Workshop

Statistical Computing: descriptive statistics with R

Andy Buerki

Overview

Descriptive statistics aim to describe a data set, typically a sample of a larger population of interest, by summarising and visualising selected trends and features. Deriving descriptive statistics can be the goal of an analysis or, more often, a vital step in understanding the structure of data before the application of inferential statistics. The clear and accurate presentation of descriptive statistics is also of key importance in write-ups of research that features quantitative aspects.

This workshop focuses on using R to produce descriptive statistics from data sets, including high-quality plots and graphs that are of the right quality for submission to journals. The advantage of using R is that once the basics are mastered, it is very quick and easy to produce an array of high quality measures and graphs. R is today the tool of choice for quantitative linguists due to its power, flexibility and expandability. In this workshop we are going to use R through an interface called R Studio which facilitates an enhanced user experience.

Aims

By the end of the workshop,

- participants will have produced a set of key descriptive statistics measures and visualisations for an example data set provided.
- Using guidance on handouts provided, participants should be able to apply similar techniques to produce descriptive statistics for their own data sets.
- Participants will be in a position to judge the potential of R and therefore whether they wish to invest in learning more about using R for descriptive statistics.

Topics

Basics	• Elements of the R Studio interface • Importing and exporting data into and out of R • Data manipulation in R: displaying, partially displaying, copying and creating data objects
Descriptive statistics I	• Data summarisation functions • central tendency and dispersion • checking distributions • frequency tables • layout, size formatting and file formats for graphs

Descriptive statistics II	Figures: • scatterplots, bar plots, histograms, line graphs, pie charts, boxplots, interaction plots, etc. • adjusting scales, adding axis labels and legends, titles, regression lines, dot shapes and colours, etc.

Prerequisites

No prior knowledge of R is assumed, but good computer skills and previous knowledge of basic descriptive statistics will be very helpful.

Software installation

For the workshop, university laptops with R and R Studio pre-installed will be supplied. *To install R and R Studio on other university-owned computers running Windows*, there is an installer in Cardiff Apps > Cardiff Apps > School Applications > ENCAP . To install the software on any other computer, download and install, in this order, R (http://www.stats.bris.ac.uk/R/) *and* R Studio (http://www.rstudio.com/). Both R and R Studio are free.

Reading List

No preparation is required for the workshop, but for keen participants, I would recommend chapters 2, 3 & 4 of Levshina (2015) as and introduction before the workshop, and Chang (2013) as a follow-up and resource for further learning.

Levshina, N. (2015). *How to do Linguistics with R: Data exploration and statistical analysis*, Amsterdam: Benjamins.

Chang, W. (2013). *R Graphics Cookbook*, Farnham: O'Reilly.

R Basics

First off, R is unforgiving about typos, so unless names of objects and everything else is typed exactly right, we will get errors or unexpected results.

1 Creating copying, importing and removing objects,

Objects are created using arrows to a name

```
AGE<-c(37,24,30,46) or c(8,6,5,10)->SCORES
c("m","f","m","f")->GENDER
```

The most useful data format is a data frame in R. To make a data frame out of existing variables:

```
our.data<-data.frame(GENDER, AGE, SCORES)
```

To import an existing data set (e.g. from an Excel spreadsheet, saved/exported as .csv file), use R Studio's ***Import Dataset*** button (in the 'Environment' pane) or, if the file is local:

```
our.data<-read.csv("path/to/file/location")
```

if the file is remote:

```
our.data<-read.csv("https://goo.gl/KK4qQ4")
```

to copy an object we do this: `name_of_object -> name_of_object_copy`

We remove objects like this: `rm(X)`

where 'X' is the object to be removed. The object disappears irretrievably after this command.

2 Exporting data frames

```
write.csv(X, file="FILENAME.csv",col.names=F)
```

X is the name of the data frame, FILNAME.csv is the name of the file you want to create.

3 Editing data frames

If you feel more confident doing this in Excel, that's fine, just export and re-import the data frame into R as seen above. Within R, the following command can be used to make minor changes:

```
fix(X)
```

where X is the name of the data frame. Make changes in the window that comes up, save and close. You can change the name of variables by clicking on them. Some edits (like removing or re-ordering columns or rows) cannot be done with fix(). See '6 change data frames' below for how to do such things.

4 Navigating data frames

Often we want to display only certain parts of a data frame, either because the whole thing is too big or because we want to use data in a sub-part in a certain function. Here's how to pick out subsets of values from a data frame (all commands are relative to the data frame our.data displayed on the right)

	AGE	GENDER	SCORES
1	37	m	8
2	24	f	6
3	30	m	5
4	46	f	10

Picking out values WITHOUT column names and row numbers (this only picks out the values themselves and this is usually what you want if you use the values as input to a function):
- we use the '$' sign after the name of data frame to specify the column name
- we can further specify the rows to be displayed in square brackets []

`our.data$AGE` displays the values of the variable AGE inside our.data

`our.data$AGE[c(1,4)]`
displays the values in row 1 AND 4 of the variable AGE inside our.data

`our.data$AGE[1:3]`
displays the values in row 1 TO 3 of the variable AGE inside our.data

`our.data$AGE[our.data$GENDER == "m"]`
displays the values of AGE where GENDER is 'm'

We can now put those values we pick out into a function like mean():

`mean(our.data$AGE[our.data$GENDER == "f"])`
displays the mean age of males in our data

5 Copy data frames (it's a good idea to make a backup copy before changing data frames)

To copy a data frame (for backup for example) we can export it (see above) or just put it it under a new name

`our.data->bkup.our.data`

now 2 identical data frames exist under our.data and bkup.our.data

6 Change the order of variables or cases in data frames

While this can be done within R, it is often easier to export to a spreadsheet application and re-importing into R. Here are some functions that can easily be performed within R:

To add a new column, we just tell R what data to put where, e.g.

`our.data$AGE*2` we display each value in AGE, multiplied by 2

`our.data$AGE*2->our.data$DBL.AGE` we put it into a column called DBL.AGE in our.data

`our.data[order(our.data$AGE),] -> our.data_ordered_by_age` this orders the data frame `our.data` by `AGE`

7 Converting variables between character, factor, ordered factor and numeric

Here is how we can make certain R uses the correct type for a variable

`c(1,2,3,4,5)-> a`

This creates a vector with numbers 1 to 4. This will automatically be a numeric type

`as.character(a)-> a` now the type is changed to character

`as.factor(a)-> a` now the type is changed to factor (= categorical variable)

`as.numeric(a)-> a` now the type is changed back to numeric (= interval variable)

To create an ordinal variable, we might do this

`ranks=c("first","third","second","first","third")` this created a

vector of character type
`ordered(ranks,c("first","second","third"))`
created a vector of type ordered factor (=ordinal variable). The 'ordered' function takes the data vector first, then then you need to indicate the ordering after the comma.

As a reminder, here are the levels of measurement:
Levels of measurement

- ratio scale (for present purposes no different from interval scale) *represented in R as numeric variables*
- Interval scale (values are scaled with equidistant intervals, e.g. 4 is twice as much as 2) *represented in R as numeric variables*
- Ordinal scale (values are ordered but not necessarily w/ equal intervals, e.g. 4th place is not (necessarily) twice 2nd place) *represented in R as ordered factors*
- Nominal / categorical scale (values cannot be ordered, just different, e.g. 'male' vs. 'female') *represented in R as factors*
- Frequencies: typically need to be treated as frequencies of categories, but can occasionally be abstracted into a 'measure' of an interval scale, e.g. number of letters in the orthographic form of a word as a measure of the length of a word. *Represented in R as numeric variables.*

8 Getting an overview

These functions give an overview of a data frame:

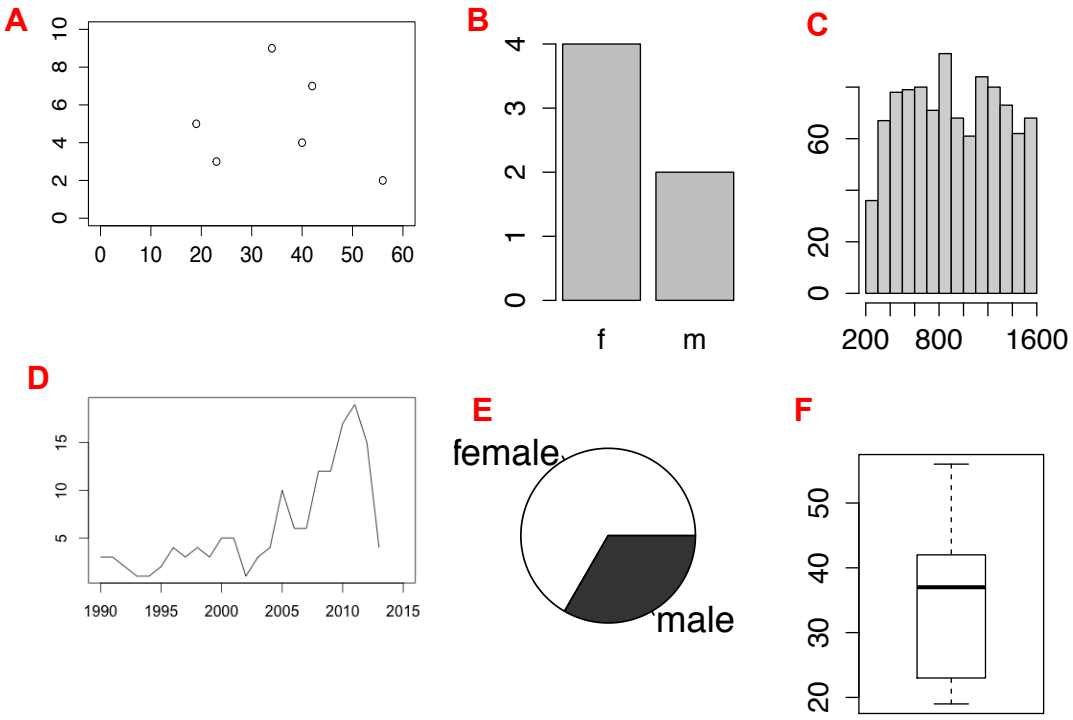
`length(X)`	gives the number of columns (or other elements) in X
`str(X)`	displays information about the data frame X
`summary(X)`	displays a summary of the data frame X

9 Obtaining basic descriptive metrics

`mean(X)` the mean of the variable X
`median(X)` the median of the variable X
`sort(table(X))` the mode of the variable X can be read off the output
`range(X)` or **`diff(range(X))`** the range of the variable X
`quantile(X)` the quartiles of variable X
`quantile(X)[4]-quantile(X)[2]` the interquartile range of X
`sd(X)` the standard deviation of the variable X
`var(X)` the variance of the variable X
`table(X)` / **`table(X,Y)`** / **`prop.table(table(X,Y))`** frequency tables

Figures

The following are some of the most often used graphs



a) scatter plot – for two interval/ratio variables
b) bar graphs – for (frequencies of) categorical variables
c) histograms – for interval/ratio variables
d) line graphs – to plot progressions over time of central tendencies of an interval/ ratio variable or frequencies of categorical variables
e) pie chart – for categorical variables
f) box plots – for interval/ratio variables

R code for graphs

Figure	Use	R command
scatterplot	typically two interval/ratio variables, although ordinal variables can be plotted here as well you can add a prediction line using abline(lm())	`plot(X, Y)` `abline(lm(Y~X))`
barplot	one or more categorical variables	`barplot(table(X,Y), beside=T, legend=c("a","b"))`
histogram	one interval/ratio variable that is continuous	`hist(X, breaks=10)`
line graph	on the x-axis you need a variable at least on an ordinal level, typically involving time periods on the y-axis you can either have the values of an interval/ratio variable for frequencies of a categorical one	`plot(type="l", X,Y)` `lines(type="l",X,Z)`
pie chart	one categorical variable	`pie(table(X),` `labels=c("a","b"))`

boxplot	one or more interval ratio variables	`boxplot(X,Y,Z)`
		`text(1:3, c(mean(X),`
		`mean(Y),mean(Z)),c("+","+","+")`

R commands to adjust graphs

`xlim=c(0,10), ylim=c(0,10)` to set the minimum and maximum values for x-axis (xlim) or y-axis (ylim)

`xaxt="n", yaxt="n"` suppress the drawing of x-axis (xaxt) or y-axis (yaxt); usually because we want to add those later using axis(), see below

`main="main title"` to supply a main title for the graph

`xlab="name", ylab="name"` to name the x-axis (xlab) or y-axis (ylab)

`col=c("white", "grey20", "grey60", "grey80", "black")` to define the colours with which variables are drawn. Include as many colours as you have variables)

In combination with plot(): **`type="l"`** this indicates: `"l"` = line (as opposed to points), `"b"` = both lines and points, `"s"` = stairs, `"h"` = histogram-type lines

In combination with plot(): **`pch=1`** point character; try out values 1 to 25 to see the different styles

`lty=1` line type, you can try out different values and see what they look like

`lwd=1` # the weight of lines drawn, a higher number draws a bolder line

To add an axis: (while drawing the plot, use xaxt="n" / yaxt="n" to suppress the automatic axes)

`axis(1, at=c(1,2,3), labels=c("a","b","c"))`

1=x-axis, 2=y-axis — where (at which values) on the axis to place tick marks — labels the tick marks with the labels provided

with any plot, the tick marks can be changed by using **`yaxp=c(0,3,3)`** or (**`xaxp`** for the x-axis) where the first number is the first (normally 0) and the second number the last tick mark, and the third number is the number of tick marks you'd like.