Problem Set 1 Solution

Andreas Bender, Philipp Kopper, Philip Studener

26 October 2023

Ressources

Read chapter 2 & chapter 6.1 in The Art of R Programming. “extended examples” can generally be skipped. Paragraphs referencing matrices can also be skipped.

Exercises

Run this code before you start:

```r
bundesliga <- c(
    "FC Bayern"            = 55L,
    "BVB"                  = 51L,
    "RB Leipzig"           = 50L,
    "Borussia MGB"         = 49L,
    "Bayer 04"             = 47L,
    "FC Schalke 04"        = 37L,
    "VfL Wolfsburg"        = 36L,
    "SC Freiburg"          = 36L,
    "TSG Hoffenheim"       = 35L,
    "1. FC Köln"           = 32L,
    "Union Berlin"         = 30L,
    "Eintracht Frankfurt" = 28L,
    "Hertha Berlin"        = 28L,
    "FC Augsburg"          = 27L,
    "Mainz 05"             = 26L,
    "Fortuna Duesseldorf" = 22L,
    "Werder Bremen"        = 18L,
    "SC Paderborn"         = 16L)
```

As you may already have guessed: Our topic is soccer. In particular, this code creates a vector with the point standings in the Bundesliga before the COVID-19 break during the first wave. Note: You don’t need to use loops or to write functions to solve this exercise!

1. What is the data type (mode) of this vector? What is its scale (Skalenniveau)?

```r
typeof(bundesliga)
```

```
## [1] "integer"
```

The variable has a natural zero point (“0 points”) and a natural unit (points), therefore the variable is on the absolute scale.

2. Extract the first three teams from the `bundesliga` vector. Then extract the last three teams.

```
# Ersten drei:
bundesliga[1:3]
```

```
##  FC Bayern        BVB RB Leipzig
##         55         51         50
```

```
head(bundesliga, 3)
```

```
##  FC Bayern        BVB RB Leipzig
##         55         51         50
```

```
# Letzten drei:
bundesliga[(length(bundesliga) - 2):length(bundesliga)]
```

```
## Fortuna Duesseldorf       Werder Bremen        SC Paderborn
##                  22                  18                  16
```

```
tail(bundesliga, 3)
```

```
## Fortuna Duesseldorf       Werder Bremen        SC Paderborn
##                  22                  18                  16
```

3. Extract the points of the SC Freiburg, SC Paderborn and TSG Hoffenheim. Save the result in the vector `teams_selection`. Also compute the mean and sum of their points.

```
teams_selection <- bundesliga[c("SC Freiburg", "SC Paderborn", "TSG Hoffenheim")]
teams_selection
```

```
##    SC Freiburg   SC Paderborn TSG Hoffenheim
##             36             16             35
```

```
mean(teams_selection)
```

```
## [1] 29
```

```
sum(teams_selection)
```

```
## [1] 87
```

4. In the following we substract the mean of `teams_selection` from each value of `teams_selection` and divide the difference by `11.26943`. Is this transformation *allowed*? (*Hint*: think about the scale ("Skalenniveau") of the variable)?

```
(teams_selection - mean(teams_selection)) / 11.26943
```

```
##    SC Freiburg   SC Paderborn TSG Hoffenheim
##      0.6211494     -1.1535632      0.5324138
```

Solution: No, since `teams_selection` has a natural zero point and a natural unit (points). It is therefore a variable on the *absolute* scale.

5. Explain the code behaviour of the following chunks. *Hint*: Execute `?Comparison` in the console and read the help page.

```
bundesliga[1] == 55
```

```
## FC Bayern
##      TRUE
```

Solution: The first entry of the vector is compared with a double. This is true as the FCB actually has 55 points. For comparison, however, the integer on the left hand side is converted to double.

```r
bundesliga[1] == "55"
```

```
## FC Bayern
##      TRUE
```

Solution: The first entry of the vector is compared with a character. TRUE is returned. This is because the type is converted to character internally.

```r
bundesliga[2] > 55
```

```
##   BVB
## FALSE
```

Solution: The second entry of the vector is compared with a double. This is obv. false as the BVB has less than 55 points. For comparison, the integer on the left-hand side is converted to double.

```r
bundesliga[2] > "fifty-five"
```

```
##   BVB
## FALSE
```

```r
bundesliga[2] < "fifty-five"
```

```
##  BVB
## TRUE
```

Solution: When compared to characters, numerics get coerced to characters. Characters are compared lexicographically, so $1 < 2 < \ldots < a < b < c < \ldots$. Since numbers < letters, this returns FALSE. You can read up on character comparisons in the 2. paragraph of the Details in ?Comparison.

```r
TRUE + TRUE
```

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

```r
typeof(TRUE + TRUE)
```

```
## [1] "integer"
```

```r
TRUE > 3
```

```
## [1] FALSE
```

Solution: Boolean values (TRUE and FALSE) can also be represented via 1 (TRUE) and 0 (FALSE). If you use arithmetic operations with them they are converted to type integer or double, depending on the operation.

6. `wines` is a character vector of a selection of last year students' favorite wines/drinks.

```r
wines <- c(
  "sauvignon blanc", "chardonnay", "merlot", "pinot grigio", "riesling",
  "white", "zinfadel", "sauvignon blanc", "rose", "cabernet sauvignon",
  "primitivo", "saint emilion", "pinot blanc", "riesling", "white", "white",
  "lugana", "red", "merlot", "white", "sauvignon blanc", "amarone",
  "pinot gris", "elbling", "blanc de noirs", "merlot", "beer", "rioja")
```

Categorize the reported wines/drinks to categories "red", "white" and "other". Use the internet to become familiar with the different wines/drinks. (This course takes its educational purpose very seriously and does not limit it to programming.) Store the results in a new vector `wines_cat`. Make sure that it is a `factor` variable. As starting help we give you the category of the first 19 wines/drinks: `c("white", "white", "red", "white", "white", "white", "red", "white", "other", "red", "red", "red", "white", "white", "white", "white", "white", "red", "red")`

```r
wines_cat <- c(
  "w", "w", "r", "w", "w", "w", "r", "w", "o", "r", "r", "r", "w", "w", "w",
  "w", "w", "r", "r", "w", "w", "r", "w", "w", "o", "r", "o", "r")
wines_cat <- ifelse(wines_cat == "w", "white", wines_cat)
wines_cat <- ifelse(wines_cat == "r", "red", wines_cat)
wines_cat <- ifelse(wines_cat == "o", "other", wines_cat)
wines_cat <- as.factor(wines_cat)
```

7. Explain the following output:

- As what type are the values stored as internally?
- Why are they sorted (e.g. 3 refers to "white") this way?

```r
str(wines_cat)
```

```
##  Factor w/ 3 levels "other","red",..: 3 3 2 3 3 3 2 3 1 2 ...
```

Solution: They are internally stored as integers from 1 to 3. The ordering is due to the lexicographical order of the strings. Alternatively we could have used `factor` directly:

```r
wines_cat <- c(
  "w", "w", "r", "w", "w", "w", "r", "w", "o", "r", "r", "r", "w", "w", "w",
  "w", "w", "r", "r", "w", "w", "r", "w", "w", "o", "r", "o", "r")
wines_cat <- factor(
  wines_cat,
  levels = c("w", "r", "o"),
  labels = c("white", "red", "other"))
wines_cat
```

```
##  [1] white white red   white white white red   white other red   red   red  
## [13] white white white white white red   red   white white red   white white
## [25] other red   other red  
## Levels: white red other
```

```r
str(wines_cat)
```

```
##  Factor w/ 3 levels "white","red",..: 1 1 2 1 1 1 2 1 3 2 ...
```

8. Count the number of different wine/drink categories.

```r
table(wines_cat)
```

```
## wines_cat
## white   red other 
##    15    10     3
```

9. Assume that we explicitly want to introduce the category `Rose wine`. Change the category of the 9th vector element from `other` to `Rose wine`. Why does this not work?

```r
wines_cat[9] <- "Rose wines"
```

```
## Warning in `[<-.factor`(`*tmp*`, 9, value = "Rose wines"): invalid factor level,
## NA generated
```

Solution: `wines_cat` is a factor, "Rose wines" a character that is not part of `levels(wines_cat)`

10. Try to solve the problem. *Hint*: `?levels`.

```r
# you can also use relevel()
levels(wines_cat) <- c(levels(wines_cat), "Rose wines")
wines_cat[9] <- "Rose wines"
```

11. What is the scale (Skalenniveau) of the `wines_cat` variable?

Solution: `wines_cat` is on a nominal scale (Nominalskala).

12. Recreational sommelier P.K. has 8 wines in his cellar. The vector below gives their price range (in Euros).

```
wine_prices <- c("0-20", "20-100", "20-100", "0-20", "0-20", "0-20", "0-20", "100-200")
```

What is the scale of the variable `wine_prices`? Transform the vector from class `character` to a class appropriate for its scale.

Solution: ordinal scale

```
wine_prices <- factor(
  wine_prices,
  levels = c("0-20", "20-100", "100-200"),
  ordered = TRUE)
wine_prices
```

```
## [1] 0-20    20-100  20-100  0-20    0-20    0-20    0-20    100-200
## Levels: 0-20 < 20-100 < 100-200
```

```
typeof(wine_prices)
```

```
## [1] "integer"
```

```
class(wine_prices)
```

```
## [1] "ordered" "factor"
```

Session Info

```
sessionInfo()
```

```
## R version 4.2.2 (2022-10-31)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS Ventura 13.0
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.2-arm64/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] compiler_4.2.2  magrittr_2.0.3  fastmap_1.1.0   cli_3.4.1
##  [5] tools_4.2.2     htmltools_0.5.5 rstudioapi_0.14 yaml_2.3.6
##  [9] stringi_1.7.8   rmarkdown_2.25  knitr_1.40      stringr_1.4.1
## [13] xfun_0.40       digest_0.6.30   rlang_1.0.6     evaluate_0.17
```