Homework Assignment #4

Matt Kline

February 21, 2017

Part 1 Data Frames:

1.)

a. Use data.frame() to create a data frame called pretendDf which contains the following data:

```
x <- c(61,175,111,124)
y <- c(13,21,24,23)
z <- c(4,18,14,18)
pretendDf <- data.frame(x,y,z)
```

b. Display the row 1, column 3 element of pretendDf.

```
pretendDf[1,3]

## [1] 4
```

c. Use two different commands, one involving $ and the other involving [[ ]], to display the y column of pretendDf.

```
pretendDf$y

## [1] 13 21 24 23

pretendDf[[2]]

## [1] 13 21 24 23
```

d. Try the following and report what happens:

```
stack(pretendDf)

##    values ind
## 1      61   x
## 2     175   x
## 3     111   x
## 4     124   x
## 5      13   y
## 6      21   y
## 7      24   y
## 8      23   y
## 9       4   z
## 10     18   z
## 11     14   z
## 12     18   z
```

This lays out the columns all in one line organized by column.

2.) R is equipped with several built-in datasets, including the famous iris (flowers) data collected by Anderson and analyzed by Fisher in the 1930s. You can see its values by typing its name, iris, just as you would with datasets that you create yourself:

```
#iris
```

More information about the dataset can be found in the help file:

```
?iris

## starting httpd help server ...
##  done
```

a. Inspect the iris dataset using head(), dim(), str(), and names().

```
head(iris)

##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa

dim(iris)

## [1] 150   5

str(iris)

## 'data.frame': 150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...

names(iris)

## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
## [5] "Species"
```

b. Compute the mean and standard deviation of the pedal lengths.

```
mean(iris$Petal.Length)

## [1] 3.758

sd(iris$Petal.Length)

## [1] 1.765298
```

c. Use square brackets [ ] to extract just the rows corresponding to Iris versicolor flowers.

```
versi <- iris[iris$Species == "versicolor",]
```

d. Compute the mean and standard deviation of the Iris versicolor pedal lengths.

```
sapply(versi, mean)

## Warning in mean.default(X[[i]], ...): argument is not numeric or logical: returning
NA

## Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
##        5.936        2.770        4.260        1.326           NA

sapply(versi, sd)

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm = na.rm):
Calling var(x) on a factor x is deprecated and will become an error.
##  Use something like 'all(duplicated(x)[-1L])' to test for a constant vector.

## Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
##    0.5161711    0.3137983    0.4699110    0.1977527    0.0000000
```

e. Create a new data frame that consists of the numeric columns of the iris dataset and calculate the means of its columns using colMeans().

```
colMeans(iris[1:4])

## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##     5.843333     3.057333     3.758000     1.199333
```

3.) Two other built-in R datasets are beaver1 and beaver2. They contain body temperatures of two beavers. You can view them by typing their names, e.g.

```
#beaver1
```

More information can be found in the help file

```
?beaver1
```

a. Add a column named id to the beaver1 dataset, where the value is always 1. Similarly, add an id column to beaver2, with value 2.

```
x <- c(1)
b1 <- cbind(beaver1, x)
x <- c(2)
b2 <- cbind(beaver2, x)
```

b. Use rbind() to vertically combine the two data frames, then extract the subset where either beaver is active (activ == 1).

```r
bcomb <- rbind(b1, b2)
bcomb[bcomb$activ == 1,]
```

```
##     day time  temp activ x
## 54  346 1730 37.07     1 1
## 68  346 1950 37.10     1 1
## 80  346 2150 37.53     1 1
## 83  346 2230 37.25     1 1
## 86  346 2300 37.24     1 1
## 114 347  340 37.15     1 1
## 153 307 1550 37.98     1 2
## 154 307 1600 38.02     1 2
## 155 307 1610 38.00     1 2
## 156 307 1620 38.24     1 2
## 157 307 1630 38.10     1 2
## 158 307 1640 38.24     1 2
## 159 307 1650 38.11     1 2
## 160 307 1700 38.02     1 2
## 161 307 1710 38.11     1 2
## 162 307 1720 38.01     1 2
## 163 307 1730 37.91     1 2
## 164 307 1740 37.96     1 2
## 165 307 1750 38.03     1 2
## 166 307 1800 38.17     1 2
## 167 307 1810 38.19     1 2
## 168 307 1820 38.18     1 2
## 169 307 1830 38.15     1 2
## 170 307 1840 38.04     1 2
## 171 307 1850 37.96     1 2
## 172 307 1900 37.84     1 2
## 173 307 1910 37.83     1 2
## 174 307 1920 37.84     1 2
## 175 307 1930 37.74     1 2
## 176 307 1940 37.76     1 2
## 177 307 1950 37.76     1 2
## 178 307 2000 37.64     1 2
## 179 307 2010 37.63     1 2
## 180 307 2020 38.06     1 2
## 181 307 2030 38.19     1 2
## 182 307 2040 38.35     1 2
## 183 307 2050 38.25     1 2
## 184 307 2100 37.86     1 2
## 185 307 2110 37.95     1 2
## 186 307 2120 37.95     1 2
## 187 307 2130 37.76     1 2
## 188 307 2140 37.60     1 2
## 189 307 2150 37.89     1 2
## 190 307 2200 37.86     1 2
## 191 307 2210 37.71     1 2
## 192 307 2220 37.78     1 2
## 193 307 2230 37.82     1 2
## 194 307 2240 37.76     1 2
## 195 307 2250 37.81     1 2
## 196 307 2300 37.84     1 2
## 197 307 2310 38.01     1 2
```

```
## 198 307 2320 38.10     1 2
## 199 307 2330 38.15     1 2
## 200 307 2340 37.92     1 2
## 201 307 2350 37.64     1 2
## 202 308    0 37.70     1 2
## 203 308   10 37.46     1 2
## 204 308   20 37.41     1 2
## 205 308   30 37.46     1 2
## 206 308   40 37.56     1 2
## 207 308   50 37.55     1 2
## 208 308  100 37.75     1 2
## 209 308  110 37.76     1 2
## 210 308  120 37.73     1 2
## 211 308  130 37.77     1 2
## 212 308  140 38.01     1 2
## 213 308  150 38.04     1 2
## 214 308  200 38.07     1 2
```

c. Find the mean body temperature when the beavers are active and the mean when they're inactive and compare the two.

```
sapply(bcomb[bcomb$activ == 1,], mean)

##         day        time        temp       activ           x
##  310.647059 1590.441176   37.843088    1.000000    1.911765

sapply(bcomb[bcomb$activ == 0,], mean)

##         day        time        temp       activ           x
##  336.000000 1274.246575   36.908425    0.000000    1.260274
```

4.) The file carriers.txt contains data from a study to develop screening methods to identify carriers of a rare genetic disorder. Four measurements m1, m2, m3, m4 were made on blood sam- ples. One of these, m1, has been used before. The variables are:

ObsNo = Observation number (sequence number per patient). Note that there are several samples per patient for some patients. HospID = Hospital identification number for blood sample Age = Age of patient Date = Date that blood sample was taken (mmddyy). Note that all day entries are 00. m1 = Measurement 1 (xxx.x) m2 = Measurement 2 (xxx.x). Eight missing data values. m3 = Measurement 3 (xxx.x) m4 = Measurement 4 (xxx.x). Seven missing data values.

a. Use read.table() (with header = TRUE) to read the data into a data frame named blood. Examine blood using head(), dim(), names(), and str().

```
carriers <- read.table("/Users/Matt/Downloads/carriers.txt", header = T)
head(carriers)

##   ObsNo HospID Age   Date  m1  m2   m3  m4
## 1     1   1027  30 100078 167  89 25.6 364
## 2     1   1013  41 100078 104  81 26.8 245
## 3     1   1324  22  80079  30 108  8.8 284
## 4     2   1332  22  80079  44 104 17.4 172
## 5     1    966  20 100078  65  87 23.8 198
## 6     1    979  42  90078 440 107 20.2 239
```

```
dim(carriers)

## [1] 75  8

names(carriers)

## [1] "ObsNo"  "HospID" "Age"    "Date"   "m1"     "m2"     "m3"     "m4"

str(carriers)

## 'data.frame': 75 obs. of  8 variables:
##  $ ObsNo : int  1 1 1 2 1 1 1 1 2 3 ...
##  $ HospID: int  1027 1013 1324 1332 966 979 1327 978 1290 1139 ...
##  $ Age   : int  30 41 22 22 20 42 59 35 36 35 ...
##  $ Date  : int  100078 100078 80079 80079 100078 90078 80079 90078 60079 20079 ...
##  $ m1    : num  167 104 30 44 65 440 58 129 104 122 ...
##  $ m2    : num  89 81 108 104 87 107 88.2 93.1 87.5 88.5 ...
##  $ m3    : num  25.6 26.8 8.8 17.4 23.8 20.2 11 18.3 16.7 21.6 ...
##  $ m4    : int  364 245 284 172 198 239 259 188 256 263 ...
```

b. Use complete.cases() to create a new data frame, blood2, containing just the rows of blood for which there are no NAs.

```
blood2 <- carriers[complete.cases(carriers),]
blood2

##    ObsNo HospID Age   Date   m1    m2    m3  m4
## 1      1   1027  30 100078  167  89.0  25.6 364
## 2      1   1013  41 100078  104  81.0  26.8 245
## 3      1   1324  22  80079   30 108.0   8.8 284
## 4      2   1332  22  80079   44 104.0  17.4 172
## 5      1    966  20 100078   65  87.0  23.8 198
## 6      1    979  42  90078  440 107.0  20.2 239
## 7      1   1327  59  80079   58  88.2  11.0 259
## 8      1    978  35  90078  129  93.1  18.3 188
## 9      2   1290  36  60079  104  87.5  16.7 256
## 10     3   1139  35  20079  122  88.5  21.6 263
## 12     1   1193  29  40079  265  83.5  16.1 136
## 14     1   1208  27  40079  285  79.5  36.4 245
## 15     2   1395  27  90079   25  91.0  49.1 209
## 16     1   1209  28  40079  124  92.0  32.2 298
## 17     1    947  29  80078   53  76.0  14.0 174
## 18     2   1153  30  20079   46  71.0  16.9 197
## 19     3   1311  30  70079   40  85.5  12.7 201
## 20     4   1325  30  80079   41  90.0   9.7 342
## 22     1    923  31  60078  657 104.0 110.0 358
## 23     2   1156  32  20079  465  86.5  63.7 412
## 24     3   1266  32  50079  485  83.5  73.0 382
## 26     1   1135  37  20079  168  82.5  23.3 261
## 27     1    914  38  60078  286 109.5  31.9 260
## 28     2   1124  39  10079  388  91.0  41.6 204
## 29     3   1398  39  90079  148 105.2  18.8 221
## 30     1    913  34  60078   73 105.5  17.0 285
## 31     2   1223  35  40079   36  92.8  22.0 308
```

```
## 33     1    970  58  80078   19 100.5  10.9 196
## 34     2   1155  58  20079   34  98.5  19.9 299
## 36     1   1109  38  10079  113  97.0  18.8 216
## 37     1   1354  30  80079   57 105.0  12.9 155
## 38     1    949  42  80078   78 118.0  15.5 212
## 39     2   1066  43 110078   73 104.0  20.6 201
## 40     1   1168  29  30079   69 111.0  16.0 175
## 41     2   1447  30 100079  177 103.5  19.8 241
## 42     1    911  35  60078   48  98.0  16.4 233
## 43     2    951  35  70078   34  96.5  10.4 122
## 44     3   1009  35  90078   42 100.1  17.1 184
## 45     1   1358  44  90079  109  81.0  25.3 227
## 46     1   1115  35  90079  925  81.0  62.9 279
## 47     2   1203  35  40079 1288  82.0  51.6 368
## 48     3   1381  36  90079  325  76.3  33.9 413
## 49     1    929  53  60078   59  93.0  22.2 240
## 50     2   1236  54  40079   69  92.6  20.9 243
## 51     1   1202  30  40079  363  91.3  36.0 325
## 52     1   1050  35 110078   37  84.0  12.8 156
## 53     1   1289  53  60079  101  77.5  11.7 280
## 54     1   1173  41  30079   99  93.2  18.6 156
## 55     2   1008  40  90078  125  90.5  19.4 438
## 56     3   1328  42  80079   52  93.3  11.2 272
## 57     1   1303  59  60079  560 106.0  21.0 345
## 58     1    956  31  80078   85  94.0  20.1 198
## 60     3   1302  32  60079   72  88.0   8.3 166
## 61     1    953  52  60078  197  91.5  25.2 236
## 62     2   1163  52  30078  242  85.5  16.6 168
## 63     3   1334  53  80079  245  89.5  22.7 269
## 64     1   1030  39 100078  154 103.5  21.3 296
## 65     2   1306  39  60079  228 104.0  10.2 236
## 67     1   1323  43  80079   80  90.5  12.1 269
## 68     1    902  44  60078   28 104.0  22.0 142
## 69     2   1296  45  60079   35  86.3  14.4 184
## 70     1   1249  33  50079   57  88.0   8.9 190
## 71     1    955  26 110078  326  98.0  27.1 358
## 72     2   1307  26  60079  700  90.0  49.1 343
## 73     1    984  61  90078  100 101.0  11.8 301
## 74     2   1141  61  20079   80  97.5  15.1 262
## 75     1   1305  48  60079  115  79.0  14.2 258
```

c. Compute the mean of the m2 column of blood2.

```
mean(blood2[[6]])

## [1] 92.93134
```

5.) The files florida vote counts1.txt and florida vote counts2.txt contain county data from the 2000 presidential election in Florida. For each of the 67 Florida counties, the data include the type of voting machine used, the number of columns in the presidential ballot, the undervote, the overvote, and the vote counts for each of the presidential candidates.

The vote counts are the final certified counts reported by the Florida Division of Elections. An overvote happens when you vote for more candidates than the number of candidates you are permitted to vote for in a particular office race. Undervoting means that you cast fewer votes than you are permitted to cast.

Of particular interest are the Buchanan vote in Palm Beach county, and the overvote as a function of voting machine type and number of columns.

a. Use read.table() (with header = TRUE) to read the data into two data frames, votes1 and votes2. Examine the two data frames using head(), dim(), names(), and str().

```
votes1 <- read.table("/Users/Matt/Downloads/flordia_vote_counts1.txt", header = T)
votes2 <- read.table("/Users/Matt/Downloads/flordia_vote_counts2.txt", header = T)

head(votes1)

##     county technology columns under over   Bush   Gore
## 1  Alachua    Optical       1   217  105  34124  47365
## 2    Baker    Optical       1    79   46   5610   2392
## 3      Bay    Optical       1   541  141  38637  18850
## 4 Bradford    Optical       2    41  695   5414   3075
## 5  Brevard    Optical       1   277  136 115185  97318
## 6  Broward  Votomatic       1  4946 7826 177902 387703

dim(votes1)

## [1] 67  7

names(votes1)

## [1] "county"     "technology" "columns"    "under"      "over"
## [6] "Bush"       "Gore"

str(votes1)

## 'data.frame': 67 obs. of  7 variables:
##  $ county    : Factor w/ 67 levels "Alachua","Baker",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ technology: Factor w/ 5 levels "Datavote","Hand",..: 4 4 4 4 4 5 4 4 4 4 ...
##  $ columns   : int  1 1 1 2 1 1 1 2 1 1 ...
##  $ under     : int  217 79 541 41 277 4946 78 170 154 223 ...
##  $ over      : int  105 46 141 695 136 7826 0 2985 54 157 ...
##  $ Bush      : int  34124 5610 38637 5414 115185 177902 2873 35426 29767 41736 ...
##  $ Gore      : int  47365 2392 18850 3075 97318 387703 2155 29645 25525 14632 ...

head(votes2)

##     county Browne Nader Harris Hagelin Buchanan McReynolds Phillips
## 1  Alachua    658  3226      6      42      263          4       20
## 2    Baker     17    53      0       3       73          0        3
## 3      Bay    171   828      5      18      248          3       18
## 4 Bradford     28    84      0       2       65          0        2
## 5  Brevard    643  4470     11      39      570         11       72
## 6  Broward   1217  7104     54     135      795         37       74
##   Moorehead Chote McCarthy
## 1        21     0        0
## 2         3     0        0
## 3        27     0        0
## 4         3     0        0
## 5        76     0        0
## 6       122     0        0
```

```
dim(votes2)

## [1] 67 11

names(votes2)

##  [1] "county"     "Browne"     "Nader"      "Harris"     "Hagelin"
##  [6] "Buchanan"   "McReynolds" "Phillips"   "Moorehead"  "Chote"
## [11] "McCarthy"

str(votes2)

## 'data.frame': 67 obs. of  11 variables:
##  $ county    : Factor w/ 67 levels "Alachua","Baker",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ Browne    : int  658 17 171 28 643 1217 10 127 194 204 ...
##  $ Nader     : int  3226 53 828 84 4470 7104 39 1462 1379 562 ...
##  $ Harris    : int  6 0 5 0 11 54 0 6 5 1 ...
##  $ Hagelin   : int  42 3 18 2 39 135 1 15 16 14 ...
##  $ Buchanan  : int  263 73 248 65 570 795 90 182 270 186 ...
##  $ McReynolds: int  4 0 3 0 11 37 1 3 0 3 ...
##  $ Phillips  : int  20 3 18 2 72 74 2 18 18 6 ...
##  $ Moorehead : int  21 3 27 3 76 122 3 12 28 9 ...
##  $ Chote     : int  0 0 0 0 0 0 0 0 2 0 ...
##  $ McCarthy  : int  0 0 0 0 0 0 0 0 0 0 ...
```

b. Merge votes1 and votes2 together, by county, using merge(). Save the result in a data frame named votes.

```
votes <- merge(votes1, votes2)
```

c. Create a new column in votes named total containing the total number of votes cast in each county. Hint: Use apply(), with FUN = sum, making sure to only apply sum() to the columns of vote that contain vote counts.

```
total <- apply(votes[c("Browne", "Nader", "Harris", "Hagelin", "Buchanan", "McReynolds", "Phill
votes <- cbind(votes, total)
```

d. Sort the rows of votes according to the values in the total column, for example by typing:

```
#votes[order(votes£total), ]
```

Which county cast the fewest total votes? Which cast the most?

e. Compute the median number of overvotes for counties whose ballots had 1 column, and the median for counties whose ballots had 2 columns. Based on the medians, which type of ballot leads to more overvotes?

```
aggregate(votes$over, by = list(votes$columns), median)

##   Group.1     x
## 1       1 293.5
## 2       2 695.0
```