Assignment 9

Matthew McAvoy

September 30, 2016

1

Lets begin by reading in the data.

```
url_data <- "https://archive.ics.uci.edu/ml/machine-learning-databases/hepatitis/hepatitis.data"
url_names <- "https://archive.ics.uci.edu/ml/machine-learning-databases/hepatitis/hepatitis.names"

hep_data_start <- read.csv(url_data)
```

It looks like there are no labels on the data. I downloaded the files locally and processed hepatitis.names to add the attribute names corresponding to column names to the data file. Then from the names file, I see missing columns are labelled as '?'. I used na.strings to tell R that this means na.

```
hep_data <- read.csv("C:/Users/homur/OneDrive/New College/EDA/Week 6/hepatitis_data.txt", header=TRUE, n
```

Looking at the names file again, it looks like we have found the correct number of missing items by some queries.

```
sum(is.na(hep_data[,1]))
```

```
## [1] 0
```

```
sum(is.na(hep_data[,16]))
```

```
## [1] 29
```

```
sum(is.na(hep_data[,18]))
```

```
## [1] 16
```

```
sum(is.na(hep_data[,20]))
```

```
## [1] 0
```

4

To look at the number of complete cases, we can run a quick R command. It looks like out of 155 cases, 80 of them are complete.

```
sum(complete.cases(hep_data))
```

```
## [1] 80
```

5

lets subset the data on age, sex, bilirubin, alk, sgot, and albumin. This corresponds to column numbers 2,3,15,16,17,18. First using as.numeric on each column then subsetting allows it to become numeric.

```
hep_data$BILIRUBIN <- as.numeric(as.character(hep_data$BILIRUBIN))
hep_data$ALK_PHOSPHATE <- as.numeric(as.character(hep_data$ALK_PHOSPHATE))
hep_data$SGOT <- as.numeric(as.character(hep_data$SGOT))
hep_data$ALBUMIN <- as.numeric(as.character(hep_data$ALBUMIN))
subD1 <- hep_data[,c(2,3,15,16,17,18)]
```

There are now 120 complete cases in our subset data.

```
sum(complete.cases(subD1))
```

```
## [1] 120
```

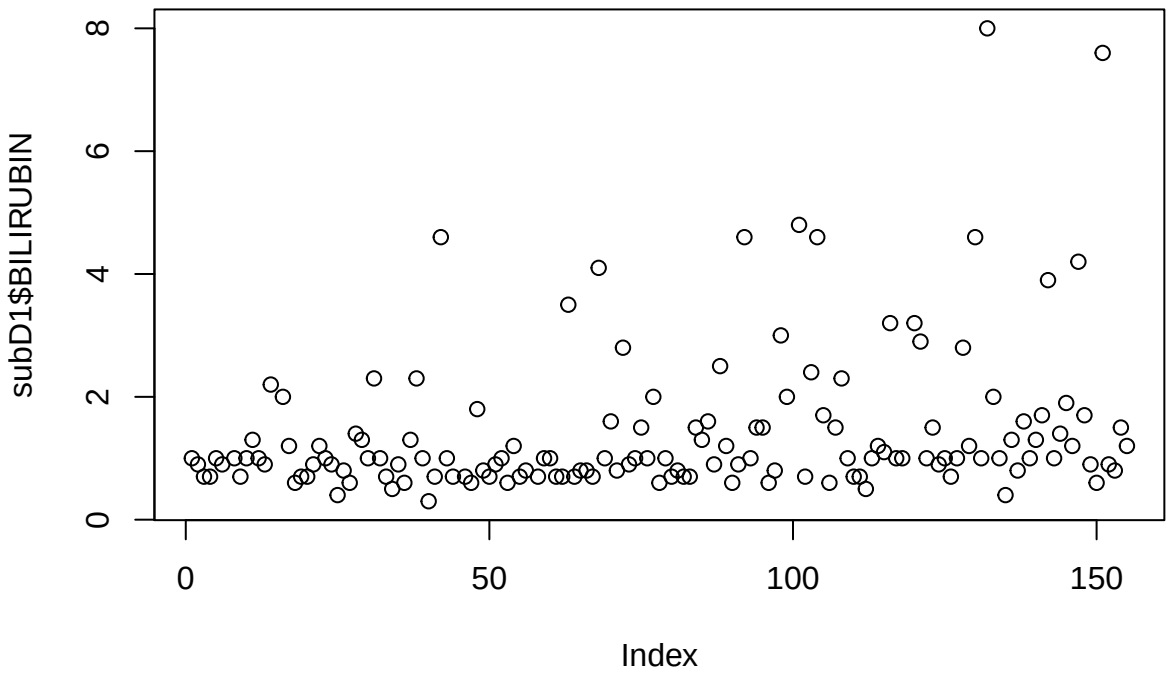
6

To look at if there are any outliers, we can plot the data as well as look at the summary of each variable.

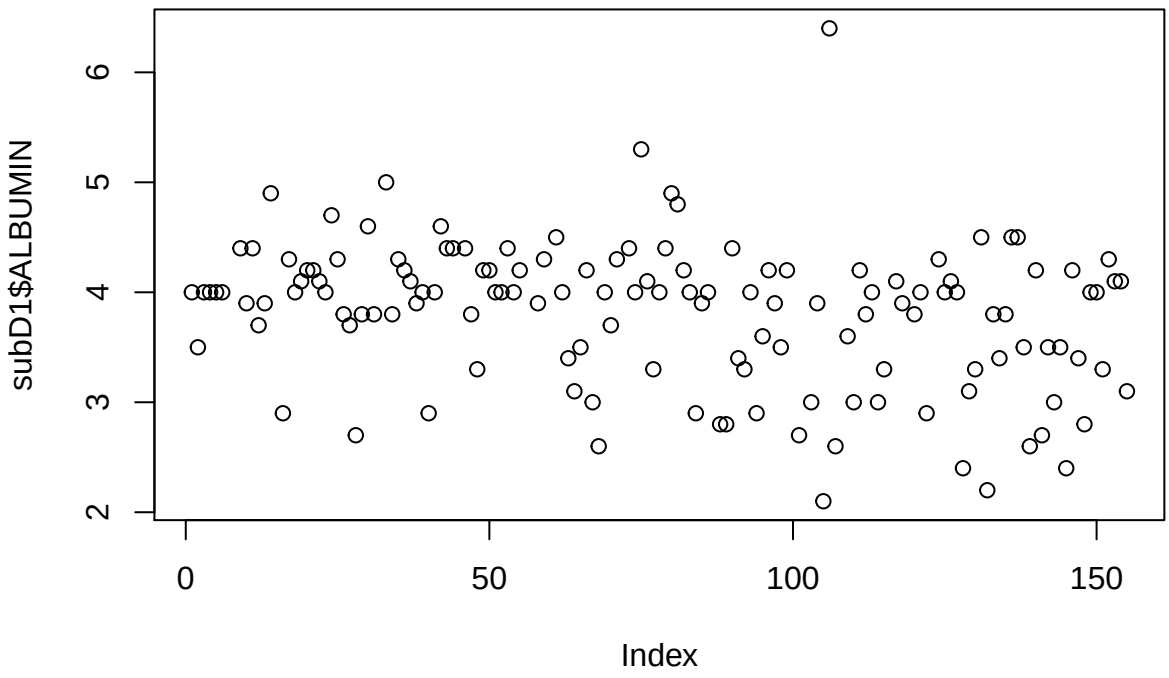
```
summary(subD1)
```

```
##       AGE             SEX           BILIRUBIN     ALK_PHOSPHATE
##  Min.   : 7.0   Min.   :1.000   Min.   :0.300   Min.   : 26.00
##  1st Qu.:32.0   1st Qu.:1.000   1st Qu.:0.700   1st Qu.: 74.25
##  Median :39.0   Median :1.000   Median :1.000   Median : 85.00
##  Mean   :41.2   Mean   :1.103   Mean   :1.428   Mean   :105.33
##  3rd Qu.:50.0   3rd Qu.:1.000   3rd Qu.:1.500   3rd Qu.:132.25
##  Max.   :78.0   Max.   :2.000   Max.   :8.000   Max.   :295.00
##                                 NA's   :6       NA's   :29
##       SGOT            ALBUMIN
##  Min.   : 14.00   Min.   :2.100
##  1st Qu.: 31.50   1st Qu.:3.400
##  Median : 58.00   Median :4.000
##  Mean   : 85.89   Mean   :3.817
##  3rd Qu.:100.50   3rd Qu.:4.200
##  Max.   :648.00   Max.   :6.400
##  NA's   :4        NA's   :16
```

```
plot(subD1$BILIRUBIN)
```




```
plot(subD1$ALBUMIN)
```

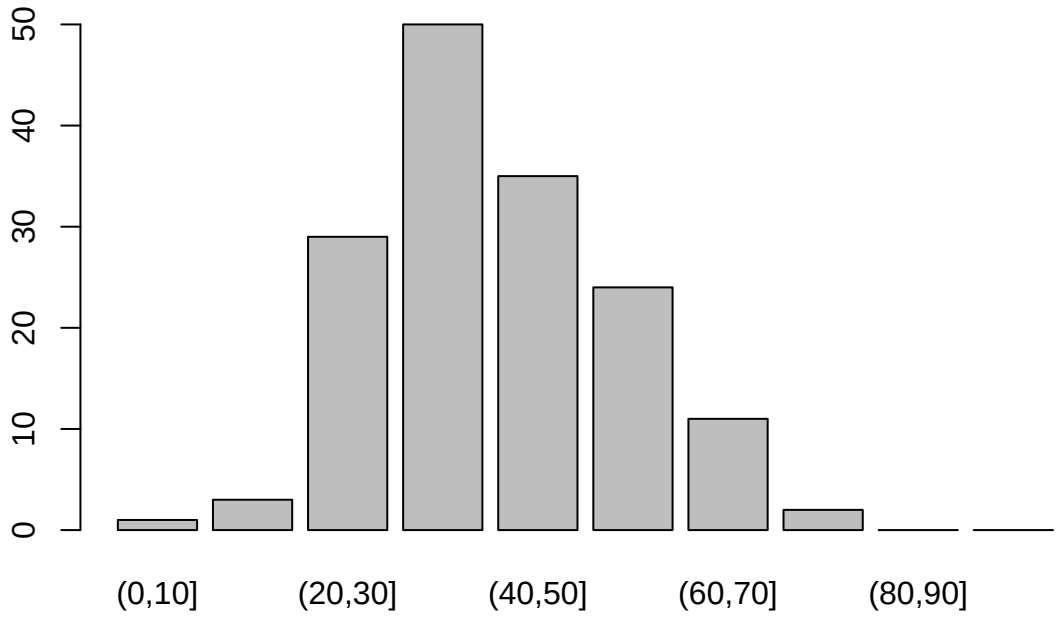


To know what we should expect for values, an internet search provided a gov site that says bilirubin should be in the range of 0.3-1.9mg/dL and albumin should be in the range of 3.4-5.4g/dL. Since we see some bilirubicin values around 8, this is startingly high. Additionally, there is only a single albumin record over six, which suggests this is also an outlier.

7

To bin age on units of decade, we can use 'with' and 'cut' to bin the data.

```
age_bin <- with(subD1, cut(AGE,c(0,10,20,30,40,50,60,70,80,90,100)))
plot(age_bin)
```



8

To aggregate on the binned ages and gender, we can use the age_bin from 7 and include the gender column to aggregate, and include the function that we want to use, being mean.

```
aggregate(subD1[,3:6],list(age_bin),mean)
```

```
##   Group.1 BILIRUBIN ALK_PHOSPHATE     SGOT ALBUMIN
## 1  (0,10]  0.700000         256.0  25.0000     4.2
## 2 (10,20]  1.400000         133.0 112.6667     3.6
## 3 (20,30]  1.189655            NA       NA      NA
## 4 (30,40]        NA            NA       NA      NA
## 5 (40,50]        NA            NA  96.4000      NA
## 6 (50,60]        NA            NA       NA      NA
## 7 (60,70]        NA            NA       NA      NA
## 8 (70,80]  0.850000         105.5  42.0000     3.7
```

The reason we see missing data in gender=2, is because there are only 16 people in the original data with this label

```
sum(hep_data[3]==1)
```

```
## [1] 139
```

```r
sum(hep_data[3]==2)
```

```
## [1] 16
```

9

Sorting data is quite easy in sql, so I will use sql to sort the data on bilirubicin

```r
library(sqldf)
```

```
## Loading required package: gsubfn
```

```
## Loading required package: proto
```

```
## Loading required package: RSQLite
```

```
## Loading required package: DBI
```

```r
bil_sort <- sqldf("SELECT * FROM subD1 ORDER BY BILIRUBIN ASC")
```

```
## Loading required package: tcltk
```

```r
head(bil_sort, 3); tail(bil_sort,3)
```

```
##   AGE SEX BILIRUBIN ALK_PHOSPHATE SGOT ALBUMIN
## 1  51   1        NA            NA   NA      NA
## 2  47   1        NA            NA   60      NA
## 3  34   1        NA            NA   86      NA
```

```
##     AGE SEX BILIRUBIN ALK_PHOSPHATE SGOT ALBUMIN
## 153  48   1       4.8           123  157     2.7
## 154  46   1       7.6            NA  242     3.3
## 155  31   1       8.0            NA  101     2.2
```

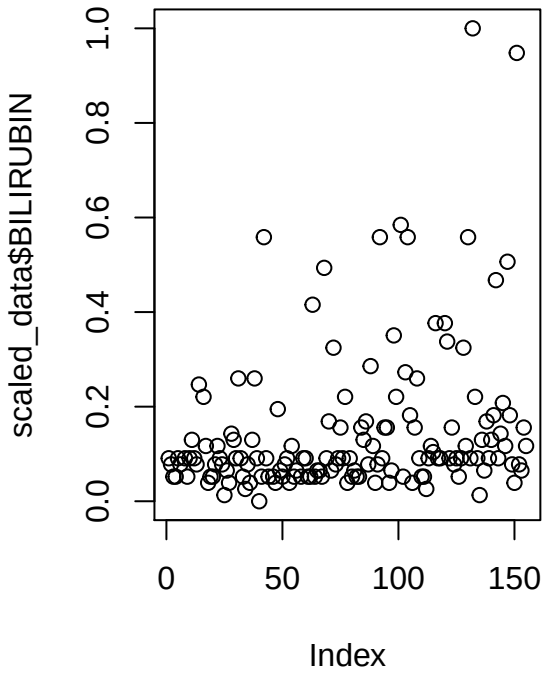
10

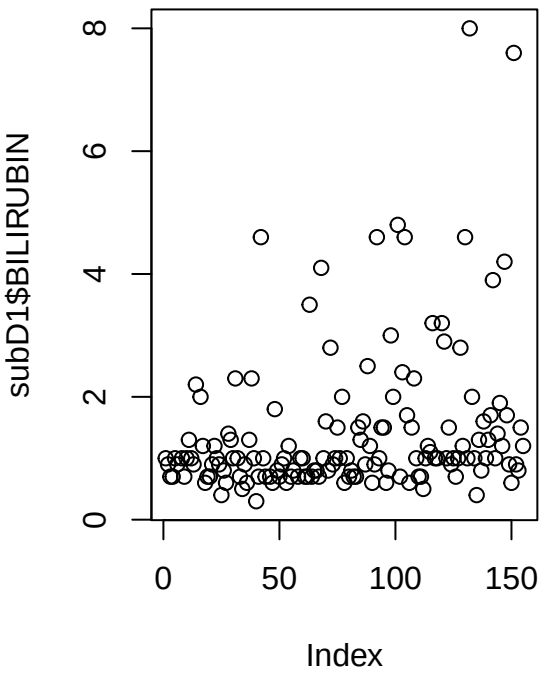
To standardize bilirubicin and albumin, we can use a function that rescales each column to a range between 0 and 1.

```r
scaled_data <- apply(subD1, MARGIN=2, FUN=function(X) (X-min(X,na.rm=TRUE))/diff(range(X,na.rm=TRUE)))
scaled_data <- data.frame(scaled_data)
```

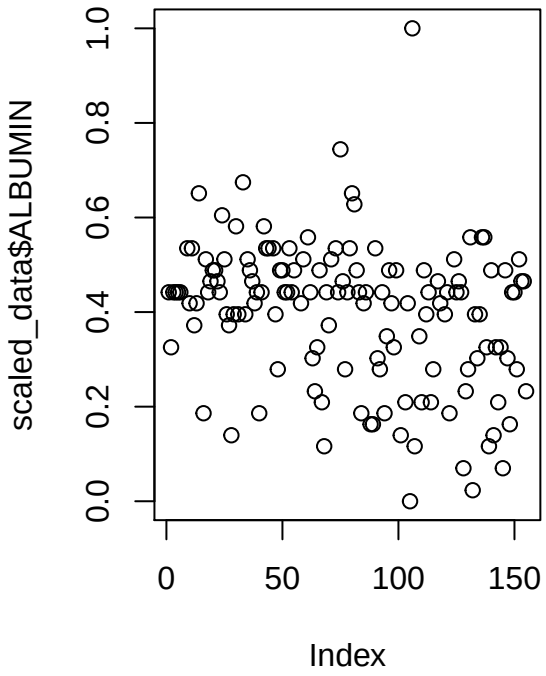
If we look at the scatterplots for each before and after, we see our scaling preserved the data.

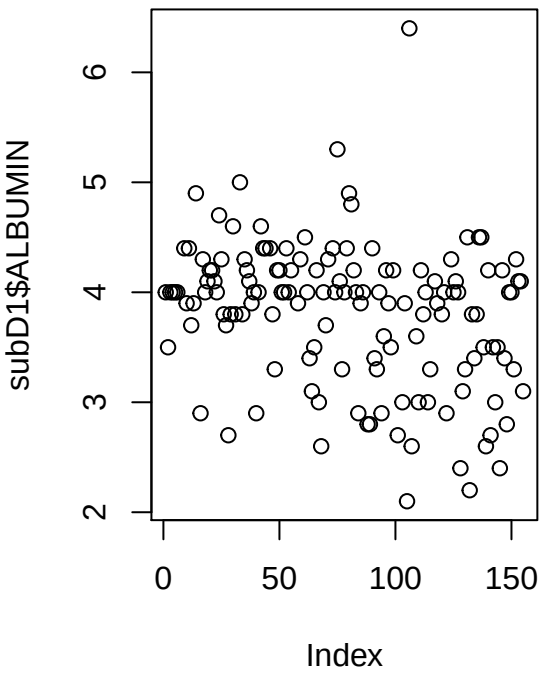
```r
par(mfrow=c(1,2))
plot(scaled_data$BILIRUBIN)
plot(subD1$BILIRUBIN)
```






```
par(mfrow=c(1,2))
plot(scaled_data$ALBUMIN)
plot(subD1$ALBUMIN)
```

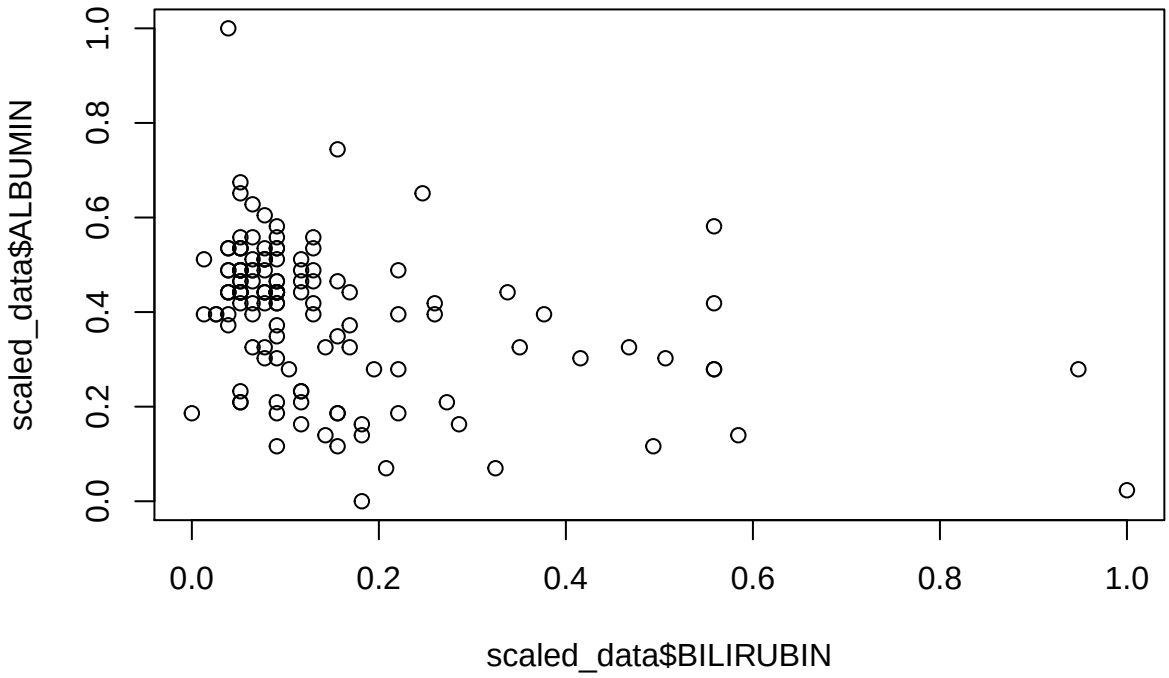





```
par(mfrow=c(1,1))
```

Now we can plot the two against each other

```
plot(scaled_data$BILIRUBIN, scaled_data$ALBUMIN)
```



11

To do PCA, we need to first get our complete cases for Bilirubin, ALK, SGOT, and ALBUMIN, and standardize them. Our scaled_data data frame already has it standardized, we just need to get complete cases. princomp is a package that performs PCA on the data frame it is given.

```
pca_complete_cases <- complete.cases(scaled_data)
data_for_pca <- scaled_data[pca_complete_cases,3:6]
data_pca <- princomp(data_for_pca)
data_pca
```

```
## Call:
## princomp(x = data_for_pca)
##
## Standard deviations:
##    Comp.1    Comp.2    Comp.3    Comp.4
## 0.2138704 0.1274491 0.1209581 0.1042220
##
##  4  variables and  120 observations.
```

The first principal component accounts for 21.4% of the variance.

12

We need to first subset a new data frame with age, sex, steroid, and antivirals (column numbers, 2,3,4,5), and adjust our previous data frame subD1 to only have complete cases.

```
viral_frame <- hep_data[,c(2,3,4,5)]
```

```
blood_complete <- complete.cases(subD1)
blood_frame <- subD1[blood_complete,]
```

Now we can try merging it.

```
new_frame <- merge(blood_frame, viral_frame)
```

Lets look at the dimensions for what we have and had.

```
dim(viral_frame)
```

```
## [1] 155   4
```

```
dim(blood_frame)
```

```
## [1] 120   6
```

```
dim(new_frame)
```

```
## [1] 481   8
```

Our first data frame with steroids and antivirals had 4 columns and 155 rows; our complete cases with bilirubin has 6 columns and 120 rows; our new frame has 8 columns and 481 rows. The columns are easy to understand. There are two matching columns between the two, age and sex. So we go from 4+6-2= 8 columns. I'm guessing since our keys for joining are not unique and don't line up exactly, merging the two will find repeat columns to join and thus enlarge the size of our merged data.