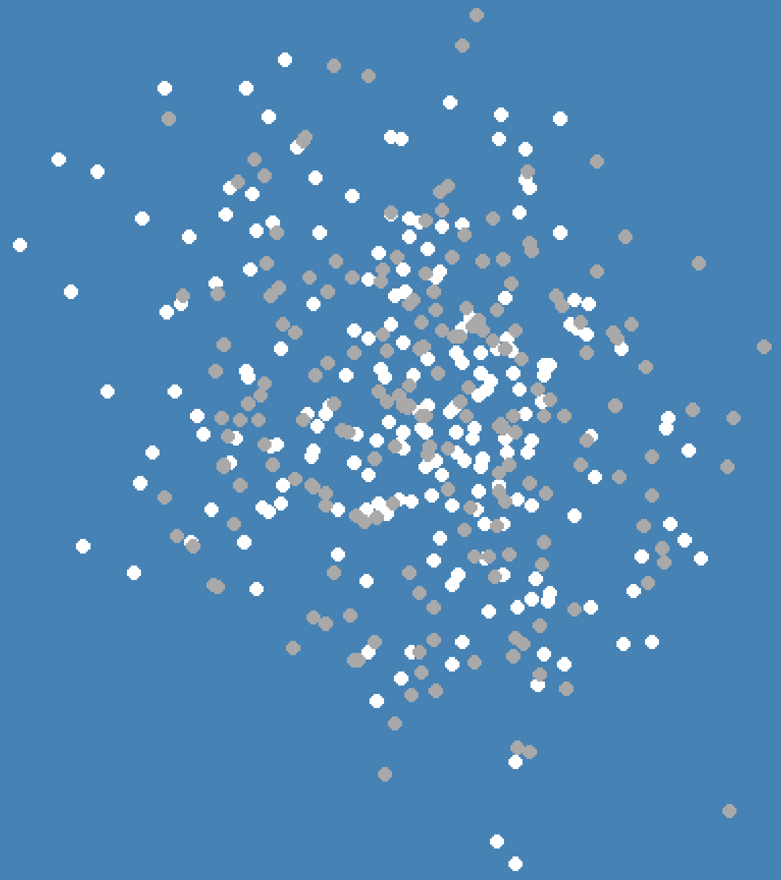
Introduction to R

6.2 Functions

Lion Behrens, M.Sc.



University of Mannheim
Chair of Social Data Science and Methodology
Chair of Quantitative Methods in the Social Sciences

A function takes an input, performs a pre-defined sequence of tasks, and returns an output.

Functions: Motivation

You already know functions really well, you have used them the whole time!

```
x <- seq(1, 20, 2)
```

```
print(x)
```

```
##  [1]  1  3  5  7  9 11 13 15 17 19
```

```
mean(x)
```

```
## [1] 10
```

Functions: Explanation

But how can I write my own user-defined functions?

The general syntax for defining a new function is this:

```
function_name <- function(input) {
  output <- perform action with input
  return(output)
}
```

Functions: Explanation

But how can I write my own user-defined functions?

The general syntax for defining a new function is this:

```
function_name <- function(input) {
  output <- perform action with input
  return(output)
}
```

Functions: Explanation

But how can I write my own user-defined functions?

The general syntax for defining a new function is this:

```
function_name <- function(input) {
  output <- perform action with input
  return(output)
}
```

Functions: Explanation

But how can I write my own user-defined functions?

The general syntax for defining a new function is this:

```
function_name <- function(input) {
  output <- perform action with input
  return(output)
}
```

Functions: Explanation

Let's write our own function to calculate the arithmetic mean of a numeric vector.

```
arith_mean <- function(input) {

}
```

Functions: Explanation

Let's write our own function to calculate the arithmetic mean of a numeric vector.

```
arith_mean <- function(vec) {

}
```

Functions: Explanation

Let's write our own function to calculate the arithmetic mean of a numeric vector.

```
arith_mean <- function(vec) {
  result <- sum(vec) / length(vec)
}
```

Functions: Explanation

Let's write our own function to calculate the arithmetic mean of a numeric vector.

```
arith_mean <- function(vec) {
  result <- sum(vec) / length(vec)
  return(result)
}
```

Functions: Explanation

Let's try out our function on the sequence of numeric values that we have generated before.

```
print(x)
```

```
##  [1]  1  3  5  7  9 11 13 15 17 19
```

```
arith_mean(x)
```

```
## [1] 10
```

```
mean(x)
```

```
## [1] 10
```

Defining your own custom functions can become really handy in your data-analytic workflow: Real Life Example

Prerequisite: Data Wrangling Pipeline (I/III)

```
library(tidyverse)
ess10 <- haven::read_dta("./dat/ESS10.dta")
ess10 <- ess10 %>% # subset variables
  select(country = cntry, # sociodemographics
         age = agea,
         gender = gndr,
         education_years = eduyrs,
         trust_social = ppltrst, # multidimensional trust
         trust_parliament = trstprl,
         trust_legalSys = trstlgl,
         trust_police = trstplc,
         trust_politicians = trstplt,
         trust_parties = trstprt,
         trust_EP = trstep,
         trust_UN = trstun,
         left_right = lrscale, # attitudes
         life_satisfaction = stflife,
         pol_interest = polintr,
         voted = vote, # turnout
         party_choice = prtvtefr # party choice
         ) %>%
  mutate_at(c("country", "gender", "voted", "party_choice"), as.character) %>% # change types
  mutate_at("pol_interest", as.numeric) %>% # change types
  filter(country == "FR") # subset cases (only include France)
```

Prerequisite: Data Wrangling Pipeline (I/III)

```
library(tidyverse)
ess10 <- haven::read_dta("./dat/ESS10.dta")
ess10 <- ess10 %>% # subset variables
  select(country = cntry, # sociodemographics
         age = agea,
         gender = gndr,
         education_years = eduyrs,
         trust_social = ppltrst, # multidimensional trust
         trust_parliament = trstprl,
         trust_legalSys = trstlgl,
         trust_police = trstplc,
         trust_politicians = trstplt,
         trust_parties = trstprt,
         trust_EP = trstep,
         trust_UN = trstun,
         left_right = lrscale, # attitudes
         life_satisfaction = stflife,
         pol_interest = polintr,
         voted = vote, # turnout
         party_choice = prtvtefr # party choice
         ) %>%
  mutate_at(c("country", "gender", "voted", "party_choice"), as.character) %>% # change types
  mutate_at("pol_interest", as.numeric) %>% # change types
  filter(country == "FR") # subset cases (only include France)
```

Prerequisite: Data Wrangling Pipeline (II/III)

```
ess10 <- ess10 %>%
  mutate(gender = recode_factor(gender,
                               `1` = "Male",
                               `2` = "Female"),
         voted = recode_factor(voted,
                              `1` = "Yes",
                              `2` = "No",
                              `3` = "Not eligible"),
         party_choice = recode_factor(party_choice,
                                     `1`  = "Lutte Ouvriére",
                                     `2`  = "Nouv. Parti Anti-Capitaliste",
                                     `3`  = "Parti Communiste Français",
                                     `4`  = "La France Insoumise",
                                     `5`  = "Parti Socialiste",
                                     `6`  = "Europe Ecologie Les Verts",
                                     `7`  = "La République en Marche",
                                     `8`  = "Mouvement Démocrate",
                                     `9`  = "Les Républicains",
                                     `10` = "Debout la France",
                                     `11` = "Front National",
                                     `12` = "Other",
                                     `13` = "Blank",
                                     `14` = "Null")
         )
```

Prerequisite: Data Wrangling Pipeline (III/III)

```
ess10 <- ess10 %>%
  mutate(education_years = na_if(education_years, 114), # set 114 to missing
         pol_interest = (pol_interest * -1) + 5, # invert scale
         life_satisfaction = life_satisfaction + 1 # change scale to [1, 11]
         ) %>%
  drop_na(trust_politicians, gender, education_years,
          life_satisfaction, pol_interest, age) # list-wise deletion of missings
```

Prerequisite: Data Wrangling Pipeline (III/III)

```
ess10 <- ess10 %>%
  mutate(education_years = na_if(education_years, 114), # set 114 to missing
         pol_interest = (pol_interest * -1) + 5, # invert scale
         life_satisfaction = life_satisfaction + 1 # change scale to [1, 11]
         ) %>%
  drop_na(trust_politicians, gender, education_years,
          life_satisfaction, pol_interest, age)
```

Functions: Real Life Example

Let's take our familiar regression model from Module 5.1 predicting trust in politicians from gender, education, life satisfaction and political interest.

Let's add age to the set of control variables.

How to include age?

```
m1 <-
  lm(trust_politicians ~ age + gender + education_years + life_satisfaction + pol_interest,
         data = ess10)
```

Functions: Real Life Example

Let's take our familiar regression model from Module 5.1 predicting trust in politicians from gender, education, life satisfaction and political interest.

Let's add age to the set of control variables.

How to include age?

```
m2 <-
  lm(trust_politicians ~ poly(age,2) + gender + education_years + life_satisfaction + pol_interest,
          data = ess10)
```

Functions: Real Life Example

Let's take our familiar regression model from Module 5.1 predicting trust in politicians from gender, education, life satisfaction and political interest.

Let's add age to the set of control variables.

How to include age?

```
m3 <-
  lm(trust_politicians ~ poly(age,3) + gender + education_years + life_satisfaction + pol_interest,
          data = ess10)
```

Functions: Real Life Example

Let's take our familiar regression model from Module 5.1 predicting trust in politicians from gender, education, life satisfaction and political interest.

Let's add age to the set of control variables.

How to include age?

```
m4 <-
  lm(trust_politicians ~ poly(age,4) + gender + education_years + life_satisfaction + pol_interest,
           data = ess10)
```

Functions: Real Life Example

The effect of age on trust in politicians is likely to be non-linear but we may have little idea how this non-linearity looks like.

Luckily, there is the Akaike Information Criterion (AIC).

Maximum Likelihood Estimation: $AIC = -2 * LogLikelihood + 2k$

Ordinary Least Squares Estimation: $AIC = n * log(\frac{SSE}{n}) + 2k$

Functions: Real Life Example

Let's program a function that

Functions: Real Life Example

Let's program a function that

- takes a statistical model

Functions: Real Life Example

Let's program a function that

- takes a statistical model
- re-calculates it n times trying out n different polynomial specifications for a specific control variable

Functions: Real Life Example

Let's program a function that

- takes a statistical model
- re-calculates it n times trying out n different polynomial specifications for a specific control variable
- stores the AIC of each model

Functions: Real Life Example

Let's program a function that

- takes a statistical model
- re-calculates it n times trying out n different polynomial specifications for a specific control variable
- stores the AIC of each model
- returns a nice overview table of which specification is most supported by the data

Functions: Real Life Example

```
poly_spec <- function(input) {


}
```

Functions: Real Life Example

```
poly_spec <- function(data, y, x, covariate, n) {


}
```

Functions: Real Life Example

```
data <- ess10
y <- "trust_politicians"
x <- "gender + education_years + life_satisfaction + pol_interest"
covariate <- "age"
n <- 4

poly_spec <- function(data, y, x, covariate, n) {


}
```

Functions: Real Life Example

```r
data <- ess10
y <- "trust_politicians"
x <- "gender + education_years + life_satisfaction + pol_interest"
covariate <- "age"
n <- 4

poly_spec <- function(data, y, x, covariate, n) {

  # prepare empty output table
  output <- matrix(data = NA,
                   nrow = n,
                   ncol = 2)
  colnames(output) <- c("Polynomial degree", "AIC")
  output[,"Polynomial degree"] <- 1:n

}
```

Functions: Real Life Example

```r
data <- ess10
y <- "trust_politicians"
x <- "gender + education_years + life_satisfaction + pol_interest"
covariate <- "age"
n <- 4

poly_spec <- function(data, y, x, covariate, n) {

  # prepare empty output table
  output <- matrix(data = NA,
                   nrow = n,
                   ncol = 2)
  colnames(output) <- c("Polynomial degree", "AIC")
  output[,"Polynomial degree"] <- 1:n

  # evaluate models
  library(stringr)
}
```

Functions: Real Life Example

```
data <- ess10
y <- "trust_politicians"
x <- "gender + education_years + life_satisfaction + pol_interest"
covariate <- "age"
n <- 4

poly_spec <- function(data, y, x, covariate, n) {

  # prepare empty output table
  output <- matrix(data = NA,
                   nrow = n,
                   ncol = 2)
  colnames(output) <- c("Polynomial degree", "AIC")
  output[,"Polynomial degree"] <- 1:n

  # evaluate models
  library(stringr)
  for (degree in 1:n) {
    model <-
      lm(as.formula(str_c(y, "~", x, "+ poly(", covariate, ",", degree, ")")),
         data = data)
  }
}
```

Functions: Real Life Example

```
data <- ess10
y <- "trust_politicians"
x <- "gender + education_years + life_satisfaction + pol_interest"
covariate <- "age"
n <- 4

poly_spec <- function(data, y, x, covariate, n) {

  # prepare empty output table
  output <- matrix(data = NA,
                   nrow = n,
                   ncol = 2)
  colnames(output) <- c("Polynomial degree", "AIC")
  output[,"Polynomial degree"] <- 1:n

  # evaluate models
  library(stringr)
  for (degree in 1:n) {
    model <-
      lm(as.formula(str_c(y, "~", x, "+ poly(", covariate, ",", degree, ")")),
         data = data)
  }
}
```

Functions: Real Life Example

```
data <- ess10
y <- "trust_politicians"
x <- "gender + education_years + life_satisfaction + pol_interest"
covariate <- "age"
n <- 4

poly_spec <- function(data, y, x, covariate, n) {

  # prepare empty output table
  output <- matrix(data = NA,
                   nrow = n,
                   ncol = 2)
  colnames(output) <- c("Polynomial degree", "AIC")
  output[,"Polynomial degree"] <- 1:n

  # evaluate models
  library(stringr)
  for (degree in 1:n) {
    model <-
      lm(as.formula(str_c(y, "~", x, "+ poly(", covariate, ",", degree, ")")),
         data = data)
    output[degree, "AIC"] <- AIC(model)
  }
}
```

Functions: Real Life Example

```
data <- ess10
y <- "trust_politicians"
x <- "gender + education_years + life_satisfaction + pol_interest"
covariate <- "age"
n <- 4

poly_spec <- function(data, y, x, covariate, n) {

  # prepare empty output table
  output <- matrix(data = NA,
                   nrow = n,
                   ncol = 2)
  colnames(output) <- c("Polynomial degree", "AIC")
  output[,"Polynomial degree"] <- 1:n

  # evaluate models
  library(stringr)
  for (degree in 1:n) {
    model <-
      lm(as.formula(str_c(y, "~", x, "+ poly(", covariate, ",", degree, ")")),
         data = data)
    output[degree, "AIC"] <- AIC(model)
  }

   return(output)
}
```

Functions: Real Life Example

```
poly_spec(data = ess10,
          y = "trust_politicians",
          x = "gender + education_years + life_satisfaction + pol_interest",
          covariate = "age",
          n = 4)
```

Functions: Real Life Example

```r
poly_spec(data = ess10,
          y = "trust_politicians",
          x = "gender + education_years + life_satisfaction + pol_interest",
          covariate = "age",
          n = 4)
```

```
##      Polynomial degree      AIC
## [1,]                 1 8123.444
## [2,]                 2 8089.051
## [3,]                 3 8090.770
## [4,]                 4 8089.617
```

Functions: Real Life Example

```
m2 <-
  lm(trust_politicians ~ poly(age,2) + gender + education_years + life_satisfaction + pol_interest,
          data = ess10)
```

Functions: Real Life Example

```
summary(m2)
```

```
##
## Call:
## lm(formula = trust_politicians ~ poly(age, 2) + gender + education_years +
##     life_satisfaction + pol_interest, data = ess10)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.4604 -1.4392  0.1297  1.4035  7.8595
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        1.02429    0.24607   4.163 3.29e-05 ***
## poly(age, 2)1     -3.78289    2.14067  -1.767   0.0774 .
## poly(age, 2)2     12.56091    2.07603   6.050 1.74e-09 ***
## genderFemale       0.22705    0.09407   2.414   0.0159 *
## education_years   -0.03189    0.01374  -2.321   0.0204 *
## life_satisfaction  0.24053    0.02147  11.202  < 2e-16 ***
## pol_interest       0.53679    0.05154  10.415  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.02 on 1897 degrees of freedom
## Multiple R-squared:  0.1381,    Adjusted R-squared:  0.1353
```

References

Parts of this course are inspired by the following resources:

- Wickham, Hadley and Garrett Grolemund, 2017. *R for Data Science - Import, Tidy, Transform, Visualize, and Model Data*. O'Reilly.
- Bahnsen, Oke and Guido Ropers, 2022. *Introduction to R for Quantitative Social Science*. Course held as part of the GESIS Workshop Series.
- Breuer, Johannes and Stefan Jünger, 2021. *Introduction to R for Data Analysis*. Course held as part of the GESIS Summer School in Survey Methodology.
- Teaching material developed by Verena Kunz, David Weyrauch, Oliver Rittmann and Viktoriia Semenova.