Why you should use functions

INTRODUCTION TO WRITING FUNCTIONS IN R

Richie Cotton
Curriculum Architect at DataCamp

The arguments to mean()

Mean has 3 arguments

- `x` : A numeric or date-time vector.
- `trim` : The proportion of outliers from each end to remove before calculating
- `na.rm` : Remove before calculating

Calling mean()

Pass arguments by position

```
mean(numbers, 0.1, TRUE)
```

Pass arguments by name

```
mean(na.rm = TRUE, trim = 0.1, x = numbers)
```

Common arguments by position, rare arguments by name

```
mean(numbers, trim = 0.1, na.rm = TRUE)
```

Analyzing test scores

```r
library(readr)
test_scores_geography_raw <- read_csv("test_scores_geography.csv")
```

```r
library(dplyr)
test_scores_geography_clean <- test_scores_geography_raw %>%
  select(person_id, first_name, last_name, test_date, score)
```

```r
library(readr)
test_scores_geography_raw <- read_csv("test_scores_geography.csv")

library(dplyr)
test_scores_geography_clean <- test_scores_geography_raw %>%
  select(person_id, first_name, last_name, test_date, score)
```

```r
library(readr)
test_scores_geography_raw <- read_csv("test_scores_geography.csv")

library(dplyr)
test_scores_geography_clean <- test_scores_geography_raw %>%
  select(person_id, first_name, last_name, test_date, score)
```

```r
library(readr)
test_scores_geography_raw <- read_csv("test_scores_geography.csv")

library(dplyr)
test_scores_geography_clean <- test_scores_geography_raw %>%
  select(person_id, first_name, last_name, test_date, score)
```

```r
library(readr)
test_scores_geography_raw <- read_csv("test_scores_geography.csv")

library(dplyr)
test_scores_geography_clean <- test_scores_geography_raw %>%
  select(person_id, first_name, last_name, test_date, score)
```

```r
library(readr)
test_scores_geography_raw <- read_csv("test_scores_geography.csv")

library(dplyr)
test_scores_geography_clean <- test_scores_geography_raw %>%
  select(person_id, first_name, last_name, test_date, score)
```

```r
library(readr)
test_scores_english_raw <- read_csv("test_scores_english.csv")

library(dplyr)
test_scores_english_clean <- test_scores_english_raw %>%
  select(person_id, first_name, last_name, test_date, score)
```

```r
library(readr)
test_scores_art_raw <- read_csv("test_scores_art.csv")

library(dplyr)
test_scores_art_clean <- test_scores_art_raw %>%
  select(person_id, first_name, last_name, test_date, score)
```

```r
library(readr)
test_scores_spanish_raw <- read_csv("test_scores_spanish.csv")

library(dplyr)
test_scores_spanish_clean <- test_scores_spanish_raw %>%
  select(person_id, first_name, last_name, test_date, score)
```

```r
library(readr)
test_scores_geography_raw <- read_csv("test_scores_geography.csv")

library(dplyr)
library(lubridate)
test_scores_geography_clean <- test_scores_geography_raw %>%
  select(person_id, first_name, last_name, test_date, score) %>%
  mutate(test_date = mdy(test_date))
```

```r
library(readr)
test_scores_english_raw <- read_csv("test_scores_english.csv")

library(dplyr)
library(lubridate)
test_scores_english_clean <- test_scores_english_raw %>%
  select(person_id, first_name, last_name, test_date, score) %>%
  mutate(test_date = mdy(test_date))
```

```r
library(readr)
test_scores_art_raw <- read_csv("test_scores_art.csv")

library(dplyr)
library(lubridate)
test_scores_art_clean <- test_scores_art_raw %>%
  select(person_id, first_name, last_name, test_date, score) %>%
  mutate(test_date = mdy(test_date))
```

```r
library(readr)
test_scores_spanish_raw <- read_csv("test_scores_spanish.csv")

library(dplyr)
library(lubridate)
test_scores_spanish_clean <- test_scores_spanish_raw %>%
  select(person_id, first_name, last_name, test_date, score) %>%
  mutate(test_date = mdy(test_date))
```

```r
library(readr)
test_scores_geography_raw <- read_csv("test_scores_geography.csv")

library(dplyr)
library(lubridate)
test_scores_geography_clean <- test_scores_geography_raw %>%
  select(person_id, first_name, last_name, test_date, score) %>%
  mutate(test_date = mdy(test_date)) %>%
  filter(!is.na(score))
```

```r
library(readr)
test_scores_english_raw <- read_csv("test_scores_english.csv")

library(dplyr)
library(lubridate)
test_scores_english_clean <- test_scores_english_raw %>%
  select(person_id, first_name, last_name, test_date, score) %>%
  mutate(test_date = mdy(test_date)) %>%
  filter(!is.na(score))
```

```r
library(readr)
test_scores_art_raw <- read_csv("test_scores_art.csv")

library(dplyr)
library(lubridate)
test_scores_art_clean <- test_scores_art_raw %>%
  select(person_id, first_name, last_name, test_date, score) %>%
  mutate(test_date = mdy(test_date)) %>%
  filter(is.na(score))
```

```r
library(readr)
test_scores_spanish_raw <- read_csv("test_scores_spanish.csv")

library(dplyr)
library(lubridate)
test_scores_spanish_clean <- test_scores_spanish_raw %>%
  select(person_id, first_name, last_name, test_date, score) %>%
  mutate(test_date = mdy(test_date)) %>%
  filter(!is.na(score))
```

Benefits of writing functions

Functions eliminate repetition from your code, which

- can reduce your workload, and
- help avoid errors.

Functions also allow code reuse and sharing.

Let's practice!

INTRODUCTION TO WRITING FUNCTIONS IN R

Converting scripts into functions

INTRODUCTION TO WRITING FUNCTIONS IN R

Richie Cotton
Curriculum Architect at DataCamp

A basic function template

```r
my_fun <- function(arg1, arg2) {
  # Do something
}
```

The signature

```r
          function(arg1, arg2)
```

The body

```r
                               {
  # Do something
}
```

```r
library(readr)
test_scores_geography_raw <- read_csv("test_scores_geography.csv")

library(dplyr)
library(lubridate)
test_scores_geography_clean <- test_scores_geography_raw %>%
  select(person_id, first_name, last_name, test_date, score) %>%
  mutate(test_date = mdy(test_date)) %>%
  filter(!is.na(score))
```

```r
library(readr)
test_scores_english_raw <- read_csv("test_scores_english.csv")

library(dplyr)
library(lubridate)
test_scores_english_clean <- test_scores_english_raw %>%
  select(person_id, first_name, last_name, test_date, score) %>%
  mutate(test_date = mdy(test_date)) %>%
  filter(!is.na(score))
```

```r
library(readr)
test_scores_art_raw <- read_csv("test_scores_art.csv")

library(dplyr)
library(lubridate)
test_scores_art_clean <- test_scores_art_raw %>%
  select(person_id, first_name, last_name, test_date, score) %>%
  mutate(test_date = mdy(test_date)) %>%
  filter(is.na(score))
```

```r
library(readr)
test_scores_spanish_raw <- read_csv("test_scores_spanish.csv")

library(dplyr)
library(lubridate)
test_scores_spanish_clean <- test_scores_spanish_raw %>%
  select(person_id, first_name, last_name, test_date, score) %>%
  mutate(test_date = mdy(test_date)) %>%
  filter(!is.na(score))
```

Make a template

```
import_test_scores <- function() {








}
```

Paste your script into the body

```r
import_test_scores <- function() {
  test_scores_geography_raw <- read_csv("test_scores_geography.csv")

  test_scores_geography_clean <- test_scores_geography_raw %>%
    select(person_id, first_name, last_name, test_date, score) %>%
    mutate(test_date = mdy(test_date)) %>%
    filter(!is.na(score))
}
```

Choose the arguments

```r
import_test_scores <- function(filename) {         # <- only 1 argument
  test_scores_geography_raw <- read_csv("test_scores_geography.csv")

  test_scores_geography_clean <- test_scores_geography_raw %>%
    select(person_id, first_name, last_name, test_date, score) %>%
    mutate(test_date = mdy(test_date)) %>%
    filter(!is.na(score))
}
```

Replace specific values with arguments

```
import_test_scores <- function(filename) {
  test_scores_geography_raw <- read_csv(filename)  # <- replace specific filename

  test_scores_geography_clean <- raw_data %>%
    select(person_id, first_name, last_name, test_date, score) %>%
    mutate(test_date = mdy(test_date)) %>%
    filter(!is.na(score))
}
```

Generalize variable names

```r
import_test_scores <- function(filename) {
  test_scores_raw <- read_csv(filename)   # <- variable names generalized

  test_scores_clean <- test_scores_raw %>%       # <- variable names generalized
    select(person_id, first_name, last_name, test_date, score) %>%
    mutate(test_date = mdy(test_date)) %>%
    filter(!is.na(score))
}
```

Remove the final assignment

```
import_test_scores <- function(filename) {
  test_scores_raw <- read_csv(filename)

  test_scores_raw %>%  # <- remove assignment
    select(person_id, first_name, last_name, test_date, score) %>%
    mutate(test_date = mdy(test_date)) %>%
    filter(!is.na(score))
}
```

Use your function

```r
test_scores_geography <- import_test_scores("test_scores_geography.csv")
test_scores_english <- import_test_scores("test_scores_english.csv")
test_scores_art <- import_test_scores("test_scores_art.csv")
test_scores_spanish <- import_test_scores("test_scores_spanish.csv")
```

Arguments of sample()

- `x` : A vector of values to sample from.
- `size` : How many times do you want to sample from `x` ?
- `replace` : Should you sample with replacement or not?
- `prob` : A vector of sampling weights for each value of `x` , totaling one.

Let's practice!

INTRODUCTION TO WRITING FUNCTIONS IN R

Y kant I reed ur code?

INTRODUCTION TO WRITING FUNCTIONS IN R

Richie Cotton
Curriculum Architect at DataCamp

dplyr verbs

`select()` *selects* columns

`filter()` *filters* rows

Function names should contain a verb

- get
- calculate (or maybe just calc)
- run
- process
- import
- clean
- tidy
- draw

lm() is badly named

- Acronyms aren't self-explanatory
- It doesn't contain a verb
- There are lots of different linear models

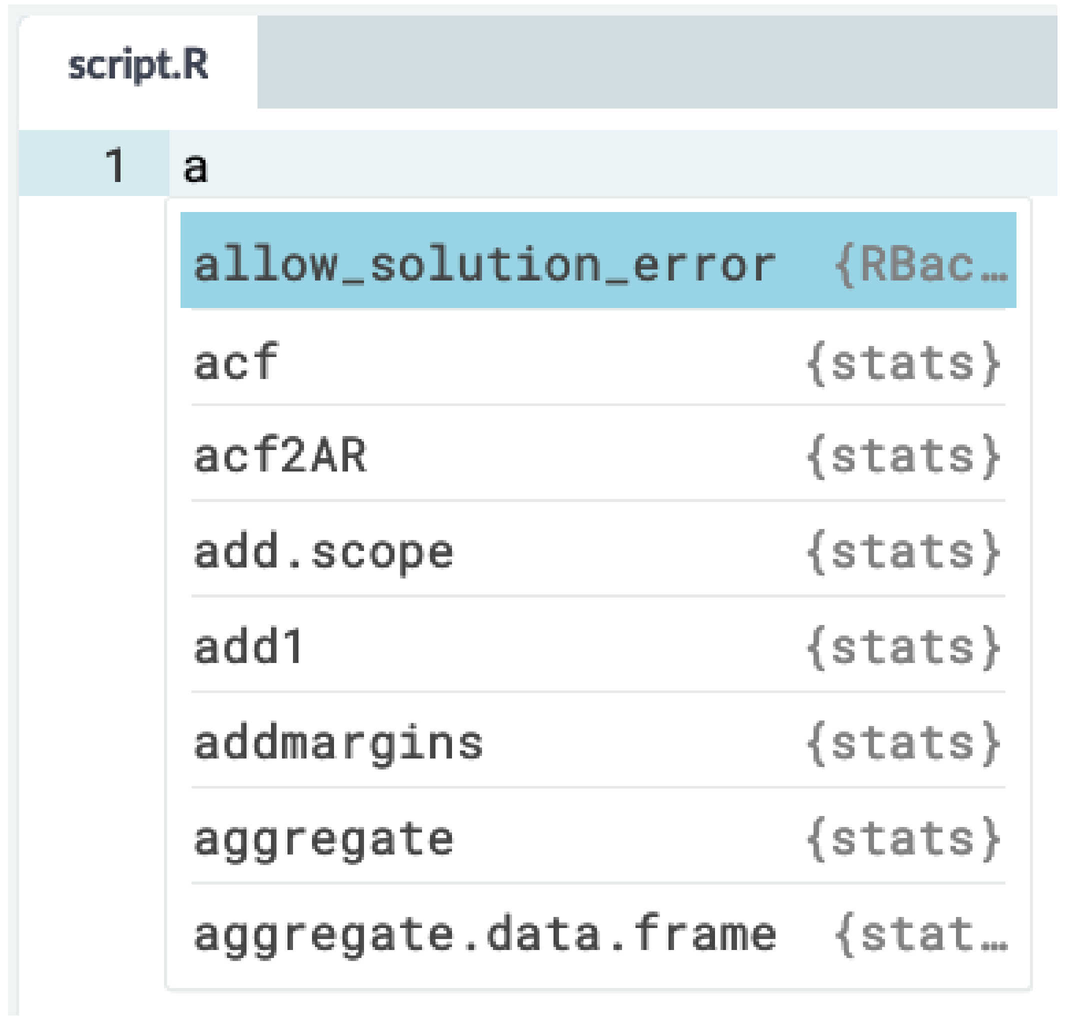
A better name would be `run_linear_regression()`

Readability vs. typeability

- Understanding code >> typing code

Readability vs. typeability

- Understanding code >> typing code
- Code editors have autocomplete


script.R

```
a
allow_solution_error   {RBac…
acf                    {stats}
acf2AR                 {stats}
add.scope              {stats}
add1                   {stats}
addmargins             {stats}
aggregate              {stats}
aggregate.data.frame   {stat…
```

Readability vs. typeability

- Understanding code >> typing code
- Code editors have autocomplete
- You can alias common functions

```
h <- head
```

```
data(cats, package = "MASS")
h(cats)
```

```
  Sex Bwt Hwt
1   F 2.0 7.0
2   F 2.0 7.4
3   F 2.0 9.5
4   F 2.1 7.2
5   F 2.1 7.3
6   F 2.1 7.6
```

Arguments of lm()

```r
args(lm)
```

```
function (formula, data, subset, weights, na.action, method = "qr",
    model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE,
    contrasts = NULL, offset, ...)
```

Types of argument

- **Data arguments:** what you compute on
- **Detail arguments:** how you perform the computation

```
args(cor)
```

```
function (x, y = NULL, use = "everything",
  method = c("pearson", "kendall", "spearman"))
```

Data args should precede detail args

This won't work

```
data %>%
  lm(formula)
```

because the data argument isn't first.

Our revised function for linear regression

```r
run_linear_regression <- function(data, formula) {
  lm(formula, data)
}
```

```r
cats %>%
  run_linear_regression(Hwt ~ Bet + Sex)
```

```
Call:
lm(formula = formula, data = data)

Coefficients:
(Intercept)          Bwt         SexM
    -0.4150       4.0758      -0.0821
```

Let's practice!

INTRODUCTION TO WRITING FUNCTIONS IN R