Course of Study Bachelor Computer Science	Exercises Statistics WS 2023/24

Sheet II - Solutions

1. Tidy Data
   Consider the following datasets

```
student1  <- tibble(
  student = c("Adam","Bernd","Christian","Doris"),
  algebra = c(NA, 5, 3, 4),
  analysis = c(2, NA, 1,3),
  diskrete.math = c(3,NA,2,4),
)

student2 <- tibble(
   name = rep(c("Adam", "Bernd", "Christian", "Doris"), each = 2),
   type = rep(c("height", "weight"), 4),
   measure = c(1.83, 81, 1.75, 71, 1.69, 55, 1.57, 62))

student3 <- tibble(
    name = c("Adam", "Bernd", "Christian", "Doris"),
    ratio = c("81/1.83", "71/1.75", "55/1.69", "62/1.57"))
```

   (a) Describe for every dataset what the dataset contains? What are the variables and what are the observations?

   (b) Why are these datasets are not tidy?

   (c) Make a tidy version of all datasets.

```
################################################
# Sheet II: tidy and messy data
#
file: tidy_messy.R
################################################

# load libraries
library(tidyverse)

# Create the datasets
student1  <- tibble(
  student = c("Adam","Bernd","Christian","Doris"),
  algebra = c(NA, 5, 3, 4),
  analysis = c(2, NA, 1,3),
  diskrete.math = c(3,NA,2,4)
)
student1

student2 <- tibble(
  name = rep(c("Adam", "Bernd", "Christian", "Doris"), each = 2),
  type = rep(c("height", "weight"), 4),
```

```
  measure = c(1.83, 81, 1.75, 71, 1.69, 55, 1.57, 62))
student2

student3 <- tibble(
  name = c("Adam", "Bernd", "Christian", "Doris"),
  ratio = c("81/1.83", "71/1.75", "55/1.69", "62/1.57"))
student3

# student1: contains 36 values representing three variables and 12 observations.
# The variables are: name, exam, grade
# Every combination of name and exam is a single measured observation. The 12
# single measured observations are the values of grade for every name and exam.

# student2 and student3: contains 12 values representing three variables and
# 3 measurements for every student. The variables are: name, height, weight
# The 4 single measured observations are the values of height and weight for every name.

# Why are these datasets are not tidy?

# student1: column headers are values of the variable exam, not variable names
# student2: the variables weight and height are stored in both rows and columns
# student3: the values of the variables height and weight are stored in one column

# tidy versions
student1 %>%
  gather('algebra','analysis','diskrete.math',
         key = "exam", value = "grade")

student2 %>%
  spread(key = type, value = measure)

student3 %>%
  separate(col = ratio, into = c("weight","height"), sep = "/") %>%
  mutate(ratio = as.double(height)/as.double(weight)) %>%
  select(ratio) %>%
  round(2)
```

2. Using the %>%-operator.

   - Calculate the value of $\sin(\log(\sqrt{5+3}))$ directly and using the %>%-operator.
   - Define a vector v with values 0.5,1,1.5,...,5 and calculate the by 2 digits rounded sum of the logarithms of the squared values of v with nested operations and using the %>%-operator.

```
################################################
# Sheet II: pipe operator
#
# file: pipe_op.R
################################################

# load libraries
library(tidyverse)

# 2 %>%-Operator
# a)
sin(log((5+3)**0.5))
# or
(5+3) %>% sqrt() %>% log() %>% sin()
# or
5 %>% sum(3) %>% sqrt() %>% log() %>% sin()
# b)
v <- seq(from = 0.5, to = 5, by = 0.5)
v

round(sum(log(v**2)),2)
# %>%-operator
v %>%
  '^'(2) %>%
  log() %>%
  sum() %>%
  round(2)
```

3. Create a tibble df with the data of 10 students, i.e. with 10 rows and the columns id (values 1,2,..., 10), sex (values are "f" and "m", age (integer values between 20 and 35) and score1 (integer values between 0 and 25). You can choose arbitrary values in the columns. If you do not like coding the values by hand you can use:

```
df <- tibble(id = 1:10,
             sex = sample(x =c("f","m"), size = 10,
                          replace = TRUE),
             age = round(runif(10,20,35)),
             score1 = round(runif(10,0,25))
            )
```

   - Select the date of all male students.
   - Add the data of a new student with id = 11, sex = "m", age = 25 and score1 = 4.
   - Add two columns score2 and score3 with random integer numbers between 0 and 25.
   - Add a column containing sum of all scores.
   - Add a column which denote the grades according to the scheme

$$\text{grad} = \begin{cases} 5 & \text{if} \quad \text{score sum} \leq 37 \\ 4 & \text{if} \quad 37 < \text{score sum} \leq 45 \\ 3 & \text{if} \quad 45 < \text{score sum} \leq 55 \\ 2 & \text{if} \quad 55 < \text{score sum} \leq 65 \\ 1 & \text{if} \quad \text{score sum} \geq 65 \end{cases}.$$

   - Find the values of the variables id, sex and grade sorted by the values of sex of all students who have passed.
   - Calculate the mean, minimum, maximum and median of the variable sum of scores grouped by the variable sex.

```
################################################
# Sheet II: creating and manipulating variables
#
# file: mutate_appl.R
################################################

# load libraries
library(tidyverse)

df <- tibble(
  id = 1:10,
  sex = sample(x =c("f","m"), size = 10,
               replace = TRUE),
  age = round(runif(10,20,35)),
  score1 = round(runif(10,0,25))
)
df

# a)
```

```
df %>% filter(sex == "m")

# b)
df <- df %>%
  add_row(id = 11, sex = "m", age = 25, score1 = 4)
df

# c)
df <-
  df %>%
  mutate(score2 = round(runif(11,0,25)),
         score3 = round(runif(11,0,25)),
         scoresum = score1+score2+score3,
         grade = case_when(
           scoresum <= 37 ~ 5,
           scoresum > 37 & scoresum <= 45 ~ 4,
           scoresum > 45 & scoresum <= 55 ~ 3,
           scoresum > 55 & scoresum <= 65 ~ 2,
           scoresum > 65 ~ 1))
df

# d)
df %>%
  select(id,sex,grade) %>%
  filter(grade < 5) %>%
  arrange(sex)

# e)
df %>%
  group_by(sex) %>%
  summarise(mean_scores = mean(scoresum),
            min_scores = min(scoresum),
            max_scores = max(scoresum),
            med_scores = median(scoresum))
```

4. The R statements

```
no <- 30
exercise.results <- tibble(
  stud.id = 1:no,
  group = sample(x=c("A","B","C"), size=no, replace = TRUE),
  ex1 = sample(x=1:10, size=no, replace = TRUE),
  ex2= sample(x=1:10, size=no, replace = TRUE),
  ex3 = sample(x=1:10, size=no, replace = TRUE),
  ex4 = sample(x=1:10, size=no, replace = TRUE),
  ex5 = sample(x=1:10, size=no, replace = TRUE)
)
```

creates a tibble containing the scores of 30 students in 5 exercises.

(a) Apply n() and count() to get the number of students in the different groups. What are the difference between n() and count()?

(b) Add the variables sum.scores and mean.scores containing the sum and the of the scores in the exercises for every student by applying the the functions sum() and mean(). What is the result if rowwise() is appplied before the mutate()?

```
################################################################
# Sheet 2: n(), count() and rowwise()
#
# file n_count_rowwise.R
################################################################

library(tidyverse)

no <- 30
exercise.results <- tibble(
  stud.id = 1:no,
  group = sample(x=c("A","B","C"), size=no, replace = TRUE),
  ex1 = sample(x=1:10, size=no, replace = TRUE),
  ex2= sample(x=1:10, size=no, replace = TRUE),
  ex3 = sample(x=1:10, size=no, replace = TRUE),
  ex4 = sample(x=1:10, size=no, replace = TRUE),
  ex5 = sample(x=1:10, size=no, replace = TRUE)
)
exercise.results

# a) Apply n() and count() to get the number of students in the different
#    groups. What are the difference between n() and count()?
exercise.results %>%
  group_by(group) %>%
  summarise(no.students = n())
# count() = first group_by() and and than n().
exercise.results %>%
  count(group)

# b) Add the variables sum.scores and mean.scores containing the sum
#    and the of the scores in the exercises for every student by applying the
#    the functions sum() and mean(). What is the result if rowwise() is applied
#    before the mutate()?
exercise.results %>%
  # Without rowwise() the sum resp. the mean of all columns ex1, ..., ex5 are
  # calculated. Applying rowwise() theses results are rowwise evaluated.
  rowwise() %>%
  mutate(
    sum.scores = sum(c(ex1,ex2,ex3,ex4,ex5)),
    mean.scores = mean(c(ex1,ex2,ex3,ex4,ex5))
  )
```

5. Some data manipulations with the data set flights.

   - Load the libraries tidyverse and nycflight13 and inspect the variable of flights.
   - Find all flights with more than 2 hours arrival delay.
   - Find all flights with more tahn 2 hours arrival delay and no departure delay.
   - Find all flights from United, American and Delta with no arrival delay.
   - Find all flights from United, American and Delta in the month May with more than 5 hours arrival delay sorted by carrier and flight number.
   - Add a column speed which denotes the average speed of the flight and determine the carrier, flight of the top 10 values of speed.
   - Find a list of carriers with a column ratio which denotes the number of flights with arr_delay less than 10 minutes to the total number of flights. The list should be sorted by ratio.

- Determine a table that shows, for each airline (carrier), the flight connection given by the airports of dest und origin that occurred most frequently in 2013. The table should contain only the columns names of airline, destination, origin and frequency and be sorted by frequency in descending order. You can find the names of the carrier from the dataset airlines and the names of the airports from the dataset airports.

```
################################################
# Sheet II: Analysis of nycflights
#
# file: nycflights_evaluations.R
################################################

# load libraries
library(tidyverse)
library(nycflights13)

# a)
?flights()

# b) Find all fights with more than 2 hours arrival delay.
res.b <-
  flights %>% filter(arr_delay > 120)
res.b

# c) Find all fights with more than 2 hours arrival delay and no
#    departure delay.
res.c <-
  flights %>% filter(arr_delay > 120 & dep_delay <= 0)
res.c

# d) Find all fights from United, American and Delta with no arrival
delay.
res.d <-
  flights %>%
  filter(carrier %in% c("AA","DL","UA") & arr_delay <= 0)
res.d

# e) Find all fights from United, American and Delta in the month
#    May with more than 5 hours arrival delay sorted by carrier and
#    flight number.
res.e <-
  flights %>%
  filter(carrier %in% c("UA","AA","DL") & month == 5 & arr_delay > 300) %>%
  select(carrier, flight) %>%
  arrange(carrier,flight) %>%
  # remove multiple entries
  unique()
res.e

# f) Exchange the values of departure time and arrvial time in minute
#    after midnight.
# Format HHMM or HMM, i.e. the last 2 numbers denote
# minute and the first or the first two are numbers denote
# the hour
# x %% y          modulus (x mod y) 5%%2 is 1
# x %/% y  integer division 5%/%2 is 2
flights.new <-
  flights %>%
  mutate(dep_time = (dep_time %/% 100)*60 + dep_time %% 100,
         arr_time = (arr_time %/% 100)*60 + arr_time %% 100)
flights.new

# g) Add a column speed which denotes the average speed of the flight
#    and determine the carrier, flight of the top 10 values of speed.
res.g <- flights %>%
  mutate(speed = distance / air_time * 60) %>%
  select(carrier,flight,speed) %>%
  arrange(desc(speed)) %>%
  top_n(10,speed)
res.g

# h) Find a list of carriers with a column ratio which denotes the number
#    of flights with arr delay less than 10 minutes to the total number of
#    flights. The list should be sorted by ratio.
res.h <-
```

```
  flights %>%
  # remove the NA's
  filter(!is.na(arr_delay)) %>%
  # boolean variable indicating a delay
  mutate(bool_del = if_else(arr_delay < 10,1,0)) %>%
  group_by(carrier) %>%
  # new columns: nof = number of flights,
  # ndel = number of delays, del_ratio = ratio
  # values calculate per carrier
  mutate(nof = n(),
         ndel = sum(bool_del),
         del_ratio = ndel / nof) %>%
  select(carrier, nof, ndel, del_ratio) %>%
  # remove multiple entries
  unique() %>%
  arrange(desc(del_ratio))
res.h

# alternative solution with count()
flights %>%
  # remove the NA's
  filter(!is.na(arr_delay)) %>%
  count(arr_delay < 10, carrier) %>%
  group_by(carrier) %>%
  # new columns: nof = number of flights,
  # ndel = number of delays, del_ratio = ratio
  # values calculate per carrier
  mutate(
    nof = sum(n),
    del_ratio = n / sum(n)) %>%
  filter(`arr_delay < 10` == TRUE) %>%
  arrange(desc(del_ratio))

# i) Find a list which denotes for every month the carrier with highest ratio.
#    The list should have the columns month, carrier, number of flights of the
#    carrier in that month and ratio.
res.i <-
  flights %>%
  # remove NA's
  filter(!is.na(arr_delay)) %>%
  # boolean variable indicating a delay
  mutate(bool_del = if_else(arr_delay < 10,1,0)) %>%
  # Calculation grouped by carrier and month
  group_by(month,carrier) %>%
  # new columns: nof = number of flights,
  # ndel = number of delays, del_ratio = ratio
  # values calculate per carrier
  mutate(nof = n(),
         ndel = sum(bool_del),
         del_ratio = ndel / nof) %>%
  # keep only 4 columns
  select(month, carrier, nof, ndel, del_ratio) %>%
  # calculation per month
  group_by(month) %>%
  # only highest ratio
  filter(del_ratio == max(del_ratio)) %>%
  # remove multiple entries
  unique() %>%
  arrange(month)
res.i

# j) Find a table with the number of cancelled flights
#    (dep delay = NA), the number of flights with no dep delay
#    (-5 <= dep delay <= 5 minutes) and the means of dep delay,
#    arr delay per month and day.
# 3 tables are generated with values per month and day
# which are joined by these variables
res.j <-
  full_join(
    flights %>%
      filter(is.na(dep_delay)) %>%
      group_by(month, day) %>%
      # number of cancelled flights
      summarise(nof_canc = n())
    ,
    flights %>%
      group_by(month,day) %>%
      # number of no departure delays
      filter(dep_delay <= 5 & dep_delay >= -5) %>%
      summarise(nof_no_delay = n())
    ,
    by = c("month","day")
    ) %>%
```

```
  full_join(
    flights %>%
      group_by(month,day) %>%
      # means
      summarise(mean_dep_del = mean(dep_delay, na.rm = TRUE),
                mean_arr_del = mean(arr_delay, na.rm = TRUE))
    ,
    by = c("month","day")
    )
res.j


# k) Table that shows, for each carrier, the flight connection (dest, origin) that
# occurred most frequently in 2013. The table should contain only the columns
# names of airline, destination, origin and frequency and be sorted
# by frequency in descending order.
flights %>%
  # count the number of connections
  count(carrier,origin,dest) %>%
  # filter the connections with maximal value for every carrier
  group_by(carrier) %>%
  filter(n == max(n)) %>%
  # cancel grouping
  ungroup() %>%
  # sort by n in descending order
  arrange(desc(n)) %>%
  # get the names of the carriers
  left_join(y=airlines, by="carrier") %>%
  # get the names of the airports (origin)
  left_join(y=airports %>% select(faa,name), by=join_by("origin"=="faa"),
            # name the column of the name by name.origin
            suffix = c("",".origin")) %>%
  # get the names of the airports (dest)
  left_join(y=airports %>% select(faa,name), by = join_by("dest"=="faa"),
            # name the column of the name by name.dest
            suffix = c("",".dest")) %>%
  # rename columns
  rename(
    Airline = name,
    Airport.Origin = name.origin,
    Airport.Destination = name.dest
  ) %>%
  # cancel columns not needed
  select(-carrier, -dest, -origin, n)
```

6. Applications of gather(), spread() and separate()

   (a) Consider the dataset who of the package tidyr.
   country, iso2, iso3, and year are already variables, so they can be left as is. But the columns from new_sp_m014 to newrel_f65 encode four variables in their names:

   - The new_/new prefix indicates these are counts of new cases. This dataset only contains new cases, so we'll ignore it here because it's constant.
   - sp/rel/ep describe how the case was diagnosed.
   - m/f gives the gender.
   - 014/1524/2535/3544/4554/65 supplies the age range.

   Break these variables up by specifying multiple column names and make the dataset tidy.

   (b) Apply the following R statements

   ```
   production <-
   ```

```
  expand_grid(
    product = c("A", "B"),
    country = c("AI", "EI"),
    year = 2000:2014
  ) %>%
  filter((product == "A" & country == "AI") | product == "B") %>%
  mutate(production = rnorm(nrow(.)))
```

The data set production contains the combination of product, country, and year.
Widen the data so we have one column for each combination of product and country.

```
#################################################################
# Sheet 2: applications of gather(), spread() and separate()
#
# file exercise_gather_spread_separate.R
#################################################################

library(tidyverse)
library(stringr)

# a) Consider the dataset who of the package tidyr.
# country, iso2, iso3, and year are already variables, so they can be left as is.
# But the columns from new_sp_m014 to newrel_f65 encode four variables in their names:
# * The new_/new prefix indicates these are counts of new cases. This dataset only
#     contains new cases, so we'll ignore it here because it's constant.
# * sp/rel/ep describe how the case was diagnosed.
# * m/f gives the gender.
# * 014/1524/2535/3544/4554/65 supplies the age range.
# Break these variables up by specifying multiple column names and make the dataset tidy.
who %>%
  # gather the values of the columuns new_sp_m014:newrel_f65 to the columns key and values
  gather(key = "key", value = "cases", new_sp_m014:newrel_f65, na.rm = TRUE) %>%
  # exchange the newrel by new_rel -> syntax of new: new_type_sexage
  mutate(key = str_replace(key,"newrel","new_rel")) %>%
  # separate the column key into 3 columns
  separate(key,c("new", "type", "sexage"), sep = "_") %>%
  # separate the column sexage into 2 columns
  separate(sexage, c("sex", "age"), sep = 1) %>%
  # remowe the column new
  select(-new, -iso2, -iso3) -> who.tidy

# b) The anscombe dataset contains four pairs of variables (x1 and y1, x2 and y2, etc.)
# that underlie Anscombe's quartet, a collection of four datasets that have the
# same summary statistics (mean, sd, correlation etc), but have quite different data.
# Produce a dataset with columns set, x and y.
anscombe[1:4] %>%
  # gather the the first 4 columns to the columns set and x
  gather(key = "set", value = "x") %>%
  # remove the char x from all values of set
  mutate(set = str_remove(set, pattern ="x")) -> anscombe.x
# Do the same for the last 4 columns
anscombe[5:8] %>%
  gather(key = "set", value = "y") %>%
  mutate(set = str_remove(set, pattern ="y")) -> anscombe.y
# create a tibble containing set, x and y
tibble(
  set = anscombe.x$set,
  x = anscombe.x$x,
  y = anscombe.y$y
  )

# c) Apply the following R statements
production <-
  expand_grid(
    product = c("A", "B"),
    country = c("AI", "EI"),
    year = 2000:2014
  ) %>%
  filter((product == "A" & country == "AI") | product == "B") %>%
  mutate(production = rnorm(nrow(.)))
```

```
production
# The data set production contains the combination of product, country, and year.
# Widen the data so we have one column for each combination of product and country.
production %>%
  mutate(pc = str_c(product, country, sep= "_")) %>%
  select(-product, -country) %>%
  spread(key = pc, value = production)

# d) The data set warpbreaks gives the number of warp breaks per loom, where a
# loom corresponds to a fixed length of yarn for every combination of wool (A and B)
# and tension (L, M, H).
# Produce a data set with the columns tension, A, B with the means of the breaks
warpbreaks %>%
  group_by(wool, tension) %>%
  summarise(Mean = mean(breaks)) %>%
  spread(key = wool, value = Mean)
```