MLW / KUHeS Statistics and R short course

Session 1 - Practical (solutions)

Marc Henrion, Evaristar Kudowa, Vester Gunsaru

2024-10-08

Session 1 - Practical (Solutions)

Go to the course website on GitHub:

https://github.com/mlw-stats/R_And_Statistics_Training_Autumn2024/Session1

From here, download the following files:

```
btTBreg.csv
btTBregHospitals.csv
btTBreg_info.txt
```

1. Load the `btTBreg.csv` data table into R.

```
btDat<-read.csv("dataAndSupportDocs/btTBreg.csv")

head(btDat) # have a look at the data
```

```
  id age sex hiv   bmi ses cd41 cd42   cd41.sk  cd42.sk hosp
1  1  44   2   0 26.32   4  346  519 313.11656 572.8906    1
2  2  32   2   0 20.79   5  237  337  43.12752 406.1971    5
3  3  32   1   0 19.21   1  198  328 338.32172 408.2427    2
4  4  20   1   0 21.34   4  246  525  77.08697 312.7572    3
5  5  30   1   0 23.98   4  270  444 169.02539 335.3739    3
6  6  32   1   0 17.97   4  283  372 255.45773 323.4773    4
```

```
dim(btDat) # check dimesnions of data table
```

```
[1] 3000   11
```

2. The variables `cd41`, `cd42` and `cd41.sk`, `cd42.sk` measure the same variables (`cd4` and `cd4.sk` respectively) in the same individuals at two different time point. This means the data are in wide format. Reformat to long format.

They key difficulty here is that you have 2 variables (at 2 times points). In the example from lectures we only had 1 variable (at 3 different conditions). One approach is to do each variable separately, then combine the resulting data frames:

```
btDatLong.cd4<-btDat %>%
  pivot_longer(names_to="time", values_to="cd4", cols=c(cd41, cd42)) %>%
  select(id,age,sex,hiv,bmi,ses,hosp,time,cd4)

btDatLong.cd4sk<-btDat%>%
  pivot_longer(names_to="time", values_to="cd4.sk", cols=c(cd41.sk, cd42.sk)) %>%
  select(id,age,sex,hiv,bmi,ses,hosp,time,cd4.sk)

btDatLong<-data.frame(btDatLong.cd4,cd4.sk=btDatLong.cd4sk$cd4.sk)

rm(btDatLong.cd4,btDatLong.cd4sk)

btDatLong$time<-factor(
  case_when(
    btDatLong$time=="cd41"~"entry",
    btDatLong$time=="cd42"~"exit",
    TRUE~NA_character_)
  ) # rename the levels of the time variable

head(btDatLong) # have a look at the data
```

```
  id age sex hiv   bmi ses hosp  time cd4    cd4.sk
1  1  44   2   0 26.32   4    1 entry 346 313.11656
2  1  44   2   0 26.32   4    1  exit 519 572.89062
3  2  32   2   0 20.79   5    5 entry 237  43.12752
4  2  32   2   0 20.79   5    5  exit 337 406.19707
5  3  32   1   0 19.21   1    2 entry 198 338.32172
6  3  32   1   0 19.21   1    2  exit 328 408.24267
```

```
dim(btDatLong) # check dimensions
```

```
[1] 6000   10
```

This can be done a bit more directly, by using regular expression (character expressions that match flexibly to names) and a combination of pivot_longer() and pivot_wider():

```
btDatLong<-btDat %>%
  pivot_longer(cols=c(cd41, cd42, cd41.sk, cd42.sk),
               names_pattern = "cd4(1|2)(.*)",
               names_to = c("time","cd4"),
               values_to="measurement") %>%
  mutate(cd4=paste(sep="","cd4",cd4)) %>%
  pivot_wider(names_from=cd4,values_from=measurement)

head(btDatLong) # have a look at the data
```

```
# A tibble: 6 x 10
     id   age   sex   hiv   bmi   ses  hosp time    cd4 cd4.sk
  <int> <int> <int> <int> <dbl> <int> <int> <chr> <dbl>  <dbl>
1     1    44     2     0  26.3     4     1 1       346  313.
2     1    44     2     0  26.3     4     1 2       519  573.
3     2    32     2     0  20.8     5     5 1       237   43.1
4     2    32     2     0  20.8     5     5 2       337  406.
5     3    32     1     0  19.2     1     2 1       198  338.
6     3    32     1     0  19.2     1     2 2       328  408.
```

```
dim(btDatLong) # check dimensions
```

```
[1] 6000   10
```

The code above requires a bit of unpacking:

- The expression in brackets in the "names_pattern" argument are regular expression matching sequences of character: "(1|2)" matches 1 or 2 and "(.*)" matches anything.
- The "mutate()" line is needed as the values store in the "cd4" column are " " and ".sk" – as the pivot_wider() statement on the next line will use those as column names, we cannot have an empty column name – " " would trigger an error message. So we just add the characters "cd4" in front of the stores values – i.e. we then have "cd4" and "cd4.sk" rather than " " and ".sk".

An alternative function that can be used is `reshape()`. To get more information on this function, type `?reshape` at the console.

```
btDatLong<-reshape(btDat,
                   direction="long",
                   varying=list(c("cd41","cd42"),c("cd41.sk","cd42.sk")),
                   ids="id",
                   v.names=c("cd4","cd4.sk"))

head(btDatLong) # have a look at the data
```

```
    id age sex hiv   bmi ses hosp time cd4    cd4.sk
1.1  1  44   2   0 26.32   4    1    1 346 313.11656
2.1  2  32   2   0 20.79   5    5    1 237  43.12752
3.1  3  32   1   0 19.21   1    2    1 198 338.32172
4.1  4  20   1   0 21.34   4    3    1 246  77.08697
5.1  5  30   1   0 23.98   4    3    1 270 169.02539
6.1  6  32   1   0 17.97   4    4    1 283 255.45773
```

```
dim(btDatLong) # check dimensions
```

```
[1] 6000   10
```

3. Save the reformatted data into a file called `btTBregLong.tab` in such a way that

   i. Columns are tab-separated.
   ii. Column names are saved.
   iii. No row number is saved in the resulting file.

```
dir.create("Session1_output",showWarnings=F)
write.table(btDatLong,sep="\t",col.names=T,row.names=F,file="Session1_output/btTBregLong.t
```

4. Copy the code below to generate some wide-format data. We will assume this dataset contains observations of 2 biomarkers, `ferritin` and `rbp4` for 10 study participants at 2 different timepoints, `day1` and `day90`.

```
set.seed(123)

df<-data.frame(
  id=paste(sep="","P",1:10),
  ferritin_day1=rexp(10,rate=1/195),
  rbp4_day1=rexp(10,rate=1/2.5)
) %>%
  mutate(
    ferritin_day90=rnorm(10,mean=ferritin_day1+5,sd=4),
    rbp4_day90=rbp4_day1+rexp(10,rate=1/0.25)
  )
```

This is what this data table looks like:

```
    id ferritin_day1  rbp4_day1 ferritin_day90 rbp4_day90
1   P1    164.474166  2.5120751     169.251422   3.021712
2   P2    112.439003  1.2005368     114.301474   1.552174
3   P3    259.165699  0.7025341     261.231686   0.829188
4   P4      6.157585  0.9427946      10.294123   1.007684
5   P5     10.961140  0.4707101      14.621489   1.119933
6   P6     61.717737  2.1244653      62.374941   2.431722
7   P7     61.274322  3.9080088      65.932629   4.105679
8   P8     28.327027  1.1969010      37.609469   1.354221
9   P9    531.616111  1.4773371     536.034536   1.790997
10 P10      5.684922 10.1025293       6.022743  10.249700
```

Reformat this to long format, i.e. so that you have 4 columns: `id`, `time`, `ferritin` and `rbp4`.

```
dfLong<-df %>%
  pivot_longer(cols=c(ferritin_day1,rbp4_day1,ferritin_day90,rbp4_day90),
               names_pattern="(.*)_(.*)",
               names_to=c("biomarker","timepoint"),
               values_to="value") %>%
  pivot_wider(names_from=biomarker,values_from=value)

print(dfLong)
```

```
# A tibble: 20 x 4
   id    timepoint ferritin   rbp4
   <chr> <chr>        <dbl>  <dbl>
 1 P1    day1         164.   2.51
 2 P1    day90        169.   3.02
 3 P2    day1         112.   1.20
 4 P2    day90        114.   1.55
 5 P3    day1         259.   0.703
 6 P3    day90        261.   0.829
 7 P4    day1          6.16  0.943
 8 P4    day90        10.3   1.01 
 9 P5    day1         11.0   0.471
10 P5    day90        14.6   1.12 
11 P6    day1         61.7   2.12 
12 P6    day90        62.4   2.43 
13 P7    day1         61.3   3.91 
14 P7    day90        65.9   4.11 
15 P8    day1         28.3   1.20 
16 P8    day90        37.6   1.35 
17 P9    day1        532.    1.48 
18 P9    day90       536.    1.79 
19 P10   day1          5.68 10.1  
20 P10   day90         6.02 10.2  
```

5. Load the `btTBregHospitals.csv` data table. Join the data frames storing `btTBreg.csv` and `btTBregHospitals.csv`.

```
btDatHosp<-read.csv("dataAndSupportDocs/btTBregHospitals.csv")

head(btDatHosp) # have a look at the data
```

```
  HID ShortName                          FullName beds     city
1   1      QECH Queen Elizabeth Central Hospital 1000 Blantyre
2   2       KCH          Kamuzu Central Hospital 1000 Lilongwe
3   3       ZCH           Zomba Central Hospital  400    Zomba
4   4       MCH           Mzuzu Central Hospital  350    Mzuzu
5   5    Mlambe          Mlambe Mission Hospital  254    Lunzu
```

```
dim(btDatHosp) # check dimensions of the data table
```

```
[1] 5 5
```

```
btDatJoined<-btDat %>%
  inner_join(btDatHosp,by=c("hosp"="HID"))

head(btDatJoined) # have a look
```

```
  id age sex hiv   bmi ses cd41 cd42    cd41.sk  cd42.sk hosp ShortName
1  1  44   2   0 26.32   4  346  519 313.11656 572.8906    1      QECH
2  2  32   2   0 20.79   5  237  337  43.12752 406.1971    5    Mlambe
3  3  32   1   0 19.21   1  198  328 338.32172 408.2427    2       KCH
4  4  20   1   0 21.34   4  246  525  77.08697 312.7572    3       ZCH
5  5  30   1   0 23.98   4  270  444 169.02539 335.3739    3       ZCH
6  6  32   1   0 17.97   4  283  372 255.45773 323.4773    4       MCH
                        FullName beds     city
1 Queen Elizabeth Central Hospital 1000 Blantyre
2          Mlambe Mission Hospital  254    Lunzu
3          Kamuzu Central Hospital 1000 Lilongwe
4           Zomba Central Hospital  400    Zomba
5           Zomba Central Hospital  400    Zomba
6           Mzuzu Central Hospital  350    Mzuzu
```

```
dim(btDatJoined) # check dimensions
```

```
[1] 3000   15
```

6. Compute the average patient age and the proportion of male patients for each hospital.

Useful functions for this are `aggregate()` and `group_by()`. You can however also do it manually.

- Manually:

```
# initialise new variables
btDatHosp$avgAge<-NA
btDatHosp$propMale<-NA

# iterate over hospitals
for(i in 1:nrow(btDatHosp)){
  btDatHosp$avgAge[i]<-mean(btDatJoined$age[btDatJoined$ShortName==btDatHosp$ShortName[i]]
  btDatHosp$propMale[i]<-sum(btDatJoined$sex==1 &
                              btDatJoined$ShortName==btDatHosp$ShortName[i]) /
                          sum(btDatJoined$ShortName==btDatHosp$ShortName[i])
}

print(btDatHosp)
```

```
  HID ShortName                         FullName beds     city   avgAge
1   1      QECH Queen Elizabeth Central Hospital 1000 Blantyre 33.14020
2   2       KCH          Kamuzu Central Hospital 1000 Lilongwe 32.80067
3   3       ZCH           Zomba Central Hospital  400    Zomba 32.99310
4   4       MCH           Mzuzu Central Hospital  350    Mzuzu 32.87382
5   5    Mlambe          Mlambe Mission Hospital  254    Lunzu 32.89950
   propMale
1 0.4763514
2 0.4757119
3 0.4948276
4 0.4731861
5 0.5242881
```

- Using `aggregate()`

```
btDat$hosp<-factor(btDat$hosp)
btDatHosp$avgAge<-aggregate(btDatJoined$age,FUN=mean,by=list(btDat$hosp))$x
btDatHosp$propMale<-aggregate(ifelse(btDatJoined$sex==1,1,0),FUN=mean,by=list(btDat$hosp))

print(btDatHosp)
```

```
  HID ShortName                         FullName beds     city   avgAge
1   1      QECH Queen Elizabeth Central Hospital 1000 Blantyre 33.14020
2   2       KCH          Kamuzu Central Hospital 1000 Lilongwe 32.80067
3   3       ZCH           Zomba Central Hospital  400    Zomba 32.99310
4   4       MCH           Mzuzu Central Hospital  350    Mzuzu 32.87382
5   5    Mlambe          Mlambe Mission Hospital  254    Lunzu 32.89950
   propMale
1 0.4763514
2 0.4757119
3 0.4948276
4 0.4731861
5 0.5242881
```

- Using `group_by()`

```
tmp<-btDat %>%
  group_by(hosp) %>%
  summarise(avgAge=mean(age,na.rm=T))
btDatHosp$avgAge<-tmp$avgAge
```

```
tmp<-btDat %>%
  group_by(hosp) %>%
  summarise(propMale=mean(ifelse(sex==1,1,0),na.rm=T))
btDatHosp$propMale<-tmp$propMale

print(btDatHosp)
```

```
  HID ShortName                        FullName beds     city   avgAge
1   1      QECH Queen Elizabeth Central Hospital 1000 Blantyre 33.14020
2   2       KCH          Kamuzu Central Hospital 1000 Lilongwe 32.80067
3   3       ZCH           Zomba Central Hospital  400    Zomba 32.99310
4   4       MCH           Mzuzu Central Hospital  350    Mzuzu 32.87382
5   5    Mlambe          Mlambe Mission Hospital  254    Lunzu 32.89950
   propMale
1 0.4763514
2 0.4757119
3 0.4948276
4 0.4731861
5 0.5242881
```

7. Write an R function that computes the following summary statistics, then, using your custom function, compute these for the `bmi`, `cd41`, `cd42` columns:

   i. mean
   ii. median
   iii. interquartile range
   iv. minimum
   v. maximum
   vi. number of missing values

```
summaryFun<-function(x){
  return(c(
    mean(x,na.rm=T),
    median(x),
    paste(sep="","(",paste(collapse=",",quantile(x,probs=c(0.25,0.75))),")"),
    min(x,na.rm=T),
    max(x,na.rm=T),
    sum(is.na(x))
  ))
}

res<-apply(btDat[,c("bmi","cd41","cd42")],MARGIN=2,FUN=summaryFun)
```

```
rownames(res)<-c("mean","median","IQR","min","max","num_MV")
print(res)
```

```
       bmi                cd41                cd42
mean   "23.0574333333333" "248.794333333333" "448.003"
median "23.05"            "249"              "447"
IQR    "(21.34,24.74)"    "(216,281)"        "(381,515)"
min    "12.64"            "57"               "81"
max    "31.14"            "447"              "843"
num_MV "0"                "0"                "0"
```

8. Do the same now, but only for female patients. Repeat for only male patients.

```
resF<-apply(btDat[btDat$sex==2,c("bmi","cd41","cd42")],MARGIN=2,FUN=summaryFun)
rownames(resF)<-c("mean","median","IQR","min","max","num_MV")
print(resF)
```

```
       bmi                cd41                cd42
mean   "23.1218644067797" "248.473924380704" "446.675358539765"
median "23.14"            "250"              "447.5"
IQR    "(21.365,24.82)"   "(215,281)"        "(379,512)"
min    "12.64"            "57"               "138"
max    "31.14"            "447"              "820"
num_MV "0"                "0"                "0"
```

```
resM<-apply(btDat[btDat$sex==1,c("bmi","cd41","cd42")],MARGIN=2,FUN=summaryFun)
rownames(resM)<-c("mean","median","IQR","min","max","num_MV")
print(resM)
```

```
       bmi                cd41                cd42
mean   "22.9900136425648" "249.129604365621" "449.392223738063"
median "22.98"            "248"              "447"
IQR    "(21.3,24.66)"     "(216,282)"        "(383,519.75)"
min    "14.44"            "71"               "81"
max    "30.9"             "414"              "843"
num_MV "0"                "0"                "0"
```