# Advanced R

## Day 3

Sereina Herzog

Institute for Medical Informatics, Statistics and Documentation
Medical University of Graz

06.03.2024

# Course Content - Advanced R (Day 3)

- Long and wide format

# Course Content - Advanced R (Day 3)

- Long and wide format
- Function in R

# Wide and long

# Long/Wide format

- A dataset can be written in two different formats: wide and long

# Long/Wide format

- A dataset can be written in two different formats: wide and long
- A **wide format** has one line for each patient/animal + i.e., column with unique identifier has only unique entries

# Long/Wide format

- A dataset can be written in two different formats: wide and long
- A **wide format** has one line for each patient/animal + i.e., column with unique identifier has only unique entries
- A **long format** can have more more than one line for each patient/animal + i.e., column with unique identifier has recurring entries

# Long/Wide format - Example 1

| id | age | weight |
|---|---|---|
| P1 | 224 | 67 |
| P2 | 31 | 63 |
| P3 | 50 | 81 |
| P4 | 26 | 88 |

# Long/Wide format - Example 1

| id | parameter | value |
|---|---|---|
| P1 | age | 224 |
| P1 | weight | 67 |
| P2 | age | 31 |
| P2 | weight | 63 |
| P3 | age | 50 |
| P3 | weight | 81 |
| P4 | age | 26 |
| P4 | weight | 88 |

# Long/Wide format - Example 2

| id | sys1 | sys2 |
|---|---|---|
| P1 | 120 | 125 |
| P2 | 118 | 125 |
| P3 | NA | 110 |

# Long/Wide format - Example 2

| id | parameter | value |
|---|---|---|
| P1 | sys1 | 120 |
| P1 | sys2 | 125 |
| P2 | sys1 | 118 |
| P2 | sys2 | 125 |
| P3 | sys1 | NA |
| P3 | sys2 | 110 |

# Long/Wide format - why?

- each format has advantages and is more useful for certain tasks

# Long/Wide format - why?

- each format has advantages and is more useful for certain tasks
- long format
  - repeated measurements
    - especially if number of repetitions differ per patient/animal
    - long format needed for plots over repeated measurements
  - can be more efficient in terms of storage space

# Long/Wide format - why?

- each format has advantages and is more useful for certain tasks
- long format
  - repeated measurements
    - especially if number of repetitions differ per patient/animal
    - long format needed for plots over repeated measurements
  - can be more efficient in terms of storage space
- wide format
  - easier to read and look up a patient/animal
  - often easier to make calculations (e.g., BMI)

# Long/Wide format - reshape

in R package *tidyr*

- **pivot_wider()** "widens" data, increasing the number of columns and decreasing the number of rows

# Long/Wide format - reshape

in R package *tidyr*

- **pivot_wider()** "widens" data, increasing the number of columns and decreasing the number of rows
- **pivot_longer()** "lengthens" data, increasing the number of rows and decreasing the number of columns

# Example - pivot_wider()

```
dt_example <-
  tibble(id = rep(paste0("P", 1:3), 2),
         sys = c(120,118,NA, 125,125,110),
         measurement = c(1, 1, 1, 2, 2, 2))

dt_example %>%
    pivot_wider(data = ., names_from = measurement, values_from = sys)
```

# Example - pivot_longer()

```
dt_example <-
  tibble(id = paste0("P", 1:4),
         age = c(224, 31,50,26),
         weight = c(67, 63, 81, 88))

dt_example %>%
  pivot_longer(data = ., cols = c(age, weight))
```

# Functions in R

# Functions in R

We are constantly working with functions

# Functions in R

We are constantly working with functions

- a function takes argument(s)
- some argument(s) are mandatory
- some argument(s) have default values

# Functions in R

We are constantly working with functions

- a function takes argument(s)
- some argument(s) are mandatory
- some argument(s) have default values

```
mean(x = c(1, 2, 3))
```

# Functions in R

We are constantly working with functions

- a function takes argument(s)
- some argument(s) are mandatory
- some argument(s) have default values

```
mean(x = c(1, 2, 3))
```

- a function returns a ‘value’ (e.g., value, ggplot, . . . )

# Example mean()

mean() - calculate the mean value of a vector

# Example mean()

mean() - calculate the mean value of a vector

- mean() has several arguments

# Example mean()

mean() - calculate the mean value of a vector

- mean() has several arguments
- x: numeric/logical vectors (others also possible, e.g. time intervals)

# Example mean()

mean() - calculate the mean value of a vector

- mean() has several arguments
- x: numeric/logical vectors (others also possible, e.g. time intervals)
- trim: the fraction (0 to 0.5) of observations to be trimmed from each end of x before the mean is computed

# Example mean()

mean() - calculate the mean value of a vector

- mean() has several arguments
- x: numeric/logical vectors (others also possible, e.g. time intervals)
- trim: the fraction (0 to 0.5) of observations to be trimmed from each end of x before the mean is computed
- na.rm: a logical evaluating to TRUE or FALSE indicating whether NA values should be stripped before the computation proceeds

# Example mean()

mean() - calculate the mean value of a vector

- mean() has several arguments
- x: numeric/logical vectors (others also possible, e.g. time intervals)
- trim: the fraction (0 to 0.5) of observations to be trimmed from each end of x before the mean is computed
- na.rm: a logical evaluating to TRUE or FALSE indicating whether NA values should be stripped before the computation proceeds
- ... : further arguments passed to or from other methods

# Example mean()

mean() - calculate the mean value of a vector

- mean() has several arguments
- x: numeric/logical vectors (others also possible, e.g. time intervals)
- trim: the fraction (0 to 0.5) of observations to be trimmed from each end of x before the mean is computed
- na.rm: a logical evaluating to TRUE or FALSE indicating whether NA values should be stripped before the computation proceeds
- ... : further arguments passed to or from other methods

```
mean(x, trim = 0, na.rm = FALSE, ...)
```

# Write your own function

```
myFunction <- function() {

}
```

- object *myFunction* is now a function

# Write your own function

```
myFunction <- function() {

}
```

- object *myFunction* is now a function
- () → currently we have no arguments

# Write your own function

```
myFunction <- function() {

}
```

- object *myFunction* is now a function
- () → currently we have no arguments
- {} is the place for all commands we want

# Write your own function

```r
myFunction <- function(name) {
   greetings <- paste0("Hallo, ", name, "!")
   cat(greetings)
}
```

- *name* is our argument
- {} creates the *greetings* and returns the last line

# Write your own function

```
myFunction <- function(name) {
   greetings <- paste0("Hallo, ", name, "!")
   cat(greetings)
}
```

- *name* is our argument
- {} creates the *greetings* and returns the last line

```
myFunction(name = "Susi")
```

```
## Hallo, Susi!
```

# Write your own function - Example

```r
CreateSampleDataset <- function(nrow = 100) {
    Condition <- rbinom(n = nrow, size = 1, prob = 0.5)
    IQ <- rnorm(n = nrow, mean = 100, sd = 15)
    Age <- rnorm(n = nrow, mean = 40, sd = 7.5)
    Motivation <- runif(n = nrow, min = 1, max = 10)
    dfSampleData <- tibble(Condition, IQ, Age, Motivation)
    return(dfSampleData)
}
```

# Write your own function - Example

```r
myFunction2 <- function(x = 0) {
   x-3
   return(c(answer = 42))
}
```

# Write your own function - Example

```
myFunction2 <- function(x = 0) {
   x-3
   return(c(answer = 42))
}
```

```
myFunction2()
```

```
## answer
##     42
```

# Write your own function - Example

```
myFunction2 <- function(x = 0) {
   x-3
   return(c(answer = 42))
}
```

```
myFunction2()
```

```
## answer
##     42
```

- *return()*
  - variable can be any R object
  - a function in R can only ever return one object (use list() for more)

# Remarks

- Functions are used when the same or similar program code is used in several places in the script

# Remarks

- Functions are used when the same or similar program code is used in several places in the script
- Variables declared inside a function are local to that function

# Remarks

- Functions are used when the same or similar program code is used in several places in the script
- Variables declared inside a function are local to that function
- Function name should make it clear what the function is good for
- Inputs and outputs should be clear

# Links

# Links (I)

- Introduction to R
  - R for Data Science (https://r4ds.hadley.nz/)
- Plots using ggplot
  - Overview with further links to course material: https://ggplot2.tidyverse.org/
- Display tables using flextable
  - flextable bool https://ardata-fr.github.io/flextable-book/
  - Function references https://davidgohel.github.io/flextable/reference/index.html
- knit_child()
  - link (https://bookdown.org/yihui/rmarkdown-cookbook/child-document.html)

# Links (II)

- Download R
  - CRAN (https://cran.r-project.org/)
- Download RStudio
  - RStudio Desktop (https://posit.co/download/rstudio-desktop/)