Descriptive analysis

R Training for NISR

Luiza Andrade, Leonardo Viotti & Rob Marty

Aug. 2018

Outline

Introduction

Descriptive statistics are used to represent the basic features of data. When we talk about descriptive analysis, it usually means that we're not making any assumptions, and we're not using probability theory to infer anything beyond the immediate data.

This session is mostly focused on how to implement descriptive analysis in R. We will not go in depth into these concepts, but you can find some useful references at the end of this presentation.

Introduction

This session will cover two topics:

1. Quick ways to extract summary information from your data
2. How to use this information to create and export tables

Introduction

Let's get started:

1. Run your master script to load the packages and set folder paths. Don't forget to deactivate the `INSTALL_PACKAGES` switch!
2. Open a new R script to write the code from this session
3. Load the data that we'll use for this session

Load the data

```r
# Load CSV data
lwh <- read.csv(file.path(finalData,"lwh_clean.csv"),
                header = T)
```

Outline

Quick summary statistics

`summary(x)` - equivalent to Stata's *summarize*, displays summary statistics. Its arguments are:

- **x:** the object you want to summarize, usually a vector or data frame

Exercise 1

Use the `summary()` function to display summary statistics for the entire *lwh* data frame.

Quick summary statistics

```r
# Summary statistics
summary(lwh)
```

```
##     panel_id         hh_code          wave          year     
##  Min.   :100103   Min.   :1001   Baseline:213   Min.   :2012  
##  1st Qu.:208677   1st Qu.:2087   Endline :307   1st Qu.:2013  
##  Median :402755   Median :4028   FUP1&2  :126   Median :2014  
##  Mean   :402210   Mean   :4022   FUP3    :345   Mean   :2015  
##  3rd Qu.:509629   3rd Qu.:5096   FUP4    :293   3rd Qu.:2016  
##  Max.   :712501   Max.   :7125                  Max.   :2018  
##     treatment_hh   treatment_site         site_code    gender_hhh 
##  Control  :689   Control  :637    Kayanza 15   :279   Female:413  
##  Treatment:595   Treatment:647    Kayanza 4    :231   Male  :784  
##                                   Rwamangana 2 :159   NA's  : 87  
##                                   Rwamangana 33:199               
##                                   Rwamangana 34:194               
##                                   Rwamangana 35:222               
```

Quick summary statistics

`table()` - equivalent to `tabulate` in Stata, creates a frequency table. Its main arguments are the objects to be tabulated.

Exercise 2

Use the `table()` function to display frequency tables for:

1. The variable *year* in the *lwh* data frame
2. The variables *gender_hhh* and *year* in the *lwh* data frame, simultaneously

Quick summary statistics

```
# Year of data collection
table(lwh$year)
```

```
##
## 2012 2013 2014 2016 2018
##  213  126  345  293  307
```

Quick summary statistics

```
# Gender of household head per year
table(lwh$gender_hhh, lwh$year)
```

```
##
##          2012 2013 2014 2016 2018
##   Female   61   41  109   94  108
##   Male    152   85  150  199  198
```

Quick summary statistics - Stargazer

We can also use the `stargazer()` function to quickly display a nice-looking descriptives table.

Stargazer was originally developed to export beautiful regression tables to LaTeX or html, but it also allows you to generate summary statistics.

Quick summary statistics - Stargazer

Exercise 3 - stargazer() summary statistics table

Use the stargazer() function to display summary statistics for the variables in the *lwh* data frame.

The stargazer() function accepts **a lot** of arguments, most of which are beyond the scope of this session. Here are the arguments you'll need for this specific table:

- **x:** the object you want to summarize – in this case a vector or data frame
- **type:** the output format – "text" to just display, "latex" (the default) to save as a LaTeX table, and "html" for, yes, html
- **digits:** the number of decimal digits to be displayed

Quick summary statistics - Stargazer

```r
# A descriptive table with stargazer
stargazer(lwh,
          digits = 1,
          type = "text")
```

```
##
## ===================================================================================
## Statistic              N     Mean    St. Dev.    Min    Pctl(25) Pctl(75)     Max
## -----------------------------------------------------------------------------------
## panel_id             1,284 402,210.3 202,211.7 100,103 208,677   509,629   712,501
## hh_code              1,284  4,022.1   2,022.1   1,001  2,086.8   5,096.2    7,125
## year                 1,284  2,015.0     2.1     2,012   2,013     2,016     2,018
## age_hhh               928    48.1      14.8     20.0    35.0      58.2      93.0
## num_dependents        339     2.1       1.4      0.0    1.0       3.0       6.0
## read_and_write        339    0.5       0.5      0.0    0.0       1.0       1.0
## w_gross_yield_a      1,284 87,599.4  112,690.1   0       0      129,342.6  483,333
## w_gross_yield_b      1,284 88,838.5  128,914.0  0.0     0.0     118,237.6 769,962.0
## expend_food_yearly   1,284 159,650.0 125,232.7  0.0   52,177.5 243,734.1 488,381.4
## expend_food_lastweek 1,284  3,059.7   2,400.1    0     1,000    4,671.2    9,360
## -----------------------------------------------------------------------------------
```

Outline

Descriptives tables

In R, it is relatively easy to construct any table you can think of by manipulating objects. You can either export an existing data frame or create others tailored to be exported.

Than, it is possible to export them to a myriad of formats. Here, we will focus on exporting tables to Microsoft Excel.

But we'll start by showing a few usefull ways to maniipulate data into exportable formats.

Descriptives tables

First, we'll construct a table by creating a data.frame To construct a table from scratch, we will use two functions:

- `sapply()`: loops through the elements of an object applying a function
- `data.frame()`: creates a data frame object by combining vectors.

Descriptives tables - sapply()

`sapply(X, FUN, ...)`: applies a function to all elements of an object and returns the result in a vector or matrix. Its arguments are:

- *X:* a matrix (or data frame) the function will be applied to
- *FUN:* the function you want to apply
- `...:` possible function options

Descriptives tables - data.frame()

`data.frame(varName1 = vector1, varName2 = vector2, ...)`: The function `data.frame()` creates data frames, tightly coupled collections of variables which share many of the properties of matrices and of lists, used as the fundamental data structure by most of R's modeling software.

- *varName1:* first column name
- *vector1:* vector that will be first column
- *varName2:*: second column name
- *vector2:* vector that will be second column

.

.

.

Descriptives tables

Before doing an exercise, lets create a simpler version of the lwh data containg fewer columns:

```
# Vector with covariates to be kept
covariates <- c("age_hhh",
                "num_dependents",
                "income_total_trim",
                "expend_food_yearly",
                "w_area_plots_a")

lwh_simp <- lwh[,covariates]
```

Descriptives tables

Exercise 4

1. Use the `sapply()` function to create a vector of means for the columns of the *lwh_simp* data frame.
2. Create vectors with standard deviations, maximum value, minimum value for the same columns.
   - TIP: set the `na.rm` argument as `TRUE`
3. Use the `data.frame()` function to combine all 4 vectors chosing column names.

Descriptives tables

```r
# Create vectors containing descriptives
  meanVec <- sapply(lwh_simp, mean, na.rm = T)
  sdVec <- sapply(lwh_simp, sd, na.rm = T)
  maxVec <- sapply(lwh_simp, max, na.rm = T)
  minVec <- sapply(lwh_simp, min, na.rm = T)

  # Combine all created vectors into a dataframe
  lwh_desc <- data.frame( Mean = meanVec,
                          StdDev = sdVec,
                          Max = maxVec,
                          Min = minVec)
  lwh_desc
```

```
##                             Mean       StdDev        Max         Min
## age_hhh            4.812069e+01 1.483241e+01     93.000 2.00000e+01
## num_dependents     2.117994e+00 1.440414e+00      6.000 0.00000e+00
## income_total_trim  6.854368e+04 6.846873e+04 274400.000 0.00000e+00
## expend_food_yearly 1.596500e+05 1.252327e+05 488381.410 0.00000e+00
## w_area_plots_a     4.234501e-01 3.338379e-01      1.379 2.47097e-04
```

Descriptives tables

The `data.frame()` function is not the only way of creating data frames. There are several other functions that can process existing objects and output a data frame object.

For the next exercises, we'll use two of them:

- `aggregate()` - Similar to collapse in Stata, it can compute statistics of a variable based on the values of other variable
- `reshape()` - Reshapes data sets from long to wide and vice-versa

Descriptives tables - aggregate()

`aggregate(X, by, FUN)`:

- **x**: a data frame or column
- **by**: a list of grouping variables
- **FUN**: a function to compute statistics

Descriptives tables - aggregate()

Exercise 5

Use the aggregate function to create a data frame called year_inc_tab with the mean of the total income per year and treatment status. The syntax of the aggregate function is very similar to that of the collapse function in Stata.

Descriptives tables - aggregate()

```
* Aggregate income by year and treatment status in stata
collapse  (mean) income_total_win, ///
          by     (year treatment_hh)
```

```r
# Aggregate income by year and treatment status in R
year_inc_tab <-
  aggregate(x = lwh$income_total_win, # data.frame
            by = list(year = lwh$year, # list
                      treatment = lwh$treatment_hh),
            FUN = mean) # function
```

Note that the `income_total_win` variable is now named `x` in the `income` data frame

Descriptives tables - aggregate()

```r
print(year_inc_tab)
```

```
##    year treatment         x
## 1  2012   Control  47958.37
## 2  2013   Control  63482.25
## 3  2014   Control  59232.51
## 4  2016   Control  71106.46
## 5  2018   Control  98294.61
## 6  2012 Treatment  41459.83
## 7  2013 Treatment 104023.15
## 8  2014 Treatment  80672.20
## 9  2016 Treatment  85192.96
## 10 2018 Treatment  99510.29
```

Descriptives tables - reshape()

`reshape(data, varying, idvar, timevar, direction)`:

- **data**: a data frame
- **idvar**: the variables that identify the group in the wide data set
- **timevar**: the variable in long format that differentiates multiple records from the same group or individual

Descriptives tables - reshape()

Exercise 6

Use the `reshape` function to make the `year_inc_tab` data frame wide per treatment status.

```r
# Aggregate income by year and treatment status
year_inc_tab <- reshape(year_inc_tab,
                        idvar = "treatment",
                        timevar = "year",
                        direction = "wide")
```

For comparison, here's how you'd do it in Stata:

```
reshape wide varlist, i(year) j(treatment)
```

Descriptives tables - reshape()

```
print(year_inc_tab)
```

```
##   treatment   x.2012    x.2013   x.2014   x.2016   x.2018
## 1   Control 47958.37  63482.25 59232.51 71106.46 98294.61
## 6 Treatment 41459.83 104023.15 80672.20 85192.96 99510.29
```

Descriptives tables - names()

Before exporting a data frame, it is often usefull to rename its columns. This can be done using the `names()` function.

The names function retrieves the column names of a data frame as a vector

```r
names(lwh_simp)
```

```
## [1] "age_hhh"            "num_dependents"     "income_total_trim"
## [4] "expend_food_yearly" "w_area_plots_a"
```

But it can also be used to relaplace the existing names attribute with a new vector. Like this

```r
names(lwh_simp) <- newNamesVector
```

Descriptives tables - names()

Exercise 6

Now, use the `names()` function to rename the columns of `year_inc_tab`.

Descriptives tables - names()

```r
names(year_inc_tab) <- c("Treatment Status", 2012, 2013, 2014, 2016, 2018)
```

Outline

Export tables to Excel

There are several ways to export R objects to Excel. We will use here the the `write.xslx()` function of the `openxlsx` package.

It takes a matrix or data frame object as input and saves it as a `.xlsx` file

`write.xslx()` is one of the most commun functions, but there are many other functions that allow you to export formatted tables to Microsoft Excel, Word or PowerPoint. Here are some examples:

- ReporteRs
- Flextable
- r2excel (only available in GitHub).

Export tables to Excel

First, install and load the `openxlsx` package:

```r
install.packages("openxlsx", dep = T)
library(openxlsx)
```

Export tables to Excel

```
write.xlsx(x, file, row.names = TRUE, col.names ...)
```

- **x**: the object to be written
- **file**: where to save the table, i.e., the file path including the file name
- **row.names**: a logical value indicating whether the row names of x are to be written along with x

Export tables to Excel

Exercise 8

Use the `write.xlsx()` function to save the `year_inc_tab` you table created in Exercise 6 into a xlsx file.

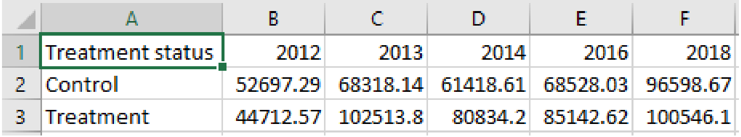
1. Set `x` argument as *year_inc_tab*.
2. Set `row.names` as `FALSE`
3. Set `file` as the folder path to your output folder plus a name for a file plus ".xlsx"

Tips:

- Make sure to save it in the *Ouput* folder. You can you the function `file.path` to do it
- Use the help function to check syntax if needed

Export tables to Excel

```
write.xlsx(year_inc_tab,
           file = file.path(Output, "year_inc_tab.xlsx"),
           row.names = F)
```



	A	B	C	D	E	F
1	Treatment status	2012	2013	2014	2016	2018
2	Control	52697.29	68318.14	61418.61	68528.03	96598.67
3	Treatment	44712.57	102513.8	80834.2	85142.62	100546.1

Outline

References and recommendations

- Johns Hopkins Exploratory Data Analysis at Coursera: https://www.coursera.org/learn/exploratory-data-analysis
- Udacity's Data Analysis with R: https://www.udacity.com/course/data-analysis-with-r--ud651
- Jake Russ stargazer cheat sheet: https://www.jakeruss.com/cheatsheets/stargazer/