Week 03 Reproducible computing

Open and reproducible science: dependable computations and statistics

Homework - Solution

Preparation

First of all you will need to download the data and save it at a location of your choice. You can use

```
download.file("https://www.bfs.admin.ch/bfsstatic/dam/assets/15964168/master",
              here::here("data","Klimadaten.xlsx"), mode="wb")
```

which will save the `xlsx` file in the directory `data` with the name `Klimadaten.xlsx`. To read a single sheet you can do

```
readxl::read_xlsx(path=here::here("data","Klimadaten.xlsx"),sheet="Sonnenscheindauer",skip=5)
```

```
## New names:
## * `` -> `...1`
```

```
## # A tibble: 102 x 14
##    ...1  `Basel-Binningen 1)` `Bern-Zollikofen 1,2)` Davos      `Genf-Cointrin~`
##    <chr> <chr>                <chr>                  <chr>      <chr>
##  1 <NA>  316 m ü. M.          553 m ü. M.            1594 m ü.~ 411 m ü. M.
##  2 <NA>  <NA>                 <NA>                   <NA>       <NA>
##  3 <NA>  <NA>                 <NA>                   <NA>       <NA>
##  4 1931  1594.3166666666666   1742.5                 1767.6     1790.7333333333~
##  5 1932  1603.6166666666666   1630.1                 1770.6     1534.4666666666~
##  6 1933  1677.6166666666666   1727.3                 1647.9     1762.8
##  7 1934  1730.6166666666666   1924.7                 1826.2     1965.75
##  8 1935  1612.3166666666666   1746.9                 1592       1770.2
##  9 1936  1376.1333333333334   1494.2833333333333     1477.2     1618.4666666666~
## 10 1937  1410.8166666666666   1626.4                 1418.5     1714.8333333333~
## # ... with 92 more rows, and 9 more variables: `Locarno-Monti` <chr>,
## #   Lugano <chr>, Luzern <chr>, Neuenburg <chr>, Säntis <chr>, Samedan <chr>,
## #   Sitten <chr>, `St. Gallen` <chr>, `Zürich-Fluntern 1)` <chr>
```

This will read all columns of the sheet `Sonnenscheindauer` starting from row number 6 (here it is assumed that the header will not change and is always the same).

```
suppressPackageStartupMessages({
  library(tidyverse)
  library(testthat)
})
```

```
## Warning: package 'tidyr' was built under R version 4.0.5
```

```
## Warning: package 'readr' was built under R version 4.0.5
```

Step 1

The first step is given as an example:

We need a function that reads a single sheet from a `xlsx` file.

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

Usage example:

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

Implementation:

```
readSheet <- function(file, sheetName){
  readxl::read_xlsx(path=file,sheet=sheetName,skip=5)
}
```

The function needs to pass the follwoing unit tests:

```
test_that("readSheet works",{
  # run function
  sheet <- readSheet(file=here::here("data","Klimadaten.xlsx"), sheetName="Neuschnee")
  # check if return value is a data.frame
  expect_is(sheet, "data.frame")
  # check if column names are as expected
  expect_true(all(stringr::str_extract(colnames(sheet),"[:alpha:]+") %in%
                    c(NA,"Basel", "Bern", "Davos", "Genf", "Locarno", "Lugano",
                      "Luzern", "Neuenburg", "Säntis", "Samedan", "Sitten",
                      "St", "Zürich")))
})
```

```
## Test passed
```

Step 2 - Exercise 1

Write a function that removes columns from a data.frame that only contains missing values (`NA`'s).

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

Usage example:

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

```r
removeMissingCol <- function(df){
  df[,purrr::map_lgl(seq_along(colnames(df)),~!all(is.na(df[,.x])))]
}
```

The function should pass the following unit tests:

```r
test_that("removeMissingCol works",{
  # read data
  sheet <- readSheet(file=here::here("data","Klimadaten.xlsx"), sheetName="Neuschnee")
  # run function
  sheet_no_na <- removeMissingCol(sheet)
  # test return value type
  expect_is(sheet_no_na, "data.frame")
  # test that number of rows stays the same
  expect_equal(dim(sheet)[1],dim(sheet_no_na)[1])
  # test that number of columns of original data.frame is greater than or equal to afterwards
  expect_gte(dim(sheet)[2],dim(sheet_no_na)[2])
  # test that column names stay the same
  expect_true(all(colnames(sheet_no_na) %in% colnames(sheet)))
})
```

```
## Test passed
```

Step 3 - Exercise 2

Write a function that removes rows with only missing values (`NA`'s)

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

Usage example:

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

```r
removeMissingRow <- function(df){
  df %>%
    rowwise() %>%
    dplyr::mutate(allna = all(is.na(dplyr::c_across(cols = dplyr::everything())))) %>%
    dplyr::filter(!allna) %>%
    dplyr::select(-allna)
}
```

The function should pass the following unit tests:

```
test_that("removeMissingRow works",{
  # read data and prepare
  sheet <- readSheet(file=here::here("data","Klimadaten.xlsx"), sheetName="Neuschnee")
  sheet_no_na <- removeMissingRow(sheet)
  # test return value type
  expect_is(sheet_no_na, "data.frame")
  # test that number of columns stays the same
  expect_equal(dim(sheet)[2],dim(sheet_no_na)[2])
  # test that number of rows of original data.frame is greater than or equal to afterwards
  expect_gte(dim(sheet)[1],dim(sheet_no_na)[1])
  # test that column names stay the same
  expect_equal(colnames(sheet_no_na), colnames(sheet))
})
```

```
## Test passed
```

Step 4 - Exercise 3

Write at least two unit tests for a given function that removes wrong `.` and digits in column names.

- function name: `adaptColNames`.
- 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)
```

Expected column names of sheet:

```
> colnames(sheet)
 [1] "Jahr"           "BaselBinningen" "BernZollikofen"
 [4] "DavosWSL"       "GenfCointrin"   "LocarnoMonti"
 [7] "Lugano"         "Luzern"         "Neuenburg"
[10] "Säntis"         "Samedan"        "Sitten"
[13] "St.Gallen"      "ZürichFluntern"
```

with the following implementation

```
adaptColNames <- function(df){
  colnames(df) <- stringr::str_replace_all(colnames(df),"\\.(?!\\.)|[:digit:]+| |\\)|-","")
  df
}
```

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

  expect_is(sheet_no, "data.frame")
  expect_equal(dim(sheet)[1],dim(sheet_no)[1])
  expect_equal(dim(sheet)[2],dim(sheet_no)[2])
  expect_false(all(stringr::str_detect(colnames(sheet_no),"[.]+") &
                     !stringr::str_detect(colnames(sheet_no),"St.Gallen")))
})
```

```
## Test passed
```

Step 5 - Exercise 4

Write at least two unit tests for a given function to extract altitude data.

- function name: `getAltitude`.
- arguments:
  - `df`, the name of the input `data.frame`
  - `rowInd`, default=1L, integer, indices of row containing altitude data
- return value: `data.frame` with two columns, `Altitude`, type = integer, and `Location`, type=character

```
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)
```

Expected outcome:

```
> altitude
   Altitude       Location
1       316 BaselBinningen
2       553 BernZollikofen
3      1560       DavosWSL
4       411   GenfCointrin
5       367   LocarnoMonti
6       273         Lugano
7       454         Luzern
8       485      Neuenburg
9      2501         Säntis
10     1709        Samedan
11      482         Sitten
12      776      St.Gallen
13      556 ZürichFluntern
```

with the following implementation

```
getAltitude <- function(df, rowInd=1L){
  df[rowInd,]  %>%
    stringr::str_extract("[:digit:]+") %>% # extract numbers
    as.integer() %>%
    as.data.frame() %>%
    setNames("Altitude") %>% # set column name
    dplyr::mutate(Location=colnames(df)) %>% # add locations
    dplyr::filter(!is.na(Altitude)) # remove NA's
}
```

```
test_that("getAltitude 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)
  expect_is(altitude, "data.frame")
  expect_equal(dim(altitude)[2],2)
  expect_equal(dim(altitude)[1],dim(sheet)[2]-1)
  expect_setequal(colnames(altitude), c("Altitude","Location"))
  expect_type(altitude$Altitude,"integer")
  expect_type(altitude$Location,"character")
  expect_true(length(altitude$Location) == length(unique(altitude$Location)))
})
```

```
## Test passed
```

Step 6 - Exercise 5

Write a function that removes rows where the value of column `Jahr` is not a valid year (e.g. four digits number)

- function name: `removeInvalidYearRow`.
- arguments:
  - `df`, the name of the input `data.frame`
  - `yearcol`, default=`Jahr`, column name containing year
- return value: `data.frame`

Usage example:

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

```
removeInvalidYearRow <- function(df, yearcol="Jahr"){
  df[stringr::str_detect(df[[yearcol]],"^[:digit:]{4}$") & !is.na(df[[yearcol]]),]
}
```

The function should pass the following unit tests:

```
test_that("removeInvalidYearRow works",{
  # read and prepare
  sheet <- readSheet(file=here::here("data","Klimadaten.xlsx"), sheetName="Neuschnee")
  colnames(sheet)[1] <- "Jahr"
  sheet_no_na <- removeInvalidYearRow(sheet)
  # check return type
  expect_is(sheet_no_na, "data.frame")
  # same number of columns
  expect_equal(dim(sheet)[2],dim(sheet_no_na)[2])
  # greater than or equal number of rows
  expect_gte(dim(sheet)[1],dim(sheet_no_na)[1])
  # same column names
  expect_equal(colnames(sheet_no_na), colnames(sheet))
})
```

```
## Test passed
```

Step 7 - Exercise 6

Write at least two unit tests for a given function to replace all occurrences of `...` with `NA`.

- function name: `replaceWithNA`.
- arguments:
  - `df`, the name of the input `data.frame`
  - `NAPattern`, default="...", pattern that should be replaced with `NA`
- 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)
```

With the following implementation:

```
replaceWithNA <- function(df, NAPattern="..."){
  escaped_pattern <- stringr::str_replace_all(NAPattern, "\\.","\\\\.")
  dplyr::mutate(df, dplyr::across(dplyr::everything(),
                                  ~ stringr::str_replace(.x, escaped_pattern,"")))
}
```

```
test_that("replaceWithNA 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_na <- replaceWithNA(sheet)
  expect_is(sheet, "data.frame")
  expect_equal(dim(sheet)[1],dim(sheet_na)[1])
  expect_equal(dim(sheet)[2],dim(sheet_na)[2])
  expect_equal(colnames(sheet),colnames(sheet_na))
})
```

```
## Test passed
```