Introduction I - R basics

R for Stata Users

Luiza Andrade, Leonardo Viotti & Rob Marty

June 2019

Outline

1. Installation
2. Introduction
3. Getting started
4. RStudio interface
5. Data in R
6. Functions
7. R objects
8. Basic types of data
9. Advanced types of data
10. Help, Google and Stackoverflow
11. Useful resources
12. Appendix

Installation

This training requires that you have R installed in your computer:

Instructions

- Please visit (https://cran.r-project.org) and select a Comprehensive R Archive Network (CRAN) mirror close to you.
- If you're in the US, you can directly visit the mirror at Berkley university at (https://cran.cnr.berkeley.edu).
- we also strongly suggest installing R studio. You can get it in (https://www.rstudio.com/), but you need to install R first.

Outline

1. Installation
2. Introduction
3. Getting started
4. RStudio interface
5. Data in R
6. Functions
7. R objects
8. Basic types of data
9. Advanced types of data
10. Help, Google and Stackoverflow
11. Useful resources
12. Appendix

Introduction

These training sessions will offer a quick introduction to R, its amazing features and why it is so much better than Stata.

Introduction

This first session will present the basic concepts you will need to use R.

The next sessions will include:

- **Introduction to R part II**
- **Data Processing**
- **Descriptive Analysis**
- **Data Visualization**

For the most recent versions of these trainings, visit the R-training GitHub repo at https://github.com/worldbank/dime-r-training

Introduction

Some advantages of R over Stata:

- It is less specialized:
  - More flexibility when programming.
  - Many more functionalities.
- Much broader network of users:
  - More resources online, which makes using Google a lot easier. You'll never want to see Statalist again in your life.
  - Development of new features and bug fixes happens faster.
- It is way cooler.

Introduction

Some possible disadvantages of R:

- Higher cost of entry than Stata.
- Stata is more specialized:
  - Certain common tasks are simpler in Stata.
- Stata has wider adoption among micro-econometricians.
  - Network externalities in your work environment.
  - Development of new specialized techniques and tools could happen faster (e.g. *ietoolkit*).

Introduction

Here are some other advantages:

- R is a free and open source software!
- It allows you to have several data sets open simultaneously.
- It can run complex Geographic Information System (GIS) analyses.
- You can use it for web scrapping.
- You can run machine learning algorithms with it.
- You can create complex Markdown documents. This presentation, for example, is entirely done in RStudio.
- You can create interactive dashboards and online applications with the Shiny package.

Introduction

Python is even more flexible and has more users than R. So, why should I bother to learn R?

- Despite being super popular for data science, Python has fewer libraries developed for econometrics.
- Python is a bit harder to set up and get started.
- It can be a harder to find help only for statistics and econometrics especially for beginners.

Outline

1. Installation
2. Introduction
3. Getting started
4. RStudio interface
5. Data in R
6. Functions
7. R objects
8. Basic types of data
9. Advanced types of data
10. Help, Google and Stackoverflow
11. Useful resources
12. Appendix

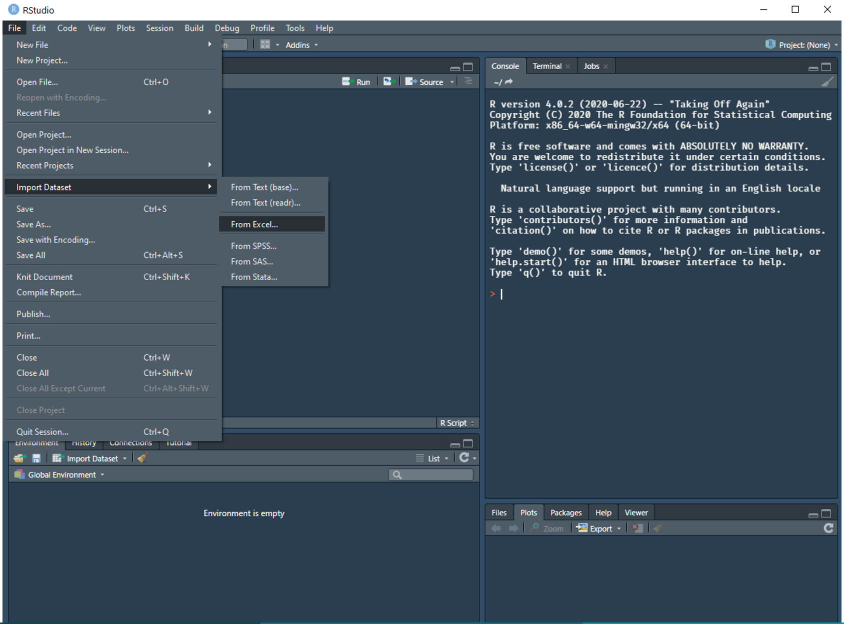
Getting started

Let's start by loading the data set we'll be using:

Exercise 1: Import data

1. In RStudio, go to File > Import Dataset > From Excel and open the cs_s0_s5_household.xls file.
   - Depending on your Rstudio version, it might be File > Import Dataset > From CSV
2. The file should be in GitHub/dime-r-training/DataWork/DataSets/Final
3. Change the name to cs on the import window

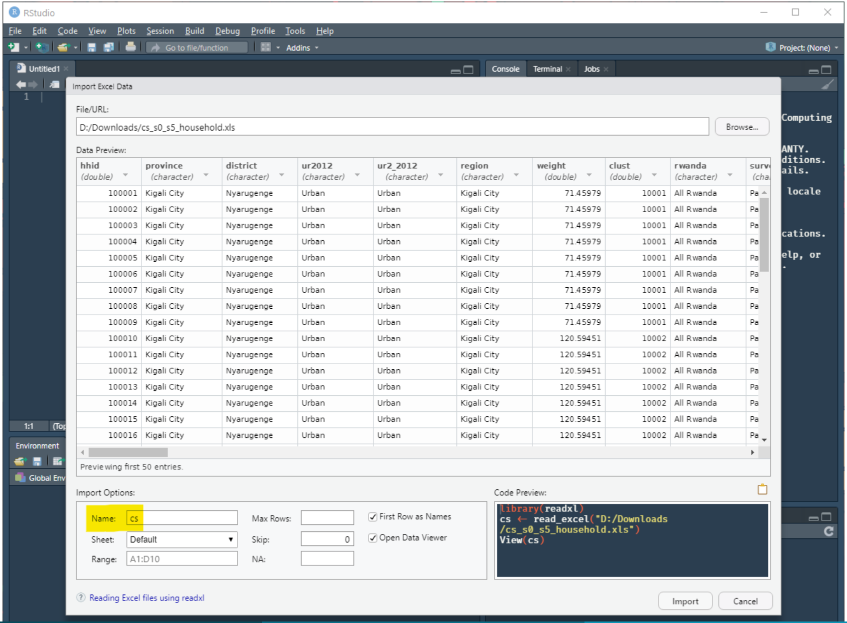
Getting started



```
R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

Environment is empty

Getting started



Import Excel Data

File/URL:

D:/Downloads/cs_s0_s5_household.xls

Browse...

Data Preview:

hhid (double)	province (character)	district (character)	ur2012 (character)	ur2_2012 (character)	region (character)	weight (double)	clust (double)	rwanda (character)	surv (cha
100001	Kigali City	Nyarugenge	Urban	Urban	Kigali City	71.45979	10001	All Rwanda	Pa
100002	Kigali City	Nyarugenge	Urban	Urban	Kigali City	71.45979	10001	All Rwanda	Pa
100003	Kigali City	Nyarugenge	Urban	Urban	Kigali City	71.45979	10001	All Rwanda	Pa
100004	Kigali City	Nyarugenge	Urban	Urban	Kigali City	71.45979	10001	All Rwanda	Pa
100005	Kigali City	Nyarugenge	Urban	Urban	Kigali City	71.45979	10001	All Rwanda	Pa
100006	Kigali City	Nyarugenge	Urban	Urban	Kigali City	71.45979	10001	All Rwanda	Pa
100007	Kigali City	Nyarugenge	Urban	Urban	Kigali City	71.45979	10001	All Rwanda	Pa
100008	Kigali City	Nyarugenge	Urban	Urban	Kigali City	71.45979	10001	All Rwanda	Pa
100009	Kigali City	Nyarugenge	Urban	Urban	Kigali City	71.45979	10001	All Rwanda	Pa
100010	Kigali City	Nyarugenge	Urban	Urban	Kigali City	120.59451	10002	All Rwanda	Pa
100011	Kigali City	Nyarugenge	Urban	Urban	Kigali City	120.59451	10002	All Rwanda	Pa
100012	Kigali City	Nyarugenge	Urban	Urban	Kigali City	120.59451	10002	All Rwanda	Pa
100013	Kigali City	Nyarugenge	Urban	Urban	Kigali City	120.59451	10002	All Rwanda	Pa
100014	Kigali City	Nyarugenge	Urban	Urban	Kigali City	120.59451	10002	All Rwanda	Pa
100015	Kigali City	Nyarugenge	Urban	Urban	Kigali City	120.59451	10002	All Rwanda	Pa
100016	Kigali City	Nyarugenge	Urban	Urban	Kigali City	120.59451	10002	All Rwanda	Pa

Previewing first 50 entries.

Import Options:

Name: cs

Max Rows:

☑ First Row as Names

Sheet: Default

Skip: 0

☑ Open Data Viewer

Range: A1:D10

NA:

Code Preview:

```
library(readxl)
cs ← read_excel("D:/Downloads
/cs_s0_s5_household.xls")
View(cs)
```

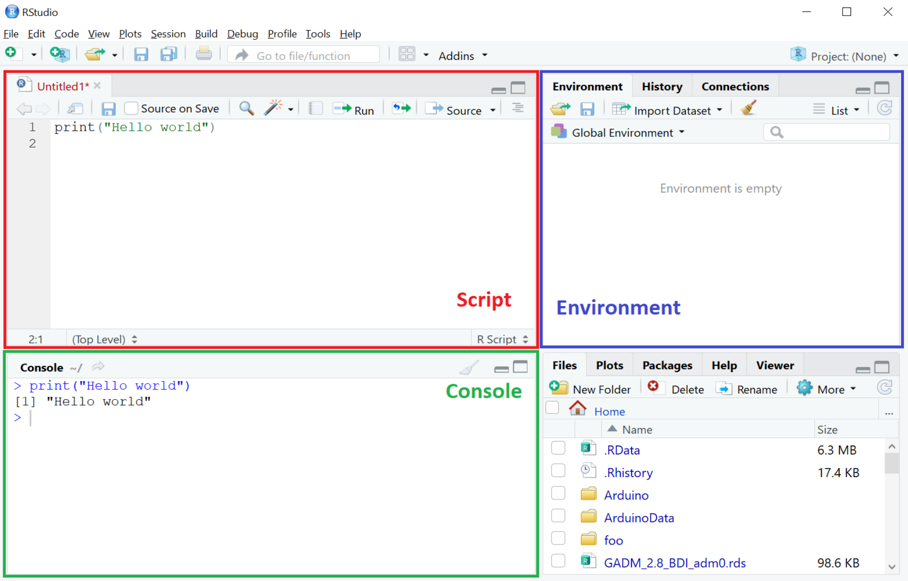
Reading Excel files using readxl

Import Cancel

Outline

1. Installation
2. Introduction
3. Getting started
4. **RStudio interface**
5. Data in R
6. Functions
7. R objects
8. Basic types of data
9. Advanced types of data
10. Help, Google and Stackoverflow
11. Useful resources
12. Appendix

RStudio interface

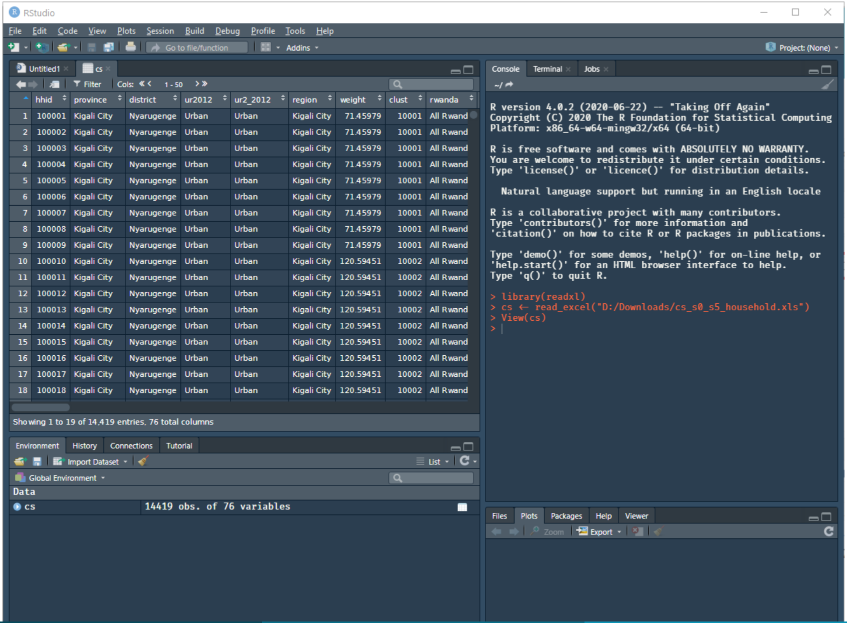


```
print("Hello world")
```

```
> print("Hello world")
[1] "Hello world"
>
```

Name	Size
.RData	6.3 MB
.Rhistory	17.4 KB
Arduino	
ArduinoData	
foo	
GADM_2.8_BDI_adm0.rds	98.6 KB

RStudio interface



	hhid	province	district	ur2012	ur2_2012	region	weight	clust	rwanda
1	100001	Kigali City	Nyarugenge	Urban	Urban	Kigali City	71.45979	10001	All Rwand
2	100002	Kigali City	Nyarugenge	Urban	Urban	Kigali City	71.45979	10001	All Rwand
3	100003	Kigali City	Nyarugenge	Urban	Urban	Kigali City	71.45979	10001	All Rwand
4	100004	Kigali City	Nyarugenge	Urban	Urban	Kigali City	71.45979	10001	All Rwand
5	100005	Kigali City	Nyarugenge	Urban	Urban	Kigali City	71.45979	10001	All Rwand
6	100006	Kigali City	Nyarugenge	Urban	Urban	Kigali City	71.45979	10001	All Rwand
7	100007	Kigali City	Nyarugenge	Urban	Urban	Kigali City	71.45979	10001	All Rwand
8	100008	Kigali City	Nyarugenge	Urban	Urban	Kigali City	71.45979	10001	All Rwand
9	100009	Kigali City	Nyarugenge	Urban	Urban	Kigali City	71.45979	10001	All Rwand
10	100010	Kigali City	Nyarugenge	Urban	Urban	Kigali City	120.59451	10002	All Rwand
11	100011	Kigali City	Nyarugenge	Urban	Urban	Kigali City	120.59451	10002	All Rwand
12	100012	Kigali City	Nyarugenge	Urban	Urban	Kigali City	120.59451	10002	All Rwand
13	100013	Kigali City	Nyarugenge	Urban	Urban	Kigali City	120.59451	10002	All Rwand
14	100014	Kigali City	Nyarugenge	Urban	Urban	Kigali City	120.59451	10002	All Rwand
15	100015	Kigali City	Nyarugenge	Urban	Urban	Kigali City	120.59451	10002	All Rwand
16	100016	Kigali City	Nyarugenge	Urban	Urban	Kigali City	120.59451	10002	All Rwand
17	100017	Kigali City	Nyarugenge	Urban	Urban	Kigali City	120.59451	10002	All Rwand
18	100018	Kigali City	Nyarugenge	Urban	Urban	Kigali City	120.59451	10002	All Rwand

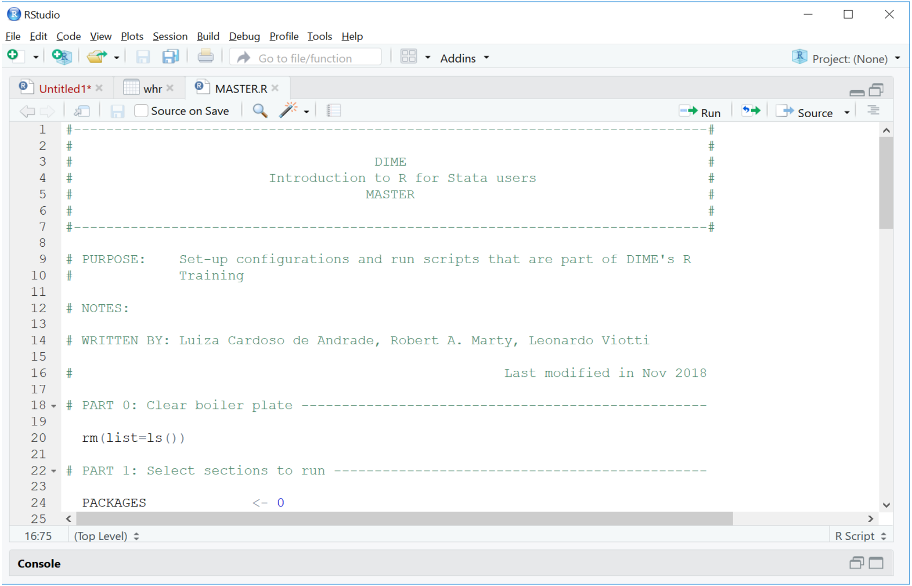
Showing 1 to 19 of 14,419 entries, 76 total columns

Data

cs 14419 obs. of 76 variables

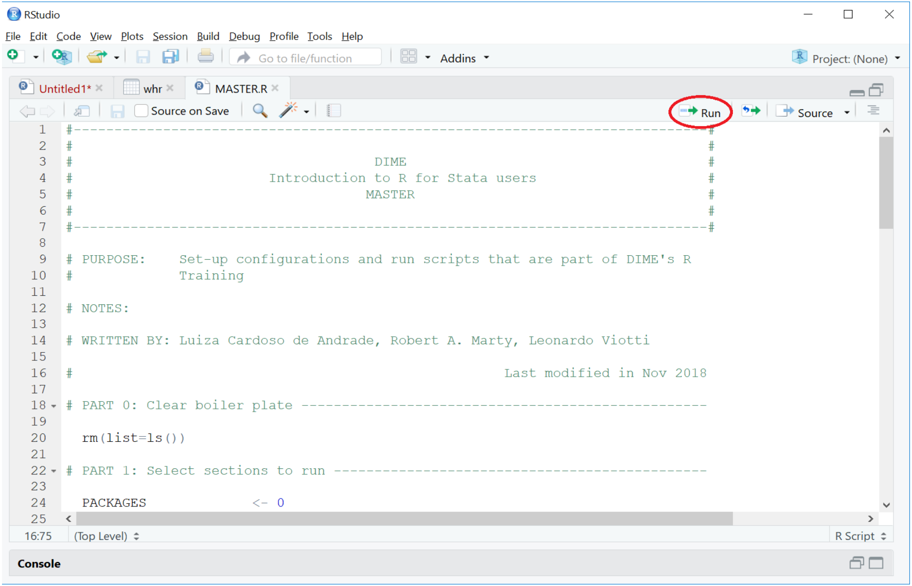
```
R version 4.0.2 (2020-06-22) — "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> library(readxl)
> cs ← read_excel("D:/Downloads/cs_s0_s5_household.xls")
> View(cs)
> |
```

RStudio interface



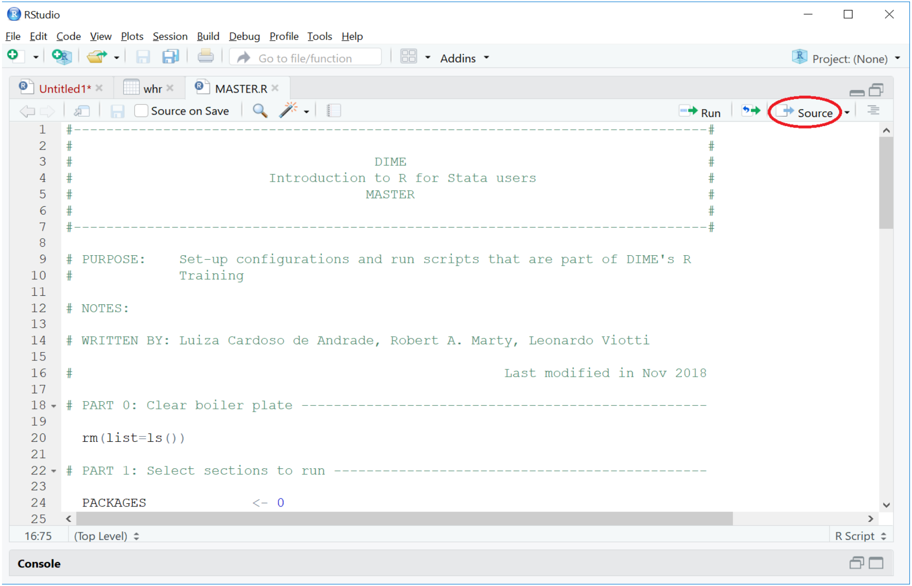
```
#----------------------------------------------------------------------------#
#                                                                            #
#                                   DIME                                     #
#                    Introduction to R for Stata users                       #
#                                  MASTER                                    #
#                                                                            #
#----------------------------------------------------------------------------#

# PURPOSE:    Set-up configurations and run scripts that are part of DIME's R
#             Training

# NOTES:

# WRITTEN BY: Luiza Cardoso de Andrade, Robert A. Marty, Leonardo Viotti

#                                                     Last modified in Nov 2018

# PART 0: Clear boiler plate --------------------------------------------------

  rm(list=ls())

# PART 1: Select sections to run ----------------------------------------------

  PACKAGES             <- 0
```

RStudio interface



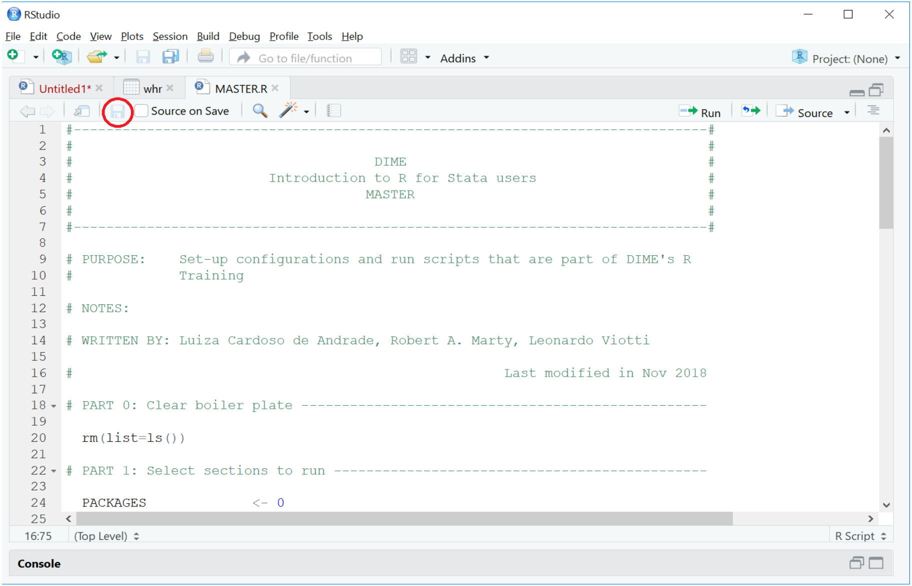
```
#-----------------------------------------------------------------------------#
#                                                                             #
#                                   DIME                                      #
#                   Introduction to R for Stata users                         #
#                                  MASTER                                     #
#                                                                             #
#-----------------------------------------------------------------------------#

# PURPOSE:    Set-up configurations and run scripts that are part of DIME's R
#             Training

# NOTES:

# WRITTEN BY: Luiza Cardoso de Andrade, Robert A. Marty, Leonardo Viotti

#                                                   Last modified in Nov 2018

# PART 0: Clear boiler plate --------------------------------------------------

  rm(list=ls())

# PART 1: Select sections to run ----------------------------------------------

  PACKAGES              <- 0
```

RStudio interface



```
#-------------------------------------------------------------------------------#
#                                                                               #
#                                     DIME                                      #
#                        Introduction to R for Stata users                      #
#                                    MASTER                                     #
#                                                                               #
#-------------------------------------------------------------------------------#

# PURPOSE:    Set-up configurations and run scripts that are part of DIME's R
#             Training

# NOTES:

# WRITTEN BY: Luiza Cardoso de Andrade, Robert A. Marty, Leonardo Viotti

#                                                    Last modified in Nov 2018

# PART 0: Clear boiler plate --------------------------------------------------

  rm(list=ls())

# PART 1: Select sections to run ----------------------------------------------

  PACKAGES             <- 0
```

RStudio interface



```
#-----------------------------------------------------------------------------#
#                                                                             #
#                                    DIME                                     #
#                      Introduction to R for Stata users                      #
#                                   MASTER                                    #
#                                                                             #
#-----------------------------------------------------------------------------#

# PURPOSE:    Set-up configurations and run scripts that are part of DIME's R
#             Training

# NOTES:

# WRITTEN BY: Luiza Cardoso de Andrade, Robert A. Marty, Leonardo Viotti

#                                                   Last modified in Nov 2018

# PART 0: Clear boiler plate --------------------------------------------------

  rm(list=ls())

# PART 1: Select sections to run ----------------------------------------------

  PACKAGES            <- 0
```

Outline

1. Installation
2. Introduction
3. Getting started
4. RStudio interface
5. **Data in R**
6. Functions
7. R objects
8. Basic types of data
9. Advanced types of data
10. Help, Google and Stackoverflow
11. Useful resources
12. Appendix

Data in R

- In Stata, you can open ONE dataset at a time, and perform operations that can change the dataset.
- You can also have other things, such as matrices, macros and tempfiles, but they are secondary, and most functions only use the main dataset.
- If you wish to do any non-permanent changes to your data, you'll need to preserve the original data to keep it intact.

Data in R

R works in a completely different way:

- You can have as many datasets (objects) as you wish or your computer's memory allows.
- Operations will only have lasting effects if you store them.

Data in R

- Everything that exists in R's memory – variables, datasets, functions – is an object.
- You could think of an object like a chunk of data with some properties that has a name by which you call it.
- If you create an object, it is going to be stored in memory until you delete it or quit R.
- Whenever you run anything you intend to use in the future, you need to store it as an object.

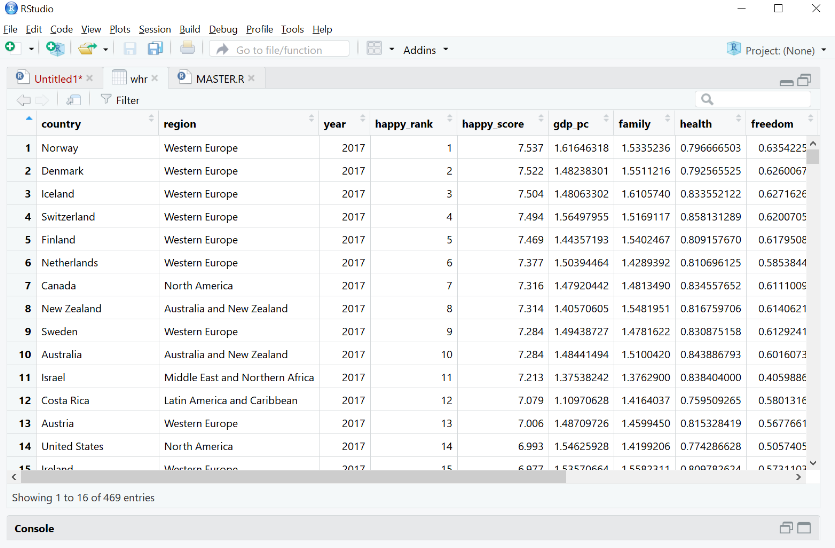
Data in R

To better understand the idea, we're going to use the data we opened from the Rwanda's EICV data. First, let's take a look at the data.

Type the following code to explore the data:

```r
# We can use the function View() to browse the whole data
View(eicv)

# Alternatively we can print the first 6 obs. with head()
head(eicv)
```

Data in R



	country	region	year	happy_rank	happy_score	gdp_pc	family	health	freedom
1	Norway	Western Europe	2017	1	7.537	1.61646318	1.5335236	0.796666503	0.6354225
2	Denmark	Western Europe	2017	2	7.522	1.48238301	1.5511216	0.792565525	0.6260067
3	Iceland	Western Europe	2017	3	7.504	1.48063302	1.6105740	0.833552122	0.6271626
4	Switzerland	Western Europe	2017	4	7.494	1.56497955	1.5169117	0.858131289	0.6200705
5	Finland	Western Europe	2017	5	7.469	1.44357193	1.5402467	0.809157670	0.6179508
6	Netherlands	Western Europe	2017	6	7.377	1.50394464	1.4289392	0.810696125	0.5853844
7	Canada	North America	2017	7	7.316	1.47920442	1.4813490	0.834557652	0.6111009
8	New Zealand	Australia and New Zealand	2017	8	7.314	1.40570605	1.5481951	0.816759706	0.6140621
9	Sweden	Western Europe	2017	9	7.284	1.49438727	1.4781622	0.830875158	0.6129241
10	Australia	Australia and New Zealand	2017	10	7.284	1.48441494	1.5100420	0.843886793	0.6016073
11	Israel	Middle East and Northern Africa	2017	11	7.213	1.37538242	1.3762900	0.838404000	0.4059886
12	Costa Rica	Latin America and Caribbean	2017	12	7.079	1.10970628	1.4164037	0.759509265	0.5801316
13	Austria	Western Europe	2017	13	7.006	1.48709726	1.4599450	0.815328419	0.5677661
14	United States	North America	2017	14	6.993	1.54625928	1.4199206	0.774286628	0.5057405
15	Ireland	Western Europe	2017	15	6.977	1.53570664	1.5582311	0.809782624	0.5731103

Showing 1 to 16 of 469 entries

Data in R

```
eicv
```

```
## # A tibble: 14,419 x 76
##      hhid province district ur2012 ur2_2012 region weight clust rwanda surveyh
##     <dbl> <chr>    <chr>    <chr>  <chr>    <chr>   <dbl> <dbl> <chr>  <chr>  
##  1 100001 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel  
##  2 100002 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel  
##  3 100003 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel  
##  4 100004 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel  
##  5 100005 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel  
##  6 100006 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel  
##  7 100007 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel  
##  8 100008 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel  
##  9 100009 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel  
## 10 100010 Kigali ~ Nyaruge~ Urban  Urban    Kigal~  121.  10002 All R~ Panel  
## # ... with 14,409 more rows, and 66 more variables: quintile <chr>,
## #   poverty <dbl>, Consumption <dbl>, hhtype <chr>, s0q19m <dbl>, s0q19y <dbl>,
## #   s0qb <chr>, s5aq1 <chr>, s5aq2 <chr>, s5aq3 <chr>, s5aq4 <chr>,
## #   s5aq5 <chr>, s5aq6 <chr>, s5aq7y <chr>, s5aq7m <chr>, s5aq8 <chr>,
## #   s5aq9 <chr>, s5aq10 <chr>, s5aq11 <chr>, s5bq1 <chr>, s5bq2 <chr>,
## #   s5bq3a <chr>, s5bq3b <chr>, s5bq4a <chr>, s5bq4b <chr>, s5bq5 <chr>,
## #   s5bq6a <chr>, s5bq6b <chr>, s5bq7 <chr>, s5bq8 <chr>, s5bq9a <chr>,
## #   s5bq9b <chr>, s5bq10 <chr>, s5bq11 <chr>, s5cq1 <chr>, s5cq2 <chr>,
## #   s5cq3 <chr>, s5cq4 <chr>, s5cq5 <chr>, s5cq6 <chr>, s5cq7 <chr>,
## #   s5cq8 <chr>, s5cq9a <chr>, s5cq9b <chr>, s5cq10 <chr>, s5cq11 <chr>,
```

Data in R

Now, let's try some simple manipulations. First, assume we're only interested in data of the year 2015.

Exercise 2: Subset the data

1. Subset the data set, keeping only observations where variable `region` equals `Kigali City`.

```
# To do that we'll use the subset() function
subset(eicv, region == "Kigali City")
```

2. Then, look again at the first 6 observations

```
# Use the head() function again
head(eicv)
```

Data in R

```r
head(eicv)
```

```
## # A tibble: 6 x 12
##     hhid province district ur2012 ur2_2012 region weight clust rwanda surveyh
##    <dbl> <chr>    <chr>    <chr>  <chr>    <chr>   <dbl> <dbl> <chr>  <chr>
## 1 100001 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel
## 2 100002 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel
## 3 100003 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel
## 4 100004 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel
## 5 100005 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel
## 6 100006 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel
## # ... with 2 more variables: quintile <chr>, poverty <dbl>
```

Data in R

We can see that nothing happened to the original data. This happens because we didn't store the edit we made anywhere.

To store an object, we use the assignment operator (<-):

```
# Assign the Answer to the Ultimate Question of Life,
# the Universe, and Everything
x <- 42
```

From now on, *x* is associated with the stored value (until you replace it, delete it, or quit the R session).

Data in R

Exercise 3: Create an object

Create a new data set, called `kigali`, that is a subset of the `eicv` data set containing only data from the Kigali City region.

```r
# Using the same function but now assigning it to an object
kigaly <- subset(eicv, region == "Kigali City")

# Display the 5 first obs. of the new data
head(kigaly)

# Notice that we still have the original data set intact
head(kigaly)
```

Data in R

```
head(kigaly)
```

```
## # A tibble: 6 x 12
##      hhid province district ur2012 ur2_2012 region weight clust rwanda surveyh
##     <dbl> <chr>    <chr>    <chr>  <chr>    <chr>   <dbl> <dbl> <chr>  <chr>
## 1 100001 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel
## 2 100002 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel
## 3 100003 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel
## 4 100004 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel
## 5 100005 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel
## 6 100006 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel
## # ... with 2 more variables: quintile <chr>, poverty <dbl>
```

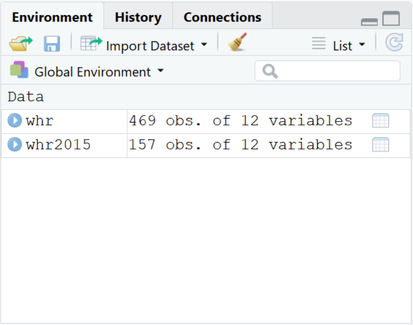
Data in R

```r
head(kigaly)
```

```
## # A tibble: 6 x 12
##     hhid province district ur2012 ur2_2012 region weight clust rwanda surveyh
##    <dbl> <chr>    <chr>    <chr>  <chr>    <chr>   <dbl> <dbl> <chr>  <chr>
## 1 100001 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel
## 2 100002 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel
## 3 100003 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel
## 4 100004 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel
## 5 100005 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel
## 6 100006 Kigali ~ Nyaruge~ Urban  Urban    Kigal~   71.5 10001 All R~ Panel
## # ... with 2 more variables: quintile <chr>, poverty <dbl>
```

Data in R

You can also see that your environment pane now has two objects:



Data

whr	469 obs. of 12 variables
whr2015	157 obs. of 12 variables

Data in R

Two important concepts to take note:

1. In R, if you want to change your data, you need to store it in an object.
2. It is possible to simply replace the original data. This happens if you assign the new object to the same name as the original.
3. Print (display) is built into R. If you execute any action without storing it, R will simply print the results of that action but won't save anything in the memory.

Outline

1. Installation
2. Introduction
3. Getting started
4. RStudio interface
5. Data in R
6. **Functions**
7. R objects
8. Basic types of data
9. Advanced types of data
10. Help, Google and Stackoverflow
11. Useful resources
12. Appendix

Quick intro to functions

`head()`, `View()`, `subset()` and `read.csv()` are functions!

- Functions in R take named arguments (unlike in Stata that you have arguments and options).
- Usually the first argument is the obeject you want to use the function on, e.g. `subset(kigali, ...)`
- Functions usually return values that you can store in an object, print or use directly as an argumet of another function.

We will explore this ideas in depth in the next session.

Outline

1. Installation
2. Introduction
3. Getting started
4. RStudio interface
5. Data in R
6. Functions
7. R objects
8. Basic types of data
9. Advanced types of data
10. Help, Google and Stackoverflow
11. Useful resources
12. Appendix

R objects

Objects are the building blocks of R programming. This section will explore some of the most common classes, focused on data structures.

This will give you the foundation to explore your data and construct analytical outputs.

R objects

What is an object?

- An object is like a global in Stata, it's something you can refer to later in your code to get a value.
- But while you can only put a number or a string in a global you can put anything into an object: strings, data sets, vectors, graphs, functions, etc.
- Objects also have attributes that can be used to manipulate it.

R objects

Here are the object classes we will cover today:

- **Vectors:** an uni-dimensional object that stores a sequence of values
- **Data frames:** a combination of different vectors of the same length (the same as your data set in Stata)
- **Lists:** a multidimensional object that can store several objects of different dimension

R objects

Vectors

A vector is an uni-dimensional object composed by one or more scalars of the same type.

Use the following code to create vectors in two different ways

```
# Creating a vector with the c() function
v1 <- c(1,1,2,3,5)

# Alternative way to create an evenly spaced vector
v2 <- 1:5
```

You can use brackets for indexing

```
# Print the 4th element of the vector
v2[4]
```

```
## [1] 4
```

R objects

Vectors

To R, each of the columns of eicv is a vector.

Calling a vector from a data.frame column

We use the $ to call vectors (variables) by their names in a data.frame

Type the following code:

```
# Create a vector with the values of the `year` variable
region_vec <- eicv$region

# See the 3 first elements of the year column
eicv$region[1:3]
```

```
## [1] "Kigali City" "Kigali City" "Kigali City"
```

R objects

Data Frames

The eicv and kigali objects are both data frames. You can also construct a new data frame from scratch by combining vectors with the same number of elements .

Now, type the following code to create a new data frame

```r
# Dataframe created by biding vectors
df1 <- data.frame(v1,v2)
df1
```

```
##   v1 v2
## 1  1  1
## 2  1  2
## 3  2  3
## 4  3  4
## 5  5  5
```

R objects

Data Frames

Since a data frame has two dimensions, you can use indexing on both:

Numeric indexing

```r
# The first column of whr
eicv[,1]

# The 45th line of whr
eicv[45,]

# Or the 45th element of the first column
eicv[45,1]
```

R objects

Data Frames

Alternatively, you can use the column names for indexing, which is the same as using the $ sign.

Names indexing

```
# Or the 22th element of the province column
eicv[22,"province"] # The same as eicu$province[22]
```

```
## # A tibble: 1 x 1
##   province
##   <chr>
## 1 Kigali City
```

R objects

Data Frames

Lists are more complex objects that can contain many objects of different classes and dimensions.

The outputs of many functions, a regression for example, are simmilar to lists.

It would be beyond the scope of this introduction to go deep into them, but here's a quick example

Combine several objects of different types in a list

```
# Use the list() function
lst <- list(v1, df1, 45)
```

Print the list yourself to see how it looks like.

R objects

Lists

```
# Check the contents of lst
print(lst)
```

```
## [[1]]
## [1] 1 1 2 3 5
##
## [[2]]
##   v1 v2
## 1  1  1
## 2  1  2
## 3  2  3
## 4  3  4
## 5  5  5
##
## [[3]]
## [1] 45
```

Outline

1 Installation

2 Introduction

3 Getting started

4 RStudio interface

5 Data in R

6 Functions

7 R objects

8 Basic types of data

9 Advanced types of data

10 Help, Google and Stackoverflow

11 Useful resources

12 Appendix

Basic types of data

R has different kinds of data that can be recorded inside objects. They are very similar to what you have in Stata, and the main types are string, integer, numeric, factor and boolean.

Let's start with the simpler ones:

Strings

A sequence of characters and are usually represented between double quotes. They can contain single letters, words, phrases or even some longer text.

Integer and numeric

As in Stata, these are two different ways to store numbers. They are different because they use memory differently. As default, R stores numbers in the numeric format (double).

Basic types of data

Strings

Now we'll use string data to practice some basic object manipulations in R.

Exercise 4: Create a vector of strings

Create a string vector containing the names of commonly used statistical software in order of importance:

```
# Creating string vector
str_vec <- c("R",
             "Python",
             "SAS",
             "Excel",
             "Stata")
```

Now print them to check them out.

Basic types of data

Strings

Exercise 5: Concatenate strings

1. Create a scalar (a vector of one element) containing the phrase “is better than” and cal it `str_scalar`.
2. Use the function `paste()` with 3 arguments separated by commas:
   - The first argument as the 1st element of `str_vec`.
   - The second argument as the `str_scalar`.
   - The third argument as the 5th element of `str_vec`.
3. If you’re not sure where to start, type:

```r
help(paste)
```

Basic types of data

Strings

```r
### Using the paste function to combine strings

# Scalar
str_scalar <- "is better than"

# Using the paste() function
paste(str_vec[1], str_scalar, str_vec[5])
```

```
## [1] "R is better than Stata"
```

Outline

1. Installation
2. Introduction
3. Getting started
4. RStudio interface
5. Data in R
6. Functions
7. R objects
8. Basic types of data
9. **Advanced types of data**
10. Help, Google and Stackoverflow
11. Useful resources
12. Appendix

Advanced types of data

R also has other more complex ways of storing data. These are the most used:

Factors

Factors are numeric categorical values with text label, equivalent to labelled variables in Stata. Turning strings into factors makes it easier to run different analyses on them and also uses less space in your memory.

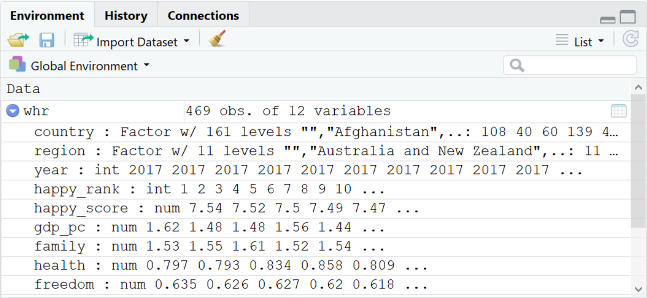
Booleans

Booleans are logical binary variables, accepting either `TRUE` or `FALSE` as values. They are automatically generated when performing logical operations

Advanced types of data

Factors

In `eicv`, we can see that `country` and `region` are factor variables. In your environment panel, there is the information of the type of all variables, and for factors the number of levels.



```
Data
whr                     469 obs. of 12 variables
    country : Factor w/ 161 levels "","Afghanistan",..: 108 40 60 139 4…
    region : Factor w/ 11 levels "","Australia and New Zealand",..: 11 …
    year : int 2017 2017 2017 2017 2017 2017 2017 2017 2017 2017 ...
    happy_rank : int 1 2 3 4 5 6 7 8 9 10 ...
    happy_score : num 7.54 7.52 7.5 7.49 7.47 ...
    gdp_pc : num 1.62 1.48 1.48 1.56 1.44 ...
    family : num 1.53 1.55 1.61 1.52 1.54 ...
    health : num 0.797 0.793 0.834 0.858 0.809 ...
    freedom : num 0.635 0.626 0.627 0.62 0.618 ...
```

Advanced types of data

Factors

We'll learn how to deal with factors in detail on the next session, since they are very important for us. For now, here are two important things to keep in mind when using them:

Warning:

Unlike Stata, in R

1. You use the labels to refer to factors
2. You cannot choose the underlying values

Advanced types of data

Booleans

Boolean data is the result of logical conditions

- Whenever you're using an `if` statement in Stata, you're implicitly using boolean data.
- The difference is that in R, this can be done in 2 steps.

Advanced types of data

Booleans

Exercise 6:

Create boolean vector with the condition of `Consumption` below average:

```r
# Create vector
bool_vec <- eicv$Consumption < mean(eicv$Consumption)

# See the 6 first elements of the vector
head(bool_vec)
```

```
## [1]  TRUE FALSE FALSE  TRUE FALSE  TRUE
```

Advanced types of data

Booleans

Let's use the boolean vector created to add a dummy variable in the whr data set for the same condition.

Exercise 6:

1. Create a column in whr containing zeros and call it rank_low. You can do this by typing:

```
eicv$rank_low <- 0
```

2. Use bool_vec to index the lines of the income_low column and replace all observations that meet the condition with the value 1.

```
eicv$rank_low[bool_vec] <- 1
```

Advanced types of data

Booleans

```
# Replace with 1 those obs that meet the condition
eicv$rank_low [bool_vec] <- 1
# is the same as
eicv$rank_low[eicv$Consumption < mean(eicv$Consumption)] <- 1
# This in stata would be
# replace rank_low = 1 if (...)


# We can use the summary function to get descriptives
summary(eicv$rank_low)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  1.0000  0.7427  1.0000  1.0000
```

Outline

1. Installation
2. Introduction
3. Getting started
4. RStudio interface
5. Data in R
6. Functions
7. R objects
8. Basic types of data
9. Advanced types of data
10. Help, Google and Stackoverflow
11. Useful resources
12. Appendix

Help, Google and Stackoverflow

Help in R works very much like in Stata: the help files usually start with a brief description of the function, explain its syntax and arguments and list a few examples. There are two ways to access help files:

Exercise 7: Use help

```r
# The help() function
help(summary)

# and its abbreviation
?summary
```

Help, Google and Stackoverflow

- The biggest difference, however, is that R has a much wider user community and it has a lot more online resources.
- For instance, in 2014, Stata had 11 dedicated blogs written by users, while R had 550.[1]
- The most powerful problem-solving tool in R, however, is Google. Searching the something yields tons of results.
- Often that means a Stack Overflow page where someone asked the same question and several people gave different answers. Here's a typical example: https://stackoverflow.com/questions/1660124/how-to-sum-a-variable-by-group

[1] Check http://r4stats.com/articles/popularity/ for more.

Outline

1. Installation
2. Introduction
3. Getting started
4. RStudio interface
5. Data in R
6. Functions
7. R objects
8. Basic types of data
9. Advanced types of data
10. Help, Google and Stackoverflow
11. **Useful resources**
12. Appendix

Useful resources

Blogs and online courses:

- Surviving graduate econometrics with R: https://thetarzan.wordpress.com/2011/05/24/surviving-graduate-econometrics-with-r-the-basics-1-of-8/
- CRAN's manuals: https://cran.r-project.org/manuals.html
- R programming in Coursera: https://www.coursera.org/learn/r-programming
- R programming for dummies: http://www.dummies.com/programming/r/
- R bloggers: https://www.r-bloggers.com/
- R statistics blog: https://www.r-statistics.com/
- The R graph gallery: https://www.r-graph-gallery.com/

Useful resources

Books:

- R for Stata Users - Robert A. Muenchen and Joseph Hilbe
- R Graphics Cookbook - Winston Chang
- R for Data Science - Hadley Wickham and Garrett Grolemund

Thank you!

Outline

1. Installation
2. Introduction
3. Getting started
4. RStudio interface
5. Data in R
6. Functions
7. R objects
8. Basic types of data
9. Advanced types of data
10. Help, Google and Stackoverflow
11. Useful resources
12. Appendix

Appendix - Syntax

R's syntax is a bit heavier than Stata's:

- Parentheses to separate function names from its arguments.
- Commas to separate arguments.
- For comments we use the # sign.
- You can have line breaks inside function statements.
- In R, functions can be treated much like any other object Therefore, they can be passed as arguments to other functions.

Similarly to Stata:

- Square brackets are used for indexing.
- Curly braces are used for loops and if statements.
- Largely ignores white spaces.

Appendix - RStudio interface

Script

Where you write your code. Just like a do file.

Console

Where your results and messages will be displayed. But you can also type commands directly into the console, as in Stata.

Environment

What's in R's memory.

The 4th pane

Can display different things, including plots you create, packages loaded and help files.

Appendix - Matrices

A matrix a bi-dimensional object composed by one or more vectors of the same type.

Type the following code to test two different ways of creating matrices

```r
# Matrix created by joining two vectors:
m1 <- cbind(v1,v1)

# Matrix using the
m2 <- matrix(c(1,1,2,3,5,8), ncol = 2)
```

Appendix - Matrices

Now use the following code to check the elements of these matrices by indexing

```
# Matrix indexing: typing matrix[i,j] will give you
# the element in the ith row and jth column of that matrix
#m2[1,2]

# Matrix indexing: typing matrix[i,] will give you the
# ith row of that matrix
m1[1,]

# Matrix indexing: typing matrix[,j] will give you the
# jth column of that matrix (as a vector)
m1[,2]
```

Appendix - Advanced types of data - Factors

Factors

Create a factor verctor using the following code

```
# Basic factor vector
num_vec <- c(1,2,2,3,1,2,3,3,1,2,3,3,1)
fac_vec <- factor(num_vec)

# A bit fancier factor vector
fac_vec <- factor(num_vec,labels=c("A","B","C"))

# Change labels
levels(fac_vec) = c('One','Two','Three')
```

Appendix - Numbers and integers

Two scalars, one with a round number the other with a fractional part

```
# a numeric scalar with an integer number
int <- 13
num <- 12.99
```

Appendix - Numbers and integers

Now we can see the objects classes with the *class()* function and test it with the *is.integer()* and *is.numeric()* functions.

```
# you can see the number's format using the class function:
class(int)
```

```
## [1] "numeric"
```

```
class(num)
```

```
## [1] "numeric"
```

```
# you can test the class with the is. method
is.integer(int)
```

```
## [1] FALSE
```

```
is.numeric(int)
```

```
## [1] TRUE
```

Did you notice anything strange? That happens because the default way R stores numbers is *numeric*, which is equivalent to *double* in Stata.

Appendix - Numbers and integers

Numbers and integers

We can, however, coerce objects into different classes. We just need to be careful because the result might not be what we're expecting.

Use the *as.integer()* and *round()* functions on the *num* object to see the difference:

```
as.integer(num)
```

```
## [1] 12
```

```
# and


round(num)
```

```
## [1] 13
```