Sheet II: Exercises Datamanipulations with R

Dr. Falkenberg

WS 2021/22

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
```

student <chr>	algebra <dbl>	analysis <dbl>	diskrete.math <dbl>
Adam	*NA*	2	3
Bernd	5	*NA*	*NA*
Christian	3	1	2
Doris	4	3	4

4 rows

```
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
```

name <chr>	type <chr>	measure <dbl>
Adam	height	1.83

name <chr>	type <chr>	measure <dbl>
Adam	weight	81.00
Bernd	height	1.75
Bernd	weight	71.00
Christian	height	1.69
Christian	weight	55.00
Doris	height	1.57
Doris	weight	62.00

8 rows

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

name <chr>	ratio <chr>
Adam	81/1.83
Bernd	71/1.75
Christian	55/1.69
Doris	62/1.57

4 rows

- 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")
```

student <chr>	exam <chr>	grade <dbl>
Adam	algebra	*NA*
Bernd	algebra	5
Christian	algebra	3
Doris	algebra	4
Adam	analysis	2
Bernd	analysis	*NA*
Christian	analysis	1
Doris	analysis	3
Adam	diskrete.math	3
Bernd	diskrete.math	*NA*

1-10 of 12 rows Previous **1** 2 Next

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

name <chr>	height <dbl>	weight <dbl>
Adam	1.83	81
Bernd	1.75	71
Christian	1.69	55
Doris	1.57	62

4 rows

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

name <chr>	weight <chr>	height <chr>
Adam	81	1.83
Bernd	71	1.75
Christian	55	1.69
Doris	62	1.57

4 rows

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 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))
)
df
```

id <int>	sex <chr>	age <dbl>	score1 <dbl>
1	f	28	14
2	m	20	5
3	f	32	7
4	f	31	4
5	m	34	8
6	m	22	20
7	m	32	24
8	m	21	8
9	f	24	21
10	m	28	6

1-10 of 10 rows

- Select the date of all male students.

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

id <int>	sex <chr>	age <dbl>	score1 <dbl>
2	m	20	5
5	m	34	8
6	m	22	20
7	m	32	24
8	m	21	8

id	sex	age	score1
<int>	<chr>	<dbl>	<dbl>
10	m	28	6

6 rows

- 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
```

id	sex	age	score1
<dbl>	<chr>	<dbl>	<dbl>
1	f	28	14
2	m	20	5
3	f	32	7
4	f	31	4
5	m	34	8
6	m	22	20
7	m	32	24
8	m	21	8
9	f	24	21
10	m	28	6

1-10 of 11 rows Previous **1** 2 Next

- 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
```

id <dbl>	sex <chr>	age <dbl>	score1 <dbl>	score2 <dbl>	score3 <dbl>	scoresum <dbl>	grade <dbl>
1	f	28	14	10	2	26	5
2	m	20	5	14	7	26	5
3	f	32	7	13	9	29	5
4	f	31	4	19	22	45	4
5	m	34	8	19	15	42	4
6	m	22	20	24	22	66	1
7	m	32	24	24	2	50	3
8	m	21	8	7	24	39	4
9	f	24	21	9	10	40	4
10	m	28	6	9	24	39	4

1-10 of 11 rows Previous **1** 2 Next

- 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)
```

id <dbl>	sex <chr>	grade <dbl>
4	f	4
9	f	4
5	m	4
6	m	1
7	m	3
8	m	4
10	m	4
11	m	4

8 rows

- 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))
```

sex <chr>	mean_scores <dbl>	min_scores <dbl>	max_scores <dbl>	med_scores <dbl>
f	35.00000	26	45	34.5
m	43.14286	26	66	40.0

2 rows

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)
```

```
## Warning: package 'nycflights13' was built under R version 4.0.5
```

```
?flights()
```

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

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

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

y… <int>	mo… <int>	da <int>	dep_time <int>	sched_dep_time <int>	dep_delay <dbl>	arr_time <int>	sched_arr_time <int>	arr_delay <dbl>	ca <c
2013	1	1	811	630	101	1047	830	137	M
2013	1	1	848	1835	853	1001	1950	851	M
2013	1	1	957	733	144	1056	853	123	U
2013	1	1	1114	900	134	1447	1222	145	U
2013	1	1	1505	1310	115	1638	1431	127	E
2013	1	1	1525	1340	105	1831	1626	125	B
2013	1	1	1549	1445	64	1912	1656	136	E

y…	mo…	da	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay	ca
<int>	<int>	<int>	<int>	<int>	<dbl>	<int>	<int>	<dbl>	<(
2013	1	1	1558	1359	119	1718	1515	123	E
2013	1	1	1732	1630	62	2028	1825	123	E
2013	1	1	1803	1620	103	2008	1750	138	M

1-10 of 10,000 rows | 1-10 of 19 columns Previous **1** 2 3 4 5 6 … 1000 Next

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

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

y…	mo…	da	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay	ca
<int>	<int>	<int>	<int>	<int>	<dbl>	<int>	<int>	<dbl>	<(
2013	1	27	1419	1420	-1	1754	1550	124	M
2013	10	7	1350	1350	0	1736	1526	130	E
2013	10	7	1357	1359	-2	1858	1654	124	A
2013	10	16	657	700	-3	1258	1056	122	B
2013	11	1	658	700	-2	1329	1015	194	V
2013	3	18	1844	1847	-3	39	2219	140	U
2013	4	17	1635	1640	-5	2049	1845	124	M
2013	4	18	558	600	-2	1149	850	179	A
2013	4	18	655	700	-5	1213	950	143	A
2013	5	22	1827	1830	-3	2217	2010	127	M

1-10 of 29 rows | 1-10 of 19 columns Previous **1** 2 3 Next

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

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

y…	mo…	da	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay	ca
<int>	<int>	<int>	<int>	<int>	<dbl>	<int>	<int>	<dbl>	<(
2013	1	1	554	600	-6	812	837	-25	D
2013	1	1	558	600	-2	923	937	-14	U
2013	1	1	559	600	-1	854	902	-8	U
2013	1	1	602	610	-8	812	820	-8	D

y… <int>	mo… <int>	da <int>	dep_time <int>	sched_dep_time <int>	dep_delay <dbl>	arr_time <int>	sched_arr_time <int>	arr_delay <dbl>	ca <ch
2013	1	1	606	610	-4	858	910	-12	AA
2013	1	1	606	610	-4	837	845	-8	DL
2013	1	1	607	607	0	858	915	-17	UA
2013	1	1	615	615	0	833	842	-9	DL
2013	1	1	628	630	-2	1137	1140	-3	AA
2013	1	1	643	646	-3	922	940	-18	UA

1-10 of 10,000 rows | 1-10 of 19 columns Previous **1** 2 3 4 5 6 … 1000 Next

- 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()
```

carrier <chr>	flight <int>
AA	257
AA	341
AA	731
AA	753
DL	141
DL	781
DL	985
DL	1174
DL	1619
DL	1947

1-10 of 17 rows Previous **1** 2 Next

- 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)
```

y… <int>	mo… <int>	da <int>	dep_time <dbl>	sched_dep_time <int>	dep_delay <dbl>	arr_time <dbl>	sched_arr_time <int>	arr_delay <dbl>	ca <ch
2013	1	1	317	515	2	510	819	11	U
2013	1	1	333	529	4	530	830	20	U
2013	1	1	342	540	2	563	850	33	A
2013	1	1	344	545	-1	604	1022	-18	B
2013	1	1	354	600	-6	492	837	-25	D
2013	1	1	354	558	-4	460	728	12	U
2013	1	1	355	600	-5	553	854	19	B
2013	1	1	357	600	-3	429	723	-14	E
2013	1	1	357	600	-3	518	846	-8	B
2013	1	1	358	600	-2	473	745	8	A

1-10 of 10,000 rows | 1-10 of 19 columns Previous **1** 2 3 4 5 6 … 1000 Next

- 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)
```

carrier <chr>	flight <int>	speed <dbl>
DL	1499	703.3846
EV	4667	650.3226
EV	4292	648.0000
EV	3805	641.1429

carrier <chr>	flight <int>	speed <dbl>
DL	1902	591.4286
DL	315	564.0000
B6	707	557.4419
AA	936	556.4571
DL	347	554.2197
B6	1503	554.2197

1-10 of 15 rows Previous **1** 2 Next

- 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))
```

carrier <chr>	nof <int>	del_ratio <dbl>
HA	342	0.8099415
AS	709	0.8095910
VX	5116	0.7705238
DL	47658	0.7678249
AA	31947	0.7664256
US	19831	0.7650648
OO	29	0.7586207
UA	57782	0.7293794

carrier <chr>	nof <int>	del_ratio <dbl>
9E	17294	0.7088586
WN	12044	0.6926270

1-10 of 16 rows Previous **1** 2 Next

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

month <int>	carrier <chr>	nof <int>	del_ratio <dbl>
1	VX	314	0.9235669
2	HA	28	0.8928571
3	VX	303	0.8481848
4	HA	30	0.8666667
5	HA	31	0.9354839
6	AS	60	0.7833333
7	AS	62	0.8225806
8	AS	62	0.9032258
9	AS	60	0.9666667

month <int>	carrier <chr>	nof <int>	del_ratio <dbl>
10	HA	21	0.9523810

1-10 of 12 rows Previous **1** 2 Next

- 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")
  )
```

```
## `summarise()` has grouped output by 'month'. You can override using the `.groups`
argument.
## `summarise()` has grouped output by 'month'. You can override using the `.groups`
argument.
## `summarise()` has grouped output by 'month'. You can override using the `.groups`
argument.
```

month <int>	day <int>	nof_canc <int>	nof_no_delay <int>	mean_dep_del <dbl>	mean_arr_del <dbl>
1	1	4	471	11.54892601	12.651022864
1	2	8	488	13.85882353	12.692887931

month <int>	day <int>	nof_canc <int>	nof_no_delay <int>	mean_dep_del <dbl>	mean_arr_del <dbl>
1	3	10	496	10.98783186	5.733333333
1	4	6	486	8.95159516	-1.932819383
1	5	3	429	5.73221757	-1.525801953
1	6	1	470	7.14801444	4.236429433
1	7	3	522	5.41720430	-4.947311828
1	8	4	515	2.55307263	-3.227578475
1	9	5	479	2.27647715	-0.264277716
1	10	3	506	2.84499462	-5.898815931

1-10 of 365 rows Previous **1** 2 3 4 5 6 ... 37 Next