programmes that were needed in the past for formalization of jobs, but when the system changes, then these programmes are no longer needed.
  - d. Programmes whose targeted population has changed. E.g. more and more young women are willing to join the labor market and need new policies to support them.

### Manual structure and final remarks

The present manual is not meant to serve as a self-learning material. It's a teaching aid produced by ESCWA that complements the trainings syllabus on quantitative analysis for social assistance programmes and that can also be used as a quick refresher for the trainees. Moreover, to get the most of the manual, it comes hand in hand with the files associated to the codes and datasets used for the explanation of different contents.

The manual is divided into two parts. Part 1, which includes chapters 1 to 5 is aimed to provide an overall introduction of the statistical software used to implement the SPP-RAF. The chosen statistical software is R. Due to its great flexibility and its open source characteristics, the statistical programme R has become more popular for developing quantitative analysis that requires standardization of large data sets and massive generation of reports. For those reasons, the current training programme is based on this statistical package. While trainees who are familiar with this software can go directly to part 2 and review the statistical tools that implement SPP-RAF, they are still encouraged to quickly go through the last chapters of part 1 (in particular Chapters 4 and 5) as they introduce ways to connect the software with Excel in ways that facilitate the creation of reports. Once the manual covers the knowledge of the software, the second part of the manual, which includes chapters 6 to 8, discusses the different steps of the SPP-RAF at a conceptual level and then shows the relevant tools and codes required for their implementation.

Finally, to improve the link between the concept and the applications, all the examples provided for the implementation of SPP-RAF are based on a fabricated country that has a social protection programme that wants to assess. The full case of study starts early on chapter 2 where the functionalities of R are illustrated by describing the data of this country. For that reason, trainees that want to go directly to the implementation tools, are recommended to quickly read the fragments of Chapter 2 to 5 that describe the case study in order to have a better understanding of the illustrative examples of part 2.

# Part 1: Standardizing work with R



## Chapter 1: Introduction to R

Aiming to level the trainees that have not been exposed to R before, this chapter aims to describe the basic functionalities of the program. It begins in introducing the software and the way to install it, then it describes the different components of the interface that the programme uses, and finally, it explains how to create basic codes<sup>1</sup>.

### What is R?

- R is a language and environment for statistical computing and graphics
- R provides a wide variety of statistical and graphical techniques (linear and nonlinear modeling, classical statistical tests, time series analysis, classification, clustering, etc.). These techniques can be used for the **systematization of quantitative analysis** of social assistance programmes to support **evidence-based decisions**.
- In itself, R does not have a user-friendly interface that facilitates the visualization of inputs and outputs that are used by the program. Therefore, to be able to use R in a comfortable way, it is essential to download an interface called “Rstudio” that improves the way in which the user communicates with the software.

Due to this last point, the next section presents the steps on how to download both R and R Studio. For a more detailed explanation on the installation process, please go to:

<https://courses.edx.org/courses/UTAustinX/UT.7.01x/3T2014/56c5437b88fa43cf828bff5371c6a924/>

---

<sup>1</sup> For avid learners, a recommended reading to expand the knowledge on the R software the live book “Applied Statistics with R” by David Dalpiaz, which is regularly updated and can be found on the link ([https://davidalpiaz.github.io/appliedstats/applied\\_statistics.pdf](https://davidalpiaz.github.io/appliedstats/applied_statistics.pdf)). The current is based on Chapters 1-7 of this book.

## Installing R and R Studio

**Step 1:** Go to <https://www.r-project.org/> and then click on “CRAN” link that is at the left of the webpage.



[Home]

### Download

CRAN



### R Project

About R

Logo

Contributors

What's New?

Reporting Bugs

Conferences

Search

Get Involved: Mailing Lists

Developer Pages

R Blog

# The R Project for Statistical Computing

## Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To **download R**, please choose your preferred **CRAN mirror**.

If you have questions about R like how to download and install the software, or what the license terms are, please read our [answers to frequently asked questions](#) before you send an email.

## News

- **R version 4.1.0 (Camp Pontanezen)** has been released on 2021-05-18.
- **R version 4.0.5 (Shake and Throw)** was released on 2021-03-31.
- Thanks to the organisers of useR! 2020 for a successful online conference. Recorded tutorials and talks from the conference are available on the [R Consortium YouTube channel](#).
- You can support the R Foundation with a renewable subscription as a [supporting member](#)

## News via Twitter

**Step 2:** Press any link to proceed from the list that starts with 0-Cloud

### CRAN Mirrors

The Comprehensive R Archive Network is available at the following URLs, please choose a location close to you. Some statistics on the status of the mirrors can be found here: [main page](#), [windows release](#), [windows old release](#).

If you want to host a new mirror at your institution, please have a look at the [CRAN Mirror HOWTO](#).

0-Cloud	<a href="https://cloud.r-project.org/">https://cloud.r-project.org/</a>
Algeria	<a href="https://cran.usthb.dz/">https://cran.usthb.dz/</a>
Argentina	<a href="http://mirror.fcaglp.unlp.edu.ar/CRAN/">http://mirror.fcaglp.unlp.edu.ar/CRAN/</a>
Australia	<a href="https://cran.csiro.au/">https://cran.csiro.au/</a> <a href="https://mirror.aarnet.edu.au/pub/CRAN/">https://mirror.aarnet.edu.au/pub/CRAN/</a> <a href="https://cran.ms.unimelb.edu.au/">https://cran.ms.unimelb.edu.au/</a> <a href="https://cran.curtin.edu.au/">https://cran.curtin.edu.au/</a>
Austria	<a href="https://cran.wu.ac.at/">https://cran.wu.ac.at/</a>
Belgium	<a href="https://www.freeststatistics.org/cran/">https://www.freeststatistics.org/cran/</a> <a href="https://flp.belnet.be/mirror/CRAN/">https://flp.belnet.be/mirror/CRAN/</a>
Brazil	<a href="https://nbcgib.uesc.br/mirrors/cran/">https://nbcgib.uesc.br/mirrors/cran/</a> <a href="https://cran-cc3sl.ufpr.br/">https://cran-cc3sl.ufpr.br/</a> <a href="https://cran.fiocruz.br/">https://cran.fiocruz.br/</a> <a href="https://yvs.fmvz.usp.br/CRAN/">https://yvs.fmvz.usp.br/CRAN/</a> <a href="https://brieger.esalq.usp.br/CRAN/">https://brieger.esalq.usp.br/CRAN/</a>
Bulgaria	<a href="https://flp.uni-sofia.bg/CRAN/">https://flp.uni-sofia.bg/CRAN/</a>

Automatic redirection to servers worldwide, currently sponsored by Rstudio
University of Science and Technology Houari Boumediene
Universidad Nacional de La Plata
CSIRO AARNET School of Mathematics and Statistics, University of Melbourne Curtin University
Wirtschaftsuniversität Wien
Patrick Wessa Belnet, the Belgian research and education network
Computational Biology Center at Universidade Estadual de Santa Cruz Universidade Federal do Parana Oswaldo Cruz Foundation, Rio de Janeiro University of Sao Paulo, Sao Paulo University of Sao Paulo, Piracicaba
Sofia University

**Step 3:** Click on “Download R” for either Windows or Mac (depends on your device)

## The Comprehensive R Archive Network

**Download and Install R**

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- [Download R for Linux \(Debian, Fedora/Redhat, Ubuntu\)](#)
- [Download R for macOS](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

**Source Code for all Platforms**

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2021-05-18, Camp Pontanezen) [R-4.1.0.tar.gz](#), read [what's new](#) in the latest version.
- Sources of [R alpha and beta releases](#) (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are [available here](#). Please read about [new features and bug fixes](#) before filing corresponding feature requests or bug reports.
- Source code of older versions of R is [available here](#).
- Contributed extension [packages](#)

**Questions About R**

- If you have questions about R like how to download and install the software, or what the license terms are, please read our [answers to frequently asked questions](#) before you send an email.

### What are R and CRAN?

R is 'GNU S', a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, etc. Please consult the [R project homepage](#) for further information.

CRAN is a network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for R. Please use the CRAN [mirror](#) nearest to you to minimize network load.

## Step 4: Click on "install R for the first time"

### R for Windows

Subdirectories:

<a href="#">base</a>	Binaries for base distribution. This is what you want to <a href="#">install R for the first time</a> .
<a href="#">contrib</a>	Binaries of contributed CRAN packages (for R >= 2.13.x; managed by Uwe Ligges). There is also information on <a href="#">third party software</a> available for CRAN Windows services and corresponding environment and make variables.
<a href="#">old contrib</a>	Binaries of contributed CRAN packages for outdated versions of R (for R < 2.13.x; managed by Uwe Ligges).
<a href="#">Rtools</a>	Tools to build R and R packages. This is what you want to build your own packages on Windows, or to build R itself.

Please do not submit binaries to CRAN. Package developers might want to contact Uwe Ligges directly in case of questions / suggestions related to Windows binaries.

You may also want to read the [R FAQ](#) and [R for Windows FAQ](#).

Note: CRAN does some checks on these binaries for viruses, but cannot give guarantees. Use the normal precautions with downloaded executables.

## Step 5: Then, click on "Download R for Windows" (the illustration is based on a Window's computer, but the installation for Mac is equivalent).

## R-4.1.0 for Windows (32/64 bit)

[Download R 4.1.0 for Windows](#) (86 megabytes, 32/64 bit)

[Installation and other instructions](#)  
[New features in this version](#)

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the [md5sum](#) of the .exe to the [fingerprint](#) on the master server. You will need a version of md5sum for windows: both [graphical](#) and [command line versions](#) are available.

### Frequently asked questions

- [Does R run under my version of Windows?](#)
- [How do I update packages in my previous version of R?](#)
- [Should I run 32-bit or 64-bit R?](#)

Please see the [R FAQ](#) for general information about R and the [R Windows FAQ](#) for Windows-specific information.

### Other builds

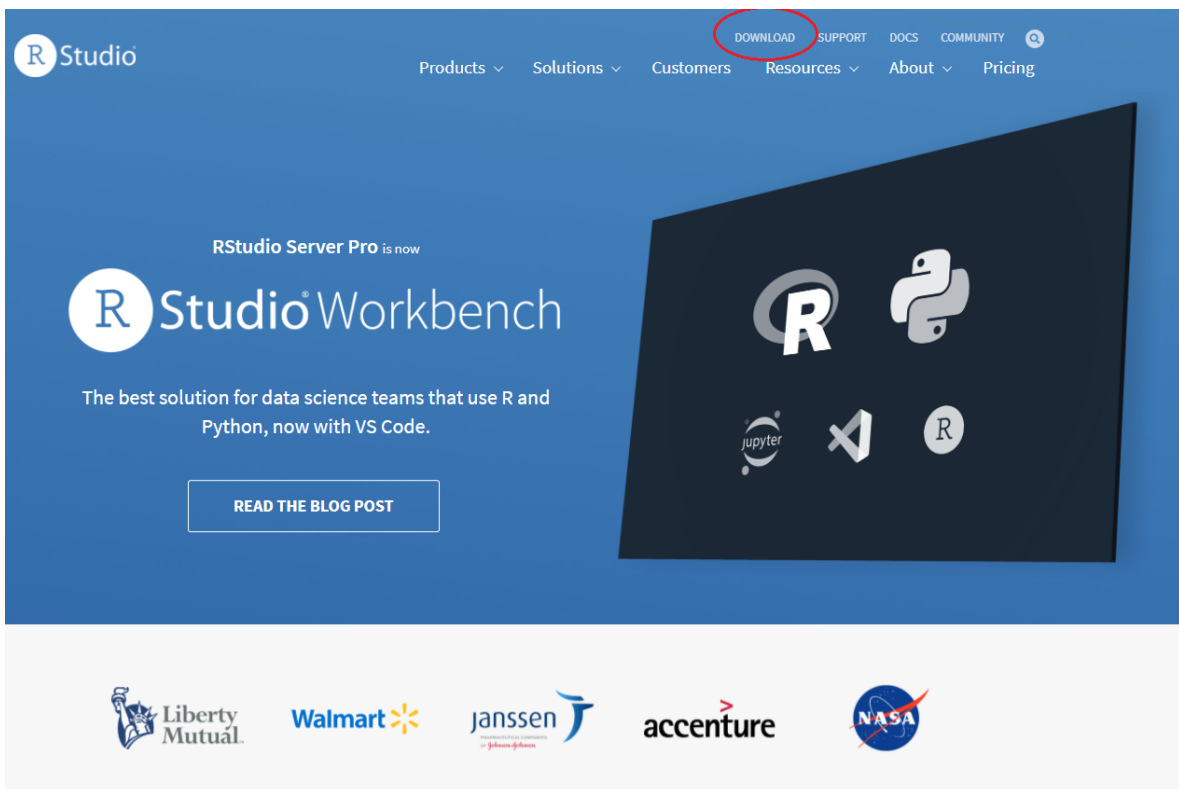
- Patches to this release are incorporated in the [r-patched snapshot build](#).
- A build of the development version (which will eventually become the next major release of R) is available in the [r-devel snapshot build](#).
- [Previous releases](#)

Note to webmasters: A stable link which will redirect to the current Windows binary release is [<CRAN.MIRROR>/bin/windows/base/release.html](#).

**Step 6:** Run the installation file on your device and follow the instructions from the installation program.

After successfully installing “R”, proceed with the installation “Rstudio”

**Step 1:** Go to [www.rstudio.com](http://www.rstudio.com), and click on “Download”.



The screenshot shows the RStudio website homepage. The navigation bar at the top includes links for Products, Solutions, Customers, Resources, About, and Pricing. The 'DOWNLOAD' link is highlighted with a red circle. The main content area features the RStudio Workbench logo and a 'READ THE BLOG POST' button. Below the main content are logos for partner organizations: Liberty Mutual, Walmart, Janssen, accenture, and NASA.


**Step 2:** Click on the “Free” version of Rstudio

# Download the RStudio IDE

## Choose Your Version

The RStudio IDE is a set of integrated tools designed to help you be more productive with R and Python. It includes a console, syntax-highlighting editor that supports direct code execution, and a variety of robust tools for plotting, viewing history, debugging and managing your workspace.

[LEARN MORE ABOUT THE RSTUDIO IDE](#)



**RStudio Team**

RStudio's recommended professional data science solution for every team. RStudio Team is a bundle of RStudio's popular professional software for data analysis, package management, and sharing data products.

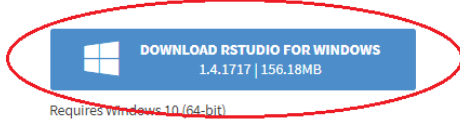
[Learn more about RStudio Team](#)

	RStudio Desktop	RStudio Desktop Pro	RStudio Server	RStudio Workbench
	Open Source License	Commercial License	Open Source License	Commercial License
	<b>Free</b>	<b>\$995</b> /year	<b>Free</b>	<b>\$4,975</b> /year (5 Named Users)
	<a href="#">DOWNLOAD</a>	<a href="#">BUY</a>	<a href="#">DOWNLOAD</a>	<a href="#">BUY</a>
	<a href="#">Learn more</a>	<a href="#">Learn more</a>	<a href="#">Learn more</a>	<a href="#">Evaluation   Learn more</a>
Integrated Tools for R	✓	✓	✓	✓
Priority Support		✓		✓

**Step 3:** Finally, click on “Download Rstudio”. Once you have downloaded the installation file, open it and follow the instructions that appear on your screen.

## RStudio Desktop 1.4.1717 - Release Notes

1. Install R. RStudio requires R 3.0.1+.
2. Download RStudio Desktop. Recommended for your system:



Requires Windows 10 (64-bit)



## All Installers

Linux users may need to import RStudio's public code-signing key prior to installation, depending on the operating system's security policy.

RStudio requires a 64-bit operating system. If you are on a 32 bit system, you can use an older version of RStudio.

OS	Download	Size	SHA-256
Windows 10	<a href="#">RStudio-1.4.1717.exe</a>	156.18 MB	71b36e64
macOS 10.14+	<a href="#">RStudio-1.4.1717.dmg</a>	203.06 MB	2cf2549d
Ubuntu 18/Debian 10	<a href="#">rstudio-1.4.1717-amd64.deb</a>	122.51 MB	e27b2645
Fedora 19/Red Hat 7	<a href="#">rstudio-1.4.1717-x86_64.rpm</a>	138.42 MB	648e2be0
Fedora 28/Red Hat 8	<a href="#">rstudio-1.4.1717-x86_64.rpm</a>	138.39 MB	c76f620a
Debian 9	<a href="#">rstudio-1.4.1717-amd64.deb</a>	123.29 MB	e4ea3a60
OpenSUSE 15	<a href="#">rstudio-1.4.1717-x86_64.rpm</a>	123.15 MB	e69d55db

## RStudio Interface

When you launch R studio on your computer, you should see four blocks of Rstudio interface:

The screenshot displays the RStudio environment. The top-left pane shows the source code editor with the following R script:

```

1 rm(list = ls())
2 graphics.off()
3 gc()
4 startTime = Sys.time()
5
6 # 1| Preparation -----
7 # 1.1| Libraries
8 myPackages = c('readxl','gridExtra','rpart','partykit','ggdendro','ggparty',
9               'factoextra','ppcor','glmnet','caret','gbm','Metrics',
10              'cobalt','gtools','fastDummies','openxlsx','extrafont',
11              'here','tidyverse','reshape2')
12 notInstalled = myPackages[!(myPackages %in% rownames(installed.packages()))]
13 if(length(notInstalled)) {
14   install.packages(notInstalled)
15 }
16
17 invisible(sapply(myPackages, library, character.only = TRUE, quietly = TRUE))
18 loadfonts(device = 'win', quiet = T)
19 options(scipen = 999) # Disable scientific notation.
20
21 # 1.2| Initial values -----
22 # 1: English.
23 # 0: Another.
24 language = 1
25
26

```

The bottom-left pane shows the console output:

```

CV: 1
CV: 2
CV: 3
CV: 4
CV: 5
Distribution not specified, assuming gaussian ...
CV: 1
CV: 2
CV: 3
CV: 4
CV: 5
Distribution not specified, assuming gaussian ...
CV: 1
CV: 2
CV: 3
CV: 4
CV: 5

```

The top-right pane shows the Environment pane with a list of objects:

- c.f.t1: 8 obs. of 1 variable
- characteristics: chr [1:140] "1" "2" "3" "4" "5" "6" "7" "8" ...
- clusterLabel: chr [1:4] "1" "Cluster" "Cluster" "تجميع"
- confirmation: chr [1:8] "1" "2" "No" "Yes" "No" "Yes" "نعم" "لا"
- correctionDecile: chr [1:4] "1" "decile" "p" "q"
- Oct: List of 15
- cv\_las1: List of 12
- dataF: 6006 obs. of 3 variables
- dataFrame: 40000 obs. of 16 variables
- dataFrameFinal: 40000 obs. of 16 variables
- dataFS: 6 obs. of 2 variables
- dataMini: 40000 obs. of 33 variables
- dataParticipation: 2941 obs. of 7 variables
- dataParticipationT: 2941 obs. of 7 variables

The bottom-right pane shows a plot of Squared error loss versus Iteration. The x-axis ranges from 0 to 500, and the y-axis ranges from 154 to 160. A blue curve starts at approximately 158.5 and decreases to about 154.5. A vertical dashed blue line is drawn at approximately iteration 150.

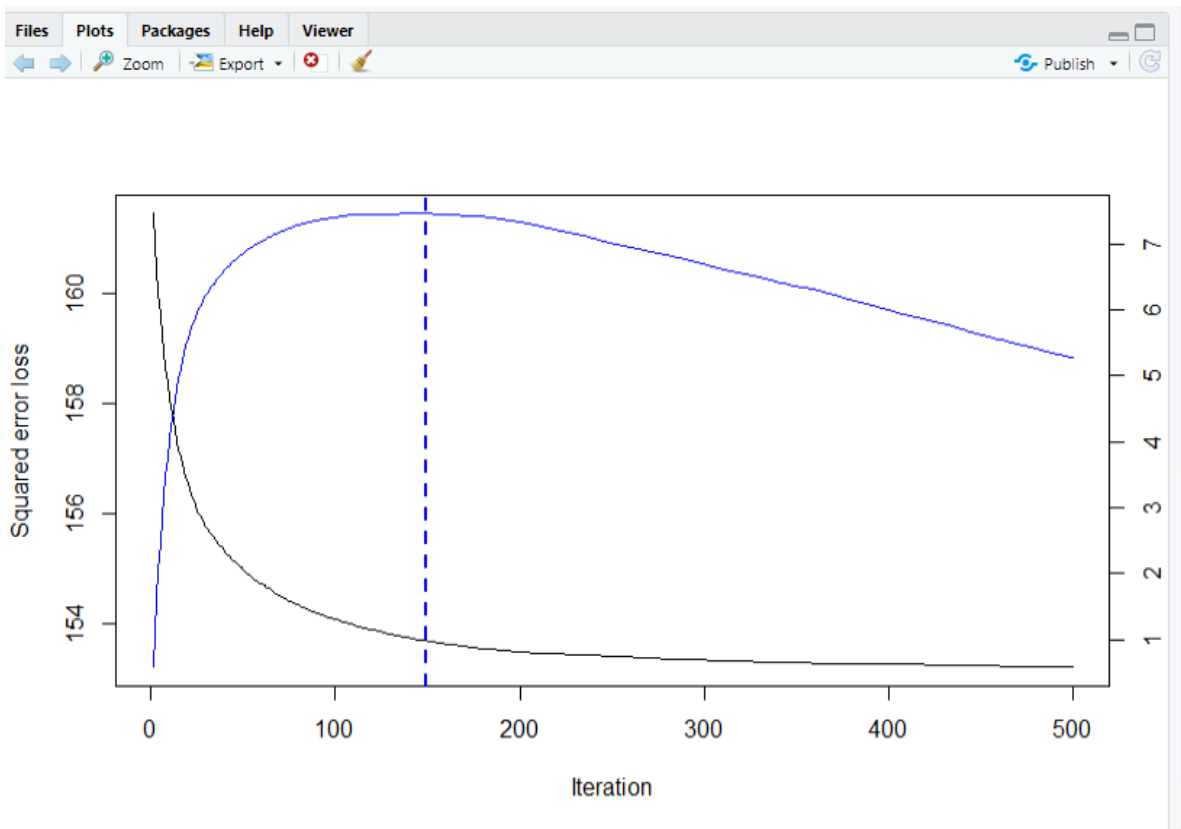
The first quadrant is called the source; here you will be able to write the code scripts, save it, see data bases, and define the code lines you want to execute.

This screenshot shows a closer view of the R script in the source editor. The code is identical to the one shown in the previous screenshot, including the package installation and library loading steps.

The second quadrant is called the environment and history and it will display all the variables that you have uploaded into the system. Among these variables you will find lists, functions, datasets, and other objects that will be useful to develop the corresponding exercises. This window also allow you to check history, connections, and tutorials, but these tabs are not relevant for the current manual.

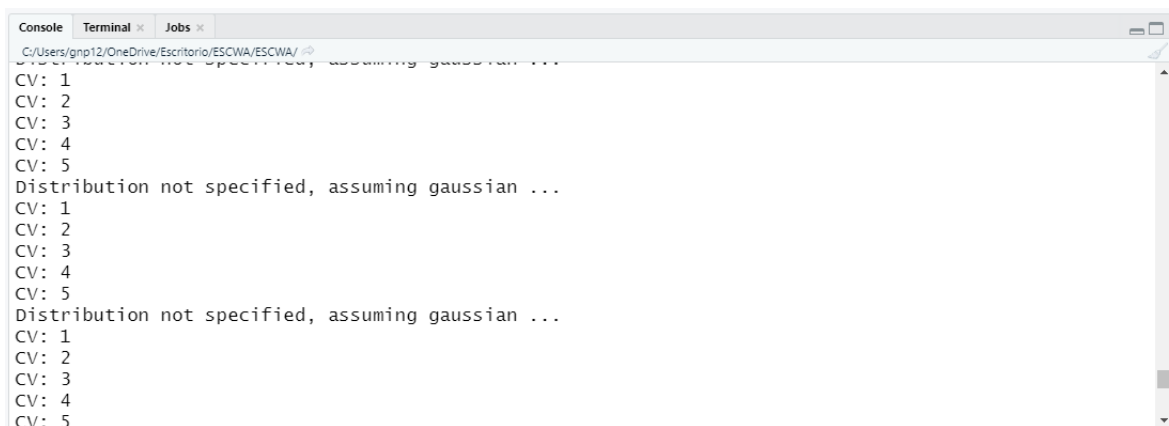
Environment	History	Connections	Tutorial
R - Global Environment			
c_fit1	8 obs. of 1 variable		
characteristics	chr [1:140] " 1" " 2" " 3" " 4" " 5" " 6" " 7" " 8..."		
clusterLabel	chr [1:4] "1" "Cluster" "Cluster" "تجمع"		
confirmation	chr [1:8] "1" "2" "No" "Yes" "No" "Yes" "نعم" "كلا"		
correctionDecile	chr [1:4] "1" "Decile" "D" "ك"		
ct	List of 15		
cv_las1	List of 12		
dataF	6006 obs. of 3 variables		
dataFrame	40000 obs. of 16 variables		
dataFrameFinal	40000 obs. of 16 variables		
dataFS	6 obs. of 2 variables		
dataMini	40000 obs. of 33 variables		
dataParticipation	2941 obs. of 7 variables		
dataParticipationT	2941 obs. of 7 variables		

The third quadrant, named here the multipurpose quadrant, is the most useful as it will allow you to quickly check your files, preview the plots that you have created, check the packages that are uploaded in the system (explained in the next section) and finally, check the help manuals of the software.





The final quadrant is the console, and it is where you will see the results of the code you run in the source section. In the console you can also write basic code lines, but this practice is not recommended as it doesn't let you to keep a clear trace as the source does.



```
CV: 1
CV: 2
CV: 3
CV: 4
CV: 5
Distribution not specified, assuming gaussian ...
CV: 1
CV: 2
CV: 3
CV: 4
CV: 5
Distribution not specified, assuming gaussian ...
CV: 1
CV: 2
CV: 3
CV: 4
CV: 5
```

## Packages and libraries

Packages represent groups of R functions and codes that the programme uses to do the statistical analysis. There is a set of basic packages that are used as part of the R source code and are directly accessible as part of the R installation, i.e. you don't need to upload in to the system. These basic packages contain the main functions that permit R to work, and perform typical statistical and graphical functions on datasets.

The packages are kept in the “libraries” and, if you want to use it, you have to let R know about it. To do this, you have to use the `library()` function.

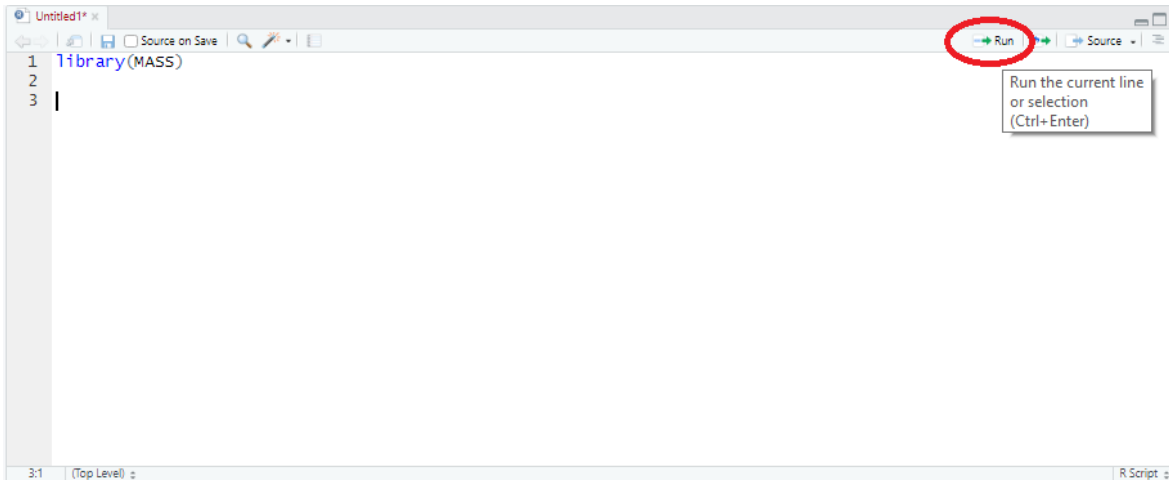
```
1 library(MASS)
```

Once you write the previous statement in the source quadrant, the natural question is how to execute this line (i.e. to let R know that you want it to perform the instruction). There are two ways to do this.

The first option is to select the command line you want to run and press “Ctrl+Enter” on the keyboard (recommended option).

```
1 library(MASS)
```

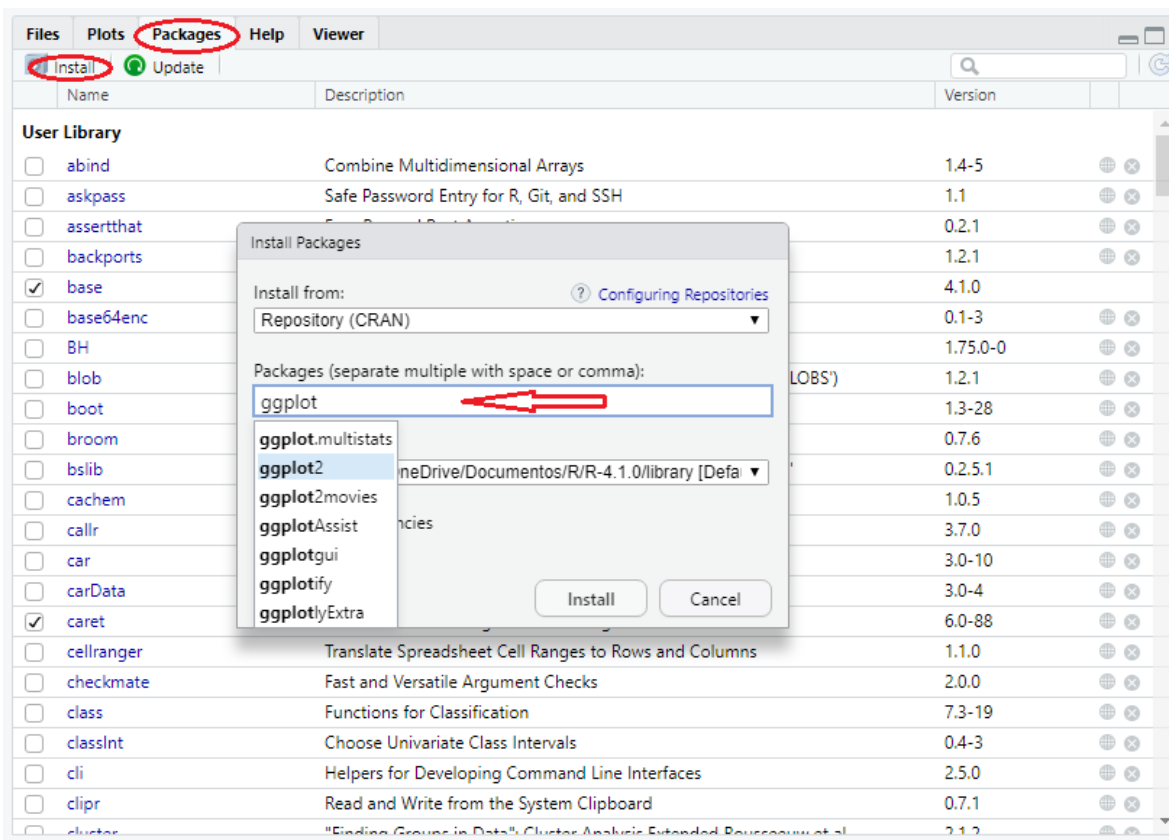
The second option is to select the command line and press in the bottom: “Run”.



### Installing packages

R is an open software, and every day new packages created by users appear to facilitate the analytical tasks of the users. In order to use them, these packages need to be downloaded and installed. The set-up process can be done in two ways.

The first way involves using the current tabs on the multipurpose quadrant. There you click on the Packages tab and then the Install bottom. There you can write the name of the package you want to download and install it. Notice that as you begin to write the name of the package you want to install (for the sake of example, we want to install “ggplot2”), the menu will help you giving you suggestions on the packages that are available. This will help you to avoid spelling mistakes.



You can also install a package a package by executing the function `install.packages()` in the command line. This option requires you to know the exact expelling of the program, but it helps you to keep your codes clean and complete. This can help you to improve teamwork and avoid coordination issues. For example, if you share your code with another person, and that code needs a package, then the code will not run. However, if you added this line to the code, the computer automatically installs the package in that computer and you will be able to run the code.

```
1 install.packages("ggplot2")
```

On a final note, remember that downloading the package is one thing and uploading it into R is another thing. Hence, always remember to load all the packages that you need by using the library function.

## Elements in R

Having covered the most structural elements of the program, this section reviews the types of objects that R uses to develop its work.

### Variables

The variables are evaluated by the system. R contains different types of variables: numeric, character and logical. To create a variable you just name it, and via an “=” sign you define it in the way you want.

The screenshot displays the RStudio interface. The script editor on the left contains the following code:

```
1 #Types of variables
2 A=4.5
3 B="House"
4 C=FALSE
5
6 #To see the variable in the console
7 C
```

The console at the bottom shows the execution of these commands:

```
> #Types of variables
> A=4.5
> B="House"
> C=FALSE
>
> #To see the variable in the console
> C
[1] FALSE
>
```

The Environment pane on the right shows the current environment (Global Environment) with the following variables:

Variable	Value
A	4.5
B	"House"
C	FALSE

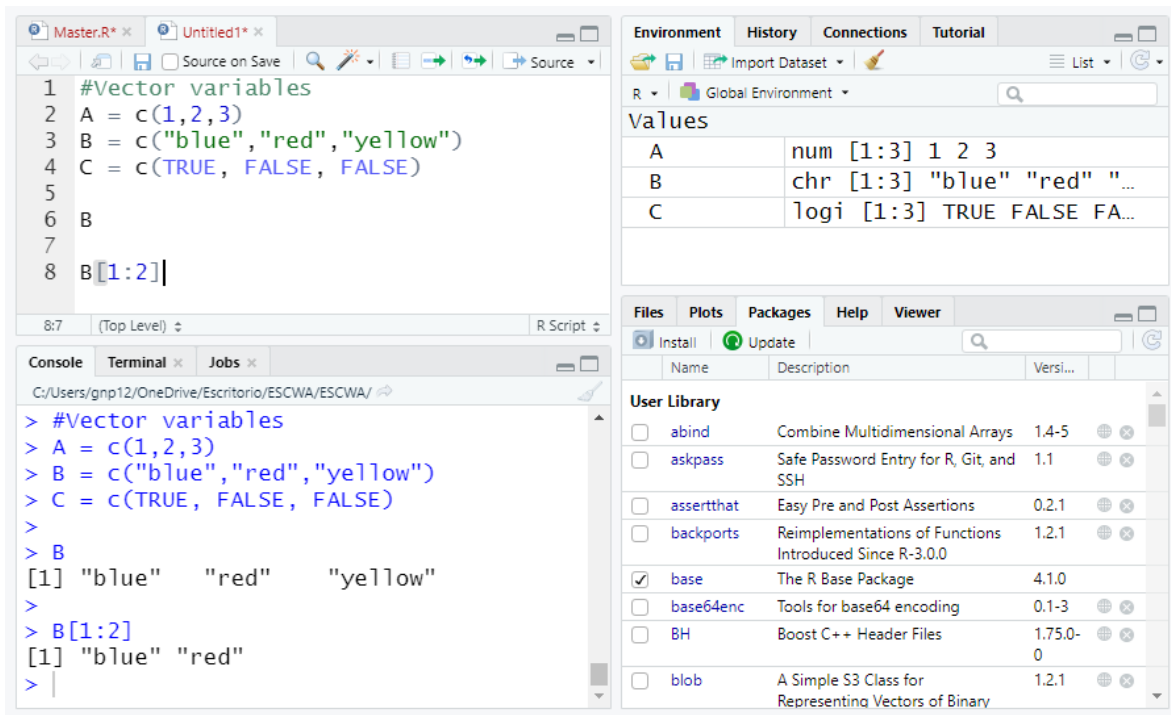
The Packages pane at the bottom right shows the installed packages in the User Library, including 'base' (checked), 'abind', 'askpass', 'assertthat', 'backports', 'base64enc', and 'BH'.

Notice two things:

1. As soon as you run the command lines, the environment and history quadrant will report the variables that you defined. However, you will not be able to see the values in the console. If you want to see them, you have to run the command only with the name of the variable, as in the example it is done with C.
2. There are some command lines beginning with “#”. R is design to stop reading a line after this sign is written. So you can use it to add comments to your code that makes it easy for other people to understand you.

## Vectors

Another basic element in R is an array or a vector that gathers variables of the same type: e.g. all the variables of the same vector are, either numeric, or character or logical.



The screenshot shows the RStudio interface. The script editor on the left contains the following R code:

```
1 #Vector variables
2 A = c(1,2,3)
3 B = c("blue","red","yellow")
4 C = c(TRUE, FALSE, FALSE)
5
6 B
7
8 B[1:2]
```

The console on the bottom left shows the execution of this code:

```
> #Vector variables
> A = c(1,2,3)
> B = c("blue","red","yellow")
> C = c(TRUE, FALSE, FALSE)
>
> B
[1] "blue" "red" "yellow"
>
> B[1:2]
[1] "blue" "red"
> |
```

The Environment pane on the right shows the following table of values:

Variable	Type	Value
A	num [1:3]	1 2 3
B	chr [1:3]	"blue" "red" "yellow"
C	logi [1:3]	TRUE FALSE FALSE

The Packages pane on the bottom right shows the installed packages in the User Library, including 'base' (The R Base Package, version 4.1.0).

In this example, notice the following things:

1. To create the vectors, a function `c()` is used. This function stands for concatenate, and as you see, it puts all the elements inside in a list.
2. Notice the subtle changes in the description of the variable in the second quadrant. For example, for variable A, it says "num" stating that is a numerical variable, and then it says [1:3] meaning that you have three entries indexed from number 1 to number 3.
3. When you call the vector, as in the example of B, the console displays all the elements.
4. However, if you want to specify a given item, you can use the square brackets and make explicit what entries you want. In the example, the code line requests to see only entries 1 and 2.

## Matrices

Matrices are similar to vectors, but with more dimensions. The next example shows you how matrices look like, and also how you can create them.

The screenshot displays the RStudio interface. The top-left pane contains the following R code:

```

1 #Vector variables
2 A = c(1,2,3)
3 B = c(4,5,6)
4
5 #Join them into matrices
6 C = rbind(A,B)
7 D = cbind(A,B)
8
9 C
10 D
11 dim(D)
12 D[2,2]

```

The bottom-left pane (Console) shows the execution of this code:

```

> #Vector variables
> A = c(1,2,3)
> B = c(4,5,6)
>
> #Join them into matrices
> C = rbind(A,B)
> D = cbind(A,B)
>
> C
  [,1] [,2] [,3]
A    1    2    3
B    4    5    6
> D
  A B
[1,] 1 4
[2,] 2 5
[3,] 3 6
> dim(D)
[1] 3 2
> D[2,2]
B
5
>

```

The top-right pane (Environment) shows the objects created in the workspace:

Object	Class	Dimensions	Values
C	num	[1:2, 1:3]	1 4 2 5 ...
D	num	[1:3, 1:2]	1 2 3 4 ...
A	num	[1:3]	1 2 3
B	num	[1:3]	4 5 6

The bottom-right pane (Packages) shows the installed and available packages in the user library.

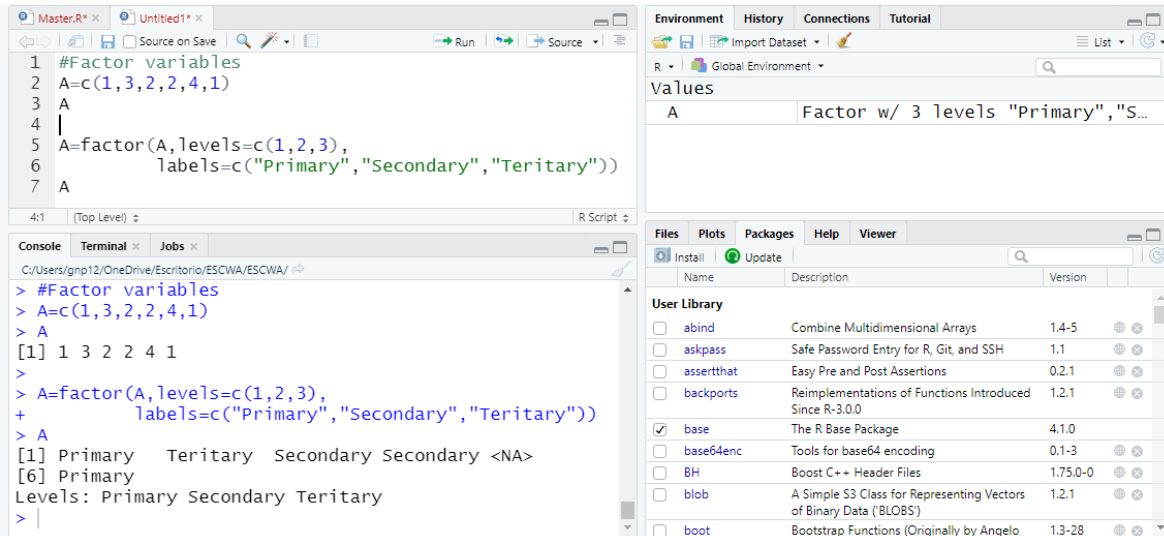
Name	Description	Vers...
<input type="checkbox"/> abind	Combine Multidimensional Arrays	1.4-5
<input type="checkbox"/> askpass	Safe Password Entry for R, Git, and SSH	1.1
<input type="checkbox"/> assertthat	Easy Pre and Post Assertions	0.2.1
<input type="checkbox"/> backports	Reimplementations of Functions Introduced Since R-3.0.0	1.2.1
<input checked="" type="checkbox"/> base	The R Base Package	4.1.0
<input type="checkbox"/> base64enc	Tools for base64 encoding	0.1-3
<input type="checkbox"/> BH	Boost C++ Header Files	1.75.0-0
<input type="checkbox"/> blob	A Simple S3 Class for Representing Vectors of Binary Data ('BLOBS')	1.2.1
<input type="checkbox"/> boot	Bootstrap Functions (Originally by Angelo Canty for S)	1.3-28
<input type="checkbox"/> broom	Convert Statistical Objects into Tidy Tibbles	0.7.6
<input type="checkbox"/> bslib	Custom 'Bootstrap' 'Sass' Themes for 'shiny' and 'rmarkdown'	0.2.5.1
<input type="checkbox"/> cachem	Cache R Objects with Automatic Pruning	1.0.5
<input type="checkbox"/> callr	Call R from R	3.7.0
<input type="checkbox"/> car	Companion to Applied Regression	3.0-10
<input type="checkbox"/> carData	Companion to Applied Regression Data Sets	3.0-4
<input checked="" type="checkbox"/> caret	Classification and Regression	6.0-88

The previous code goes by steps. First it creates two vectors, A and B, and then, using the functions `rbind()` and `cbind()`, it “pastes them” to form a matrix, where the first one attaches them by rows and the second one by columns. In contrast to a vector, a matrix has two dimensions, for that reason on the second quadrant you see a double structure mentioning first the number of rows and second the number of columns. A quick way to know the dimension is to use the function `dim()`. As you can see in the example, the output of this function is a vector with values 3 (for the three rows of D) and 2 (for the two columns of D).

Finally, similar to vectors you can use square brackets to call specific entries. Yet, as they have two dimensions, you have to specify values in both. In this example, calling `D[2,2]` shows us the value on the second row (first entry in the bracket) and the second column (second entry in the bracket).

## Factors

Factors are used to identify categories with variables. For example, when a survey asks for your education level, you like to see values such as “Primary”, “Secondary” or “Tertiary”, but the way they give the data set you see values such as 1, 2, or 3 (representing each education level). So you need to let the software know that those numbers are not from a numerical variable, but are representing specific categories. Factors are the way to do this exercise.



```
1 #Factor variables
2 A=c(1,3,2,2,4,1)
3 A
4 |
5 A=factor(A,levels=c(1,2,3),
6         labels=c("Primary","Secondary","Teritary"))
7 A
```

```
> #Factor variables
> A=c(1,3,2,2,4,1)
> A
[1] 1 3 2 2 4 1
>
> A=factor(A,levels=c(1,2,3),
+         labels=c("Primary","Secondary","Teritary"))
> A
[1] Primary  Teritary  Secondary Secondary <NA>
[6] Primary
Levels: Primary Secondary Teritary
```

The Environment pane shows the variable A as a Factor w/ 3 levels "Primary", "S...". The Packages pane shows the installed R packages, including base (4.1.0).

Following the previous code, the original vector has numeric values, so when you run it (lines 2 and 3) you just see numbers. However, after using the function `factor()`, the programme understands that these values represent things. Still, notice that this is not immediate, this function, in contrast to the previous ones that you have seen, has three elements. First you put the vector you want to specify. Second you specify the levels that are categories, and finally, in the same order as the levels, you write the labels that you want for these categories.

Now, notice that when the content of A is seen after the factor exercise, you do not see any number, but you see categories. Yet, there is an <NA> in the fifth position. As you can see, there are only three specified levels (1,2,3), but the original vector has a value “4”. Given that this level is not specified, the programme marks it as a missing value as it is probably caused by a typing error when the enumerators processes the survey.

## Data frames

In contrast to matrices, data frames can contain variables of different types. This allows you to work along with categories, numbers, characters, and logical variables in a single frame.

The screenshot displays the RStudio environment with the following components:

- Source Editor:** Contains R code for creating a data frame with variables Name, Age, Education, and Beneficiaries.
- Console:** Shows the execution of the code and the resulting data frame structure and values.
- Environment:** Shows the 'data' object with 3 observations and 4 variables.
- Files/Plots/Packages/Help/Viewer:** Shows the R package library with 'base' selected.

```

1 #Database example
2 edu=c(1,2,3)
3 edu=factor(edu, levels=c(1,2,3),
4             labels=c("Primary", "Secondary", "Tertiary"))
5 Name=c("Amira", "Jumana", "Carole")
6 Age=c(35,32,43)
7 Beneficiaries=c(TRUE, TRUE, FALSE)
8 data=data.frame(Name, Age, Education=edu, Beneficiaries)
9 data
10
11 data$Education
12 data$Age

```

```

> #Database example
> edu=c(1,2,3)
> edu=factor(edu, levels=c(1,2,3),
+            labels=c("Primary", "Secondary", "Tertiary"))
> Name=c("Amira", "Jumana", "Carole")
> Age=c(35,32,43)
> Beneficiaries=c(TRUE, TRUE, FALSE)
> data=data.frame(Name, Age, Education=edu, Beneficiaries)
> data
  Name Age Education Beneficiaries
1 Amira 35   Primary            TRUE
2 Jumana 32 Secondary            TRUE
3 Carole 43   Tertiary           FALSE
>
> data$Education
[1] Primary Secondary Tertiary
Levels: Primary Secondary Tertiary
> data$Age
[1] 35 32 43

```

Environment	History	Connections	Tutorial
R	Global Environment		
<b>Data</b>			
data	3 obs. of 4 varia...		
<b>Values</b>			
Age	num [1:3]	35 32 43	
Benefi...	logi [1:3]	TRUE TRU...	
edu	Factor w/ 3 levels ...		
Name	chr [1:3]	"Amira" "...	

Files	Plots	Packages	Help	Viewer
Install	Update			
Name	Description	Ver...		
<b>User Library</b>				
<input type="checkbox"/>	abind	Combine Multidimensional Arrays	1.4-5	
<input type="checkbox"/>	askpass	Safe Password Entry for R, Git, and SSH	1.1	
<input type="checkbox"/>	assertr...	Easy Pre and Post Assertions	0.2.1	
<input type="checkbox"/>	backpo...	Reimplementations of Functions Introduced Since R-3.0.0	1.2.1	
<input checked="" type="checkbox"/>	base	The R Base Package	4.1.0	
<input type="checkbox"/>	base64...	Tools for base64 encoding	0.1-3	
<input type="checkbox"/>	BH	Boost C++ Header Files	1.75.0-0	
<input type="checkbox"/>	blob	A Simple S3 Class for Representing Vectors of Binary Data (BLOBs)	1.2.1	
<input type="checkbox"/>	boot	Bootstrap Functions (Originally by Angelo Canty for S)	1.3-28	
<input type="checkbox"/>	broom	Convert Statistical Objects into Tidy Tibbles	0.7.6	
<input type="checkbox"/>	bslib	Custom 'Bootstrap' 'Sass' Themes for 'shiny' and	0.2.5.1	

Let's highlight some elements of the previous code. First, the code creates four variables: Name, edu, Age, and Beneficiaries, referring to three ladies participating in a social protection program. Each variable has a different type. There are numbers (age), categories (edu), characters (names), and logical (beneficiaries). You need to have them as a single object to be able to work with them, so you use the `data.frame()` function to put them together. The function is easy, you add the vectors you want and it automatically arranges them. However, notice subtly that for education, there is a difference. By using the trick presented in the code, you can adjust the names on the dataset so that the data frame has exactly the name you want. Finally, in contrast to matrices, if you want to call a value from a data frame you use the "\$" symbol with the name of the variable.

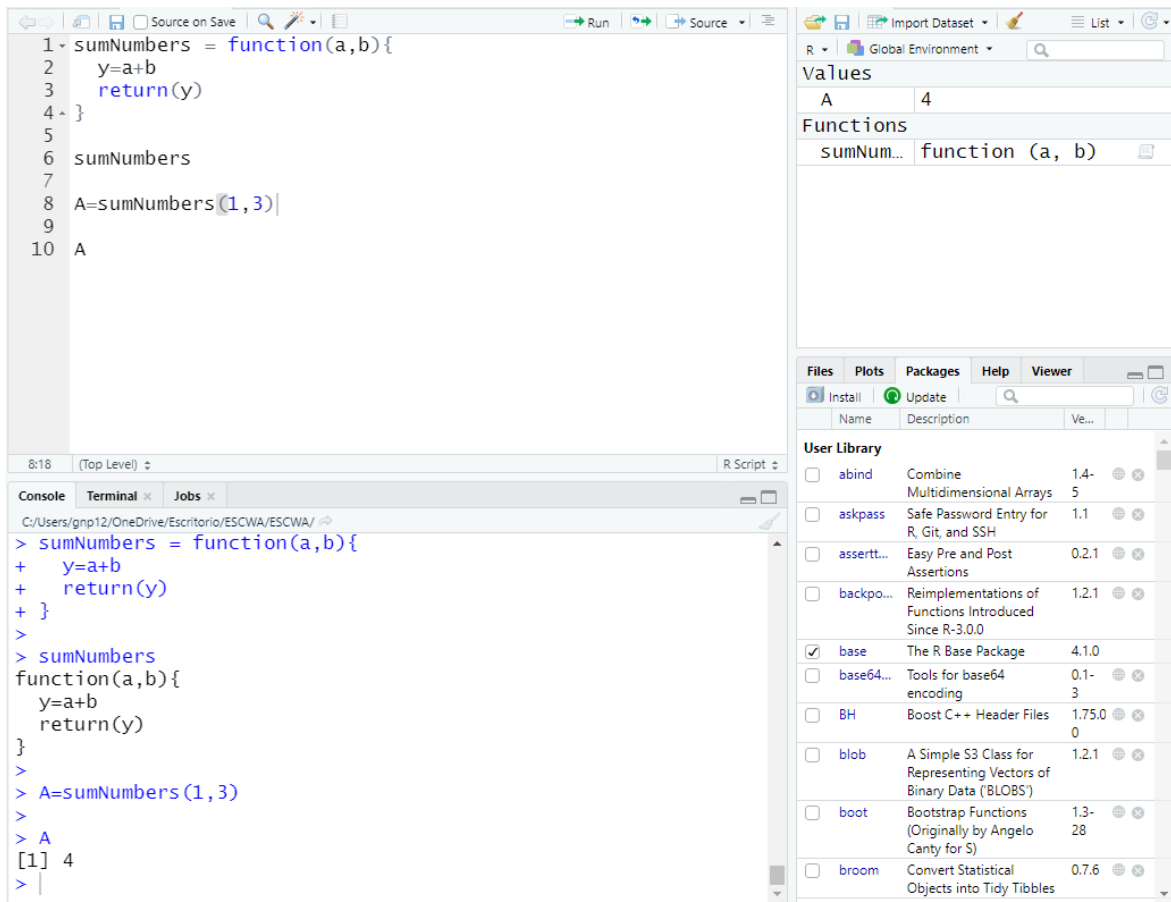
## Function

A Function is a code or set of instructions to perform a specific task. It can receive some input values and then, work with them until a desired result is obtained. Then, the code gives the user the result. R comes with several built-in functions, but it also allows users to create their own. Currently you have seen several of them, like `rbind()` or `factor()`, but this section discusses the point deeper.

A function is represented by the following:

```
Function-name = function (arg_1, arg_2, ...) {  
  Expression  
}
```

- **Function-name:** The name of the function that is stored in R.
- **Arguments:** Inputs that a function requires. They are optional and can have default values, but in case they are needed, they have to be defined.
- **Expression:** A set of statements that defines what the function does.



In the previous example, the outlined structure was used to create a function. First you tell R that the function has two inputs “a”, and “b”. Then you explain that the function adds them in “y” and reports its value. Running only the function (lines 1-4) does not show results because it’s an instruction not a process. However, in the second quadrant a comment appears to show that you created that function. Also, if you call the function, the only thing you see is its script. In order to use it, you to call it, and then put parenthesis with the inputs. So, as you can see in line 8, it calls the function to add 1 and 3, and saves it in A, where A is 4.



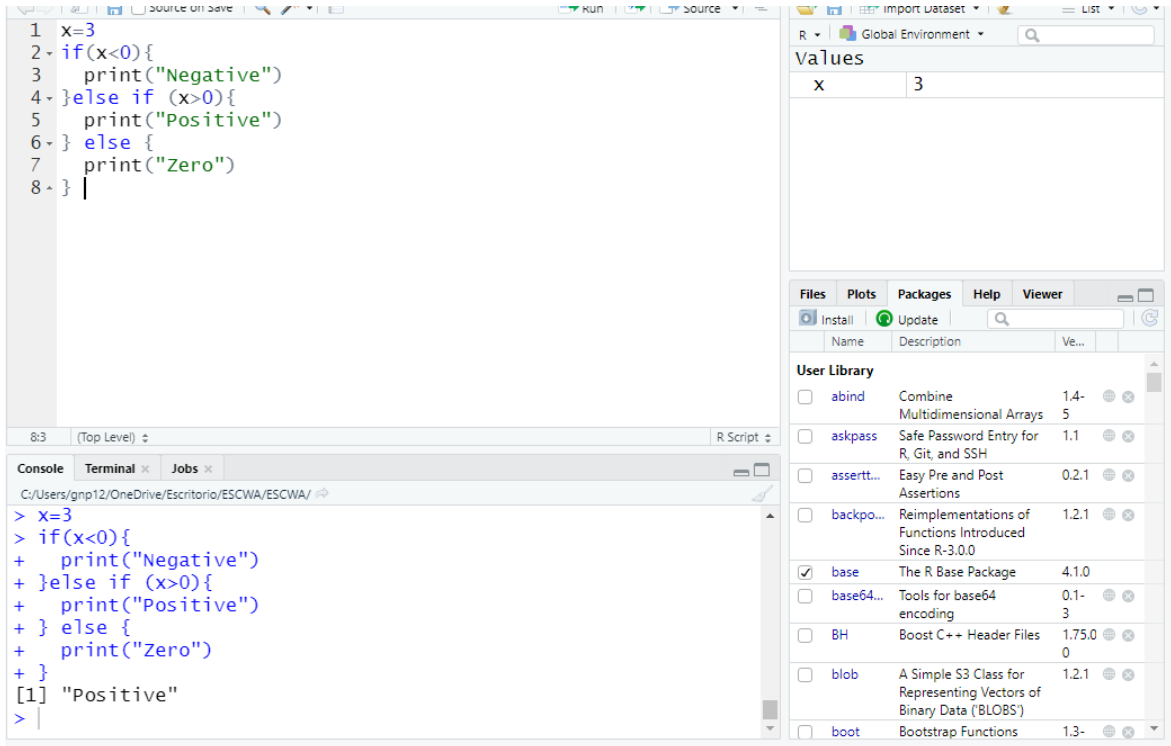
Besides having your thoughts structured, a function can help you to have a cleaner code. Notice that even though the function uses a “y” variable, this doesn’t appear in the output because it is an internal process.

### Conditional statements

The *if/else* command assesses a condition and depending on it, the command guides the code into two different outputs. After the assessment of the condition (statement1 in the example below), if the value indicates that it is TRUE then the first declaration is executed (statement2 in the example below); in contrast, if the condition is FALSE, then the second declaration following “else” is executed (statement3 in the example below). The main composition is:

```
if (statement1) {  
  statement2  
} else {  
  statement3  
}
```

In the example below, the command *if* performs the following assessment: if x is less than zero, it will return or print “Negative number” text on the screen; if x is greater than zero, it will return “Positive number” text; otherwise it will return “Zero”.



Notice from this example that the last *else* has no conditions because it will print whatever goes inside as long as it haven’t satisfied any of the previous instruction

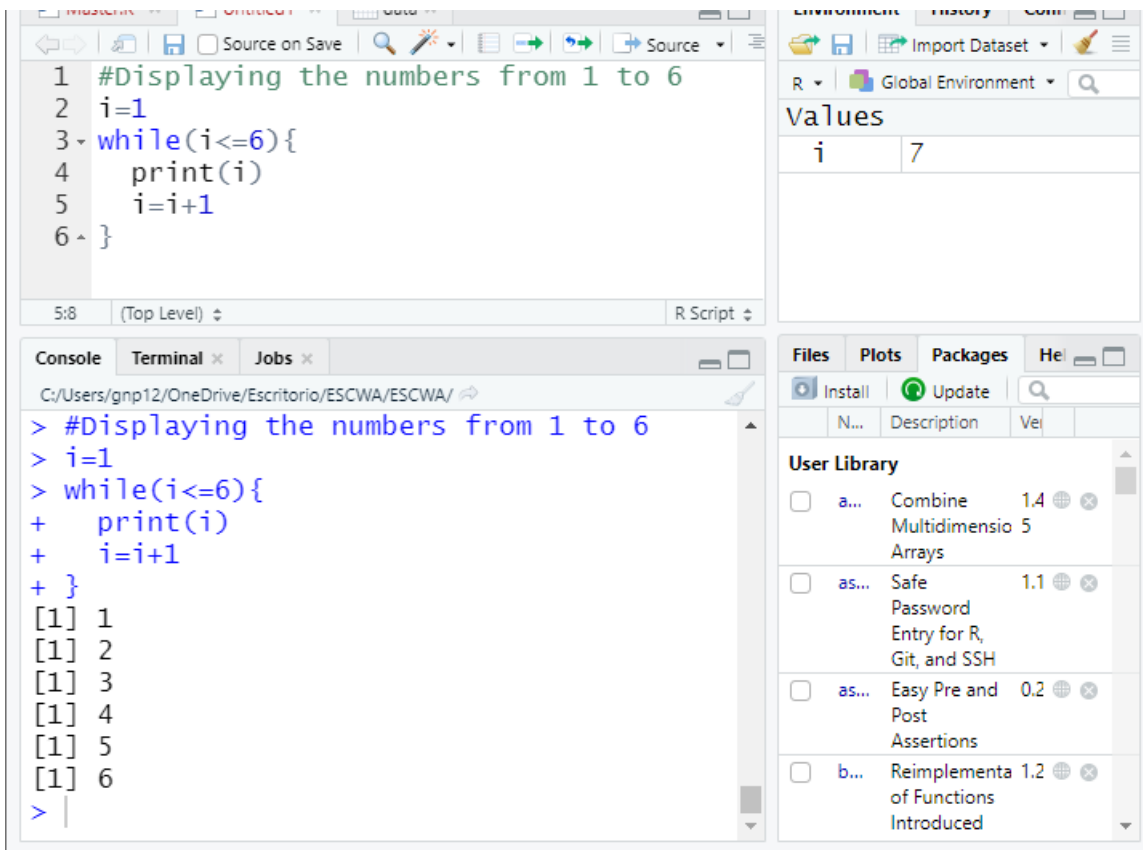
## Loops

Finally, some codes need instructions to be repeated for several times. For this case, R presents two easy ways to do it.

### While loop

```
while  
(test_expression) {  
  statement
```

In this case, you need to create a stop condition and once that condition is met, the code inside the while stops repeating.



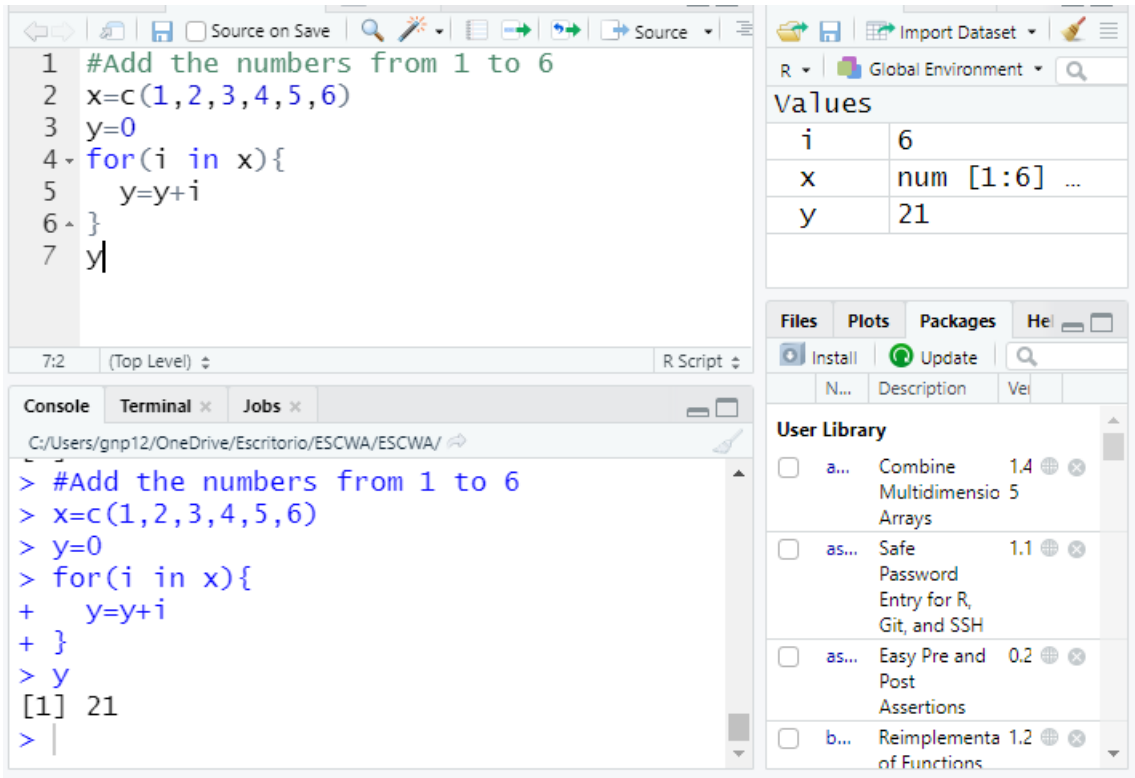
The example code displays numbers from 1 to 6. However, there are important elements to highlight:

1. It begins initializing the counter in 1.
2. Once inside the loop, it updates “i” to increase one value. If you don’t do that, the condition will never be reached, and the computer will run forever.
3. The second quadrant only shows the last result.

## For loop

```
for (element in a sequence)
{
  statement
}
```

For loops do not have a stopping condition, but you have to specify the exact values that you want it to evaluate.



The screenshot shows the RStudio interface. The script editor contains the following code:

```
1 #Add the numbers from 1 to 6
2 x=c(1,2,3,4,5,6)
3 y=0
4 for(i in x){
5   y=y+i
6 }
7 y
```

The console shows the execution of this code:

```
> #Add the numbers from 1 to 6
> x=c(1,2,3,4,5,6)
> y=0
> for(i in x){
+   y=y+i
+ }
> y
[1] 21
```

The Environment pane on the right shows the following values:

Variable	Value
i	6
x	num [1:6] ...
y	21

The Package pane shows the following installed packages:

Package Name	Version
Combine	1.4
Multidimensio 5 Arrays	
Safe	1.1
Easy Pre and Post Assertions	0.2
Reimplementa of Functions	1.2

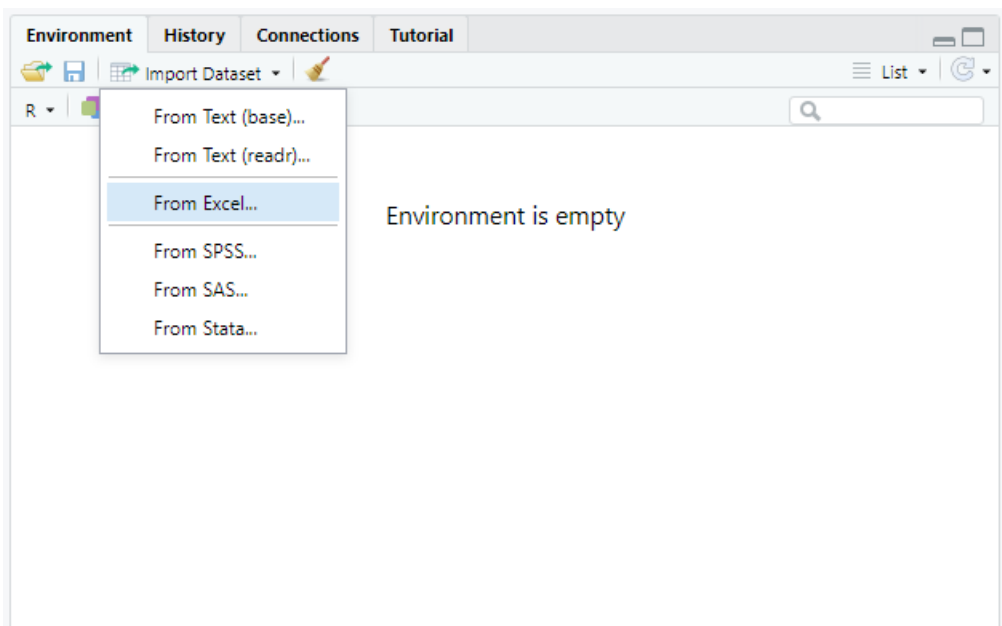
Notice from the previous exercise that to do the loop, it was important to specify a vector  $x$  with all the values that are relevant. Then, you initialize  $y$  in 0, and then in each step you add to the original value of  $y$  each of the entries of  $x$ , until you get the result of 21 which is  $0+1+2+3+4+5+6$ .

## Chapter 2: Data management

The purpose of the following section is to work on how to manage datasets. For that purpose, a simulated case has been built to help you understand better the most relevant functions associated with data management. During each of the following exercises you learn different perspectives of the dataset that will be useful later for the analysis of social assistance programmes.

### Uploading data

In order to begin the analysis of a dataset, the first step is to know how to upload it to the file. For this exercise, you will use the “CleanData.xlsx” file that is inside the Input folder and there, it is inside the Data folder. While this example is done based on an excel file, a similar process can be done by uploading data from different formats.



To upload the dataset, go to the second quadrant and click on “Import Dataset” there as this example shows, click on “From Excel...”

Import Excel Data

File/URL:  
C:/Users/gnp12/OneDrive/Escritorio/ESCWA/ESCWA/input/Data/CleanData.xlsx

Data Preview:

region (character)	gender (character)	age (double)	education (character)	employment (character)	incomeQuantile (character)	householdMember (double)	popRoom (double)	incomeSources (double)	consumptionShare (double)	publicServiceShare (double)	indebtiness (double)	internetAcces (character)	healthAcces (character)
D	Male	44	Primary	Employed	D3	3	NA	NA	0.57	0.30	0.01	Yes	No
E	Male	40	Bachelor	Employed	D3	3	NA	NA	0.73	0.17	0.01	No	Yes
E	Female	19	Primary	Employed	D3	4	NA	NA	0.74	0.19	0.17	Yes	Yes
A	Male	41	Primary	Employed	D3	3	NA	NA	0.58	0.15	0.04	Yes	Yes
A	Female	23	Primary	Employed	D4	4	NA	NA	0.59	0.18	0.15	Yes	Yes
A	Male	46	Secondary	Unemployed	D10	4	NA	NA	0.55	0.17	0.19	No	Yes
A	Male	25	Primary	Employed	D4	3	NA	NA	0.49	0.15	0.02	Yes	Yes
A	Female	45	Primary	Out of labor force	D7	5	NA	NA	0.50	0.19	0.19	No	Yes
A	Male	25	Secondary	Employed	D9	3	NA	NA	0.79	0.17	0.00	Yes	No
C	Female	19	Secondary	Employed	D3	3	NA	NA	0.59	0.18	0.00	Yes	Yes
F	Male	29	Secondary	Out of labor force	D5	5	NA	NA	0.67	0.19	0.02	Yes	Yes
A	Female	55	Primary	Out of labor force	D2	2	NA	NA	0.64	0.16	0.03	No	Yes
D	Female	19	TVET	Employed	D1	6	NA	NA	0.68	0.16	0.10	Yes	No
B	Female	19	TVET	Employed	D6	6	3.000000	3	0.58	0.19	0.18	No	No
A	Female	30	Primary	Employed	D10	3	NA	NA	0.58	0.17	0.10	No	No
F	Female	36	Postgrade	Employed	D9	3	NA	NA	0.57	0.18	0.33	Yes	Yes
G	Male	54	Primary	Employed	D6	3	NA	NA	0.67	0.17	0.02	Yes	Yes
F	Female	30	TVET	Employed	D2	3	NA	NA	0.76	0.18	0.01	Yes	No
G	Female	43	Primary	Unemployed	D1	4	NA	NA	0.62	0.16	0.50	Yes	Yes
E	Female	65	Secondary	Employed	D6	2	NA	NA	0.53	0.17	0.06	Yes	No

Import Options:

Name: CleanData Max Rows:   First Row as Names

Sheet: Default Skip:   Open Data Viewer

Range: A1:D10 NA

```
Code Preview:
library(readxl)
CleanData <- read_excel("Input/Data/CleanData.xlsx")
View(CleanData)
```

Reading Excel files using readxl

Import Cancel

In this emergent window, you see the bottom “Browse” where you can find the relevant dataset and upload it. Once it is uploaded and you verify that all the parameters are OK, you can click import. However, it is recommended to notice the code. If you copy these codes into your main script, then you do not need to run everything from 0 again, but you can run it directly from the fist quadrant.

ESCWA - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function Addins

Environment History Connections Tutorial

R Global Environment

Data

CleanD... 40000 obs. of 16 ...

Files Plots Packages Help Viewer

New Folder Delete Rename More

OneDrive > Escritorio > ESCWA > ESCWA > Code

Name	Size
..	
.Rhistory	25.3 KB
Aux - Simulation.R	15.8 KB
Chapter 00 - Descriptive Statistics.R	68 KB
Chapter 01 - Profiling of beneficiari...	27 KB
Chapter 02 - Targeting characterist...	12.1 KB
Chapter 03 - Coverage evaluation.R	41.6 KB
Master.R	2 KB

Console Terminal Jobs

```
C:/Users/gnp12/OneDrive/Escritorio/ESCWA/ESCWA/
> library(readxl)
> CleanData <- read_excel("Input/Data/CleanData.xlsx")
> View(CleanData)
>
```

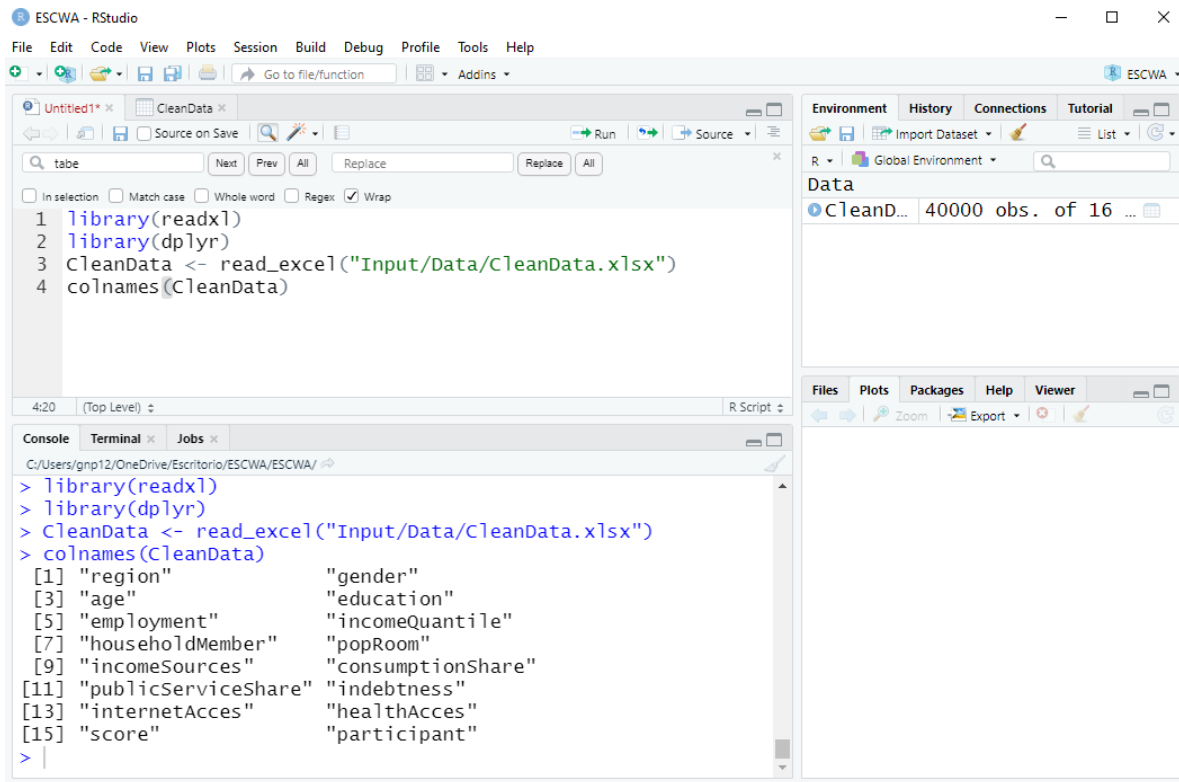
You will now be able to visualize your dataset in R and in the second quadrant you will find a quick summary saying that you have 40000 observations from 16 variables.

### Managing the data

To understand better the dataset, you need to work more on its content and for that purpose this section relies on the package “dplyr”. Remember that for any package you want to use, you need to upload the library first, so please make sure to include this first line in any code you create:

# 1 library(dplyr)

The library dplyr is very useful for data management as it allows to organize complex processes into simple segmented instructions. However, to do it in a proper way, it is important to know the variables in the dataset and for that purpose the best starting point is by knowing the variables inside.



The screenshot shows the RStudio interface. The editor window contains the following R code:

```
1 library(readxl)
2 library(dplyr)
3 CleanData <- read_excel("Input/Data/CleanData.xlsx")
4 colnames(CleanData)
```

The console window shows the execution of the code and the output of the `colnames()` function:

```
> library(readxl)
> library(dplyr)
> CleanData <- read_excel("Input/Data/CleanData.xlsx")
> colnames(CleanData)
 [1] "region"          "gender"
 [3] "age"             "education"
 [5] "employment"     "incomeQuantile"
 [7] "householdMember" "popRoom"
 [9] "incomeSources"  "consumptionShare"
[11] "publicServiceShare" "indebttness"
[13] "internetAcces"  "healthAcces"
[15] "score"          "participant"
>
```

The Environment pane on the right shows a data object named 'CleanD...' with 40000 observations and 16 variables.

Notice that in this command line we copied part of the code used to upload the data so that now we can do it automatically, but we added the line for the dplyr package so that it uploads automatically. Please notice that the path of the files depends on the computer, so the green text can change depending on the computer you are using.

Now, the new function that is presented here is `colnames()`. This function allows you to understand better the names of the variables within the file. Depending on the data set, sometimes names are confusing, so it's always recommended to clarify any doubt with the data source to understand each of them. In this case, this is the description of the dataset.

**Dataset description:**

The dataset was collected from the social protection program of the kingdom of AP. The data includes the information of 40,000 individuals, and 16 variables from them.

region: The kingdom is divided in 6 regions, represented by the letters A, B, C, D, E, F, G.

gender: This variable represents the gender of the individuals "Male" or "Female".

age: This is a numerical variable with the age in years of the individual.

education: Describes the maximum education level achieved by the individual. The categories are "Primary", "Secondary", "Bachelor", TVET", "Postgrade" (Yes, the data comes with a spelling mistake that we need to correct in the following steps).

employment: Describe the labor force situation of the individual (In this case the instruction does not give us the categories and we will check them by ourselves).

incomeQuantile: The income quantile of the individual. It goes from D1 (the poorest) to D10 (the richest).

householdMember: Number of individuals in the household.

popRoom: Number of people per room. This variable is only registered for individuals that participate in the social aid program.

incomeSources: Number of income sources in the household. This variable is only registered for individuals that participate in the social aid program.

consumptionShare: Share of expenditures dedicated to food consumption. The kingdom records this variable from a different source, so it is available for all individuals, even if they are not in the program.

publicServiceShare: Share of expenditures dedicated to public services. The kingdom records this variable from a different source, so it is available for all individuals, even if they are not in the program.

indebtness: Indebtnex index calculated by the financial sector. The kingdom records this variable from a different source, so it is available for all individuals, even if they are not in the program.

internetAccess: "Yes" if the individual has access to internet, "No" otherwise. The kingdom records this variable from a different source, so it is available for all individuals, even if they are not in the program.

healthAccess: "Yes" if the individual has access to health services, "No" otherwise. The kingdom records this variable from a different source, so it is available for all individuals, even if they are not in the program.

score: Using the previous seven variables, the government calculates a score and based on the score people can access the programme. This value is only available for beneficiaries.

participant: "Yes" if the individual participates in the programme, "No" otherwise.

While inspecting the list of variables, now it is clear why some variables appear as NA in the dataset.

	region	gender	age	education	employment	incomeQuantile	householdMember	popRoom	ir
26324	F	Female	42	Primary	Employed	D1	3	NA	
26325	A	Male	31	Primary	Employed	D9	3	NA	
26326	D	Male	19	Secondary	Unemployed	D8	2	0.6666667	
26327	B	Male	36	Secondary	Employed	D9	2	NA	
26328	B	Male	19	Secondary	Employed	D10	2	NA	
26329	D	Male	23	TVET	Employed	D3	5	NA	
26330	F	Female	57	Primary	Employed	D6	5	NA	
26331	B	Male	28	Secondary	Employed	D6	2	NA	
26332	A	Male	40	TVET	Employed	D7	3	NA	
26333	F	Female	50	Secondary	Out of labor force	D2	2	NA	
26334	D	Male	19	Secondary	Employed	D7	1	NA	
26335	G	Male	28	TVET	Employed	D6	3	0.7500000	

For example, notice that observations 26325 has NA in popRoom but 26326 has a value. This means that the first one represents a person who is not a beneficiary of the program, while the second one represents a beneficiary.

The description of the data set also allowed us to evidence some elements we wish to correct such the spelling mistake in education, and some additional information we would like to gather, such as the categories of employment. As it was explained in the previous chapter, some commands from a data frame can be used directly as if they refer to vectors, as long as we mentioned them correctly.

```

1 library(readxl)
2 library(dplyr)
3 CleanData <- read_excel("Input/Data/CleanData.xlsx")
4 unique(CleanData$employment)

```

```

> library(readxl)
> library(dplyr)
> CleanData <- read_excel("Input/Data/CleanData.xlsx")
> unique(CleanData$employment)
[1] "Employed"      "Unemployed"
[3] "Out of labor force"
>

```

Notice that to solve the problem of the unknown information, using vector functions as `unique()` is sufficient for the task. By applying this command to the employment vector, the console shows that this variable only has three categories.



The correction of the name can also be done, by using these basic functions, but for this case the `dplyr` package can make things easier. However, before using it, it is important to introduce the command pipeline “`%>%`”. Once this symbol is used, it means that you can use whatever is in the left of the symbol as an input to the function that is on the right.

```

1 library(readxl)
2 library(dplyr)
3 CleanData <- read_excel("Input/Data/CleanData.xlsx")
4 unique(CleanData$education)
5 CleanDataNew=CleanData%>%
6   mutate(education=ifelse(education%in%c("Postgrade"),"Postgrad",education))
7 unique(CleanDataNew$education)

```

```

> library(readxl)
> library(dplyr)
> CleanData <- read_excel("Input/Data/CleanData.xlsx")
> unique(CleanData$education)
[1] "Primary" "Bachelor" "Secondary" "TVET" "Postgrade"
> CleanDataNew=CleanData%>%
+   mutate(education=ifelse(education%in%c("Postgrade"),"Postgrad",education))
> unique(CleanDataNew$education)
[1] "Primary" "Bachelor" "Secondary" "TVET" "Postgrad"
>

```

In this example, the focus is on lines 5 and 6. In line 5 you create a new data frame called “CleanDataNew” that is the same as “CleanData”, BUT, then you add the pipeline, and add a new instruction in line six. This instruction is the function `mutate()`, that allows you to modify existing variables or to create new variables according to a given criteria. In this case that criteria is in itself another function called `ifelse()`, as the name suggests, it has three parts, the first part is the condition. Here, the condition is for those values in education that are equal to “Postgrade”. The second part is what happen if the condition is true. In this case, we want to change the spelling to “Postgrad”. Finally, if the condition is not achieved then you want to have the same value as it originally has (i.e. education). In this way, with a set of intuitive commands the spelling has been corrected.

The function `mutate()` is quite open and allows us also to create new variables. In the following example the command is used to create a new variable that measures the age in days. For this purpose, a new variable is created with the name “ageDays” and is calculated as  $age \times 365$ . Notice that the function `mutate()` performs this task automatically and allocates this new variable at the end of the dataset.

```
ESCWA - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function Addins
CleanData.R
1 library(readxl)
2 library(dplyr)
3 CleanData <- read_excel("Input/Data/CleanData.xlsx")
4 CleanDataNew=CleanData%>%
5   mutate(ageDays=age*365)
6   colnames(CleanDataNew)

Environment History Connections Tutorial
Data
CleanData 40000 obs. of 16 variables
CleanDataNew 40000 obs. of 17 variables

Files Plots Packages Help Viewer
Zoom Export

Console Terminal Jobs
C:/Users/gmp12/OneDrive/Escritorio/ESCWA/ESCWA/
> CleanDataNew=CleanData%>%
+   mutate(ageDays=age*365)
> colnames(CleanDataNew)
[1] "region" "gender" "age"
[4] "education" "employment" "incomeQuantile"
[7] "householdMember" "popRoom" "incomeSources"
[10] "consumptionShare" "publicServiceShare" "indebtiness"
[13] "internetAcces" "healthAcces" "score"
[16] "participant" "ageDays"
>
```

Sometimes datasets are quite large and include multiple variables that are not relevant for the exercise that you need to perform. In these cases, there are functions that help you select only the variables that you want or remove those that you do not want.

```

1 library(readxl)
2 library(dplyr)
3 CleanData <- read_excel("Input/Data/CleanData.xlsx")
4 CleanDataNew=CleanData%>%
5   select(c("region", "age", "gender"))
6 colnames(CleanDataNew)
7 CleanDataNew=CleanData%>%
8   select(-c("region", "age", "gender"))
9 colnames(CleanDataNew)
10
11
12
13
14
15
16
9:23 (Top Level) R Script

```

```

Console Terminal Jobs
C:/Users/gnp12/OneDrive/Escritorio/ESCWA/ESCWA/
> library(dplyr)
> CleanData <- read_excel("Input/Data/CleanData.xlsx")
> CleanDataNew=CleanData%>%
+   select(c("region", "age", "gender"))
> colnames(CleanDataNew)
[1] "region" "age" "gender"
> CleanDataNew=CleanData%>%
+   select(-c("region", "age", "gender"))
> colnames(CleanDataNew)
[1] "education" "employment" "incomeQuantile"
[4] "householdMember" "popRoom" "incomeSources"
[7] "consumptionShare" "publicServiceShare" "indebtness"
[10] "internetAcces" "healthAcces" "score"
[13] "participant"
>

```

As shown in the previous example, the key function is `select()`. However, in each case you use it differently. In the first case, you are selecting only the three variables that you want to show (and notice that they appear in the same order that you selected them). In the second case, the code adds a subtly “-” sign before the vector and this allows the programme to know that the idea is to remove these variables from the data set.

Similar to the excess of columns, sometimes there are more observations than those needed in the dataset. In this case, the function `filter()` becomes a useful tool to only select the observations that satisfy the conditions needed. The most useful logical instructions here are:

<	smaller than
>	greater than
==	equal to
<=	less or equal to
>=	greater or equal to
%in%	Value in the vector (we used this command before in the <i>ifelse()</i> )
&	and
	or

The screenshot shows an R Studio window with the following R code in the script editor:

```

1 library(readxl)
2 library(dplyr)
3 CleanData <- read_excel("Input/Data/CleanData.xlsx")
4 CleanDataNew=CleanData%>%
5   filter(age<30 & employment %in% c("Employed", "Unemployed"))
6
7
8
9
10
11

```

The Environment pane on the right shows the following data objects:

Object	Description
CleanData	40000 obs. of 16 variables
CleanDataNew	14639 obs. of 16 variables

The Console pane shows the execution of the code:

```

> library(readxl)
> library(dplyr)
> CleanData <- read_excel("Input/Data/CleanData.xlsx")
> CleanDataNew=CleanData%>%
+   filter(age<30 & employment %in% c("Employed", "Unemployed"))
>

```

Accordingly, the data has been filtered and only individuals below 30 that are unemployed or employed are selected. While you can check the data and see how it changes, it is good to highlight that the new dataset has only 14,639 observations instead of 40,000.

Another very common exercise is to summarize variables according to given criteria. For this case two functions become convenient. The first one is *group\_by()* which, as the name suggests, tells R that every command after that is done by groups. The second one is *summarise()*, a command that helps to collapse the data according to the criteria selected.

```

1 library(readxl)
2 library(dplyr)
3 CleanData <- read_excel("Input/Data/CleanData.xlsx")
4 CleanDataNew=CleanData%>%
5   group_by(employment)%>%
6   summarise(age=mean(age))
7 CleanDataNew
8 CleanDataNew=CleanData%>%
9   group_by(employment)%>%
10  summarise(age=min(age))
11 CleanDataNew

```

```

> library(readxl)
> library(dplyr)
> CleanData <- read_excel("Input/Data/CleanData.xlsx")
> CleanDataNew=CleanData%>%
+   group_by(employment)%>%
+   summarise(age=mean(age))
> CleanDataNew
# A tibble: 3 x 2
  employment      age
  <chr>          <dbl>
1 Employed        33.8
2 Out of labor force 35.1
3 Unemployed     33.4
> CleanDataNew=CleanData%>%
+   group_by(employment)%>%
+   summarise(age=min(age))
> CleanDataNew
# A tibble: 3 x 2
  employment      age
  <chr>          <dbl>
1 Employed         19
2 Out of labor force 19
3 Unemployed     19
> |

```

The previous example uses these two functions in two ways. The first one groups the data by employment status and then calculates the average age of each group. The second one calculates the youngest individual in each group. This command subtly explores two new elements:

1. The command *summarise* drastically changes the data structure. After it is applied, now there is only one observation per group member (3).
2. Two pipelines are concatenated to develop the exercise.

By exploring this last observation, you can see the value of the pipeline. Consider the following instructions:

1. Create a variable of age in days
2. Choose only people that are beneficiaries.
3. Select only the variables of age (in days), gender, and region.
4. Calculate by gender and region the average age (in days) of the beneficiaries.

```

1 library(readxl)
2 library(dplyr)
3 CleanData <- read_excel("Input/Data/CleanData.xlsx")
4 CleanDataNew=CleanData%>%
5   mutate(ageDays=age*365)%>%
6   filter(participant%in%c("Yes"))%>%
7   select(c("gender", "region", "ageDays"))%>%
8   group_by(region, gender)%>%
9   summarise(age=mean(ageDays))
10 CleanDataNew
11

```

10:13 (Top Level) R Script

Console Terminal Jobs

```

C:/Users/gnp12/OneDrive/Escritorio/ESCWA/ESCWA/
> library(readxl)
> library(dplyr)
> CleanData <- read_excel("Input/Data/CleanData.xlsx")
> CleanDataNew=CleanData%>%
+   mutate(ageDays=age*365)%>%
+   filter(participant%in%c("Yes"))%>%
+   select(c("gender", "region", "ageDays"))%>%
+   group_by(region, gender)%>%
+   summarise(age=mean(ageDays))
`summarise()` has grouped output by 'region'. You can over
ride using the `.groups` argument.
> CleanDataNew
# A tibble: 14 x 3
# Groups:   region [7]
  region gender   age
  <chr>  <chr>   <dbl>
1 A      Female 10980.
2 A      Male   12189.
3 B      Female 10853.
4 B      Male   12867.
5 C      Female 12539.
6 C      Male   11886.
7 D      Female 14320.
8 D      Male   10845.
9 E      Female 14614.
10 E     Male   10731.
11 F     Female 12732.
12 F     Male   14630.

```

In this case the previous example shows how to do the four instructions in a single command line.

Finally, while this table is useful for other computer processes observed in the next chapter, the format is rather unfriendly for a reader. For that purpose, many users prefer to display the table in a wide format (in contrast to the long format currently produced). For this purpose, the function `spread()` supports that requirement.

```

1 library(readxl)
2 library(dplyr)
3 CleanData <- read_excel("Input/Data/CleanData.xlsx")
4 CleanDataNew=CleanData%>%
5   mutate(ageDays=age*365)%>%
6   filter(participant%in%c("Yes"))%>%
7   select(c("gender", "region", "ageDays"))%>%
8   group_by(region, gender)%>%
9   summarise(age=mean(ageDays))%>%
10  spread(region, age)
11 CleanDataNew
12

```

```

> library(readxl)
> library(dplyr)
> CleanData <- read_excel("Input/Data/CleanData.xlsx")
> CleanDataNew=CleanData%>%
+   mutate(ageDays=age*365)%>%
+   filter(participant%in%c("Yes"))%>%
+   select(c("gender", "region", "ageDays"))%>%
+   group_by(region, gender)%>%
+   summarise(age=mean(ageDays))%>%
+   spread(region, age)
`summarise()` has grouped output by 'region'. You can over
ride using the `.groups` argument.
> CleanDataNew
# A tibble: 2 x 8
  gender      A      B      C      D      E      F
  <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 Female  10980. 10853. 12539. 14320. 14614. 12732.
2 Male   12189. 12867. 11886. 10845. 10731. 14630.
# ... with 1 more variable: G <dbl>
>

```

As it can be seen from the previous exercise, this format is friendlier for the reader. Notice two elements:

1. The function `spread()` uses two arguments, the first one is the variable you use to put in the wide format (in each column), and the second is the value that goes to the content of the table (in this case age).

- Sometimes, the R console does not display some parts of the resulting data frame. This does not mean that the column or row disappears. R internally uses it, but just for the sake of the screen size and the reader preferences it doesn't show it.

Two final notes:

First, the opposite function of `spread()` is `melt()`.

```

1 library(readxl)
2 library(dplyr)
3
4 CleanData <- read_excel("Input/Data/CleanData.xlsx")
5 CleanDataNew=CleanData%>%
6   group_by(gender, region)%>%
7   summarise(age=mean(score, na.rm=TRUE))%>%
8   spread(region, age)
9 CleanDataNew
10 CleanDataNew2=CleanDataNew%>%melt(id=c("gender"))
11 CleanDataNew2

```

```

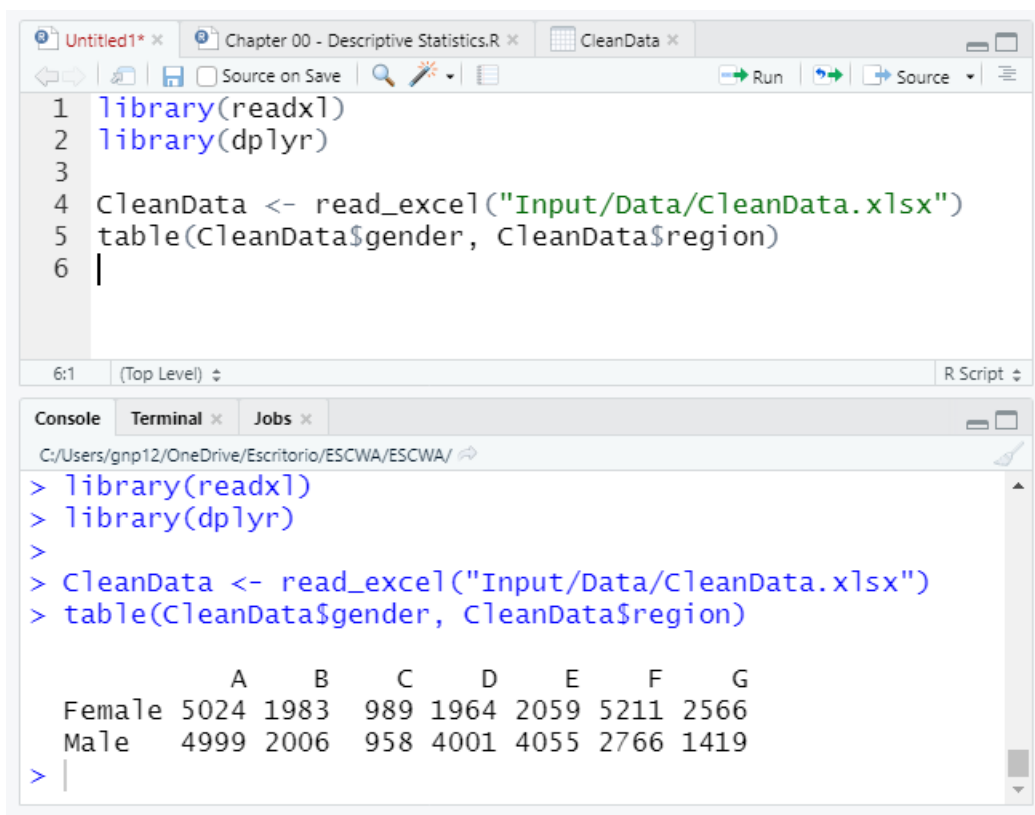
11:14 (Top Level) R Script
Console Terminal Jobs
C:/Users/gnp12/OneDrive/Escritorio/ESCWA/ESCWA/
+ group_by(gender, region)%>%
+ summarise(age=mean(score, na.rm=TRUE))%>%
+ spread(region, age)
`summarise()` has grouped output by 'gender'. You can override using the `.groups` argument.
> CleanDataNew
# A tibble: 2 x 8
# Groups:   gender [2]
  gender     A     B     C     D     E     F     G
  <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 Female  3.80  3.79  3.81  3.91  3.87  4.21  4.22
2 Male    3.79  3.77  3.84  3.91  3.94  4.20  4.18
> CleanDataNew2=CleanDataNew%>%melt(id=c("gender"))
> CleanDataNew2
  gender variable  value
1 Female         A 3.796130
2 Male          A 3.794738
3 Female         B 3.785031
4 Male          B 3.772002
5 Female         C 3.812440
6 Male          C 3.841964
7 Female         D 3.912997
8 Male          D 3.912269
9 Female         E 3.869048
10 Male          E 3.939630
11 Female         F 4.206899
12 Male          F 4.199351
13 Female         G 4.216062
14 Male          G 4.177279
> |

```

As it can be seen in the example, the function `melt()` reshape the data to long format, but during that process the name of the variables changes, so be careful for this.



Second, sometimes frequency tables need to be generated and instead of taking a long path with transformations summarizing values, there are already some predetermined functions to do this.



```
1 library(readxl)
2 library(dplyr)
3
4 CleanData <- read_excel("Input/Data/CleanData.xlsx")
5 table(CleanData$gender, CleanData$region)
6 |
```

6:1 (Top Level) R Script

Console Terminal Jobs

```
C:/Users/gnp12/OneDrive/Escritorio/ESCWA/ESCWA/
> library(readxl)
> library(dplyr)
>
> CleanData <- read_excel("Input/Data/CleanData.xlsx")
> table(CleanData$gender, CleanData$region)
```

	A	B	C	D	E	F	G
Female	5024	1983	989	1964	2059	5211	2566
Male	4999	2006	958	4001	4055	2766	1419

```
> |
```

In this example and in contrast to all the previously stated cases, using a direct function without dplyr helps to get quicker the desired table of number of observations per category.

In general, the functions introduced in this chapter are selected based on the usefulness of the codes that are relevant for the SPP-RAF, however there are many other interesting functions that can help the enthusiastic reader to improve his knowledge in managing databases. For those readers, two starting points for advanced relearning are recommended:

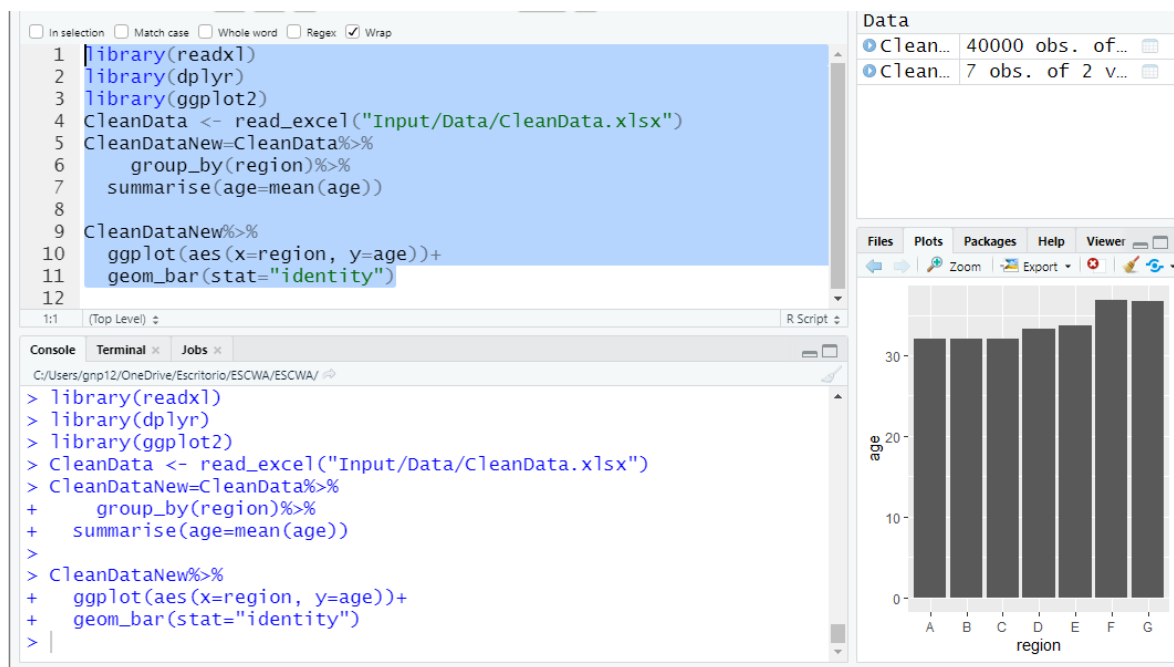
- <https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html>
- <https://datacarpentry.org/R-ecology-lesson/03-dplyr.html>

## Chapter 3: Graphs

It is frequently said that an image is worth a thousand words and this implies that the image is good. The process of representing information in graphs and charts is an art and the trainee needs to be aware that depending on the type of data and the values that you want to highlight, some graphs work better than others. Fortunately, the package “ggplot2” allows you to have a wide spectrum of graphs with many flexible options that let you standardize the reporting processes while taking care of the design. There is a vast number of designs for graphs and each day new graphs appear. Hence, rather than pretending to be exhaustive, this section explains the basic elements of the ggplot2 package and then guides the reader on how to work with the graphs by themselves. Due to the flexibility, different programmes teach this package in different ways, but the way presented here is expected to produce easier interpretation of the codes and facilitate the process for the trainees who are new to R.

### Basics

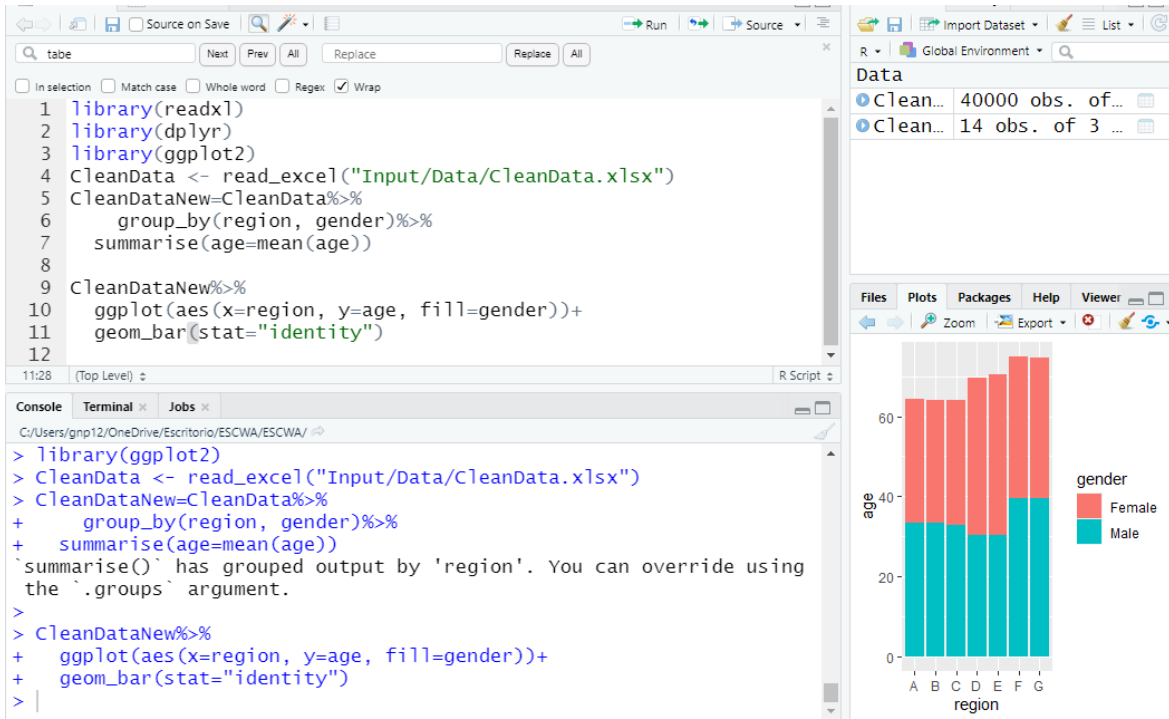
To understand the basic elements of a ggplot2, let's start with a bar plot that displays the average age of individuals based on their region.



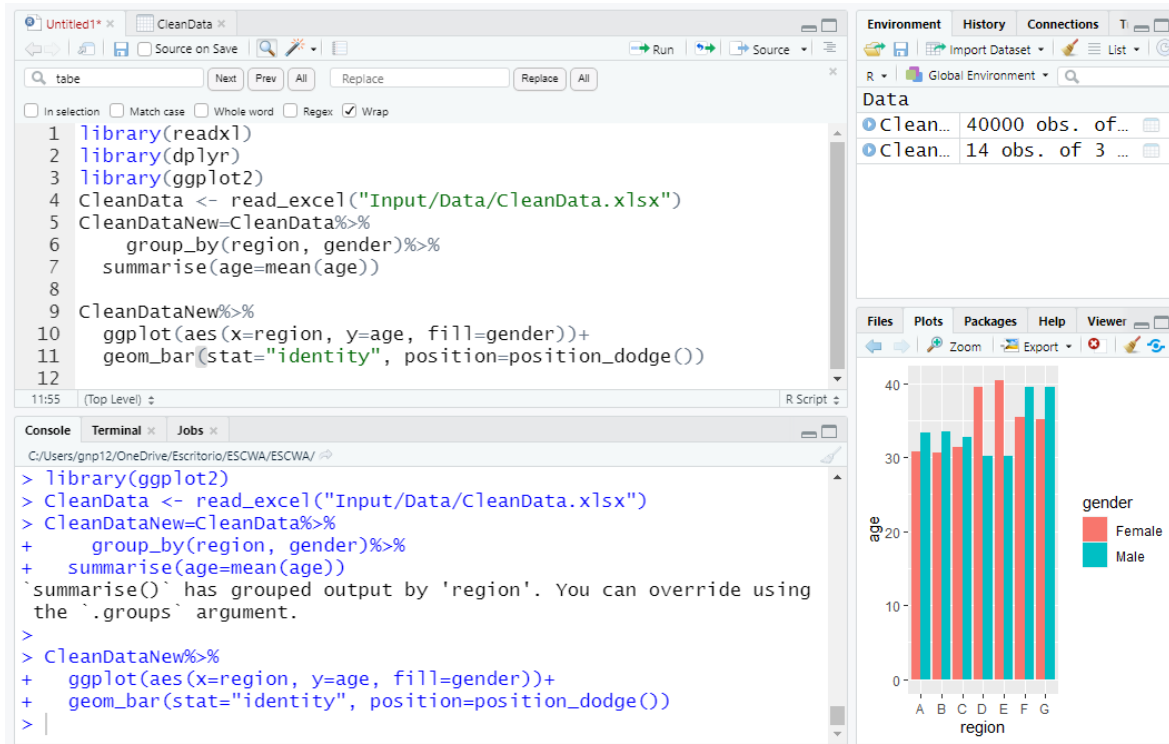
To create the graph, three steps are needed:

1. A pipeline that links the data that will be used for the plot with the `ggplot()` function.
2. Inside the `ggplot()` argument there is another function called `aes()` where the user specifies the names of the variables in the x and y axes.
3. The command line ends with a “+” sign that works as a pipeline to connect with the next instruction of the graph; in this case the instruction is the one that determines the form of the graph, which in this case is a bar plot, thus the function is `geom_bar()`.

To increase the level of challenge, suppose that the task is to produce a bar plot that displays the average age of individuals based on their region and gender.



There is only one change between this graph and the previous one. Now, there is an additional parameter in `aes()` associated to this new variable. Different graphs have different attributes, as it will be shown in the following examples, but the package `ggplot2` allows to do this in a user-friendly way that assigns all these characteristics to the `aes()` command so that the user knows where to place them. Having said that, the current graph is not good. It does not make sense to pile up the age of women and the age of men, and this makes the graph confusing and uninformative.



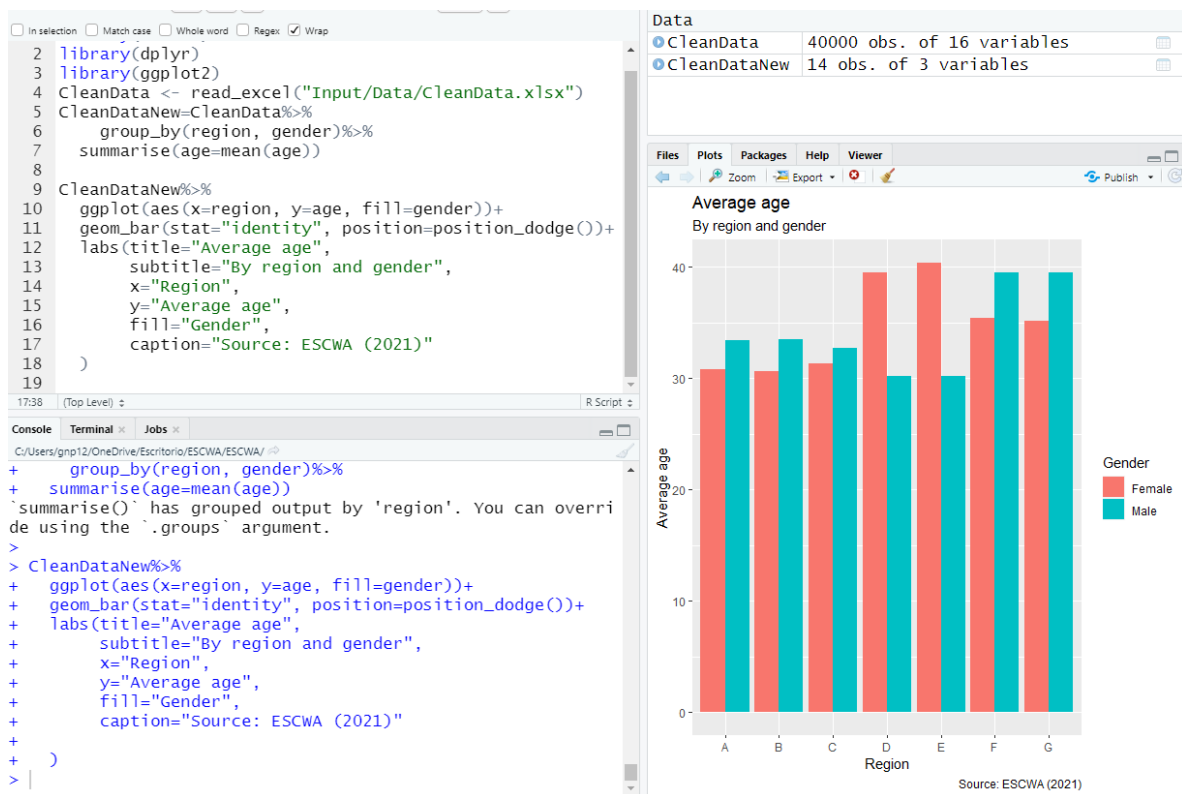
Now, by specifying the position as “position\_dodge()” inside the function, the graph now looks better and the values are easily interpreted, showing that region D and E tend to have older female population, while region F and G have older male population. In a similar way, the gap in regions A and B is higher for men, but the difference is not as dramatic.

## Formatting

Once the initial structure of a graph is designed, it is important to format it so that it looks appealing and contains the relevant information. Some of the tips explained here are based on the World Bank (2016) editorial guidelines that allow you(after few modifications) to generate professional graphs that can be easily incorporated into a report.

## Labels

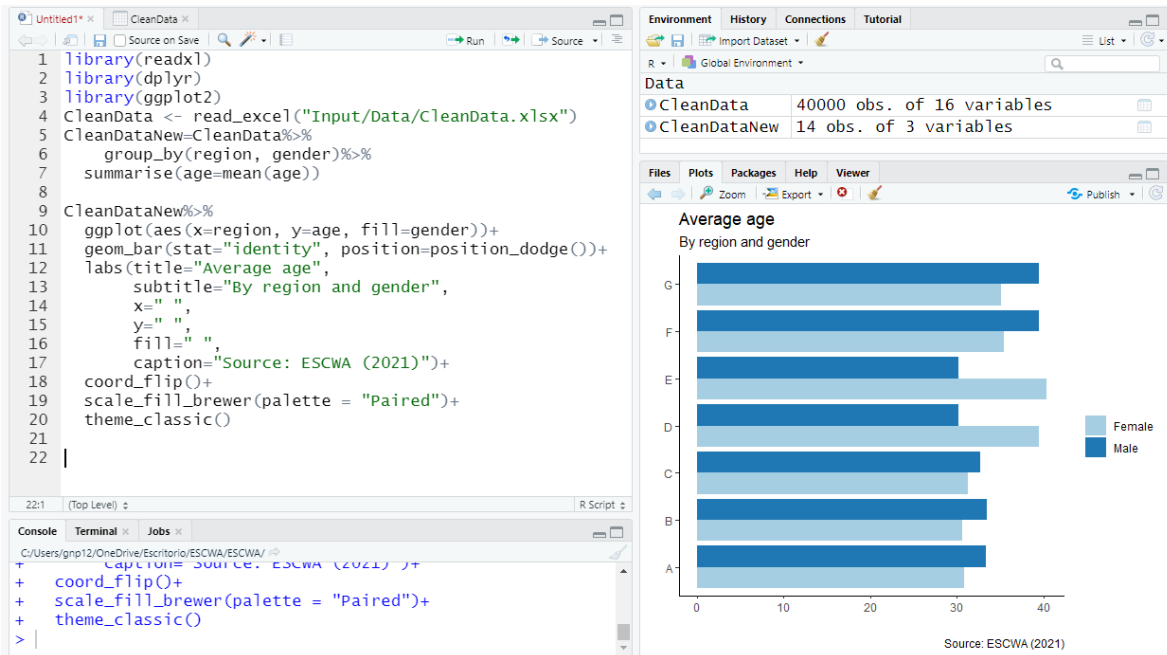
The first point to begin with are the labels that appear in the graph. In this case, ggplot2 provides the function *labs()* which allows a great control of these parameters.



Actually, some of these values are redundant. For example, given that age is already in the title, it is unnecessary to place it in the y-axis. Similarly, if it is clear by context, that A to G are regions, then the x axis is also redundant. However, this is presented completely for the sake of learning how to incorporate these values as part of the *labs()* function. These values are removed on the following graphs so that the design becomes more stylish.

## Position, gridlines, and colours

While these elements depend on the person presenting the report, it is recommended to flip the coordinates if it helps to visualize the labels better, remove internal gridlines, and use sober colours that provide a nice gradient used consistently along the full report.

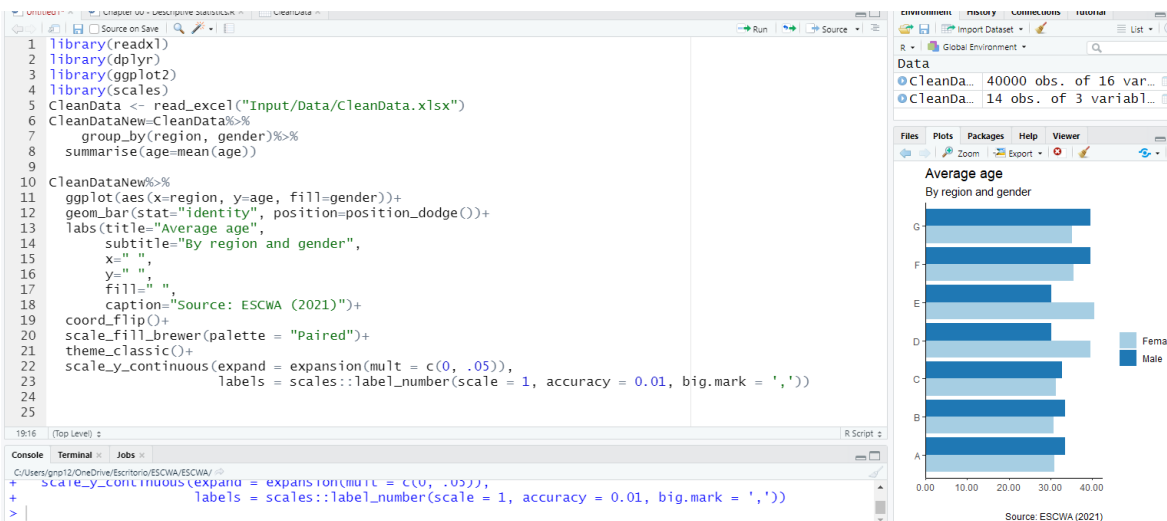


In this case, to achieve this result, three commands were added to the command line:

1. `coord_flip()`: This is in charge of changing the axis.
2. `Scale_fill_brewer()`: This changes the colour palette. In this case, the palette use is "Paired" but this is not the only one. For avid readers, please go to <https://www.r-graph-gallery.com/38-r-colorbrewers-palettes.html> to find dozens of other designs and even how to create your own palettes.
3. `theme_classoc()`: This removes the gridlines and keep the white background.

### Axes and spaces

Notice in the previous exercise that there is a gap between 0 and the axis that damages the aesthetic feature of the graph. In a similar way, consider the case that you would also want to add decimals (2) to the y axis (that now is the x axis because of the flip).



This new example changes two elements in the code. First, notice that a new library is added. The package “scales” helps to work on the plot axis. Second, a new command is added to the graph (lines 22 and 23). This new function `scale_y_continuous()` has two arguments. The first one is related to “expand” which is the one that removes the 0 gap from the graph. The second one is related to “labels” defining the accuracy and scale of the axis. In this case as the accuracy is 0.01, the axis now has 2 decimals. As in the previous case, the package scales bring a wide variety of options and avid readers are encouraged to explore more some links such as <https://scales.r-lib.org/>. On that same note, notice the similarity in lines 20 and 22. The graphical function of ggplot is set to keep these patterns in a way that is easy for the users to remember how to write the command lines.

### Other formatting criteria

While many other formatting criteria will be added to the illustration codes seen in the next chapters, it is recommended to practice turning them on and off to understand what effects they have over the graphs . Some detailed guidance on these topics can be seen in references such as <https://ggplot2.tidyverse.org/reference/ggtheme.html> and <http://www.sthda.com/english/wiki/ggplot2-themes-and-background-colors-the-3-elements>.

### Clustering graphs

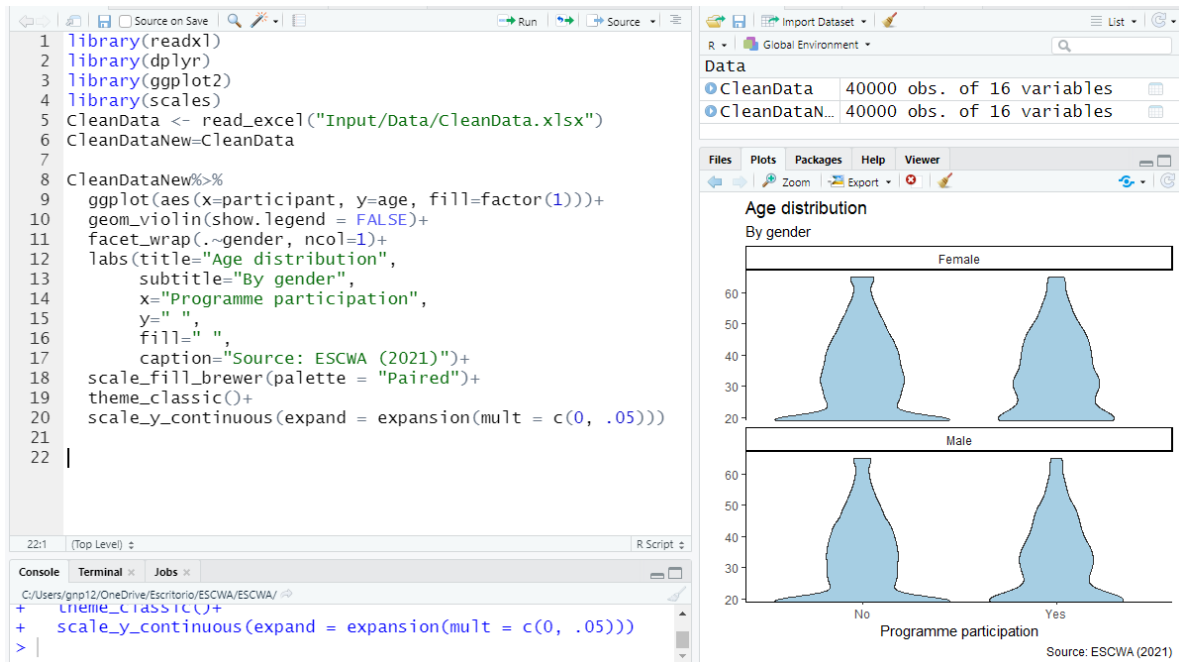
Sometimes the amount of information is excessive and there is a need to create sub graphs to understand a topic. For example, suppose that the task is to produce a bar plot that displays the average age of individuals based on their region, gender, and whether they are beneficiaries.



In this new code the function `facet_wrap()` allows the creation of subgraphs within the same frame. Here, the argument `ncol` is used to define in how many columns the subgraphs are arranged. Hence, notice that, even with these few commands, it is already possible to have significant insights of the data. At the moment, it is possible to see regions with age gaps between males and females, and more interesting, for A and B, there is a small gap for non-beneficiaries, but for beneficiaries the gap increases meaning that the programme is tending to choose older men than women in this region. With these insights, the next tasks allow you to understand where these differences come from. However, you still need to know more graphical and code related elements before you arrive to that stage.

## Graph types

Choosing the type of graph is an art and, depending on the message that we want to produce, it is possible to prefer one graph over others. For example, consider a researcher that is interested in learning not only the mean values, but the whole age distribution.

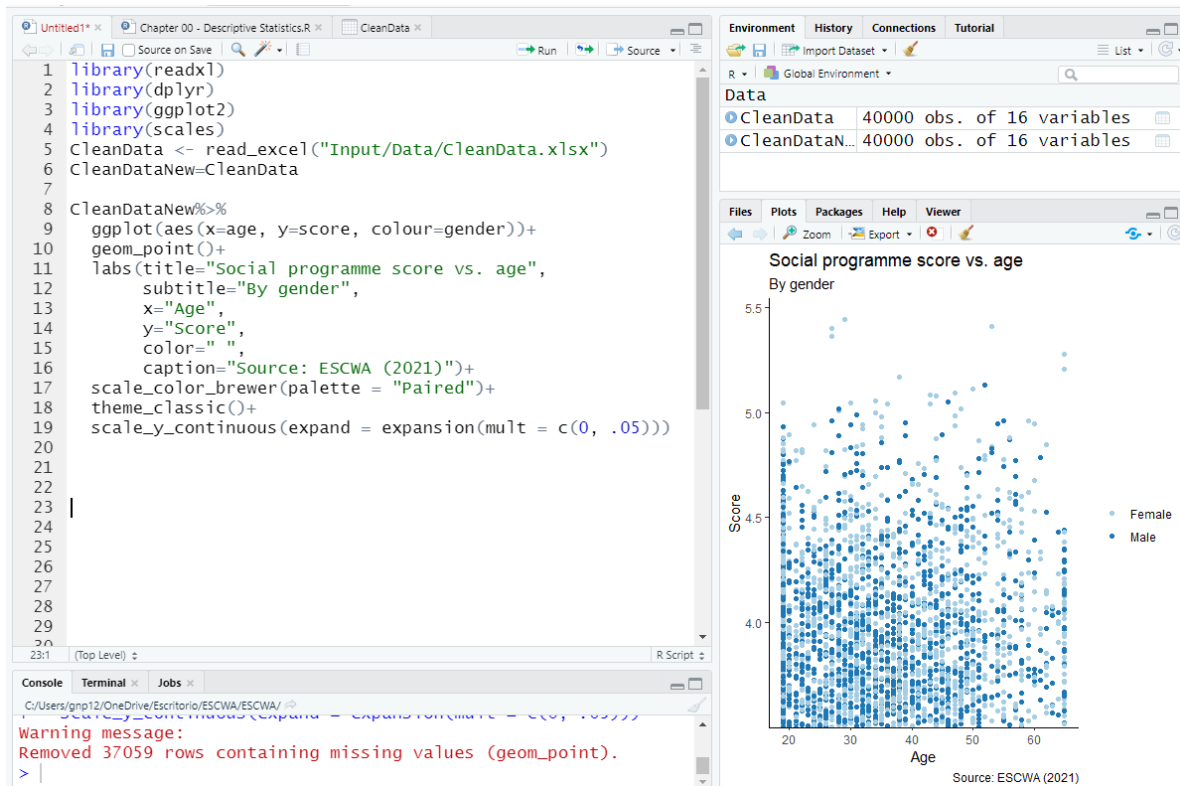


Three elements are important to highlight in the code:

1. Line 10 changes and instead of specifying a bar plot, it now specifies a violin plot.
2. In contrast to the bar plot, the violin plot can be done directly over the full data set and does not require any pre-processing of it.
3. A factor has been added, but is not used, and it has been removed by the legend. This is a trick to use when there is no fill to make sure that the graphs have a determined colour palette and are not grey as the first graphs in the chapter.

In contrast to the previous bar graphs a violin plot helps you to notice that younger individuals are less represented in the beneficiary group than in the non-beneficiary group. This gives you a hint that the older individuals are more likely to be beneficiaries of a programme.

Once you have this hypothesis you can create another graph that helps you to support this claim further. In this case, let's create a scatter plot that has the score of the beneficiaries against their age.

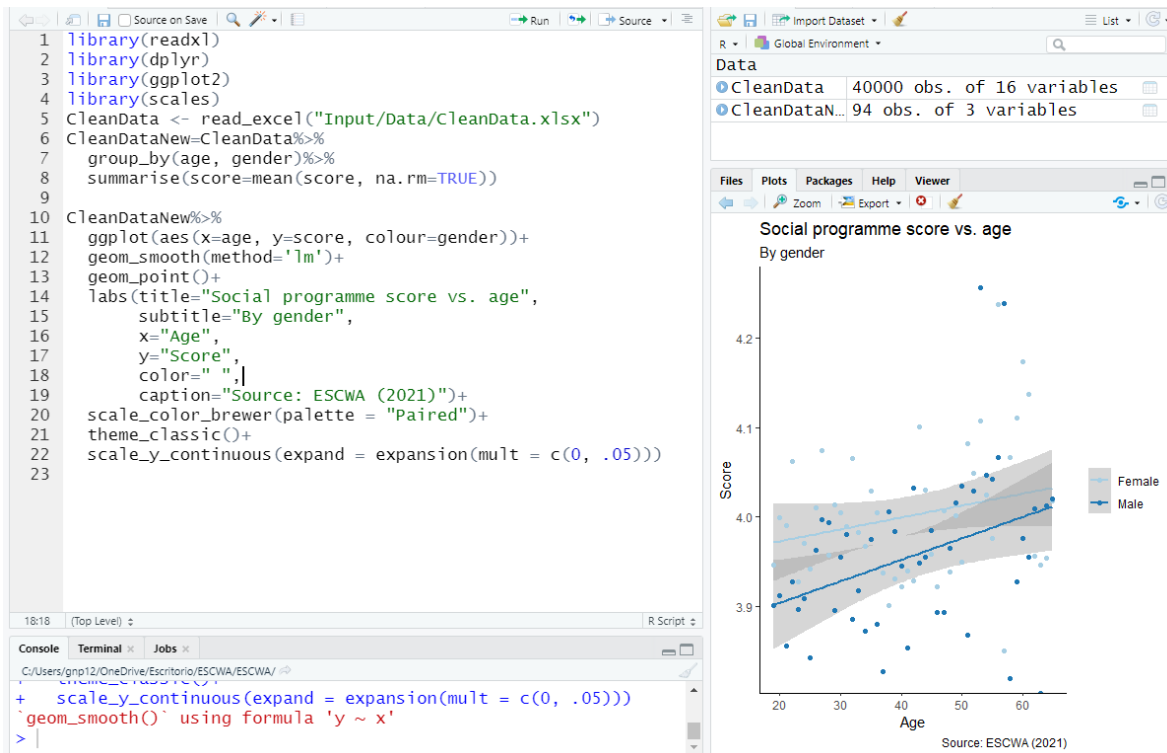


Several elements to highlight in this example:

1. Line 10 changes to have the new geometry of the graph.
2. While the previous ones have “fill” as an aesthetic variable, points have “colour”, and this has three implications, the first one is in line 9, the second one is in line 15, and the third one is in line 17.
3. Sometimes graphs are not useful, for example, this graph can look nice, but it is very difficult to conclude anything from it.
4. There is a red warning in the console. Warnings are not errors, but they are ways that the software use to indicate that there is something strange happening. In this case, as you use all the dataset, and individuals who are not beneficiaries have no scores, then there are a lot of pairs that cannot be painted. So accordingly, there is no problem. However, sometimes the warnings have useful information.

In cases like this one, maybe it is better to change the graphical strategy, so instead of a scatter plot the next example displays a line plot constructed along the mean scores at each age.





Again, several elements to highlight in this example:

1. While it looks like a single graph, in reality there are two overlaying graphs, there are points and linear regression lines, reflected along lines 11 and 12.
2. In this case, you need to prepare the data before the graph. However, you find a “na.rm=TRUE” in the summary. As it is discussed in the previous case, there are NA observations in the score variable and if these are included in the average, the whole result is NA. Therefore, by adding this line, which stands for remove NA, R understands that the procedure is done only over values that are not NA.
3. While there is still noise in the data, the graph shows a clear tendency that highlights that older individuals tend to have higher scores, meaning that they have more needs. However, the difference in the score across the ages is higher for males than females.

In this way, you notice how understanding the problem and changing the graph design can help you to get better insights about the programme.

During the previous exercises, you have been exposed to different types of graphs, where the common patterns among the codes have been highlighted. Currently there are multiple portals with amazing ideas on how to display information. One recommendation for curious minds is <https://www.r-graph-gallery.com/ggplot2-package.html> where you find access to multiple designs using these same code patterns. Naturally, as it is mentioned at the beginning, different webpage develop slightly different codes, but using the tips highlighted in this section, the trainee can quickly identify the places were codes need to be modified and successfully start designing new graphs.

## Chapter 4: Excel synchronization

Excel is a very powerful tool. However, it is very difficult to systematize exercises on it. For example, consider a case where you need to create a hundred graphs without title, and out of a sudden your supervisor requires you to add a generic title to all these graphs. Then, it takes a large amount of time (approximately 30 seconds per graph if you already know the titles; 50 minutes non-stop). However, what if your supervisor, after seeing this, decides that this is not a good idea and it is better to remove the titles. Then just by doing this task, which has no value added, you lose at least 100 minutes of your time. This chapter and the following chapter work on the coordination of Excel and R to make sure that this process can be automatized, similar to the previous one and those changes can be made only by modifying few line codes.

To start the chapter, one new package is introduced: “openxlsx”. This package makes working with Excel (pasting tables and graphs) way faster.

### Main commands

In contrast to other sections, the current section presents the code, and once its parts have been explained, it presents the results.

```
1 library(readxl)
2 library(dplyr)
3 library(ggplot2)
4 library(scales)
5 library(openxlsx)
6
7 wb = openxlsx::createWorkbook(creator = 'ESCWA')
8 newSheet = addWorksheet(wb, sheetName = "Age vs. Score")
9
10 CleanData <- read_excel("Input/Data/CleanData.xlsx")
11 CleanDataNew=CleanData%>%
12   group_by(age, gender)%>%
13   summarise(score=mean(score, na.rm=TRUE))
14
15 newPlot=CleanDataNew%>%
16   ggplot(aes(x=age, y=score, colour=gender))+
17   geom_smooth(method='lm')+
18   geom_point()+
19   labs(title="Social programme score vs. age",
20        subtitle="By gender",
21        x="Age",
22        y="Score",
23        color=" ",
24        caption="Source: ESCWA (2021)")+
25   scale_color_brewer(palette = "Paired")+
26   theme_classic()+
27   scale_y_continuous(expand = expansion(mult = c(0, .05)))
28
29 fileName="Output/Part1 Examples/Scatterplot age vs score.png"
30 cleanTable=CleanDataNew%>%spread(gender,score)
31 ggsave(fileName, plot = newPlot, width = 6 , height = 4 , scale = 1)
32 insertImage(wb, file = fileName, sheet = newSheet, startRow = 1, startCol = 1, width = 6, height = 4)
33 writeDataTable(wb, sheet = newSheet, x = cleanTable, startRow = 1, startCol = 10)
34
35 saveWorkbook(wb,
36              file = "Output/Part1 Examples/Excel report.xlsx",
37              overwrite = TRUE)
```

The exercise is based on the last graph plotted in chapter 3. In this case a new library is added in line 5.

Then, in line 7 and 8 two new commands appear:

1. `Openxlsx::createWorkbook()`: This function is requesting R to create an Excel file that, in this case will appear as if it is created by “ESCWA”. At this stage you are not able to see the file, but in the last command line, R takes all the instructions produced in the way and create that final excel with all the desired traits.

2. `addWorksheet()`: This function informs the excel file that you begin to design (that's why its first argument is the excel file -in this case wb- appears there), the other argument is the name of the spreadsheet, but be careful, Excel has some limitations on these names and if you don't follow them, you will get an error.

After those commands are introduced, you find an exercise to create the graphs. However, there is a subtle change in the code. In line 15 all the graphs are assigned to a variable. In this case, it is not be displayed in the interface, but it is stored in the disk for other purposes. If you want to still see it, you can create a new line only with the name of the variable.

Line 29 presents a variable named filename that tells R where to print the graph and in what format. This is printed in png format.

Line 30 adjusts the dataset used to produce the graph into a nice table by using a `spread()` function previously discussed.

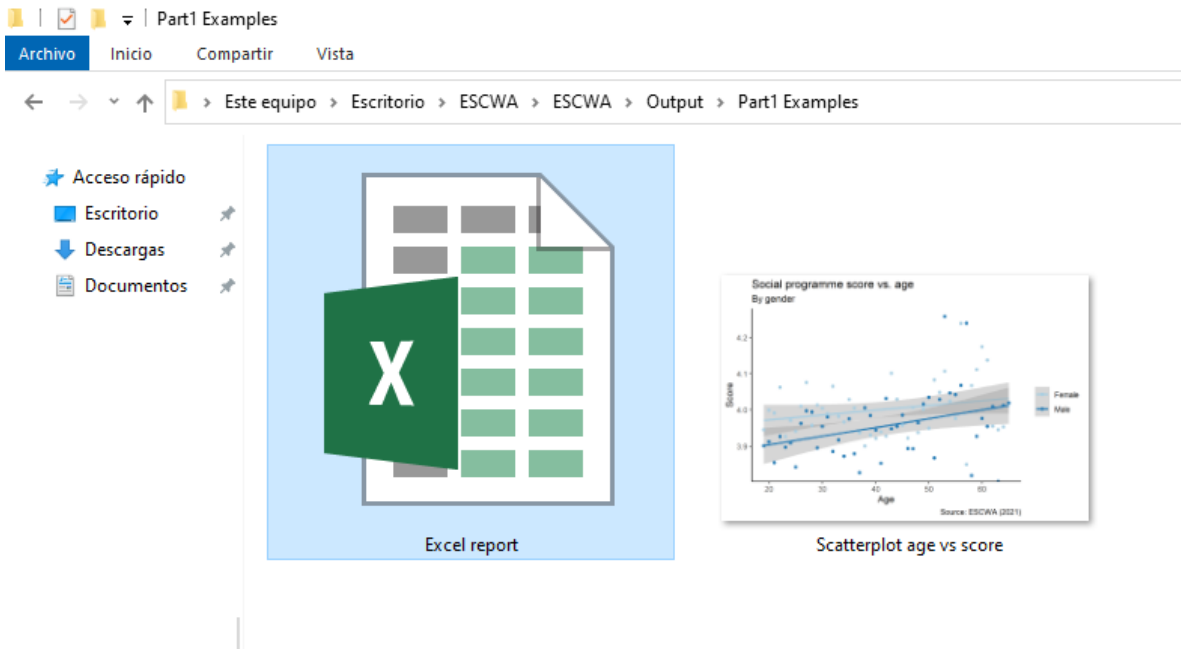
Line 31 saves the graphs addressed by filename. The function doing this is `ggsave()`. Along its arguments you find plot, that is where you explicitly add the graph, and details such as width, height, and scale that are related to the proportion of the printed graph.

Line 32 tells R to add this graph into the Excel file. The function doing this is `insertImage()`. In it, you find several items that need to be specified. First, you need to insert the Excel file you are designing, then tell the file about the image (identified with `fileName`), followed by the sheet you want to paste , and then specify the rows, columns where you want to paste it, and the dimensions of the graph.

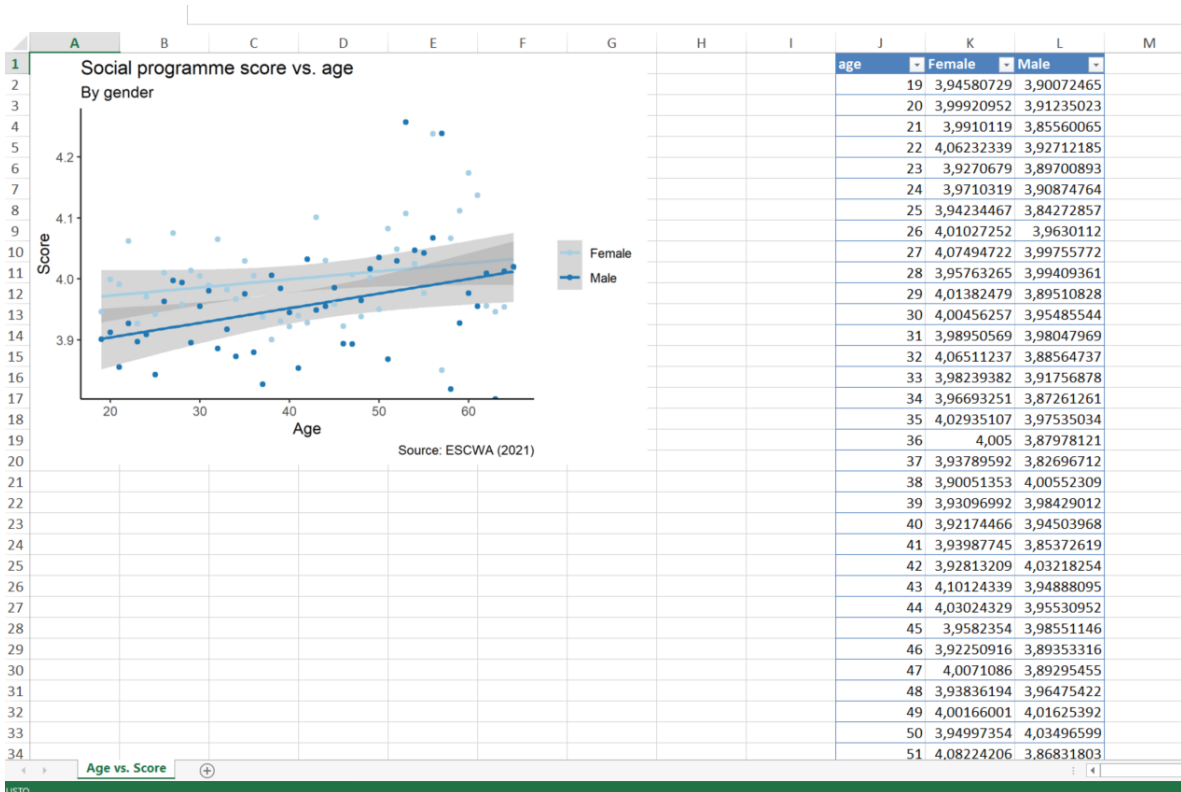
Line 33 tells R to add the table into the Excel file. The function that is doing this is `writeDataTable ()`. Similar to the previous case, you need to specify the Excel file you are designing, followed by the sheet you want to paste . This is followed by the variable where the table is saved (the one in line 30), and then specify the rows and columns where you want to paste it.

Finally, line 35 to 37 allow R, through the function `saveWorkbook()`, to create the Excel file in the path that you determined. It also allows you to overwrite it if there are previous files with the same name.

As a result of this process you will find two new files in your computer. The first file is the image that is created, that is ready to be pasted into any report you want.



The second is the Excel report that you can access to see the results of the work done.



Relevant things to highlight:

1. Notice that the starting cells of the graph and table matches those that are assigned by the code.
2. Notice the name of the spreadsheet that matches the one defined by the code.

- If this data needs to go into a report, now it is fairly easy to copy and paste it from the Excel file.

### Dictionary

The previous section shows the main elements needed to connect Excel with R. This section uses all the knowledge accumulated until now to create an automatized functional report system where Excel and R interact quickly. However, for this exercise a new Excel file (which is called a dictionary) is needed. For this example, you find it in the input folder, in the dictionary folder, under the name: "Dictionary Reduce.xlsx".

In this Excel file you find two sheets with two tables. The first one has all the elements of a graph:

Code	Title	Subtitle	X Axis	Y Axis	Caption	Title_AS	Subtitle_AS	X Axis_AS	Y Axis_AS	Caption_AS	Color Pallet
0_1	Social programmes benefici	By gender	Age	Score	Source: ESCWA (2021)	المصدر: الإسكوا (2021)	حسب الجنس المستفيدين من	سن	نتيجة	Paired	
XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX

The second one has all the element of a categorical variable:

Code	Category	EN	AR
1	Male	Men	ذكور
2	Female	Women	إناث

At this stage it looks confusing to understand why this work is being done, but while we advance in the code, its usefulness becomes evident.

Hence, the first task is to upload these files into R.

```

1 library(readxl)
2 library(dplyr)
3 library(ggplot2)
4 library(scales)
5 library(openxlsx)
6
7 fileDictionary="Input/Dictionary/Dictionary Reduced.xlsx"
8 dictionaryLabels=read_excel(fileDictionary, sheet="labels")
9
10 genderCategories=read_excel(fileDictionary, sheet="categories",
11                             range="A1:D3")
12
13 dictionaryLabels
14 genderCategories

```

14:17 (Top Level) R Script

```

Console Terminal Jobs
C:/Users/gnp12/OneDrive/Escritorio/ESCWA/ESCWA/
> library(dplyr)
> library(ggplot2)
> library(scales)
> library(openxlsx)
>
> fileDictionary="Input/Dictionary/Dictionary Reduced.xlsx"
> dictionaryLabels=read_excel(fileDictionary, sheet="labels")
>
> genderCategories=read_excel(fileDictionary, sheet="categories",
+                             range="A1:D3")
>
> dictionaryLabels
# A tibble: 2 x 12
  Code Title Subtitle `X Axis` `Y Axis` Caption Title_AS Subtitle_AS `X Axis_AS` `Y Axis_AS` Caption_AS
  <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
1 0_1 Social~ By gender Age Score "Sourc~ <U+0627><U+0644><U+0645><U+0633><U+062A><U+0641><U+064A><U
+062F>~<U+062D><U+0633><U+0628> <U+0627><U+0644><U+062C><U+0646><U+0633> <U+0633><U+0646> <U+0646>
<U+062A><U+064A><U+062C><U+0629> " <U+0627><U+0644><U+0645><U+0635><U+062F><U+0631>:\~
2 XXX XXX XXX XXX XXX "XXX" XXX XXX XXX XXX "XXX"
# ... with 1 more variable: Color Pallete <chr>
> genderCategories
# A tibble: 2 x 4
  Code Category EN AR
  <dbl> <chr> <chr> <chr>
1 1 Male Men <U+0630><U+0643><U+0648><U+0631>
2 2 Female Women <U+0625><U+0646><U+0627><U+062B>
>

```

Notice that in this code, the function that was used in the past `read_excel()` has been expanded such as in one case we specified the sheet, and in the other we specified the sheet and the range. Sometimes R is “smart” and can identify the table you want, but when there are many tables, R gets confused and therefore, it is better to specify it.

Now notice two other elements:

1. The first table has an odd row of XXXXXXXXX in the last line. This is a practical suggestion that helps you to avoid mistakes that the programme generates when tables have columns with many missing values.
2. The Arabic script does not appear, and instead shows strange codes. These codes are ways in which R displays the characters, but once the codes are done, the output will have the right content.

The second step is to create a variable that helps with the identification of the language of the graphs and tables. Once this is done, all variables with categories need to be adjusted via factors so that the labels are in the right language.

```

1 library(readxl)
2 library(dplyr)
3 library(ggplot2)
4 library(scales)
5 library(openxlsx)
6
7 fileDictionary="Input/Dictionary/Dictionary Reduced.xlsx"
8 dictionaryLabels=read_excel(fileDictionary, sheet="labels")
9
10 genderCategories=read_excel(fileDictionary, sheet="categories",
11                             range="A1:D3")
12
13 #Language= 1 for English 0 for Arabic
14 language=1
15
16 CleanData <- read_excel("Input/Data/CleanData.xlsx")
17 CleanDataNew=CleanData%>%
18   group_by(age, gender)%>%
19   summarise(score=mean(score, na.rm=TRUE))%>%
20   mutate(gender=factor(gender, levels=as.matrix(genderCategories)[,2],
21                       labels= as.matrix(genderCategories)[,4-language]))
22 CleanDataNew
23
24

```

23:1 (Top Level) R Script

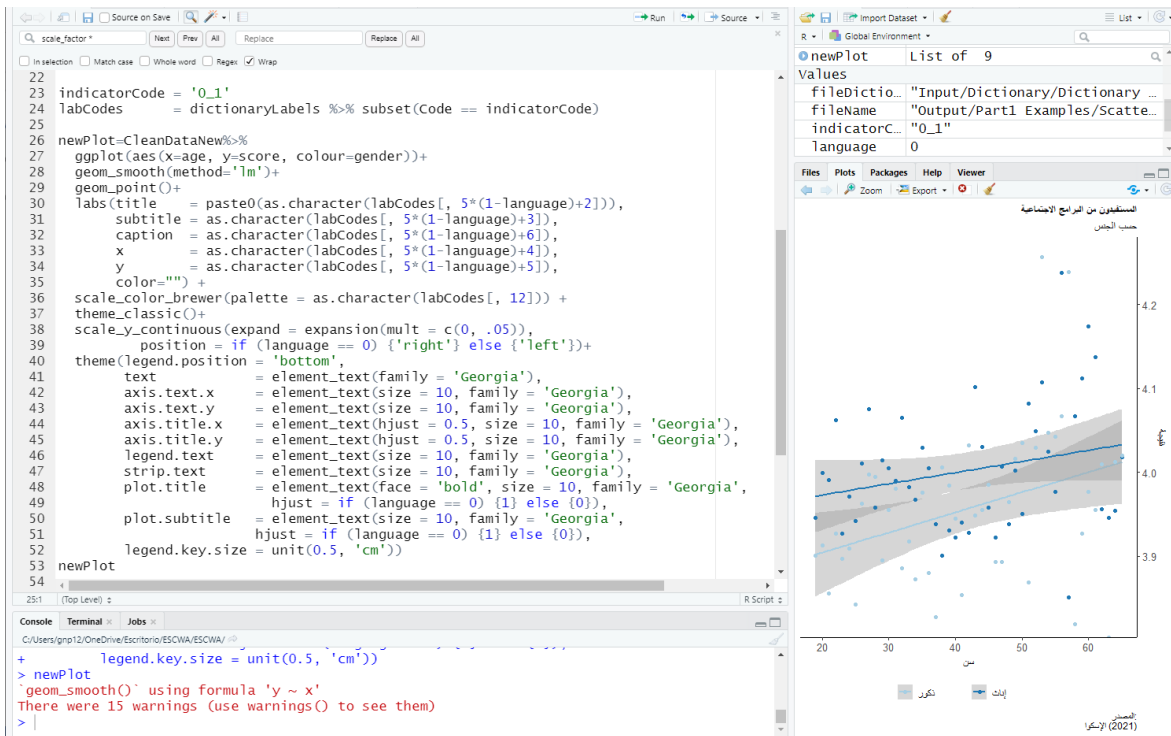
```

Console Terminal Jobs
C:/Users/gnp12/OneDrive/Escritorio/ESCWA/ESCWA/
> CleanDataNew
# A tibble: 94 x 3
# Groups:   age [47]
  age gender score
  <dbl> <fct> <dbl>
1    19 Women  3.95
2    19 Men    3.90
3    20 Women  4.00
4    20 Men    3.91
5    21 Women  3.99
6    21 Men    3.86
7    22 Women  4.06
8    22 Men    3.93
9    23 Women  3.93
10   23 Men    3.90
# ... with 84 more rows
> |

```

You can notice in this example that the data is uploaded and processed, and at the last stage, the variable gender is turned into a factor. In this case, `as.matrix(genderCategories)[,2]` tells R that the levels are those written in the second column of the excel table associated with categories. Then, `as.matrix(genderCategories)[,4-language]` tells R that if `language = 1` then  $4 - language = 3$ . Thus, take column 3, which has the new English labels. Therefore, now the data says Men and Women instead of Male and Female. In contrast, if `language = 0` then  $4 - language = 4$ . Thus, take column 4, which has the new Arabic labels. While easy, this trick is extremely powerful as it allows to rapidly change categories into different languages. Moreover, if the translation goes wrong or there is a need to change it to a synonym, you can quickly change the label in the Excel dictionary, run the code in R and then all the results with this variable are automatically adjusted. On that note, notice that column 2 should never be touched because it has the values with the exact spelling that they have in the original dataset.

Using the same idea done with the factor, the excel of the graphs is linked to the graph parameters.



Please notice that for visualization purposes, this code starts right after the code of the previous example finishes. There are some small differences between this code and the one used in the previous section. First, line 23 and 24 appear and their role is to assign the graph a code (in this case “0\_1”) and then find the code in the dictionary to know what graph labels should be used. This is done under the function `subset()`.

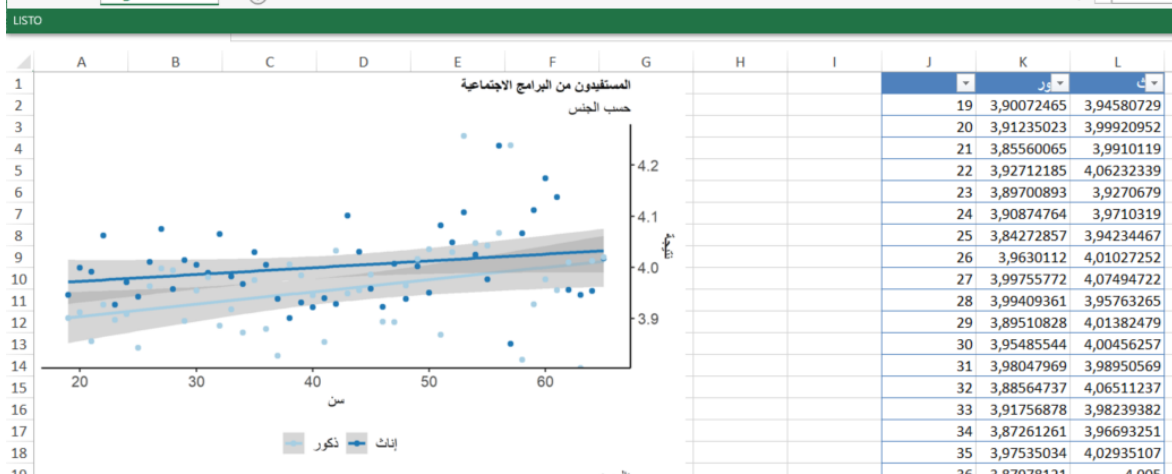
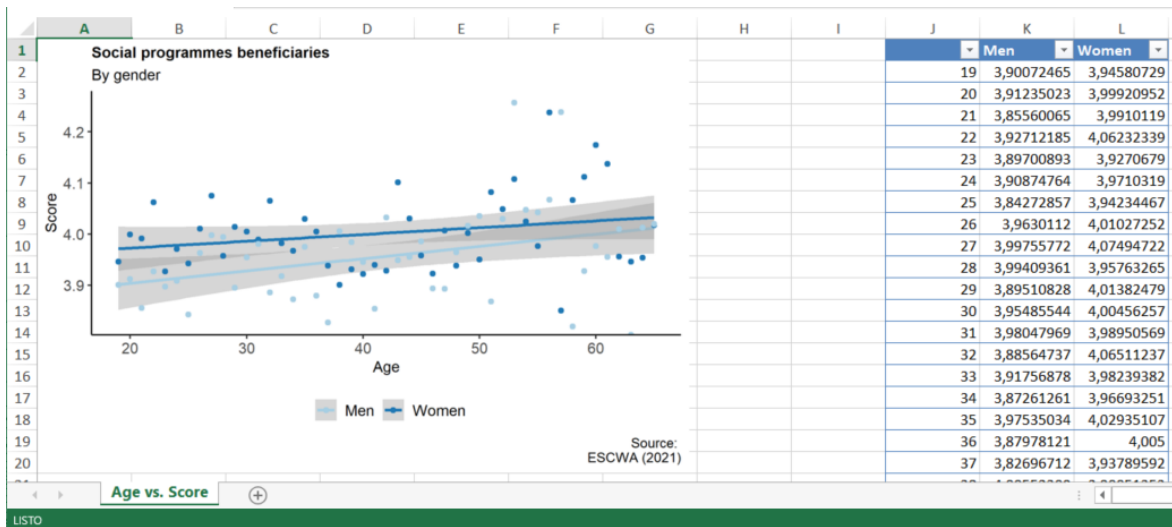
Then, regarding the graph, you find in the lines associated with the labels (31-35) and with the colours a similar trick as the one used for the factor that uses the right version of the Excel table depending on the language variable (for this exercise the variable is set to 0 for the Arabic version).

There is also an addition to the function `scale_y_continuous()`. The purpose of this addition is to acknowledge that the position of the y-axis changes in Arabic. As you see, it has an `if else` statement similar to the ones in the first chapter that changes the axis accordingly to the language that the graph uses.

Finally, there is a large set of instructions related to `theme()` that ensures a better format of the graph. As part of the learning process, it is recommended that the reader plays with these lines to understand better how they are affecting the graph.

Having defined with the previous part, avid readers will be keen on adding the code that is needed to produce the Excel files. While this is not done here, as it is a repetition of the previous section, the two results are displayed so that the trainee can have a clear idea of the possibilities of this technique.





In conclusion of this chapter, notice that by producing an organized dictionary, R can automate the generation of reports in multiple languages. Moreover, as promised in the introduction of this chapter, if there are changes that need to be done in a large number of graphs, then the user only needs to change the dictionaries and after running the codes all these changes will be automatically incorporated into the report. This saves time and helps in avoiding mistakes, since once the correction of the dictionary is done, then there is no need to check graph by graph and table by table to identify where changes need to be done.

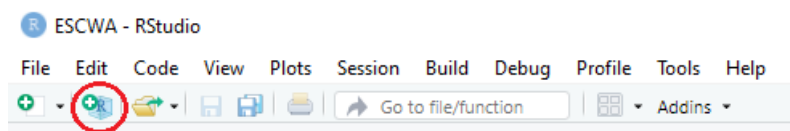
## Chapter 5: Automatizing of large processes

The previous chapter went into details on how to connect Excel and R to produce rapid reports. This chapter implements these techniques massively to start in the generation of the SPP-RAF report. Keen readers may wonder why this chapter is not in part 2. However, the answer is that part 2 is left for statistical techniques, and this chapter is doing the introduction of that, i.e. producing the relevant descriptive statistics. Accordingly, this chapter, once placed into a report can be called Chapter 0 or an introductory chapter where the data set is described via graphs and tables and creates the environment for the analysis that comes afterwards. For this chapter as well as for the coming chapters the Excel files that will be used are RawData.xlsx (present in Inputs/Data) and Dictionary.xlsx (present in Inputs /Dictionary).

### The project and the master file

A useful recommendation before starting any project is to have an organized arrangement of folders.

The suggested arrangement is the one you find in the teaching aid material, where you find separated folders for inputs, codes, and outputs. Once you have everything organized, you can create an R project and save it in the root folder (in the teaching aid, this project can be found under the name ESCWA). Still for the general case, you can save a project using the icon highlighted in the illustration.



A project, for practical purpose, is like a folder where you can put many codes together, and the codes know, that they are all interacting over the same computer files. In general, you can work with all scripts, as we have done until now. But, when there are big projects, a good practice is to split the code into parts that are easy to review separately (as chapters). This facilitates debugging processes and the codes look cleaner. In that context the project is a powerful structure that allows the integration of these separate files via a well synchronized master file. Hence by calling the master file through the project, it gathers information about the folder structure and improves the organization of results, as it appears in this section. The exact code of the master file is in the folder code of the teaching aid, but in this section, you go through its components. In contrast to other sections, several functions here are covered superficially since they do not add much value to the reader's general comprehension and their exact functionalities can be found on the internet.

```
1 rm(list = ls())
2 graphics.off()
3 gc()
4 startTime = Sys.time()
```

The first four lines of the master file guarantee that the software is clean and has no graphs or variables saved from previous exercises. This helps to make the calculations go faster and avoid errors that can come from previously stored data. The last line is a time controller. It tells the computer when the code begins to run. Later, at the end of the master file, there is a complementary line that tells the computer when the code stops running and tells the user about the duration of the process. This is highly useful as the user can estimate running times and plan its work accordingly.

```

6 # 1| Preparation -----
7 # 1.1| Libraries -----
8 myPackages = c('readxl','gridExtra','rpart','partykit','ggdendro','ggparty',
9               'factoextra','ppcor','glmnet','caret','gbm','Metrics',
10              'cobalt','gtools','fastDummies','openxlsx','extrafont',
11              'here','tidyverse','reshape2','StatMatch')
12 notInstalled = myPackages[!(myPackages %in% rownames(installed.packages()))]
13 if(length(notInstalled)) {
14   install.packages(notInstalled)
15 }
16
17 invisible(sapply(myPackages, library, character.only = TRUE, quietly = TRUE))
18 loadfonts(device = 'win', quiet = T)
19 options(scipen = 999) # Disable scientific notation.

```

It is a good practice that codes are neatly written and commented. This helps other users to quickly understand what has been written in the script. For this particular section, the code asks the computer if the packages that are described between lines 8 and 11 have already been installed and if they haven't, then the code installs them. Finally, it makes sure to upload all these libraries so that all the other codes can run smoothly, and the user does not need to worry about checking if the library has been loaded or not.

```

21 # 1.2| Initial values -----
22 # 1: English.
23 # 0: Arabic
24 language = 0
25
26 userLocation = enc2native(here()) # Replace by your own path.
27 if(!file.exists("Output")){
28   dir.create("Output")
29 }
30 if(!file.exists("Output/Chapter 00")){
31   dir.create("Output/Chapter 00")
32 }
33 if(!file.exists("Output/Chapter 01")){
34   dir.create("Output/Chapter 01")
35 }
36 if(!file.exists("Output/Chapter 02")){
37   dir.create("Output/Chapter 02")
38 }
39 if(!file.exists("Output/Chapter 03")){
40   dir.create("Output/Chapter 03")
41 }
42
43 scriptLocation = paste0(userLocation, '/Code/')
44 inputLocation = paste0(userLocation, '/Input/')
45

```

This next section initializes the output folders and defines the language of the full report. Notice the function on line 26. As you recall from previous exercises, when you want to call an excel file or another file you need to use excessively large paths. However with the function `here()` R looks for the location of the project and defines all the folders with respect to it. For example, if you want to check if there is a file called Output and if not, you can create it (by using lines 27 to 29), you don't need to specify the full file, but you just call the name of the folder as it works starting from the project. Similarly, in lines 43 and 44, the full path is needed, it was already extracted in line 26, so there is no need to type it explicitly and the computer automatically does this task.

```

46 # 2| Chapters -----
47 source(paste0(scriptLocation, 'Chapter 00 - Descriptive Statistics.R'), encoding = 'UTF-8')
48 gc()
49 source(paste0(scriptLocation, 'Chapter 01 - Profiling of beneficiaries.R'), encoding = 'UTF-8')
50 gc()
51 source(paste0(scriptLocation, 'Chapter 02 - Targeting characteristics.R'), encoding = 'UTF-8')
52 gc()
53 source(paste0(scriptLocation, 'Chapter 03 - Coverage evaluation.R'), encoding = 'UTF-8')
54
55 endTime = Sys.time()
56 endTime - startTime

```

Finally, the last part of the master runs each of the relevant scripts that are associated with the chapters of the report and finishes the time counter. In this way it is possible to calculate the time that the computer took to process all the report.

This chapter of the manual only covers chapter 00 of the report, as explained earlier. Part 2 will cover each of the other chapters.

### Producing descriptive statistics

In a similar way as the master file was presented, the presentation of chapter 00 highlights only the key elements of the code and the rest are left for the trainee to practice.

```

1  outputLocation = paste0(userLocation, '/Output/Chapter 00/')
2
3 # 1.1| Figure labels and frame-----
4 dataPlots <- read_excel(paste0(inputLocation, 'Dictionary/Dictionary.xlsx'),
5                         sheet = 'Labels')
6 dataPlots[is.na(dataPlots)] <- ''
7
8 dataframeFinal <- read_excel(paste0(inputLocation, 'Data/RawData.xlsx'))
9
10 # 1.2| Dictionaries -----
11 Region <- as.matrix(read_excel(paste0(inputLocation, 'Dictionary/Dictionary.xlsx'), sheet = 'categories', range = 'A1:D8'))
12 quantiles1 <- as.matrix(read_excel(paste0(inputLocation, 'Dictionary/Dictionary.xlsx'), sheet = 'categories', range = 'F1:I11'))
13 response1 <- as.matrix(read_excel(paste0(inputLocation, 'Dictionary/Dictionary.xlsx'), sheet = 'categories', range = 'K1:N3'))
14 characteristics <- as.matrix(read_excel(paste0(inputLocation, 'Dictionary/Dictionary.xlsx'), sheet = 'categories', range = 'P1:S36'))
15 studies <- as.matrix(read_excel(paste0(inputLocation, 'Dictionary/Dictionary.xlsx'), sheet = 'categories', range = 'AE1:AH7'))
16 labor <- as.matrix(read_excel(paste0(inputLocation, 'Dictionary/Dictionary.xlsx'), sheet = 'categories', range = 'AJ1:AM4'))
17 gender <- as.matrix(read_excel(paste0(inputLocation, 'Dictionary/Dictionary.xlsx'), sheet = 'categories', range = 'AO1:AR3'))
18 deciles <- as.matrix(read_excel(paste0(inputLocation, 'Dictionary/Dictionary.xlsx'), sheet = 'categories', range = 'AT1:AW11'))
19 confirmation <- as.matrix(read_excel(paste0(inputLocation, 'Dictionary/Dictionary.xlsx'), sheet = 'categories', range = 'KS:N7'))
20 translationVar <- as.matrix(read_excel(paste0(inputLocation, 'Dictionary/Dictionary.xlsx'), sheet = 'categories', range = 'CH1:CK10'))
21

```

The first part of the code creates a variable for all the outputs created in this script and then uploads all the tables associated with the rad data (line 8) and with the dictionaries (line 4-7 and lines 11-20). Doing this process at the beginning guarantees that the code is organized and that any change on the dictionary that requires a change in the code (for example, a new category appears) is easily spotted. Finally notice that this code does not need libraries as it will be run by the master file which already did this. It is worth mentioning that you cannot run this code alone. If you want to do so, first run the first codes of the Master (until section 2) and then run this code.

```

22- # 1.3| Factors -----
23 dataframe = dataframeFinal
24 dataframe$region = factor(dataframe$region,
25                           levels = Region[,2],
26                           labels = Region[,4-language])
27 dataframe$internetAcces = factor(dataframe$internetAcces,
28                                  levels = response1[,2],
29                                  labels = response1[,4-language])
30 dataframe$healthAcces = factor(dataframe$healthAcces,
31                                 levels = response1[,2],
32                                 labels = response1[,4-language])
33 dataframe$gender = factor(dataframe$gender,
34                            levels = gender[,2],
35                            labels = gender[,4-language])
36 dataframe$education = factor(dataframe$education,
37                               levels = studies[,2],
38                               labels = studies[,4-language])
39 dataframe$employment = factor(dataframe$employment,
40                                levels = labor[,2],
41                                labels = labor[,4-language])
42 dataframe$incomeQuantile = factor(dataframe$incomeQuantile,
43                                   levels = deciles[,2],
44                                   labels = deciles[,4-language])
45 dataframe$participant = factor(dataframe$participant,
46                                 levels = confirmation[,2],
47                                 labels = confirmation[,4-language])
48 |
49
50 dataParticipation = dataframe
51
52- # 2| Analysis -----
53 wb = openxlsx::createWorkbook(creator = 'ESCWA')
54

```

This next section of the code does the translation exercise of all the variables that are in the dataset. It is recommended to do it at a very early stage, preferably, from the original dataset so that it is automatically incorporated in any subsequent line of code. Once this is done, the Excel design is created, and the production of graphs and tables begin.

```

55- # 2.1| Descriptive statistics -----
56- # '0_1' -----
57 rowLine = 1
58 jumpOfRows = 20
59 colPlots = 1
60 colTables = 12
61
62 indicatorCode <- '0_1'
63 part <- ''
64 labCodes <- dataPlots %>% subset(Code == paste0(indicatorCode, part))
65 newSheet <- addWorksheet(wb, sheetName = indicatorCode)
66
67 # To export:
68 base_exp = 1
69 heightExp = 1.1^6
70 widthExp = 1
71 scale_factor = base_exp/widthExp
72
73 newPlot = dataFrame %>%
74 ggplot(aes(x=gender, y=age, fill=region)) +
75 geom_violin(show.legend = F) +
76 facet_wrap(~region, ncol=2)+
77 labs(title = paste0(as.character(labCodes[, 5*(1-language)+2])),
78 subtitle = as.character(labCodes[, 5*(1-language)+3]),
79 caption = as.character(labCodes[, 5*(1-language)+6]),
80 x = as.character(labCodes[, 5*(1-language)+4]),
81 y = as.character(labCodes[, 5*(1-language)+5])) +
82 scale_fill_brewer(palette = as.character(labCodes[, 12])) +
83 theme_classic() +
84 scale_y_continuous(expand = expansion(mult = c(0, .05)),
85 limits = c(0, NA),
86 position = if (language == 0) {'right'} else {'left'}) +
87 theme(legend.position = 'bottom',
88 text = element_text(family = 'Georgia'),
89 axis.text.x = element_text(size = scale_factor * 10, family = 'Georgia'),
90 axis.text.y = element_text(size = scale_factor * 10, family = 'Georgia'),
91 axis.title.x = element_text(hjust = 0.5, size = scale_factor * 10, family = 'Georgia'),
92 axis.title.y = element_text(hjust = 0.5, size = scale_factor * 10, family = 'Georgia'),
93 legend.text = element_text(size = scale_factor * 10, family = 'Georgia'),
94 strip.text = element_text(size = scale_factor * 10, family = 'Georgia'),
95 plot.title = element_text(face = 'bold', size = 10, family = 'Georgia', hjust = if (language == 0) {1} else {0}),
96 plot.subtitle = element_text(size = 10, family = 'Georgia', hjust = if (language == 0) {1} else {0}),
97 legend.key.size = unit(0.5 * scale_factor, 'cm')) +
98 guides(fill = guide_legend(title = ''))
99
100 # To excel:
101 fileName = paste0(fileName = paste0(outputLocation, indicatorCode, part,
102 if(language == 1) {'(English)'} else {'(Arab)'}, '.png'))
103 ggsave(fileName, plot = newPlot, width = 6 * widthExp, height = 4 * heightExp * widthExp, scale = scale_factor)
104 insertImage(wb, file = fileName, sheet = indicatorCode, startRow = rowLine, startCol = colPlots, width = 6 * widthExp, height = 4 * heightExp)
105 rowLine = rowLine + jumpOfRows
106

```

All this block of text is used to create a single variable. Most of it has already been covered in Chapter 4, but it is important to highlight three elements:

1. Lines 66-71 are used to fix some of the proportions of the graph so that it looks better when printed.
2. Sometimes several graphs can go one below the other in the same sheet of the excel file (specially in those cases when the topic is similar). Then, line 105 helps to tell R that there is a new graph and in this case it does not begin in the first row of the Excel sheet, but jumps some rows down so that the new graph appears just after the former one.
3. Notice that the file name is coded in a way that the *png* file appears with the code of the indicator and also indicates the language of the graph so that the analyst can quickly know about the graph and its language just by checking the file name.

Then, the rest of the script is just a long repetition of different indicators until finally, the last lines print the Excel file.

```

1323- # Final step -----
1324 saveWorkbook(wb,
1325 file = paste0(outputLocation, if(language == 1) {'CH 0 EN.xlsx'} else {'CH 0 AR.xlsx'}),
1326 overwrite = TRUE)

```

As a final remark notice that the script is coded in a way that converts the dictionary in a data framework. While being useful for writing the report, the dictionary is also a structured list of all the indicators that have been produced, graphed, and tabulated. Thus, it becomes a useful summary tools that synthesizes the content of the report.

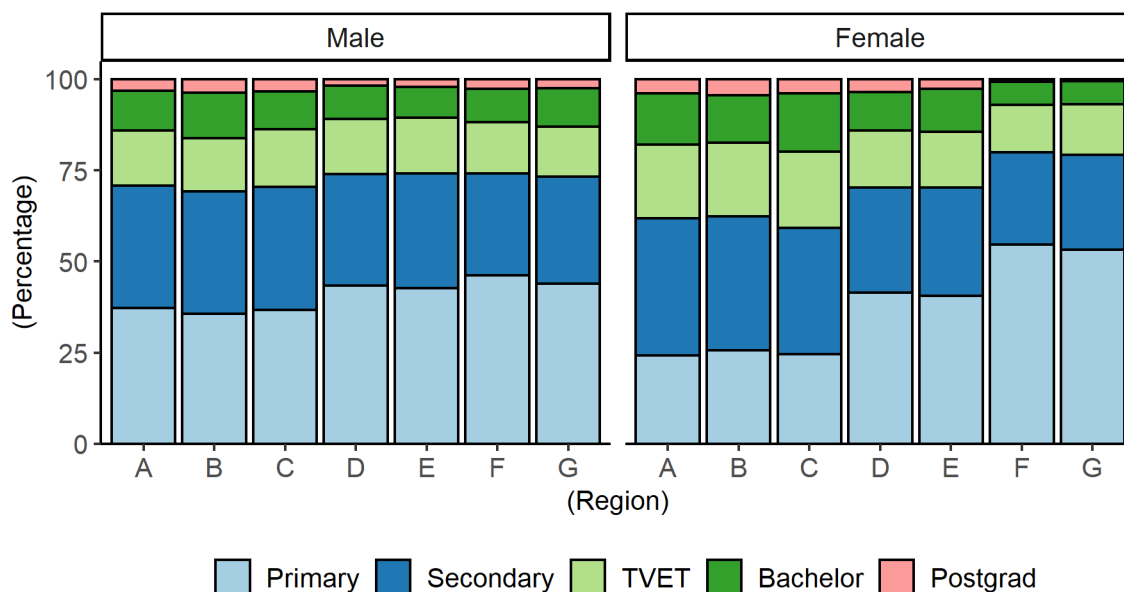
## Analyzing the results

Everything that is done until this point allows you to get to a point where the graphs can be produced, and the results can be evaluated. So now, with all the heavy lifting done, let's choose some variables

that can help you understand better the reality of the kingdom of AP. Notice that some of the graphs are not explained in Chapter 3, yet by checking the code, the trainee is expected to know where to add command lines to obtain the given result.

### Distribution of the highest educational level

By highest achieved grade

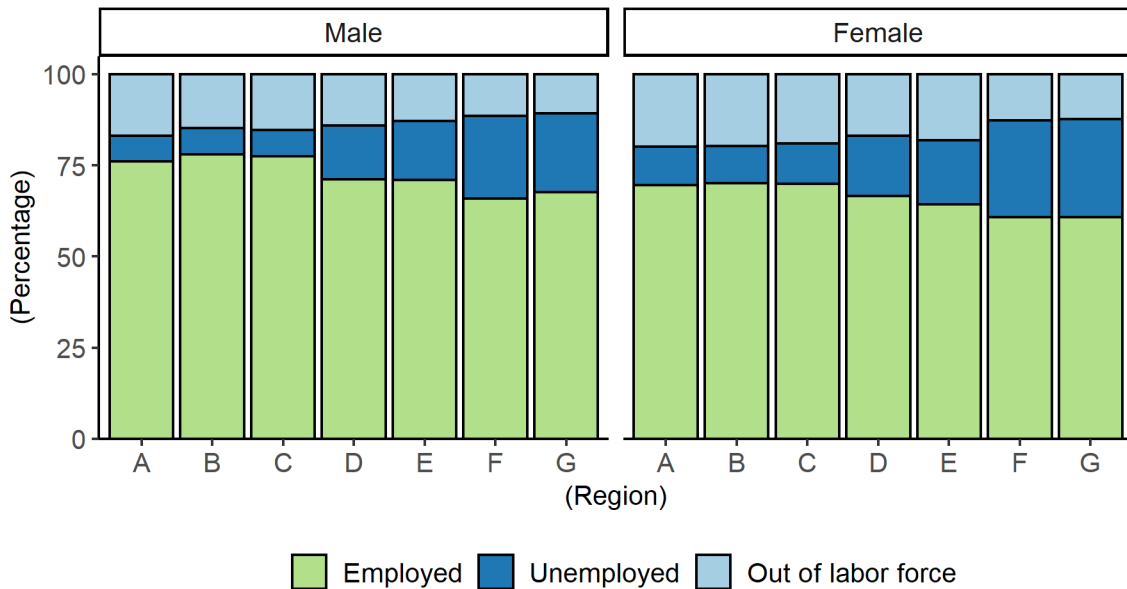


Source:  
ESCWA

Let's begin reviewing the education levels of the population. While for men these levels are relatively similar across regions, for women there are clear differences where Regions A-C have the most educated women and regions F-G have the least educated ones.

## Distribution of economically active population

By status



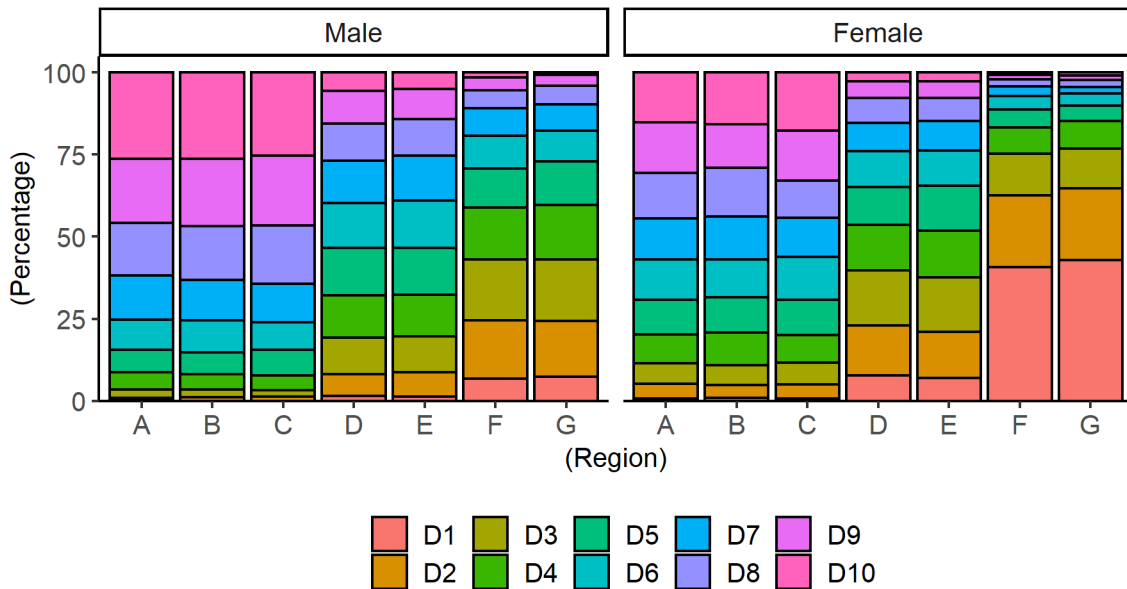
Source:  
ESCWA

In general women have lower engagement than men in the labour force, for both D and E and even more in F and G, the unemployment rates of women are higher. Notice that there is a big difference between being out of the labour force (not willing to work) and being unemployed (willing to work, but not being able to find the job). Hence, checking the history of the kingdom of AP, the analyst notices that regions F and G are quite poor and even though women are not so integrated in the labour force, the need for money has pushed many women in these regions to work. Yet, as the regions are poor, they are having issues to get the jobs.



### Distribution of income deciles

By region

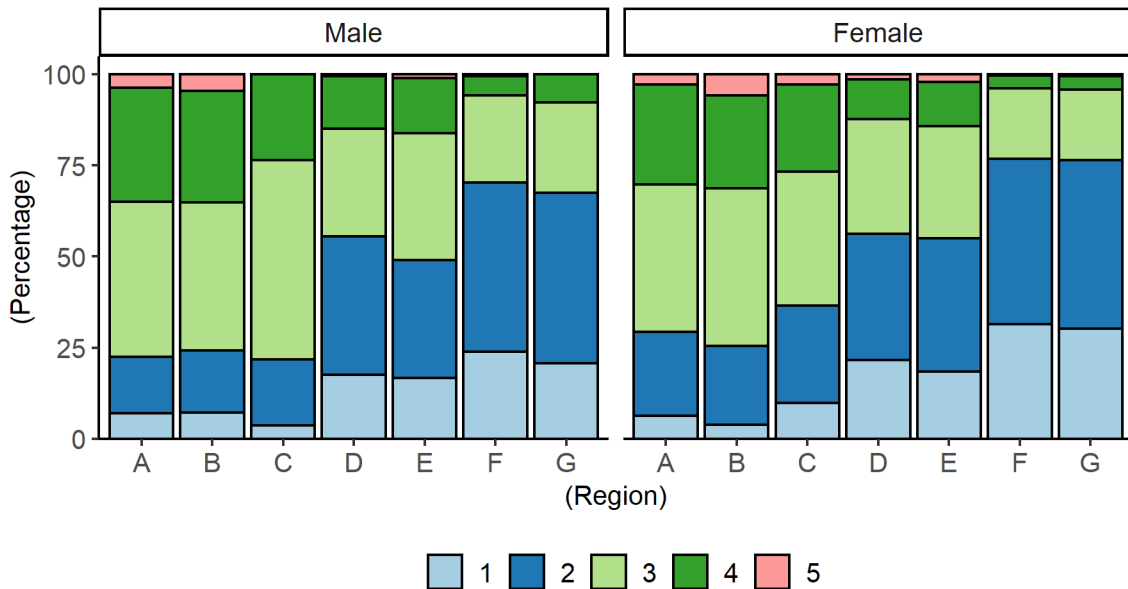


Source:  
ESCWA

Complementing the previous story, notice that the distribution of the income deciles highlights how poor women are concentrated in F and G. This is also true for men; however, the concentration is less. In these cases the analyst checks the history again and realizes that F and G are not only poor but suffered from war and this has taken away most of the men. Thus, in those areas most of the remaining households are led by women. In contrast to that, regions A – C appear as prosperous areas that concentrate the richest individuals of the country.

### Distribution of income sources

By income source

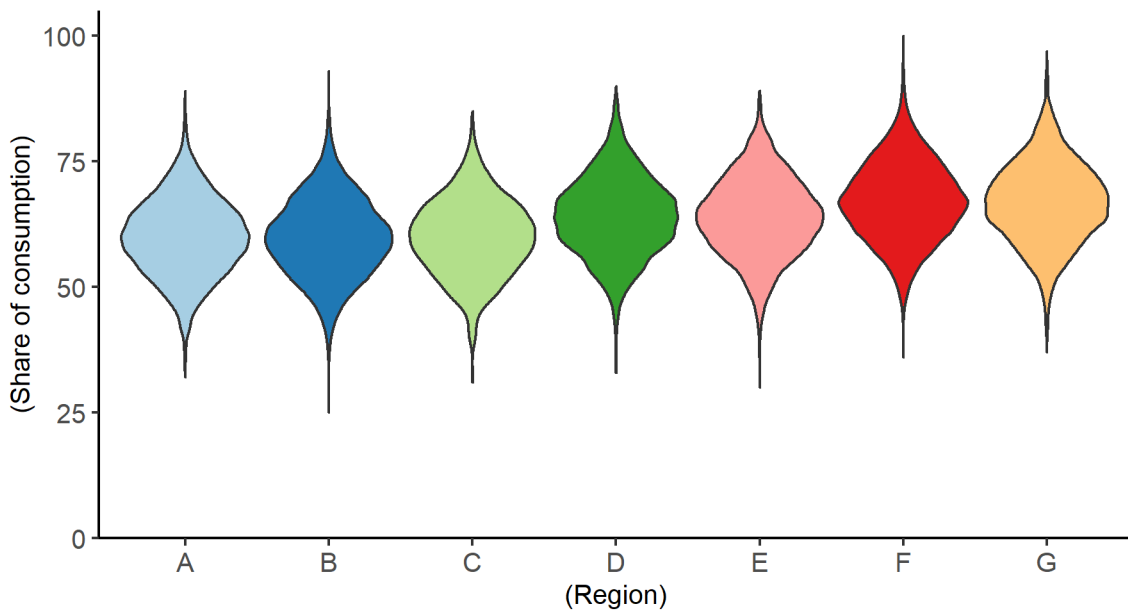


Source: ESCWA

Consequently, in relation to the previous story, individuals from the first regions have multiple sources of income, while individuals in the last regions have fewer sources.

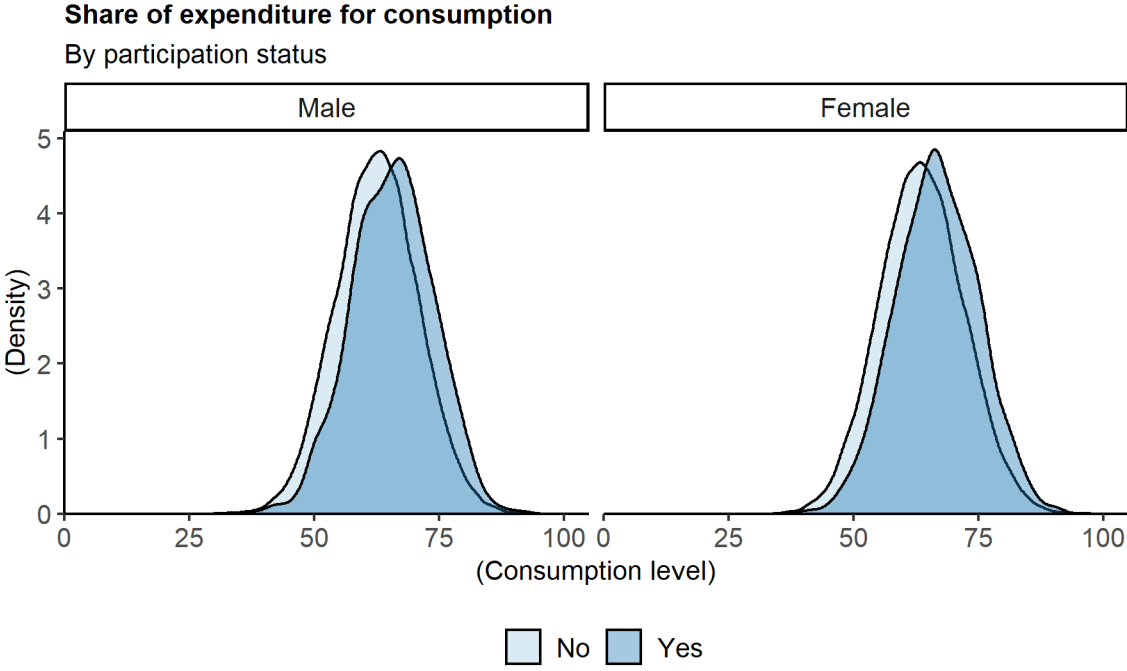
### Share of expenditure

Consumption



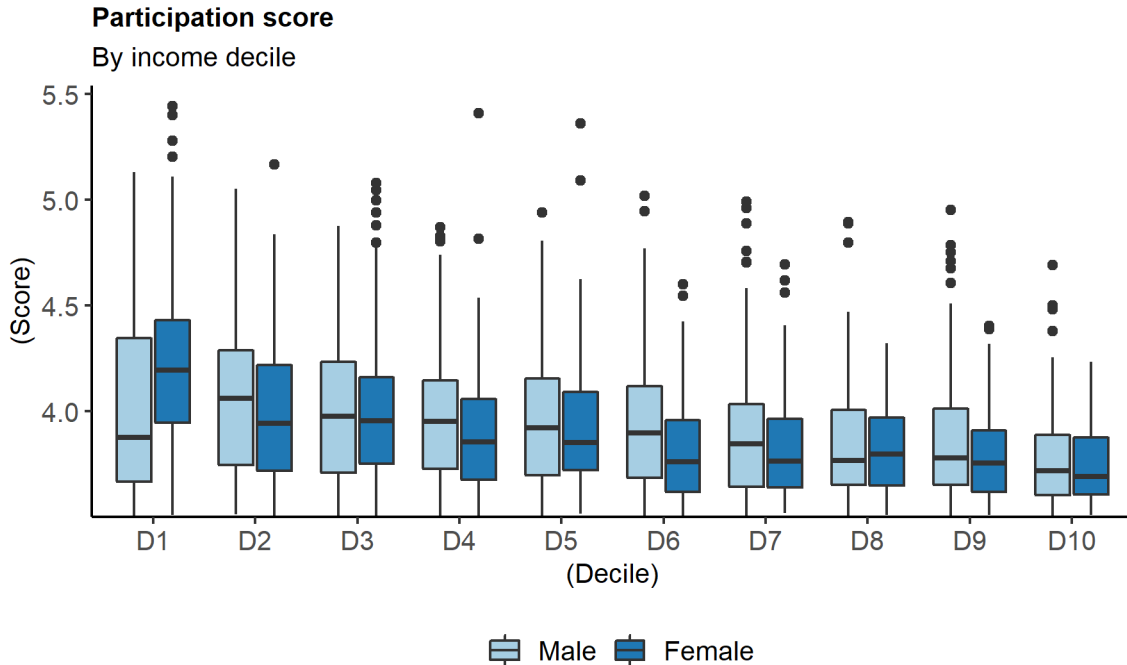
Source: ESCWA

Aligned with the previous results individuals in the poor areas tend to dedicate larger shares of their income to support their basic needs.



Source:  
ESCWA

On this note, it is interesting to see how the social programme is taking into account this, and in general you can see that the individuals participating in the programme are, in general, dedicating larger shares of their expenditure to basic needs. This is good, as it means that the programme is targeting the population that needs more support. Nevertheless, the fact that the distributions are very near each other indicates that the prioritization of beneficiaries does not focus on this variable, and probably gives more weight to other variables.



Source:  
ESCWA

Finally, by analysing the distribution of scores along deciles, you notice that (as expected), the score (representing the need of the individuals) decrease with income. However, two elements are important to highlight here. First, there is an odd gap in decile 6. Second it is interesting to observe people from the highest decile being beneficiaries of the programme.

At this moment the descriptive review allows you to have a better picture of the regional differences of the country and some points that might be relevant to highlight during the SPP-RAF. Now it is left for the trainee to check the remaining graphs and develop complementary stories to the ones presented in this chapter.

### Conclusions

At this stage, the trainee has acquired the basic tools needed to work in R and is capable of generating reports that produce descriptive statistics that can guide the policy maker in understanding the general situation of the social assistance programme. Having these fundamental ideas cleared, the second part of the manual goes directly to the description of the analytical tools and how they can provide deeper insights for the assessment.

## Part 2: SPP —RAF

## Chapter 6: Profiling beneficiaries

The following three chapters focus on the first three steps of the SPP-RAF, these chapters use the example that has been introduced as a guideline in chapter 2, and incorporates the knowledge obtained about the context of this country to illustrate the creation of the analyses.

### Purpose of profiling beneficiaries

The first stage of the SPP-RAF aims to identify the socio-demographic characteristics of the programme beneficiaries. These characteristics allow policymakers to combine individuals that have similar profiles into groups, and to visualize structural factors that make these population groups vulnerable. Once these common characteristics are identified it is possible to design policies that tackle the sources of vulnerability and improve the welfare of these populations.

#### Output:

- Construction of clusters (or groups) of beneficiaries based on socio-demographic characteristics that they share.
- Better understanding of the nature on poverty and vulnerability, as well as its regional variations.
- Rapid screening method to identify individuals belonging to each cluster.

#### Outcome:

- Strategic identification of structural vulnerability determinants derived from the common characteristics of the beneficiaries.
- Obtaining evidence that allows you to devise targeted services to meet the needs of specific clusters (or groups) of beneficiaries and reduce their vulnerability in the longer run.

#### Data requirements:

This stage requires the creation of a dataset that includes all the individuals that are beneficiaries of a given programme during a specific period and their characteristics. Ideally, the study unit should be “individual”, but household-level information can be used depending on data availability.

For each individual, the following lists of socio-demographic characteristics are required:

- Gender, age, education, location, marital status, income level, type of contract, occupation and industry (if working), and nationality.
- Household values of the targeting variables used by the social assistance programme to determine eligibility of applicants (or as many as possible).
- Other variables that are of interest to the policy makers.

### Clustering

This section explains the conceptual elements of a statistical clustering. While it explains the basic concepts of the clustering and focuses on the ones recommended for the SPP-RAF, avid learners can find the book of Zelterman (2015) as a good source to go deeper into these topics; in particular chapters 11 and 10 (in that order).

The problem of clustering is relatively easy to state. There is a number of individuals with different traits and the challenge is to group them in few clusters where the characteristics within each group are similar, and the characteristics between groups are different. However, that sentence in itself brings two challenges:

1. What criteria can we use to define how similar or different two individuals are (step 1)?

2. How can we decide when two individuals are too different to belong to different groups (step 2)?

Each of these questions comes with challenges and multiple alternatives that are going to be explored. Still, as in previous topics, there is no unique answer to this issues. Indeed, there is a theorem called “Ugly duckling” (Watanabe, 1969) that proves that for clustering, there is always a degree of subjective criteria, and it is at this point that the analyst needs to have background knowledge that allow him to understand the data better in light of the context.

### Step 1: Distances

The first step of a clustering is the definition of a distance between individuals. And for illustration purposes, these are three examples of them:

**Euclidean:** This measures the distance between two points  $p$  and  $q$  is defined by the length of the segment that joins both points. This is calculated by the Pythagorean Theorem, as shown in the next example.

-Example: Ahmed has 40 years and 20 years of working experience. In contrast Mohammed has 37.5 years and 15 years of working experience. In this case the distance is

$$\sqrt{(40 - 37.5)^2 + (20 - 15)^2} = 5.59$$

**Manhattan:** This measurement defines the distance between two points by the sum of the absolute differences between each measurement. This measurement is more robust than the Euclidean against outliers because it doesn't squares the distance.

-Example: Ahmed has 32 years and 8 years of working experience. In contrast Mohammed has 21 years and 3 years of working experience. In this case the distance is

$$|40 - 37.5| + |20 - 15| = 7.5$$

**Jaccard Index:** this measurement is applied to determine the similarity between categories. Then to turn it into a distance, you subtract the index to the value of 1.

-Example: Ahmed has internet, TV, but has no car. Instead Mohammed has no internet or car, but he has a TV. Notice that they coincide in two options. Thus, the distance is

$$1 - \frac{2}{3} = 0.3333$$

As you can see, each one is different and depending on the context one can be preferred than the others. For example, if the distance was to measure car movements, usually Manhattan is better, as it mimics the need of cars to go along a grid of streets instead of doing diagonals that might cut buildings. Similarly, Jaccard is quite effective for some cases where the variables are categories. Still for this section all the work will be concentrated in the Euclidean as it can also incorporate categories, and, given the right modifications, can incorporate the structure of the data. But before going there, lets understand the problems of the Euclidean in itself.

Let's add two more individuals to the example, Mona with 40 years and 15 years of experience and Maryam with 37.5 years and 20 years of experience. By doing the same procedure as before, the trainee will find that the distance between Maryam and Mona is again 5.59. However, if we decide to measure the age in days instead of years, suddenly the distances are 16,303 for the two men and 15,572 for the women. This is not only a change of scale (indeed is not a change of scale), but is a structural change in the measurement. For example, before, the ratio between the men and the women

distances was 1 (they were the same) but now it is 1.05. This changes of proportions affect the next step of the clustering and therefore are undesirable. Yet there is another conceptual challenge presented in figure 2.



Figure 2: Distances and variances

Naturally age and work experience are highly related, as it is illustrated at the top of the graph. Moreover, as you can see from the location of the four individuals in the example, geometrically they are at the same distance, but conceptually this is not the case. While Mohammed and Ahmed are aligned with the tendency, Maryam and Mona are against the tendency, so it is more common to see the pattern of Ahmed and Mohammed instead of the pattern of Maryam and Mona. Fortunately, if variables are adequately transformed adjusting them by the covariance relation, then the Euclidean distance will be sufficient. When this transformation is done, the distance is no longer called Euclidean but becomes the Mahalanobis distance, and its main advantages is that it is invariant of scale and it includes the covariance structure (Mahalanobis, 1936). In the later section, while discussing the code, the manual explains how this modification to the Euclidean distance is done in R.

### Step 2: Clustering

For illustration purposes consider a new scenario where Mona, Maryam, Ahmed, and Mohammed have the distance matrix represented in table 1 and done via a distance criterion (at this step the only thing that matters is the matrix, not the way it is constructed).



Table 1 Distance matrix

	Mona	Maryam	Ahmed	Mohammed	Fatih	Balsam
Mona	0	2	3	9	4	5
Maryam	2	0	1	3	10	1
Ahmed	3	1	0	8	5	3
Mohammed	9	3	8	0	10	5
Fatih	4	10	5	10	0	2
Balsam	5	1	3	5	2	0

Naturally, the diagonal is 0 because the distance with yourself is null. Now, how can we begin to group individuals? Naturally Ahmed and Maryam need to be together, because they are at distance 1. But now, what about Balsam? She is quite near to Maryam (distance 1), but is distance 3 of Ahmed. Moreover, Balsam is near Fatih, but Fatih is very far from Maryam. Currently there are several options to do this exercise, three of them are represented in figure 3.

**Single linkage:** The distance between individuals in group A and individuals in group B is the minimum distance between someone from A and someone from B.

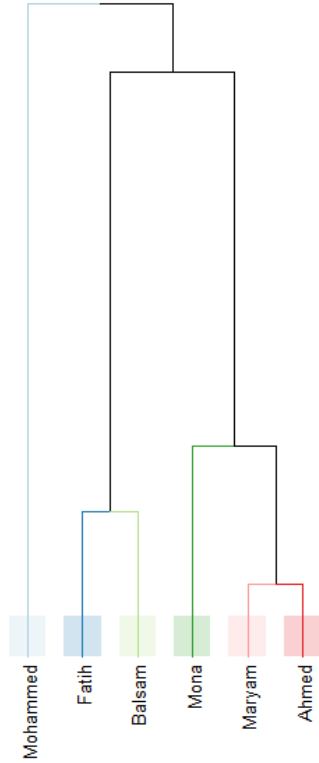
**Complete linkage:** The distance between individuals in group A and individuals in group B is the maximum distance between someone from A and someone from B.

**Ward linkage:** The distance between groups is calculated in a way that the variance of the groups is incorporated (Ward, 1963).

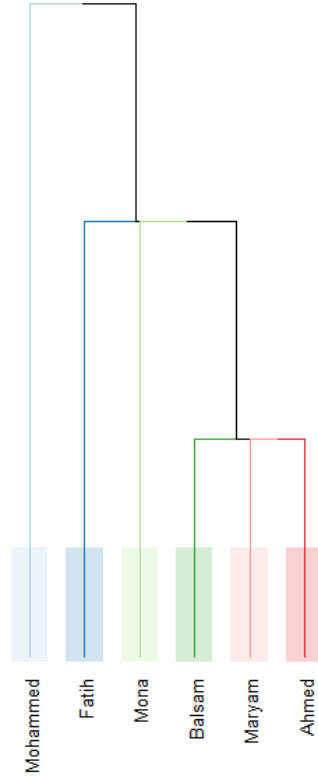
As you can see in the results, the way to cluster them is different. For example, Ahmed and Maryam are always together, but for example, in the complete linkage, the affinity between Balsam and Fatih makes her far away from Mary and Ahmed, yet in the single linkage, the fact that she is near Maryam weights more. In general, there is no clear guideline on what method to choose, and the recommendation is to test with the different linkages to see which ones reflects the data better. Yet, if the idea is to have balanced group, Ward tends to work better since the other two methods usually produce extreme trees due to the minimum and maximum criteria they manage.

Once you have a decision tree, the next challenge, at the policy level, is to identify the number of groups you want to profile. Let's take for example the Ward Linkage of Figure 3, and assume the government is willing to create three targeting strategies. Hence the recommendation from this clustering is to develop a targeting strategy for Mona, Ahmed and Maryam, another one for Balsam and Fatih, and a last one for Mohammed. In contrast if we use the single linkage, the recommendation would be Balsam, Maryam, and Ahmed in one profile, Mona and Fatih in a different group, and finally Mohammed. Hence, the techniques have different groups but they also have commonalities that help the policy makers to understand the situation. In this case, definitely Mohammed is too different from the rest, and Ahmed and Maryam can always be grouped together.

Ward Linkage



Single Linkage



Complete Linkage

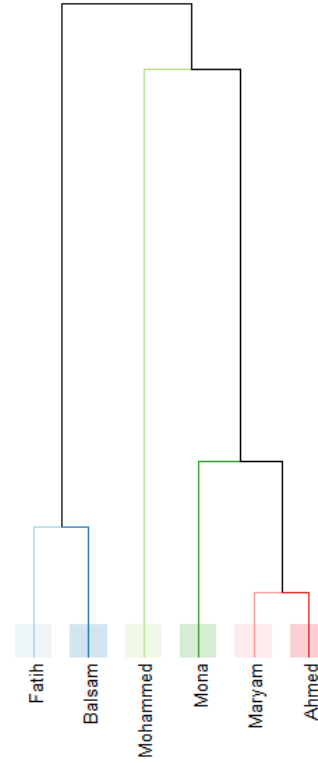
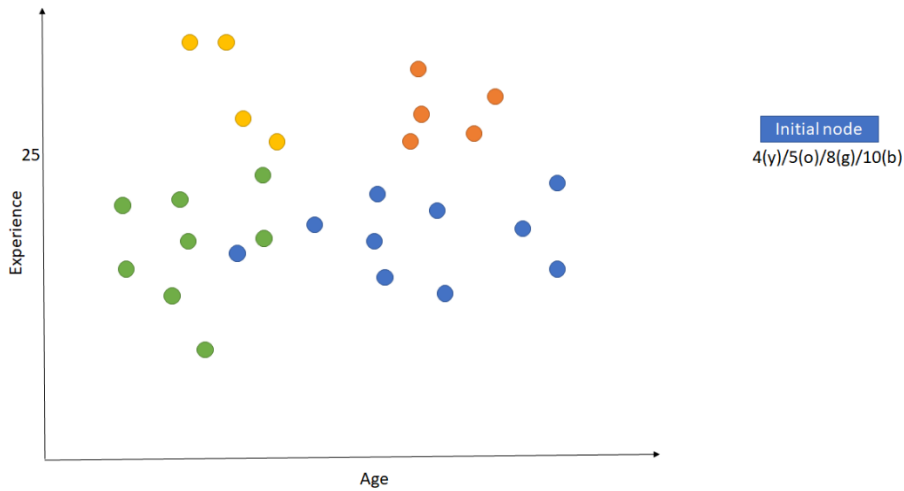


Figure 3 Dendrograms

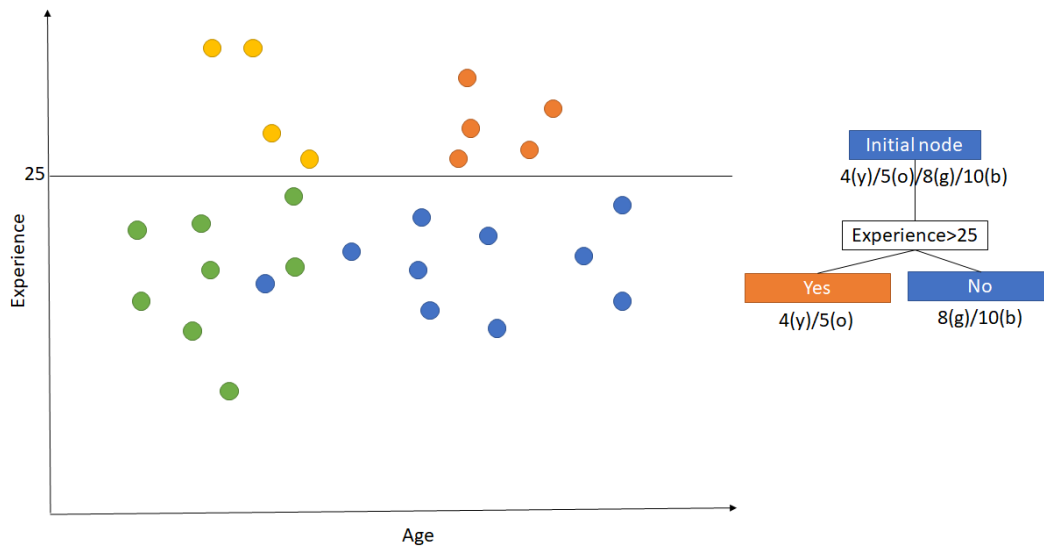
### Classification trees

While the clustering is a nice visual device that help you to group individuals, it doesn't allow you to understand why these individuals are aggregated in that way; what are the variables leading these commonalities. To solve that problem, two techniques are developed, one is highly descriptive, therefore its concept is directly explained in the example, and the other is more technical and therefore this section is dedicated to it.

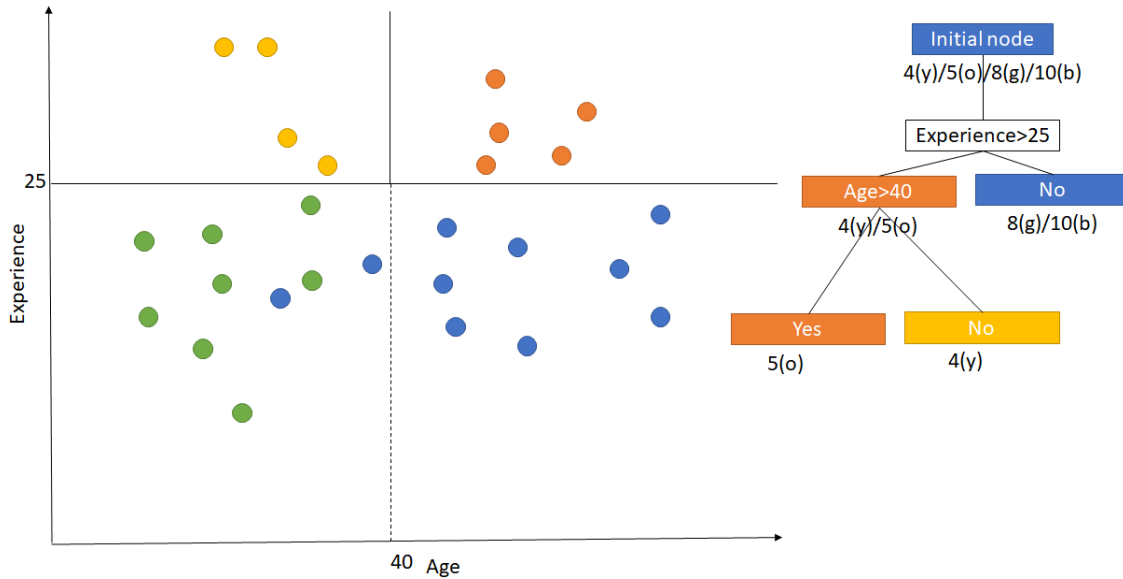
The technical method is known as classifications and regression trees (CART) and is one of the fundamentals of machine learning. Due to background prerequisites, this chapter explains a sketch version of how it works, yet for avid readers, it is recommended to read Breiman (1984) to understand the details of its function.



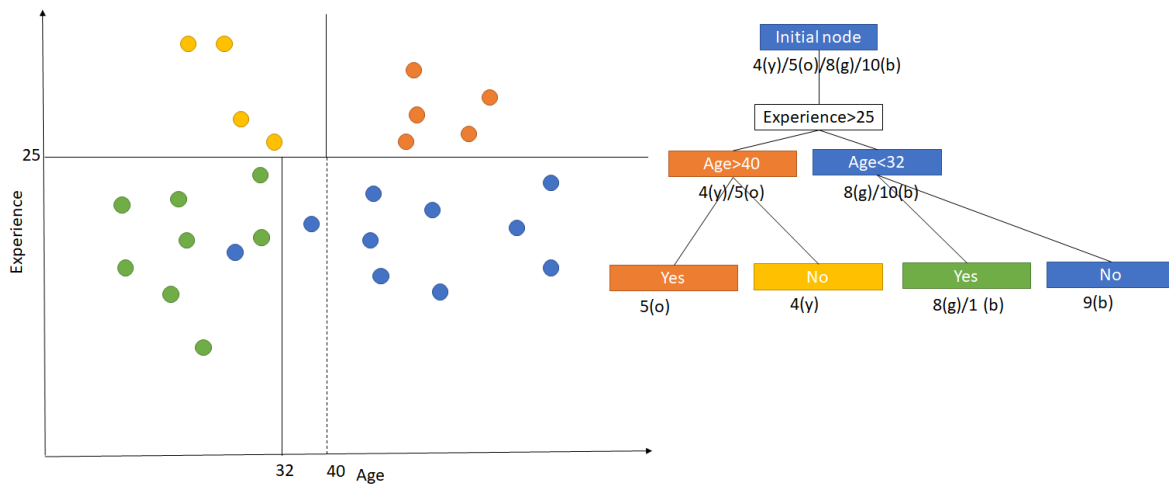
Originally the tree begins in the root, meaning all the population. In this case the bigger group is blue with 10 observations, so the root is assigned to blue. In other words, if you have no additional information of anything, your best guess to select a person randomly, is blue.



Then the algorithm realizes that the next most important criteria is the experience as it can create a clean division between groups. Now if the person is above 25 years of experience then he/she is orange (majority group) otherwise he/she is blue.



Following that logic, the next clean cut is at age above 40, for the group over 25 years of experience. So notice that the algorithm is prioritizing the questions on the profiling to understand what are the driving characteristics for each group. In this case. For orange and yellow, the first item is the experience and the second is the age.



Similar division is later done for the people with low experience, noting that the age cut is now 32 instead of 40. Also notice that the classification is not perfect (and doesn't need to be). In this case, there is one blue that got misclassified as green if you follow the tree criteria. However, from the statistical perspective, it is possible to keep on growing the tree until each leaf (end node) will have only one colour. However, by over fitting we have the risk that this different individual is appearing due to a typing error from the enumerator, and now our algorithm believes it is something structural. On the other hand, a tree is a quick way for profiling, so if a new individual arrives and you want to rapidly know what group he belongs to, then you ask two questions. Of course, asking more questions gives you more precise answers, but questions cost resources and therefore the fewer, the better.

Having clarified the two main theoretical elements of this section, the next section goes through the codes, in a similar way as it was done in chapter 5, and highlights the steps of the full profiling analysis.

## Profiling analysis

This section proceeds with the code identified as Chapter 01 (because the profiling is associated to the chapter 1 of the SPP-RAF). Notice, that the code starts in the same way as Chapter 00 because it uploads the data and format the dictionaries and categories to have all in the same language. However, in contrast to the previous code lines 37 to 39 change to make it clear that the profiling only applies for people benefiting from the program.

```
37 # 1.4| Subset -----
38 # Dataset related to those individuals that participate in social assistance programs.
39 dataParticipation = dataFrame %>% filter(participant == 'Yes')
```

The next part is the creation of dummy variables. In this dataset, there are several categorical variables. Thus to make sense with scores, these categories needs to be transformed into dummies, with the function `dummy_cols()`.

```
44 # 2.1| Machine learning techniques -----
45 # 2.1.1| Hierarchical clustering -----
46 # - Incremental dummies:
47 incrementalDummy = dataFrame %>%
48   dummy_cols(select_columns = c('region','gender','education',
49     | 'employment','incomeQuantile',
50     | 'internetAcces','healthAcces'),
51   remove_first_dummy = T)
52 dataParticipation = incrementalDummy %>% filter(participant == 'Yes')
53 dataParticipation = dataParticipation %>%
54   dplyr::select(-c('region', 'gender', 'education', 'employment',
55     'incomeQuantile', 'participant','internetAcces', 'healthAcces'))
56
57 dataParticipation = dataParticipation %>% dplyr::select(-c('score'))
58
59 rownames(dataParticipation)=paste0('Case', 1:dim(dataParticipation)[1])
60
61 # - Factores for the original dataset:
62 dataFrame$gender = factor(dataFrame$gender,
63   levels = gender[,2],
64   labels = gender[,4-language])
65 dataFrame$education = factor(dataFrame$education,
66   levels = studies[,2],
67   labels = studies[,4-language])
68 dataFrame$employment = factor(dataFrame$employment,
69   levels = labor[,2],
70   labels = labor[,4-language])
71 dataFrame$incomeQuantile = factor(dataFrame$incomeQuantile,
72   levels = deciles[,2],
73   labels = deciles[,4-language])
74 dataFrame$participant = factor(dataFrame$participant,
75   levels = confirmation[,2],
76   labels = confirmation[,4-language])
77
```

The other important element here is the inclusion of `dplyr::` before the function `select()`. Sometimes different packages that R uses have the same name for different functions. In this case, to avoid mistakes, the best practice is to use “`::`” to specify explicitly the package where the function comes from.

Once this transformation is done, the process to calculate the distance matrix starts

```

78 # '1_1'| Distance matrix -----
79 #Variance of dataset variables
80
81 aux1=apply(dataParticipation, 2, var)
82
83 #Select variable whose variance isn't 0 and scaling them
84
85 dataParticipation=dataParticipation[, aux1!=0 ]
86 dd <- mahalanobis.dist(dataParticipation)
87 dd =as.dist(dd)

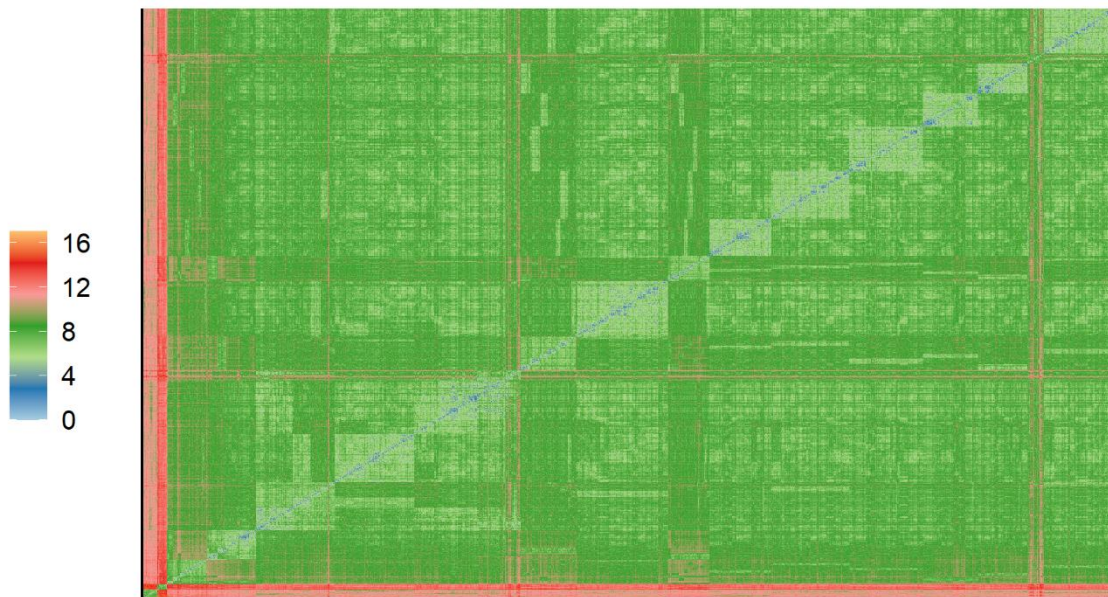
```

Among those lines, the key elements are the definition of the Mahalanobis distance, used in line 86 thru the function *mahalanobis.dist()*, and the transformation of that object into a distance object so that R can do the calculations (via *as.dist()*).

The next section of the code, until line 137 shows the way in which R represents a distance matrix. That part of the code is not covered here as it follows the same logic presented in chapter 3. However, the distance matrix is presented here to evaluate its main elements.

### Distance matrix

Measured by Mahalanobis norm



Source:  
ESCWA

This graphical representation of the distance matrix reveals individuals that are near or far from each other. Hence each column (and row) represents an individual, and the color of the cell reflects the distance between them, where blue is the minimum distance and red is the maximum distance. Naturally, the diagonal is blue. However, looking carefully, along the diagonal there are light blocks that represent those individuals who have relatively small distances within them. In contrast, notice that small light green square in the lower left corner. While they are similar within them, they are very different from any other individual (reflected in those red rows and columns around). This visualization is nice, yet it does not allow you to identify the clustering order, and that's why, similar to the conceptual explanation, the dendrogram is needed.

```

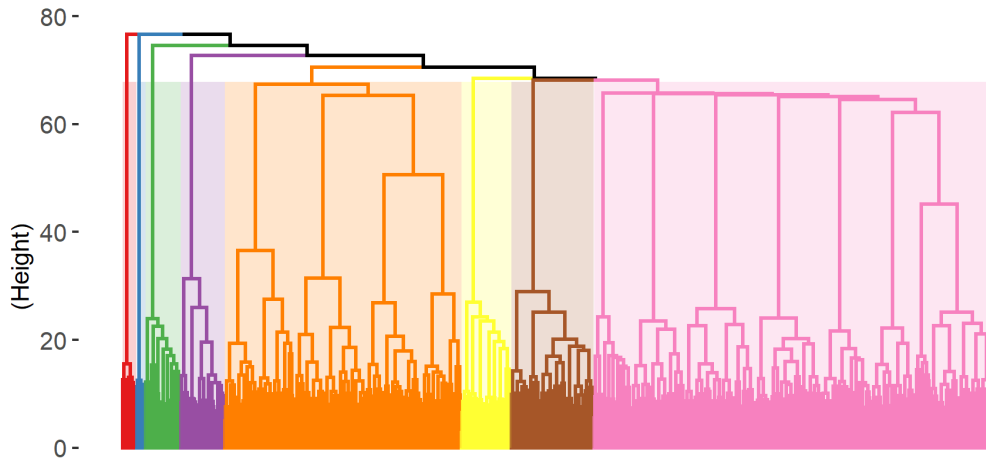
156 hc           = hclust(dd, method = 'ward.D2')
157 otter.dendro = as.dendrogram(hc)
158
159 dendro.plot = gg dendrogram(data = otter.dendro, rotate = TRUE)+
160   scale_x_discrete(expand = expansion(mult = c(0,0))) +
161   scale_y_continuous(expand = expansion(mult = c(0,0)))+
162   theme(axis.title.y = element_blank(),
163         axis.ticks.y = element_blank())
164
165 cutNumber=8
166
167 treeOrdering = hc$labels[hc$order]
168 sub_grp      = cutree(hc, k = cutNumber)
169 dataParticipation$Clusters = factor(sub_grp)
170
171 newPlot = fviz_dend(hc, k = cutNumber, color_labels_by_k = T, palette = 'Set1', show_labels = F,
172                   rect = T, lower_rect = -0.02, rect_border = 'Set1', rect_fill = T) +
173   labs(title = paste0(as.character(labCodes[, 5*(1-language)+2])),
174        subtitle = as.character(labCodes[, 5*(1-language)+3]),
175        caption = as.character(labCodes[, 5*(1-language)+6]),
176        x = as.character(labCodes[, 5*(1-language)+4]),
177        y = as.character(labCodes[, 5*(1-language)+5])) +
178   scale_y_continuous(position = if (language == 0) {'right'} else {'left'}) +
179   theme_classic() +
180   theme(legend.position = 'bottom',
181         text = element_text(family = 'Georgia'),
182         axis.line.x = element_blank(),
183         axis.line.y = element_blank(),
184         axis.text.x = element_blank(),
185         axis.ticks.x = element_blank(),
186         axis.text.y = element_text(size = scale_factor * 10, family = 'Georgia'),
187         axis.title.x = element_text(hjust = 0.5, size = scale_factor * 10, family = 'Georgia'),
188         axis.title.y = element_text(hjust = 0.5, size = scale_factor * 10, family = 'Georgia'),
189         legend.text = element_text(size = scale_factor * 10, family = 'Georgia'),
190         strip.text = element_text(size = scale_factor * 10, family = 'Georgia'),
191         plot.title = element_text(face = 'bold', size = 10, family = 'Georgia', hjust = if (language == 0) {1} else {0}),
192         plot.subtitle = element_text(size = 10, family = 'Georgia', hjust = if (language == 0) {1} else {0}),
193         legend.key.size = unit(0.5 * scale_factor, 'cm'))

```

For the dendrogram, the key code lines are between 156 and 193. Line 56 is the one that uses the distance matrix previously calculated in line 86 and produces the clustering using the Ward Linkages via the function `hclust()`. In case single or complete linkages are needed, it is in this line where the change needs to take place. The next line is just making it an object useful for the interpretation of R. Then, between lines 159 and 163, there is a small dendrogram created, but this one is not going to be used at this stage. Line 165 defines the number of groups we want to analyse and between lines 167-169 this information is saved so that it can be used later for posterior analyses. Finally, lines 171-193 create a nice dendrogram that provides insightful information about the structure of the beneficiaries. Notice that this plot is also a ggplot member, but there are subtle differences at the beginning and the reader is encouraged to play with these parameters to understand how to produce different graphs.

The resulting dendrogram is quite appealing. There are two very small groups of individuals (red and blue) that seem to have extreme differences with all the other groups. Then there are some reasonably sized groups, and finally there are two large groups, represented by the orange and pink colours, that are telling about a large set that s similar traits. With this information, the policymaker can begin his review on where the targeting can be most efficient. On one hand, if the policy targets the medium and smaller groups, it might be possible to have fast results, as the groups are very specific, yet the impact over the whole population is limited. On the contrary, if the policy maker aims at the big group, the results might be less clear due to the high variety within them (notice for example that the pink one can be subdivided also in 7 subgroups based on the clustering), but they are able to cover more people. These decisions depend on the need for results, the available budget, and the political preferences of the policy maker. Yet, by itself, the tool provides guidance on the alternatives.

### Cluster dendrogram



(Individuals)

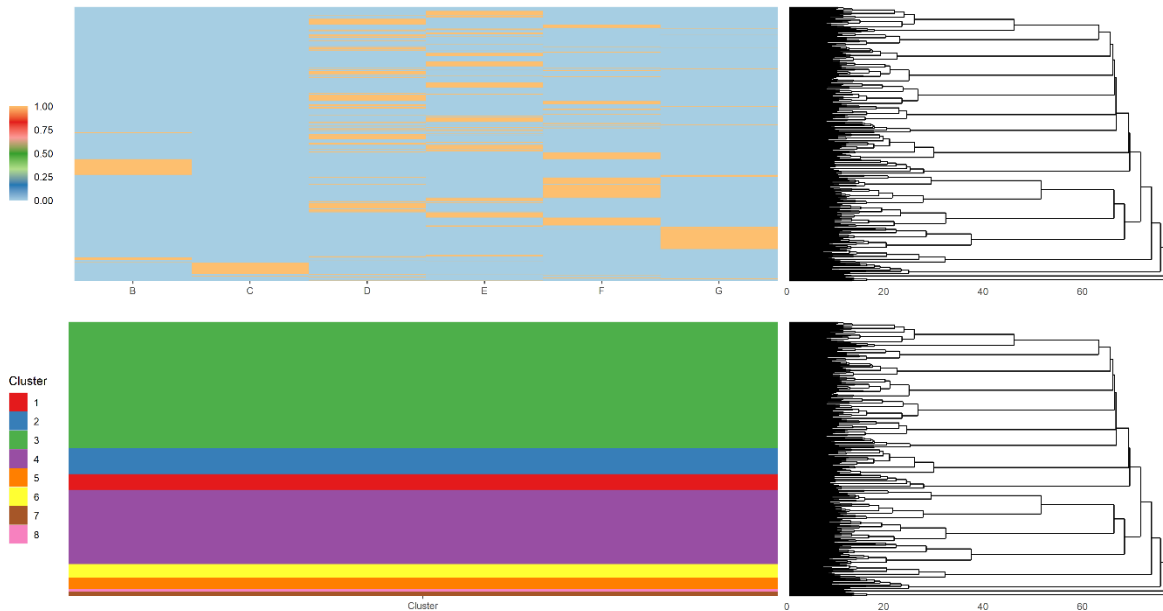
Source:  
ESCWA

However, as it was mentioned in the conceptual part, it is not enough to see the groups, the key element is to understand how they are defined. The first analysis is purely descriptive and it incorporates a dendrogram with a tile plot in a way that let us infer about the relevant variables.

In this case, the key code lines are those similar to 274

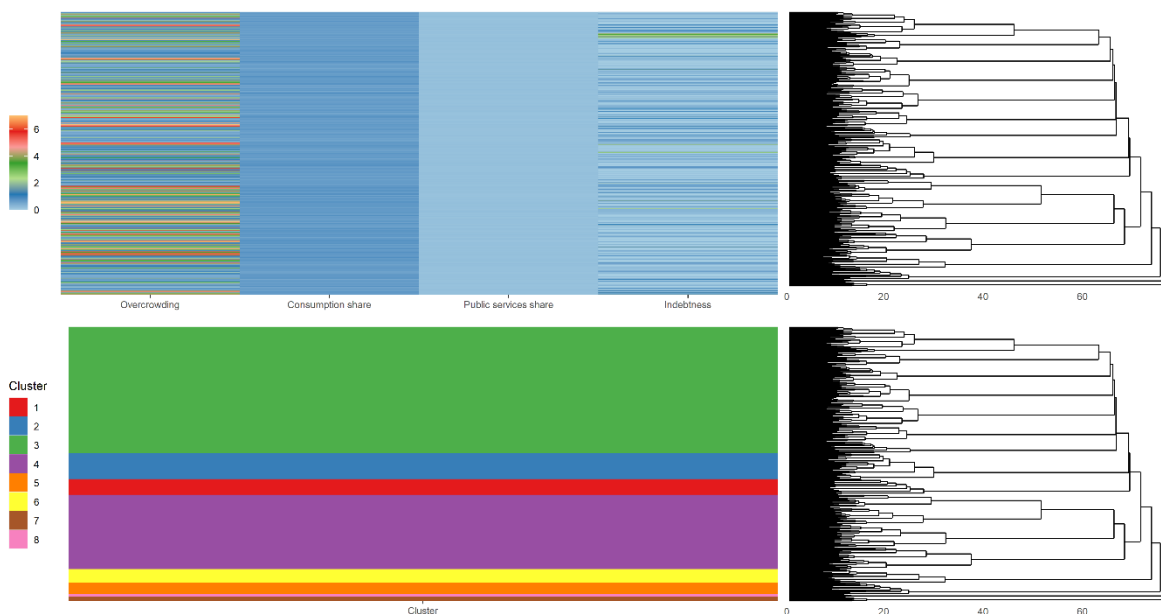
```
273 # Export:
274 newPlot = grid.arrange(heatmap.plot, dendro.plot, heatmap.plot2, dendro.plot, ncol = 2, widths = c(2, 1))
```

This line creates a plot by placing four plots together, in a frame that has 2 columns, but the width of the first column is twice as big as the width of the first one.





Take for illustration purposes the first graph that the code creates. The lower graphs highlight the dendrogram and the cluster. Hence, cluster 3, identified with the green colour takes the first part of the dendrogram. In contrast cluster 1, identified with the red colour, covers a small line in the middle of the dendrogram. Now, by matching those spaces in the upper figure, it is possible to check on the content of each cluster. For example, notice the orange frame on C (upper graph) that matches cluster 5 (orange). This highlights that cluster 5 is mostly driven by people in C. Similarly cluster 1 is driven by people in region B.



While not all the variables are helpful to conclude, some have very distinctive patterns. For example, checking the overcrowding, higher values are linked to cluster 4, and just above it, in cluster 1, values are quite low. For the sake of the manual, all variables have been displayed by groups, but for reports, and based on the criteria that the analyst wants to highlight, graphs can be done with more or less variables.

```

416 #1.3.4 Tabulate results
417 indicatorCode <- '1_35'
418 part <- ''
419 newSheet <- addWorksheet(wb, sheetName = indicatorCode)
420
421 presentData=dataParticipation%>%dplyr::select(c("Individual","Clusters"))%>%
422   right_join(Aux)%>%
423   group_by(variable, Clusters)%>%
424   summarise(value=mean(value))%>%
425   spread(variable,value)
426
427 writeDataTable(wb, sheet = indicatorCode, x = presentData, startRow = 1, startCol = 1)

```

Finally, the last part of that code produces a table with the summary of all the means that have been calculated. This can help in the revision of some elements that were not evident from the graphs. For example, notice that group 8 is the only one with Postgraduate students. Similarly, F and G, the poorest clusters, are concentrated in clusters 4 and 7, which as expected are also the ones with highest unemployment rates. In this way, for example, clusters 4 and 7 can begin to be identified as profiles or individuals at high risk.

Clusters	Age	Household	Overcrowd	Consumpt	Public ser	Indebtnes	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	31,6287425	3,33532934	1,55796113	0,62532934	0,1708982	0,26694611	1	0	0	0	0	0	0	0	0	0	0	0	0,26946108	0,37724551	0,18562874	0	0,15568862
2	34,0395683	4,01079137	2,40935252	0,6707554	0,17010791	0,28892086	0	0	0,32014388	0,29496403	0,25899281	0	0	0,39928058	0,23741007	0,13669065	0	0	0,39928058	0,23741007	0,13669065	0	0,21223022
3	32,5402214	3,59926199	1,79766854	0,65056089	0,16859041	0,30923247	0,00811808	0,00295203	0,28413284	0,26420664	0,09815498	0,01697417	0,26273063	0,30479705	0,12841328	0	0	0	0,26273063	0,30479705	0,12841328	0	0,22435424
4	35,5219573	4,39774153	2,92029187	0,69397742	0,16971142	0,26200753	0	0	0,13174404	0,12797992	0,38895859	0,32622334	0,39523212	0,15683814	0,11919699	0	0	0	0,39523212	0,15683814	0,11919699	0	0,3174404
5	33,4453782	3,1512605	1,45853341	0,62663866	0,16705882	0,31201681	0	1	0	0	0	0	0	0,22689076	0,35294118	0,1512605	0	0	0,22689076	0,35294118	0,1512605	0	0,16806723
6	32,5743243	2,85135135	1,10119316	0,63040541	0,16743243	0,30871622	0,21621622	0	0,12837838	0,12837838	0	0	0	0,20945946	0,34459459	0,12162162	0	0	0,20945946	0,34459459	0,12162162	0	0,19594595
7	36,212766	5,0212766	3,55851064	0,69170213	0,1693617	0,73340426	0,0212766	0	0,25531915	0,25531915	0,23404255	0,10638298	0,46808511	0,19148936	0,08510638	0	0	0	0,46808511	0,19148936	0,08510638	0	0,31914894
8	38,4666667	2,66666667	1,08746032	0,64333333	0,165	0,338	0,06666667	0,1	0,1	0,23333333	0,16666667	0,03333333	0	0	0	0	0	0	0,16666667	0,03333333	0	0	1,03333333

The final part of the process is the creation of the decision tree. The decision tree is long yet relatively easy to calculate.

```

430- # '1_4' Decision tree -----
431- # Tuning -----
432 # Separation of samples:
433 set.seed(123)
434 dataParticipation=dataFrameFinal%>%filter(participant %in% as.character(confirmation[2,3:4]))%>%dplyr::select(-c('score', 'partici
435 lang=1
436
437 dataParticipation$region = factor(dataParticipation$region,
438   levels = Region[,2],
439   labels = Region[,4-lang])
440 dataParticipation$internetAcces = factor(dataParticipation$internetAcces,
441   levels = response1[,2],
442   labels = response1[,4-lang])
443 dataParticipation$healthAcces = factor(dataParticipation$healthAcces,
444   levels = response1[,2],
445   labels = response1[,4-lang])
446 dataParticipation$gender = factor(dataParticipation$gender,
447   levels = gender[,2],
448   labels = gender[,4-lang])
449 dataParticipation$education = factor(dataParticipation$education,
450   levels = studies[,2],
451   labels = studies[,4-lang])
452 dataParticipation$employment = factor(dataParticipation$employment,
453   levels = labor[,2],
454   labels = labor[,4-lang])
455 dataParticipation$incomeQuantile = factor(dataParticipation$incomeQuantile,
456   levels = deciles[,2],
457   labels = deciles[,4-lang])
458
459 dataParticipation$Clusters=factor($sub_grp)

```

The first part takes the original data frame, filters for those individuals that were part of the clustering exercise, put the relevant factors and finalizes (line 459) and add to it the clusters that have been previously calculated.

In this point there is an important technical debate to be done regarding the tree. As it was explained in the conceptual discussion, the length of the tree (how many times we cut) gives better fitting, but it comes at a cost of over fitting the data. Therefore, the analyst have to be balanced and identify the right side. Yet, this is not an exact science. For avid readers, the book of Gareth et al. (2017) provides an in-depth analysis of this issue. However for the readers interested in the concept but not in the details, the following link provides practical explanations about the issue: <https://towardsdatascience.com/how-to-find-decision-tree-depth-via-cross-validation-2bf143f0f3d6>.

Coping with both sides of the debate, the written code uses an automatic optimization algorithm (lines 415-518) where a cross-validation technique is used to check trees based on their fitting and overfitting. However, the overall recommendation is to do some manual experimentation as it allows a better understanding for the problem. Thus, the code will be there, but it will not be explained in the manual.

```

519- # Estimation -----
520 #In case optimal parameters want to be used, change the line 524 to control =bestModel$control
521 ct = rpart(Clusters ~ .,
522   data = dataParticipation,
523   method = 'class',
524   control = rpart.control(minsplit=30, minbucket=15, cp=0.001))
525 plotcp(ct)

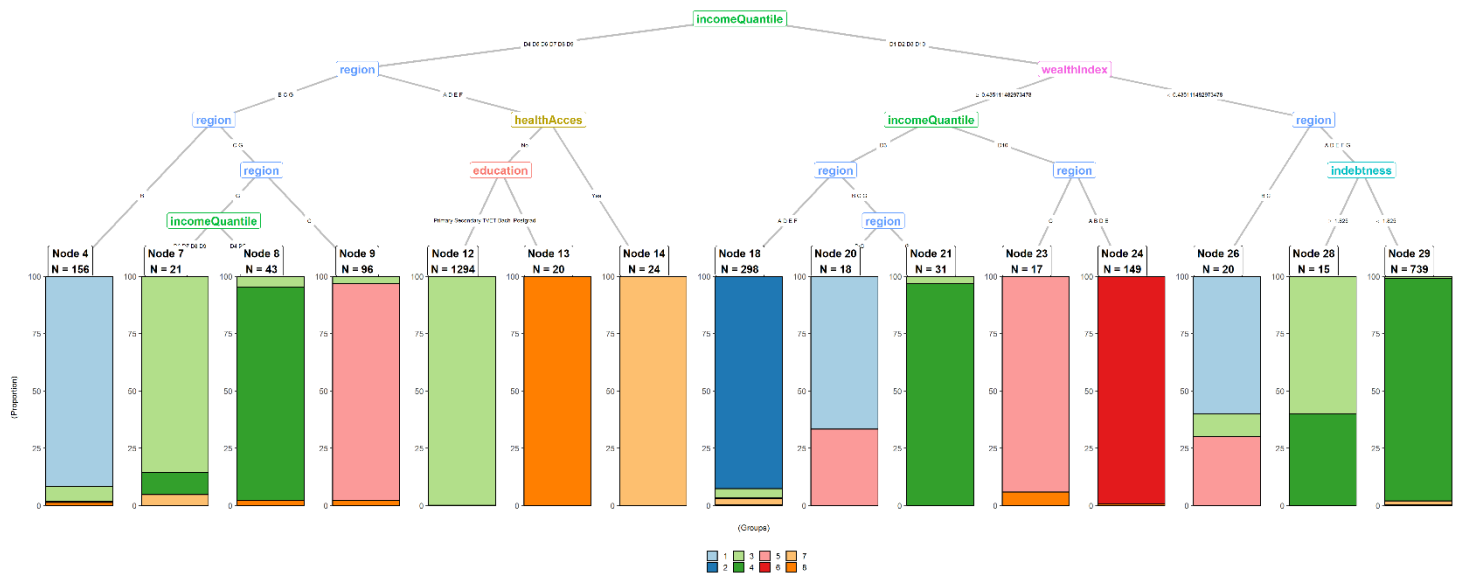
```

In contrast, from line 521 onwards, the manual tree is created with the function *rpart()*. The first part “Clusters ~ .,” tells the programme to identify the best way to define the clusters based on all the variables. However, if you want to do the clusters for specific variables, replace the “.” with the list of

variables separated by a “+”. For example, “Clusters ~ age+region,” only use age and region to predict the clusters. The following line defines the data, and is followed by the method. As it is explained in chapter 8, CART can also be used for continuous variables, so, here in “class” you specify that you are using categorical variables. Finally, in the control line, you specify three elements, where the relevant ones for this manual are *minsplit* and *minbucket*. The first one tells the code about the minimum number of elements in a group in order for the tree to check if it is worthy to break it more. The second one specifies the minimum size of and end node (or leaf) of a tree, so in this case you make sure that every group has at least 15 observations.

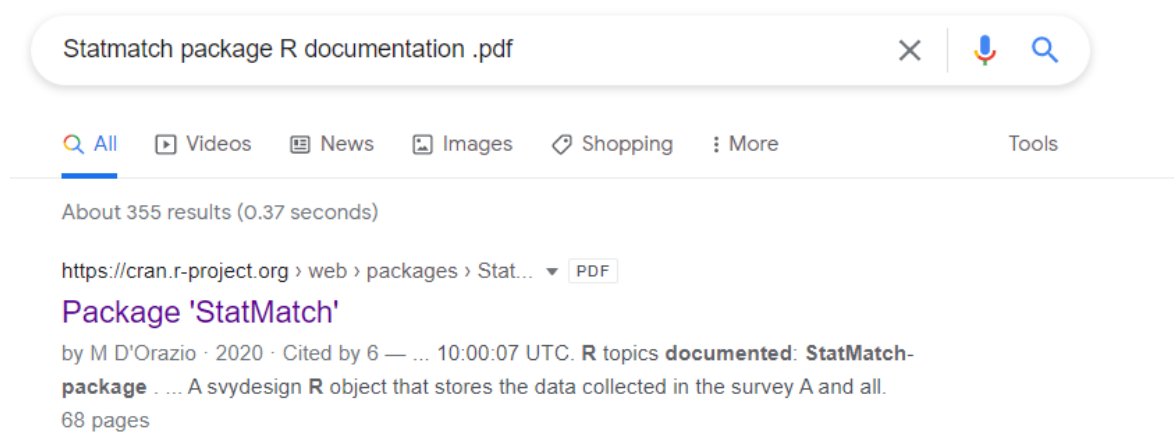
The remaining lines are the graphical parameters and in this case you will appreciate the potential of ggplot at its full. Notice that the graph is a mixture of a tree, a set of bar plots, several colourful labels with internal information and some texts between the arrows. While the details are not relevant for this manual, the reader is encouraged to analyse the full structure as it expands his skills when working with graphs in R.

As explained in the conceptual section the tree begins to split the individuals by the relevant variables and then, for each group it flags what are the main characteristics describing it. Hence, for example, group 8 (orange) are people from higher deciles who are mostly from regions ADEF, and who have postgrads but do not have access to health. In contrast group 6 (red) are the high income individuals (highest decile) from all regions except C. On the one hand side, this allows the creation of quick profiling guidelines, but on the other hand this tool also raises important policy questions, for example, why C is excluded? Therefore, and concluding the chapter, it is possible to understand now how relatively simple techniques, once they are standardized, can provide rapid insights on the type of beneficiaries of the social assistance programmes and by developing these accurate profiling tools, policy-makers can identify the specific needs of these groups and how to target assertive strategies to improve the livelihood of the people.



## Final remarks

In this chapter, as well as in the coming chapters, several packages have been used that haven't been mentioned before. All these packages have been specified in the master file and that's why they do not need to be specified again. For avid readers, it is recommended to check the packages in the master file and put them in Google as in this example for "StatMatch".



The screenshot shows a Google search interface. The search bar contains the text "Statmatch package R documentation .pdf". Below the search bar, there are navigation options: "All", "Videos", "News", "Images", "Shopping", "More", and "Tools". The search results show "About 355 results (0.37 seconds)". The first result is a PDF document titled "Package 'StatMatch'" from the URL "https://cran.r-project.org/web/packages/StatMatch/PDF". The document is by M D'Orazio, dated 2020, and is cited by 6 sources. The description states: "A svydesign R object that stores the data collected in the survey A and all." The document is 68 pages long.

With very few exceptions R packages come with lengthy documentations files, all following a similar format to the one below, and they come with conceptual and technical explanations, as well as with examples that can help you to improve your skills.

## Package 'StatMatch'

July 16, 2020

**Version** 1.4.0

**Title** Statistical Matching or Data Fusion

**Author** Marcello D'Orazio

**Maintainer** Marcello D'Orazio <mdo.statmatch@gmail.com>

**Depends** R (>= 2.7.0), proxy, survey, lpSolve, ggplot2

**Suggests** Hmisc, MASS, mipfp, R.rsp, clue, RANN,

**Description** Integration of two data sources referred to the same target population which share a number of variables. Some functions can also be used to impute missing values in data sets through hot deck imputation methods. Methods to perform statistical matching when dealing with data from complex sample surveys are available too.

**License** GPL (>= 2)

**VignetteBuilder** R.rsp

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2020-07-16 10:00:07 UTC

## Chapter 7: Targeting characteristics

Once the profiling of beneficiaries is concluded, the next step in the SPP-RAF is the evaluation of the criteria used to select who participates in the program and who does not.

### Purpose of targeting characteristics

The second stage of the framework aims to identify the relevance of the selection criteria of variables that are related to the profiled groups of the previous stage of beneficiaries. During this stage, the scores that beneficiaries obtained in each selection criteria are reviewed to understand their capacity to influence the total score and selection or non-selection of beneficiaries.

#### Output:

- Set of measurements that ranks the selected variables according to their relevance among the current beneficiaries and their impact on the score of eligibility/non-eligibility of applicants.
- Creation of contrasts between filter weights and variable scores to evaluate the match between the intended beneficiaries and the actual beneficiaries.

#### Outcome:

- A set of graphical and numeric indicators that provide guidance to policymakers and enable the targeting methodology to be continuously updated and improved.

#### Data requirements:

This stage requires the creation of a dataset that includes all the households that are beneficiaries of a given programme during a specific period. For each household the following variables are required:

- For the beneficiaries, all the variables that have been used as part of the selection criteria.
- For the programme that will be evaluated, it is required to know the weights used to score individuals.

### Targeting strategy

Social assistance programmes have limited resources and with very few exceptional cases, the amount of resources is less than the population that needs the support. For this reason, governments need to develop selection criteria to choose only those that will be benefitted the most by the programmes. However, one thing is the conceptual design of the strategy and another thing is its realization. For example, consider a programme that requires people to have a mobile phone and in the region everyone has one. Then, there isn't really a prioritization going on. Or on the other hand, consider a programme that gives equal weight to families with kids, and families with members that go to primary school. As these variables are highly related, the common topic (kids in schools) might be weighted more than other variables that can also be relevant for choosing beneficiaries. For this reason, this chapter develops two complementary techniques to understand how relevant the current decision criteria for the selection of beneficiaries are.

#### Variance decomposition technique

While ideally the policymakers have specified how much each variable weights on the decision criteria, once the theory faces the data, there is no specific order. For example, consider a scenario where two variables, X and Y are used to develop a score that defines if an individual participates in a programme or not. To avoid complications, the formula is:

$$score = 1 \times X + 1 \times Y$$

Now, while naively it seems that X and Y are equally represented, let's consider the variances of the variables. Let's suppose that  $Var(X) = 1$ , while  $Var(Y) = 4$ , meaning that individuals have very similar answers on X and very different answers on Y. Assuming X and Y are independent and using basic statistical principles, the variance of the score (how spread the score values are) is 5. Therefore, 80% of the variation of the score is explained by Y and only 20% is explained by X. Placing this information into context, what if X is very difficult (and costly to obtain). In that case is the 20% of variability worthy?

In this case, the technique used by this manual is based on semi-partial correlations (Kim, 2015). Briefly described in figure 4, consider that variable Z is explained by X and Y, but X has a little part that is explained by Y as well. So, to understand better what is the part that X explains of Z without mixing with Y, first the effect of Y on X needs to be removed, and then only on the residual variation of X, it is possible to see how related it is to the effect on Z. Some people refer to it as a decomposition of  $R^2$ , however technically is not because depending on the correlation structure of the variables, the sum of these semi-partial correlations does not need to add up to the  $R^2$ . Yet the  $R^2$  is a useful benchmark (after squaring the semi-partial correlation) and the values are usually calculated with respect to it. To see a different presentation of this topic, the link <https://www.statisticshowto.com/partial-correlation/> provides a quite illustrative case.

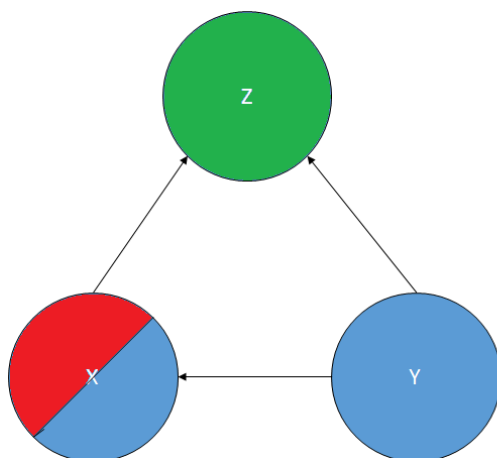


Figure 4 Semi-partial correlation

Concluding from this technique, as you will see in the last section, the software will provide a guidance on the relative importance of each variable for the selection of beneficiaries and this will guide the policymaker on the relevancy of these variables.

### Lasso regression

The previous section focussed on the explanation power of the variables, but in a similar perspective, a different study can be done over the variable coefficients. Let's say that the equation to define the score is now:

$$score = 0.1 \times X + 10 \times Y$$

While visually the weight of 0.1 is significantly smaller than 10, however, is it small enough to be able neglected? To solve this problem, you can use a nice statistical method that is known as the Lasso regression (Tibshirani, 1996). To explain this concept consider a typical regression with two variables and an error term.

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$$

And the linear regression exercise is to calculate the values of  $\beta_0$ ,  $\beta_1$ , and  $\beta_2$  that minimize the sum of squared errors:

$$\min_{\beta_0, \beta_1, \text{and } \beta_2} (Y - \beta_0 - \beta_1 X_1 - \beta_2 X_2)^2$$

Now, assume that having values of  $\beta$  different than 0 will cost. So you also want to include that in the minimization exercise:

$$\min_{\beta_0, \beta_1, \text{and } \beta_2} (Y - \beta_0 - \beta_1 X_1 - \beta_2 X_2)^2 + \lambda(|\beta_1| + |\beta_2|)$$

Hence, if  $\lambda = 0$ , this will be a typical ordinary least square; in contrast, if  $\lambda = \infty$ , all  $\beta$  will be 0 due to the high costs associated in having them. Thus, by increasing  $\lambda$  slowly, the method begins to tell what the most important variables are, and there are even some technical criteria, to explain what is the best  $\lambda$  that explain the Y by using the least number of variables. Therefore, this exercise not only allows to remove those variables that are less meaningful, but also readjust the weights so that with fewer variables, the scores can still be properly forecasted. Recall the example of the programme that gives equal weights to families with kids, and families with members that go to primary school. The probable result by using the lasso is to remove one of these variables and duplicate the coefficient of the second one to show that in reality, that element has a significantly higher weight.

### Targeting analysis

The code, named Chapter 02, begins similar to Chapter 00 and Chapter 01 by uploading the data, the dictionaries, and the relevant format topics. Also, notice that similar to the exercises in Chapter 02, there is also a need to create the relevant dummy variables so that the exercises can be produced. Still there is a very important line to notice

```
38 dataParticipation$score=dataParticipation$score+rnorm(n = length(dataParticipation$score), mean = 0, sd = 0.01)
```

In contrast to a linear regression where there is always an error variable, in this exercise, in theory, the score should be a direct calculation of the variables used for the targeting. But realistically, there might be reporting errors, especially if some questions depend on the perspective of the beneficiary candidate. Thus, a small error term is added to the score reflecting that there may be some noise in the data.

Once this is done, the semi-partial correlations are calculated.

```
42 # '2_1' Influence plot -----
43 rowLine = 1
44 jumpOfRows = 20
45 colPlots = 1
46 colTables = 12
47
48 indicatorCode <- '2_1'
49 part <- ''
50 labCodes <- dataPlots %>% subset(Code == paste0(indicatorCode, part))
51 newSheet <- addWorksheet(wb, sheetName = indicatorCode)
52
53 # Data:
54 sp = spcor(dataParticipation, method = c('pearson')) # Semi-partial correlation.
55
56 # Weights of each variable on the R2:
57 weights = sp$estimate^2
58 weights = data.frame(Values = weights[dim(weights)[1], -dim(weights)[2]])
59 weights$names = dataParticipation %>% dplyr::select(-c('score')) %>% names() %>% .[1:nrow(weights)]
60 colnames(weights) = c('Values', 'Variable')
61 #Factors
62 weights$Variable = factor(weights$Variable,
63                           levels = influence[,2],
64                           labels = influence[,4-language])
65 R2 = summary(lm(score~., dataParticipation))$r.squared
```



In this part of the code there are several relevant lines to highlight:

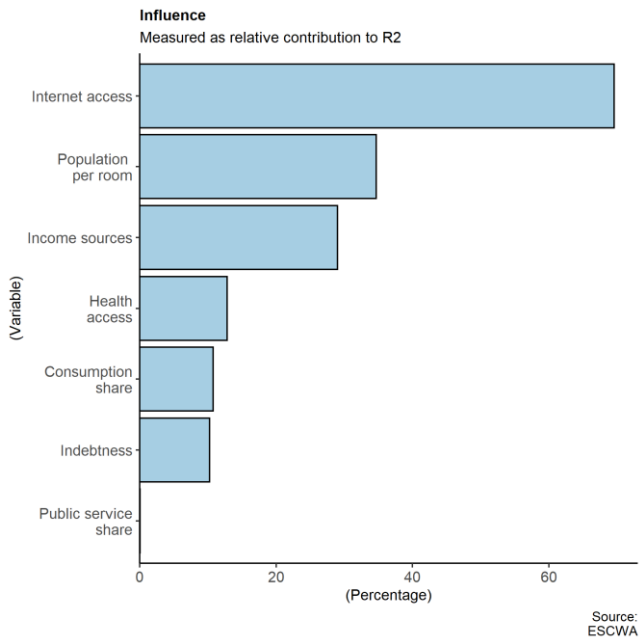
- Line 54 calculates the semi partial correlations among all variables, but only lines 58 and 59 select the relevant lines of the matrix.
- Line 65 calculates a linear regression, and the reader will identify a similar pattern as the one used in `rpart()` for the decision tree. This is done to obtain later the  $R^2$ .

```

73 newPlot = weights %>%
74   mutate(Values = Values/R2,
75          Variable = fct_reorder(Variable, Values)) %>%
76   ggplot(aes(x = Variable, y = Values, fill = factor(1)))+
77   geom_bar(colour = 'black', stat = 'identity', position = position_dodge(), show.legend = F) +
78   labs(title = paste0(as.character(labCodes[, 5*(1-language)+2])),
79        subtitle = as.character(labCodes[, 5*(1-language)+3]),
80        caption = as.character(labCodes[, 5*(1-language)+6]),
81        x = as.character(labCodes[, 5*(1-language)+4]),
82        y = as.character(labCodes[, 5*(1-language)+5])) +
83   scale_y_continuous(expand = expansion(mult = c(0,.05)),
84                     labels = scales::percent_format(scale = 100, suffix = ''))+
85   scale_fill_brewer(palette = as.character(labCodes[, 12])) +
86   coord_flip() +
87   scale_x_discrete(position = if (language == 0) {'top'} else {'bottom'}) +
88   theme_classic() +
89   theme(legend.position = 'bottom',
90         text = element_text(family = 'Georgia'),
91         axis.text.x = element_text(size = scale_factor * 10, family = 'Georgia'),
92         axis.text.y = element_text(size = scale_factor * 10, family = 'Georgia'),
93         axis.title.x = element_text(hjust = 0.5, size = scale_factor * 10, family = 'Georgia'),
94         axis.title.y = element_text(hjust = 0.5, size = scale_factor * 10, family = 'Georgia'),
95         legend.text = element_text(size = scale_factor * 10, family = 'Georgia'),
96         strip.text = element_text(size = scale_factor * 10, family = 'Georgia'),
97         plot.title = element_text(face = 'bold', size = 10, family = 'Georgia', hjust = if (language == 0) {1} else {0}),
98         plot.subtitle = element_text(size = 10, family = 'Georgia', hjust = if (language == 0) {1} else {0}),
99         legend.key.size = unit(0.5 * scale_factor, 'cm'))
100
101 # Table:
102 newData = weights %>% mutate(Values = (Values/R2)*100) %>% arrange(Values)

```

Finally the graph is a standard bar graph, but notice that in lines 73 of the graph, and 102 of the table, the data has been divided by the  $R^2$  so it can be a benchmark to it.



Based on the previous analysis, it is possible to see that with the exception of the public service share, all the variables have some degree of relevance, where the most relevant variables are internet access, population per room, and income sources.

```

114 # Redundant variables -----
115 dataParticipationT=as.data.frame(scale(dataParticipation))
116 X <- model.matrix(score~.,dataParticipationT)
117 y <- dataParticipation$score
118
119
120 lb <- rep(-Inf,length(colnames(X)))
121 ub <- rep(Inf,length(colnames(X)))
122
123 # Estimation of the optimum lambda:
124 cv_las1 = cv.glmnet(x = X, y = y,
125                   lower.limits = lb,
126                   upper.limits = ub)
127 lambda = cv_las1$lambda.min
128
129 # Lasso regression with the calculated value:
130 las1 = glmnet(x = X, y = y,
131             lower.limits = lb,
132             upper.limits = ub,
133             lambda      = lambda)
134
135 # Delete variables which don't contribute:
136 c.fit1      = coef(las1) %>% as.matrix() %>% as.data.frame()
137 colnames(c.fit1) = c('Coefficient')
138 namesVar     = rownames(c.fit1)
139 c.fit1$Coefficient[abs(c.fit1$Coefficient)<0.05]=0
140
141 RMSE=sum((y-predict(las1, newx = X))^2/(length(y)-(dim(X)[2]-1)))
142 result=as.data.frame(c(RMSE, c.fit1$Coefficient))
143 namesChosen=c("RMSE",rownames(c.fit1))
144 result$Variable=namesChosen
145 result$Lambda=lambda
146 colnames(result)[1]="Values"
147 result$Variable[2]="Intercept"
148 result=result[!result$Variable%in%result$Variable[3],]

```

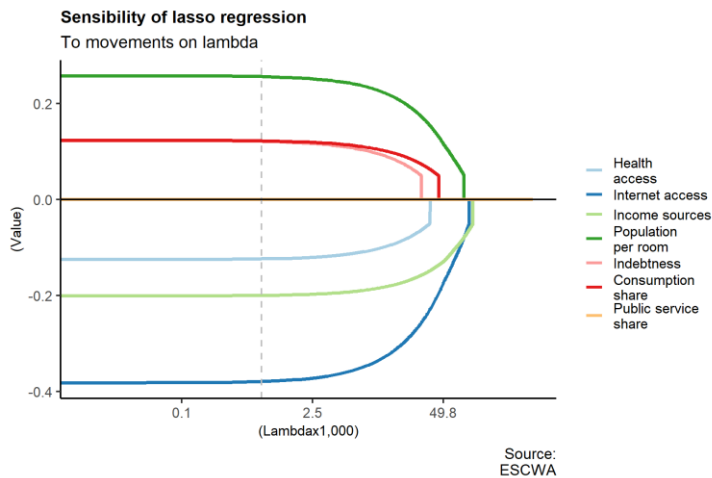
The code linked with the Lasso is slightly complicated. Lines 115-117 are stating the model and lines 120-121 regulate the scope of the search of  $\beta$ . Currently there is no limitation, but if, for example, it is well known that the weights have to be positive, then line 120 can be modified accordingly. The next block calculates the value of the optimal  $\lambda$  (lines 123-127) and uses it to identify what variables can be removed from the equation.

```

157 # Estimation -----
158 # Lasso regression with the calculated for values around the optimum:
159 lMin=0.01*lambda
160 lMax=500*lambda
161
162 lambdaList=seq(from=lMin, to=lMax, length.out=1000)
163 for(i in lambdaList){
164 las1 = glmnet(x = X, y = y,
165             lower.limits = lb,
166             upper.limits = ub,
167             lambda      = i)
168
169 # Delete variables which don't contribute:
170 c.fit1      = coef(las1) %>% as.matrix() %>% as.data.frame()
171 colnames(c.fit1) = c('Coefficient')
172 namesVar     = rownames(c.fit1)
173 c.fit1$Coefficient[abs(c.fit1$Coefficient)<0.05]=0
174
175 RMSE=sum((y-predict(las1, newx = X))^2/(length(y)-(dim(X)[2]-1)))
176 resultT=as.data.frame(c(RMSE, c.fit1$Coefficient))
177 namesChosen=c("RMSE", rownames(c.fit1))
178 resultT$Variable=namesChosen
179 resultT$Lambda=i
180 colnames(resultT)[1]="Values"
181 resultT$Variable[2]="Intercept"
182 resultT=resultT[!resultT$Variable%in%resultT$Variable[3],]
183 result=rbind(result,resultT)
184 }
185
186 dataF=result
187

```

This second block of codes calculates the Lasso of other variables near the optimal  $\lambda$ , defined between lines 159 and 160. Then, the code uses a for loop where each of these  $\lambda$ s is evaluated giving some perspective of the dynamics of the coefficients around the optimal value. The rest of the codes are the graphical steps associated with the procedure, and provide no new elements to the manual.



The interesting result is the analysis plot. In this graph public services share is nulled since the beginning and only after increasing the restriction by a large factor it is possible to delete the indebtedness, health access, and consumption share variables. Moreover, the exercise shows that people with internet access, health access, and income sources have lower scores (as expected) while individuals in

overcrowded houses, with high shares of their expenditure dedicated to consumption, and high levels of indebtedness have high scores.

Then, two topics emerge. The first one is about the relevance of public services, given that it is not adding that much value, how relevant and costly is to obtain this last variable. The second one is to check if these weights are aligned with the conceptualization of the programme, and for this purpose, it is important to understand what formula was used to identify beneficiaries.

**Beneficiaries' formula:**

According to the kingdom of AP. The formula to choose beneficiaries is:

$$score = \frac{popRoom}{7} + \frac{6 - incomeSource}{5} + \frac{3 \times consumptionShare}{2} + \frac{1 \times publicServiceShare}{2} + \frac{indebtness}{4} + (1 - healthAcces) + (1 - internetAcces)$$

Finally, individuals with scores higher than 3.5 participate in the program while individuals with lower scores do not participate.

Notice that the indebtedness weight is 0.25 while the share of consumption is 1.5. However, the Lasso is flashing that both have relatively similar weights around 0.1 (both are lower than expected, yet, for consumption the reduction is more dramatic). Hence, by performing these types of analysis the policy makers can better understand the weights that are being applied for the screening and modify the programme accordingly.



## Chapter 8: Coverage evaluation

The final chapter of this manual covers the third step of the SPP-RAF, as it has been explained, the fourth step has higher data and technical requirements and therefore it makes the current manual way extensive. Still in the coming section the requirement of these two stages are presented so that the reader can have in mind the integrity of the framework.

### Purpose of the coverage evaluation

The third stage of the framework aims to identify inclusion and exclusion errors that the programmes are generating. On the one hand, this stage of the analysis reviews the characteristics of the current beneficiaries to flag those whose situation might have changed and might be removed from the programme or moved into a different programme. On the other hand, it identifies individuals that might potentially meet the criteria to participate in the programmes (or are likely to meet them soon) but are currently not enrolled. By doing so, the framework allows policy-makers to make decisions that maximize the efficient use of resources, and determines whether outreach plans need to be deployed in order to make enrolment efforts more effective.

#### Output:

- Reveal indicators that might help to identify potential beneficiaries that have been omitted by the programme in any specific geographical area as well as the likelihood of individuals to belong to a group - i.e. beneficiaries or non-beneficiaries - other than those they are assigned to.

#### Outcome:

- Reducing inclusion and exclusion errors.
- Identification of geographical areas where the implementation of programmes may be prioritized in order to reduce exclusion errors.

#### Data requirements:

This stage requires the creation of a dataset containing the same set of variables as stage one, but extending its population also to individuals that are currently not part of the present programme.

It is expected that data for non-beneficiaries will be quite limited. To control for this challenge, flexible machine learning techniques will be developed to be robust to missing variables. Nevertheless, the success of the analysis strongly depends on the capacity to acquire the largest possible set of variables for both beneficiaries and non-beneficiaries.

### Purpose of the beneficiary evaluation

The fourth stage of the framework aims to measure the impact of the social assistance programme on its beneficiaries and to identify those characteristics that allow individuals to overcome their vulnerable conditions and successfully graduate from the programme. It is therefore important to understand if the programmes generate the intended consequences on the livelihoods of the beneficiaries, if there are individual characteristics that support the success of individuals in their personal lives (and should be encouraged), and if there are characteristics related to undesirable perverse incentives that require policy discussions on the design and validity of the programmes.

#### Output:

- Quantify the evolution of social assistance indicators over time and across different policy areas.

- Measure the possible perverse incentives and behavioural changes among beneficiaries affecting their likelihood to graduate from the programme.

**Outcome:**

- Creation of a rapid-assessment mechanism to measure the impact of social assistance measures.
- Identification of the characteristics and actions that improve the livelihood of the individual to a point that there is no need for them to belong to the assistant programmes.

**Data requirements:**

- This stage requires the creation of a dataset containing the same set of variables as at the stage one, but extending it across different periods. Therefore, the same individual should be followed with a given frequency (e.g. monthly) to evaluate changes strategic programme characteristics (in particular targeting variables, or as many as possible).
- Individual follow-ups should go beyond the individual participation in the programme and their variables should be monitored for several periods after they stop participating in the programmes.

**The missing data challenge**

For the third chapter of the report, there is a need to contrast data from the beneficiaries and contrast it with the data from the non-beneficiaries. Notice that, if all the data relevant for the selection of beneficiaries was collected from every person in the country, there shouldn't be any misallocation of individuals (except for wrong values that will mislead the allocation, topic covered in the next section). However, this is usually not the case. Data collection is costly and therefore only beneficiaries are expected to have all the relevant variables (and it is important to highlight that it is expected that the beneficiaries have all the variables, otherwise how would the system calculate their score?). Hence many unfortunate realities begin to emerge. For example, what if the news of a social programme couldn't reach a population of difficult access? In this case, these people that really need the programme will not know about it and will not register; still the government will not know about this because not all of their variables are in the system. Similarly, what if the individual, due to poverty is too busy working for the living and is unable to go to registry, or what if there is a peer pressure effect where some members of the society are discriminated if they access the government programmes? For all these reasons, it is important to create a technique that will guide us on the degree of people that are not being covered. The traditional way to do this is by surveys done in specific areas that ask the individuals all the relevant variables and are able to directly assess how many people that match the requirements are currently covered. Unfortunately surveys are costly, take time, and are localized (so if they are done in the municipality A, but municipality B was the one with the problem, no one will know). Still, as a spark of hope, now a days is very difficult to be completely out of the system (and for those that are out of the system, the government will probably have a good idea about it). So there are some administrative records available for everyone. Hence, the methodology displayed in this chapter will use this data to estimate the missing values and calculate the probable score of the individual. Of course, is not an accurate measurement to declare that someone has been misallocated, but it gives a probabilistic idea about which geographical areas, socioeconomic sectors, etc., are being underrepresented and design targeted outreach strategies.

Naturally, the relationship between the variable can be highly complex and therefore having a uniform approach, like a linear regression will be insufficient. On the other extreme, if the number of relevant variables being used for the selection of beneficiaries is high, doing a tailored model for each case is an exhausting and tedious quest, that needs to be regularly updating and it would be difficult to

guarantee the use of the right model. Fortunately, machine learning algorithms had advanced, and similar to the example developed on Chapter 6 with the CART, there are other algorithms that independently can find the best predictor. These methods are not usually used for data analysis because, tracing the linkages between variables is not as simple as in a regression or as in CART, however, for prediction purposes they perform quite well. In this particular case, the manual proposes an algorithm called Gradient Boosting (Hastie, Tibshirani, & Friedman, 2009). This algorithm extend the idea of the decision tree and makes several of them in parallel, compare their results, and create a bigger tree with better optimization results. The algorithm will not be explained further as the added value of the explanation, for this manual purposes is minimal. Yet, for keen learners, one recommended introduction can be found at <https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/>. Now, the reason why the decision tree was explained is because it presentation allow us to understand better ways to design rapid profiling guidelines. In contrast, for this section, the only value of the algorithm is to know that it doesn't need expert knowledge to give a good prediction of a variable, and this allows us to automatize the process relatively fast. One final advantage of this method, that was also observed for the decision tree, is that in cases of limited information it will still produce an informed guess of the value of a variable, which allows us to implement these process even in scenarios with low data resources available.

Having done those remarks, the process in this point is relatively straight-forward. Consider for example, the illustration case of the kingdom of AP. In this case, there are seven criteria for the beneficiaries, and some of them, for example income sources is only available for beneficiaries. Hence, the first step is to develop a gradient boosting algorithm to predict income sources based on people per room, consumption share, and the other four variables. Then, we use that model to predict the values for income sources for beneficiaries and then replace change the missing value for this predicted value, as it is shown in the following diagram.

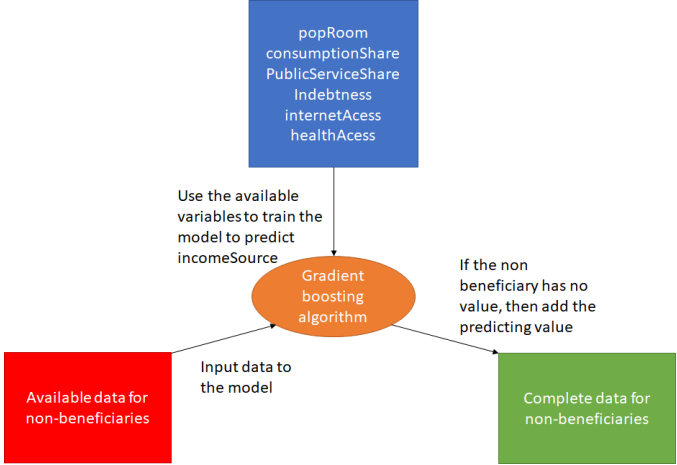


Figure 5 Prediction process for missing data

After the previous process has been performed, all the individuals, both beneficiaries and non-beneficiaries will have a score and using the criteria of who can be eligible, it is possible to identify individuals that have the potential to be in the programme but are not linked to it. Thus, by crossing this new variable with other variables, the policy maker will have some guidance on areas where exclusion errors can be taking place.

**The outlier challenge**

The final analysis is associated with inclusion errors. Unfortunately, when the information that a person provides is linked to the chance of receiving some benefits, there exist perverse incentives for

individuals to alter their information. Moreover, while the previous challenge (missing data) can be solved very accurately by doing surveys, this other problem can't be directly solved because if the individual links the survey with the possibility of losing benefits, or earning more, the survey might also suffer from a bias. However, the previous idea of completing data can be restructured to identify individuals where the pattern of answers is not common and flag more detailed studies. This same idea can also help to identify the variables where these outliers are more common.

Consider again, the illustration case of the kingdom of AP. Due to the previous step there is already a model that predicts income sources based on all the other variables. Now instead of predicting this value for those who have it missing, we are going to use the model to predict the value for everyone. Now, for the beneficiaries, we have the information of the data that they provide and the data that they were expecting to provide. Using this information, we can now identify if they were not expected to participate and in those cases pay more attention. We will also analyse for each variable how different is the model prediction with the real value of the data and understand what variables are having clear prediction patterns and can facilitate inspection, and which ones have less predictability and should be reviewed with care as they can be used to affect the system.

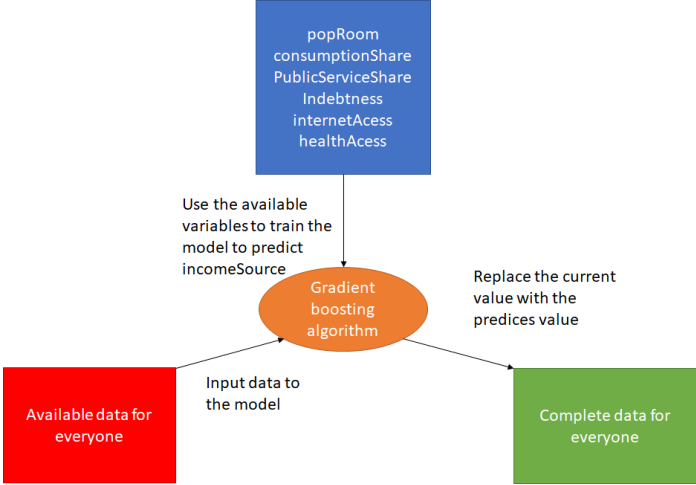


Figure 6 Identification of outliers

Coverage analysis

The code relevant for this section is under the name Chapter 03. Similar to the previous cases, the first 55 lines are uploading data and dictionaries.



```

56 ppopRoom <- gbm(formula = popRoom ~ .,
57                 data = dataMini,
58                 n.trees = 500,
59                 cv.folds = 5,
60                 n.cores = 1)
61
62 ntree_opt_oob <- gbm.perf(ppopRoom, method = "OOB", oobag.curve = TRUE) # n.trees l
63 ntree_opt_cv <- gbm.perf(ppopRoom, method = "cv") # n.trees by cross-validation.
64
65 ppopRoom <- gbm(formula = popRoom ~ .,
66                 data = dataMini,
67                 n.trees = ntree_opt_oob,
68                 cv.folds = ntree_opt_cv,
69                 n.cores = 1)
70 pincomeSources <- rpart(formula = factor(incomeSources) ~ .,
71                         data = dataMini,
72                         method="class")
73 pconsumptionShare <- gbm(formula = consumptionShare ~ .,
74                           data = dataMini,
75                           n.trees = ntree_opt_oob,
76                           cv.folds = ntree_opt_cv,
77                           n.cores = 1)
78 ppublicServiceShare <- gbm(formula = publicServiceShare ~ .,
79                             data = dataMini,
80                             n.trees = ntree_opt_oob,
81                             cv.folds = ntree_opt_cv,
82                             n.cores = 1)
83 pindebtiness <- gbm(formula = indebtiness ~ .,
84                     data = dataMini%>%filter(indebtiness>0)%>%
85                     mutate(indebtiness=log(indebtiness)),
86                     n.trees = ntree_opt_oob,
87                     cv.folds = ntree_opt_cv,
88                     n.cores = 1)
89 pinternetAcces <- rpart(formula = factor(internetAcces_1) ~ .,
90                         data = dataMini,
91                         method="class")
92 phealthAcces <- rpart(formula = factor(healthAcces_1) ~ .,
93                      data = dataMini,
94                      method="class")
95

```

Then, from lines 56 to 94 each of the variables is ran in a gradient boost algorithm (identified with the function `gbm()`) against all the other variables (“~ .”) to create these prediction models. Line 62 and 63 provide some optimal parameters for the prediction according to R programs. Yet, similar to the case of the trees, if the user has a preferred parameter, it is possible to place it manually. Then in each prediction model, there are other parameters associated with the number of decisions trees it will produce, the number of cross-validation folds, and the type. In particular, as income sources, health access, and internet access are categories, the method is specified so that the tree model them accordingly. Finally, the variable `indebtiness` provides an example of variable transformation. Sometimes, the predicted model works better when the variable is in logarithm or exponential forms (the topic is not covered in this manual but for the interested reader, the following link is a good starting point

[http://sciences.usca.edu/biology/zelmer/305/trans/#:~:text=For%20regression%2C%20it%20is%20the,and%20meet%20the%20linearity%20assumption.&text=If%20transformation%20succeeds%20in%20producing,the%20dependent%20variable%20\(Y\)](http://sciences.usca.edu/biology/zelmer/305/trans/#:~:text=For%20regression%2C%20it%20is%20the,and%20meet%20the%20linearity%20assumption.&text=If%20transformation%20succeeds%20in%20producing,the%20dependent%20variable%20(Y)) ). As it can be seen in line 84 and 85 the data can be transformed at this stage so that the prediction model can include it.

```

96 # Prediction of the missing values:
97 temp$participant='No'
98 temp$participant[!is.na(temp$popRoom)]= 'Yes'
99 temp$participant=factor(temp$participant)
100 temp2=temp
101 temp$popRoom[is.na(temp2$popRoom)] = predict(ppopRoom, newdata = temp2%>%filter(participant=='No'), n.trees = ntree_opt_oob)
102 temp$incomeSources[is.na(temp2$incomeSources)] =
103   as.numeric(as.character(predict(pincomeSources, newdata = temp2%>%filter(participant=='No'), "class")))

```

The following lines create a variable to record who was a participant and who wasn't and then it identify those individuals with missing values in popRoom and incomeSources (which are the non-beneficiaries) and use the models to predict them. For this process, the key function is *predict()* which takes as input the model and the data, and use it to predict the value that the observation should have.

```

105 # 3.1| Calculation of score -----
106
107 temp=temp%>%mutate(adjpopRoom=popRoom/7)%>%
108   mutate(adjincomeSources=(6-incomeSources)/5)%>%
109   mutate(adjconsumptionShare=consumptionShare*1.5)%>%
110   mutate(adjpublicServiceShare=publicServiceShare*0.5)%>%
111   mutate(adjindebtness=indebtness/4)%>%
112   mutate(adjinternetAcces=1-internetAcces_1)%>%
113   mutate(adjhealthAcces=1-healthAcces_1)%>%
114   mutate(score=adjpopRoom+adjincomeSources+adjconsumptionShare+adjpublicServiceShare+
115     adjindebtness+adjinternetAcces+adjhealthAcces)%>%
116   dplyr::select(-c('adjpopRoom', 'adjincomeSources', 'adjconsumptionShare',
117     'adjpublicServiceShare', 'adjindebtness', 'adjinternetAcces', 'adjhealthAcces'))%>%
118   mutate(Expected_Participant=case_when(score>3.5~ 1,
119     score<=3.5~ 0))%>%
120   mutate(Expected_Participant=factor(Expected_Participant, levels=c(0,1), labels=confirmation[,4-language]))

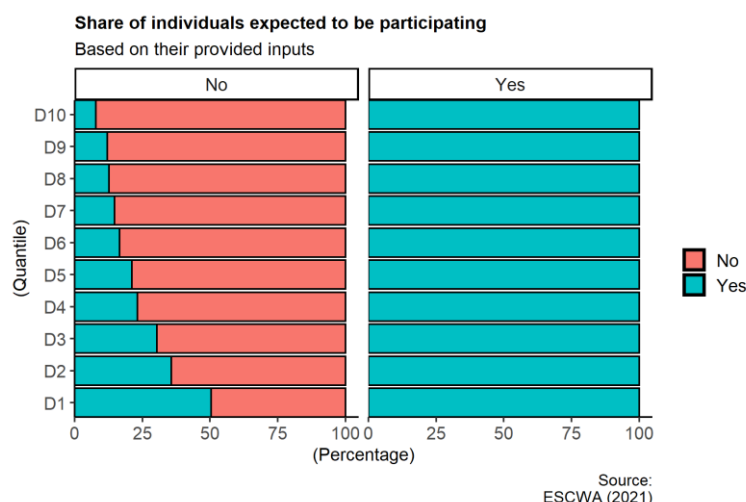
```

Once the dataset is complete the next part uses the instructions received to calculate the score of all the individuals and to identify who are expected to participate. These lines are just following up those instructions and calculating this new variable called Expected\_Participant.

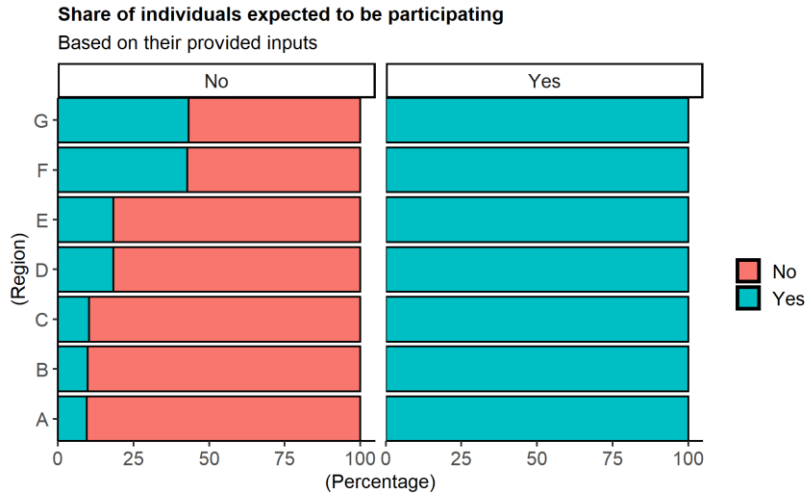
The next lines are reporting on tables and graphs on these values and code wise have been covered in previous chapters. Yet, the relevant part are their interpretation.

Expected participant	Participation	Frequency
No	No	28827
Yes	No	8232
No	Yes	0
Yes	Yes	2941

In this case, because the values of the participants were not touched, everyone that was participating still is expected to be participating (the numbers are the same). However, in the individuals not participating there is a 22% of cases that are expected to be participating but currently aren't.



This graph present the previous results by income deciles and states that for poorer deciles there probably is higher exclusion errors than for richer deciles.



Repeating the exercise by region provides an explicit result highlighting that most of the exclusion error can come from the poorest regions. Then, recalling that these places have suffered from conflict and that it has been very difficult for the government to access them, then the result is realistic. Thus, policy-makers can begin thinking of better outreach policies to target these difficult areas where people need the support and they are having struggles to get it.

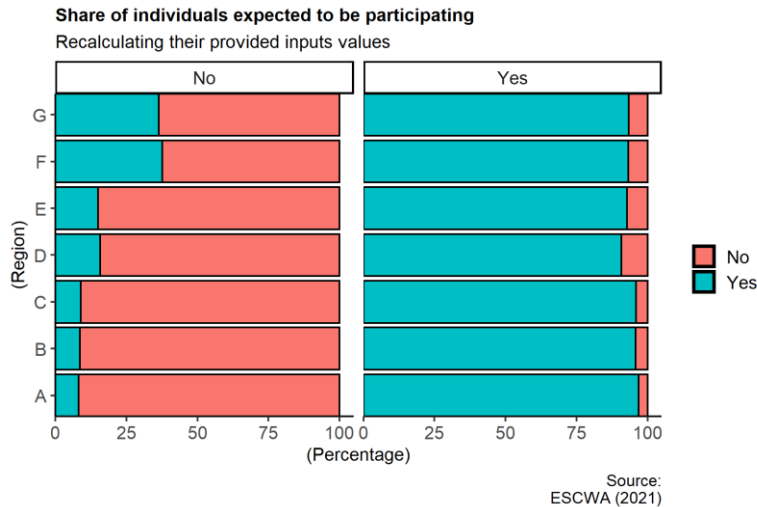
The next key part of the code is in lines 277 to 308 where the outliers are beginning to be identified.

```

277- # 4| Outlier identification -----
278
279 temp$popRoom0 = temp$popRoom
280 temp$incomeSources0 = temp$incomeSources
281 temp$consumptionShare0 = temp$consumptionShare
282 temp$publicServiceShare0 = temp$publicServiceShare
283 temp$indebtness0 = temp$indebtness
284 temp$internetAcces_10 = temp$internetAcces_1
285 temp$healthAcces_10 = temp$healthAcces_1
286 temp1=temp2
287 temp$popRoom[temp2$participant=="Yes"] = predict(ppopRoom, newdata = temp1%>%filter(participant=="Yes"), n.trees = ntree_opt_oob)
288 temp$incomeSources[temp2$participant=="Yes"] = as.numeric(as.character(predict(pincomeSources , newdata = temp1%>%filter(participant=="Yes"), "class")))
289 temp$consumptionShare[temp2$participant=="Yes"] = predict(pconsumptionShare , newdata = temp1%>%filter(participant=="Yes"), n.trees = ntree_opt_oob)
290 temp$publicServiceShare[temp2$participant=="Yes"] = predict(ppublicServiceShare, newdata = temp1%>%filter(participant=="Yes"), n.trees = ntree_opt_oob)
291 temp$indebtness[temp2$participant=="Yes"] = exp(predict(pindebttness, newdata = temp1%>%filter(participant=="Yes"), n.trees = ntree_opt_oob))
292 temp$internetAcces_1[temp2$participant=="Yes"] = as.numeric(as.character(predict(pinternetAcces , newdata = temp1%>%filter(participant=="Yes"), "class")))
293 temp$healthAcces_1[temp2$participant=="Yes"] = as.numeric(as.character(predict(phealthAcces , newdata = temp1%>%filter(participant=="Yes"), "class")))
294
295 temp=temp%>%mutate(adjpopRoom=popRoom/7)%>%
296 mutate(adjincomeSources=(6-incomeSources)/5)%>%
297 mutate(adjconsumptionShare=consumptionShare*1.5)%>%
298 mutate(adjpublicServiceShare=publicServiceShare*0.5)%>%
299 mutate(adjindebtness=indebtness/4)%>%
300 mutate(adjinternetAcces_1=internetAcces_1)%>%
301 mutate(adjhealthAcces_1=healthAcces_1)%>%
302 mutate(score=adjpopRoom+adjincomeSources+adjconsumptionShare+
303 adjpublicServiceShare+adjindebtness+adjinternetAcces+adjhealthAcces)%>%
304 dplyr::select(-c('adjpopRoom', 'adjincomeSources', 'adjconsumptionShare',
305 'adjpublicServiceShare', 'adjindebtness', 'adjinternetAcces', 'adjhealthAcces'))%>%
306 mutate(Expected_Participant=case_when(score>=3.5~ 1,
307 score<=3.5~ 0))%>%
308 mutate(Expected_Participant=Factor(Expected_Participant, levels=c(0,1), labels=c(confirmation[,4-language]))

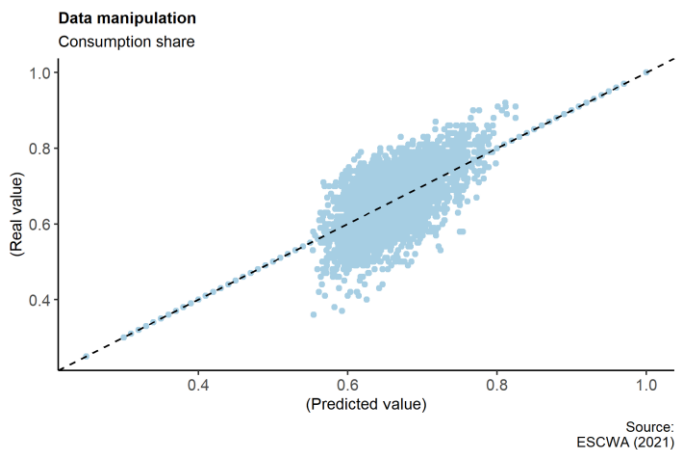
```

In these code lines, the first step is to create some temporal variables associated with the original data values. These will be used to contrast the prediction power of the model. Then, all the variables are predicted (lines 287-293) and used to predict expected participation (lines 295-38). After these lines there are some additional code lines to map the results. Notice that the update is only done for the beneficiaries. This is done because it is not expected that people that are non-beneficiaries provide incorrect information to avoid participating in the programme. Moreover, sometimes the non-beneficiary don't report anything.



In contrast to the previous graph, this technique the side of the graph linked to current beneficiaries. In this case, it is possible to observe a specific anomaly around region D where there is a larger share of individuals who are presenting atypical values and therefore, it might be prudent to understand better what is happening there.

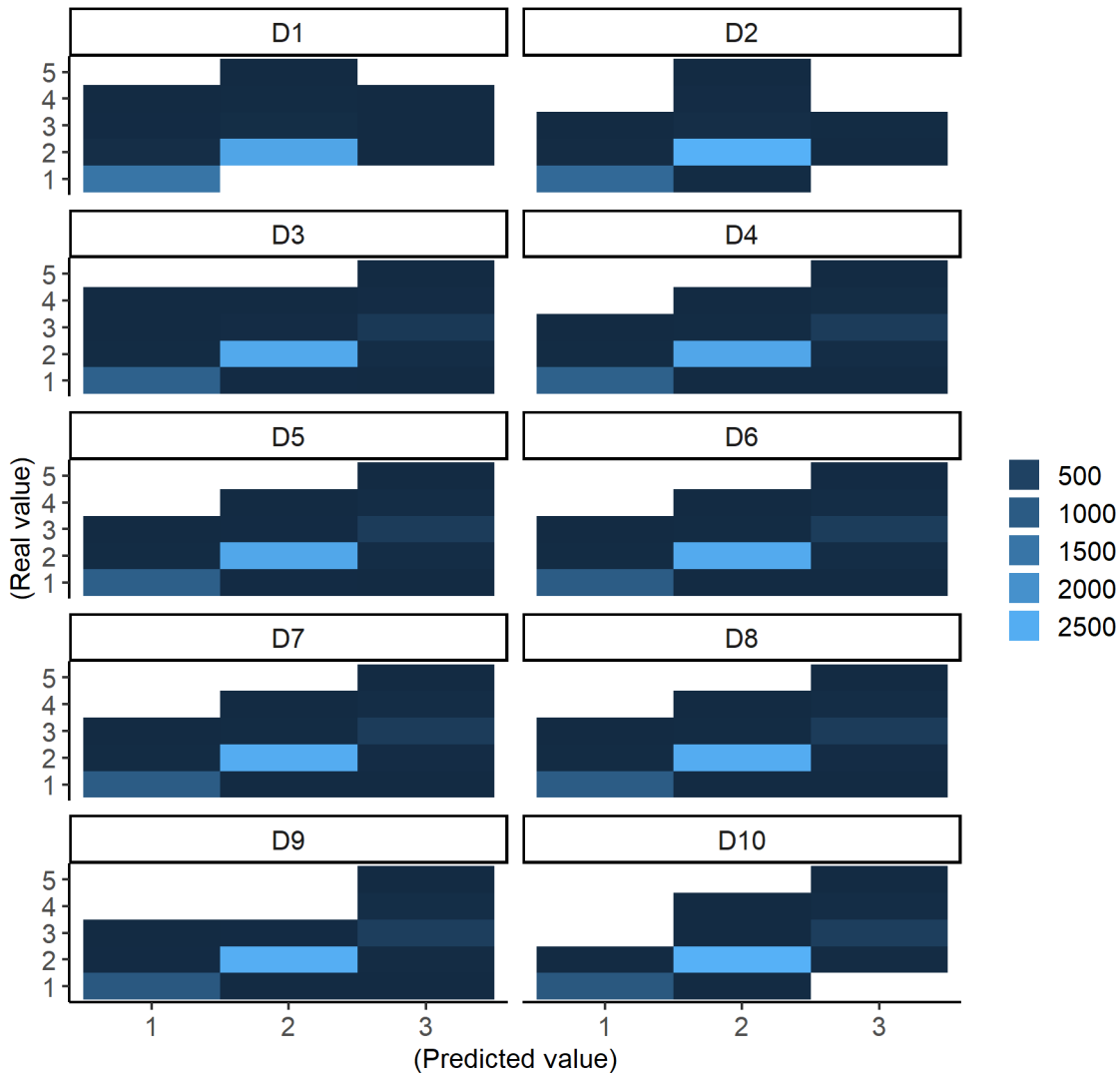
In this context, it is important to review the prediction capacity of the models.



As you can observe here, this chart is what is expected from a prediction model. The predictions are more or less aligned with the real values, but there is some natural noise around this prediction (note that the points on the diagonal are the non-beneficiaries, as their values were not replaced).

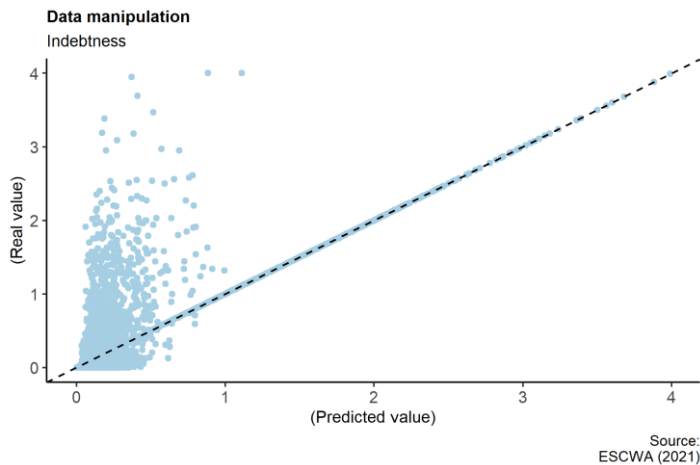
## Data manipulation

Income sources (Number of cases)



Source:  
ESCWA (2021)

This graph is used for categories. Notice that this model is presenting prediction problems because it never predicts more than three sources; all the predictions are either 1, 2, or 3. Moreover, based on the colour gradient, usually the prediction of 2 sources is quite accurate, but for 1 and three sources the prediction is not so reliable. This suggests that due to the noise within this variable, it is better to create additional validation mechanisms as it can be a source of errors.



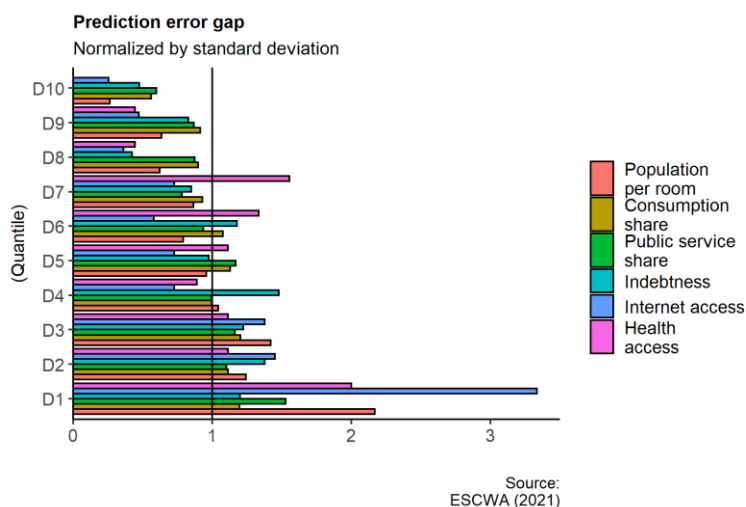
Finally, notice that indebttness clearly has an awkward trend where a significant group of individuals are expecting to have significantly lower values of the ones they claim to have. Mixing this with the idea that region D, a middle class region, was flagged as a potential issue, it is possible to develop a working hypothesis. For example, given that middle class can be in near the cut of benefits, the people around this values will alter the reporting of income sources to make sure that the calculation of the indebttness is higher and be able to access to the system. In this case it will be important for the policy regulator to implement more monitoring controls in this group, and in region D. This hypothesis is consistent with the story stated along the previous four graphs. Although it is impossible to guarantee that this is what is happening, having this hypothesis can help the policy makers to create better and more focused monitoring mechanisms that will improve the allocation of resources to the people that needs them the most. In this section some other results haven't been included in the manual, but the trainee is welcomed to review them and identify other possible hypothesis related to the coverage of the programme.

```

711 # 6| Outliers by quantile -----
712
713 temp$error_popRoom= (temp$popRoom-temp$popRoom0)^2
714 temp$error_popRoom=temp$error_popRoom/mean(temp$error_popRoom)
715
716 temp$error_consumptionShare= (temp$consumptionShare-temp$consumptionShare0)^2
717 temp$error_consumptionShare=temp$error_consumptionShare/mean(temp$error_consumptionShare)
718
719 temp$error_publicServiceShare= (temp$publicServiceShare-temp$publicServiceShare0)^2
720 temp$error_publicServiceShare=temp$error_publicServiceShare/mean(temp$error_publicServiceShare)
721
722 temp$error_indebttness= (temp$indebttness-temp$indebttness0)^2
723 temp$error_indebttness=temp$error_indebttness/mean(temp$error_indebttness)
724
725 temp$error_internetAcces_1= (temp$internetAcces_1-temp$internetAcces_10)^2
726 temp$error_internetAcces_1=temp$error_internetAcces_1/mean(temp$error_internetAcces_1)
727
728 temp$error_healthAcces_1= (temp$healthAcces_1-temp$healthAcces_10)^2
729 temp$error_healthAcces_1=temp$error_healthAcces_1/mean(temp$error_healthAcces_1)
730
731 temp=temp%>%dplyr::select(starts_with('error'), 'incomeQuantile')%>%
732   melt(id='incomeQuantile')%>%group_by(variable, incomeQuantile)%>%
733   summarise(value=mean(value))

```

To finalize the analysis, lines 711-733 measure the prediction error. In this case the error measurement is calculated as the squared error (predicted minus observed, all squared), and then it is standardized by the average error. By doing this procedure, the error can always act as a benchmark around the value of 1.



In general, there is more predictive power on high deciles than in the low deciles. This makes sense because rich individuals will tend to have more constant values of positive elements. For example, regularly have lower ratios of people per room, and lower consumption shares. In contrast, for poor individuals, the range is bigger as poverty can be manifested in many ways. However, there are two variables worthy to highlight. The first one is health, which has an atypical peak on the seventh decile, and the indebtedness with an atypical peak on the fourth decile. For the latter case, this result supports the previously stated hypothesis that there might be some dynamics going on in the middle class that can be misallocating individuals. Finally, by noticing those variables with larger errors, the policymakers can identify easier what are the variables that need the most control from part of the authorities due to their large variances, which make them difficult to validate statistically. In this case, for example, the variable associated with people per room, or internet for the poorest decile are highly random, so it is very difficult to create an algorithm that accurately guess if the individual is providing the correct information. Thus, it is necessary to implement additional controls to support the monitoring of the beneficiaries to avoid aid misallocation.

### Some notes on the beneficiary evaluation

While this topic will not be formally discussed in the current manual, this subsection provides some thoughts that guide the trainee on how to proceed. The purpose of this last step is to trace individuals across time. For that reason, the first element is to identify what are the key variables to check the evolution of the beneficiary. For example, if the social programme was focused on improving the employability condition of an individual, a natural tracing variable is the time that the individual spends in the programme. Another example are health related programmes, where the tracing variable is the health of the beneficiary.

However, there are three challenges upfront. The first challenge is that sometimes the situation improves or gets worsen regardless of the programme. For example, talking about the unemployment aid programmes, if the economy is performing better, people are expected to get the jobs faster, independent of participating in the program. Thus, the statistical strategy needs to consider the evolution of the beneficiary and contrast it with the evolution of people that are not beneficiaries (control group).

The second challenge is related to the participation of the individuals in the programme. Several programmes are go beyond income transfers and have workshops, events, and activities aiming to support the beneficiaries. Knowing that the individual participate in this activities is needed to know if the improvement can be attributed to the programme. Yet, signing an assistance list is not the same as

being engaged in these events. Thus, programmes need to develop monitoring mechanism that can quantify the engagement of individuals.

The final challenge is linked to the participation in multiple programmes. Depending on the country, individuals tend to be beneficiaries of more than a unique programme and therefore, any improvement needs to be analyse realizing that it can come from only one of these programmes, or as a synergy between them. Thus, in contrast to the previous steps, it is important to have good knowledge of the different programmes in which an individual participates.

Without doubt, the large data requirements for this step makes it impossible for many countries to implement it given the current state of their information systems. Nevertheless, it is important to note these ideas so that the system can evolve to include more and more of these variables and in that way, the analysis of programmes can constantly improve.



## Final remarks

The current manual is a training aid generated by ESCWA to support a set of capacity building activities oriented for countries that want to improve the way in which information is used to improve the efficiency and efficacy of their social assistance programs. As it has been stated along the manual, each country has its own particularities and therefore, the statistical analysis will be incomplete without a proper in-situ knowledge of the institutions, culture, environment, social practices, economy, and political context of a country. Nevertheless, the data analysis tools can be used to create a rapid assessment to guide those aspects that can create issues, challenges, and opportunities for the programmes. On that note, this manual can be seen as serving two purposes. On the one hand, the manual is a conceptual tool, and across the theoretical explanations in part 2 the policymakers can learn about useful tools, organized in a logical analytical framework (SPP-RAF). On the other the manual is an applied tool that guides the analysis on good practices for the use of the statistical software R and how it can be used to improve the automatization and professionalization of reports. Hence, by serving these two audiences, the manual, is expected to have a positive impact on the social assistance programmes of the countries, and through them, improve the livelihood of the people.

## References

- Breiman, L. (1984). Classification and regression trees. *The Wadsworth and Brooks-Cole statisticsprobability series*. Chapman & Hall.
- Gareth, J., Witten, D., Hastie, T., & Tibshirani, R. (2017). *An Introduction to Statistical Learning: with Applications in R*. New York: Springer.
- GIZ. (2017). *Vulnerability and capacity assessment with regard to social protection*. Berlin: BMZ.
- Hastie, T., Tibshirani, R., & Friedman, J. H. (2009). *The Elements of Statistical Learning (2nd ed.)*. New York: Springer.
- ILO. (2016). *Social protection assessment-based national dialogue: A global guide*. Geneva: ILO.
- Kim, S. (2015). ppcor: An R Package for a Fast Calculation to Semi-partial Correlation Coefficients. Kim, Seongho. "ppcor: An R Package for a Fast Calculation to Semi-partial Correlation Coefficients." *Communications for statistical applications and methods* vol. 22,6 (2015): 665-674. doi:10.5351/CSAM.2015.22.6.665, 665-674.
- Mahalanobis, P. C. (1936). On the generalized distance in statistics. *Proceedings of the National Institute of Sciences of India*, 49-55.
- OECD. (2019). *Monitoring and evaluating social protection systems*. Paris: OECD.
- Tibshirani, R. (1996). Regression Shrinkage and Selection via the lasso. *Journal of the Royal Statistical Society. Series B (methodological)*, 267-288.
- UNICEF. (2019). *UNICEF's Global Social Protection Programme Framework*. New York: UNICEF.
- Ward, J. H. (1963). Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association*, 236-244.
- Watanabe, S. (1969). *Knowing and guessing; a quantitative study of inference and information*. New York: Wiley.
- World Bank. (2016). *Editorial Style Guide*. Washington: World Bank.
- Zelterman, D. (2015). *Applied Multivariate Statistics with R*. New Haven: Springer.

