Homework 7 - Data Manipulation

Amin Baabol

07/25/2020

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.

# 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**

colonies <- read.csv("~/Desktop/work/Summer2020/STATS600/week7/colonies.csv", header = TRUE)  
colonies.dat <- data.frame(colonies)  
plains.states.dat <- colonies.dat[colonies.dat$State %in% c("NEBRASKA","KANSAS","SOUTH DAKOTA","MINNESOTA","IOWA","MISSOURI","OKLAHOMA"), ]  
plains.states.dat

## State X1995 X1996 X1997 X1998 X1999 X2000 X2001 X2002  
## 14 IOWA 45000 52000 45000 50000 40000 30000 33000 33000  
## 15 KANSAS 17000 16000 17000 16000 13000 15000 13000 17000  
## 22 MINNESOTA 165000 150000 145000 140000 145000 150000 135000 117000  
## 24 MISSOURI 23000 22000 24000 23000 24000 23000 22000 18000  
## 26 NEBRASKA 60000 65000 61000 64000 58000 50000 43000 43000  
## 35 OKLAHOMA 4000 4000 4000 4000 6000 7000 4000 3000  
## 40 SOUTH DAKOTA 240000 240000 240000 225000 224000 235000 235000 225000  
## X2003 X2004 X2005 X2006 X2007 X2008 X2009 X2010 X2011 X2012  
## 14 32000 35000 28000 26000 26000 24000 26000 27000 25000 37000  
## 15 16000 14000 16000 14000 14000 10000 9000 9000 7000 6000  
## 22 120000 135000 120000 125000 130000 122000 122000 128000 120000 125000  
## 24 17000 16000 15000 15000 14000 11000 11000 11000 8000 7000  
## 26 45000 51000 40000 47000 45000 36000 48000 41000 41000 43000  
## 35 3000 NA NA NA NA NA NA NA NA NA  
## 40 215000 215000 220000 225000 255000 225000 270000 265000 250000 260000  
## X2013 X2014  
## 14 39000 35000  
## 15 6000 7000  
## 22 130000 132000  
## 24 10000 12000  
## 26 46000 50000  
## 35 NA NA  
## 40 265000 280000

## Part b.

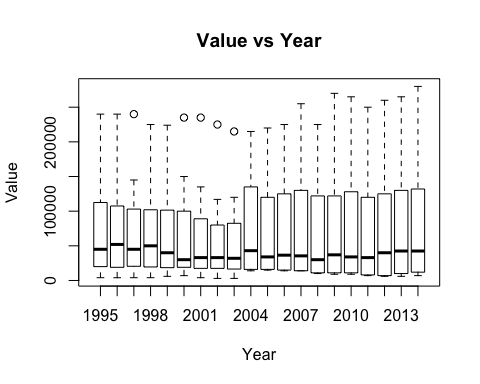
Reshape the data into the long format. There should be only 3 columns in the long data set, one column identifying `State, one column identifying Year and one column with Value of colony inventory. **Do not print this table**

reshaped.table <- reshape(plains.states.dat,varying = 2:21,times = 1995:2014,direction = "long",v.names = "Value",idvar="State",timevar = "Year")

## 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(reshaped.table$Value ~ factor(reshaped.table$Year), data = reshaped.table, main = "Value vs Year", xlab = "Year", ylab = "Value")



# 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'.

table.production <- read.csv("~/Desktop/work/Summer2020/STATS600/week7/production.csv", header = TRUE)  
table.production <- data.frame(table.production)

**Do not print this table**

## 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.

reshaped.table2 <- table.production[table.production$State %in% c("NEBRASKA","KANSAS","SOUTH DAKOTA","MINNESOTA","IOWA","MISSOURI","OKLAHOMA"), ]  
ProductionWide <- reshape(reshaped.table2,   
 direction = "wide",  
 idvar="State",  
 timevar = "Year")  
ProductionWide

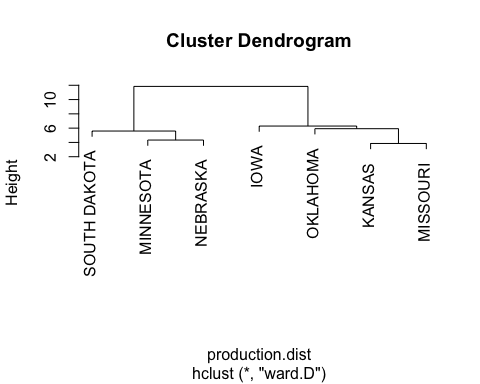
## State Value.1995 Value.1996 Value.1997 Value.1998 Value.1999  
## 12 IOWA 68 67 74 78 65  
## 13 KANSAS 67 51 71 46 67  
## 19 MINNESOTA 82 77 73 79 82  
## 21 MISSOURI 67 74 77 66 65  
## 23 NEBRASKA 73 75 67 70 77  
## 31 OKLAHOMA 76 59 58 51 45  
## 36 SOUTH DAKOTA 85 97 65 95 104  
## Value.2000 Value.2001 Value.2002 Value.2003 Value.2004 Value.2005  
## 12 67 51 70 59 67 88  
## 13 68 51 52 57 80 50  
## 19 90 81 73 83 75 74  
## 21 75 61 50 53 41 50  
## 23 87 48 75 74 89 68  
## 31 35 47 53 47 NA NA  
## 36 121 65 51 70 105 79  
## Value.2006 Value.2007 Value.2008 Value.2009 Value.2010 Value.2011  
## 12 84 81 62 42 49 62  
## 13 55 40 61 63 52 44  
## 19 80 68 78 65 66 53  
## 21 46 46 53 47 52 43  
## 23 73 49 67 56 55 59  
## 31 NA NA NA NA NA NA  
## 36 47 52 95 66 58 66  
## Value.2012 Value.2013 Value.2014  
## 12 61 48 43  
## 13 55 46 75  
## 19 67 58 60  
## 21 53 47 47  
## 23 65 60 75  
## 31 NA NA NA  
## 36 63 56 87

## 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 and Diff from . Add to the table a column D by calculating Cohen’s for each row of the table. Also calculate for each row a required replicates using the -score formula and name this RR.

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

table5 <- data.frame(CV = rep(seq(8,28,4), each = 5), Diff = rep(seq(5,25,5),6))  
table5

## CV Diff  
## 1 8 5  
## 2 8 10  
## 3 8 15  
## 4 8 20  
## 5 8 25  
## 6 12 5  
## 7 12 10  
## 8 12 15  
## 9 12 20  
## 10 12 25  
## 11 16 5  
## 12 16 10  
## 13 16 15  
## 14 16 20  
## 15 16 25  
## 16 20 5  
## 17 20 10  
## 18 20 15  
## 19 20 20  
## 20 20 25  
## 21 24 5  
## 22 24 10  
## 23 24 15  
## 24 24 20  
## 25 24 25  
## 26 28 5  
## 27 28 10  
## 28 28 15  
## 29 28 20  
## 30 28 25

### 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.

RR <- function (cv, percent.diff, alpha = 0.05, beta = 0.2){  
 cv <- cv  
 percent.diff <- percent.diff  
 n <- 2\*(((cv/percent.diff)^2)\*(qnorm((1-alpha/2)) + qnorm((1-beta)))^2)   
 n <- round(n,0)  
 z <- list(CV = cv, PercentDiff= percent.diff, RequiredReplicates = round(n,0))  
 return(z)  
}  
value <- RR(cv = table5$CV, percent.diff = table5$Diff)  
  
# Adding RequiredReplicates column  
table5$RequiredReplicates <- value$RequiredReplicates  
  
  
# Sort the table by decreasing Replicate values  
ordered\_table <- table5[order(table5$RequiredReplicates,decreasing = TRUE),]  
  
# Creating the required subset tables  
LargestFive <- head(ordered\_table, 5)  
  
# Now for smallest  
SmallestFive <- tail(ordered\_table, 5)  
  
LargestFive

## CV Diff RequiredReplicates  
## 26 28 5 492  
## 21 24 5 362  
## 16 20 5 251  
## 11 16 5 161  
## 27 28 10 123

SmallestFive

## CV Diff RequiredReplicates  
## 15 16 25 6  
## 3 8 15 4  
## 10 12 25 4  
## 4 8 20 3  
## 5 8 25 2

### Part c.

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

# adding rank for LargestFive table. Largest value gets rank 1  
LargestFive$Rank <- rank(order(LargestFive$RequiredReplicates, decreasing = TRUE))  
# printing larget five in descending order, Largest value gets the first rank  
LargestFive

## CV Diff RequiredReplicates Rank  
## 26 28 5 492 1  
## 21 24 5 362 2  
## 16 20 5 251 3  
## 11 16 5 161 4  
## 27 28 10 123 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 pumpkin data and compute mean , standard deviation and count for POM in each level of Composite for

lacanne.dat <- read.csv(file("~/Desktop/work/Summer2020/STATS600/week5/lacanne2018.csv"), stringsAsFactors=T)  
lacanne.dat

## Order Town State Lat Lon Organic Cover Insecticide  
## 1 1 Bladen NE 40.31971 -98.57358 N Y N  
## 2 2 Bladen NE 40.33703 -98.56301 N N Y  
## 3 3 York NE 40.63054 -97.66534 N Y N  
## 4 4 York NE 40.97390 -97.49031 N N Y  
## 5 5 Bismarck ND 46.85280 -100.60131 N Y N  
## 6 6 Bismarck ND 46.85280 -100.35145 N N Y  
## 7 7 Bismarck ND 46.81734 -100.51257 N Y N  
## 8 8 Bismarck ND 47.14250 -100.19720 N N Y  
## 9 9 White SD 44.42572 -96.58806 Y Y N  
## 10 10 White SD 44.41155 -96.60008 N N Y  
## 11 12 Pipestone MN 44.12416 96.36422 N N Y  
## 12 13 Toronto SD 44.59248 -96.57923 N Y Y  
## 13 16 Gary SD 44.80689 -96.35465 N N Y  
## 14 17 Arlington SD 44.41566 -97.18795 N Y N  
## 15 18 Arlington SD 44.42644 -97.25077 N N Y  
## 16 19 Lake Norden SD 44.58976 -97.08649 N Y Y  
## Pesticide Tillage Grazed Composite POM Study.Year  
## 1 Y N N 3 6.23 2015  
## 2 Y Y N 0 4.52 2015  
## 3 Y N N 3 6.21 2015  
## 4 Y Y N 0 5.55 2015  
## 5 N N Y 5 4.19 2015  
## 6 Y N N 1 8.17 2016  
## 7 Y N Y 4 5.82 2015  
## 8 Y N N 1 3.85 2015  
## 9 N Y N 3 8.18 2016  
## 10 Y Y N 0 5.52 2015  
## 11 Y Y N 0 4.75 2015  
## 12 Y N N 4 7.36 2016  
## 13 Y Y N 0 6.38 2015  
## 14 Y N Y 4 7.53 2016  
## 15 Y Y N 0 7.60 2015  
## 16 Y N Y 3 4.56 2016

#lacanne.table <- aggregate(lacanne.dat$POM, by = list(lacanne.dat$Composite),FUN = function(x) c(Mean = mean(x), SD = sd(x), Count.n = length(x)))  
  
mean.i <- setNames(aggregate(lacanne.dat$POM,   
 by = list(lacanne.dat$Composite), FUN = mean), c("Composite", "Mean"))  
sd.i <- setNames(aggregate(lacanne.dat$POM,   
 by = list(lacanne.dat$Composite), FUN = sd), c("Composite", "SD"))  
count.i <- setNames(aggregate(lacanne.dat$POM,   
 by = list(lacanne.dat$Composite), FUN = length), c("Composite", "Counts"))  
combined <- Reduce(function(x, y) merge(x, y, all=TRUE),   
 list(mean.i, sd.i, count.i))  
mean.i;sd.i;count.i;combined

## Composite Mean  
## 1 0 5.720000  
## 2 1 6.010000  
## 3 3 6.295000  
## 4 4 6.903333  
## 5 5 4.190000

## Composite SD  
## 1 0 1.1331196  
## 2 1 3.0547013  
## 3 3 1.4804166  
## 4 4 0.9420368  
## 5 5 NA

## Composite Counts  
## 1 0 6  
## 2 1 2  
## 3 3 4  
## 4 4 3  
## 5 5 1

## Composite Mean SD Counts  
## 1 0 5.720000 1.1331196 6  
## 2 1 6.010000 3.0547013 2  
## 3 3 6.295000 1.4804166 4  
## 4 4 6.903333 0.9420368 3  
## 5 5 4.190000 NA 1

### Part b

Create a table over all possible pairs of Composite means from these data.

Let one table column be and another column be . Let and . There will be rows in this table.

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

composite.levels <- levels(factor(lacanne.dat$Composite))  
  
#how the combination matrix should look like  
temp.table <- as.data.frame(t(combn(5,2)))  
  
  
#empty table to fill the required values in  
combined.table <- data.frame(col1 = numeric(10), col2 = numeric(10))  
  
for (i in 1:length(combined.table)){  
composite.levels[temp.table$V1[i]] <- combined.table$col1[i]  
composite.levels[temp.table$V2[i]] <- combined.table$col2[i]   
}  
  
# Finally,combining the possible pairs including sd and counts values we created above  
  
combined.table <- merge(combined.table, combined, by.x = "col1", by.y = "Composite" )  
combined.table <- merge(combined.table, combined, by.x = "col2", by.y = "Composite" )  
combined.table

## col2 col1 Mean.x SD.x Counts.x Mean.y SD.y Counts.y  
## 1 0 0 5.72 1.13312 6 5.72 1.13312 6  
## 2 0 0 5.72 1.13312 6 5.72 1.13312 6  
## 3 0 0 5.72 1.13312 6 5.72 1.13312 6  
## 4 0 0 5.72 1.13312 6 5.72 1.13312 6  
## 5 0 0 5.72 1.13312 6 5.72 1.13312 6  
## 6 0 0 5.72 1.13312 6 5.72 1.13312 6  
## 7 0 0 5.72 1.13312 6 5.72 1.13312 6  
## 8 0 0 5.72 1.13312 6 5.72 1.13312 6  
## 9 0 0 5.72 1.13312 6 5.72 1.13312 6  
## 10 0 0 5.72 1.13312 6 5.72 1.13312 6

### Part c.

Calculate Cohen’s for each combination of 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

where . You will need to remove missing sd estimates from the calculations, and any n not greater than 1 (When , cannot be calculated).

#### 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

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

cohen.d <- function(m1,s1,m2,s2){  
 cohens\_d <-(abs(m1-m2)/sqrt((s1^2+s2^2)/2))  
return(cohens\_d)  
}  
  
  
for(i in 1:nrow(combined.table)) {  
 combined.table$cohens\_d[i] <-  
 cohen.d(  
 m1 = combined.table$Mean.x[i],  
 s1 = combined.table$SD.x[i],  
 m2 = combined.table$Mean.y[i],  
 s2 = combined.table$SD.y[i]  
 )  
}  
  
combined.table <- combined.table[order(combined.table$cohens\_d, decreasing = TRUE),]  
# This is the final table we need  
combined.table

## col2 col1 Mean.x SD.x Counts.x Mean.y SD.y Counts.y cohens\_d  
## 1 0 0 5.72 1.13312 6 5.72 1.13312 6 0  
## 2 0 0 5.72 1.13312 6 5.72 1.13312 6 0  
## 3 0 0 5.72 1.13312 6 5.72 1.13312 6 0  
## 4 0 0 5.72 1.13312 6 5.72 1.13312 6 0  
## 5 0 0 5.72 1.13312 6 5.72 1.13312 6 0  
## 6 0 0 5.72 1.13312 6 5.72 1.13312 6 0  
## 7 0 0 5.72 1.13312 6 5.72 1.13312 6 0  
## 8 0 0 5.72 1.13312 6 5.72 1.13312 6 0  
## 9 0 0 5.72 1.13312 6 5.72 1.13312 6 0  
## 10 0 0 5.72 1.13312 6 5.72 1.13312 6 0

# 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 for the values in y = POM, independent of group, with the smallest value is given the smallest rank (1).

### Part b.

Calculate

where (quoting from Kruskal and Wallis)

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

### Part c.

can be approximated as with degrees of freedom. Use pchisq to calculate an upper-tail probability. How does this compare with the 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.

ncaa2018 =read.table("~/Desktop/work/Summer2020/STATS600/week7/ncaa2018.csv",header = T, sep = ",")  
head(ncaa2018)

## Weight Last First Conference Finish ELO  
## 1 125 Atkins Thayer ACC NQ 1297.06  
## 2 125 Bentley LJ ACC NQ 1343.66  
## 3 125 Fausz Sean ACC cons 24 1380.84  
## 4 125 Hayes Louie ACC cons 12 1404.51  
## 5 125 Norstrem Kyle ACC cons 24 1348.79  
## 6 125 Bianchi Paul Big 12 cons 32 1312.73

ncaa2019 =read.table("~/Desktop/work/Summer2020/STATS600/week7/ncaa2019.csv",header = T, sep = ",")  
head(ncaa2019)

## Weight Last First Finish  
## 1 125 Lee Spencer 1  
## 2 125 Mueller Jack 2  
## 3 125 Rivera Sebastian 3  
## 4 125 Arujau Vitali 4  
## 5 125 Piccininni Nicholas 5  
## 6 125 Glory Pat 6

ncaa.joined <- merge(ncaa2019, ncaa2018, by = c("Last", "First"), all = FALSE)  
ncaa.joined

## Last First Weight.x Finish.x Weight.y Conference  
## 1 Aiello Jay 197 cons 24 197 ACC  
## 2 Alber Josh 141 cons 24 141 Big 12  
## 3 Allen Alonzo 125 cons 32 125 SoCon  
## 4 Amine Malik 149 cons 32 149 Big Ten  
## 5 Amine Myles 174 3 174 Big Ten  
## 6 Anthony Tejon 149 cons 24 141 EWL  
## 7 Ashworth Branson 165 cons 24 165 Big 12  
## 8 Barone Eric 157 cons 16 149 Big Ten  
## 9 Bastian Kimball 174 cons 24 174 Big 12  
## 10 Berger Tyler 157 2 157 Big Ten  
## 11 Blees Ryan 149 cons 24 149 ACC  
## 12 Bleise Steve 157 cons 12 149 Big Ten  
## 13 Bresser Ronnie 125 8 125 Pac 12  
## 14 Bridges Montorie 133 cons 16 133 Big 12  
## 15 Brock Kaid 141 cons 12 133 Big 12  
## 16 Brucki Patrick 197 4 197 EIWA  
## 17 Brunner Christian 197 cons 12 197 Big Ten  
## 18 Bullard Daniel 174 cons 24 174 ACC  
## 19 Bulsak Greg 197 cons 32 184 EWL  
## 20 Butterbrodt Ian 285 cons 12 285 EIWA  
## 21 Campbell Te`shan 165 cons 12 165 Big Ten  
## 22 Carr Michael 141 cons 16 141 Big Ten  
## 23 Caywood Rocco 197 cons 32 197 EIWA  
## 24 Chalifoux Trey 125 cons 32 125 EIWA  
## 25 Christensen Ryan 174 cons 24 174 Big Ten  
## 26 Colbray Samuel 184 cons 12 197 Big 12  
## 27 Coleman Bob 184 cons 32 184 Pac 12  
## 28 Curry Gage 125 cons 32 125 EIWA  
## 29 De La Riva Lorenzo 174 cons 32 165 Pac 12  
## 30 Deakin Ryan 157 6 149 Big Ten  
## 31 Dean Maxwell 184 2 184 EIWA  
## 32 Debien Chris 141 cons 32 133 SoCon  
## 33 Degen Jarrett 149 7 149 Big 12  
## 34 Deprez Vincent 174 cons 32 165 EIWA  
## 35 DeSanto Austin 133 5 133 EIWA  
## 36 Dhesi Amar 285 5 285 Pac 12  
## 37 Diabe Randall 197 cons 32 197 SoCon  
## 38 Diakomihalis Yianni 141 1 141 EIWA  
## 39 Ducharme Dom 184 cons 32 184 Pac 12  
## 40 Duncan Dylan 133 cons 33 133 Big Ten  
## 41 Early Larry 157 8 157 MAC  
## 42 Eierman Jaydin 141 3 141 MAC  
## 43 Erneste John 133 6 133 MAC  
## 44 Fausz Sean 125 cons 12 125 ACC  
## 45 Fehlman DJ 133 cons 16 133 EWL  
## 46 Findlay Matt 141 cons 12 141 Big 12  
## 47 Finesilver Josh 141 cons 32 133 ACC  
## 48 Finesilver Matt 174 cons 24 174 ACC  
## 49 Finesilver Mitch 149 4 157 ACC  
## 50 Finesilver Zach 165 cons 32 165 ACC  
## 51 Flynn Connor 165 cons 24 165 MAC  
## 52 Fogarty Andrew 165 cons 16 165 Big 12  
## 53 Foley RayVon 125 7 125 Big Ten  
## 54 Foster Drew 184 1 184 Big 12  
## 55 Gil Nicholas 141 cons 24 141 EIWA  
## 56 Gilliland-Daniel Cory 285 cons 32 285 ACC  
## 57 Girard Willy 125 cons 33 125 EWL  
## 58 Gonser Noah 133 cons 24 133 MAC  
## 59 Gravina Nick 184 cons 32 184 Big Ten  
## 60 Grello Joe 174 cons 16 174 Big Ten  
## 61 Guillen Mario 133 cons 32 141 MAC  
## 62 Gunther Joseph 165 cons 16 174 Big Ten  
## 63 Haines Thomas 285 cons 24 285 EWL  
## 64 Hall Mark 174 2 174 Big Ten  
## 65 Harvey Ben 174 cons 12 174 EIWA  
## 66 Hayes Ke-Shawn 157 cons 24 149 Big Ten  
## 67 Hazel Corey 184 cons 16 184 EWL  
## 68 Headlee Austin 141 cons 24 141 ACC  
## 69 Hemida Youssif 285 6 285 Big Ten  
## 70 Hidlay Hayden 157 4 157 ACC  
## 71 Hildebrandt Drew 125 cons 24 125 MAC  
## 72 Jarrell Ebed 165 cons 32 165 EIWA  
## 73 Jeffries Davion 149 cons 24 149 Big 12  
## 74 Jennings Conan 285 cons 16 285 Big Ten  
## 75 Jensen David 285 cons 24 285 Big Ten  
## 76 Jordan Micah 149 2 157 Big Ten  
## 77 Joseph Vincenzo 165 2 165 Big Ten  
## 78 Kelly Cameron 141 cons 24 133 MAC  
## 79 Kiussis Nick 165 cons 24 165 Big 12  
## 80 Kober Chris 184 cons 33 197 SoCon  
## 81 Kolodzik Matthew 149 5 149 EIWA  
## 82 Krivus Sam 141 cons 32 149 ACC  
## 83 Kutler Jordan 174 7 174 EIWA  
## 84 LaFragola Christian 184 cons 32 184 EIWA  
## 85 Lane Thomas 197 cons 12 197 Pac 12  
## 86 Lantry Bryan 141 cons 32 133 MAC  
## 87 LaPrade BC 157 cons 24 157 ACC  
## 88 Lee Nick 141 5 141 Big Ten  
## 89 Lee Spencer 125 1 125 Big Ten  
## 90 Lewis Daniel 174 4 174 MAC  
## 91 Limmex Nate 141 cons 32 141 Big Ten  
## 92 Lizak Ethan 133 7 125 Big Ten  
## 93 Loiseau Stephen 197 cons 32 197 EIWA  
## 94 Lujan Taylor 174 cons 12 174 Big 12  
## 95 Lydy Dylan 174 cons 12 174 Big Ten  
## 96 Marinelli Alex 165 7 165 Big Ten  
## 97 Marinelli Tyler 165 cons 32 157 SoCon  
## 98 Marsteller Chance 165 3 165 EWL  
## 99 Martin Bryce 165 cons 32 165 Big Ten  
## 100 Martin Cole 149 cons 24 149 Big Ten  
## 101 Martin Myles 184 3 184 Big Ten  
## 102 Massa Logan 165 cons 12 165 Big Ten  
## 103 Mattin Drew 125 cons 16 125 Big Ten  
## 104 McFadden David 174 5 165 ACC  
## 105 McGee Michael 125 cons 12 125 MAC  
## 106 McKee Mitch 141 6 133 Big Ten  
## 107 McKenna Joey 141 2 141 Big Ten  
## 108 McLaughlin Anthony 197 cons 24 197 Big 12  
## 109 Meink Korbin 125 cons 24 125 SoCon  
## 110 Micic Stevan 133 3 133 Big Ten  
## 111 Miller Cary 285 cons 24 285 SoCon  
## 112 Moisey Zeke 125 cons 32 125 Big 12  
## 113 Monserrat Christian 149 cons 24 141 Big 12  
## 114 Montoya Rico 125 cons 32 133 Big 12  
## 115 Moody Christian 125 cons 32 125 Big 12  
## 116 Moore Kollin 197 2 197 Big Ten  
## 117 Morgan Andrew 174 cons 24 174 SoCon  
## 118 Mueller Jack 125 2 133 ACC  
## 119 Myers Korbin 133 cons 16 133 EWL  
## 120 Ness Chip 184 6 184 ACC  
## 121 Nevills AJ 285 cons 32 285 Big 12  
## 122 Ngati Brandon 285 cons 32 285 Big 12  
## 123 Nickal Bo 197 1 184 Big Ten  
## 124 Nickell Sean 133 cons 24 133 Pac 12  
## 125 Nolf Jason 157 1 157 Big Ten  
## 126 Olivas Khristian 149 cons 32 149 Big 12  
## 127 Oliver Elijah 125 cons 24 125 Big Ten  
## 128 Oliver Justin 149 cons 16 149 MAC  
## 129 Orndorff Tanner 197 cons 16 197 Big 12  
## 130 Oster Shayne 149 cons 32 157 Big Ten  
## 131 Pantaleo Alec 157 3 157 Big Ten  
## 132 Parker Ian 141 cons 24 141 Big 12  
## 133 Parker Kevin 184 cons 32 184 EIWA  
## 134 Parker Zack 285 cons 33 285 MAC  
## 135 Parks Logan 157 cons 24 165 MAC  
## 136 Pelusi Antonio 285 cons 32 285 EIWA  
## 137 Pengilly Mason 133 cons 12 133 Pac 12  
## 138 Perry Brett 197 cons 24 184 MAC  
## 139 Perry Sa`Derian 141 cons 24 141 MAC  
## 140 Phipps Drew 197 cons 24 184 EIWA  
## 141 Piccininni Nicholas 125 5 125 Big 12  
## 142 Piotrowski Travis 125 cons 24 125 Big Ten  
## 143 Pletcher Luke 133 4 133 Big Ten  
## 144 Pohlmeyer Henry 149 cons 32 141 Big 12  
## 145 Preisch Ryan 184 4 184 EIWA  
## 146 Prince Jared 149 cons 12 149 EIWA  
## 147 Rahmani Taleb 157 cons 12 157 ACC  
## 148 Rasheed Shakur 184 cons 16 197 Big Ten  
## 149 Red Chad 141 8 141 Big Ten  
## 150 Rivera Sebastian 125 3 125 Big Ten  
## 151 Rohlfing Russell 149 cons 16 141 Pac 12  
## 152 Romero Demetrius 165 cons 12 165 Big 12  
## 153 Rooney Tim 133 cons 32 141 MAC  
## 154 Root Sawyer 197 cons 32 197 SoCon  
## 155 Russell Sean 125 cons 12 125 EWL  
## 156 Schany Will 184 cons 24 174 ACC  
## 157 Schmitt Matthew 133 cons 24 133 Big 12  
## 158 Schultz Eric 197 cons 16 197 Big Ten  
## 159 Schuyler Cortlandt 149 cons 32 149 EIWA  
## 160 Seely Jacob 197 cons 32 197 Big 12  
## 161 Sherry Dean 174 cons 32 174 EWL  
## 162 Shields Joshua 165 6 157 Pac 12  
## 163 Shoop Kyle 141 7 141 EWL  
## 164 Skatzka Devin 174 8 174 Big Ten  
## 165 Smith Jacobe 174 cons 16 174 Big 12  
## 166 Sprague Michael 149 cons 16 149 EIWA  
## 167 Stencel Matt 285 8 285 MAC  
## 168 Stewart Noah 184 cons 24 184 EIWA  
## 169 Straw Chase 157 cons 24 157 Big 12  
## 170 Suriano Nick 133 1 125 Big Ten  
## 171 Sweany Jeramy 285 cons 32 285 EIWA  
## 172 Sykora Cam 133 cons 24 133 Big 12  
## 173 Terao Josh 133 cons 24 133 EIWA  
## 174 Thomas Justin 157 cons 12 157 Big 12  
## 175 Thomsen Max 149 cons 12 149 Big 12  
## 176 Thorn Thomas 149 cons 12 141 Big Ten  
## 177 Thornton Ben 133 cons 16 133 Big Ten  
## 178 Townsell Gabe 125 cons 24 125 Pac 12  
## 179 Traxler Nathan 197 cons 12 197 Pac 12  
## 180 Tucker Charles 133 cons 24 133 EIWA  
## 181 Turner Sam 141 cons 16 149 Big 12  
## 182 Valencia Zahid 174 1 174 Pac 12  
## 183 Van Brill John 157 cons 16 157 Big Ten  
## 184 Venz Taylor 184 cons 12 184 Big Ten  
## 185 Viruet Jonathan 165 cons 32 165 EIWA  
## 186 Voss Matt 285 cons 16 285 EWL  
## 187 Weigel Preston 197 3 197 Big 12  
## 188 Weiland Luke 157 cons 24 157 EIWA  
## 189 White Derek 285 2 285 Big 12  
## 190 White Isaiah 165 5 165 Big Ten  
## 191 Wick Evan 165 4 165 Big Ten  
## 192 Wilcke Cash 184 cons 16 197 Big Ten  
## 193 Willits Hunter 157 cons 32 157 Pac 12  
## 194 Wilson Tariq 133 cons 12 133 ACC  
## 195 Wolf Gordon 165 cons 12 165 EIWA  
## 196 Womack Brandon 174 cons 16 174 EIWA  
## 197 Wood Jordan 285 4 285 EIWA  
## 198 Zavatsky Zachary 184 8 184 ACC  
## Finish.y ELO  
## 1 NQ 1299.28  
## 2 cons 32 1380.17  
## 3 cons 32 1314.60  
## 4 cons 24 1323.92  
## 5 3 1467.78  
## 6 cons 24 1322.67  
## 7 cons 16 1401.23  
## 8 NQ NA  
## 9 cons 24 1337.72  
## 10 3 1431.11  
## 11 cons 24 1340.88  
## 12 cons 24 1372.79  
## 13 7 1421.76  
## 14 8 1374.80  
## 15 5 1458.41  
## 16 cons 24 1401.05  
## 17 cons 24 1346.76  
## 18 cons 24 1395.03  
## 19 cons 32 1352.57  
## 20 NQ NA  
## 21 cons 16 1391.54  
## 22 cons 12 1401.78  
## 23 cons 24 1324.03  
## 24 NQ NA  
## 25 cons 24 1344.23  
## 26 NQ NA  
## 27 NQ NA  
## 28 cons 32 1309.86  
## 29 NQ NA  
## 30 cons 16 1462.21  
## 31 8 1399.22  
## 32 NQ NA  
## 33 cons 12 1365.25  
## 34 NQ 1351.68  
## 35 cons 12 1454.45  
## 36 3 1420.42  
## 37 cons 32 1289.91  
## 38 1 1480.20  
## 39 NQ NA  
## 40 cons 32 1349.45  
## 41 cons 24 1348.99  
## 42 4 1495.22  
## 43 cons 12 1390.86  
## 44 cons 24 1380.84  
## 45 NQ NA  
## 46 NQ NA  
## 47 cons 32 1311.38  
## 48 cons 24 1376.51  
## 49 cons 16 1403.95  
## 50 cons 32 1351.19  
## 51 cons 24 1365.55  
## 52 cons 24 1359.57  
## 53 cons 33 1373.86  
## 54 cons 24 1378.35  
## 55 cons 16 1371.56  
## 56 cons 32 1227.70  
## 57 NQ NA  
## 58 NQ NA  
## 59 cons 16 1375.74  
## 60 NQ 1355.77  
## 61 NQ NA  
## 62 cons 32 1327.22  
## 63 cons 16 1388.44  
## 64 2 1546.21  
## 65 cons 12 1350.03  
## 66 cons 12 1420.70  
## 67 cons 24 1315.59  
## 68 cons 32 1340.85  
## 69 8 1366.21  
## 70 2 1483.14  
## 71 NQ 1333.27  
## 72 NQ 1349.30  
## 73 cons 32 1347.67  
## 74 cons 32 1335.59  
## 75 NQ NA  
## 76 6 1470.37  
## 77 1 1485.34  
## 78 cons 33 1377.86  
## 79 NQ NA  
## 80 NQ NA  
## 81 3 1463.11  
## 82 cons 16 1349.97  
## 83 6 1423.24  
## 84 cons 16 1315.78  
## 85 cons 24 1282.98  
## 86 cons 24 1370.78  
## 87 NQ NA  
## 88 5 1460.41  
## 89 1 1539.30  
## 90 4 1487.71  
## 91 cons 24 1333.40  
## 92 4 1451.39  
## 93 cons 24 1363.14  
## 94 cons 12 1413.14  
## 95 cons 12 1372.59  
## 96 6 1421.35  
## 97 cons 33 1349.61  
## 98 4 1433.82  
## 99 cons 32 1317.56  
## 100 cons 32 1342.50  
## 101 2 1469.69  
## 102 cons 24 1469.33  
## 103 cons 16 1415.50  
## 104 5 1467.48  
## 105 cons 24 1387.47  
## 106 cons 12 1407.65  
## 107 3 1475.00  
## 108 NQ NA  
## 109 NQ 1336.37  
## 110 2 1475.74  
## 111 NQ 1301.86  
## 112 8 1378.20  
## 113 NQ NA  
## 114 cons 16 1337.08  
## 115 cons 32 1347.97  
## 116 4 1420.78  
## 117 NQ 1338.58  
## 118 cons 12 1442.25  
## 119 cons 24 1372.13  
## 120 7 1369.49  
## 121 cons 32 1356.21  
## 122 NQ NA  
## 123 1 1534.24  
## 124 cons 16 1339.70  
## 125 1 1549.47  
## 126 cons 24 1323.32  
## 127 cons 24 1334.17  
## 128 cons 12 1414.52  
## 129 cons 32 1290.29  
## 130 NQ NA  
## 131 5 1436.88  
## 132 NQ NA  
## 133 NQ NA  
## 134 NQ NA  
## 135 NQ 1314.75  
## 136 cons 24 1292.01  
## 137 cons 32 1317.20  
## 138 NQ NA  
## 139 8 1329.69  
## 140 NQ 1315.69  
## 141 cons 12 1416.54  
## 142 cons 16 1363.70  
## 143 4 1418.38  
## 144 cons 24 1332.27  
## 145 cons 12 1420.43  
## 146 cons 32 1406.92  
## 147 cons 24 1382.65  
## 148 7 1424.86  
## 149 7 1385.89  
## 150 6 1418.76  
## 151 cons 32 1321.94  
## 152 cons 24 1309.42  
## 153 NQ NA  
## 154 cons 32 1272.51  
## 155 cons 16 1405.63  
## 156 cons 33 1307.74  
## 157 cons 16 1379.68  
## 158 cons 16 1328.63  
## 159 cons 16 1376.71  
## 160 cons 32 1287.58  
## 161 cons 32 1338.26  
## 162 7 1440.47  
## 163 cons 32 1358.13  
## 164 cons 32 1368.90  
## 165 8 1412.81  
## 166 cons 32 1331.59  
## 167 cons 32 1324.63  
## 168 NQ 1333.81  
## 169 NQ NA  
## 170 2 1486.05  
## 171 cons 24 1310.36  
## 172 cons 32 1357.53  
## 173 cons 32 1392.45  
## 174 NQ NA  
## 175 cons 12 1414.76  
## 176 cons 32 1372.99  
## 177 cons 32 1344.25  
## 178 cons 24 1347.01  
## 179 cons 16 1353.18  
## 180 cons 24 1371.36  
## 181 cons 32 1327.54  
## 182 1 1520.85  
## 183 cons 12 1392.23  
## 184 4 1375.92  
## 185 cons 24 1346.24  
## 186 cons 24 1320.35  
## 187 cons 32 1364.22  
## 188 cons 16 1381.89  
## 189 cons 12 1369.95  
## 190 cons 12 1447.74  
## 191 3 1467.33  
## 192 cons 12 1363.97  
## 193 cons 32 1364.97  
## 194 3 1399.49  
## 195 cons 24 1366.99  
## 196 cons 32 1385.26  
## 197 cons 12 1375.96  
## 198 6 1417.19

### 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*.

qualified.players <- merge(ncaa2018, ncaa2019, by=c("Last","First"), all = FALSE)   
print(nrow(qualified.players))

## [1] 198

sum(is.na(qualified.players$Finish.x))

## [1] 0

### 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\_contingency <-table(qualified.players$Weight.x,qualified.players$Weight.y,dnn = c("Weight2018", "Weight2019"))  
weight\_contingency

## Weight2019  
## Weight2018 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

#sum of all cells should equal to the total number of wrestlers that competed in both tournaments  
print(sum(weight\_contingency))

## [1] 198

#the sum of the main diagonal will be the number of wrestlers that competed in the same weight class for both  
print(sum(diag(weight\_contingency)))

## [1] 156

#How many wrestlers changed weight classes?  
print(sum(weight\_contingency)-sum(diag(weight\_contingency)))

## [1] 42