Introduction to R

Session 4 – Data exploration

Statistical Consulting Centre

consulting@stat.auckland.ac.nz
The Department of Statistics
The University of Auckland

19 July, 2017

for loop to get column summary statistics

```r
for (i in c("Age.group", "Sex", "Smoke.group", "Race.group",
    "BMI.group")) {
    print(i)
    print(table(combined.df[, i]))
}
```

```
## [1] "Age.group"
##
## Under 35 36 to 60  Over 61
##     5585     5969     5476
## [1] "Sex"
##
## Female   Male
##   9077   7953
## [1] "Smoke.group"
##
##   No  Yes
```

for loop to get column summary statistics

```
Categorical.df <- combined.df[, c("Age.group", "Sex",
    "Smoke.group", "Race.group", "BMI.group")]

for (i in 1:ncol(Categorical.df)) {
    print(table(Categorical.df[, i]))
}
```

```
##
## Under 35 36 to 60  Over 61
##     5585     5969     5476
##
## Female   Male
##   9077   7953
##
##   No  Yes
## 4255 4371
##
```

for loop to get column summary statistics

```r
Continous.df <- combined.df[, c("Age", "Height", "Weight",
    "BMI", "Baseline", "PreTrt", "PostTrt")]


for (i in 1:ncol(Continous.df)) {
    print(i)
    print(mean(Continous.df[, i], na.rm = TRUE))
}
```

```
## [1] 1
## [1] 48.7919
## [1] 2
## [1] 65.43787
## [1] 3
## [1] 165.0315
## [1] 4
## [1] 27.03084
```

Smart way: (apply) to get column summary statistics

```
apply(X, MARGIN, FUN, ...)
```

- `X`: A data frame.
- `MARGIN`: 1 indicates rows, 2 indicates columns.
- `FUN`: function, what do you want `R` to do with the rows or columns of the data frame
- `...`: optional arguments to `FUN`.

Translation: Do something (`FUN`) to every row (or column) (`MARGIN`) of a data frame (`X`).

Smart way: (apply) to get column summary statistics

```
apply(Categorical.df, 2, table)
```

```
## $Age.group
##
## 36 to 60  Over 61 Under 35
##     5969     5476     5585
##
## $Sex
##
## Female   Male
##   9077   7953
##
## $Smoke.group
##
##   No  Yes
## 4255 4371
##
```

Smart way: (apply) to get column summary statistics

```
apply(Continous.df, 2, mean, na.rm = TRUE)
```

```
##       Age    Height    Weight       BMI  Baseline
##  48.79190  65.43787 165.03151  27.03084 206.04918
##    PreTrt   PostTrt
## 206.03711 206.03717
```

Smart way: (apply) to get column summary statistics

```r
apply(Continous.df, 2, sd, na.rm = TRUE)
```

```
##       Age    Height    Weight       BMI  Baseline
## 19.720956  3.913986 39.568004  5.833157 44.837071
##    PreTrt   PostTrt
## 45.254414 45.618231
```

apply() using self-defined R function

Functions used in apply() can be self-defined.

```
na.check <- function(someinput) {
    test.na <- is.na(someinput)
    sum(test.na)
}
```

Take an educated guess at what na.check() does.

apply() using a self-defined R function

Let's look at what each row of na.check() does.

```
test1 <- Continous.df$BMI[1:10]
test1
```

```
##  [1] 25.46094       NA 27.55498 29.36357       NA
##  [6] 27.04702 22.56770 20.81527 25.43695 36.90559
```

```
test.na <- is.na(test1)
test.na
```

```
##  [1] FALSE  TRUE FALSE FALSE  TRUE FALSE FALSE
##  [8] FALSE FALSE FALSE
```

```
sum(test.na)
```

```
## [1] 2
```

apply() using a self-defined R function

Let's now use na.check() in apply().

```
apply(Continous.df, 2, na.check)
```

```
##      Age   Height   Weight      BMI Baseline
##        0       40       55       63      968
##   PreTrt  PostTrt
##      968      968
```

Another way,

```
apply(Continous.df, 2, function(x) sum(is.na(x)))
```

```
##      Age   Height   Weight      BMI Baseline
##        0       40       55       63      968
##   PreTrt  PostTrt
##      968      968
```

A slightly more complicated function

```
mytab <- function(someinput){
 n <- length(someinput)
 n.missing <- na.check(someinput)
 n.complete <- n - n.missing
 mymean <- round(mean(someinput, na.rm = TRUE), 2)
 mysd <- round(sd(someinput, na.rm = TRUE), 2)
 mystder <- round(mysd/sqrt(n.complete), 2)
 Lower.CI <- round(mymean - 1.96*mystder, 2)
 Upper.CI <- round(mymean + 1.96*mystder, 2)
 c(Complete.obs = n.complete, Missing.obs = n.missing,
   Mean = mymean, Std.Error = mystder,
   Lower.CI = Lower.CI, Upper.CI = Upper.CI)
}
```

Take a *more* educated guess at what `mytab()` does?

A slightly more complicated function

- For the R novice, `mytab()` is possibly terrifying!
- We too were R novices once!
- Our advice on understanding what an R function does?

"Use a data set for input into the function and work through it one line of code at a time."

- We "experts" still do this!

mytab()

```r
apply(Continous.df, 2, mytab)
```

```
##                   Age   Height   Weight      BMI
## Complete.obs 17030.00 16990.00 16975.00 16967.00
## Missing.obs      0.00    40.00    55.00    63.00
## Mean            48.79    65.44   165.03    27.03
## Std.Error        0.15     0.03     0.30     0.04
## Lower.CI        48.50    65.38   164.44    26.95
## Upper.CI        49.08    65.50   165.62    27.11
##              Baseline   PreTrt  PostTrt
## Complete.obs 16062.00 16062.00 16062.00
## Missing.obs    968.00   968.00   968.00
## Mean           206.05   206.04   206.04
## Std.Error        0.35     0.36     0.36
## Lower.CI       205.36   205.33   205.33
## Upper.CI       206.74   206.75   206.75
```

More descriptive stats

Calculate the mean `Baseline` score for male and female patients

```
with(combined.df, mean(Baseline[Sex == "Male"],
                   na.rm = TRUE))
```

```
## [1] 203.637
```

```
with(combined.df, mean(Baseline[Sex == "Female"],
                   na.rm = TRUE))
```

```
## [1] 208.1786
```

Better way

```r
with(combined.df, tapply(Baseline, Sex, mean, na.rm = T))
```

```
##   Female     Male
## 208.1786 203.6370
```

tapply()

```
with(issp.df, tapply(total.lik, Income, mean, na.rm = TRUE))
```

```
tapply(X, INDEX, FUN, ...)
```

Translation: Apply function `FUN` to `X` for each level in the grouping factor `INDEX`

How about calculating the mean `Baseline` score for each Race group (3 levels)?

tapply()

```r
with(combined.df, tapply(Baseline, Race.group, mean,
    na.rm = TRUE))
```

```
##   African Caucasian     Other
##  201.5053  208.1396  199.9884
```

tapply()

```
with(combined.df, tapply(Baseline, Race.group, mytab))
## $African
## Complete.obs  Missing.obs         Mean    Std.Error
##      4427.00       433.00       201.51         0.68
##     Lower.CI     Upper.CI
##       200.18       202.84
##
## $Caucasian
## Complete.obs  Missing.obs         Mean    Std.Error
##     11111.00       501.00       208.14         0.42
##     Lower.CI     Upper.CI
##       207.32       208.96
##
## $Other
## Complete.obs  Missing.obs         Mean    Std.Error
##       519.00        34.00       199.99         1.90
##     Lower.CI     Upper.CI
##       196.27       203.71
```

Data cleaning

```r
str(combined.df)
```

```
## 'data.frame':    17030 obs. of  13 variables:
##  $ ID         : int  3 4 9 10 11 19 34 44 45 48 ...
##  $ Age        : int  21 32 48 35 48 44 42 24 67 56 ...
##  $ Sex        : Factor w/ 2 levels "Female","Male": 2 1 1 2
##  $ Weight     : num  180 NA 150 204 155 ...
##  $ Height     : num  70.4 63.9 61.8 69.8 NA 70.2 62.6 64.4
##  $ Smoke.group: Factor w/ 2 levels "No","Yes": NA NA 1 NA 1
##  $ Race.group : Factor w/ 3 levels "African","Caucasian",..
##  $ BMI        : num  25.5 NA 27.6 29.4 NA ...
##  $ BMI.group  : Factor w/ 3 levels "normal","overweight",..
##  $ Age.group  : Factor w/ 3 levels "Under 35","36 to 60",..
##  $ Baseline   : int  268 160 236 225 260 187 216 137 NA 156
##  $ PreTrt     : int  276 170 245 231 256 194 212 135 NA 157
##  $ PostTrt    : int  281 170 252 235 257 195 222 136 NA 159
```

Data cleaning

```r
str(combined.df[, 11:13])
```

```
## 'data.frame':    17030 obs. of  3 variables:
##  $ Baseline: int  268 160 236 225 260 187 216 137 NA 156 ..
##  $ PreTrt  : int  276 170 245 231 256 194 212 135 NA 157 ..
##  $ PostTrt : int  281 170 252 235 257 195 222 136 NA 159 ..
```

- a common problem is the dataset where some of the column names are not names of the variables, but *values* of variable.
- the last three columns are Serum Cholesterol levels, mg/100ml, measured on: Day 1, Day 5 and Day 10, which we have recoded to `Baseline`, `PreTrt` and `PostTrt`, resepctively.

tidyr R package

```
library(tidyr)
```

Two main functions: - `gather()` - takes multiple columns and combines based at a *key* value, i.e. `Time`. - `spread()` - opposite of `gather()`

gather()

```r
combined.long.df <- gather(combined.df, "Baseline", "PreTrt",
    "PostTrt", key = Time, value = Cholesterol)
```

gather()

```
combined.long.df <- gather(combined.df, key = Time,  value= Ch
                           -ID, -Age, -Age.group, -Sex, -Weigh
                           -Smoke.group, -Race.group, -BMI,
                           -BMI.group)
```

gather()

```
head(combined.long.df)
```

```
##   ID Age    Sex Weight Height Smoke.group Race.group
## 1  3  21   Male  179.5   70.4        <NA>  Caucasian
## 2  4  32 Female     NA   63.9        <NA>  Caucasian
## 3  9  48 Female  149.7   61.8          No  Caucasian
## 4 10  35   Male  203.5   69.8        <NA>  Caucasian
## 5 11  48   Male  155.3     NA          No  Caucasian
## 6 19  44   Male  189.6   70.2         Yes    African
##        BMI  BMI.group Age.group     Time Cholesterol
## 1 25.46094 overweight  Under 35 Baseline         268
## 2       NA       <NA>  Under 35 Baseline         160
## 3 27.55498 overweight  36 to 60 Baseline         236
## 4 29.36357 overweight  Under 35 Baseline         225
## 5       NA       <NA>  36 to 60 Baseline         260
## 6 27.04702 overweight  36 to 60 Baseline         187
```

gather()

```r
str(combined.long.df)
```

```
## 'data.frame':    51090 obs. of  12 variables:
##  $ ID          : int  3 4 9 10 11 19 34 44 45 48 ...
##  $ Age         : int  21 32 48 35 48 44 42 24 67 56 ...
##  $ Sex         : Factor w/ 2 levels "Female","Male": 2 1 1 2
##  $ Weight      : num  180 NA 150 204 155 ...
##  $ Height      : num  70.4 63.9 61.8 69.8 NA 70.2 62.6 64.4
##  $ Smoke.group: Factor w/ 2 levels "No","Yes": NA NA 1 NA 1
##  $ Race.group : Factor w/ 3 levels "African","Caucasian",..
##  $ BMI         : num  25.5 NA 27.6 29.4 NA ...
##  $ BMI.group  : Factor w/ 3 levels "normal","overweight",..
##  $ Age.group  : Factor w/ 3 levels "Under 35","36 to 60",..
##  $ Time        : chr  "Baseline" "Baseline" "Baseline" "Base
##  $ Cholesterol: int  268 160 236 225 260 187 216 137 NA 156
```

spread()

```
combined.wide.df <- spread(combined.long.df, key = Time, value
```

dplyr R package

```
Patient.df <- read.csv("..\\..\\data\\Patient.csv", stringsAsF

step1 <- mutate(Patient.df, Sex = as.factor(Sex), Smoke.group
    1, "Yes", "No")), Race.group = factor(ifelse(Race == 1, "C
    ifelse(Race == 2, "African", "Other"))), BMI = (Weight/Hei
    703)

step2 <- select(step1, -Smoke, -Race)

step3 <- mutate(step2, BMI.group = factor(ifelse(BMI >= 30, "o
    ifelse(BMI >= 25, "overweight", "normal")), levels = c("no
    "overweight", "obese")))

step4 <- mutate(step3, Age.group = factor(ifelse(Age <= 35, "U
    ifelse(Age <= 60, "36 to 60", "Over 61")), levels = c("Und
    "36 to 60", "Over 61")))
```

Piping %>%

`x %>% f(y)` is equivalent to `f(x, y)`, where `x` is typically the data-frame.

Piping operator %>%

```r
combine.wide.df <- Patient.df %>% mutate(
  Sex = as.factor(Sex),
  Smoke.group = factor(ifelse(Smoke == 1, "Yes", "No")),
  Race.group =  factor(ifelse(
    Race == 1, "Caucasian",
    ifelse(Race == 2, "African", "Other")
  )),
  BMI =  (Weight / Height ^ 2) * 703
) %>%
  select(-Smoke,-Race) %>%
  mutate(BMI.group = factor(
    ifelse(BMI >= 30, "obese",
           ifelse(BMI >= 25, "overweight",
                  "normal")),
    levels = c("normal", "overweight", "obese")
  )) %>%
```

Piping operator %>%

```
 mutate(Age.group =   factor(
    ifelse(Age <= 35, "Under 35",
           ifelse(Age <= 60, "36 to 60", "Over 61")),
    levels = c("Under 35", "36 to 60", "Over 61")
  )) %>%
  left_join(Cholesterol.df) %>% rename(
    ID = Patient.ID,
    Baseline = Day1,
    PreTrt = Day5,
    PostTrt = Day10
  )


combined.long.df <- combine.wide.df %>%
  gather("Baseline", "PreTrt", "PostTrt",
         key = Time, value= Cholesterol)
```

Grouping

```
with(combined.df, tapply(Baseline, Race.group, mean, na.rm = T
```

```
##   African Caucasian     Other
##  201.5053  208.1396  199.9884
```

```
combine.wide.df %>% group_by(Race.group) %>% summarise(BaseMea
    na.rm = TRUE))
```

```
## # A tibble: 4 x 2
##   Race.group BaseMean
##       <fctr>    <dbl>
## 1    African 201.5053
## 2  Caucasian 208.1396
## 3      Other 199.9884
## 4         NA 213.0000
```

Grouping

```
with(combined.df, tapply(Baseline, Race.group, mytab))
```

```
## $African
## Complete.obs  Missing.obs         Mean    Std.Error 
##      4427.00       433.00       201.51         0.68 
##     Lower.CI     Upper.CI 
##       200.18       202.84 
## 
## $Caucasian
## Complete.obs  Missing.obs         Mean    Std.Error 
##     11111.00       501.00       208.14         0.42 
##     Lower.CI     Upper.CI 
##       207.32       208.96 
## 
## $Other
## Complete.obs  Missing.obs         Mean    Std.Error 
##       519.00        34.00       199.99         1.90 
```

Grouping

```
combine.wide.df %>% group_by(Age.group, Race.group, BMI.group)
    count()
```

```
## # A tibble: 38 x 4
## # Groups:   Age.group, Race.group, BMI.group [38]
##    Age.group Race.group  BMI.group     n
##       <fctr>     <fctr>     <fctr> <int>
##  1  Under 35    African     normal   906
##  2  Under 35    African overweight   517
##  3  Under 35    African      obese   470
##  4  Under 35    African         NA     1
##  5  Under 35  Caucasian     normal  1800
##  6  Under 35  Caucasian overweight  1031
##  7  Under 35  Caucasian      obese   602
##  8  Under 35  Caucasian         NA    10
##  9  Under 35      Other     normal   139
## 10  Under 35      Other overweight    67
```

Grouping

```r
combine.wide.df %>% group_by(Age.group, Race.group, BMI.group)
    count() %>% arrange(desc(n))
```

```
## # A tibble: 38 x 4
## # Groups:   Age.group, Race.group, BMI.group [38]
##    Age.group Race.group  BMI.group     n
##       <fctr>     <fctr>     <fctr> <int>
##  1  Under 35  Caucasian     normal  1800
##  2   Over 61  Caucasian overweight  1715
##  3   Over 61  Caucasian     normal  1674
##  4  36 to 60  Caucasian overweight  1422
##  5  36 to 60  Caucasian     normal  1300
##  6  36 to 60  Caucasian      obese  1143
##  7  Under 35  Caucasian overweight  1031
##  8  Under 35    African     normal   906
##  9   Over 61  Caucasian      obese   882
## 10  36 to 60    African      obese   656
```

Grouping

```r
combine.wide.df %>% group_by(Age.group, Race.group, BMI.group)
    count() %>% spread(Age.group, n)
```

```
## # A tibble: 14 x 5
## # Groups:   Race.group, BMI.group [14]
##    Race.group  BMI.group `Under 35` `36 to 60` `Over 61`
##  *     <fctr>     <fctr>      <int>      <int>     <int>
##  1    African     normal        906        580       372
##  2    African overweight        517        654       375
##  3    African      obese        470        656       313
##  4    African         NA          1          3        13
##  5  Caucasian     normal       1800       1300      1674
##  6  Caucasian overweight       1031       1422      1715
##  7  Caucasian      obese        602       1143       882
##  8  Caucasian         NA         10          7        26
##  9      Other     normal        139         88        55
## 10      Other overweight         67         57        25
```

Grouping

```
combine.wide.df %>% group_by(Age.group, Race.group, BMI.group)
    count() %>% spread(Age.group, n) %>% filter(!is.na(BMI.gro
    !is.na(Race.group))
```

```
## # A tibble: 9 x 5
## # Groups:   Race.group, BMI.group [9]
##   Race.group  BMI.group `Under 35` `36 to 60` `Over 61`
##       <fctr>     <fctr>      <int>      <int>     <int>
## 1    African     normal        906        580       372
## 2    African overweight        517        654       375
## 3    African      obese        470        656       313
## 4  Caucasian     normal       1800       1300      1674
## 5  Caucasian overweight       1031       1422      1715
## 6  Caucasian      obese        602       1143       882
## 7      Other     normal        139         88        55
## 8      Other overweight         67         57        25
## 9      Other      obese         38         57        24
```

Summary

- `apply()`
- `tapply()`
- `gather()` and `spread()` of `tidyr` R package
- `dplyr` R package
- piping operator %>%