Sheet II: Exercises Datamanipulations with R

Dr. Falkenberg

WS 2020/21

Tibbles, %>%-Operator and Data Manipulations with tidyverse

Task 1 Tidy Data

- 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
```

```
## # A tibble: 4 x 4
##   student   algebra analysis diskrete.math
##   <chr>       <dbl>    <dbl>         <dbl>
## 1 Adam           NA        2             3
## 2 Bernd           5       NA            NA
## 3 Christian       3        1             2
## 4 Doris           4        3             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))
student2
```

```
## # A tibble: 8 x 3
##   name      type   measure
##   <chr>     <chr>    <dbl>
## 1 Adam      height    1.83
## 2 Adam      weight   81   
## 3 Bernd     height    1.75
## 4 Bernd     weight   71   
## 5 Christian height    1.69
## 6 Christian weight   55   
## 7 Doris     height    1.57
## 8 Doris     weight   62   
```

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

```
## # A tibble: 4 x 2
##   name      ratio
##   <chr>     <chr>
## 1 Adam      81/1.83
## 2 Bernd     71/1.75
## 3 Christian 55/1.69
## 4 Doris     62/1.57
```

- Description of the datasets
  - 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.
  - student2 and student3: contains 12 values representing three variables and 3 observations. The variables are: name, height, weight. The 3 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")
```

```
## # A tibble: 12 x 3
##    student   exam          grade
##    <chr>     <chr>         <dbl>
##  1 Adam      algebra          NA
##  2 Bernd     algebra           5
##  3 Christian algebra           3
##  4 Doris     algebra           4
##  5 Adam      analysis          2
##  6 Bernd     analysis         NA
##  7 Christian analysis          1
##  8 Doris     analysis          3
##  9 Adam      diskrete.math     3
## 10 Bernd     diskrete.math    NA
## 11 Christian diskrete.math     2
## 12 Doris     diskrete.math     4
```

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

```
## # A tibble: 4 x 3
##   name      height weight
##   <chr>      <dbl>  <dbl>
## 1 Adam        1.83     81
## 2 Bernd       1.75     71
## 3 Christian   1.69     55
## 4 Doris       1.57     62
```

```
student3 %>%
  separate(col = ratio, into = c("weight","height"), sep = "/")
```

```
## # A tibble: 4 x 3
##   name      weight height
##   <chr>     <chr>  <chr>
## 1 Adam      81     1.83
## 2 Bernd     71     1.75
## 3 Christian 55     1.69
## 4 Doris     62     1.57
```

Task 2 %>%-Operator

- Calculate the value of sin(log(5+3)) directly and using the %>%-operator.

```
sin(log((5+3)**0.5))
```

```
## [1] 0.8622628
```

```
# or
(5+3) %>% sqrt() %>% log() %>% sin()
```

```
## [1] 0.8622628
```

- 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.

```
v <- seq(from = 0.5, to = 5, by = 0.5)
v
```

```
##  [1] 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

```
# nested operations
v1 <- v**0.5
v2 <- log(v1)
s <- sum(v2)
sr <- round(s,2)
sr
```

```
## [1] 4.09
```

```
# or
round(sum(log(v**0.5)),2)
```

```
## [1] 4.09
```

```
# %>%-operator
v %>% sqrt() %>% log() %>% sum() %>% round(2)
```

```
## [1] 4.09
```

Task 2

- 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))
)
df
```

```
## # A tibble: 10 x 4
##       id sex     age score1
##    <int> <chr> <dbl>  <dbl>
##  1     1 f        29     17
##  2     2 f        21      9
##  3     3 m        28      7
##  4     4 f        24     16
##  5     5 f        29     18
##  6     6 f        24      5
##  7     7 f        26     14
##  8     8 m        30     21
##  9     9 m        33      5
## 10    10 f        33      4
```

- Select the date of all male students.

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

```
## # A tibble: 3 x 4
##      id sex     age score1
##   <int> <chr> <dbl>  <dbl>
## 1     3 m        28      7
## 2     8 m        30     21
## 3     9 m        33      5
```

- Add the data of a new student with id = 11, sex = ``m”, age = 25 and score1 = 4.

```
df <- add_row(df, id = 11, sex = "m", age = 25, score1 = 4)
df
```

```
## # A tibble: 11 x 4
##       id sex     age score1
##    <dbl> <chr> <dbl>  <dbl>
##  1     1 f        29     17
##  2     2 f        21      9
##  3     3 m        28      7
##  4     4 f        24     16
##  5     5 f        29     18
##  6     6 f        24      5
##  7     7 f        26     14
##  8     8 m        30     21
##  9     9 m        33      5
## 10    10 f        33      4
## 11    11 m        25      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 described scheme

```
df <-
  df %>%
  mutate(score2 = round(runif(11,0,25))) %>%
  mutate(score3 = round(runif(11,0,25))) %>%
  mutate(scoresum = score1+score2+score3) %>%
  mutate(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
```

```
## # A tibble: 11 x 8
##       id sex     age score1 score2 score3 scoresum grade
##    <dbl> <chr> <dbl>  <dbl>  <dbl>  <dbl>    <dbl> <dbl>
##  1     1 f        29     17      2      0       19     5
##  2     2 f        21      9      1     18       28     5
##  3     3 m        28      7      9     23       39     4
##  4     4 f        24     16     14      2       32     5
##  5     5 f        29     18     18      3       39     4
##  6     6 f        24      5     14     17       36     5
##  7     7 f        26     14     22      1       37     5
##  8     8 m        30     21      3      2       26     5
##  9     9 m        33      5     17     14       36     5
## 10    10 f        33      4      3      7       14     5
## 11    11 m        25      4      5     23       32     5
```

- Find the values of the variables id, sex and grade sorted by the values of sex of all students who have passed.

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

```
## # A tibble: 2 x 3
##      id sex   grade
##   <dbl> <chr> <dbl>
## 1     5 f         4
## 2     3 m         4
```

- Calculate the mean, minimum, maximum and median of the variable sum of scores grouped by the variable sex.

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

```
## # A tibble: 2 x 5
##   sex   mean_scores min_scores max_scores med_scores
##   <chr>       <dbl>      <dbl>      <dbl>      <dbl>
## 1 f            29.3         14         39         32
## 2 m            33.2         26         39         34
```

Task 4: Some data manipulations with the data set flights.

- Load the libraries tidyverse and nycflight13 and inspect the variable of flights.

```
library(tidyverse)
library(nycflights13)
?flights()
```

```
## starting httpd help server ... done
```

- Find all flights with more than 2 hours arrival delay.

```
flights %>% filter(arr_delay > 120)
```

```
## # A tibble: 10,034 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      811            630       101     1047            830
##  2  2013     1     1      848           1835       853     1001           1950
##  3  2013     1     1      957            733       144     1056            853
##  4  2013     1     1     1114            900       134     1447           1222
##  5  2013     1     1     1505           1310       115     1638           1431
##  6  2013     1     1     1525           1340       105     1831           1626
##  7  2013     1     1     1549           1445        64     1912           1656
##  8  2013     1     1     1558           1359       119     1718           1515
##  9  2013     1     1     1732           1630        62     2028           1825
## 10  2013     1     1     1803           1620       103     2008           1750
## # ... with 10,024 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

- Find all flights with more than 2 hours arrival delay and no departure delay.

```
flights %>% filter(arr_delay > 120 & dep_delay <= 0)
```

```
## # A tibble: 29 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1    27     1419           1420        -1     1754           1550
##  2  2013    10     7     1350           1350         0     1736           1526
##  3  2013    10     7     1357           1359        -2     1858           1654
##  4  2013    10    16      657            700        -3     1258           1056
##  5  2013    11     1      658            700        -2     1329           1015
##  6  2013     3    18     1844           1847        -3       39           2219
##  7  2013     4    17     1635           1640        -5     2049           1845
##  8  2013     4    18      558            600        -2     1149            850
##  9  2013     4    18      655            700        -5     1213            950
## 10  2013     5    22     1827           1830        -3     2217           2010
## # ... with 19 more rows, and 11 more variables: arr_delay <dbl>, carrier <chr>,
## #   flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
## #   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

- Find all flights from United, American and Delta with no arrival delay.

```
flights %>%
  filter(carrier %in% c("AA","DL","UA")) %>%
  filter(arr_delay <= 0)
```

```
## # A tibble: 88,046 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      554            600        -6      812            837
##  2  2013     1     1      558            600        -2      923            937
##  3  2013     1     1      559            600        -1      854            902
##  4  2013     1     1      602            610        -8      812            820
##  5  2013     1     1      606            610        -4      858            910
##  6  2013     1     1      606            610        -4      837            845
##  7  2013     1     1      607            607         0      858            915
##  8  2013     1     1      615            615         0      833            842
##  9  2013     1     1      628            630        -2     1137           1140
## 10  2013     1     1      643            646        -3      922            940
## # ... with 88,036 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

- 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.

```
flights %>%
  filter(carrier %in% c("UA","AA","DL")) %>%
  filter(month == 5) %>%
  filter(arr_delay > 300) %>%
  select(carrier, flight) %>%
  arrange(carrier,flight) %>%
  # remove multiple entries
  unique()
```

```
## # A tibble: 17 x 2
##    carrier flight
##    <chr>    <int>
##  1 AA         257
##  2 AA         341
##  3 AA         731
##  4 AA         753
##  5 DL         141
##  6 DL         781
##  7 DL         985
##  8 DL        1174
##  9 DL        1619
## 10 DL        1947
## 11 UA         497
## 12 UA         595
## 13 UA         691
## 14 UA         810
## 15 UA        1105
## 16 UA        1112
## 17 UA        1164
```

- 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 %>%
  mutate(dep_time =
           (dep_time %/% 100)*60 + dep_time %% 100) %>%
  mutate(arr_time =
           (arr_time %/% 100)*60 + arr_time %% 100)
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <dbl>          <int>     <dbl>    <dbl>          <int>
##  1  2013     1     1      317            515         2      510            819
##  2  2013     1     1      333            529         4      530            830
##  3  2013     1     1      342            540         2      563            850
##  4  2013     1     1      344            545        -1      604           1022
##  5  2013     1     1      354            600        -6      492            837
##  6  2013     1     1      354            558        -4      460            728
##  7  2013     1     1      355            600        -5      553            854
##  8  2013     1     1      357            600        -3      429            723
##  9  2013     1     1      357            600        -3      518            846
## 10  2013     1     1      358            600        -2      473            745
## # ... with 336,766 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

- Add a column speed which denotes the average speed of the flight and determine the carrier, flight of the top 10 values of speed.

```
flights %>%
  mutate(speed = distance / air_time * 60) %>%
  select(carrier,flight,speed) %>%
  arrange(desc(speed)) %>% top_n(10,speed)
```

```
## # A tibble: 15 x 3
##    carrier flight speed
##    <chr>    <int> <dbl>
##  1 DL        1499  703.
##  2 EV        4667  650.
##  3 EV        4292  648
##  4 EV        3805  641.
##  5 DL        1902  591.
##  6 DL         315  564
##  7 B6         707  557.
##  8 AA         936  556.
##  9 DL         347  554.
## 10 B6        1503  554.
## 11 DL         301  554.
## 12 DL         347  554.
## 13 AA        1029  554.
## 14 DL         329  554.
## 15 AA        1613  554.
```

- 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.

```
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, del_ratio) %>%
  # remove multiple entries
  unique() %>%
  arrange(desc(del_ratio))
```

```
## # A tibble: 16 x 3
## # Groups:   carrier [16]
##    carrier   nof del_ratio
##    <chr>   <int>     <dbl>
##  1 HA        342     0.810
##  2 AS        709     0.810
##  3 VX       5116     0.771
##  4 DL      47658     0.768
##  5 AA      31947     0.766
##  6 US      19831     0.765
##  7 OO         29     0.759
##  8 UA      57782     0.729
##  9 9E      17294     0.709
## 10 WN      12044     0.693
## 11 B6      54049     0.684
## 12 MQ      25037     0.665
## 13 EV      51108     0.634
## 14 YV        544     0.627
## 15 FL       3175     0.574
## 16 F9        681     0.551
```

- Find a list which denotes for every month the carrier with highest ratio. The list sould have the columns month, carrier, number of flights of the carrier in that month and ratio.

```
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,
         max_ratio = max(del_ratio)) %>%
  # keep only 4 columns
  select(month, carrier, nof, max_ratio) %>%
  # calculation per month
  group_by(month) %>%
  # only highest ratio
  filter(max_ratio == max(max_ratio)) %>%
  # remove multiple entries
  unique() %>%
  arrange(month)
```

```
## # A tibble: 12 x 4
## # Groups:   month [12]
##    month carrier   nof max_ratio
##    <int> <chr>   <int>     <dbl>
##  1     1 VX        314     0.924
##  2     2 HA         28     0.893
##  3     3 VX        303     0.848
##  4     4 HA         30     0.867
##  5     5 HA         31     0.935
##  6     6 AS         60     0.783
##  7     7 AS         62     0.823
##  8     8 AS         62     0.903
##  9     9 AS         60     0.967
## 10    10 HA         21     0.952
## 11    11 AS         52     0.865
## 12    12 HA         28     0.857
```

- Find a table with the number of cancelled flights (dep_delay = NA), the number of flights with no 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
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")
  )
```

```
## # A tibble: 365 x 6
## # Groups:   month [12]
##    month   day nof_canc nof_no_delay mean_dep_del mean_arr_del
##    <int> <int>    <int>        <int>        <dbl>        <dbl>
##  1     1     1        4          471        11.5        12.7
##  2     1     2        8          488        13.9        12.7
##  3     1     3       10          496        11.0         5.73
##  4     1     4        6          486         8.95       -1.93
##  5     1     5        3          429         5.73       -1.53
##  6     1     6        1          470         7.15        4.24
##  7     1     7        3          522         5.42       -4.95
##  8     1     8        4          515         2.55       -3.23
##  9     1     9        5          479         2.28       -0.264
## 10     1    10        3          506         2.84       -5.90
## # ... with 355 more rows
```