Cheatography

R BASIC Cheat Sheet

by user111 via cheatography.com/37570/cs/11844/

Basic of basic

print	no need any other exp
exponentiation	^
modulo	%%
assignment	= or <-
equal sign	==
case sensitive	YES
or	|
double operators	&&, || only examine 1st elements in vectors

Factor

factor()	
2 types	nominal vs. ordinal

Data structures

vector	c() one dim-array, same data type, arithmetic calculations OK, names()
matrix	two dim-array, same data type, arithmetic calculations OK
data frame	two dim-object, different data types
list	1 dim object, different data types

df is a list where each element is a column so df[[col]] equals column

FUNCTION

fun <- function(arg1, arg2) {...}

3 elements: arguments, body, environment

return() stop execution and return a value, used to return early for special cases not routinely

functions are objects

descriptive, use consistent naming convention, no overriding existing object names, data args come first

functions can be arguments to other functions --> functional programming

pipe operator: %>%

FUNCTION (cont)

safely(), possibly(), quietly() - handling errors

side effects: beyond the results of a function, i.e. print output, plot, save files to disk

3 main problems: type-unstable functions([, sapply), non-standard evaluation, hidden arguments

stop("error msg", call. = FALSE)

view global options: options(), getOption-("digits")

purr - map(), map_dbl(), map_int(), etc. apply the same function to a list. Similar to the apply family - sapply(), lapply()
map2(.x, .y, .f, ...), pmap(.l, .f, ...) - multiple args
invoke_map(.f, .x = list(NULL), ...) - multiple functions
walk() similar to map() but for side effects

Basic functions

class()	str()
rbind(), cbind()	colSums(), rowSums(), nrow(), ncol()
data.frame(), subset()	order(), arrange(), rank()
file.path()	return file path
seq_along()	generate sequence for "for loop" can handle empty cases

Subsetting

starts from 1	
[2:5] equals to	c(2,3,4,5)
df[,1]	1st column
df[1,]	1st row
df[2,3]	2nd row & 3rd column
df$x	select element name x
ls[x] vs. ls[[x]] or $	sublist vs. element

IF Syntax

```
if (condition1) {
   ex1
} else if (condi tion2) {
   ex2
} else {
   ex3
}
```

FOR syntax

```
for (i in list) {
    pri nt(i)
}
for (i in 1:leng th( list)) {
    pri nt( lis t[[i]])
}
```

break & next

break - abandons the active loop: the remaining code in the loop is skipped and the loop is not iterated over anymore.
next - skips the remainder of the code in the loop, but continues the iteration.

Importing data

read.csv(), read.delim()

read.csv2(), read.delim2() - local diff. in decimal convention

read.table("foo.txt", header = TRUE, sep = "/", stringsAsFactors = FALSE)

readxl - read_excel("foo.xlsx", sheet=bar); lapply(excel_sheets("foo.xlsx"), read_excel, path = "foo.xlsx")

XLConnect - loadWorkbook(), getSheets(), readWorksheet() & a lots more

read SQL - DBI, dbListTables, dbReadTable, dbGetQuery(efficient!)

read JSON - jsonlite: fromJSON()

read stats. software - haven, foreign

packages: readr - similar to utils but simpler; data.table:fread - fast, convenient

By **user111**
cheatography.com/user111/

Not published yet.
Last updated 22nd May, 2017.