Week 03 Reproducible computing

Open and reproducible science: dependable computations and statistics

In-class tasks - Solution

Step 8 - Exercise 7

Write at least two unit tests for a given function to set correct types (numerical) of columns.

- function name: `toNumericCol`.
- arguments:
  - `df`, the name of the input `data.frame`
- return value: `data.frame`

```
sheet <- readSheet(file=here::here("data","Klimadaten.xlsx"), sheetName="Neuschnee")
colnames(sheet)[1] <- "Jahr"
sheet <- removeMissingRow(sheet)
sheet <- removeMissingCol(sheet)
sheet <- adaptColNames(sheet)
sheet <- removeInvalidYearRow(sheet)
sheet <- replaceWithNA(sheet)
sheet <- toNumericCol(sheet)
```

with the following implementation:

```
toNumericCol <- function(df){
  df %>%
    dplyr::mutate(dplyr::across(dplyr::everything(), ~ as.double(.x)))
}
```

```
test_that("toNumericCol works",{
  sheet <- readSheet(file=here::here("data","Klimadaten.xlsx"), sheetName="Neuschnee")
  colnames(sheet)[1] <- "Jahr"
  sheet <- removeMissingRow(sheet)
  sheet <- removeMissingCol(sheet)
  sheet <- adaptColNames(sheet)
  sheet <- removeInvalidYearRow(sheet)
  sheet <- replaceWithNA(sheet)
  sheet_num <- toNumericCol(sheet)

  expect_is(sheet_num, "data.frame")
  expect_equal(dim(sheet)[1],dim(sheet_num)[1])
  expect_equal(dim(sheet)[2],dim(sheet_num)[2])
  expect_equal(colnames(sheet),colnames(sheet_num))
  expect_true(all(purrr::map_lgl(seq_along(colnames(sheet_num)),~is.numeric(sheet_num[[.x]]))))
})
```

```
## Test passed
```

Step 9 - Exercise 8

Write a function to convert data.frame to long format and add Altitude data

- function name: `toLongFormat`.
- arguments:
  - `df`, the name of the input `data.frame`
  - `values_to`, name of numeric column values, e.g. sheet name
  - `altitude`, altitude data.frame
- return value: `data.frame`

```
sheet <- readSheet(file=here::here("data","Klimadaten.xlsx"), sheetName="Neuschnee")
colnames(sheet)[1] <- "Jahr"
sheet <- removeMissingRow(sheet)
sheet <- removeMissingCol(sheet)
sheet <- adaptColNames(sheet)
altitude <- getAltitude(sheet)
sheet <- removeInvalidYearRow(sheet)
sheet <- replaceWithNA(sheet)
sheet <- toNumericCol(sheet)
sheet <- toLongFormat(sheet,"Neuschnee",altitude)
```

Expected outcome:

```
> head(sheet)
  Jahr       Location Neuschnee Altitude
1 1931 BaselBinningen        86      316
2 1931 BernZollikofen       192      553
3 1931       DavosWSL        NA     1560
4 1931   GenfCointrin        NA      411
5 1931   LocarnoMonti        NA      367
6 1931         Lugano        16      273
```

```
toLongFormat <- function(df, values_to, altitude){
  df %>%
    tidyr::pivot_longer(colnames(.)[colnames(df) != "Jahr"], names_to = "Location", values_to = values_
    dplyr::inner_join(altitude, by="Location")
}
```

The function should pass the following unit tests:

```
test_that("toLongFormat works",{
  sheet <- readSheet(file=here::here("data","Klimadaten.xlsx"), sheetName="Neuschnee")
  colnames(sheet)[1] <- "Jahr"
  sheet <- removeMissingRow(sheet)
  sheet <- removeMissingCol(sheet)
  sheet <- adaptColNames(sheet)
  altitude <- getAltitude(sheet)
```

```
  sheet <- removeInvalidYearRow(sheet)
  sheet <- replaceWithNA(sheet)
  sheet <- toNumericCol(sheet)
  sheet_long <- toLongFormat(sheet,"Neuschnee",altitude)


  expect_is(sheet_long, "data.frame")
  expect_equal(dim(sheet_long)[2],4)
  expect_equal(dim(sheet_long)[1],dim(sheet)[1]*(dim(sheet)[2]-1))
  expect_true(all(c("Jahr","Location","Altitude") %in% colnames(sheet_long)))
})
```

```
## Test passed
```

Step 10 - Exercise 9

Write a function to read sheet from xlsx file and convert to correct format. I.e. combine all previous functions.

- function name: `sheetToDF`.
- arguments:
  - `file`, the name of the `xlsx` file
  - `sheetName`, the name of the sheet
- return value: `data.frame`

```
sheet <- sheetToDF(file=here::here("data","Klimadaten.xlsx"), sheetName="Neuschnee")
```

```
sheetToDF <- function(file, sheetName){
  sheet <- readSheet(file=file, sheetName=sheetName)
  colnames(sheet)[1] <- "Jahr"
  sheet <- sheet %>%
    removeMissingRow() %>%
    removeMissingCol() %>%
    adaptColNames()
  altitude <- getAltitude(sheet)
  sheet %>%
    removeInvalidYearRow() %>%
    replaceWithNA() %>%
    toNumericCol() %>%
    toLongFormat(sheetName,altitude)
}
```

The function should pass the following unit tests:

```
test_that("sheetToDF works",{
  sheet <- readSheet(file=here::here("data","Klimadaten.xlsx"), sheetName="Neuschnee")
  colnames(sheet)[1] <- "Jahr"
  sheet <- removeMissingRow(sheet)
  sheet <- removeMissingCol(sheet)
  sheet <- adaptColNames(sheet)
```

```
  altitude <- getAltitude(sheet)
  sheet <- removeInvalidYearRow(sheet)
  sheet <- replaceWithNA(sheet)
  sheet <- toNumericCol(sheet)
  sheet_long <- sheetToDF(file=here::here("data","Klimadaten.xlsx"), sheetName="Neuschnee")

  expect_is(sheet_long, "data.frame")
  expect_equal(dim(sheet_long)[2],4)
  expect_equal(dim(sheet_long)[1],dim(sheet)[1]*(dim(sheet)[2]-1))
  expect_true(all(c("Jahr","Location","Altitude") %in% colnames(sheet_long)))
})
```

```
## Test passed
```

Step 11 - Exercise 10

Write at least two unit tests for a given function to read sheets from xlsx file, convert to correct format and combine them.

- function name: `fileToDF`.
- arguments:
  - `file`, the name of the `xlsx` file
  - `sheetName`, the names of the sheet
- return value: `data.frame`

```
sheets <- fileToDF(file=here::here("data","Klimadaten.xlsx"), sheetName=c("Sonnenscheindauer","Neuschnee
```

Expected outcome:

```
> head(sheets)
  Jahr       Location Sonnenscheindauer Altitude Neuschnee
1 1931 BaselBinningen          1594.317      316        86
2 1931 BernZollikofen          1742.500      553       192
3 1931          Davos          1767.600     1594        NA
4 1931   GenfCointrin          1790.733      411        NA
5 1931   LocarnoMonti                NA      367        NA
6 1931         Lugano          2293.900      273        16
```

with the following implementation

```
fileToDF <- function(file, sheetName){
  purrr::map(sheetName,function(sn){
    sheetToDF(file=file, sheetName=sn)
  }) %>%
    purrr::reduce(dplyr::full_join, by=c("Jahr","Location","Altitude"))
}
```

```
test_that("sheetToDF works",{
  sheet_long <- sheetToDF(file=here::here("data","Klimadaten.xlsx"), sheetName="Neuschnee")
  sheets <- fileToDF(file=here::here("data","Klimadaten.xlsx"), sheetName="Neuschnee")

  expect_is(sheets, "data.frame")
  expect_equal(sheet_long,sheets)

  sheet_long <- sheetToDF(file=here::here("data","Klimadaten.xlsx"), sheetName="Neuschnee")
  sheets <- fileToDF(file=here::here("data","Klimadaten.xlsx"), sheetName=c("Sonnenscheindauer","Neuschn

  expect_is(sheets, "data.frame")

  expect_gte(dim(sheets)[1],dim(sheet_long)[1])
  expect_equal(dim(sheets)[2],dim(sheet_long)[2]+1)
  expect_true(all(c("Jahr","Location","Altitude") %in% colnames(sheets)))
})
```

```
## Test passed
```