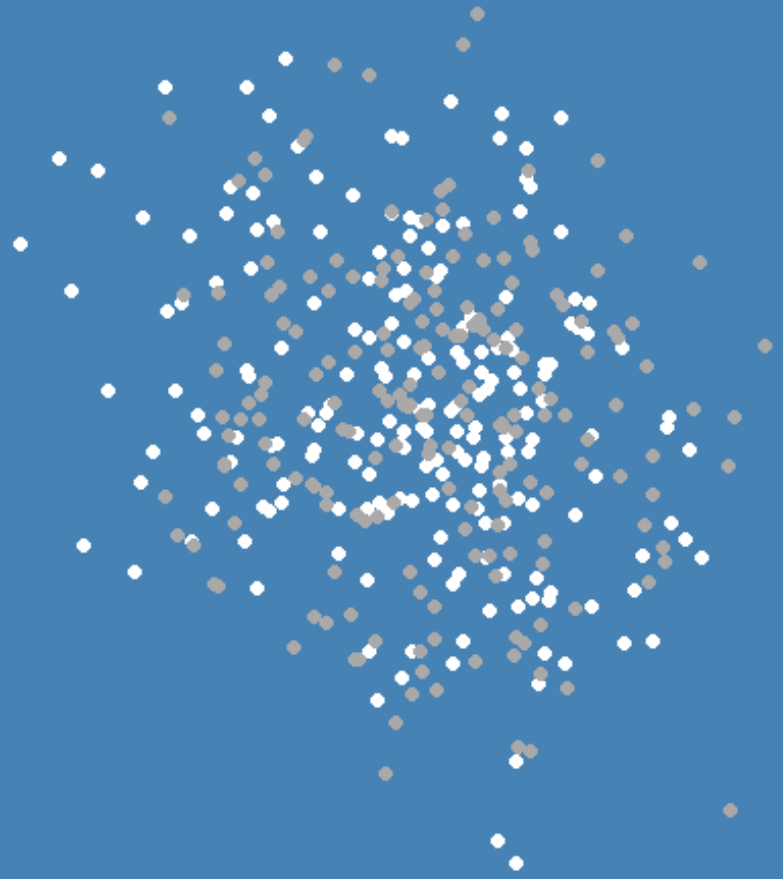
Introduction to R

6.1 For-loops

Lion Behrens, M.Sc.

UNIVERSITY OF MANNHEIM

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



For-loops are used to iteratively evaluate a sequence of expresssions that only differ in one parameter

Prerequisite: Data Wrangling Pipeline (I/III)

```
library(tidyverse)
ess10 <- haven::read_dta("./dat/ESS10.dta")
ess10 <- ess10 %>% # subset variables
  select(country = cntry, # sociodemographics
         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) # list-wise deletion of missings
```

For-loops: Motivation

For-loop: Motivation

Let's say we are, again, interested in predicting trust from gender, education, life satisfaction and political interest based on European Social Survey Data Wave 10.

This was our model from Module 5.1:

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

For-loop: Motivation

Let's say we are, again, interested in predicting trust from gender, education, life satisfaction and political interest based on European Social Survey Data Wave 10.

However, what if we also want to explain

- trust in political parties
- trust in the national parliament
- trust in the UN
- etc.?

```
col_ids <- stringr::str_detect(colnames(ess10), "trust")
colnames(ess10)[col_ids]
```

```
## [1] "trust_social"      "trust_parliament"  "trust_legalSys"    "trust_police"      "trust_politicians"
## [7] "trust_EP"          "trust_UN"
```

Why is manually setting up eight different regression models not a good idea?

For-loop: Motivation

A naive approach would look like this:

```
m1 <- lm(trust_social ~ gender + education_years + life_satisfaction + pol_interest,
         data = ess10)
m2 <- lm(trust_parliament ~ gender + education_years + life_satisfaction + pol_interest,
         data = ess10)
m3 <- lm(trust_legalSys ~ gender + education_years + life_satisfaction + pol_interest,
         data = ess10)
m4 <- lm(trust_police ~ gender + education_years + life_satisfaction + pol_interest,
         data = ess10)
m5 <- lm(trust_politicians ~ gender + education_years + life_satisfaction + pol_interest,
         data = ess10)
m6 <- lm(trust_parties ~ gender + education_years + life_satisfaction + pol_interest,
         data = ess10)
m7 <- lm(trust_EP ~ gender + education_years + life_satisfaction + pol_interest,
         data = ess10)
m8 <- lm(trust_UN ~ gender + education_years + life_satisfaction + pol_interest,
         data = ess10)
```

For-loops: Basic Explanation

For-loop: Explanation

The basic syntax of a for-loop in R is the following:

```
for (index in sequence) {
  expression
}
```

For-loop: Explanation

The basic syntax of a for-loop in R is the following:

```
for (index in sequence) {
  expression
}
```

For-loop: Explanation

The basic syntax of a for-loop in R is the following:

```
for (index in sequence) {
  expression
}
```

For-loop: Explanation

The basic syntax of a for-loop in R is the following:

```
for (i in seq(1,10,1)) {
  print(i)
}
```

For-loop: Explanation

The basic syntax of a for-loop in R is the following:

```
for (i in seq(1,10,1)) {
  print(i)
}
```

For-loop: Explanation

The basic syntax of a for-loop in R is the following:

```
for (i in seq(1,10,1)) {
  print(i)
}
```

For-loop: Explanation

The basic syntax of a for-loop in R is the following:

```
for (i in seq(1,10,1)) {
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
```

For-loop: Explanation

We can also adapt the expression:

```
for (i in seq(1,10,1)) {
  print(stringr::str_c("This is the ", i, "th Iteration"))
}
```

For-loop: Explanation

We can also adapt the expression:

```
for (i in seq(1,10,1)) {
  print(stringr::str_c("This is the ", i, "th Iteration"))
}
```

For-loop: Explanation

We can also adapt the expression:

```
for (i in seq(1,10,1)) {
  print(stringr::str_c("This is the ", i, "th Iteration"))
}
```

```
## [1] "This is the 1th Iteration"
## [1] "This is the 2th Iteration"
## [1] "This is the 3th Iteration"
## [1] "This is the 4th Iteration"
## [1] "This is the 5th Iteration"
## [1] "This is the 6th Iteration"
## [1] "This is the 7th Iteration"
## [1] "This is the 8th Iteration"
## [1] "This is the 9th Iteration"
## [1] "This is the 10th Iteration"
```

For-loop: Explanation

Or we adapt the sequence:

```
for (i in c(1, 3, 28, 105)) {
  print(stringr::str_c("This is the ", i, "th Iteration"))
}
```

For-loop: Explanation

Or we adapt the sequence:

```
for (i in c(1, 3, 28, 105)) {
  print(stringr::str_c("This is the ", i, "th Iteration"))
}
```

For-loop: Explanation

Or we adapt the sequence:

```
for (i in c(1, 3, 28, 105)) {
  print(stringr::str_c("This is the ", i, "th Iteration"))
}
```

```
## [1] "This is the 1th Iteration"
## [1] "This is the 3th Iteration"
## [1] "This is the 28th Iteration"
## [1] "This is the 105th Iteration"
```

For-loop: Explanation

Or we change the name of the index:

```
for (index in c(1, 3, 28, 105)) {
  print(stringr::str_c("This is the ", index, "th Iteration"))
}
```

For-loop: Explanation

Or we change the name of the index:

```
for (index in c(1, 3, 28, 105)) {
  print(stringr::str_c("This is the ", index, "th Iteration"))
}
```

For-loop: Explanation

Or we change the name of the index:

```
for (index in c(1, 3, 28, 105)) {
  print(stringr::str_c("This is the ", index, "th Iteration"))
}
```

For-loop: Explanation

Or we change the name of the index:

```
for (index in c(1, 3, 28, 105)) {
  print(stringr::str_c("This is the ", index, "th Iteration"))
}
```

```
## [1] "This is the 1th Iteration"
## [1] "This is the 3th Iteration"
## [1] "This is the 28th Iteration"
## [1] "This is the 105th Iteration"
```

For-loop: Explanation

Really any index name will work:

```
for (house in c(1, 3, 28, 105)) {
  print(stringr::str_c("This is the ", house, "th Iteration"))
}
```

For-loop: Explanation

Really any index name will work:

```
for (house in c(1, 3, 28, 105)) {
  print(stringr::str_c("This is the ", house, "th Iteration"))
}
```

```
## [1] "This is the 1th Iteration"
## [1] "This is the 3th Iteration"
## [1] "This is the 28th Iteration"
## [1] "This is the 105th Iteration"
```

For-loops: Example

For-loop: Example

A naive approach would look like this:

```
m1 <- lm(trust_social ~ gender + education_years + life_satisfaction + pol_interest,
         data = ess10)
m2 <- lm(trust_parliament ~ gender + education_years + life_satisfaction + pol_interest,
         data = ess10)
m3 <- lm(trust_legalSys ~ gender + education_years + life_satisfaction + pol_interest,
         data = ess10)
m4 <- lm(trust_police ~ gender + education_years + life_satisfaction + pol_interest,
         data = ess10)
m5 <- lm(trust_politicians ~ gender + education_years + life_satisfaction + pol_interest,
         data = ess10)
m6 <- lm(trust_parties ~ gender + education_years + life_satisfaction + pol_interest,
         data = ess10)
m7 <- lm(trust_EP ~ gender + education_years + life_satisfaction + pol_interest,
         data = ess10)
m8 <- lm(trust_UN ~ gender + education_years + life_satisfaction + pol_interest,
         data = ess10)
```

For-loop: Example

A sophisticated approach looks like this:

```
col_ids <- stringr::str_detect(colnames(ess10), "trust")
y_vars <- colnames(ess10)[col_ids]
print(y_vars)
```

For-loop: Example

A sophisticated approach looks like this:

```
col_ids <- stringr::str_detect(colnames(ess10), "trust")
y_vars <- colnames(ess10)[col_ids]
print(y_vars)
```

```
## [1] "trust_social"      "trust_parliament"  "trust_legalSys"    "trust_police"      "trust_politicians"
## [7] "trust_EP"          "trust_UN"
```

For-loop: Example

A sophisticated approach looks like this:

```
model_list <- list()
```

For-loop: Example

A sophisticated approach looks like this:

```
model_list <- list()
for (y_variable in y_vars) {

}
```

For-loop: Example

A sophisticated approach looks like this:

```
model_list <- list()
for (y_variable in y_vars) {
  model_list[[y_variable]] <-
    lm(as.formula(stringr::str_c(y_variable,
                                 "~ gender + education_years + life_satisfaction + pol_interest")),
       data = ess10)
}
```

```
summary(model_list$trust_social)
```

```
##
## Call:
## lm(formula = as.formula(stringr::str_c(y_variable, "~ gender + education_years + life_satisfaction + po
##     data = ess10)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7776 -1.2371  0.1996  1.3727  6.4276
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        1.19348    0.23861   5.002 6.20e-07 ***
## genderFemale       0.09018    0.09206   0.980    0.327
## education_years    0.09893    0.01262   7.837 7.63e-15 ***
## life_satisfaction  0.19244    0.02088   9.214  < 2e-16 ***
## pol_interest       0.24582    0.04918   4.998 6.32e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.981 on 1901 degrees of freedom
##   (1 Beobachtung als fehlend gelöscht)
## Multiple R-squared:  0.109,    Adjusted R-squared:  0.1071
## F-statistic: 58.13 on 4 and 1901 DF,  p-value: < 2.2e-16
```

```
summary(model_list$trust_politicians)
```

```
##
## Call:
## lm(formula = as.formula(stringr::str_c(y_variable, "~ gender + education_years + life_satisfaction + po
##     data = ess10)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.4468 -1.4587  0.1752  1.4482  8.4405
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        1.09014    0.24550   4.440 9.49e-06 ***
## genderFemale       0.23163    0.09474   2.445   0.0146 *
## education_years   -0.04214    0.01299  -3.243   0.0012 **
## life_satisfaction  0.25784    0.02150  11.994  < 2e-16 ***
## pol_interest       0.50653    0.05062  10.006  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.039 on 1902 degrees of freedom
## Multiple R-squared:  0.1197,    Adjusted R-squared:  0.1179
## F-statistic: 64.67 on 4 and 1902 DF,  p-value: < 2.2e-16
```

```
summary(model_list$trust_UN)
```

```
##
## Call:
## lm(formula = as.formula(stringr::str_c(y_variable, "~ gender + education_years + life_satisfaction + po
##     data = ess10)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.7575 -1.3991  0.1698  1.6207  6.2196
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        1.21867    0.28448   4.284 1.94e-05 ***
## genderFemale       0.37679    0.10929   3.448 0.000578 ***
## education_years    0.06630    0.01510   4.390 1.20e-05 ***
## life_satisfaction  0.28467    0.02482  11.472  < 2e-16 ***
## pol_interest       0.25699    0.05799   4.431 9.94e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.279 on 1781 degrees of freedom
##   (121 Beobachtungen als fehlend gelöscht)
## Multiple R-squared:  0.1092,    Adjusted R-squared:  0.1072
## F-statistic: 54.59 on 4 and 1781 DF,  p-value: < 2.2e-16
```

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.