R help

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 t/m 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)`	Create a matrix of 1 t/m 9 with 3 rows and 3 columns
`diag(x)`	Get the diagonal from matrix `x`
`head(x, 2)`	Gives the first two rows of matrix or data frame `x`
`t(x)`	Gives transpose of matrix `x`
`rbind(x, y)`	Add rows from matrix `x` and `y` together
`cbind(x, y)`	Add columns from matrix `x` and `y` together

1.6 Data frames

`data.frame('X'= x)`	Create a data frame with data `x` (`X` is the column name)
`head(x, 2)`	Look at first two rows of data frame `x`
`names(x)`	Names of data frame `x`
`colnames(x)`	Column names of data frame `x`
`rownames(x)`	Row names of data frame `x`

1.7 Lists

`x <- list()`	Create an empty list `x`
`x[['title']] <- m`	Insert structure `m` into list `x` (`title` is the new title)

1.8 Indexing

`x[2]`	Get the second element from the vector `x`
`x[1:5]`	Get the first to the fifth element from vector `x`
`x[-1]`	Get all elements except first element from vector `x`
`x[x > 5]`	Get all elements greater than 5 from vector `x`
`x[x > 3 & x < 6]`	Get all values from `x` greater than 3 and less than 6
`x[1:3, 1]`	Get first 3 values from the first column from data `x`
`x[1:3, 1:4]`	Get first 3 values from the first four columns from `x`
`x$h or x['h']`	Select element `h` from data frame `x`

1.9 Operators

`x == y`	Check if `x` equals `y`
`x != 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')`	Import data set `x`
`file.choose()`	Get access to interface to select a file
`read.table('x')`	Read in a `.txt` file `x` from the working directory
`write.table(x)`	Writes data `x` to a `.txt` file from the working directory
`read.csv('x')`	Reads in a `.csv` file `x` from the working directory
`write.csv(x)`	Writes data `x` to a `.csv` file from the working directory

1.11 Basic functions

`TAB`	Scrolling through functions beginning with that letter
`ls()`	See all variables available in the environment
`rm(list = ls())`	Delete all variables in your environment
`getwd()`	See the location of your working directory
`setwd()`	Set the location of your working directory
`help(x)`	Read help about the function `x`
`str(x)`	Finds out the structure of data `x`
`summary(x)`	Gives summary of the object `x`
`print('Hello')`	Print `Hello` to the output
`round(x, digits = 2)`	Round number(s) in `x` to a specified number of digits
`which.max(x)`	Indicates the place of the highest value of data `x`
`which(x == 10)`	Shows the place of each object in `x` that equals 10
`unique(x)`	Gives only the unique values in `x`
`length(x)`	Gives the number of elements in `x`
`nrow(x)`	Gives number of rows of matrix/data frame `x`
`ncol(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

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

2.2 Quick plot functions

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

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

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

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

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

4.2 Simple hypothesis testing

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

4.3 Regression

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