R programming

José R. Valverde
jrvalverde@cnb.csic.es
CNB/CSIC

Scripting

2024

Statistical facts
Natality rate is double the mortality rate. Therefore one out of every two persons is imnmortal.

May the “source” be with you

- Instead of writing all the commands each time, we can save them to a file.
- Remember
  - ***source( )*** tells R to read commands from the specified file
  - The reason is that files with commands are called “**source files**” (they are the “source” of the “stream” of commands for the computer)
  - R source files usually have a name ending in “.R”, but that is only a non-compulsory mnemotecnical aid.

Avoiding scripts

- We can expand R functionality to add new capacities
  - And share our expansions!
- There is a shared repository of such extensions written and shared by fellow humans: **CRAN**.
- install.packages(...)
  - Contacts CRAN and downloads and installs such an expansion
- library(...)
  - Tells R to use this expansion (and thus extend its functionality, the list of commands that it knows)

Handling data

Preparation

- Download the archive with the R exercises data files
- **exercises/R_exercises_datafiles.zip**
- Create a directory for it
- Extract the contents in said directory
- Make the directory your working directory
  - Launch R
- Or launch R and make that directory your working directory
  - using **`setwd('..how to go there..')`**

Opening an external file

- <u>**The safest way**</u> is to convert any file to CSV (or TAB) format
  - This allows you to inspect the file contents with a text editor
- Open cherry.xlsx in Excel or OOCalc
- Save the data as a CSV file with Save as...

```
cherry <- read.csv(file.choose(),
                           dec='.', sep=';')
cherry
```

- Choose the relevant file

Opening an Excel file

```
library(xlsx)
cherry <- read.xlsx(file.choose(), sheetIndex=1)
```

- This will open the chosen Excel (.xlsx) file and load the first spreadsheet in the file as “cherry”
- Inspect the contents

```
head(cherry)
cherry
plot(cherry)
summary(cherry)
```

NOTE: if you do not have an R extension you can add it with install.packages("name"), for instance: install.packages("xlsx", dependencies=TRUE)

)

Accessing named data

- Data in a dataset or frame can be labelled with names (e.g. column and row names)
- We refer to data by name using the “$” sign
  - `cherry$Girth`
  - `mean(cherry$Girth)`
  - `hist(cherry$Girth)`

Working with named data (2)

```
with(cherry, hist(Height))
with(cherry, mean(Height))
attach(cherry)
Girth                # now you see me
hist(Girth)
plot(Height, Volume)
detach(cherry)
Girth                # now you don't
```

Transforming variables

- Let us construct a new dataset with all data log-transformed

```
cherry1 <- transform(cherry,
                     logVolume=log(Volume),
                     logGirth=log(Girth))
cherry1
head(cherry1)
hist(cherry1$logVolume)
```

- **log( )** gives the natural logarithm, you can also use **log2( )** or **log10( )** instead.
- If your data contains zeros, you can use **log1p()** which computes log(x+1)

Selecting subsets

```
• cherry[3,]          # third row
• cherry[3:5,]        # rows 3 to 5
• cherry[-c(2,4),]    # all rows except
                      # rows 2 and 4
```

- Now try the following and see if you can understand what you get after each call.
- We are not assigning the result to a variable, but you certainly can, if you want to use the subset afterwards

Subsets

```
subset(cherry, Height>70)
subset(cherry, Height>=70)
subset(cherry, Height==80)
subset(cherry, Height==80,
       select=c(Girth,Volume))
subset(cherry, Height>80 & Girth>15)
subset(cherry, Height>80 | Girth>15)
```

Merging data

- Sometimes data comes from different sources and we need to merge it into a new dataset to facilitate analysis. Try the following and see what happens

```
newData <- data.frame(Girth=c(11.5, 17.0),
                      Height=c(71, 75),
                      Volume=c(22, 40))
newData
allData <- rbind(cherry, newData)
allData
```

- What does each command do?

Merging data from various sources

- Let us assume we have data from other source (we'll simply make it up this time)

```
precipitation <- rnorm(n=31, mean=50,
sd=10)
precipitation
allData2 <- cbind(cherry, precipitation)
allData2
```

- Obviously, variables should have the same order in both data sets.

Linear Regression

Linear regression

- Use the cherry dataset

```
plot(cherry$Girth, cherry$Volume)
lm(cherry$Volume ~ cherry$Girth)
lm(Volume ~ Girth, data=cherry)
```

  - Remember '~' means 'depending on', or 'by'
- This gives you the estimated intercept and slope. If we specify the dataset with data=, then variables will be sought in the dataset, if not, then we would have to use cherry$....

Using a linear model

- It is usually convenient to save the regression results for further analysis

```
linreg1 <- lm(Volume ~ Girth,
              data=cherry)
linreg1
abline(linreg1)
summary(linreg1)
confint(linreg1) # confidence
                 # intervals
```

LR explained

- The Coefficients part is the most interesting one
- It has one line for each parameter in the model with four columns
  - Estimate: the estimated value
  - Std. Error: the associated standard deviation
  - T value: the t-test statistic for the hypothesis that the value is zero
  - Pr(>|t|): p-value for the hypothesis that the parameter has no effect on the outcome

Model validation

- We can extract the fitted values, raw residuals and standardized residuals with functions **fitted()**, **residuals()** and **rstandard()** and use them to validate the model

```
plot(fitted(linreg1),
     residuals(linreg1))
plot(fitted(linreg1),
     rstandard(linreg1))
qqnorm(rstandard(linreg1))
abline(0,1)
```

Multiple Linear Regression

- Try to fit using two explanatory variables instead of one

```
linreg3 <- lm(log(Volume) ~ log(Girth) +
              log(Height), data=cherry)
```

  - Here ‘+’ does not mean ‘add’ but ‘and’: this would read as *log(Volume) depends on log(Girth) and on long(Height)*

- Check the model
- Compare the last two models with ANOVA

```
anova(linreg3, linreg2)
```

  We use ANOVA because we compare several coefficients

Example:

Canonical Correspondence Analysis

CCA

- We collect data from a number of observables under a number of experimental conditions and look for associations
  - E.g. a collection of control samples, a collection of samples with different treatments, etc... vs. observed species
- CCA looks for “linear combinations” of observations that associate with “linear combinations” of conditions.
- For instance: the association may be between a group of combined environmental variables (weather, soil, nutrients, latitude, i.e. a geographical region) and another group of combined variables (*phalaenopsis, pahiopedilum, cattleya*... i.e. orchids)

Biotechnology

- Let’s say we want to make a tropical garden in Atocha’s station
- We could try to reproduce **all** tropical conditions (even artficial latitude?)
- Or we could seek which are the main conditions required for the growth of the main tropical plants.
- It is not a 1:1 correlation but a many:many one

An easy approach: **vegan**

```
install.packages('vegan')
library(vegan)
library(labdsv)

data(varespec)     # different species
data(varechem)     # different chemical soil
properties
vare.cca <- cca(varespec ~
        Baresoil+Humdepth+pH+N+P+K+Ca+Mg+S+Al+Fe,
        data=varechem)
vare.cca
plot(vare.cca)
summary(vare.cca)
anova(vare.cca, by='term') # try by='mar', by='axis'
```

Thank you

Questions?