R-Bootcamp: Series 3

Dr. Matteo Tanadini

February 2022

```
*** Solutions ***
```

1 Exercise: Fitting models

1.1 Question:

The `cars` built-in dataset has two columns: "speed" (of the car) and "distance" (needed to stop the travelling car). Perform the following tasks.

1. use the `str()` function to check that both columns are indeed numeric variables and to see how many observations are present in the data
2. create a new dataframe named `cars.short` that contains only the first 36 observations present in the original dataset.
3. add a new column that contains the values 1 to 6 repeated 6 times (for a total length of 36). Name this column "test_day"
4. display the data (you can use "base" graphs or the add-on packages {*lattice*} / {*ggplot2*}) (i.e. create a scatter plot for "dist" against "speed" and a boxplot for "dist" against "test_day"
5. fit a linear model where "dist" is the response variable, "speed" and "test_day" the predictors
6. look at the summary of the model, how good is the fit? (hint: look at the adjusted-r-squared value)
7. fit a model that only contains "speed" as predictor. Then compare the two models. What is the p-value of the ANOVA test?
8. plot the model diagnostics in one single device (hint: you may want to set the parameter "mfrow" via the `par()` function).
9. the very first plot produced by the command you just ran displays the residuals against the fitted values. Reproduce a equivalent graph with the `ggplot()` function

Going further (*)

10. get the regression coefficient from the fitted model. Use the corresponding extractor function first. Then use `str()` or `names()` to find out in which slot of the lm object the coefficients are stored and extract them as you would do when access a slot in a list.
11. get the adjusted-r-squared and the regression coefficients and store them in a list. Pay attention to the choice of the list name (remember must comprehensible and consistent)
12. use the `gam()` function in {*mgcv*} package to fit a Generalise Additive Model on the same data. Look at the summary of the fitted model and compare the adjusted-r-squared with the one of the linear model.

Answers

1.

```
?cars
```

```
str(cars)
```

```
'data.frame': 50 obs. of  2 variables:
 $ speed: num  4 4 7 7 8 9 10 10 10 11 ...
 $ dist : num  2 10 4 22 16 10 18 26 34 17 ...
```

There are 50 observations and both columns are numerical.

2.

```
cars.short <- cars[1:36, ]
nrow(cars.short)
```

```
[1] 36
```

3.

```
cars.short$test_day <- rep(1:6, times = 6)
```

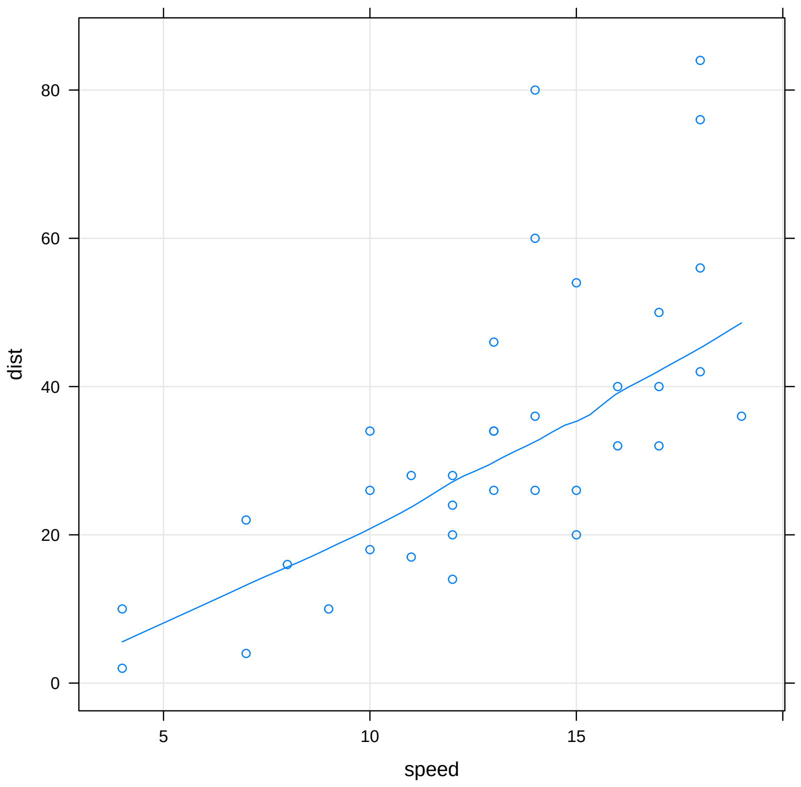
Actually, the argument "times" could be omitted and the `rep()` function would use "recycling to fill the column. Note also that there would be no warning message here, as the length of the final vector is a multiple of the shortest (i.e. 6 and 36).

```
cars.short$test_day_same <- rep(1:6)
head(cars.short, n = 13)
```

```
   speed dist test_day test_day_same
1      4    2        1             1
2      4   10        2             2
3      7    4        3             3
4      7   22        4             4
5      8   16        5             5
6      9   10        6             6
7     10   18        1             1
8     10   26        2             2
9     10   34        3             3
10    11   17        4             4
11    11   28        5             5
12    12   14        6             6
13    12   20        1             1
```

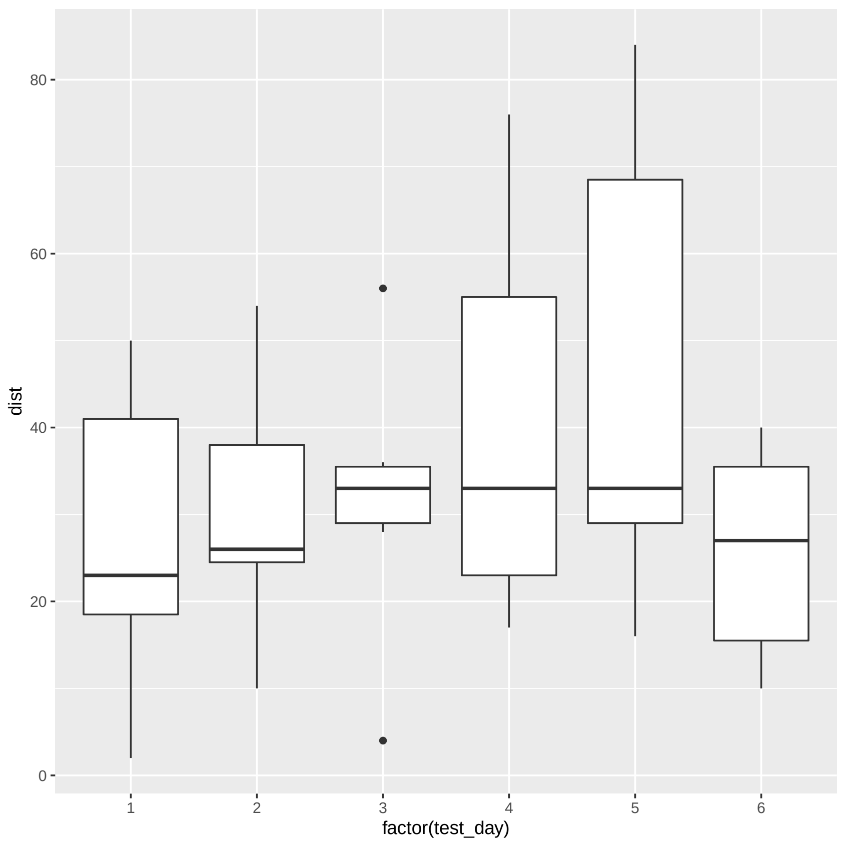
4.

```
library(lattice)
xyplot(dist ~ speed, data = cars.short,
       type = c("p", "smooth", "g"))
```



Here I decided to create a scatterplot with a smoother and a grid in the background.

```
library(ggplot2)
ggplot(data = cars.short,
       mapping = aes(y = dist,
                     x = factor(test_day))) +
  geom_boxplot()
```



Here I created a boxplot. Note that the function `ggplot()` requires factor to draw boxplots.

5.

```
lm.cars.short <- lm(dist ~ speed + test_day, data = cars.short)
```

6.

```
summary(lm.cars.short)


Call:
lm(formula = dist ~ speed + test_day, data = cars.short)

Residuals:
   Min     1Q Median     3Q    Max 
-20.09  -9.47  -1.75   5.34  43.21 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  -11.838      9.242   -1.28     0.21    
speed          3.568      0.650    5.49  4.3e-06 ***
test_day      -0.264      1.462   -0.18     0.86    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 15 on 33 degrees of freedom
Multiple R-squared:  0.485,Adjusted R-squared:  0.454 
F-statistic: 15.5 on 2 and 33 DF,  p-value: 1.76e-05
```

The adjusted-r-squared is about 0.45. Whether this is a good fit or not depends on the situation. Those of you that are familiar with the output of linear models, may have understood that day_test as been taken as a numeric variable here. We may want to consider this variable to be categorical here as there is no reason to believe that there is a linear trend in days. A simple solution to that would be to define a new varaible "test_day.factor" and refit the model.

```
cars.short$test_day.factor <- as.factor(cars.short$test_day)
str(cars.short)

'data.frame': 36 obs. of  5 variables:
 $ speed          : num  4 4 7 7 8 9 10 10 10 11 ...
 $ dist           : num  2 10 4 22 16 10 18 26 34 17 ...
 $ test_day       : int  1 2 3 4 5 6 1 2 3 4 ...
 $ test_day_same  : int  1 2 3 4 5 6 1 2 3 4 ...
 $ test_day.factor: Factor w/ 6 levels "1","2","3","4",..: 1 2 3 4 5 6 1 2 3 4 ...

##
lm.cars.short <- lm(dist ~ speed + test_day.factor, data = cars.short)
summary(lm.cars.short)


Call:
lm(formula = dist ~ speed + test_day.factor, data = cars.short)

Residuals:
   Min     1Q Median     3Q    Max 
-26.26  -9.46  -1.54   6.57  32.53 

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)       -15.570      9.012   -1.73    0.095 .  
speed               3.597      0.601    5.99  1.7e-06 ***
test_day.factor2    2.134      7.834    0.27    0.787    
test_day.factor3    1.069      7.855    0.14    0.893    
test_day.factor4    8.370      7.873    1.06    0.296    
test_day.factor5   12.671      7.896    1.60    0.119    
test_day.factor6   -9.727      7.956   -1.22    0.231    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 14 on 29 degrees of freedom
Multiple R-squared:  0.613,Adjusted R-squared:  0.533 
F-statistic: 7.66 on 6 and 29 DF,  p-value: 5.48e-05
```

Note that here we decided to overwrite the model because we won't need the old one any more as we realised it was not was we expected it to be. In other cases, we may want to keep the old model for future use and therefore overwriting is not a good option.

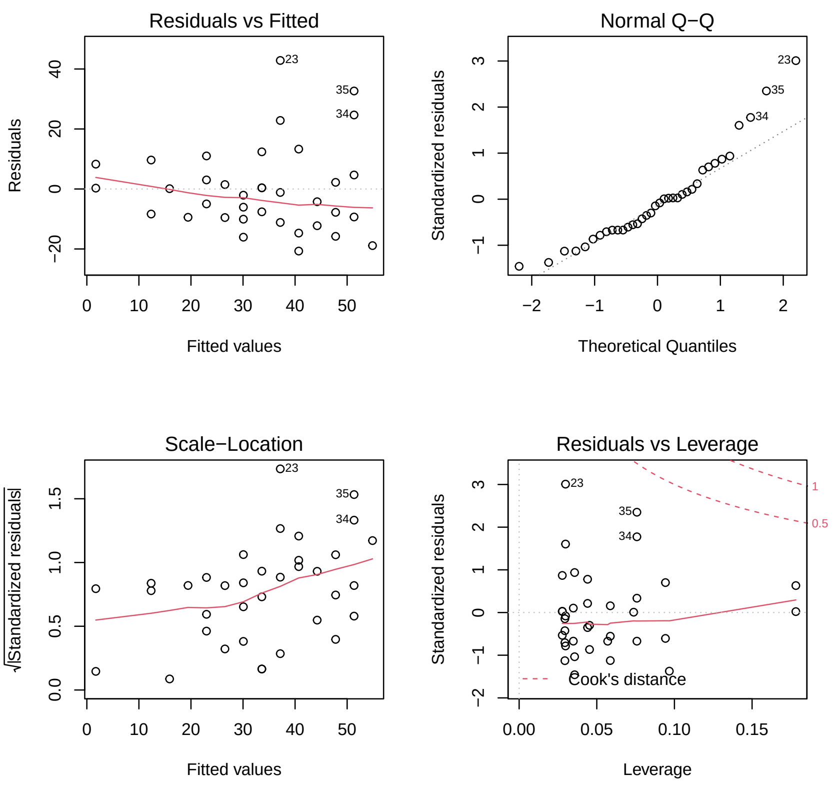
7.

```
lm.cars.short.speed.only <- update(lm.cars.short, . ~ . - test_day)
## same as
lm.cars.short <- lm(dist ~ speed, data = cars.short)
##
## compare the two models
anova(lm.cars.short.speed.only, lm.cars.short)

Analysis of Variance Table

Model 1: dist ~ speed + test_day.factor
Model 2: dist ~ speed
  Res.Df  RSS Df Sum of Sq    F Pr(>F)
1     29 5336                         
2     34 7110 -5     -1773 1.93   0.12
```

The p-value for this test is about 0.12.

8.

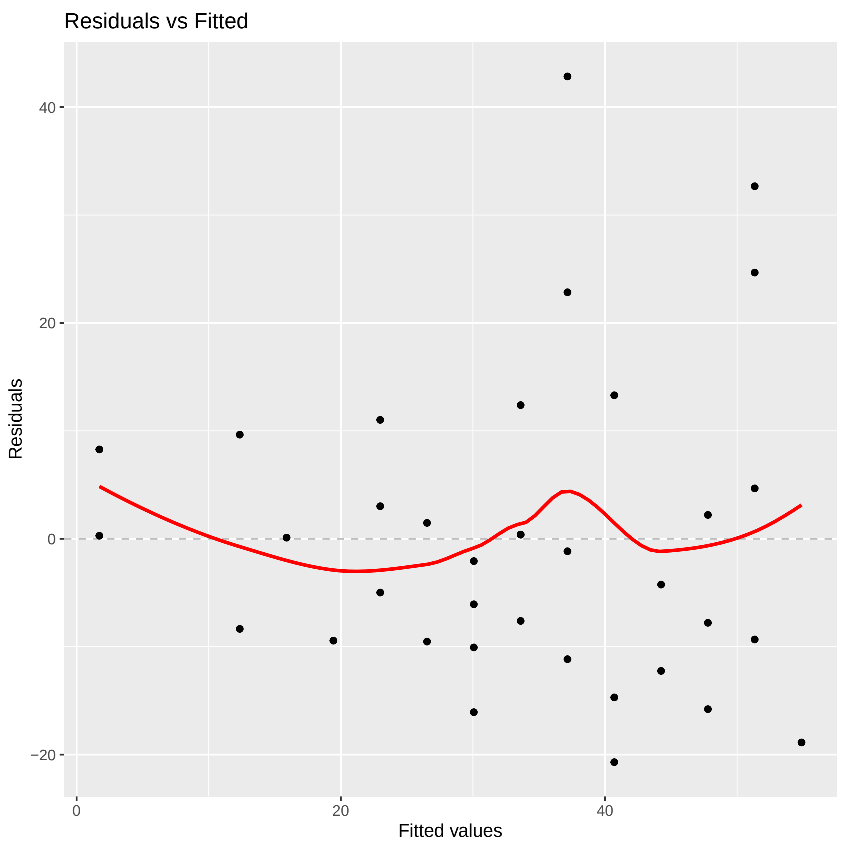
```
par(mfrow = c(2, 2))
plot(lm.cars.short)
```



9.

```
res.lm.cars.short <- resid(lm.cars.short)
fit.lm.cars.short <- fitted(lm.cars.short)
##
ggplot(mapping = aes(y = res.lm.cars.short,
                     x = fit.lm.cars.short)) +
  geom_hline(yintercept = 0, linetype = "dashed", col = "gray") +
  geom_point() +
  geom_smooth(se = FALSE, col = "red") +
  ylab("Residuals") +
  xlab("Fitted values") +
  ggtitle("Residuals vs Fitted")
```

```
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Note: here we named the residuals after the model.
Side note: You may have noticed that the smoother on the plot you created with the `ggplot()` function is more wiggly. The reason is simply that they use different methods.

Going further (*)

10. Let's get the regression coefficients with the extract function.

```
coef(lm.cars.short)

(Intercept)       speed
      -12.5         3.5
```

Let get them from the lm object.

```
names(lm.cars.short)
```

Ok, the first slot seems to contain the regression coefficients.

```
lm.cars.short$coefficients

(Intercept)       speed
      -12.5         3.5
```

11.

```
## where is the adjusted-r-squared?
names(lm.cars.short)

 [1] "coefficients" "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"       "qr"            "df.residual"
 [9] "xlevels"       "call"         "terms"         "model"

## mhmm, not there...
names(summary(lm.cars.short))

 [1] "call"          "terms"         "residuals"     "coefficients"
 [5] "aliased"       "sigma"         "df"            "r.squared"
 [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
## ok, here it is
##
## put everything in a list
list.cars.short <- list("Regression coefficients" = coef(lm.cars.short),
                        "Adj-R-Squared" = summary(lm.cars.short)$adj.r.squared)
##
list.cars.short
$`Regression coefficients`
(Intercept)       speed
      -12.5         3.5

$`Adj-R-Squared`
[1] 0.47
```

12.

```
library(mgcv)
Loading required package:  nlme
This is mgcv 1.8-38.  For overview type 'help("mgcv-package")'.
gam.cars.short <- gam(dist ~ s(speed) + test_day.factor, data = cars.short)
summary(gam.cars.short)


Family: gaussian
Link function: identity

Formula:
dist ~ s(speed) + test_day.factor

Parametric coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)         31.00       5.58    5.56  5.4e-06 ***
test_day.factor2     2.13       7.83    0.27     0.79
test_day.factor3     1.07       7.85    0.14     0.89
test_day.factor4     8.37       7.87    1.06     0.30
test_day.factor5    12.67       7.90    1.60     0.12
test_day.factor6    -9.73       7.96   -1.22     0.23
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
         edf Ref.df    F p-value
s(speed)   1      1 35.9 1.9e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.533   Deviance explained = 61.3%
GCV = 228.42  Scale est. = 184.01    n = 36
##
list.cars.short[[2]]
[1] 0.47
```

The two adjusted-r-squared coefficients are the same. Indeed, the model fitted in this very case fully equivalent.

1.2 Question:

The `airquality` built-in dataset has several columns. Fit a linear model where you try to predict "Ozone" concentration from "Solar.R" and "Wind". Comment on the data and results.

Answers

Lets's give a look at the data

```
str(airquality)

'data.frame': 153 obs. of  6 variables:
 $ Ozone  : int  41 36 12 18 NA 28 23 19 8 NA ...
 $ Solar.R: int  190 118 149 313 NA NA 299 99 19 194 ...
 $ Wind   : num  7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...
 $ Temp   : int  67 72 74 62 56 66 65 59 61 69 ...
 $ Month  : int  5 5 5 5 5 5 5 5 5 5 ...
 $ Day    : int  1 2 3 4 5 6 7 8 9 10 ...
```

```
head(airquality)

  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
```

There seems to be some values which are set to `NA`! Let's fit the model.

```
lm.airquality <- lm(Ozone ~ Solar.R + Wind, data = airquality)
summary(lm.airquality)


Call:
lm(formula = Ozone ~ Solar.R + Wind, data = airquality)

Residuals:
   Min     1Q Median     3Q    Max
-45.65 -18.16  -5.96  18.51  85.24

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  77.2460     9.0675    8.52  1.1e-13 ***
Solar.R       0.1004     0.0263    3.82  0.00022 ***
Wind         -5.4018     0.6732   -8.02  1.3e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 25 on 108 degrees of freedom
  (42 observations deleted due to missingness)
Multiple R-squared:  0.449,Adjusted R-squared:  0.439
F-statistic: 44.1 on 2 and 108 DF,  p-value: 1e-14
```

The sentence "(42 observations deleted due to missingness)" found in the summary output explains everything. These `NA` values are actually missing values. Indeed, NA stands for Not Available.

2 Exercise: Missing values

2.1 Question:

*Create a dataset in a spreadsheet (e.g. excel) that contains empty cells. Make sure your dataset has both numeric and categorical variables that contain empty cells. Read the data in **R** and comment on how where the empty cells coded.*

Answers

We get the dataset we used for the democode.

```
d.testNAs_2 <- read.csv("../../../DataSets/dataset_NAs.csv",
                        na.strings = -999,
                        header = TRUE)
str(d.testNAs_2)
```

```
'data.frame': 5 obs. of  5 variables:
 $ FirstName : chr  "Gianni" "Julia" "James " "Andrea" ...
 $ FamilyName: chr  "Pestalozzi" "Mueller" "Watts" "Pagani" ...
 $ Age       : int  20 23 43 19 NA
 $ Gender    : chr  "M" "F" "M" "" ...
 $ Salary    : num  4.5 NA 120 5.5 6.7
```

```
d.testNAs_2
```

```
  FirstName FamilyName Age Gender Salary
1    Gianni Pestalozzi  20      M    4.5
2     Julia    Mueller  23      F     NA
3    James       Watts  43      M  120.0
4    Andrea     Pagani  19             5.5
5    Judith Kartoffeln  NA      F    6.7
```

We note that NA are only created for numeric variables. For categorical variables (i.e. factors) empty cells are are not directly coded as missing values.

As a matter of fact, the variable **Gender** has three levels and no missing values.

```
levels(d.testNAs_2$Gender)
```

```
NULL
```

```
##
table(d.testNAs_2$Gender, useNA = "always")
```

```

        F    M <NA>
   1    2    2    0
```

3 Exercise: Packages

3.1 Question:

1. how many packages are available on CRAN?
2. search on CRAN for the "MASS" package
   - is this a recommended package?
   - how many authors are there?
   - who is the maintainer? Is that a good reference?
   - is it true that we can find this package in the "econometric" task view on CRAN?
   - is there any companion book to this package?
3. go to the "Machine Learning" task view on CRAN and find what is the reference implementation of the "Random Forest" algorithm
4. go to CRAN and click on the "Contributed" section. There search for a tutorial on how to create **R** packages. Name a document that seems to be good

Answers

1. how many packages are available on CRAN?
   About 15,000
2. search on CRAN for the "MASS" package

- is this a recommended package?
  Yes (see "priority" at the top of the package page)
- how many authors are there?
  6
- who is the maintainer? Is that a good reference?
  Brian Ripley. He is supposed to be a good reference as he is professor at a Dept. of Statistics (Uni Oxford). Btw: he is one the biggest names in the **R**-world (now retired).
- is it true that we can find this package in the "econometric" task view on CRAN?
  Yes
- is there any companion book to this package?
  YES, "Modern Applied Statistics with S" (4th edition, 2002). This used to be the reference book for **R**, is now outdated

3. go to the "Machine Learning" task view on CRAN and find what is the reference implementation of the "Random Forest" algorithm?
   The package $\{randomForest\}$
4. go to CRAN and click on the "Contributed" section. There search for a tutorial on how to create **R** packages. Name a document that seems to be good
   There you can find a document named "Creating R Packages: A Tutorial" by Friedrich Leisch (another big name in the **R**-world). The document is supposed to be good as it was written by one of the **R**-core members (i.e. Leisch).

   Note that in the "Contributed" section you can find very many introductory documents. They can be very helpful, however, keep in mind that everyone can write such a document and therefore the quality is again variable.

3.2 Question:

In which package can you find the following objects:

1. the `ls()` function
2. the `plot()` function
3. a function that contains the word "sunflower"
4. a dataset named "swiss"
5. the `apropos()` function

Answers

1. the `ls()` function
   `find("ls")` returns "package:base"
2. the `plot()` function
   "package:graphics"
3. a function that contains the word "sunflower"
   `apropos("sunflower")` returns "sunflowerplot". Don't ask me what this function is about. I just found cool that a function can have such a name :)
4. a dataset named "swiss" "package:datasets"
5. the `apropos()` function "package:utils"

3.3 Question:

*Assume you realised that loading a package, say package "nlme", creates a conflict. Assumed that you don't really need this package, how can you unload it from your **R**-session?*

Answers

You can unload a package by unticking its check box in the ”Packages” pane in Rstudio. Alternatively, you can use the `detach()` function directly. For example, `detach("package:nlme", unload = TRUE)`. NB: don’t forget to use the quotation marks around package:nlme.

If you were to prefer the Rstudio way to remove a package, make sure you will copy and paste the corresponding **R**-command in your script!

Finally, note that every time you restart `R` all package must be reloaded again. You can force some packages to load be default whenever starting a new session, **however, this is bad practice. Indeed, every new session must start ”clean” with no data or packages being loaded by default.**

3.4 Question:

Assume you wrote a few functions yourself and want to write a package to store them. It that feasible? Could you then pass this package to someone else or are you forced to put it on CRAN?

Answers

Absolutely, you can write a package with your own functions and datasets (note however, that this is usually done by (very) experienced people). Indeed, you can also write a script that contains your functions and data and load these with the `source()` function.

You can pass the package to anyone who wants to have it. You only must pay attention to the operating system. Packages come in two format: one that is suitable for mac and Linux users and one that is suitable to windows users.

Finally, there is no need to put your package on CRAN. CRAN is the official repository, but you could very well keep your package for yourself of put it on a another repository (e.g. github or sourceforge.net)