

R Cheat Sheet

Adam Rawles

Defining variables

There are three different ways to define variables:

```
x <- 1
y = 2
assign("z", 3)
```

Testing data types

```
x <- 1
is.xxxxxxx(x)
#Tests if value is of a specific type
is(x)
#The first value returned is the data type
```

Converting data types

```
x <- as.xxxxxxxx(value)
```

Valid types include integer, numeric, character, logical and factor

Logical operators

```
x == y #Is equal to
x != y #Is not equal to
x > y #Is greater than
x < y #Is less than
x <! y #Is not less than
x >! y #Is not greater than
x == y | x == z #Or
x == y & x == z #And
x %in% vector/list
```

Converting to date

```
df$column <- as.Date(df$column, format = "%d/%m/%Y")
#Use this if the column is a character
df$column <- as.Date(df$column, origin = as.Date("01/01/1970", format = "%d/%m/%Y")) #Use this if the column is a number
```

Date format codes can be found online

Installing & loading packages

```
install.packages("name of package")
#You only need to call this once
library(nameofpackage) #Put this line in #any script that uses the package
```

Creating & subsetting vectors

```
x <- c(10,20,30,40,50)
x[1:3] #Returns values at indices 1 to 3
x[c(1,2,4)]
#Returns values at indices 1, 2, and 4
```

Creating & subsetting dataframes

```
df <- data.frame(numeric_col = c(1,2,3),
  character_col = c("My", "name", "is"), s
stringsAsFactors = TRUE/FALSE)
df$numeric_col #Returns just numeric_col
df[,1]
#Returns all rows in the first column
df[1,] #Returns only the first row
```

Loading data

Load data via .csv or .xlsx

```
read.csv("path_to_file", stringsAsFactors = TRUE/FALSE, header = TRUE/FALSE)
#Specify whether you want columns
#with strings as factors and
#if the file has headers
library(readxl)
#read_excel requires the readxl package
read_excel("path_to_file", col_names = TRUE/FALSE, sheet = "sheetname"/sheet_index)
```

Filtering

```
dataframe[dataframe$column ==/!=/</>/%in% value,]
subset(dataframe, dataframe$column ==/!=/</>/%in% value)
```

You can use any logical operator in your filter criteria Note: subset and the [] method may return the results in different data structures!

Plotting

Use plots to visualise data and its relationships

```
plot(x = df$column, y = df$column, main = "Title of plot", xlab = "X axis label", ylab = "Y axis label", pch = 1-25)
#Creates a plot of x against y
hist(df$column)
#Creates a histogram from df$column
```

R Cheat Sheet

Adam Rawles

For loops

Use for loops to do something with every value in a vector/list or to do something a certain number of times

```
for (identifier in vector/list){  
  do something  
}  
  
for (identifier in 1:somenumber){  
  do something  
}
```

If else statements

Use if else statements to only do something if one criteria (or more) is fulfilled

```
if (value1 == value2){  
  do this  
}  
else if (value1 < value3){  
  do this  
}  
else {  
  do something else  
}
```

Connecting to SQL Databases

```
library(odbc)  
con <- dbConnect(odbc(), Driver = "SQL Server", Server = e.g. "PSADS004", Database  
= e.g. "TrailBlazer", trusted_connection  
= TRUE)
```

Querying SQL Databases

Note: you must connect to a database first

```
result <- dbGetQuery(conn = con, statemen  
t = "SELECT * FROM table1...")
```

Summary statistics

```
mean(df$column) #Returns the mean  
median(df$column) #Returns the median  
sd(df$column)  
#Returns the standard deviation  
quantile(df$column)  
#Returns the 0, 25th, 50th, 100th quantile
```

```
summary(df)  
#Returns summary stats for all  
#columns in df
```

```
library(psych)  
describeBy(df, group)  
#Returns statistics by group
```

The group argument should be a factor

Functions

Use functions to repeat common actions and to help compartmentalise your code

```
function_name <- function(argument_1, arg  
ument_2, ...){  
  do something  
  return(return_value)  
}
```