StatMethodsHW1

Joshua Burkhart

January 8, 2016

BMI 651: HW1

1. Develop an annotated R script that utilizes key exploratory data analysis techniques to identify if there are any issues in the data or potential problems for analysis.

See the source .Rmd document.

2. Provide any key output (figures or tables).

See below output.

Load data

```
data <- read.csv("~/SoftwareProjects/StatisticalMethodsInCompBio/HW1/hw1data.txt",sep="")

# add column names to data frame
names(data) <- c(
  "numPreg",     #Number of pregnancies
  "glucTol",     #Glucose tolerance (Plasma glucose concentration @ 2 hrs)
  "diasBldPrs", #Diastolic blood pressure (mm Hg)
  "bdyFat",      #Body Fat: Triceps skin fold thickness (mm)
  "insln",       #Insulin: 2-Hour serum insulin (mu U/ml)
  "bmi",         #Body mass index (weight in kg/(height in m)^2)
  "expGenInf",  #Expected Genetic Influence of affected and unaffected relatives
                 #on subject's eventual risk
  "age",         #Age (years)
  "tstPosDbts") #Class variable (0 or 1) where 1 is interpreted as "tested positive
                 #for diabetes"

#copy to matrix
data_matrix <- as.matrix(data)
```

Data types?

```
str(data)
```

```
'data.frame':   767 obs. of  9 variables:
 $ numPreg   : int  1 8 1 0 5 3 10 2 8 4 ...
 $ glucTol   : int  85 183 89 137 116 78 115 197 125 110 ...
 $ diasBldPrs: int  66 64 66 40 74 50 0 70 96 92 ...
```

```
 $ bdyFat    : int  29 0 23 35 0 32 0 45 0 0 ...
 $ insln     : int  0 0 94 168 0 88 0 543 0 0 ...
 $ bmi       : num  26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 37.6 ...
 $ expGenInf : num  0.351 0.672 0.167 2.288 0.201 ...
 $ age       : int  31 32 21 33 30 26 29 53 54 30 ...
 $ tstPosDbts: int  0 1 0 1 0 1 0 1 1 0 ...
```

tstPosDbts is listed as an int. We should change it to a factor to prevent accidental rescaling/etc.

```
data$tstPosDbts <- as.factor(data$tstPosDbts)
```

Columns same size?

```
for (i in 0:ncol(data)) {
    print(nrow(data[i]))
}
```

```
[1] 767
[1] 767
[1] 767
[1] 767
[1] 767
[1] 767
[1] 767
[1] 767
[1] 767
[1] 767
```

Missing data?

```
sapply(data, function(x) sum(is.na(x))) #[1]
```

```
   numPreg    glucTol diasBldPrs     bdyFat      insln        bmi
         0          0          0          0          0          0
 expGenInf        age tstPosDbts
         0          0          0
```

Ranges?

```
for (i in 1:8) {
    print(max(data[i]) - min(data[1]))
}
```

```
[1] 17
[1] 199
[1] 122
[1] 99
```

```
[1] 846
[1] 67.1
[1] 2.42
[1] 81
```

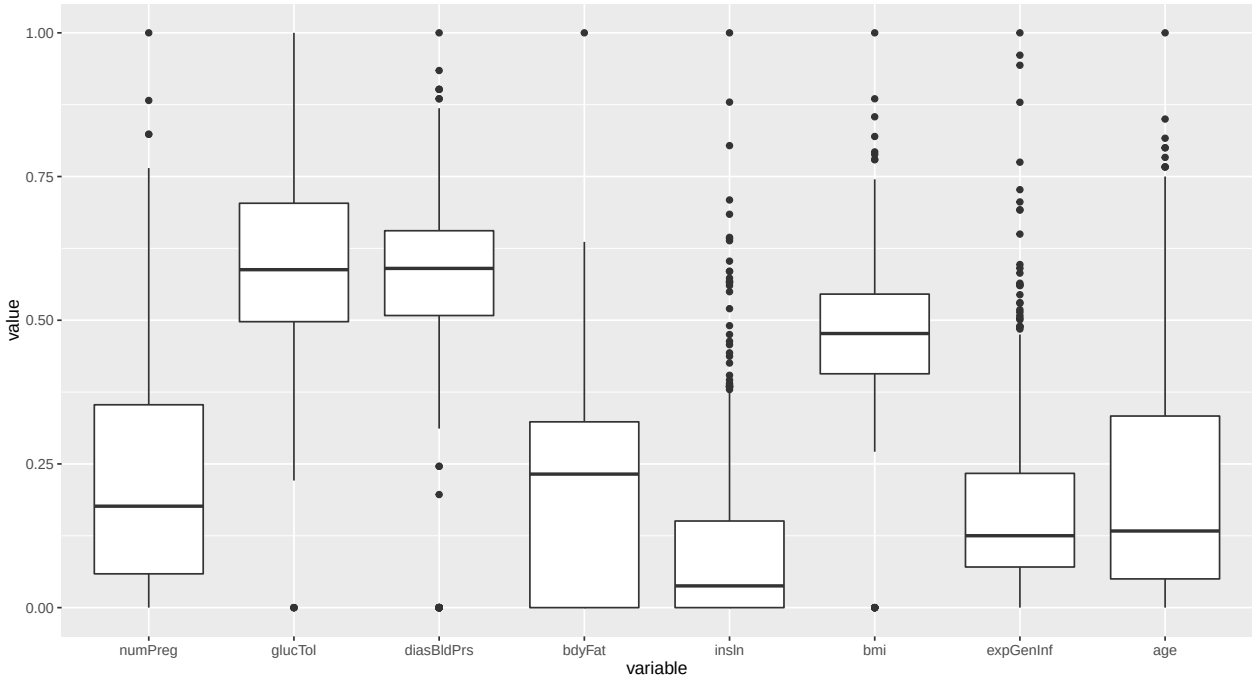
Ranges differ too widely. Data should be transformed onto singular scale to avoid delayed convergence of gradient descent or similar convex optimization algorithms.

```
for (i in 1:8) {
    data[i] <- rescale(data[i], to = c(0,1), from = range(data[i]))
    print(max(data[i]) - min(data[1]))
}
```

```
[1] 1
[1] 1
[1] 1
[1] 1
[1] 1
[1] 1
[1] 1
[1] 1
```

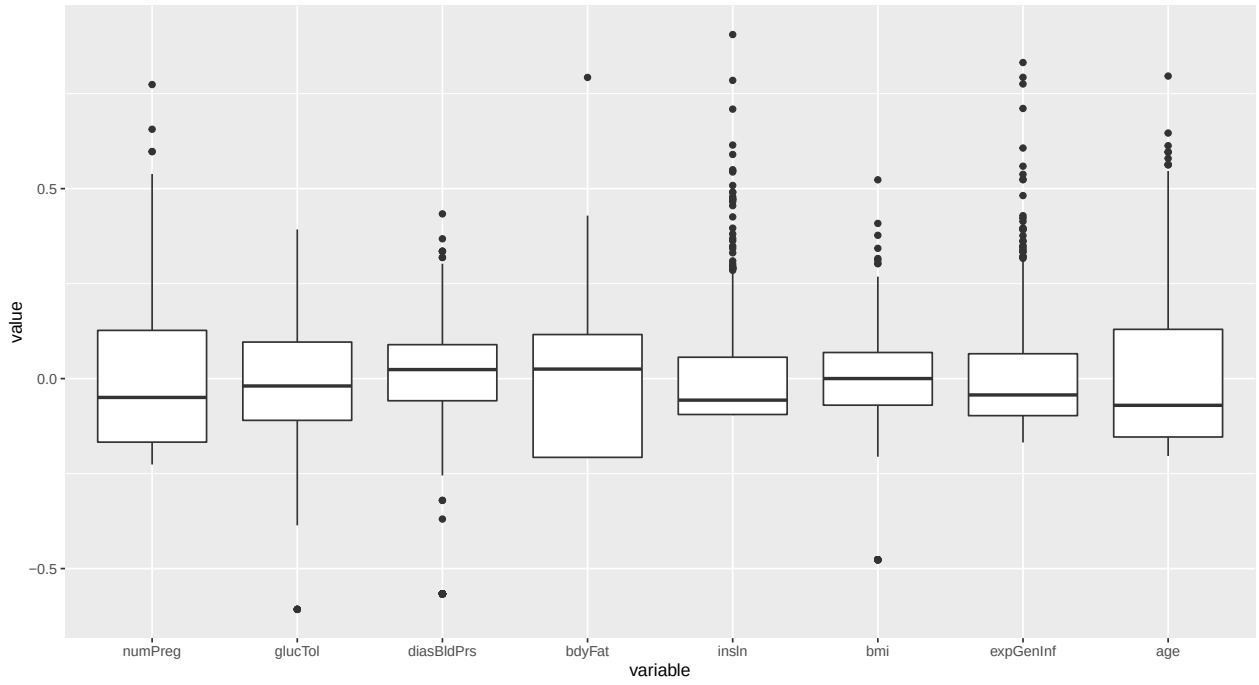
Distributions?

```
melt(data) %>% ggplot(aes(x=variable,y=value)) +
  geom_boxplot() #[2]
```



Distributions vary widely. We should recenter our data prior to further analysis.

```
for (i in 1:8) {
    data[i] <- scale(data[i], center=TRUE, scale=FALSE)
}
melt(data) %>% ggplot(aes(x=variable,y=value)) +
  geom_boxplot() #[2]
```



Rank deficient/singular?

```
data_inv <- ginv(data_matrix)
zapsmall(data_inv %*% data_matrix)
```

```
      numPreg glucTol diasBldPrs bdyFat insln bmi expGenInf age tstPosDbts
 [1,]       1       0          0      0     0   0         0   0          0
 [2,]       0       1          0      0     0   0         0   0          0
 [3,]       0       0          1      0     0   0         0   0          0
 [4,]       0       0          0      1     0   0         0   0          0
 [5,]       0       0          0      0     1   0         0   0          0
 [6,]       0       0          0      0     0   1         0   0          0
 [7,]       0       0          0      0     0   0         1   0          0
 [8,]       0       0          0      0     0   0         0   1          0
 [9,]       0       0          0      0     0   0         0   0          1
```

Mutual Information?

```
maxInfPos <- mutinformation(discretize(data[9]),discretize(data[9]))
for (i in 1:ncol(data)) {
    print(
```

```
      sprintf("%s: %f",
              names(data)[i],
              mutinformation(discretize(data[i]),discretize(data[9]))/maxInfPos))
}
```

```
[1] "numPreg: 0.058112"
[1] "glucTol: 0.194859"
[1] "diasBldPrs: 0.024818"
[1] "bdyFat: 0.044958"
[1] "insln: 0.068801"
[1] "bmi: 0.100684"
[1] "expGenInf: 0.031511"
[1] "age: 0.089489"
[1] "tstPosDbts: 1.000000"
```

Column 2, Glucose tolerance (Plasma glucose concentration @ 2 hrs), appears to be the best individual predictor and column 3, Diastolic blood pressure (mm Hg), appears to be the worst. This may be useful in narrowing features later on.

Principal Component Analysis?

```
#[3]

log_features <- log(data_matrix[,1:8])
log_features[is.infinite(log_features)] <- -99999
data_matrix[,9][data_matrix[,9] == 0] <- "FALSE"
data_matrix[,9][data_matrix[,9] == 1] <- "TRUE"

class <- as.factor(data_matrix[,9])
data_pc <- prcomp(log_features,center=TRUE,scale. = TRUE)
print(data_pc)
```

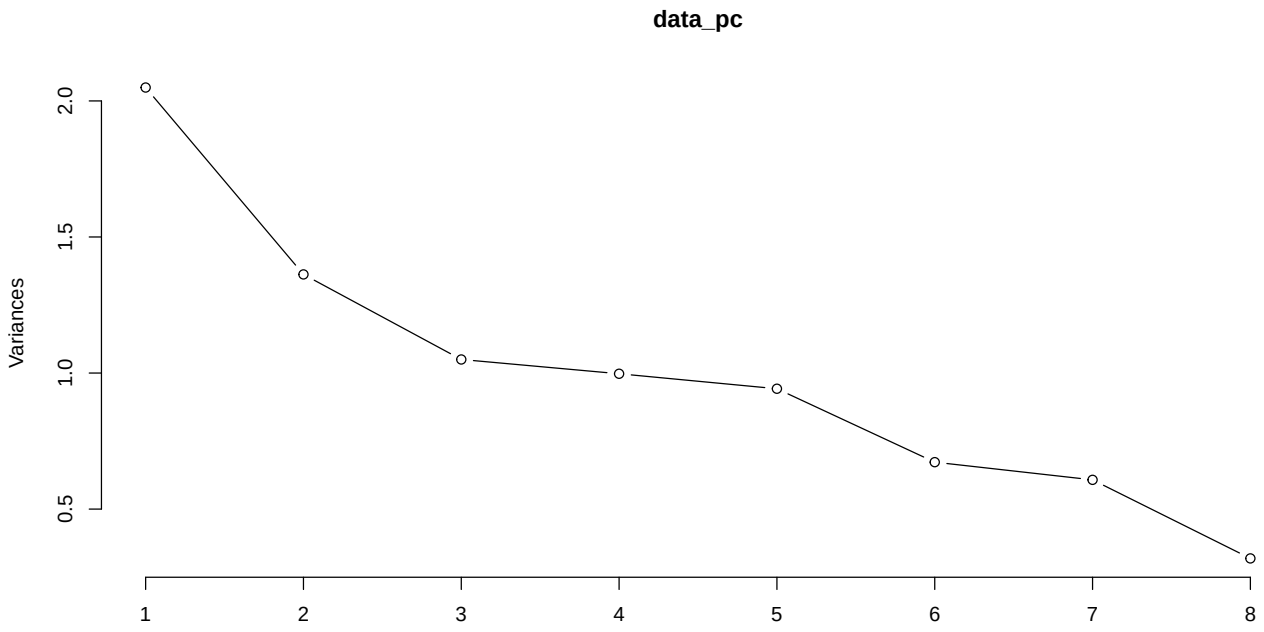
```
Standard deviations:
[1] 1.4315012 1.1672455 1.0246092 0.9986878 0.9707405 0.8200528 0.7794484
[8] 0.5646038

Rotation:
                  PC1         PC2         PC3        PC4         PC5
numPreg    -0.01537302 -0.51034683  0.04339613  0.4889199  0.51315134
glucTol    -0.01371083  0.03221469  0.66970431 -0.5810247  0.43773130
diasBldPrs  0.38748292 -0.42592778 -0.15791232 -0.2364834 -0.05549539
bdyFat      0.59688191  0.10699275 -0.01939787  0.1650911  0.14965664
insln       0.57276822  0.15052666  0.13298443  0.1483135  0.22615009
bmi         0.25866501 -0.44982709 -0.21833360 -0.4698654 -0.25486114
expGenInf   0.23151125  0.01899837  0.58438490  0.2696473 -0.62259894
age        -0.21117808 -0.56593532  0.34269550  0.1571882 -0.12594704
                  PC6         PC7          PC8
numPreg     0.45487984  0.16649341  0.020588873
glucTol     0.07050806  0.07468597  0.102579779
diasBldPrs -0.36933069  0.65984242 -0.112716048
bdyFat     -0.16884093 -0.18574626  0.720588046
```

```
insln      -0.09685796 -0.30477450 -0.675413729
bmi         0.47050777 -0.41938679  0.009373311
expGenInf   0.30025989  0.23170916  0.018753044
age        -0.54980496 -0.41754801  0.021867935
```

```r
summary(data_pc)
```

```
Importance of components:
                          PC1    PC2    PC3    PC4    PC5     PC6     PC7
Standard deviation     1.4315 1.1672 1.0246 0.9987 0.9707 0.82005 0.77945
Proportion of Variance 0.2561 0.1703 0.1312 0.1247 0.1178 0.08406 0.07594
Cumulative Proportion  0.2561 0.4265 0.5577 0.6824 0.8002 0.88421 0.96015
                           PC8
Standard deviation     0.56460
Proportion of Variance 0.03985
Cumulative Proportion  1.00000
```

```r
plot(data_pc,type="l")
```



From the summary() printout above, we see that 97.07% of the variance is explained with the first 5 PC's.

```r
ggbiplot(data_pc, choices = 1:2, obs.scale = 1, var.scale = 1,
              groups = class, ellipse = TRUE,
              circle = TRUE) +
  scale_color_discrete(name = "tstPosDbts") +
  theme(legend.direction = 'horizontal',
               legend.position = 'top')
```

The above plot is a visualization of the first two principal components.

3. Briefly summarize any issues with the data, how you have addressed them (transforms etc) and your recommendations for further downstream analysis. Do you have any concerns about using the attribute data to predict class outcome (key here is to translated biological question->hypothesis->statistical test)?

Technical Assessment

The glaring issues with this data are the differing ranges, distributions, and amounts of mutual information among the features. Mutual information between the test result, tstPosDbts, and the features: glucTol, bmi, and age are the highest, indicating a simple risk classifier may be built using only these three features. Following a log transform, recentering, and rescaling, 5 principal components (PC's) are shown to account for over 97% of the variability in the data. These 5 PC's could then be fed into a more sophisticated learning algorithm such as an Artificial Neural Network (ANN) or a Support Vector Machine (SVM).

Discussion

The biological question, "How can we predict diabetes with the provided features?", can be considered as more than one technical question thus, a single best data transformation is not possible. On the one hand, if our goal is to design a system by which clinicians can memorize a system to quickly assess the diabetes risk of a patient based on a few factors, perhaps a simple classifier, based on only one to three features would be desired. Those features would be the ones with the most mutual information: glucTol, bmi, and age. If, on the other hand, we wanted to train a machine learning algorithm to best predict diabetes likelyhood, we should use the PC's (produced above in the data_pc object) to train a classifier. We can likely get away with using only the first five PC's, as they account for most of the variance in the data.

References

[1] http://stackoverflow.com/questions/8317231/elegant-way-to-report-missing-values-in-a-data-frame
[2] http://stackoverflow.com/questions/15071334/boxplot-of-table-using-ggplot2
[3] http://www.r-bloggers.com/computing-and-visualizing-pca-in-r/