**R tutorial 1** *Cheat sheet*

**Utrecht Data School**

Version 1: October 2017

Written by Maranke Wieringa

|  |  |
| --- | --- |
| **General** | |
| Index | Starts at 1 |
| Comments | # This is a comment |

|  |  |
| --- | --- |
| **Variables/types** | |
| String | text <- "Text here" |
| Integer | x <- 6 |
| Boolean | aBool <- TRUE  aBool <- FALSE |
| Vector | aVector <- c(1, 2, 3, 4) |
| List | aList <- list(vec1, vec2) |
| Dataframe | aDF <- data.frame(vec1, vec2) |

|  |  |
| --- | --- |
| **Working with strings** | |
| paste(aString, "!", sep="") | "Text here!" | |
| "x" %in% aString | TRUE | |
| toupper(aString) | "TEXT HERE!" | |
| tolower(aString) | "TEXT HERE!" | |

|  |  |
| --- | --- |
| **Working with integers** | |
| x <- x + 3 | Addition | |
| x <- x - 3 | Subtraction | |
| x <- x \* 3 | Multiplication | |

|  |  |
| --- | --- |
| **Working with vectors** | |
| aVector[3] | Select object at index 3 | |
| aVector[-2] | Remove object at index 2 | |
| aVector[1:2] | Select items at index 1 & 2 | |
| c(aVector, vec2) | Append values using c() | |
| **Working with lists (cont.)** | | |
| ind <- which( aVector == 3)  aVector[indices] | Find index, and subset the vector using found index | |
| aVector[-1] | Remove item at first index | |
| 2 %in% aVector | TRUE | |

|  |  |
| --- | --- |
| **Working with lists** | |
| aList[[1]][2] | Select 2nd item from 1st vector | |
| aList[[1]][1:2] | Slicing list | |
| append(aList, list(aVector)) | Append to list | |

|  |  |
| --- | --- |
| **Working with dataframes** | |
| aDF$column1 | Select column | |
| aDF$column1[2] | Index dataframe | |
| DF2 <- rbind(aDF, aVector) | Adding row to dataframe | |

|  |  |
| --- | --- |
| **Comparison operators** | |
| == | Is equal or similar to. |
| != | Is not equal or similar to. |
| > | More/bigger than. |
| < | Less/smaller than. |
| >= | More/bigger than or equal to. |
| <= | Less/smaller than or equal to. |

|  |  |
| --- | --- |
| **Loops** | |
| while(condition){  # do something  # make sure you’ll be able to  break out of the loop} |
| for(x in y){  # do something} |

|  |  |
| --- | --- |
| **If/elif/else** | |
| if (\_\_\_){  # code goes  here  } else if (\_\_\_){  # code goes  here  } else{  # code goes  here  } | If is mandatory when using if/else if/else, can have 1. |
| Else if is optional, can have multiple. |
| Else is optional, can have 1. |

|  |  |
| --- | --- |
| **Printing** | |
| print("Hello, world!") | Use keyword ‘print’ to start a print statement. |
| print(6 \* 7) | Can do calculations inside print statement |
| print(paste(x, ": hello.", sep="")) | You can also print variables inside a string. To do so, separate the string and variable by comma’s, wrap in a paste() function. |

|  |  |
| --- | --- |
| **Other useful built-in functions** | |
| length(\_\_\_) | Specifies how many elements are in a data type. |
| typeof(\_\_\_) | Return the type of an object/variable. |
| mean(\_\_) | Return the mean |
| median(\_\_) | Return the median |
| quantile(\_\_\_) | Return quantiles |
| range(\_\_\_) | Return min/max |
| boxplot(\_\_\_) | Return box plot |
| hist(\_\_\_) | Return histogram |
|  |  |

**>> Continues on the other side. >>**

|  |  |
| --- | --- |
| **CSV** | |
| file <- "filepath/csvname.csv"  fileDF <- read.csv(file, header=TRUE, sep=";", stringsAsFactors = FALSE) | File path is the exact location of the file you want to read. (Replace back slashes with forward slashes).  Next, we specify if the file contains headers or not, and what the separator is (‘;’ for European CSVs, ‘,’ for American). Finally, we tell R that we want to import strings as strings, and not as factors (which is the default). This function imports CSV as a data frame. | |
| write.csv(newAwesome, file = "newAwesomeUS.csv",row.names=TRUE)  write.csv2(newAwesome, file = "newAwesomeEUR.csv",row.names=TRUE) | We can use the write.csv() and the write.csv2() function to write a data frame to a CSV. The difference between the two is that the write.csv() function works with the American separator, whereas the write.csv2() function works with an European separator. | |

|  |  |  |
| --- | --- | --- |
| **Errors** | | |
| Naming error | object '\_\_\_' not found | In this instance you’re using a name which is not declared anywhere. In other words, look into your names and see where there are discrepancies. Remember that Python is case-sensitive. | |
| Type error | Non-numeric argument to binary operator | A TypeError means that you’re trying to do something which the data types do not allow. In this instance it might help to look into the documentation on the datatype. The example here is that you’re trying to do calculations on a string. | |
| Syntax error | Unexpected symbol in "\_\_\_" | Ther’s something wrong with your syntax (e.g. your punctuation or the manner in which you have structured an if statement). | |

|  |  |
| --- | --- |
| **Delimiters of CSV** | |
| ‘;’ | Semi-colon (EU standard) |
| ‘,’ | Comma (US standard) |
| '\t' | Tab |
| ‘|’ | Pipes |
| ‘^’ | Caret |
|  |  |