# EDA - Assignment 9

*Aaron Niskin*

*September 29, 2016*

Navigate to the website

https://archive.ics.uci.edu/ml/datasets/Hepatitis

Read the annotation provided. Download the associated hepatitis.data and the hepatitis.names files. Prepare an R Markdown file which documents the steps you take in carrying out the following

1. **Read the data into memory as a csv file**

```
data_url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/hepatitis/hepatitis.data"
data_file <- "data.csv"
download.file(data_url, data_file)
names_url <- "https://archive.ics.uci.edu/ml/machine-learning-databases/hepatitis/hepatitis.names"
names_file <- "names.csv"
download.file(names_url, names_file)
```

I do that at first so that every time I compile this PDF it won't try to download the files again and again. So now we want to read these files into memory. But before we do that, we're going to want to get rid of those pesky "?" NA values. So let's replace all of those using our favorite bash stream editing program, "sed"!

```
cat data.csv | sed 's/\?/NA/g' > data_na.csv
```

Now when we import the data into R, it should identify the types far more easily.

```
hep_data <- read.csv("data_na.csv", header=FALSE)
str(hep_data)
```

```
## 'data.frame':    155 obs. of  20 variables:
##  $ V1 : int  2 2 2 2 2 2 1 2 2 2 ...
##  $ V2 : int  30 50 78 31 34 34 51 23 39 30 ...
##  $ V3 : int  2 1 1 1 1 1 1 1 1 1 ...
##  $ V4 : int  1 1 2 NA 2 2 1 2 2 2 ...
##  $ V5 : int  2 2 2 1 2 2 2 2 2 2 ...
##  $ V6 : int  2 1 1 2 2 2 1 2 1 2 ...
##  $ V7 : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ V8 : int  2 2 2 2 2 2 1 2 2 2 ...
##  $ V9 : int  1 1 2 2 2 2 2 2 2 2 ...
##  $ V10: int  2 2 2 2 2 2 2 2 1 2 ...
##  $ V11: int  2 2 2 2 2 2 1 2 2 2 ...
##  $ V12: int  2 2 2 2 2 2 1 2 2 2 ...
##  $ V13: int  2 2 2 2 2 2 2 2 2 2 ...
##  $ V14: int  2 2 2 2 2 2 2 2 2 2 ...
##  $ V15: num  1 0.9 0.7 0.7 1 0.9 NA 1 0.7 1 ...
##  $ V16: int  85 135 96 46 NA 95 NA NA NA NA ...
##  $ V17: int  18 42 32 52 200 28 NA NA 48 120 ...
##  $ V18: num  4 3.5 4 4 4 4 NA NA 4.4 3.9 ...
##  $ V19: int  NA NA NA 80 NA 75 NA NA NA NA ...
##  $ V20: int  1 1 1 1 1 1 1 1 1 1 ...
```

2. **Name the features as described in the names file**

   So, first we check out that "names.txt" file. Then we notice section 7 seems to be describing the names of each of the columns. But then section 9 says,

   9. Class Distribution:

      DIE: 32

      LIVE: 123

      So if we were unsure of whether our names are listed in ascending or descending order, we can check by confirming that there are indeed 32 deaths and 123 live records. Unfortunately, if you check out column 1, the data is binary {1,2}, and not labeled as `live` or `die`. But we can count them. If there is a 32/123 split, we can be ralatively assured that the data is presented in ascending order (especially since this is the most logical way to present it anyway).

```
sum(hep_data$V1 == 1)
```

```
## [1] 32
```

```
sum(hep_data$V1 == 2)
```

```
## [1] 123
```

   This also gave us the beneifit of identifying that 1 corresponds with "die" and 2 with "live". Now since there are 20 different names here, and I'm pretty lazy, let's see if we can get this done programatically. First we're going to want to cat lines 30 - 50 into sed, then use sed to grab just the names, then feed that into a new file. And of course it turns out, as most things of this nature do, it would have been a little easier if I had just done it directly in a text editor. But this way was far more fun.

```
head -n -41 names.txt | tail -n -21 | sed -r '/^\ *--.*$/d;s/^[0-9 \.]+([a-zA-Z ]+)\:.*$/\1/g;' > r
```

   Now we're just about ready to read that file in!

```
names(hep_data) <- lapply(strsplit(tolower(readLines("namesJust.txt")), "\n"), as.character)
```

3. **Write the result to memory as a csv file**

```
write.csv(hep_data, "hep_data.csv", row.names = FALSE)
```

Continuing your document, addressing the following questions

4. **How many complete cases are there?**

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

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

5. **Subsetting the data on Age, Sex, Bilirubin, ALK, SGOT and Albumin, compute the number of missing values for the Bilirubin feature. Convert the last four features to numeric values. How many complete cases are there for the subsetted frame?**

```r
str(hep_data)
```

```
## 'data.frame':    155 obs. of  20 variables:
##  $ class          : int  2 2 2 2 2 2 1 2 2 2 ...
##  $ age            : int  30 50 78 31 34 34 51 23 39 30 ...
##  $ sex            : int  2 1 1 1 1 1 1 1 1 1 ...
##  $ steroid        : int  1 1 2 NA 2 2 1 2 2 2 ...
##  $ antivirals     : int  2 2 2 1 2 2 2 2 2 2 ...
##  $ fatigue        : int  2 1 1 2 2 2 1 2 1 2 ...
##  $ malaise        : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ anorexia       : int  2 2 2 2 2 2 1 2 2 2 ...
##  $ liver big      : int  1 1 2 2 2 2 2 2 2 2 ...
##  $ liver firm     : int  2 2 2 2 2 2 2 2 1 2 ...
##  $ spleen palpable: int  2 2 2 2 2 2 1 2 2 2 ...
##  $ spiders        : int  2 2 2 2 2 2 1 2 2 2 ...
##  $ ascites        : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ varices        : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ bilirubin      : num  1 0.9 0.7 0.7 1 0.9 NA 1 0.7 1 ...
##  $ alk phosphate  : int  85 135 96 46 NA 95 NA NA NA NA ...
##  $ sgot           : int  18 42 32 52 200 28 NA NA 48 120 ...
##  $ albumin        : num  4 3.5 4 4 4 4 NA NA 4.4 3.9 ...
##  $ protime        : int  NA NA NA 80 NA 75 NA NA NA NA ...
##  $ histology      : int  1 1 1 1 1 1 1 1 1 1 ...
```

```r
tmp <- as.data.frame(cbind(hep_data[,c("age", "sex")], sapply(
      hep_data[, c("bilirubin", "alk phosphate", "sgot", "albumin")],
        as.numeric)))
str(tmp)
```

```
## 'data.frame':    155 obs. of  6 variables:
##  $ age          : int  30 50 78 31 34 34 51 23 39 30 ...
##  $ sex          : int  2 1 1 1 1 1 1 1 1 1 ...
##  $ bilirubin    : num  1 0.9 0.7 0.7 1 0.9 NA 1 0.7 1 ...
##  $ alk phosphate: num  85 135 96 46 NA 95 NA NA NA NA ...
##  $ sgot         : num  18 42 32 52 200 28 NA NA 48 120 ...
##  $ albumin      : num  4 3.5 4 4 4 4 NA NA 4.4 3.9 ...
```

```r
sum(complete.cases(tmp))
```

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

```r
sum(complete.cases(tmp[,"bilirubin"]))
```

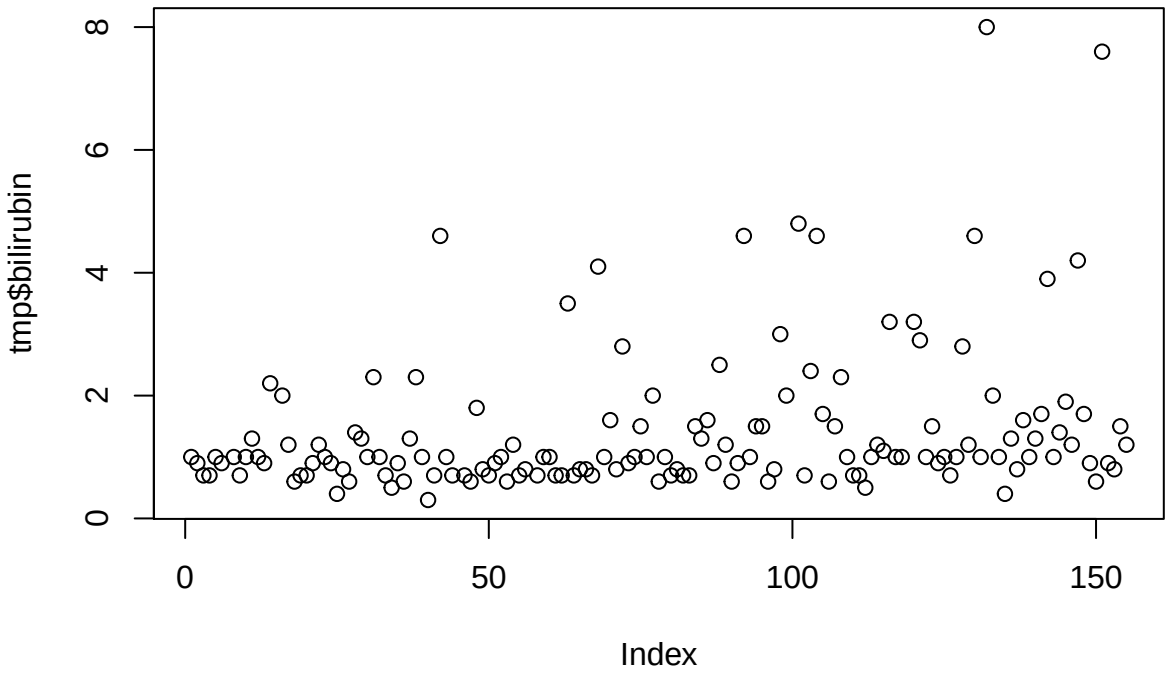
```
## [1] 149
```

6. **Are there any outliers in the Bilirubin and Albumin entries?**

```r
quantile(tmp$bilirubin, na.rm = TRUE)
```

```
##   0%  25%  50%  75% 100%
##  0.3  0.7  1.0  1.5  8.0
```

```
plot(tmp$bilirubin)
```




There do seem to be outliers, and there is a note about bilirubin specifically, but that seems to be mainly concerning the data's continuity. After doing a bit of research, it seems very likely that the bilirubin data corresponds to something called "direct bilirubin" measured in units of mmol. I owuld need more information to actually assess with any validity of this data with any measure of accuracy, but according to MedicalHealthTests.com, if the previous assumption/semi-conclusion is true, then 5.1 is on the high side and our two data points around 8 are way too high.

7. **Bin the age variables in units of decades**

   So just to make this easier, let's do this...

```
max(tmp$age)
```

```
## [1] 78
```

```
min(tmp$age)
```

```
## [1] 7
```

   So we can bin our data from 0 to 80.

```
age_groups <- cut(tmp$age, breaks = seq(0, 80, 10))
tmp[,"age"] <- age_groups
```

8. **Aggregate the data to obtain mean readings for the last 4 variables as a function of sex and age, with age as a binned factor.**

   By last four variables, I'm going to assume that you mean, "BILIRUBIN", "ALK PHOSPHATE", "SGOT", "ALBUMIN".

```
agg <- aggregate(tmp[,3:6], by=list(age=tmp$age, sex=tmp$sex), FUN=mean, na.rm=TRUE)
agg
```

```
##        age sex bilirubin alk phosphate      sgot  albumin
## 1   (0,10]   1 0.7000000      256.0000  25.00000 4.200000
## 2  (10,20]   1 0.9500000      124.5000 135.00000 3.450000
## 3  (20,30]   1 1.2458333      105.1875  77.73913 4.218182
## 4  (30,40]   1 1.2086957      100.1000  77.48936 3.847727
## 5  (40,50]   1 1.8580645      102.2593  97.78125 3.578571
## 6  (50,60]   1 1.9150000      102.3750  93.04762 3.700000
## 7  (60,70]   1 1.0428571      104.0000 111.00000 3.700000
## 8  (70,80]   1 0.8500000      105.5000  42.00000 3.700000
## 9  (10,20]   2 2.3000000      150.0000  68.00000 3.900000
## 10 (20,30]   2 0.9200000      100.8000 101.00000 3.920000
## 11 (30,40]   2 0.6500000       50.0000  24.00000 4.050000
## 12 (40,50]   2 0.8666667      132.0000  81.66667 4.200000
## 13 (50,60]   2 1.4500000      128.0000  37.00000 3.400000
## 14 (60,70]   2 2.0000000      146.3333 120.33333 3.400000
```

9. **Sort the data on the Bilirubin columns (ascending)**

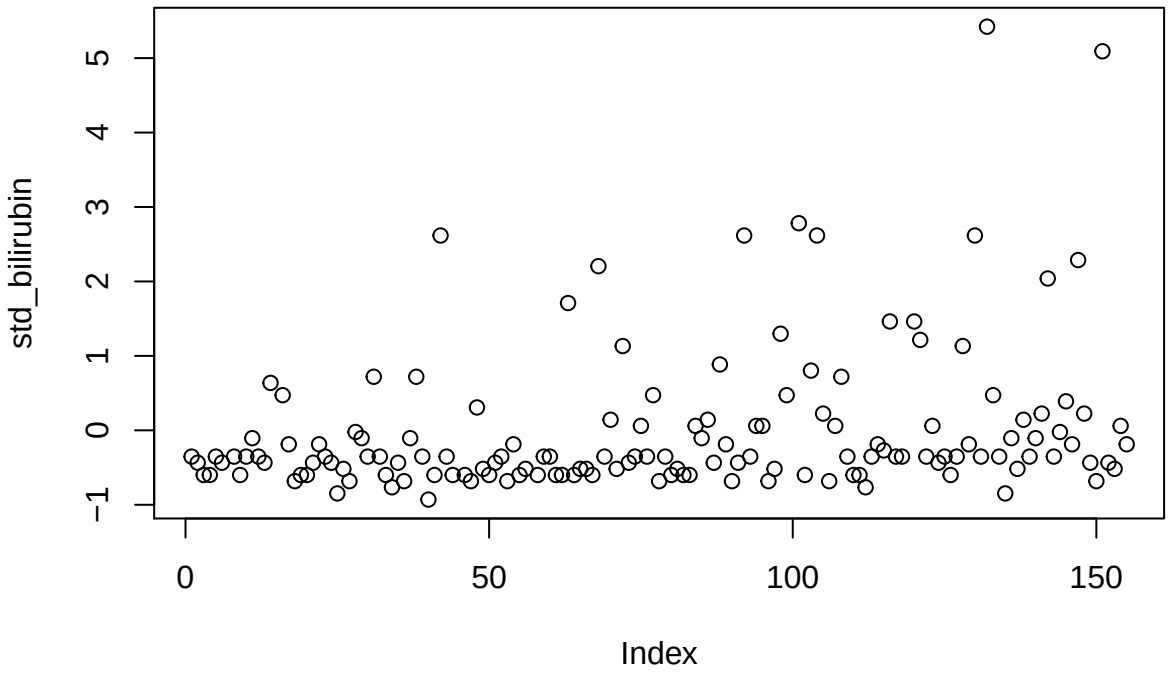
```
agg[order(agg$bilirubin),]
```

```
##        age sex bilirubin alk phosphate      sgot  albumin
## 11 (30,40]   2 0.6500000       50.0000  24.00000 4.050000
## 1   (0,10]   1 0.7000000      256.0000  25.00000 4.200000
## 8  (70,80]   1 0.8500000      105.5000  42.00000 3.700000
## 12 (40,50]   2 0.8666667      132.0000  81.66667 4.200000
## 10 (20,30]   2 0.9200000      100.8000 101.00000 3.920000
## 2  (10,20]   1 0.9500000      124.5000 135.00000 3.450000
## 7  (60,70]   1 1.0428571      104.0000 111.00000 3.700000
## 4  (30,40]   1 1.2086957      100.1000  77.48936 3.847727
## 3  (20,30]   1 1.2458333      105.1875  77.73913 4.218182
## 13 (50,60]   2 1.4500000      128.0000  37.00000 3.400000
## 5  (40,50]   1 1.8580645      102.2593  97.78125 3.578571
## 6  (50,60]   1 1.9150000      102.3750  93.04762 3.700000
## 14 (60,70]   2 2.0000000      146.3333 120.33333 3.400000
## 9  (10,20]   2 2.3000000      150.0000  68.00000 3.900000
```

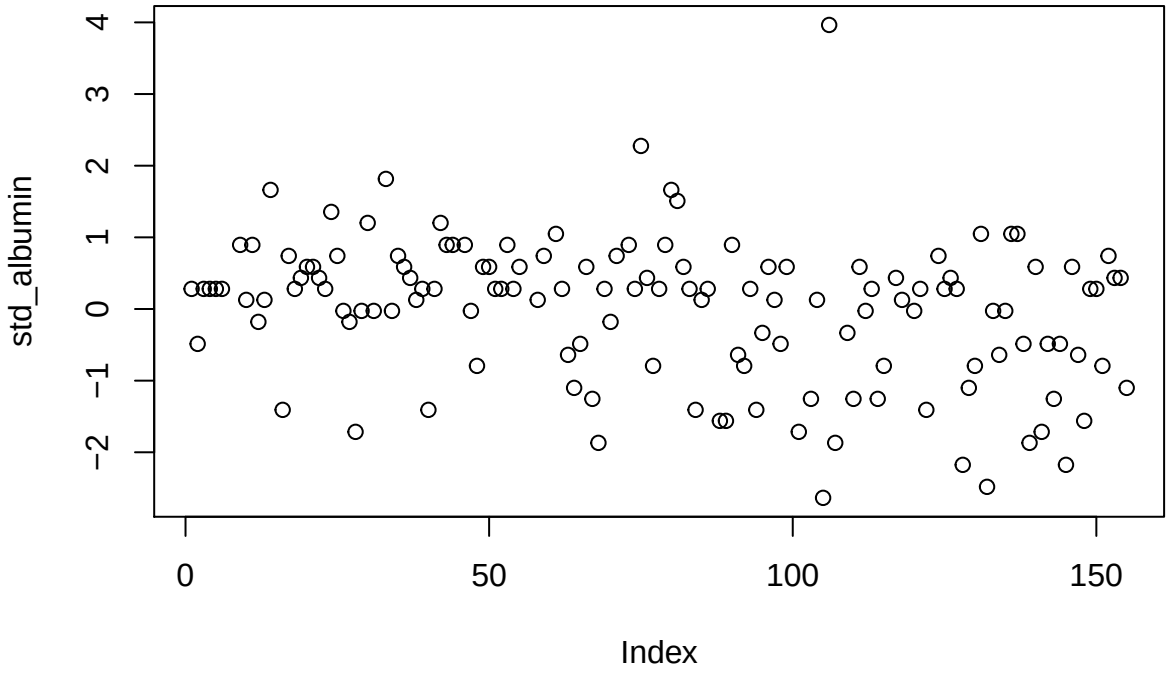
10. **Standardize Bilirubin and Albumin and plot the outcome as a scatterplot.**

    So this depends on what you mean by standardize. If you mean something like a z-score, then we can compute it thusly:

```
std_bilirubin <- scale(tmp$bilirubin)
std_albumin   <- scale(tmp$albumin)
plot(std_bilirubin)
```




```
plot(std_albumin)
```




11. **Consider the data frame consisting of the complete cases for the variables Bilirubin, ALK, SGOT and Albumin. What fraction of the variance does the first principal component account for?**

```
tmp2 <- tmp[,c("bilirubin", "alk phosphate", "sgot", "albumin")]
a <- princomp(tmp2[complete.cases(tmp2),])
a
```

```
## Call:
```

```
## princomp(x = tmp2[complete.cases(tmp2), ])
##
## Standard deviations:
##     Comp.1     Comp.2     Comp.3     Comp.4
## 81.8700900 49.9311803  0.9133541  0.5196922
##
##  4  variables and  120 observations.
```

```
a["loadings"]
```

```
## $loadings
##
## Loadings:
##               Comp.1 Comp.2 Comp.3 Comp.4
## bilirubin                    0.984  0.179
## alk phosphate -0.243  0.970
## sgot          -0.970 -0.243
## albumin                     -0.179  0.984
##
##                Comp.1 Comp.2 Comp.3 Comp.4
## SS loadings      1.00   1.00   1.00   1.00
## Proportion Var   0.25   0.25   0.25   0.25
## Cumulative Var   0.25   0.50   0.75   1.00
```

So about 25%.

12. **Subsetting the data on Age, Sex, Steroid and Antivirals columns and join the resulting data frame with the data frame of complete cases for Age, Sex, Bilirubin, ALK, SGOT and Albumin. What are the dimensions of the resulting frame?**

```
tmpA <- hep_data[,c("sex", "age", "steroid", "antivirals")]
tmpB <- hep_data[,c("sex", "age", "bilirubin", "alk phosphate", "sgot", "albumin")]
tmpC <- merge.data.frame(tmpA[complete.cases(tmpA),],
                         tmpB[complete.cases(tmpB),])
dim(tmpC)
```

```
## [1] 478   8
```