7 Data Manipulation

07-18-20

There are six exercises below. You are required to provide five solutions, with the same options for choosing languages as with the last exercise. You can provide solutions in two languages for one exercise only (for example, Ex. 1,2,3,5 in R and Ex. 1 in SAS is acceptable, Ex. 1,2,3 in SAS and Ex. 1,2 in R is not).

If you choose SAS for an exercise, you may use `IML`, `DATA` operations or `PROC SQL` at your discretion.

Warning I will continue restricting the use of external libraries in R, particularly `tidyverse` libraries. You may choose to use `ggplot2`, but take care that the plots you produce are at least as readable as the equivalent plots in base R. You will be allowed to use whatever libraries tickle your fancy in the midterm and final projects.

Reuse

For some of these exercises, you may be able to reuse functions written in prior homework. Define those functions here.

```
required.replicates <- function(cv,dif,alpha = 0.05, beta = 0.20) {
  z_alpha <- qnorm(1-alpha/2)
  z_beta <- qnorm(1-beta)
  return(ceiling(2*((cv/dif)^2)*((z_alpha + z_beta)^2)))
}

cohen.d <- function(m_1,m_2,s.pooled) {
  return(abs(m_1-m_2)/s.pooled)
}
```

Exercise 1.

Background

I was interested in health of bee colonies in the United States, so I downloaded data from the USDA NASS site (https://quickstats.nass.usda.gov, listed under `SURVEY > ANIMALS & PRODUCTS > SPECIALTY > HONEY`)

Part a.

Download the file `colonies.csv` if you choose R, or `coloniesSAS.csv` for SAS. This file has been edited to be in the wide format. The first column identifies the state and the next 20 columns are `HONEY, BEE COLONIES - INVENTORY, MEASURED IN COLONIES` for the years

1995-2014. Read the data into a data frame or data table, and subset the data to include only the Central Plains states,

`'NEBRASKA','KANSAS','SOUTH DAKOTA','MINNESOTA','IOWA','MISSOURI','OKLAHOMA'.`

Do not print this table

```
path = "colonies.csv"
colonies.dat <- read.csv(path, header = TRUE)
colonies.dat <- colonies.dat[colonies.dat$State %in%
c('NEBRASKA','KANSAS','SOUTH
DAKOTA','MINNESOTA','IOWA','MISSOURI','OKLAHOMA'),]
```

Part b.

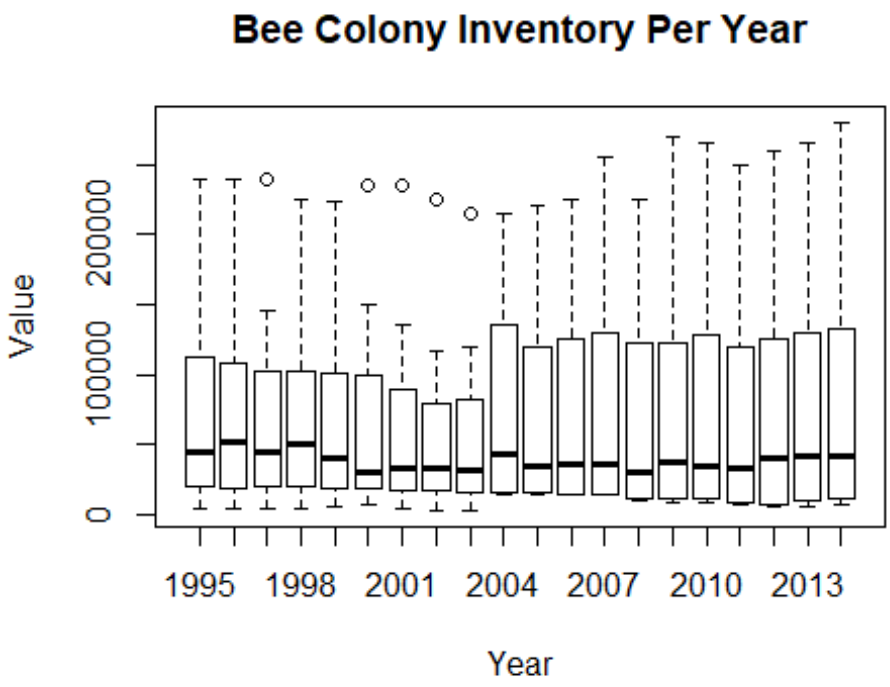
Reshape the data into the long format. There should be only 3 columns in the long data set, one column identifying `State, one column identifying `Year` and one column with `Value` of colony inventory. **Do not print this table**

```
names(colonies.dat)[2:21] <- paste(1995:2014)
v.list <- list(names(colonies.dat[2:21]))
colonies_long.dat <- reshape(colonies.dat,
                     direction = 'long',
                     varying = v.list,
                     timevar = 'Year',
                     ids = colonies.dat$State,
                     drop = 'State',
                     times = 1995:2014)
names(colonies_long.dat)[2:3] <- paste(c('Value','State'))
row.names(colonies_long.dat) <- 1:dim(colonies_long.dat)[1]
colonies_long.dat <- na.omit(colonies_long.dat)
```

Part c.

Plot `Value` by `Year`, with `Year` as the independent variable. We will want to see a boxplot, so you may need to specify `Year` to be a factor (or class). The actual Year values may not be correct after the reshape; you are not required to edit the values, but you may if you choose.

```
boxplot(Value~Year,data=colonies_long.dat,xlab = 'Year',ylab = 'Value',main =
'Bee Colony Inventory Per Year')
```



Exercise 2.

Background

The data for this exercise comes from the same source as Exercise 1, but instead the values are from `HONEY - PRODUCTION, MEASURED IN LB / COLONY`. However, the data in this exercise are in the long format.

Part a.

Download the file `production.csv` if you choose R, or `productionSAS.csv` for SAS. The first column identifies the `State`, the second column `Year` and the third column is the `Value` for `HONEY - PRODUCTION, MEASURED IN LB / COLONY`. Read the data into a data frame or data table, and subset the data to include only the Central Plains states,

`'NEBRASKA','KANSAS','SOUTH DAKOTA','MINNESOTA','IOWA','MISSOURI','OKLAHOMA'`.

Do not print this table

```
path = "production.csv"
production.dat <- read.csv(path, header = TRUE)
production.dat <- production.dat[production.dat$State %in%
c('NEBRASKA','KANSAS','SOUTH
DAKOTA','MINNESOTA','IOWA','MISSOURI','OKLAHOMA'),]
```

Part b.

Reshape or transpose this data from the long form to the wide form. This table should have 7 rows, one for each state in the selection.

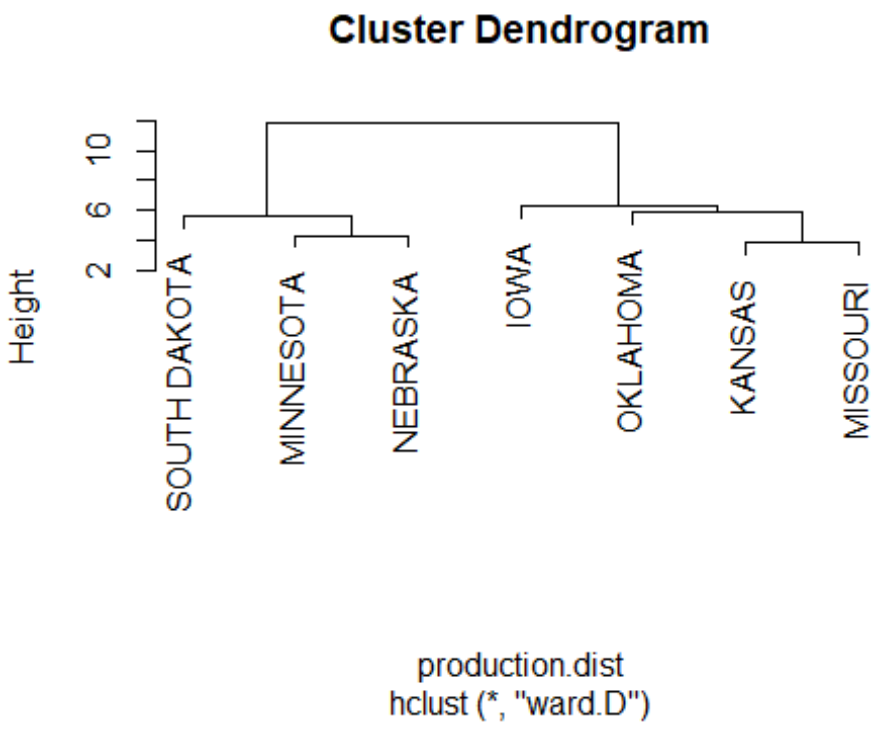
```
ProductionWide <- reshape(production.dat,idvar='State',timevar =
'Year',direction='wide')
names(ProductionWide)[2:21] <- paste(1995:2014)
row.names(ProductionWide) <- 1:dim(ProductionWide)[1]
```

Part c.

Name the reshaped table `ProductionWide`. The first column should be the name of the `State`. If you've reshaped correctly, the code below will produce a cluster plot with 7 leaves; edit `eval=FALSE` to `eval=TRUE` to include the plot in your output.

If you choose SAS, I've included similar code to call PROC CLUSTER in the template.

```
row.names(ProductionWide) <- ProductionWide[,1]
production.dist <- dist(scale(ProductionWide[,-1]), method="euclidean")
production.clust <- hclust(production.dist,method="ward.D")
plot(production.clust)
```



Exercise 3.

Part a.

Repeat the table from Homework 5, Exercise 2. The table will contain 30 rows, each corresponding to a unique combination of CV from 8,12,...,28 and Diff from 5,10,...,25. Add to the table a column `D` by calculating Cohen's d for each row of the table. Also calculate for each row a required replicates using the z-score formula and name this `RR`.

Define the table in the space below. **Do not print this table**.

```
Ex3a.dat <- data.frame(
CV = matrix(seq(8,28,by=4),nrow=30),
Diff = matrix(seq(5,25,by=5),nrow=30)
)
Ex3a.dat$D <- abs(Ex3a.dat$Diff)/Ex3a.dat$CV
Ex3a.dat$RR <- required.replicates(Ex3a.dat$CV,Ex3a.dat$Diff)
```

Part b.

Create two subset tables, one that contains the combinations of CV and Diff that require the five largest number of replicates and one the contains the combinations of CV and Diff the five smallest number of replicates. You can determine the subset by ranking or sorting by required replicates. You can add a rank column to your table if you wish. Call one table `LargestFive` and one table `SmallestFive`.

```
Ex3a.dat$Rank <- rank(Ex3a.dat$RR)
SmallestFive <- Ex3a.dat[Ex3a.dat$Rank <= 5,]
LargestFive <- Ex3a.dat[Ex3a.dat$Rank >= 26,]
```

Part c.

Print `LargestFive` sorted by required replicates in descending order, and print `SmallestFive` in ascending order.

```
print(LargestFive[rev(order(LargestFive$Rank)),])
```

```
##    CV Diff         D  RR Rank
## 6  28    5 0.1785714 493   30
## 11 24    5 0.2083333 362   29
## 16 20    5 0.2500000 252   28
## 21 16    5 0.3125000 161   27
## 12 28   10 0.3571429 124   26
```

```
print(SmallestFive[order(SmallestFive$Rank),])
```

```
##    CV Diff        D RR Rank
## 25  8   25 3.125000  2    1
## 19  8   20 2.500000  3    2
## 20 12   25 2.083333  4    3
```

```
## 13  8   15 1.875000  5    4
## 14 12   20 1.666667  6    5
```

Exercise 4

Create an ordered treatment pairs table from the Lacanne data. In the submitted work print the table only once at the end of the exercise.

Part a.

Read the lacanne data and compute mean m_i, standard deviation s_i and count n_i for `POM` in each level i of `Composite` for $i = 1, \ldots, k$

```
path = "lacanne2018.csv"
lacenne.dat <- read.csv(path, header = TRUE)
lac_summary.dat <- data.frame(
Composite = sort(unique(lacenne.dat$Composite)),
n_i = aggregate(POM ~ Composite,data = lacenne.dat,FUN = length)$POM,
m_i = aggregate(POM ~ Composite,data = lacenne.dat,FUN = mean)$POM,
s_i = aggregate(POM ~ Composite,data = lacenne.dat,FUN = sd)$POM
)
```

Part b

Create a table over all possible pairs i, j of k `Composite` means from these data.

Let one table column be i and another column be j. Let $i = 1,2, \ldots (k-1)$ and $j = i+1, i+2, \ldots k$. There will be $(k \times (k-1))/2$ rows in this table.

I usually create an empty table, then fill the table using a pair of nested loops, the outer loop over i and the inner loop over j. Use a counter variable to keep track of the current row and increment the counter in each step of the inner loop.

```
i_seq <- 1:4
j_seq <- 2:5
i_j_len <- 4
k <- 1

lac_mean.dat <- data.frame(
  m_1 = vector(length=10),
  m_2 = vector(length=10),
  row.names = 1:10
)

for(i in i_seq){ # Begin fist loop
  j_seq <- (i+1):5
    for(j in j_seq){ # Begin second loop
      lac_mean.dat$m_1[k] <- lac_summary.dat$m_i[i]
      lac_mean.dat$m_2[k] <- lac_summary.dat$m_i[j]
```

```
      k <- k + 1
    } # End second loop
} # End first loop
```

Part c.

Calculate Cohen's d for each combination of $\{m_i, m_j\}$ in this table. Note that there may be missing `sd` estimates. You have two options.

Option 1.

Calculate a single pooled standard deviation for all treatment mean pairs, using

$$s_{pooled} = \sqrt{\frac{\sum_i (n_i - 1)s_i^2}{N - k}}$$

where $N = \sum_i^k n_i$. You will need to remove missing `sd` estimates from the calculations, and any `n` not greater than 1 (When $n_i = 1$, s_i cannot be calculated).

```
s.pooled <- function(n_i,s_i){ # begin function
N <- 0

for(i in 1:length(n_i)){ # begin for loop
  if(n_i[i]>1){ # begin if statement
    N <- N + n_i[i]
  } # end if statement
  else next
} # end for loop

x <- subset(s_i,!is.na(s_i))
k <- length(x)

temp.sd <- 0
for(i in 1:length(n_i)){ # begin for loop
  if(!is.na(s_i[i])){ # begin if statement
    temp.sd <- temp.sd + (n_i[i] - 1)*(s_i[i]^2)
  } # end if statement
} # end for loop
  return(sqrt(temp.sd/(N-k)))
} # end function
```

Option 2.

Subset your table to exclude any rows with treatments corresponding to `sd == NA` and calculate a pooled standard deviation for each pair, using

$$s_{pooled} = \sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}}$$

Add d to the table, sort the table by d in descending order, and print the table.

```
s.p_1 <- s.pooled(lac_summary.dat$n_i,lac_summary.dat$s_i)

for(i in 1:dim(lac_mean.dat)[1]){
  lac_mean.dat$D[i] <- cohen.d(lac_mean.dat$m_1[i],lac_mean.dat$m_2[i],s.p_1)
}
lac_mean.dat[rev(order(lac_mean.dat$D)),]
```

```
##         m_1      m_2         D
## 10 6.903333 4.190000 1.8330912
## 9  6.295000 4.190000 1.4221094
## 7  6.010000 4.190000 1.2295673
## 4  5.720000 4.190000 1.0336472
## 3  5.720000 6.903333 0.7994439
## 6  6.010000 6.903333 0.6035239
## 8  6.295000 6.903333 0.4109817
## 2  5.720000 6.295000 0.3884622
## 1  5.720000 6.010000 0.1959201
## 5  6.010000 6.295000 0.1925421
```

Exercise 5.

Kruskal and Wallis describe a one-way analysis of variance method based on ranks (https://www.jstor.org/stable/2280779) We will use this method to analyze the Lacanne data.

Part a.

Determine the rank $r_j = \text{rank}(y_j)$ for the $j = 1, \ldots, N$ values in `y = POM`, independent of group, with the smallest value is given the smallest rank (1).

Part b.

Calculate

$$H = \frac{12}{N(N+1)} \sum_{i=1}^{C} \frac{R_i^2}{n_i} - 3(N+1)$$

where (quoting from Kruskal and Wallis)

C = the number of samples,
n_i = the number of observations in the ith sample,
N = $\sum n_i$ the number of observations in all samples combined,
R_i = the sum of ranks in the ith sample,

For the Lacanne data, the ith sample will be the ith `Composite`, so C will be the number of unique levels of `Composite` and R_1 will be the sum for ranks for the first level of `Composite`, etc.

Part c.

H can be approximated as χ^2 with $C - 1$ degrees of freedom. Use `pchisq` to calculate an upper-tail probability. How does this compare with the p value calculated in Homework 5?

You can compare your results with

```
kruskal.test(POM ~ Composite, lacanne.dat)
```

Exercise 6.

Part a.

Download the two files from D2L `ncaa2018.csv` and `ncaa2019.csv` (`ncaa2018SAS.csv` and `ncaa2019SAS.csv` for SAS), and read into data frames or tables. `ncaa2018.csv` comes from the same source as `elo.csv` from Homework 5, while `ncaa2019.csv` is the corresponding more recent data. These tables do not contain identical sets of columns, but we will be able to merge `Finish` by individual wrestlers.

```
path = 'ncaa2018.csv'
ncaa2018.dat <- read.csv(path,header = TRUE)
path = 'ncaa2019.csv'
ncaa2019.dat <- read.csv(path,header = TRUE)
```

Part b.

The tables list the wrestlers qualifying for the NCAA 2018 and 2019 National Championships, respectively. Merge the tables into a single table that contains only those wrestlers who qualified for both tournaments. Use the columns `Last` and `First` to merge on; neither is unique for all wrestlers.

Along with `Last` and `First`, the merged table should have columns corresponding to `Finish` 2018, `Finish` 2019, `Weight` 2018 and `Weight` 2019. You can leave the column names as the defaults produced by R or SAS. To check the merge, print the number of rows in the table, and determine if there are any missing values in either `Finish` column (`sum` or `any` are sufficient. *Do not print the table*.

```
merge.dat <- merge(ncaa2018.dat,ncaa2019.dat,by = c('First','Last'))

ncaa.merge.dat <- data.frame(Last = merge.dat$Last,
                             First = merge.dat$First,
                             Finish_2018 = merge.dat$Finish.x,
                             Finish_2019 = merge.dat$Finish.y,
                             Weight_2018 = merge.dat$Weight.x,
                             Weight_2019 = merge.dat$Weight.y
```

```
)

F_2018_check <- sum(!is.na(ncaa.merge.dat$Finish_2018))
cat('The number of non null values in 2018 Finishes is',F_2018_check,'\n')

## The number of non null values in 2018 Finishes is 198

F_2019_check <- sum(!is.na(ncaa.merge.dat$Finish_2019))
cat('The number of non null values in 2019 Finishes is',F_2018_check,'\n')

## The number of non null values in 2019 Finishes is 198

cat('The total rows in merged NCAA data table is',dim(ncaa.merge.dat)[1])

## The total rows in merged NCAA data table is 198
```

Part c.

Print a contingency table comparing `Weight` for 2018 and `Weight` for 2019. The sum of all cells in this table will be equal to the total number of wrestlers that competed in both tournaments; the sum of the main diagonal will be the number of wrestlers that competed in the same weight class for both. How many wrestlers changed weight classes?

```
weight_comp <- with(ncaa.merge.dat, table(Weight_2018, Weight_2019))
print(weight_comp)

##             Weight_2019
## Weight_2018 125 133 141 149 157 165 174 184 197 285
##         125  20   2   0   0   0   0   0   0   0   0
##         133   2  17   6   0   0   0   0   0   0   0
##         141   0   2  14   5   0   0   0   0   0   0
##         149   0   0   2  12   4   0   0   0   0   0
##         157   0   0   0   3  12   2   0   0   0   0
##         165   0   0   0   0   1  17   3   0   0   0
##         174   0   0   0   0   0   1  18   1   0   0
##         184   0   0   0   0   0   0   0  14   4   0
##         197   0   0   0   0   0   0   0   4  15   0
##         285   0   0   0   0   0   0   0   0   0  17

cat('\n','The number of wrestlers that changed weight classes
is',sum(weight_comp)-sum(diag(weight_comp)))

##
##  The number of wrestlers that changed weight classes is 42
```