NZSSN Courses: Introduction to R

Session 4 – Data Exploration

Statistical Consulting Centre

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

1 March, 2017

SCIENCE
DEPARTMENT OF STATISTICS

WHOQOL

- Questionnaire usually comprise many items.
- Sometimes we add the likert scale of a set of questions (from the same section) to obtain a score. Then we will assume it's a measure of a certain aspect.
- For example, World Health Organisation Quality of Life (WHOQOL) measures. It develops three scores (physical score, psychosocial score and total generic score) from a large number of questions.
- Suppose we add the likert scale from Q1 to Q4 to get a total score, and let's call it the "Feminist score"

First step

Convert responses to Q1 – Q4 to likert scale.

We can do this using nested `ifelse()` statements, e.g.

```
Q1.lik<- with(issp.df,
              ifelse(Q1 == "strongly agree", 1,
              ifelse(Q1 == "agree", 2,
              ifelse(Q1 == "neither agree nor dis", 3,
              ifelse(Q1 == "disagree", 4,
              ifelse(Q1 == "strongly disagree", 5, NA))))))
```

```
# Generate one-way frequency table for likert scale
table(Q1.lik)
```

```
Q1.lik
  1   2   3   4   5
 98 297 331 263  21
```

Easier way

Recall that `Q1` – `Q4` are now factors. We can check this using `R`'s structure function, i.e.

```
str(issp.df[, c("Q1", "Q2", "Q3", "Q4")])

'data.frame': 1047 obs. of  4 variables:
 $ Q1: Factor w/ 5 levels "strongly agree",..: 4 5 4 NA 4 4 1
 $ Q2: Factor w/ 5 levels "strongly agree",..: 2 3 5 4 3 4 3 5
 $ Q3: Factor w/ 5 levels "strongly agree",..: 3 4 5 4 4 4 4 5
 $ Q4: Factor w/ 5 levels "strongly agree",..: 2 2 2 2 2 2 2 1
```

We can look at the levels of `Q1`, e.g.,

```
levels(issp.df$Q1)
[1] "strongly agree"        "agree"
[3] "neither agree nor dis" "disagree"
[5] "strongly disagree"
```

as.numeric()

Integer values assigned to levels of Q1: strongly agree = 1, agree = 2, ..., strongly disagree = 5. We defined this order using the levels argument in factor.

as.numeric() uses this factor property to convert Q1 to type numeric.

```
issp.df$Q1[1:10]
```

```
 [1] disagree                 strongly disagree
 [3] disagree                 <NA>
 [5] disagree                 disagree
 [7] strongly agree           neither agree nor dis
 [9] <NA>                     disagree
5 Levels: strongly agree ... strongly disagree
```

```
as.numeric(issp.df$Q1[1:10])
```

```
 [1]  4  5  4 NA  4  4  1  3 NA  4
```

as.numeric()

Items Q1 – Q4 are on the theme of *household gender roles*. Let's convert the values in each variable to a 5-point likert scale.

```
Q1.lik <- as.numeric(issp.df$Q1)
table(Q1.lik)

Q1.lik
  1   2   3   4   5
 98 297 331 263  21

Q2.lik <- as.numeric(issp.df$Q2)
Q3.lik <- as.numeric(issp.df$Q3)
Q4.lik <- as.numeric(issp.df$Q4)
```

Items on household gender roles: Total score

```
total.lik <- Q1.lik + Q2.lik + Q3.lik + Q4.lik
summary(total.lik)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   5.00   11.00   12.00   12.44   14.00   20.00
   NA's
     73
```

Constructing a data.frame

Let's create a new dataset from Q1.lik – Q4.lik and total.lik called likert.df.

```
likert.df <- data.frame(Q1.lik, Q2.lik, Q3.lik, Q4.lik,
                        total.lik)
head(likert.df)
```

```
  Q1.lik Q2.lik Q3.lik Q4.lik total.lik
1      4      2      3      2        11
2      5      3      4      2        14
3      4      5      5      2        16
4     NA      4      4      2        NA
5      4      3      4      2        13
6      4      4      4      2        14
```

Properties of `likert.df`

What are its column names?

```
names(likert.df)
```

```
[1] "Q1.lik"    "Q2.lik"    "Q3.lik"    "Q4.lik"
[5] "total.lik"
```

For data frames, `names()` yields *column* names. Let's now change them!

```
# Change only the last column's name to "Total"
names(likert.df)[5] <- "Total"
names(likert.df)
```

```
[1] "Q1.lik" "Q2.lik" "Q3.lik" "Q4.lik" "Total"
```

```
# Change all columns names
names(likert.df) <- c("Q1", "Q2", "Q3", "Q4", "Total")
```

```
[1] "Q1"    "Q2"    "Q3"    "Q4"    "Total"
```

Constructing a data.frame

Let's create the column names we want during our construction of the data frame, i.e.

```
likert.df <- data.frame(Q1=Q1.lik, Q2=Q2.lik,
                        Q3=Q3.lik, Q4=Q4.lik,
                        Total=total.lik)
head(likert.df)
```

```
  Q1 Q2 Q3 Q4 Total
1  4  2  3  2    11
2  5  3  4  2    14
3  4  5  5  2    16
4 NA  4  4  2    NA
5  4  3  4  2    13
6  4  4  4  2    14
```

Properties of `likert.df`

```
str(likert.df)

'data.frame': 1047 obs. of  5 variables:
 $ Q1    : num  4 5 4 NA 4 4 1 3 NA 4 ...
 $ Q2    : num  2 3 5 4 3 4 3 5 NA 5 ...
 $ Q3    : num  3 4 5 4 4 4 4 5 NA 5 ...
 $ Q4    : num  2 2 2 2 2 2 2 1 NA 2 ...
 $ Total: num  11 14 16 NA 13 14 10 14 NA 16 ...
```

The `str` function tells us that `likert.df`:

1. is a data frame,
2. comprises 1047 `obs.` (cases or rows) and 5 `variables` (columns), and
3. all 5 variables are `numeric`.

likert.df: Column summary statistics

I want to generate the summary statistics for all variables in `likert.df`.

```
summary(likert.df$Q1)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  1.000   2.000   3.000   2.814   4.000   5.000      37

summary(likert.df$Q2)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  1.000   3.000   4.000   3.531   4.000   5.000      27
```

.

.

.

for loop to get column summary statistics

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

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  1.000   2.000   3.000   2.814   4.000   5.000      37
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  1.000   3.000   4.000   3.531   4.000   5.000      27
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  1.000   3.000   4.000   3.716   4.000   5.000      26
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  1.000   2.000   2.000   2.307   3.000   5.000      19
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   5.00   11.00   12.00   12.44   14.00   20.00      73
```

Smart way: (apply) to get column summary statistics

```
apply(likert.df, 2, summary)
```

```
                Q1      Q2      Q3      Q4 Total
Min.         1.000   1.000   1.000   1.000  5.00
1st Qu.      2.000   3.000   3.000   2.000 11.00
Median       3.000   4.000   4.000   2.000 12.00
Mean         2.814   3.531   3.716   2.307 12.44
3rd Qu.      4.000   4.000   4.000   3.000 14.00
Max.         5.000   5.000   5.000   5.000 20.00
NA's        37.000  27.000  26.000  19.000 73.00
```

apply

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

- `X`: A data frame, e.g. `issp.df`.
- `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`).

apply()

Compute the mean and standard deviation of each column in `likert.df`, ignoring `NA`s.

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

```
       Q1         Q2         Q3         Q4      Total
 2.813861   3.531373   3.715965   2.307393 12.435318
```

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

```
       Q1         Q2         Q3         Q4      Total
0.9960307 1.1398725 1.0335688 0.8596006 2.3022901
```

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 <- issp.df$Age[1:10]
test1
```

```
 [1] 56 45 38 33 37 27 43 24 NA 22
```

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

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

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

```
[1] 1
```

apply() using a self-defined R function

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

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

```
   Q1    Q2    Q3    Q4 Total
   37    27    26    19    73
```

Attention programmers!

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

```
   Q1    Q2    Q3    Q4 Total
   37    27    26    19    73
```

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()

```
apply(likert.df, 2, mytab)
```

```
                   Q1       Q2       Q3       Q4  Total
Complete.obs 1010.00 1020.00 1021.00 1028.00 974.00
Missing.obs    37.00   27.00   26.00   19.00  73.00
Mean            2.81    3.53    3.72    2.31  12.44
Std.Error       0.03    0.04    0.03    0.03   0.07
Lower.CI        2.75    3.45    3.66    2.25  12.30
Upper.CI        2.87    3.61    3.78    2.37  12.58
```

More descriptive stats

Calculate the mean total score for male and female respondents.

```
issp.df$total.lik <- likert.df$Total
with(issp.df, mean(total.lik[Gender == "Male"],
                   na.rm = TRUE))

[1] 12.06436

with(issp.df, mean(total.lik[Gender == "Female"],
                   na.rm = TRUE))

[1] 12.71067
```

How about calculating the mean total score for each income group (10 levels)?

Smart way

```
issp.df$Income <- ifelse(issp.df$Income
                         == "NAV; NAP No own income",
                         NA, issp.df$Income)
with(issp.df, tapply(total.lik, Income, mean, na.rm = TRUE))

$10000 or less  $10001-$15000  $15001-$20000
      12.11261       12.19403       12.57333
 $20001-$25000  $25001-$30000  $30001-$40000
      12.36697       12.84071       12.79688
 $40001-$50000  $50001-$70000 $70001-$100000
      12.94444       12.52941       12.65000
```

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`

tapply()

```
with(issp.df, tapply(total.lik, Gender, mytab))

$Female
Complete.obs  Missing.obs         Mean    Std.Error
      553.00        54.00        12.71         0.10
    Lower.CI     Upper.CI
       12.51        12.91

$Male
Complete.obs  Missing.obs         Mean    Std.Error
      404.00        14.00        12.06         0.11
    Lower.CI     Upper.CI
       11.84        12.28
```

Summary

- Recoding
- Summary Stats
- `apply()`
- `tapply()`