

Appendix C

R help

<i>Content</i>	<i>Page</i>
Part I: Basic functionality	150
Part II: Commands for creating graphics	153
Part III: If-statements, loops, and functions	154
Part IV: Descriptive statistics and hypothesis testing	156

Part I: Basic functionality*1.1 Types of data*

numeric	Numbers
character	Words (text)
logical	TRUE or FALSE
factor	One of the above with predefined categories
Inf	Infinite
NaN	Not a number
NA	Not available

1.2 Assigning values to variables

There are multiple ways to assign a value to a variable. For example, these all do the same, which is assigning the value **1** to the variable **a**.

```
a = 1 or a <- 1
```

```
assign('a', 1)
```

```
a = b = 1
```

1.3 Data structures

vector	A one-dimensional data structure
matrix	A two-dimensional data structure
array	A multi-dimensional data structure
data frame	A data structure containing different types of data
list	A collection of different data structures

1.4 Vectors

c(1, 2, 3)	Combine the numbers 1, 2, and 3 in a vector
seq(1, 6, 2)	Create sequence from 1 to 6 in increments of 2
rep(1:3, 2)	Repeat 1 to 3, and do that 2 times
1:4	Vector of 1 to 4 (: is therefore making a vector)
paste(x, y)	Paste vectors x and y together
letters[1:5]	Vector of first 5 letters of the alphabet
sample(x, 5)	Gives a random sample of size 5 from of data x
length(x)	Indicates the length of x
cut(x, 5)	Divide x in vectors with length 5
append(x, c(4, 5))	Add the numbers 4 and 5 to vector x
x <- numeric()	Creates an empty vector x
sort(x)	Order vector x from low to high (default)

1.5 Matrices

```
matrix(1:9, 3, 3)
diag(x)
head(x, 2)
t(x)
rbind(x, y)
cbind(x, y)
```

Create a matrix of 1 t/m 9 with 3 rows and 3 columns
 Get the diagonal from matrix `x`
 Gives the first two rows of matrix or data frame `x`
 Gives transpose of matrix `x`
 Add rows from matrix `x` and `y` together
 Add columns from matrix `x` and `y` together

1.6 Data frames

```
data.frame('X' = x)
head(x, 2)
names(x)
colnames(x)
rownames(x)
```

Create a data frame with data `x` (`X` is the column name)
 Look at first two rows of data frame `x`
 Names of data frame `x`
 Column names of data frame `x`
 Row names of data frame `x`

1.7 Lists

```
x <- list()
x[['title']] <- m
```

Create an empty list `x`
 Insert structure `m` into list `x` (`title` is the new title)

1.8 Indexing

```
x[2]
x[1:5]
x[-1]
x[x > 5]
x[x > 3 & x < 6]
x[1:3, 1]
x[1:3, 1:4]
x$h or x['h']
```

Get the second element from the vector `x`
 Get the first to the fifth element from vector `x`
 Get all elements except first element from vector `x`
 Get all elements greater than 5 from vector `x`
 Get all values from `x` greater than 3 and less than 6
 Get first 3 values from the first column from data `x`
 Get first 3 values from the first four columns from `x`
 Select element `h` from data frame `x`

1.9 Operators

```
x == y
x != y
%%
```

Check if `x` equals `y`
 Check if `x` does not equal `y`
 The remainder of a division (e.g. `36%%5 = 1`)

1.10 Importing data and files

```
data('x')
file.choose()
read.table('x')
write.table(x)
read.csv('x')
write.csv(x)
```

Import data set `x`
 Get access to interface to select a file
 Read in a `.txt` file `x` from the working directory
 Writes data `x` to a `.txt` file from the working directory
 Reads in a `.csv` file `x` from the working directory
 Writes data `x` to a `.csv` file from the working directory

1.11 Basic functions

TAB

`ls()`

`rm(list = ls())`

`getwd()`

`setwd()`

`help(x)`

`str(x)`

`summary(x)`

`print('Hello')`

`round(x, digits = 2)`

`which.max(x)`

`which(x == 10)`

`unique(x)`

`length(x)`

`nrow(x)`

`ncol(x)`

Scrolling through functions beginning with that letter

See all variables available in the environment

Delete all variables in your environment

See the location of your working directory

Set the location of your working directory

Read help about the function `x`

Finds out the structure of data `x`

Gives summary of the object `x`

Print `Hello` to the output

Round number(s) in `x` to a specified number of digits

Indicates the place of the highest value of data `x`

Shows the place of each object in `x` that equals 10

Gives only the unique values in `x`

Gives the number of elements in `x`

Gives number of rows of matrix/data frame `x`

Gives number of columns of matrix/data frame `x`

1.12 Installing an add-on package

`install.packages('x')`

Installs package with name `x`

Part II: Commands for creating graphics*2.1 Creating a plot*

<code>plot(x, y, ...)</code>	The basic plot function. Can be extended by adding arguments in <code>...</code>
<code>points(x, y)</code>	Adds points with <code>x</code> and <code>y</code> values to an existing plot
<code>lines(x, y)</code>	Adds a line with <code>x</code> and <code>y</code> values to an existing plot
<code>abline()</code>	Adds a linear line to plot (see <code>?abline</code>)
<code>text(x, y, 'text')</code>	Add text to plot on <code>x</code> and <code>y</code> coordinates
<code>legend('bottomright')</code>	Inserts legend in the bottom right of the plot
<code>pdf('x')</code>	When called, writes all images to <code>.pdf</code> file <code>x</code>
<code>dev.off()</code>	Ends writing all created images to a file
<code>layout(matrix(1:6, 2, 3))</code>	Create the layout (for multiple figures)
<code>layout(1)</code>	Put every plot on one page
<code>curve(dnorm(x, 0, 1), -3, 3)</code>	Plot the curve of the standard normal distribution

2.2 Quick plot functions

<code>hist(x, ...)</code>	Creates a histogram of <code>x</code>
<code>dotchart(x, ...)</code>	Creates a point chart of <code>x</code>
<code>pairs(x, ...)</code>	Compares all variables of <code>x</code> with multiple plots
<code>boxplot(x, ...)</code>	Creates a box plot of <code>x</code>
<code>barplot(x, ...)</code>	Creates a bar plot of <code>x</code>

2.3 Additional plot arguments (to be entered in `...`)

<code>axes = FALSE</code>	Disable axis in plot
<code>add = TRUE</code>	Adds the created plot to the previous plot
<code>las = 1</code>	Rotates the labels on the <code>x</code> and <code>y</code> axes
<code>xlim = c(0, 1)</code>	Set the range of the <code>x</code> axis between 0 and 1
<code>ylim = c(0, 1)</code>	Set the range of the <code>y</code> axis between 0 and 1
<code>xlab = 'x-axis'</code>	Set the name of the <code>x</code> axis to <code>x-axis</code>
<code>ylab = 'y-axis'</code>	Set the name of the <code>y</code> axis to <code>y-axis</code>
<code>type = 'p'</code>	Creates a plot containing the data as points
<code>type = 'l'</code>	Creates a plot containing the data as line
<code>type = 'b'</code>	Creates both points and lines
<code>lty = 1</code>	Set the line type in the plot
<code>col = 'blue'</code>	Set the color in the plot to <code>blue</code>

Part III: If-statements, loops, and functions

3.1 If-statements

If -statements have the following structure:

```
if (condition){  
  # Perform an action  
}
```

Translation: If this **condition** is satisfied, then perform this action.

Note that **length(condition) == 1** must evaluate to **TRUE** .

3.2 If-else-statements

If-else -statements have the following structure:

```
if (condition){  
  # Perform an action  
} else {  
  # Perform a different action  
}
```

Translation: If this **condition** is satisfied, then perform this action. If this condition is not satisfied, then perform a different action.

Another **condition** can be added by putting another **if (condition)** after **else** .

```
if (condition){  
  # Perform an action  
} else if (other condition){  
  # Perform a different action  
}
```

Translation: If this **condition** is satisfied, then perform this action. If not, then check whether the **other condition** is satisfied. If the **other condition** is satisfied, perform a different action.

3.3 For-loops

for -loops have the following structure:

```
for(i in 1:4){  
  # Perform an action that needs to repeated  
}
```

Translation: For a specified number of times (**1:4**), perform this action on each iteration (**i**).

3.4 While-loops

while -loops have the following structure:

```
while(condition){  
  # Perform an action that needs to be repeated  
}
```

Translation: For an unspecified number of times, perform this action as long as the **condition** is **TRUE** .

3.5 Functions

Functions have the following structure:

```
myFunction <- function(x, ...){  
  # Perform action on input x to get result  
  return(result)  
}
```

Translation: Take the input **x** , perform some actions, and return the resulting **outcome** .

The function **myFunction** can then be called with the data in **x** using:

```
myFunction(x)
```

Part IV: Descriptive statistics and hypothesis testing**4.1 Descriptive statistics**

<code>mean(x)</code>	Gives the average of <code>x</code>
<code>median(x)</code>	Gives the median of <code>x</code>
<code>sum(x)</code>	Gives the sum of <code>x</code>
<code>min(x)</code>	Gives the minimum of <code>x</code>
<code>max(x)</code>	Gives the maximum of <code>x</code>
<code>sd(x)</code>	Gives the standard deviation of <code>x</code>
<code>var(x)</code>	Gives the variance of <code>x</code>
<code>cor(x, y)</code>	Gives the correlation between <code>x</code> and <code>y</code>
<code>table(x)</code>	Gives a frequency table of <code>x</code>

4.2 Simple hypothesis testing

<code>t.test(x, y)</code>	Performs a t-test on the data <code>x</code> (<code>y</code> is optional)
<code>cor.test(x, y)</code>	Performs a correlation test on the data <code>x</code> and <code>y</code>
<code>binom.test(x, n, p)</code>	Performs a binomial test on the data
<code>chisq.test(x, y)</code>	Performs a chi-square test on the data
<code>aov(formula, x)</code>	Performs an ANOVA on the data in <code>x</code> using formula

4.3 Regression

<code>lm(y ~ 1 + x)</code>	Regression model $y = \beta_0 + \beta_1 \times x$
<code>lm(y ~ 1 + x + z)</code>	Regression model $y = \beta_0 + \beta_1 \times x + \beta_2 \times z$
<code>lm(y ~ 0 + x)</code>	Regression model $y = \beta_1 \times x$
<code>lm(y ~ x:z)</code>	Regression model $y = \beta_0 + \beta_1 \times x \times z$
<code>lm(y ~ x * z)</code>	Regression model $y = \beta_0 + \beta_1 \times x + \beta_2 \times z + \beta_3 \times x \times z$
<code>lm(y ~ poly(x, 2))</code>	Regression model $y = \beta_0 + \beta_1 \times x^2$
<code>coef(x)</code>	Gives coefficients of regression model in <code>x</code>
<code>residuals(x)</code>	Gives residuals of regression model in <code>x</code>
<code>AIC(x)</code>	Gives the AIC value of regression model in <code>x</code>
<code>BIC(x)</code>	Gives the BIC value of regression model in <code>x</code>
<code>summary(x)</code>	Gives a summary of regression model in <code>x</code>
<code>confint(x)</code>	Gives the confidence interval of model <code>x</code>
<code>abline(x)</code>	When added to a plot, plots the regression line
<code>predict(x, newdata)</code>	Use the regression model in <code>x</code> to predict new data