Snippets R: Generic Cheatsheet

R Cheatsheet

Load libraries

```
if (!require(testthat)) install.packages('testthat')
library(testthat)
```

Self learning

```
library("swirl")
```

Vectors

```
rep(c(0, 1, 2), each = 10)
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
```

```
rep(c(0, 1, 2), times = 10)
```

```
##  [1] 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2
```

Memory management

```
# Sample dataset of 1000 rows
some_df <- data.frame(rep(1:100, 10),
                      rep(101:200, 10),
                      rep(201:300, 10))

object.size(some_df) # Reports the memory size allocated to the object
```

```
## 13040 bytes
```

```
rm("some_df") # Removes only the object itself and not necessarily the memory allotted to it
gc() # Force R to release memory it is no longer using
```

```
##          used (Mb) gc trigger (Mb) max used (Mb)
## Ncells 486021 26.0    1057334 56.5   662594 35.4
## Vcells 899194  6.9    8388608 64.0  1802053 13.8
```

```
ls() # Lists all the objects in your current workspace
```

```
## character(0)
```

```
rm(list = ls()) # If you want to delete all the objects in the workspace and start with a clean slate
```

Apply functions

```
# https://purrr.tidyverse.org/reference/map.html
```

```r
library(dplyr)
myList <- mtcars[1:20,] %>%
  split(.$cyl) %>%
  map(~ lm(mpg ~ wt, data = .x)) %>%
  map_dfr(~ as.data.frame(t(as.matrix(coef(.)))))
```

Standard Evaluation (SE) vs. Non Standard Evaluation (NSE)

```r
df[x<10] #Standard Evaluation (SE)
df[get('x')<10] #Non Standard Evaluation (NSE)
```

```r
df %>% dplyr::filter(x < 10) #Standard Evaluation (SE)
df %>% dplyr::filter(!!rlang::sym("x")<10) #Non Standard Evaluation (NSE)

#https://www.r-bloggers.com/2020/03/variable-name-in-functions-its-easy-with-datatable/
```

```r
my_summary <- function(df, grouping_var){
  grouping_var <- enquo(grouping_var)

  df %>%
    group_by(!!grouping_var) %>%
    summarise(
      avg = mean(air_time),
      sum = sum(air_time),
      min = min(air_time),
      max = max(air_time),
      obs = n()
    )
}

my_summary(airline_df, origin)
```

```r
my_summary <- function(df, grouping_var, summary_var){
  grouping_var <- enquo(grouping_var)
  summary_var <- enquo(summary_var)
  summary_nm <- quo_name(summary_var)
  summary_nm_avg <- paste0("avg_",summary_nm)
  summary_nm_sum <- paste0("sum_",summary_nm)
  summary_nm_obs <- paste0("obs_",summary_nm)

  df %>%
    group_by(!!grouping_var) %>%
    summarise(
      !!summary_nm_avg := mean(!!summary_var),#using "vestigial operator" := instead of =
      !!summary_nm_sum := sum(!!summary_var),
      !!summary_nm_obs := n()
    )
}
my_summary(airline_df, origin, air_time)

#https://www.r-bloggers.com/2019/07/bang-bang-how-to-program-with-dplyr/
```

testthat

Prepare package

```
install.packages("testthat")

usethis::create_package("myPackageName")
usethis::use_test("myPackageName") # creates tests/testthat/test-mypackage.R
usethis::use_description(
  fields = list(Package = "myPackageName"),
  check_name = TRUE,
  roxygen = TRUE
)

usethis::use_package("zip", min_version = "1.0.0") # adds "Imports: zip (>= 1.0.0)" to DESCRiPTION file
```

Run tests

```
library(testthat)
test_that("multiplication works", {
  expect_equal(2 * 2, 4)
})
```

Run test coverage

```
library(covr)
devtools::load_all(".")
covr <- file_coverage("R/fahrenheit_to_celsius.R", "tests/testthat/test-myPackageName.R")
covr
report(covr)
```

Automate project setup

```
library(usethis)

# Create a new package -------------------------------------------------
path <- file.path(tempdir(), "mypkg")
create_package(path)

proj_activate(path)

# Modify the description ----------------------------------------------
use_mit_license("My Name")

use_package("ggplot2", "Suggests")

# Set up other files -------------------------------------------------
use_readme_md()

use_news_md()

use_test("my-test")

x <- 1
y <- 2
use_data(x, y)
```

```
# Use git ------------------------------------------------------------
use_git()
```

Call function multiple times

Single parameter

```
lapply(format_vec, function(f)
  write_dataset(
    dataset = mtcars,
    path = output_folder,
    format = f,
    partitioning = "cyl"
  ))

write_dataset_preset <- function(f) {
  write_dataset(
    dataset = mtcars,
    path = output_folder,
    format = f,
    partitioning = "cyl"
  )
}
lapply(format_vec, write_dataset_preset)
purrr::walk(format_vec, write_dataset_preset)
```

Multiple parameter

```
write_dataset_preset_multiple <- function(data, formating, partition_by=dplyr::group_vars(data)) {
  write_dataset(
    dataset = data,
    path = output_folder,
    format = formating,
    partitioning = partition_by
  )
}

lapply(X=format_vec, FUN=write_dataset_preset_multiple, data=mtcars)
purrr::walk(.x=format_vec, .f=write_dataset_preset_multiple, data=mtcars)
```