Homework 7 - Data Manipulation, Exercise 2

Andrew Marshall

7/13/2018

# Data

As with previous homework, all the numeric values you need, other than 0.05, 0, 1, 2 and 3 are defined below:

Year=c(1936, 1946, 1951, 1963, 1975, 1997, 2006)  
CaloriesPerRecipeMean <- c(2123.8, 2122.3, 2089.9, 2250.0, 2234.2, 2249.6, 3051.9)  
CaloriesPerRecipeSD <- c(1050.0, 1002.3, 1009.6, 1078.6, 1089.2, 1094.8, 1496.2)  
CaloriesPerServingMean <- c(268.1, 271.1, 280.9, 294.7, 285.6, 288.6, 384.4)  
CaloriesPerServingSD <- c(124.8, 124.2, 116.2, 117.7, 118.3, 122.0, 168.3)  
ServingsPerRecipeMean <- c(12.9, 12.9, 13.0, 12.7, 12.4, 12.4, 12.7)  
ServingsPerRecipeSD <- c(13.3, 13.3, 14.5, 14.6, 14.3, 14.3, 13.0)  
sample.size <- 18  
tenth.increment <- 0.10  
idx.1936 <- 1  
idx.2006 <- length(CaloriesPerRecipeMean)  
idxs36\_07 <- c(idx.1936,idx.2006)  
alpha=0.05  
  
  
CookingTooMuch.dat <- data.frame(  
 Year=Year,  
 CaloriesPerRecipeMean = CaloriesPerRecipeMean,  
 CaloriesPerRecipeSD = CaloriesPerRecipeSD,  
 CaloriesPerServingMean = CaloriesPerServingMean,  
 CaloriesPerServingSD = CaloriesPerServingSD,  
 ServingsPerRecipeMean = ServingsPerRecipeMean,  
 ServingsPerRecipeSD = ServingsPerRecipeSD  
)

Similar restrictions from Homework 4,5 and 6 apply. In this homework we will be manipulating data, so you should not need to call the data.frame function directly except in the initial steps for Exercise 1 using R; in SAS you can use DATA steps as necessary, although some exercises may be easier using PROC SQL.

There are 5 exercises in total, you are required to solve at least 4 and you must solve at least 1 exercise using R and at least 1 exercise using SAS. You can, as previously, solve only 4, using both R and SAS for one problem, or you can solve all 5, as long as you provide at least one R solution and at least one SAS solution.

To simplify grading, include exercise numbers in the uploaded file names.

# Exercise 1.

Recreate the table from Exercise 1, Homework 6 but start with the table from Exercise 1, Homework 5. Specifically, you will start with a data table with 49 rows, but reduce it to 21 rows with a series of manipulations.

## Part a.

Reproduce the code from Homework 5 here. This table should have rows. Call this table ’CohenA` and print the table.

## Part b.

Create a table CohenB from CohenA by selecting only those rows where Year1 is not equal to Year2; alternatively, select only those rows when Cohen is greater than 0.

Print this table.

## Part c

*This step is dependent on knowing that the original matrices from Homework 3 were symmetric and have unique values for each pair of treatment differences*

Sort CohenB by values. This will produce a table where there are pairs of rows, each representing the same mean comparison (that is, and will be consecutive rows). Remove one row from each pair by creating an index that will select every other row (i.e. all even number, all odd numbers, alternating true and false). If you do this in SAS, you can reference the automatic variable \_n\_ in the DATA step - this variable counts the current row number.

Call this table CohenC and print this table.

## Part d.

Sort CohenC by Year1 and Year2 to reproduce the order of rows in the table from Homework 6. Print this table.

# Exercise 2

### Background

I’m working on software that produces a repeated measures analysis. To test my code, I use published data and compare results. For one analysis, I used data from **Contemporary Statistical Models for the Plant and Soil Sciences**, Oliver Schabenberger and Francis J. Pierce, 2001. These data are measurements of the diameter of individual apples from selected apple trees.

## Part a.

Download the AppleData.csv if you choose R; the SAS data is included in the SAS template. Note these files include comments for the data; you may need to specify comment character in import. (The SAS data was where I started).

To simplify this exercise, create a subset of the AppleData including only trees number 3, 7 and 10. (I was going to edit the files to only include these trees, but then I thought, “That’s what computers are for!”)

#Exported file from SAS output and import file into new data frame  
PathToApple3710 = "C:/Users/drewm/Documents/GitHub/code-stat700/sortedapples3710.csv"  
  
#Assigning data from file to data frame  
AppleData3710.df <- read.delim(PathToApple3710,header=TRUE,skip= 0,sep = ",",as.is=TRUE)  
  
#Displaying data in dataframe  
AppleData3710.df

## tree apple time diam  
## 1 3 1 1 2.91  
## 2 3 10 1 2.81  
## 3 3 16 1 2.95  
## 4 3 17 1 2.79  
## 5 3 18 1 2.98  
## 6 3 20 1 2.76  
## 7 3 22 1 2.76  
## 8 3 23 1 2.76  
## 9 3 24 1 2.80  
## 10 7 2 1 2.79  
## 11 7 4 1 2.80  
## 12 7 9 1 3.06  
## 13 7 25 1 2.84  
## 14 10 2 1 2.92  
## 15 10 5 1 2.87  
## 16 10 8 1 2.76  
## 17 10 9 1 2.91  
## 18 10 10 1 2.88  
## 19 10 17 1 3.00  
## 20 10 18 1 2.85  
## 21 10 21 1 2.76  
## 22 10 22 1 3.25  
## 23 10 23 1 3.00  
## 24 3 1 2 3.00  
## 25 3 10 2 2.89  
## 26 3 16 2 3.00  
## 27 3 17 2 2.83  
## 28 3 18 2 3.03  
## 29 3 20 2 2.82  
## 30 3 22 2 2.82  
## 31 3 23 2 2.78  
## 32 3 24 2 2.85  
## 33 7 2 2 2.89  
## 34 7 4 2 2.81  
## 35 7 9 2 3.15  
## 36 7 25 2 2.86  
## 37 10 2 2 2.95  
## 38 10 5 2 2.89  
## 39 10 8 2 2.81  
## 40 10 9 2 3.01  
## 41 10 10 2 2.88  
## 42 10 17 2 3.05  
## 43 10 18 2 2.87  
## 44 10 21 2 2.83  
## 45 10 22 2 3.34  
## 46 10 23 2 3.06  
## 47 3 1 3 3.02  
## 48 3 10 3 2.87  
## 49 3 16 3 3.03  
## 50 3 17 3 2.86  
## 51 3 18 3 3.06  
## 52 3 20 3 2.83  
## 53 3 22 3 2.85  
## 54 3 23 3 2.77  
## 55 3 24 3 2.87  
## 56 7 2 3 2.89  
## 57 7 4 3 2.85  
## 58 7 9 3 3.15  
## 59 7 25 3 2.88  
## 60 10 2 3 3.00  
## 61 10 5 3 2.94  
## 62 10 8 3 2.86  
## 63 10 9 3 3.07  
## 64 10 10 3 2.92  
## 65 10 17 3 3.05  
## 66 10 18 3 2.91  
## 67 10 21 3 2.84  
## 68 10 22 3 3.34  
## 69 10 23 3 3.08  
## 70 3 1 4 3.03  
## 71 3 10 4 2.93  
## 72 3 16 4 3.03  
## 73 3 17 4 2.87  
## 74 3 18 4 3.09  
## 75 3 20 4 2.85  
## 76 3 22 4 2.87  
## 77 3 23 4 2.79  
## 78 3 24 4 2.87  
## 79 7 2 4 2.91  
## 80 7 4 4 2.91  
## 81 7 9 4 3.23  
## 82 7 25 4 2.93  
## 83 10 2 4 3.01  
## 84 10 5 4 2.95  
## 85 10 8 4 2.90  
## 86 10 9 4 3.09  
## 87 10 10 4 2.97  
## 88 10 17 4 3.06  
## 89 10 18 4 2.95  
## 90 10 21 4 2.87  
## 91 10 22 4 3.38  
## 92 10 23 4 3.14  
## 93 3 1 5 NA  
## 94 3 10 5 2.93  
## 95 3 16 5 3.06  
## 96 3 17 5 2.87  
## 97 3 18 5 3.09  
## 98 3 20 5 2.86  
## 99 3 22 5 2.90  
## 100 3 23 5 2.79  
## 101 3 24 5 2.89  
## 102 7 2 5 2.91  
## 103 7 4 5 2.92  
## 104 7 9 5 3.27  
## 105 7 25 5 2.93  
## 106 10 2 5 3.07  
## 107 10 5 5 3.01  
## 108 10 8 5 NA  
## 109 10 9 5 3.11  
## 110 10 10 5 2.97  
## 111 10 17 5 3.11  
## 112 10 18 5 2.98  
## 113 10 21 5 2.88  
## 114 10 22 5 3.47  
## 115 10 23 5 3.18  
## 116 3 1 6 NA  
## 117 3 10 6 2.94  
## 118 3 16 6 3.08  
## 119 3 17 6 2.93  
## 120 3 18 6 3.09  
## 121 3 20 6 2.88  
## 122 3 22 6 2.90  
## 123 3 23 6 2.79  
## 124 3 24 6 2.92  
## 125 7 2 6 2.95  
## 126 7 4 6 2.96  
## 127 7 9 6 3.31  
## 128 7 25 6 2.96  
## 129 10 2 6 NA  
## 130 10 5 6 3.02  
## 131 10 8 6 NA  
## 132 10 9 6 NA  
## 133 10 10 6 2.99  
## 134 10 17 6 NA  
## 135 10 18 6 3.00  
## 136 10 21 6 2.91  
## 137 10 22 6 NA  
## 138 10 23 6 NA

## Part b.

Reshape or transpose this data from the long form to the wide form. Call this data AppleWide. This table should have one column for Tree, one column for Apple and six columns, diam.1 - diam.6. The values in the time columns come from diam in AppleData. If you use SAS, use diam1-diam6 as column names.

#Comment: No file editing was done. File was downloaded and the code provided access to the necessary file and data.  
  
#Exported file from SAS output and import file into new data frame  
PathToAppleData = "C:/Users/drewm/Documents/GitHub/code-stat700/AppleData.csv"  
  
#Assigning data from file to data frame  
AppleData.df <- read.delim(PathToAppleData,header=TRUE,skip= 12,sep = ",",as.is=TRUE)  
  
#Displaying data in dataframe  
AppleData.df

## tree apple time diam  
## 1 1 1 1 2.90  
## 2 1 1 2 2.90  
## 3 1 1 3 2.90  
## 4 1 1 4 2.93  
## 5 1 1 5 2.94  
## 6 1 1 6 2.94  
## 7 1 4 1 2.86  
## 8 1 4 2 2.90  
## 9 1 4 3 2.93  
## 10 1 4 4 2.96  
## 11 1 4 5 2.99  
## 12 1 4 6 3.01  
## 13 1 5 1 2.75  
## 14 1 5 2 2.78  
## 15 1 5 3 2.80  
## 16 1 5 4 2.82  
## 17 1 5 5 2.82  
## 18 1 5 6 2.84  
## 19 1 10 1 2.81  
## 20 1 10 2 2.84  
## 21 1 10 3 2.88  
## 22 1 10 4 2.92  
## 23 1 10 5 2.92  
## 24 1 10 6 2.95  
## 25 1 11 1 2.75  
## 26 1 11 2 2.78  
## 27 1 11 3 2.80  
## 28 1 11 4 2.82  
## 29 1 11 5 2.83  
## 30 1 11 6 2.90  
## 31 1 13 1 2.92  
## 32 1 13 2 2.96  
## 33 1 13 3 2.96  
## 34 1 13 4 3.02  
## 35 1 13 5 3.02  
## 36 1 13 6 3.04  
## 37 1 14 1 3.08  
## 38 1 15 1 3.04  
## 39 1 15 2 3.10  
## 40 1 15 3 3.11  
## 41 1 15 4 3.15  
## 42 1 15 5 3.18  
## 43 1 15 6 3.21  
## 44 1 17 1 2.78  
## 45 1 17 2 2.82  
## 46 1 17 3 2.83  
## 47 1 17 4 2.86  
## 48 1 17 5 2.87  
## 49 1 18 1 2.76  
## 50 1 18 2 2.78  
## 51 1 18 3 2.82  
## 52 1 18 4 2.85  
## 53 1 18 5 2.86  
## 54 1 18 6 2.87  
## 55 1 19 1 2.79  
## 56 1 19 2 2.86  
## 57 1 19 3 2.88  
## 58 1 19 4 2.93  
## 59 1 19 5 2.95  
## 60 1 19 6 2.98  
## 61 1 25 1 2.76  
## 62 1 25 2 2.81  
## 63 1 25 3 2.82  
## 64 1 25 4 2.86  
## 65 1 25 5 2.90  
## 66 1 25 6 2.90  
## 67 2 7 1 2.84  
## 68 2 7 2 2.89  
## 69 2 7 3 2.92  
## 70 2 7 4 2.93  
## 71 2 7 5 2.95  
## 72 2 9 1 2.75  
## 73 2 9 2 2.80  
## 74 2 9 3 2.82  
## 75 2 9 4 2.84  
## 76 2 9 5 2.86  
## 77 2 9 6 2.86  
## 78 2 11 1 2.78  
## 79 2 11 2 2.81  
## 80 2 11 3 2.84  
## 81 2 11 4 2.85  
## 82 2 11 5 2.87  
## 83 2 11 6 2.90  
## 84 2 15 1 2.84  
## 85 2 15 2 2.86  
## 86 2 15 3 2.86  
## 87 2 17 1 2.83  
## 88 2 17 2 2.88  
## 89 2 17 3 2.89  
## 90 2 17 4 2.92  
## 91 2 17 5 2.93  
## 92 2 17 6 2.93  
## 93 2 23 1 2.80  
## 94 2 23 2 2.86  
## 95 2 23 3 2.89  
## 96 2 23 4 2.92  
## 97 2 23 5 2.93  
## 98 2 23 6 2.95  
## 99 2 24 1 2.86  
## 100 2 24 2 2.89  
## 101 2 24 3 2.92  
## 102 2 24 4 2.96  
## 103 2 24 5 2.96  
## 104 2 24 6 2.99  
## 105 2 25 1 2.75  
## 106 2 25 2 2.80  
## 107 2 25 3 2.83  
## 108 2 25 4 2.85  
## 109 2 25 5 2.86  
## 110 2 25 6 2.88  
## 111 3 1 1 2.91  
## 112 3 1 2 3.00  
## 113 3 1 3 3.02  
## 114 3 1 4 3.03  
## 115 3 10 1 2.81  
## 116 3 10 2 2.89  
## 117 3 10 3 2.87  
## 118 3 10 4 2.93  
## 119 3 10 5 2.93  
## 120 3 10 6 2.94  
## 121 3 16 1 2.95  
## 122 3 16 2 3.00  
## 123 3 16 3 3.03  
## 124 3 16 4 3.03  
## 125 3 16 5 3.06  
## 126 3 16 6 3.08  
## 127 3 17 1 2.79  
## 128 3 17 2 2.83  
## 129 3 17 3 2.86  
## 130 3 17 4 2.87  
## 131 3 17 5 2.87  
## 132 3 17 6 2.93  
## 133 3 18 1 2.98  
## 134 3 18 2 3.03  
## 135 3 18 3 3.06  
## 136 3 18 4 3.09  
## 137 3 18 5 3.09  
## 138 3 18 6 3.09  
## 139 3 20 1 2.76  
## 140 3 20 2 2.82  
## 141 3 20 3 2.83  
## 142 3 20 4 2.85  
## 143 3 20 5 2.86  
## 144 3 20 6 2.88  
## 145 3 22 1 2.76  
## 146 3 22 2 2.82  
## 147 3 22 3 2.85  
## 148 3 22 4 2.87  
## 149 3 22 5 2.90  
## 150 3 22 6 2.90  
## 151 3 23 1 2.76  
## 152 3 23 2 2.78  
## 153 3 23 3 2.77  
## 154 3 23 4 2.79  
## 155 3 23 5 2.79  
## 156 3 23 6 2.79  
## 157 3 24 1 2.80  
## 158 3 24 2 2.85  
## 159 3 24 3 2.87  
## 160 3 24 4 2.87  
## 161 3 24 5 2.89  
## 162 3 24 6 2.92  
## 163 4 1 1 2.85  
## 164 4 1 2 2.88  
## 165 4 1 3 2.93  
## 166 4 1 4 2.98  
## 167 4 1 5 3.01  
## 168 4 1 6 3.01  
## 169 4 2 1 2.82  
## 170 4 2 2 2.88  
## 171 4 2 3 2.94  
## 172 4 2 4 2.96  
## 173 4 2 5 2.99  
## 174 4 2 6 3.03  
## 175 4 3 1 2.80  
## 176 4 3 2 2.86  
## 177 4 3 3 2.90  
## 178 4 3 4 2.90  
## 179 4 3 5 2.93  
## 180 4 3 6 2.95  
## 181 4 8 1 2.84  
## 182 4 8 2 2.91  
## 183 4 8 3 2.95  
## 184 4 8 4 2.96  
## 185 4 8 5 3.03  
## 186 4 8 6 3.03  
## 187 4 11 1 2.93  
## 188 4 11 2 2.98  
## 189 4 11 3 3.00  
## 190 4 11 4 3.04  
## 191 4 11 5 3.04  
## 192 4 11 6 3.10  
## 193 4 17 1 2.75  
## 194 4 17 2 2.81  
## 195 4 17 3 2.81  
## 196 4 17 4 2.84  
## 197 4 17 5 2.84  
## 198 4 18 1 2.75  
## 199 4 18 2 2.75  
## 200 4 18 3 2.75  
## 201 4 18 4 2.80  
## 202 4 18 5 2.82  
## 203 4 18 6 2.84  
## 204 4 20 1 2.90  
## 205 4 20 2 2.91  
## 206 4 20 3 2.95  
## 207 4 20 4 3.00  
## 208 4 20 5 3.00  
## 209 4 24 1 2.95  
## 210 4 24 2 3.05  
## 211 4 24 3 3.14  
## 212 4 24 4 3.14  
## 213 4 24 5 3.18  
## 214 4 24 6 3.19  
## 215 4 25 1 2.76  
## 216 4 25 2 2.81  
## 217 4 25 3 2.85  
## 218 4 25 4 2.85  
## 219 4 25 5 2.90  
## 220 5 1 1 2.90  
## 221 5 1 2 2.95  
## 222 5 1 3 2.98  
## 223 5 1 4 3.01  
## 224 5 1 5 3.02  
## 225 5 1 6 3.05  
## 226 5 11 1 2.77  
## 227 5 11 2 2.81  
## 228 5 11 3 2.85  
## 229 5 11 4 2.89  
## 230 5 14 1 2.85  
## 231 5 14 2 2.86  
## 232 5 14 3 2.92  
## 233 5 14 4 2.95  
## 234 5 14 5 2.96  
## 235 5 14 6 2.99  
## 236 5 18 1 2.75  
## 237 5 18 2 2.81  
## 238 5 18 3 2.84  
## 239 5 18 4 2.86  
## 240 5 18 5 2.90  
## 241 5 18 6 2.91  
## 242 5 20 1 2.83  
## 243 5 20 2 2.86  
## 244 5 20 3 2.90  
## 245 5 20 4 2.94  
## 246 5 20 5 2.96  
## 247 5 20 6 2.97  
## 248 5 24 1 2.93  
## 249 5 24 2 2.98  
## 250 5 24 3 3.00  
## 251 5 24 4 3.04  
## 252 5 24 5 3.05  
## 253 5 24 6 3.08  
## 254 6 3 1 2.83  
## 255 6 3 2 2.86  
## 256 6 3 3 2.90  
## 257 6 3 4 2.94  
## 258 6 3 5 2.96  
## 259 6 3 6 2.96  
## 260 6 4 1 2.95  
## 261 6 4 2 3.00  
## 262 6 4 3 3.05  
## 263 6 4 4 3.12  
## 264 6 4 5 3.12  
## 265 6 4 6 3.14  
## 266 6 5 1 2.79  
## 267 6 5 2 2.81  
## 268 6 5 3 2.84  
## 269 6 7 1 2.90  
## 270 6 7 2 2.98  
## 271 6 7 3 3.01  
## 272 6 7 4 3.02  
## 273 6 7 5 3.03  
## 274 6 7 6 3.04  
## 275 6 10 1 2.89  
## 276 6 10 2 2.93  
## 277 6 10 3 2.93  
## 278 6 10 4 2.99  
## 279 6 10 5 2.99  
## 280 6 10 6 3.02  
## 281 6 12 1 2.78  
## 282 6 12 2 2.81  
## 283 6 12 3 2.85  
## 284 6 12 4 2.85  
## 285 6 12 5 2.85  
## 286 6 12 6 2.85  
## 287 6 14 1 2.78  
## 288 6 14 2 2.82  
## 289 6 14 3 2.85  
## 290 6 14 4 2.88  
## 291 6 14 5 2.91  
## 292 6 14 6 2.92  
## 293 6 17 1 2.96  
## 294 6 17 2 3.00  
## 295 6 17 3 3.00  
## 296 6 17 4 3.05  
## 297 6 17 5 3.09  
## 298 6 17 6 3.10  
## 299 6 19 1 2.82  
## 300 6 19 2 2.84  
## 301 6 19 3 2.85  
## 302 6 19 4 2.87  
## 303 6 19 5 2.91  
## 304 6 19 6 2.92  
## 305 6 20 1 2.85  
## 306 6 20 2 2.94  
## 307 6 20 3 2.92  
## 308 6 20 4 3.00  
## 309 6 20 5 3.03  
## 310 6 20 6 3.04  
## 311 6 24 1 2.87  
## 312 6 24 2 2.93  
## 313 6 24 3 2.96  
## 314 6 24 4 3.01  
## 315 6 24 5 3.01  
## 316 6 24 6 3.04  
## 317 7 2 1 2.79  
## 318 7 2 2 2.89  
## 319 7 2 3 2.89  
## 320 7 2 4 2.91  
## 321 7 2 5 2.91  
## 322 7 2 6 2.95  
## 323 7 4 1 2.80  
## 324 7 4 2 2.81  
## 325 7 4 3 2.85  
## 326 7 4 4 2.91  
## 327 7 4 5 2.92  
## 328 7 4 6 2.96  
## 329 7 9 1 3.06  
## 330 7 9 2 3.15  
## 331 7 9 3 3.15  
## 332 7 9 4 3.23  
## 333 7 9 5 3.27  
## 334 7 9 6 3.31  
## 335 7 25 1 2.84  
## 336 7 25 2 2.86  
## 337 7 25 3 2.88  
## 338 7 25 4 2.93  
## 339 7 25 5 2.93  
## 340 7 25 6 2.96  
## 341 8 2 1 2.90  
## 342 8 2 2 2.93  
## 343 8 2 3 2.98  
## 344 8 2 4 3.00  
## 345 8 2 5 3.01  
## 346 8 2 6 3.05  
## 347 8 5 1 2.91  
## 348 8 5 2 2.95  
## 349 8 5 3 3.00  
## 350 8 5 4 3.02  
## 351 8 5 5 3.05  
## 352 8 5 6 3.11  
## 353 8 9 1 3.00  
## 354 8 9 2 3.05  
## 355 8 9 3 3.06  
## 356 8 9 4 3.09  
## 357 8 12 1 2.83  
## 358 8 12 2 2.90  
## 359 8 12 3 2.94  
## 360 8 12 4 2.98  
## 361 8 12 5 3.00  
## 362 8 12 6 3.04  
## 363 8 15 1 2.80  
## 364 8 15 2 2.86  
## 365 8 15 3 2.89  
## 366 8 15 4 2.94  
## 367 8 15 5 2.97  
## 368 8 15 6 2.99  
## 369 8 20 1 2.88  
## 370 8 20 2 2.91  
## 371 8 20 3 2.95  
## 372 8 20 4 2.95  
## 373 8 20 5 3.00  
## 374 8 20 6 3.01  
## 375 8 21 1 2.76  
## 376 8 21 2 2.80  
## 377 8 21 3 2.81  
## 378 8 21 4 2.86  
## 379 8 21 5 2.87  
## 380 8 21 6 2.89  
## 381 9 8 1 2.75  
## 382 9 8 2 2.75  
## 383 9 8 3 2.78  
## 384 9 8 4 2.80  
## 385 9 8 5 2.82  
## 386 9 8 6 2.82  
## 387 9 10 1 2.80  
## 388 9 10 2 2.84  
## 389 9 10 3 2.90  
## 390 9 10 4 2.93  
## 391 9 10 5 2.94  
## 392 9 10 6 2.95  
## 393 9 12 1 2.94  
## 394 9 12 2 2.96  
## 395 9 12 3 2.96  
## 396 9 12 4 3.02  
## 397 9 12 5 3.02  
## 398 9 12 6 3.02  
## 399 10 2 1 2.92  
## 400 10 2 2 2.95  
## 401 10 2 3 3.00  
## 402 10 2 4 3.01  
## 403 10 2 5 3.07  
## 404 10 5 1 2.87  
## 405 10 5 2 2.89  
## 406 10 5 3 2.94  
## 407 10 5 4 2.95  
## 408 10 5 5 3.01  
## 409 10 5 6 3.02  
## 410 10 8 1 2.76  
## 411 10 8 2 2.81  
## 412 10 8 3 2.86  
## 413 10 8 4 2.90  
## 414 10 9 1 2.91  
## 415 10 9 2 3.01  
## 416 10 9 3 3.07  
## 417 10 9 4 3.09  
## 418 10 9 5 3.11  
## 419 10 10 1 2.88  
## 420 10 10 2 2.88  
## 421 10 10 3 2.92  
## 422 10 10 4 2.97  
## 423 10 10 5 2.97  
## 424 10 10 6 2.99  
## 425 10 17 1 3.00  
## 426 10 17 2 3.05  
## 427 10 17 3 3.05  
## 428 10 17 4 3.06  
## 429 10 17 5 3.11  
## 430 10 18 1 2.85  
## 431 10 18 2 2.87  
## 432 10 18 3 2.91  
## 433 10 18 4 2.95  
## 434 10 18 5 2.98  
## 435 10 18 6 3.00  
## 436 10 21 1 2.76  
## 437 10 21 2 2.83  
## 438 10 21 3 2.84  
## 439 10 21 4 2.87  
## 440 10 21 5 2.88  
## 441 10 21 6 2.91  
## 442 10 22 1 3.25  
## 443 10 22 2 3.34  
## 444 10 22 3 3.34  
## 445 10 22 4 3.38  
## 446 10 22 5 3.47  
## 447 10 23 1 3.00  
## 448 10 23 2 3.06  
## 449 10 23 3 3.08  
## 450 10 23 4 3.14  
## 451 10 23 5 3.18

#Exported file from SAS output and import file into new data frame  
PathToAppleWide = "C:/Users/drewm/Documents/GitHub/code-stat700/AppleWide.csv"  
  
#Assigning data from file to data frame  
AppleWide.df <- read.delim(PathToAppleWide,header=TRUE,skip= 0,sep = ",",as.is=TRUE)  
  
#Displaying data in dataframe  
AppleWide <- AppleWide.df  
  
AppleWide

## tree apple diam1 diam2 diam3 diam4 diam5 diam6  
## 1 3 1 2.91 3.00 3.02 3.03 NA NA  
## 2 3 10 2.81 2.89 2.87 2.93 2.93 2.94  
## 3 3 16 2.95 3.00 3.03 3.03 3.06 3.08  
## 4 3 17 2.79 2.83 2.86 2.87 2.87 2.93  
## 5 3 18 2.98 3.03 3.06 3.09 3.09 3.09  
## 6 3 20 2.76 2.82 2.83 2.85 2.86 2.88  
## 7 3 22 2.76 2.82 2.85 2.87 2.90 2.90  
## 8 3 23 2.76 2.78 2.77 2.79 2.79 2.79  
## 9 3 24 2.80 2.85 2.87 2.87 2.89 2.92  
## 10 7 2 2.79 2.89 2.89 2.91 2.91 2.95  
## 11 7 4 2.80 2.81 2.85 2.91 2.92 2.96  
## 12 7 9 3.06 3.15 3.15 3.23 3.27 3.31  
## 13 7 25 2.84 2.86 2.88 2.93 2.93 2.96  
## 14 10 2 2.92 2.95 3.00 3.01 3.07 NA  
## 15 10 5 2.87 2.89 2.94 2.95 3.01 3.02  
## 16 10 8 2.76 2.81 2.86 2.90 NA NA  
## 17 10 9 2.91 3.01 3.07 3.09 3.11 NA  
## 18 10 10 2.88 2.88 2.92 2.97 2.97 2.99  
## 19 10 17 3.00 3.05 3.05 3.06 3.11 NA  
## 20 10 18 2.85 2.87 2.91 2.95 2.98 3.00  
## 21 10 21 2.76 2.83 2.84 2.87 2.88 2.91  
## 22 10 22 3.25 3.34 3.34 3.38 3.47 NA  
## 23 10 23 3.00 3.06 3.08 3.14 3.18 NA

## Part c.

To confirm that you’ve reshaped correctly, print column means for the wide data set and use an aggregate or apply function to compute time means for the long format. If SAS, use PROC MEANS. Call with var diam; by time; for one table, and var diam1-diam6; for the other.

#Exported file from SAS output and import file into new data frame  
PathToAppleWideMeans = "C:/Users/drewm/Documents/GitHub/code-stat700/AppleWideMeans.csv"  
  
#Assigning data from file to data frame  
AppleWideMeans.df <- read.delim(PathToAppleWideMeans,header=TRUE,skip= 0,sep = ",",as.is=TRUE)  
  
#Displaying data in dataframe  
AppleWideMeans.df

## X\_TYPE\_ X\_FREQ\_ X\_STAT\_ diam1 diam2 diam3 diam4  
## 1 0 23 N 23.0000000 23.0000000 23.0000000 23.0000000  
## 2 0 23 MIN 2.7600000 2.7800000 2.7700000 2.7900000  
## 3 0 23 MAX 3.2500000 3.3400000 3.3400000 3.3800000  
## 4 0 23 MEAN 2.8786957 2.9313043 2.9539130 2.9839130  
## 5 0 23 STD 0.1215171 0.1334951 0.1310704 0.1363064  
## diam5 diam6  
## 1 21.0000000 16.0000000  
## 2 2.7900000 2.7900000  
## 3 3.4700000 3.3100000  
## 4 3.0095238 2.9768750  
## 5 0.1593573 0.1153961

## Part d.

I choose this example for a test case because it shows a case where the best repeated measures model is an autoregressive model - each measure is correlated with the preceding measure. We can estimate the degree of using the following R code. You don’t need to evaluate this code for this exercise; it’s provided as a motivation for reshaping the data.

mult.lm <- lm(cbind(diam1, diam2, diam3, diam4, diam5, diam6) ~ tree, data=AppleWide)  
mult.manova <- manova(mult.lm)  
print(cov2cor(estVar(mult.lm)))

If you use SAS, the equivalent code is part of the PROC GLM block, look for the table title **Partial Correlation Coefficients from the Error SSCP Matrix / Prob > |r|**.

# Exercise 3

This is an exercise in computing the Wilcoxon Signed Rank test. We will be using an example from NIST (NATR332.DAT, under <https://www.itl.nist.gov/div898/software/dataplot/datasets.htm> ). See <https://www.itl.nist.gov/div898/software/dataplot/refman1/auxillar/signrank.htm> for a reference.

The data are provided:

NATR332.DAT <- data.frame(  
 Y1 = c(146,141,135,142,140,143,138,137,142,136),  
 Y2 = c(141,143,139,139,140,141,138,140,142,138)  
)

## Part a.

Add the column Difference by subtracting Y1 from Y2. For further analysis, exclude any rows where the difference is 0.

## Part b.

Next add the column Rank as the rank of the absolute value of Difference.

## Part c.

Add the column SignedRank by applying the sign (+ or -) of Difference, to to Rank (that is, if Difference is < 0, then SignedRank is -Rank, otherwise SignedRank is Rank).

## Part d.

Compute the sum of the positive ranks, and the absolute value of the sum of the negative ranks.

Let be the minimum of these two sums. Print .

The expected mean of is calculated by with a standard deviation of

where is the number of ranked values (excluding differences of 0). Calculate a score by

and the probability of greater . Print both and .

The NIST page gives a p-value, 0.5677, based on the continuity correction. We are not computing this correction. You can compare the of your in R by

wilcox.test(NATR332.DAT$Y1, NATR332.DAT$Y2, paired = TRUE, correct = FALSE, alternative = "greater")

## Warning in wilcox.test.default(NATR332.DAT$Y1, NATR332.DAT$Y2, paired =  
## TRUE, : cannot compute exact p-value with ties

## Warning in wilcox.test.default(NATR332.DAT$Y1, NATR332.DAT$Y2, paired =  
## TRUE, : cannot compute exact p-value with zeroes

##   
## Wilcoxon signed rank test  
##   
## data: NATR332.DAT$Y1 and NATR332.DAT$Y2  
## V = 13.5, p-value = 0.534  
## alternative hypothesis: true location shift is greater than 0

# Exercise 4

In the ‘fastest.csv’ data set from Homework 5, Exercise 5, there are several motorcycles that were the fastest in production in their time, but were not the fastest in history, up to that point. That is, some motorcycles were fastest because a faster motorcycle had gone out of production.

## Part a

Programmatically determine these motorcycles. If you can show that the gaps between the *fastest ever* and *currently fastest* is not greater than one model, you can simply compare ; otherwise you will need to iterate over each row and deterimine the maximimum MPH for all preceding rows.

Create and print a vector of the model names for these motorcycles.

## Part b.

Create a subset of the original fastest.csv that contains those motorcycles not found in the motorcycles identified in part a. Print this table.

## Part c.

Compare these two data sets by creating a staircase or step plot including both sets, plotting MPH as the dependent variable and Year as the independent variable. Use different colors for the lines for each set.

Add a vertical line at 1949, marking the year that the Vincent Black Lightning was introduced.

# Exercise 5

## Part a

Go to <http://www.itl.nist.gov/div898/strd/anova/AtmWtAg.html> and download the file listed under Data File in Table Format (ttps://www.itl.nist.gov/div898/strd/anova/AtmWtAgt.dat). You may have done this in the previous homework. This file is in the wide format; you will be expected to reshape or transpose this to the long format.

Do not read the file listed under Data File in Two-Column Format. This data file is in the long format. You can use it to check your work, but for this exercise you might write code to obtain data in the long form.

## Part b

Reshape or tranpose this table from the wide format to the long format. Make sure the resulting table has two columns - AgWt and Instrument. Name this table AtmWtAg.long and print this table. If you choose SAS, you may need to a row number as ID - you might use the automatic variable \_n\_ in the DATA step; use the table name AtmWtAgLong.

## Part c

To confirm that the table was reshaped correctly, use aggregate or tapply to calculate means from the long table and use apply or colMeans to calculate means from the wide table. Print and compare the results. If you choose SAS, see the instructions for Exercise 2, Part c.

## Part d.

Convert the Instrument column in the long data set to a factor, then set eval=TRUE in the code chunk below. Compare the intercept to the mean of instrument 1, and the slope to the different between the two means. Use the PROC GLM block for SAS.

AtmWtAg.long$Instrument <- as.factor(AtmWtAg.long$Instrument)  
summary(lm(AgWt ~ Instrument,data=AtmWtAg.long))