Stat 135: Lab Project

Zubair Marediya, Temi Lal, Charles Thorson

Tuesday, May 05, 2015

Before beginning any part of this project, let's import our data.

```
data = read.csv("diabetes.csv")
```

The last 16 patients were held out for testing, so we will remove them from our data.

```
data = data[1:359, ]
```

Part 2: Accessing Data, Visualization and Summarization

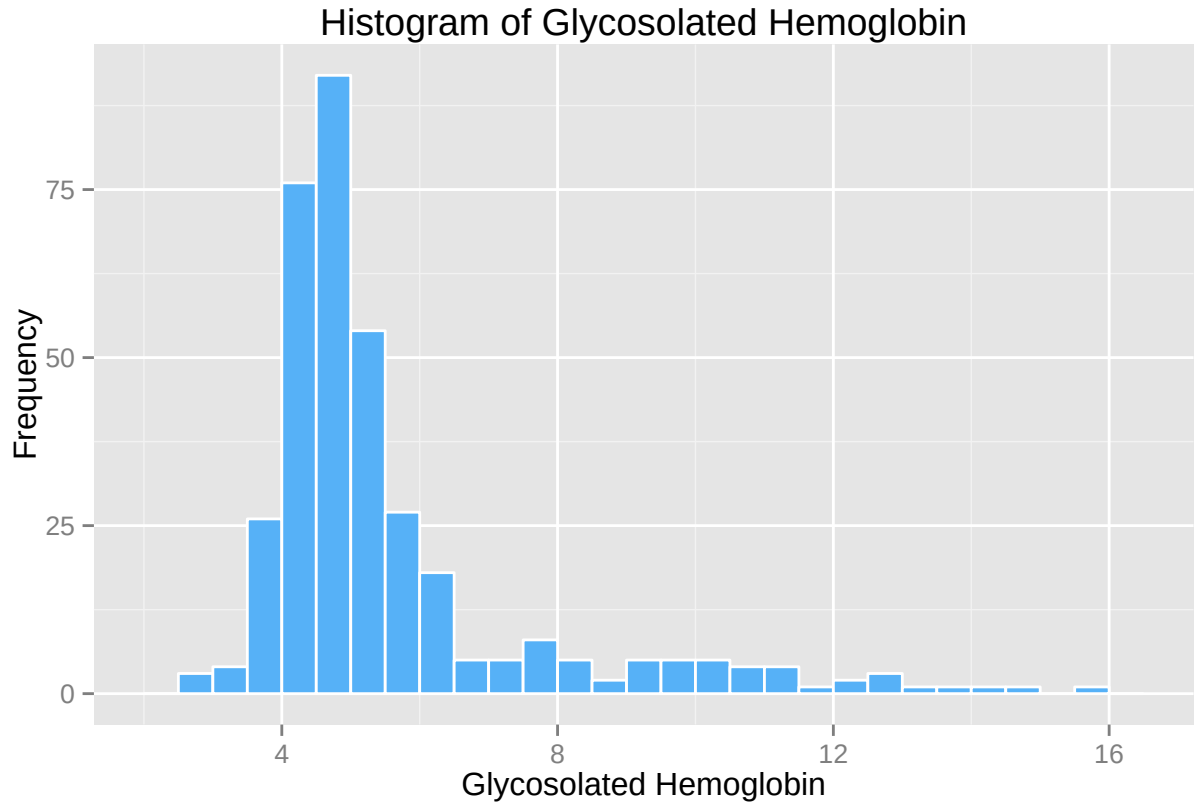
Question 1

Let's create the histogram first. We will use ggplot2 for all visuals.

```
glyhb = data$glyhb
require("ggplot2")
```

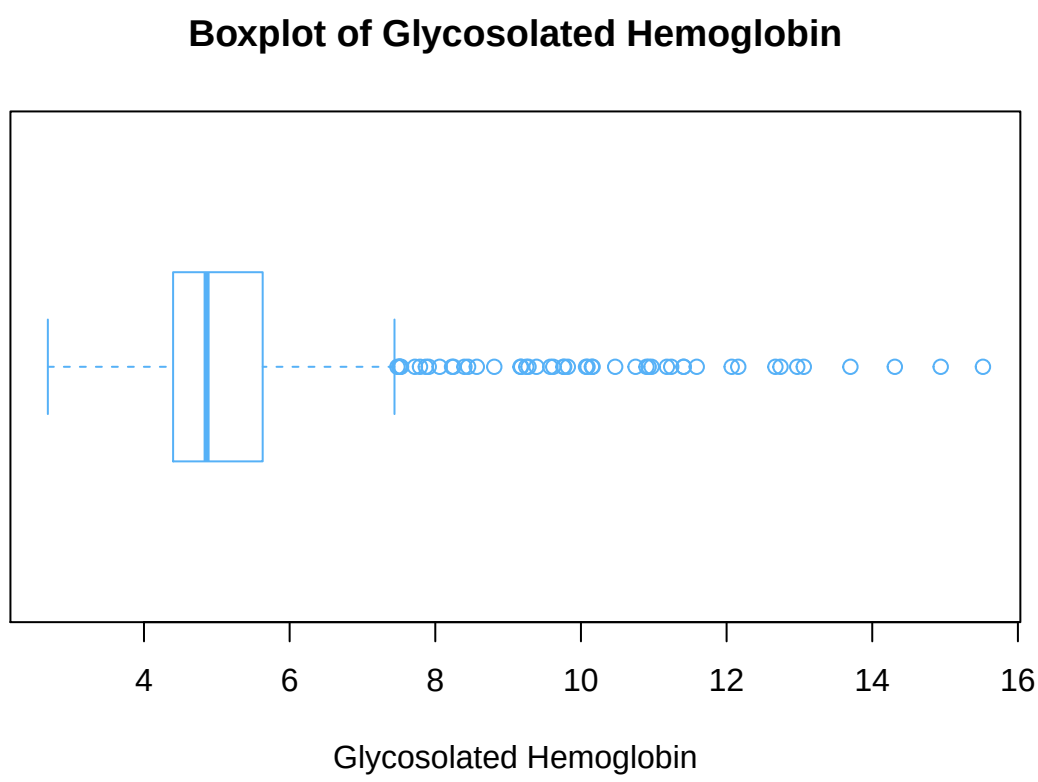
```
## Loading required package: ggplot2
```

```
ggplot(data) + geom_histogram(aes(x = glyhb), binwidth = 0.5, col = "white", fill = "#56B1F7") +
  ggtitle("Histogram of Glycosolated Hemoglobin") +
  xlab("Glycosolated Hemoglobin") + ylab("Frequency")
```



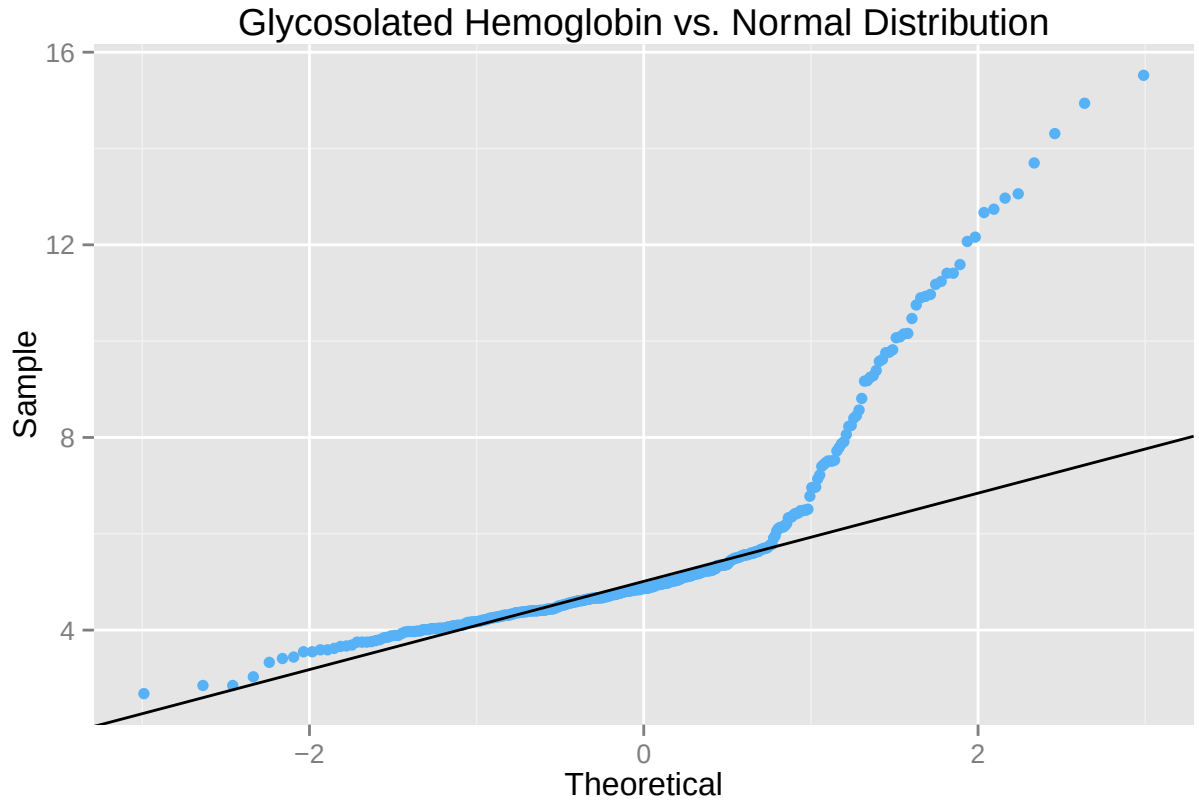
Now let's create the boxplot.

```
boxplot(glyhb, horizontal = TRUE, col = "white", border = "#56B1F7",
        xlab = "Glycosolated Hemoglobin",
        main = "Boxplot of Glycosolated Hemoglobin")
```



Lastly let's create the qqplot.

```
exp.sample = rnorm(n = 10000)
y1y2 = quantile(glyhb, probs = c(0.25, 0.75))
x1x2 = quantile(exp.sample, probs = c(0.25, 0.75))
slope = (y1y2[2] - y1y2[1])/(x1x2[2] - x1x2[1])
intercept = slope * (0 - x1x2[1]) + y1y2[1]
ggplot(data, aes(sample = glyhb)) + stat_qq(col = "#56B1F7") +
  ggtitle("Glycosolated Hemoglobin vs. Normal Distribution") +
  xlab("Theoretical") +
  ylab("Sample") +
  geom_abline(intercept = intercept, slope = slope)
```

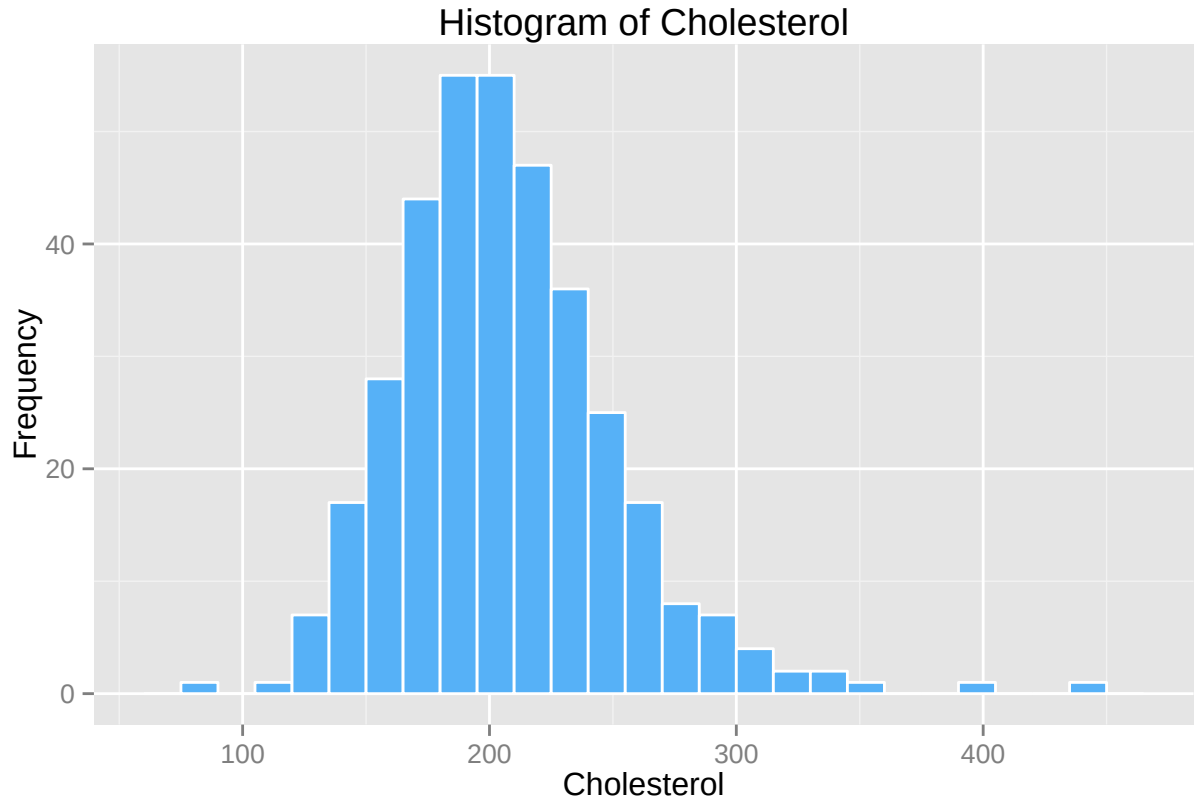


From seeing all three visuals above, we see that the distribution is skewed to the right with a heavy tail. Thus, it does not follow a Gaussian distribution. The boxplot shows us that there are many outliers. The center seems to be around 5, and the bulk of the data falls between 4 and 6. Our qq-plot follows a normal well until we hit the value of 1 on the Theoretical axis. At that point, our outliers kick in, and we do not follow a Gaussian distribution anymore. It seems there are a few people with very high levels of Glycosolated Hemoglobin. We might suspect that these people are in poor health and have Type II diabetes.

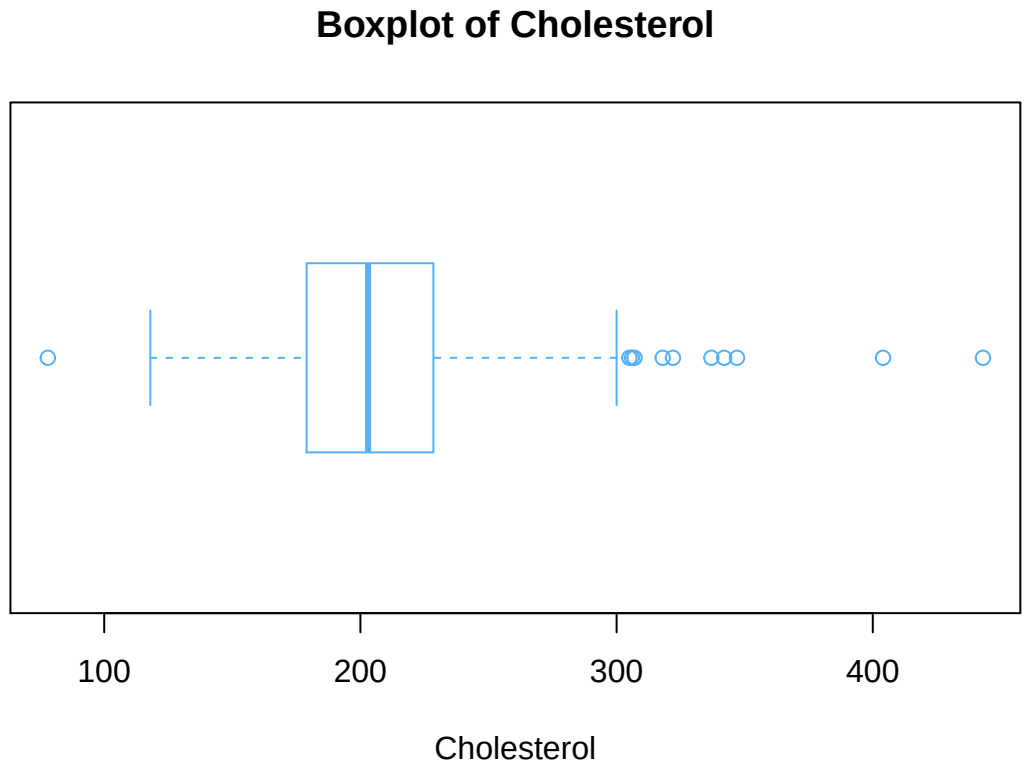
Question 2

Now let's repeat this for cholesterol.

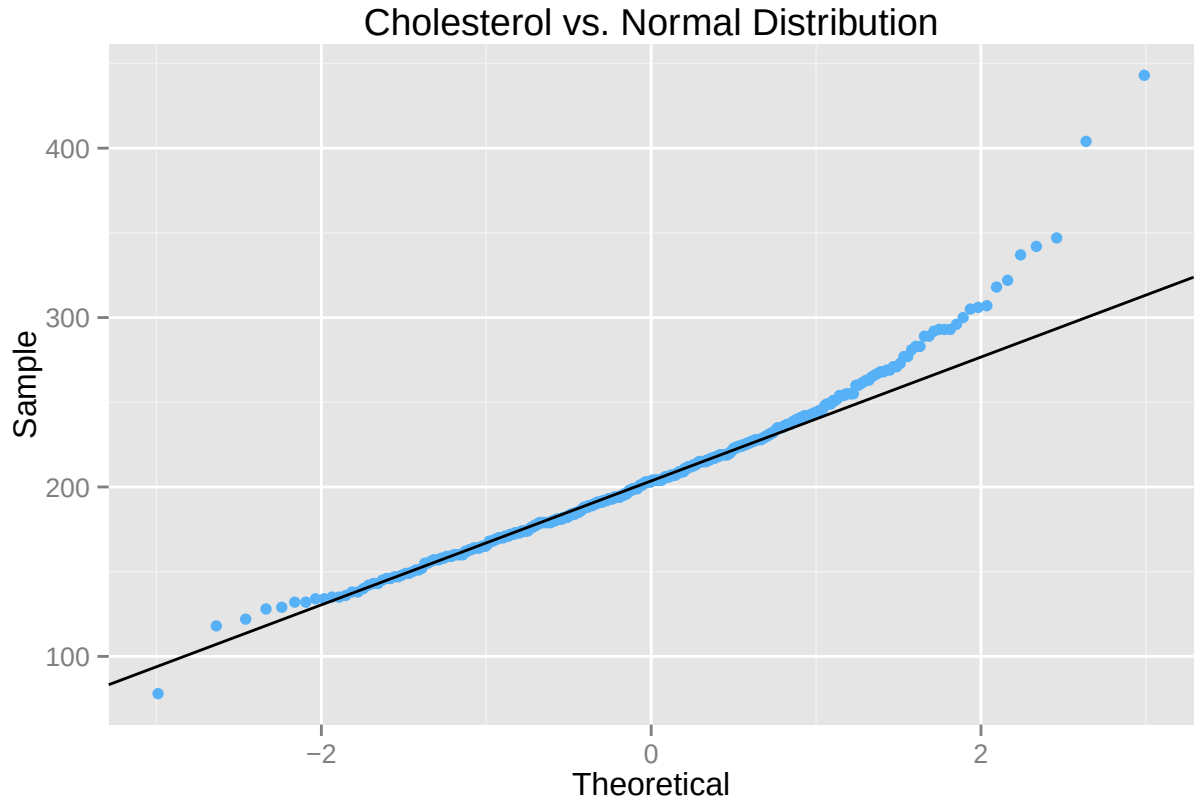
```
chol = data$chol

# Produce the histogram

ggplot(data) + geom_histogram(aes(x = chol), binwidth = 15, col = "white", fill = "#56B1F7") +
  ggtitle("Histogram of Cholesterol") +
  xlab("Cholesterol") + ylab("Frequency")
```




```
# Produce the boxplot

boxplot(chol, horizontal = TRUE, col = "white", border = "#56B1F7",
        xlab = "Cholesterol",
        main = "Boxplot of Cholesterol")
```




```
# Produce the qq-plot

exp.sample = rnorm(n = 10000)
y1y2 = quantile(chol, probs = c(0.25, 0.75))
x1x2 = quantile(exp.sample, probs = c(0.25, 0.75))
slope = (y1y2[2] - y1y2[1])/(x1x2[2] - x1x2[1])
intercept = slope * (0 - x1x2[1]) + y1y2[1]
ggplot(data, aes(sample = chol)) + stat_qq(col = "#56B1F7") +
  ggtitle("Cholesterol vs. Normal Distribution") +
  xlab("Theoretical") +
  ylab("Sample") +
  geom_abline(intercept = intercept, slope = slope)
```



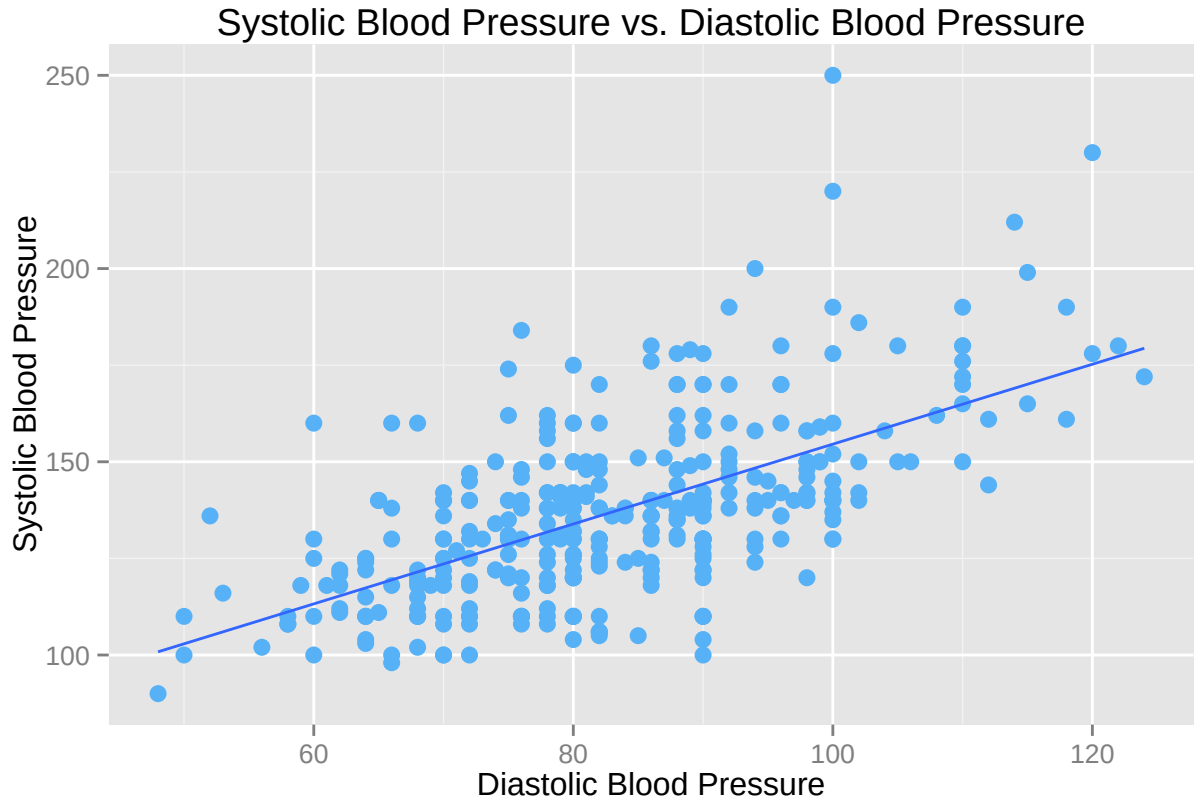
The histogram for Cholesterol looks fairly Gaussian centered around 200. There are some outliers above 400 and one low outlier below 100. When you compare the boxplot to that of Glycosolated Hemoglobin, you see that there are fewer outliers. When observing the qq-plot, the data follows the Gaussian distribution fairly well, with the exception of a few outliers.

Overall, cholesterol is better approximated with Gaussian distribution than Glycosolated Hemoglobin.

Question 3

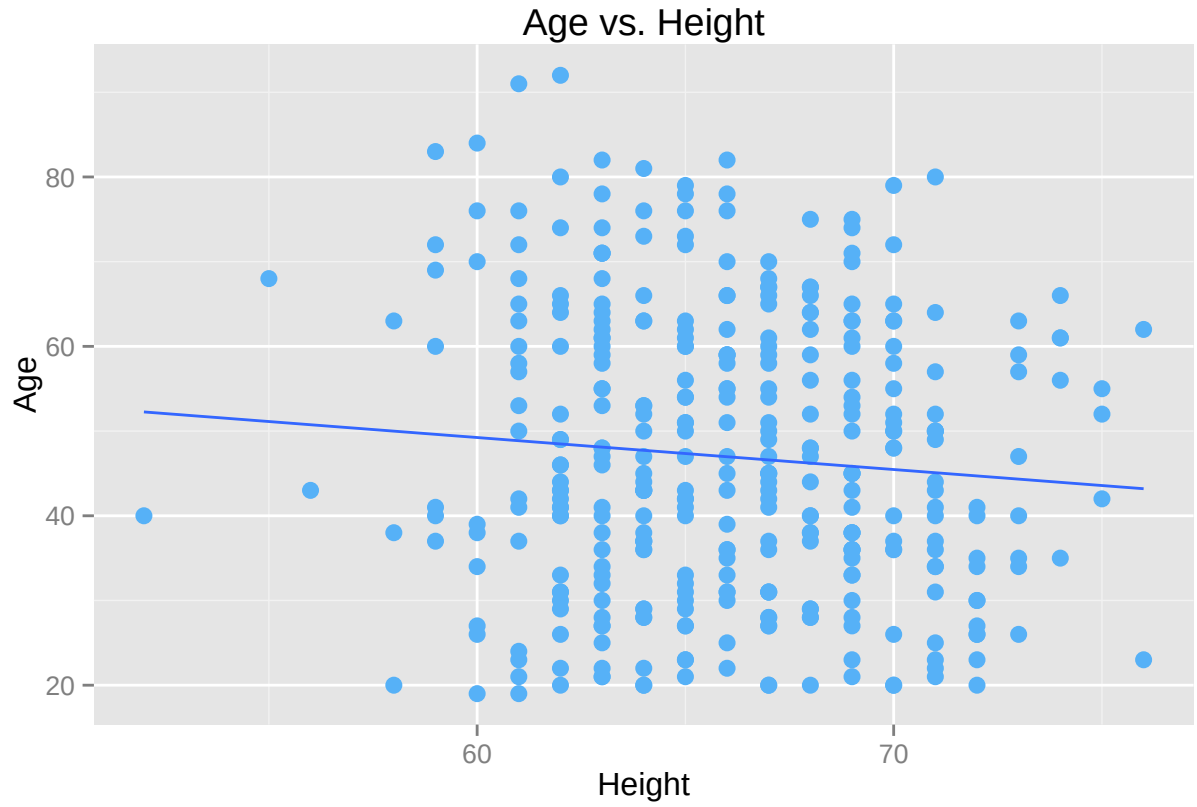
First let's make the scatterplot of bp.1s vs. bp.1d.

```
ggplot(data, aes(x = bp.1d, y = bp.1s)) +
    geom_point(shape = 16, col = "#56B1F7", size = 3) +
    geom_smooth(method = lm, se = FALSE) +
    xlab("Diastolic Blood Pressure") +
    ylab("Systolic Blood Pressure") +
    ggtitle("Systolic Blood Pressure vs. Diastolic Blood Pressure")
```



Now let's make the scatterplot of age vs. height.

```
ggplot(data, aes(x = height, y = age)) +
    geom_point(shape = 16, col = "#56B1F7", size = 3) +
    geom_smooth(method = lm, se = FALSE) +
    xlab("Height") +
    ylab("Age") +
    ggtitle("Age vs. Height")
```



Observing the first scatterplot shows a fair increase in Systolic Blood Pressure (bp.1s) as Diastolic Blood Pressure (bp.1d) increases. A slight linear relationship is evident, thus these features do not seem independent.

Observing the second scatterplot shows no relationship between the two variables. The plot shows random scatter, and the regression line is fairly horizontal. This is what we would expect because everyone in our sample is 19 years or older, and wtihin this age bracket, people are usually fully grown, so we conclude that these features are independent.

Question 4

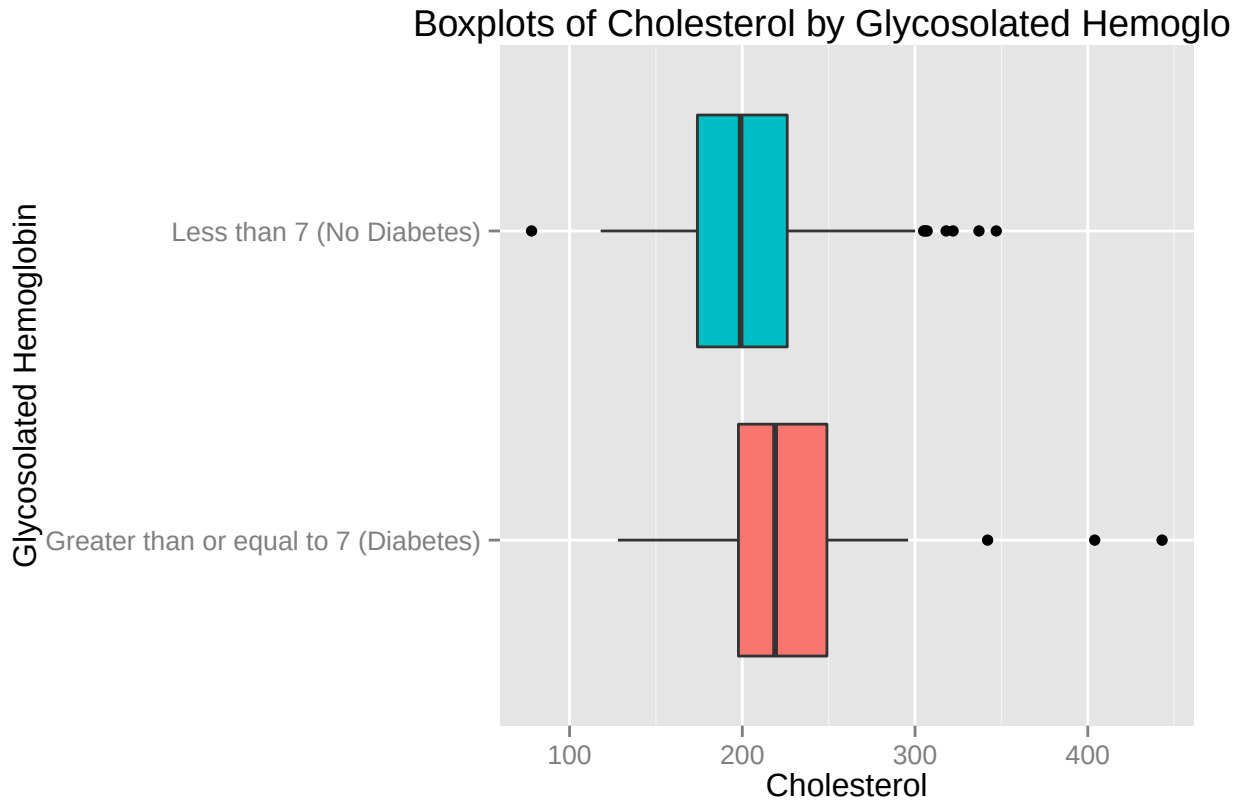
First let's make a logical vector in our "data" data frame that's conditional on whether glyhb >= 7 or glyhb < 7.

```
diabetes_locs = glyhb >= 7
diabetes = c()
diabetes[diabetes_locs] = "Greater than or equal to 7 (Diabetes)"
diabetes[!diabetes_locs] = "Less than 7 (No Diabetes)"
data$diabetes = factor(diabetes)
```

Now let's begin making boxplots for each available feature conditional on the statement above.

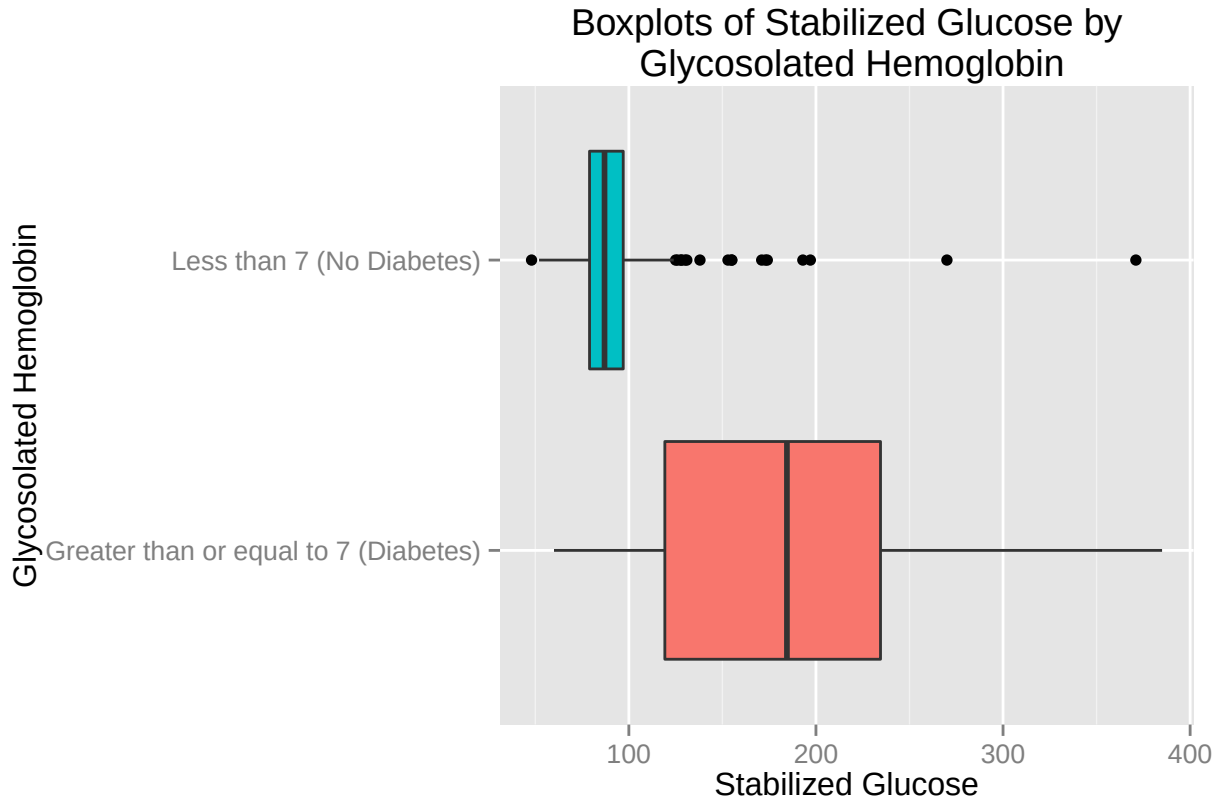
Let's first make conditional boxplots for Cholesterol.

```
ggplot(data, aes(x = diabetes, y = chol)) +
  geom_boxplot(aes(fill = diabetes)) +
  coord_flip() +
  theme(legend.position="none") +
  ylab("Cholesterol") +
  xlab("Glycosolated Hemoglobin") +
  ggtitle("Boxplots of Cholesterol by Glycosolated Hemoglobin")
```



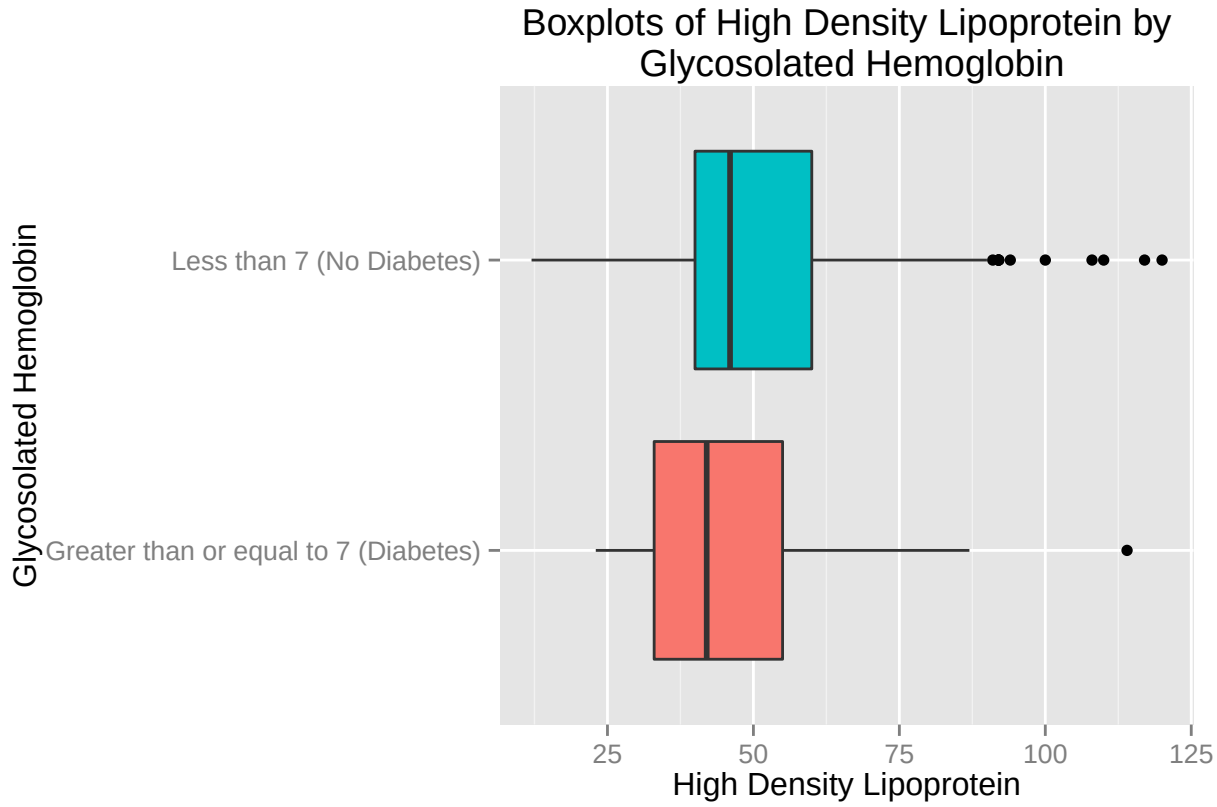
Now let's make them for Stabilized Glucose.

```
ggplot(data, aes(x = diabetes, y = stab.glu)) +
  geom_boxplot(aes(fill = diabetes)) +
  coord_flip() +
  theme(legend.position="none") +
  ylab("Stabilized Glucose") +
  xlab("Glycosolated Hemoglobin") +
  ggtitle("Boxplots of Stabilized Glucose by\n Glycosolated Hemoglobin")
```



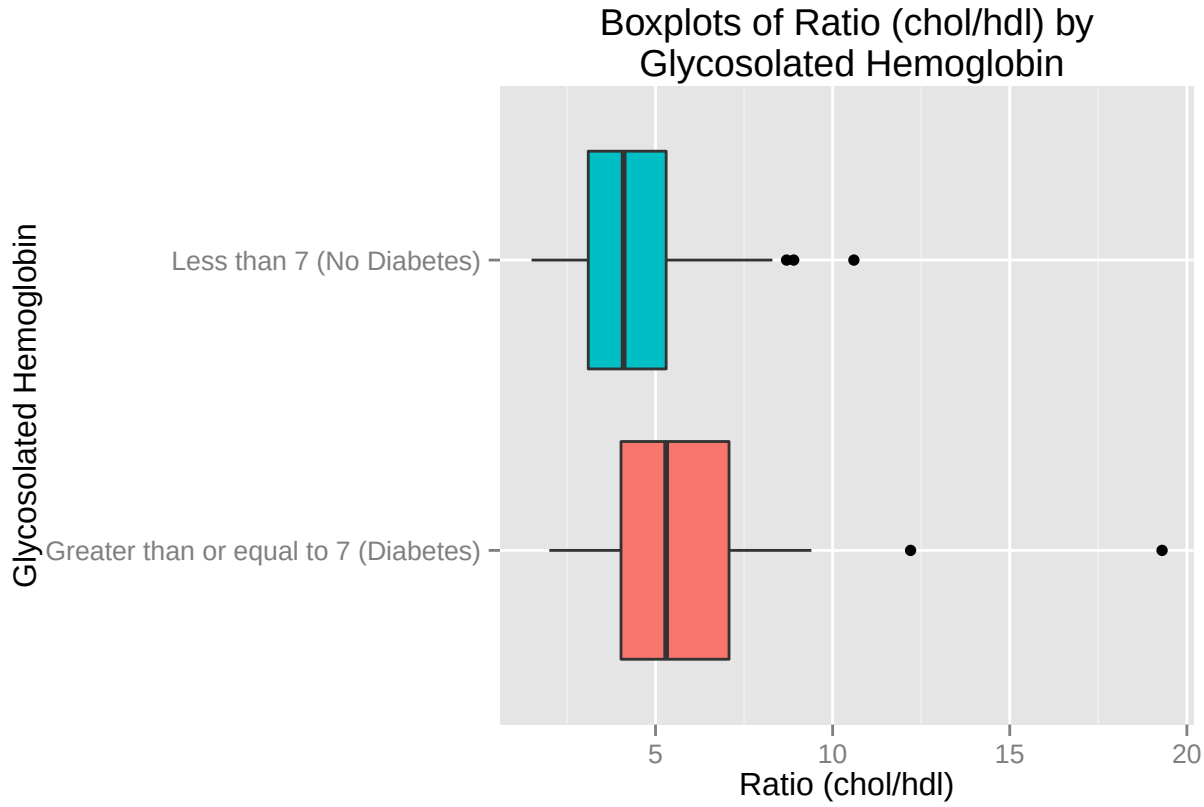
Now let's make them for High Density Lipoprotein.

```
ggplot(data, aes(x = diabetes, y = hdl)) +
  geom_boxplot(aes(fill = diabetes)) +
  coord_flip() +
  theme(legend.position="none") +
  ylab("High Density Lipoprotein") +
  xlab("Glycosolated Hemoglobin") +
  ggtitle("Boxplots of High Density Lipoprotein by\n Glycosolated Hemoglobin")
```



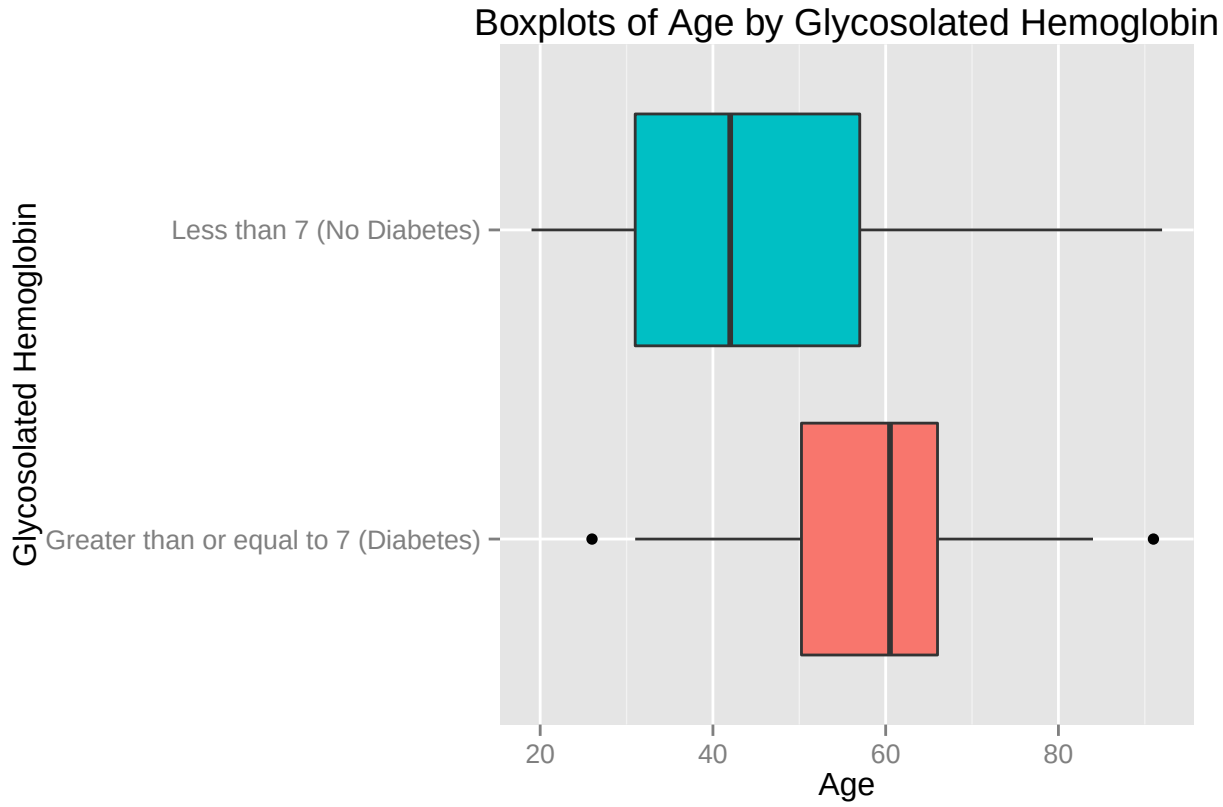
Now let's make them for Ratio (chol/hdl).

```
ggplot(data, aes(x = diabetes, y = ratio)) +
  geom_boxplot(aes(fill = diabetes)) +
  coord_flip() +
  theme(legend.position="none") +
  ylab("Ratio (chol/hdl)") +
  xlab("Glycosolated Hemoglobin") +
  ggtitle("Boxplots of Ratio (chol/hdl) by\n Glycosolated Hemoglobin")
```



Now let's make them for Age.

```
ggplot(data, aes(x = diabetes, y = age)) +
  geom_boxplot(aes(fill = diabetes)) +
  coord_flip() +
  theme(legend.position="none") +
  ylab("Age") +
  xlab("Glycosolated Hemoglobin") +
  ggtitle("Boxplots of Age by Glycosolated Hemoglobin")
```



Now let's make them for Height.

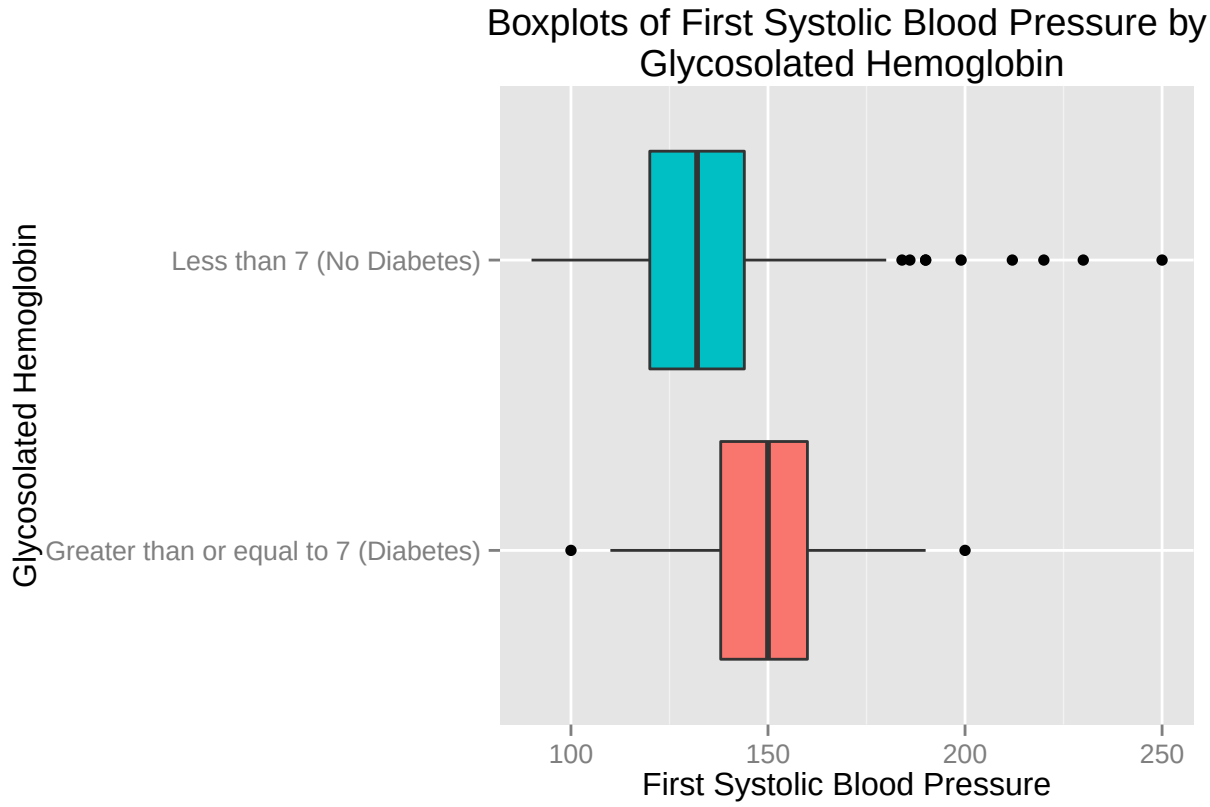
```
ggplot(data, aes(x = diabetes, y = height)) +
  geom_boxplot(aes(fill = diabetes)) +
  coord_flip() +
  theme(legend.position="none") +
  ylab("Height") +
  xlab("Glycosolated Hemoglobin") +
  ggtitle("Boxplots of Height by Glycosolated Hemoglobin")
```

Now let's make them for Weight.

```
ggplot(data, aes(x = diabetes, y = weight)) +
  geom_boxplot(aes(fill = diabetes)) +
  coord_flip() +
  theme(legend.position="none") +
  ylab("Weight") +
  xlab("Glycosolated Hemoglobin") +
  ggtitle("Boxplots of Weight by Glycosolated Hemoglobin")
```

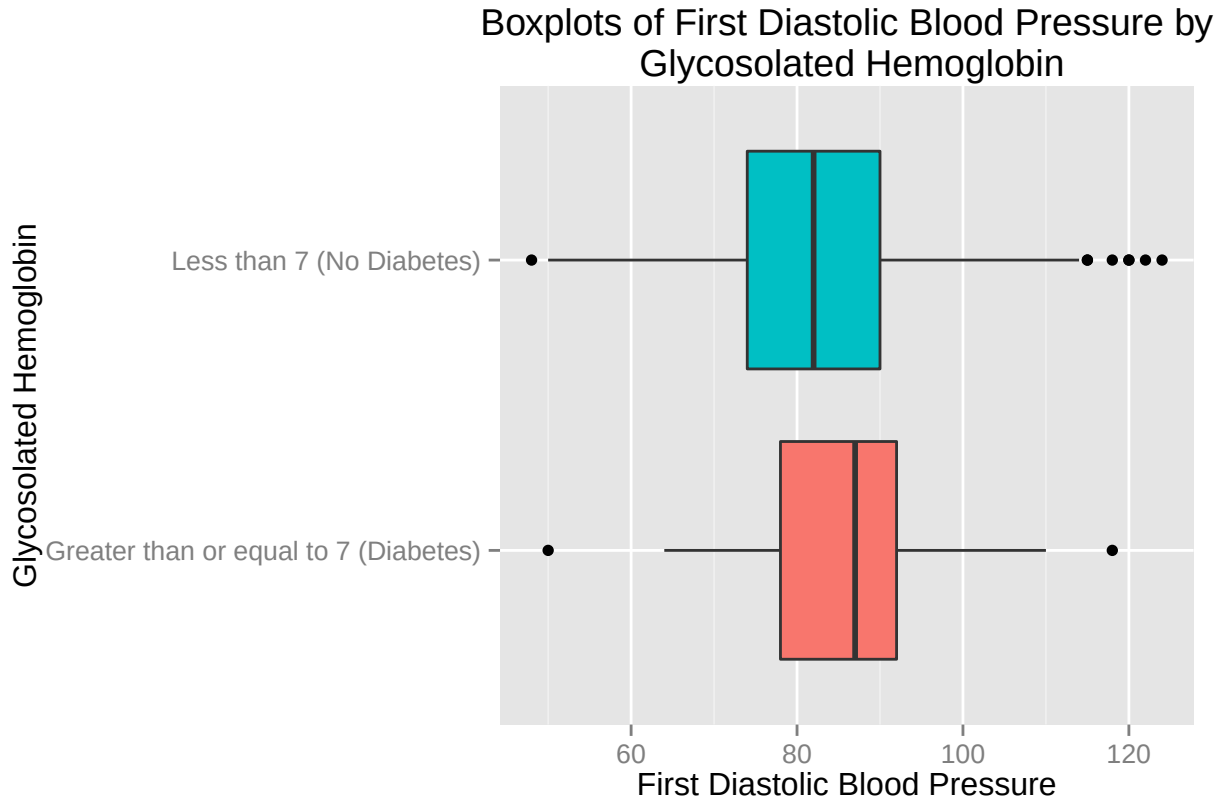
Now let's make them for Systolic Blood Pressure.

```
ggplot(data, aes(x = diabetes, y = bp.1s)) +
  geom_boxplot(aes(fill = diabetes)) +
  coord_flip() +
  theme(legend.position="none") +
  ylab("First Systolic Blood Pressure") +
  xlab("Glycosolated Hemoglobin") +
  ggtitle("Boxplots of First Systolic Blood Pressure by\n Glycosolated Hemoglobin")
```



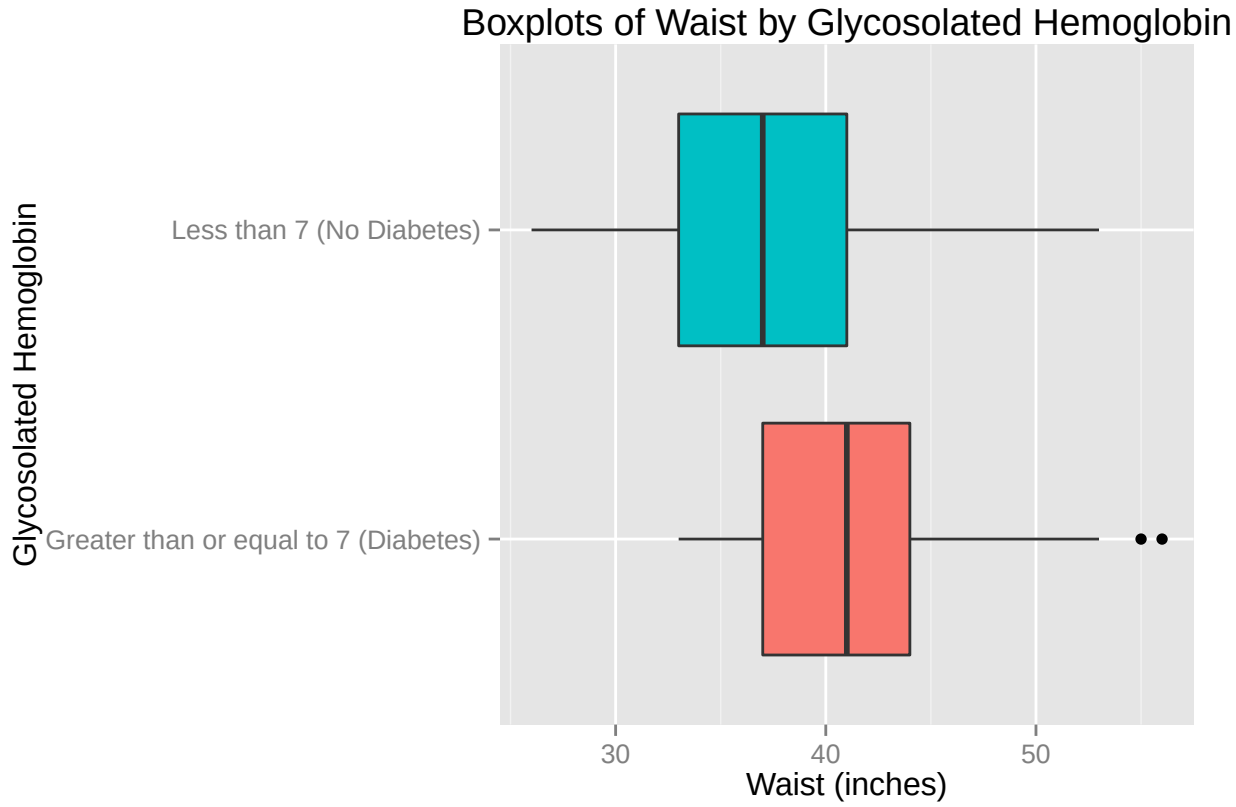
Now let's make them for Diastolic Blood Pressure.

```
ggplot(data, aes(x = diabetes, y = bp.1d)) +
  geom_boxplot(aes(fill = diabetes)) +
  coord_flip() +
  theme(legend.position="none") +
  ylab("First Diastolic Blood Pressure") +
  xlab("Glycosolated Hemoglobin") +
  ggtitle("Boxplots of First Diastolic Blood Pressure by\n Glycosolated Hemoglobin")
```



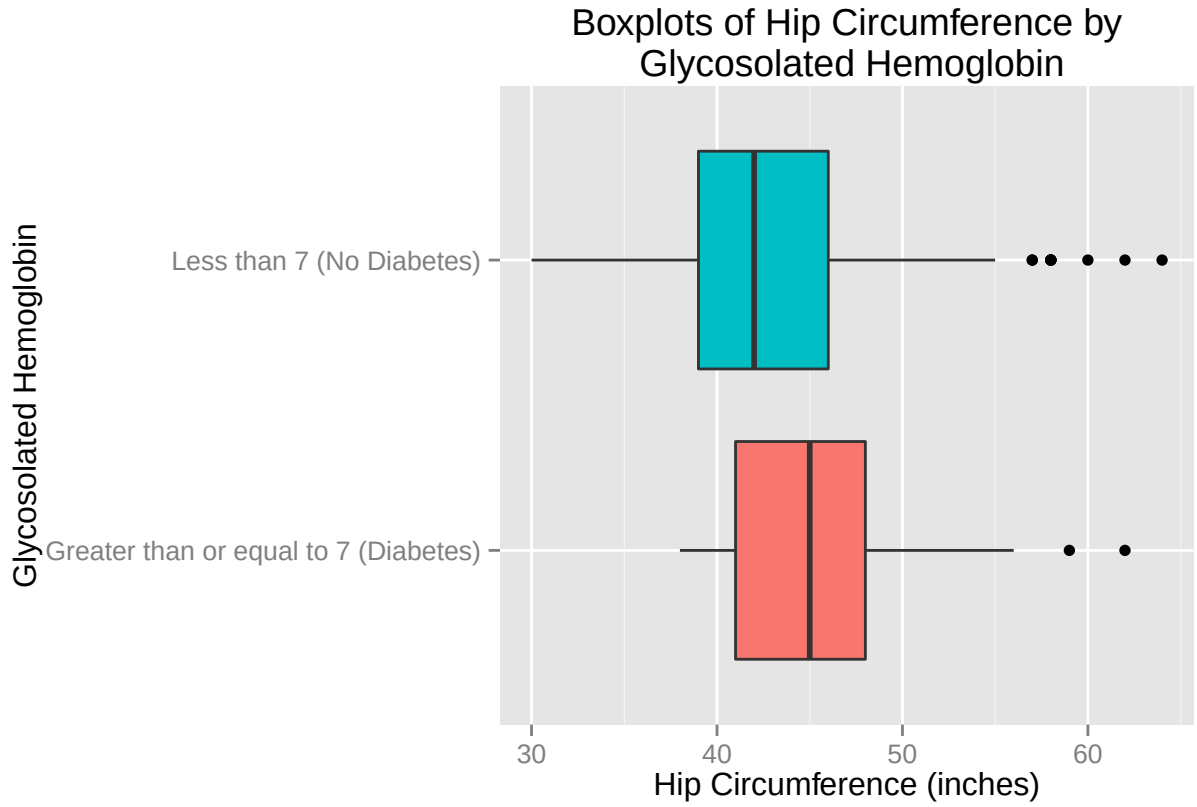
Now let's make them for Waist size.

```
ggplot(data, aes(x = diabetes, y = waist)) +
  geom_boxplot(aes(fill = diabetes)) +
  coord_flip() +
  theme(legend.position="none") +
  ylab("Waist (inches)") +
  xlab("Glycosolated Hemoglobin") +
  ggtitle("Boxplots of Waist by Glycosolated Hemoglobin")
```



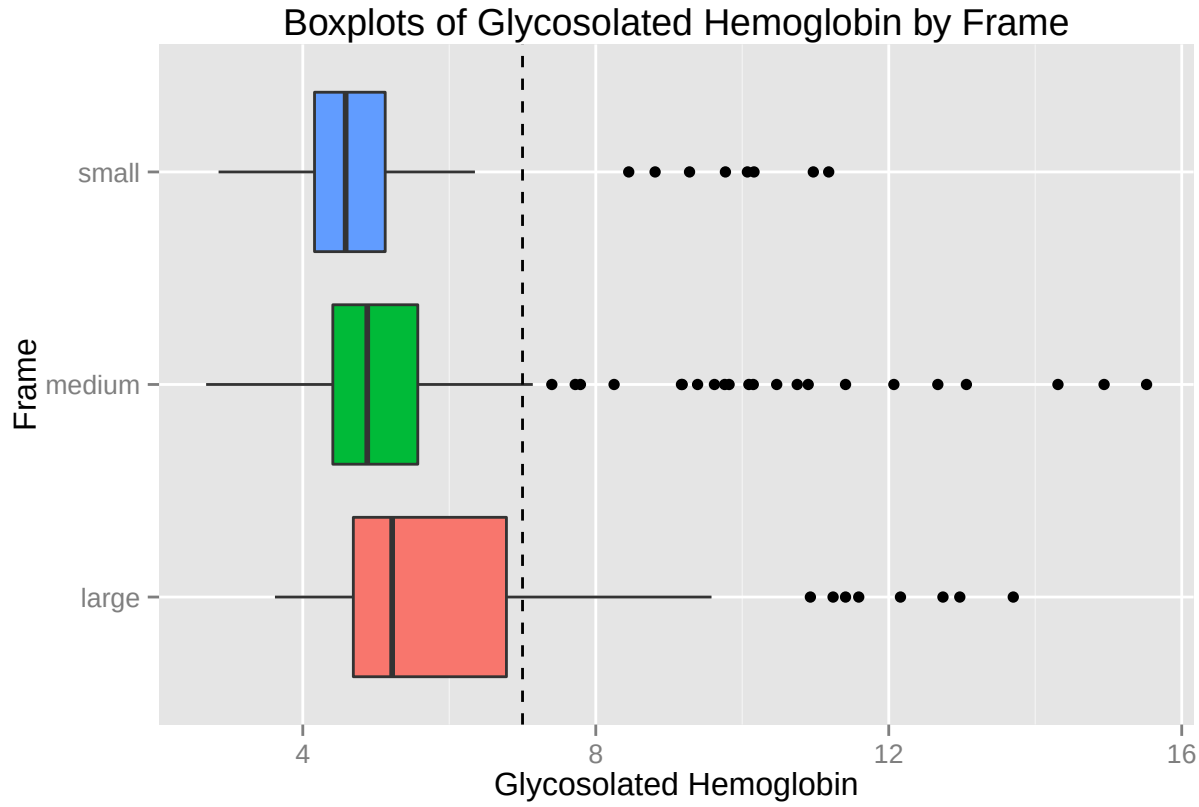
Now let's make them for Hip size.

```
ggplot(data, aes(x = diabetes, y = hip)) +
  geom_boxplot(aes(fill = diabetes)) +
  coord_flip() +
  theme(legend.position="none") +
  ylab("Hip Circumference (inches)") +
  xlab("Glycosolated Hemoglobin") +
  ggtitle("Boxplots of Hip Circumference by\n Glycosolated Hemoglobin")
```



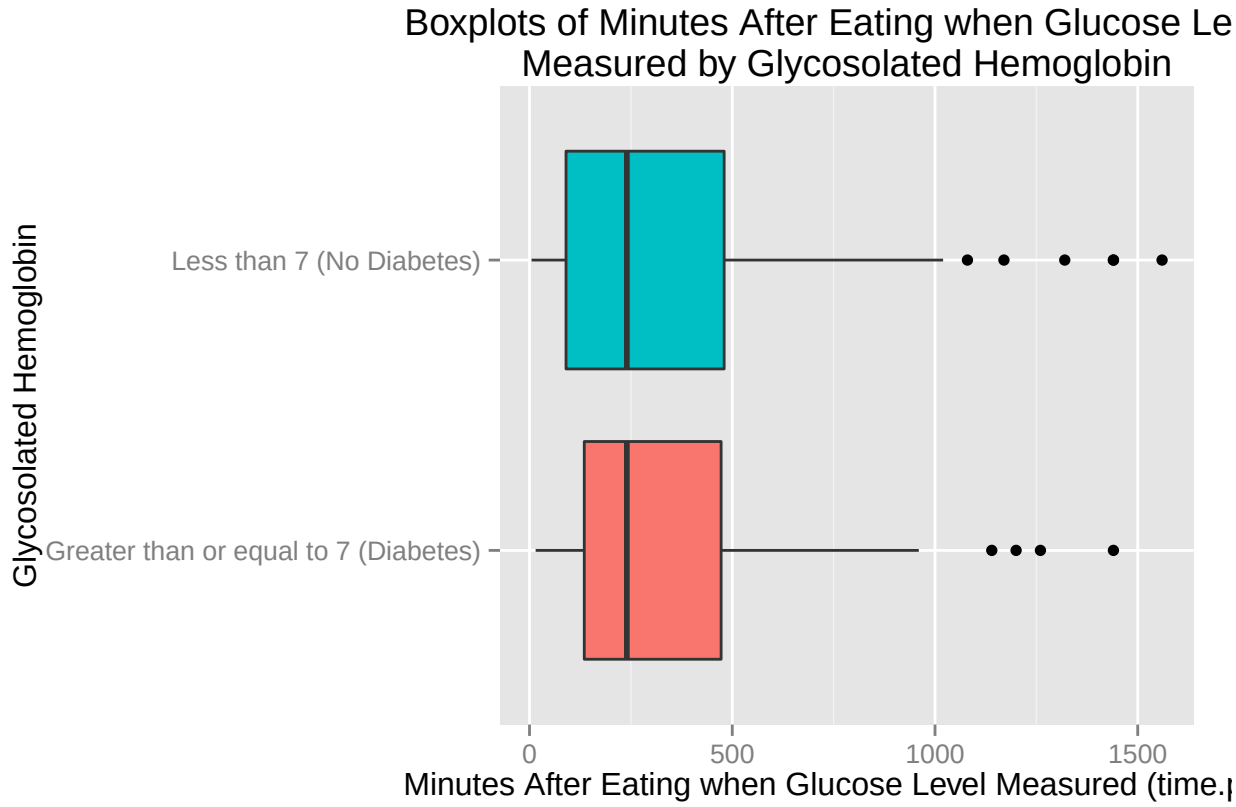
Now let's make them for Frame size. We have some blanks in our data for this variable, so we will first remove those. We will add a dotted line for Glyhb = 7, so we know where we classify by diabetes.

```
ggplot(data[data$frame != "", ], aes(x = frame, y = glyhb)) +
  geom_boxplot(aes(fill = frame)) +
  coord_flip() +
  theme(legend.position="none") +
  ylab("Glycosolated Hemoglobin") +
  xlab("Frame") +
  ggtitle("Boxplots of Glycosolated Hemoglobin by Frame") +
  geom_hline(yintercept = 7, linetype = "dashed")
```



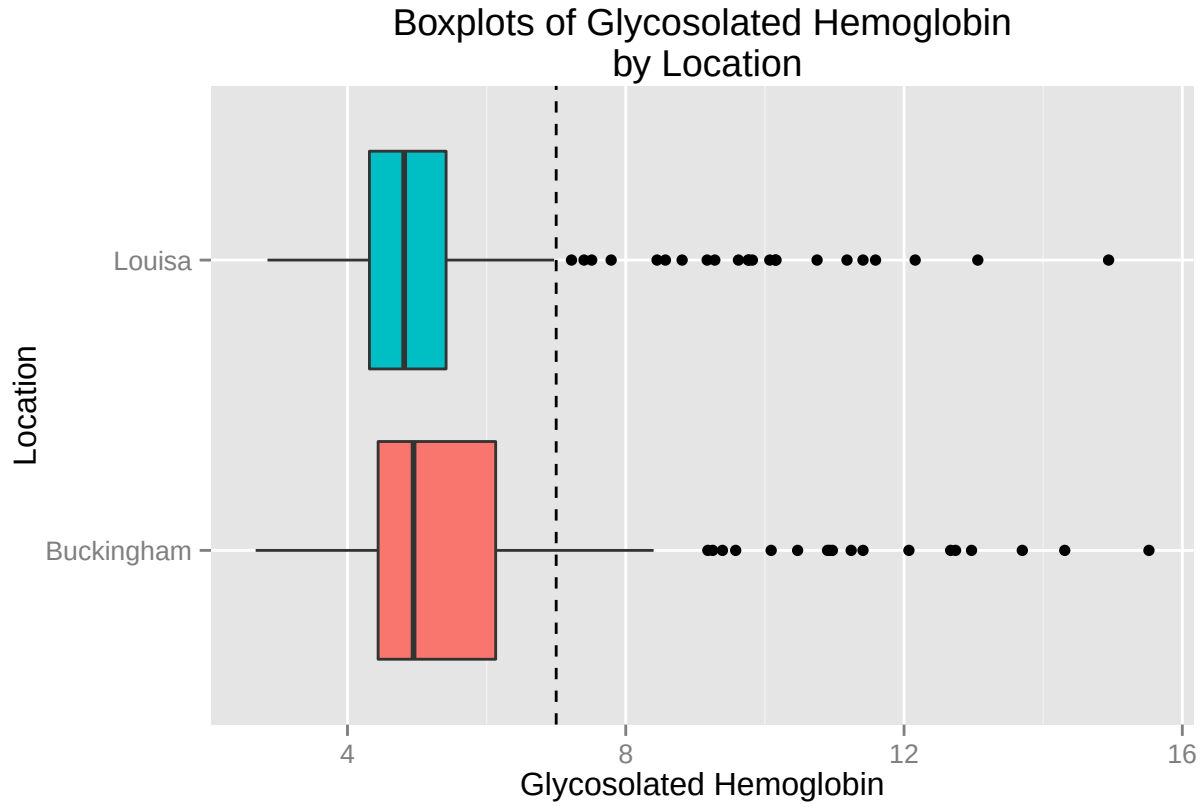
Now let's make them for Postprandial Time.

```
ggplot(data, aes(x = diabetes, y = time.ppn)) +
  geom_boxplot(aes(fill = diabetes)) +
  coord_flip() +
  theme(legend.position="none") +
  ylab("Minutes After Eating when Glucose Level Measured (time.ppn)") +
  xlab("Glycosolated Hemoglobin") +
  ggtitle("Boxplots of Minutes After Eating when Glucose Level \nMeasured by Glycosolated Hemoglobin")
```



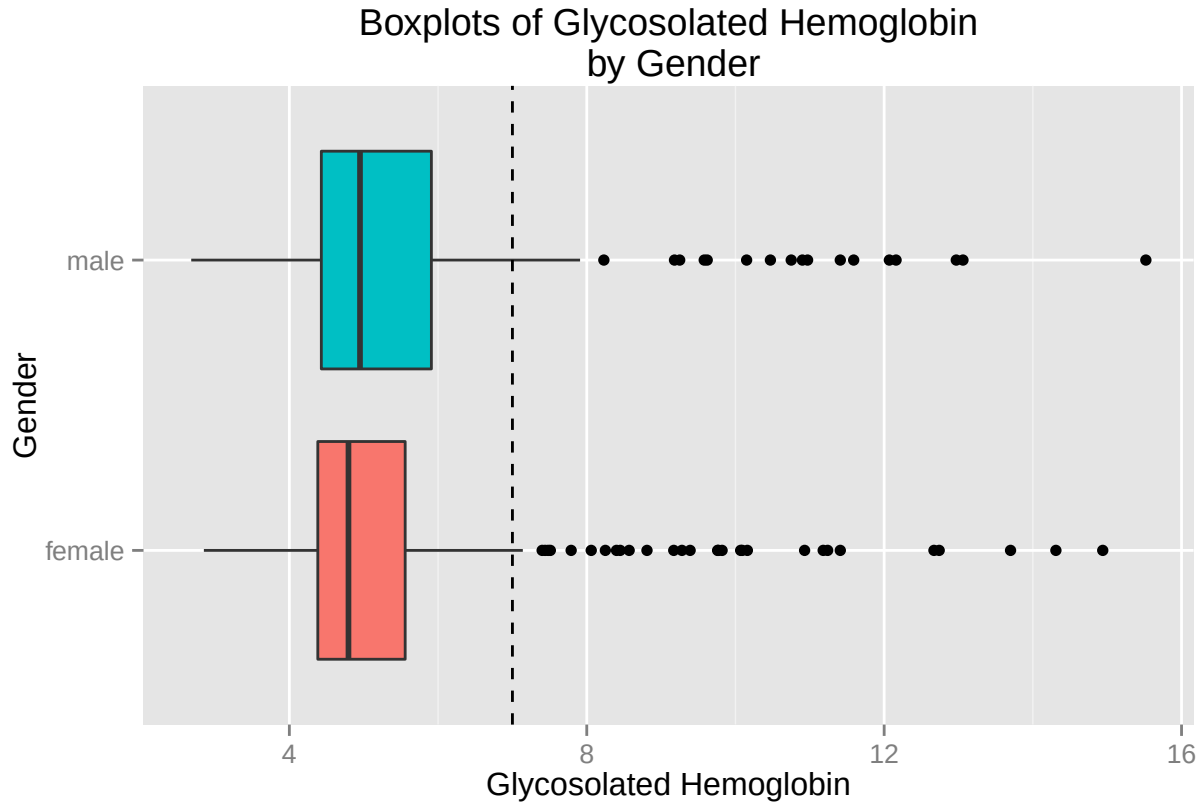
Now let's make them for Location.

```
ggplot(data, aes(x = location, y = glyhb)) +
  geom_boxplot(aes(fill = location)) +
  coord_flip() +
  theme(legend.position="none") +
  ylab("Glycosolated Hemoglobin") +
  xlab("Location") +
  ggtitle("Boxplots of Glycosolated Hemoglobin\n by Location") +
  geom_hline(yintercept = 7, linetype = "dashed")
```



Now let's make them for Gender.

```
ggplot(data, aes(x = gender, y = glyhb)) +
  geom_boxplot(aes(fill = gender)) +
  coord_flip() +
  theme(legend.position="none") +
  ylab("Glycosolated Hemoglobin") +
  xlab("Gender") +
  ggtitle("Boxplots of Glycosolated Hemoglobin\n by Gender") +
  geom_hline(yintercept = 7, linetype = "dashed")
```



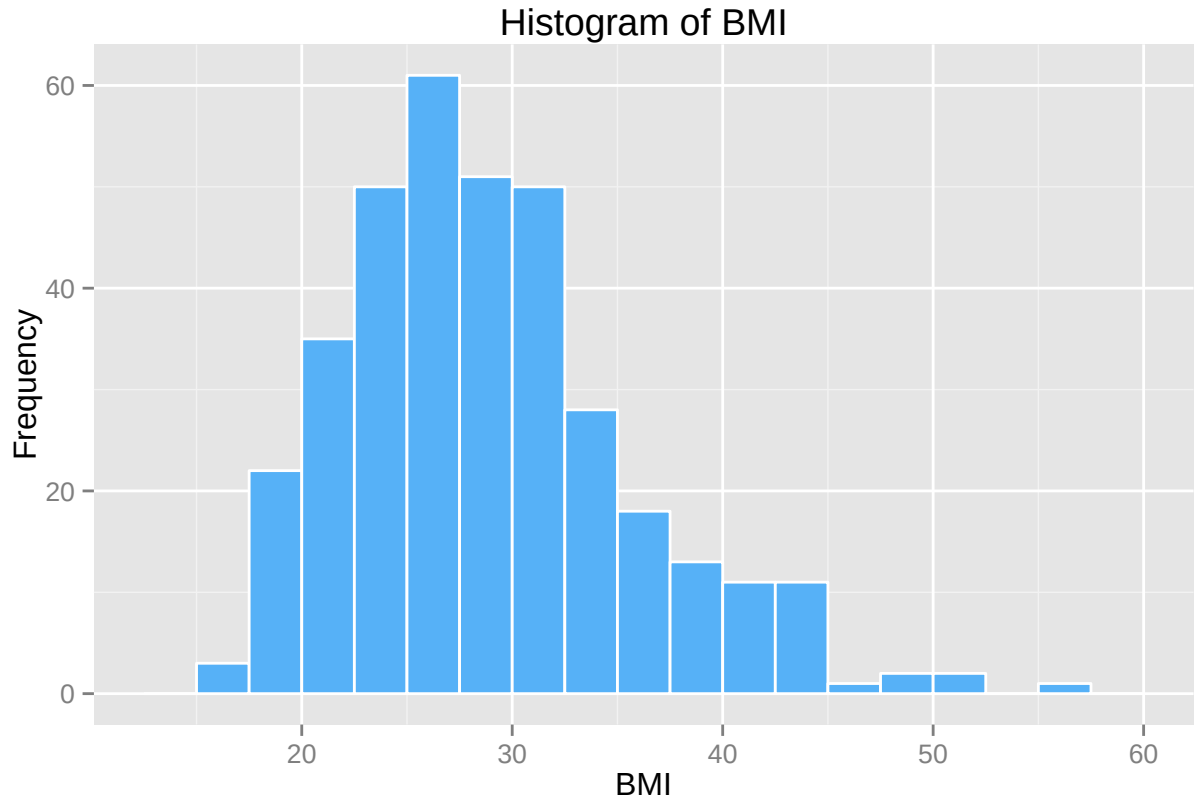
Question 5

First let's create the BMI and WHR variables and then add them to our data frame.

```
data$BMI = 703 * (data$weight / (data$height)^2)
data$WHR = data$waist / data$hip
```

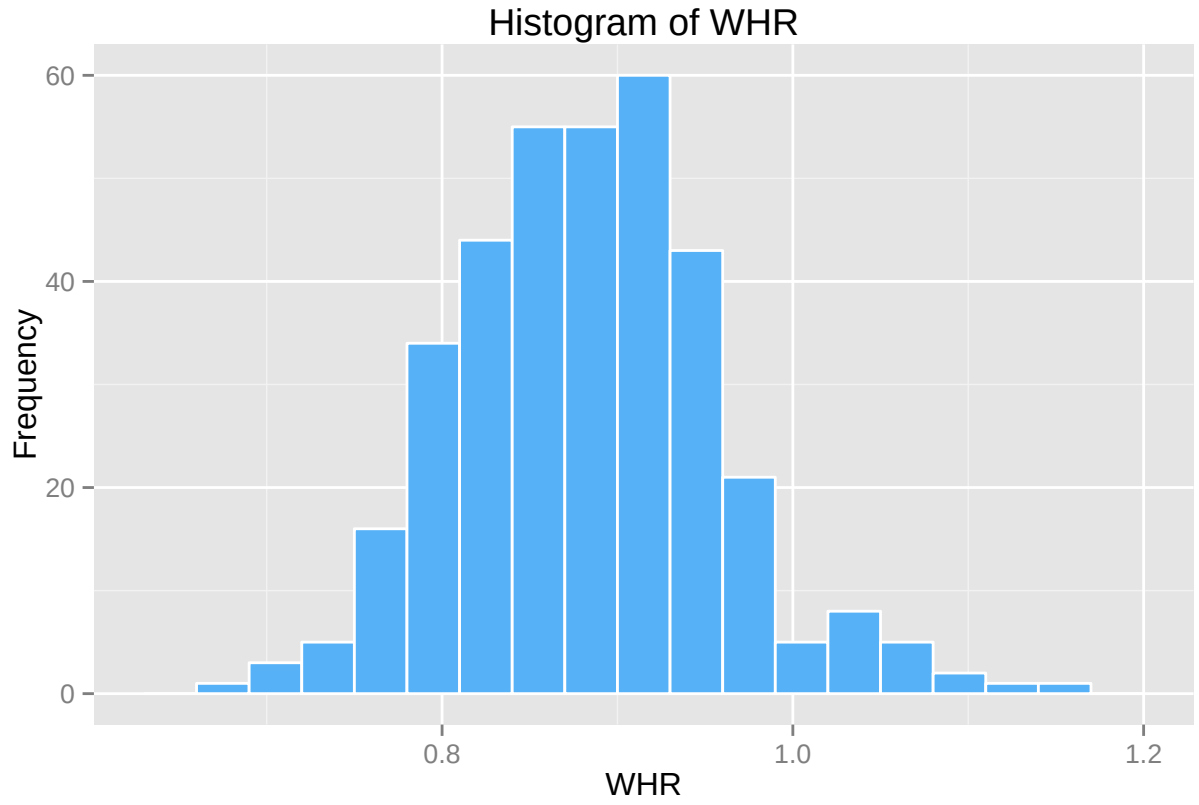
Now let's compute their empirical distributions. Let's make one for BMI first.

```
ggplot(data) + geom_histogram(aes(x = BMI), binwidth = 2.5, col = "white", fill = "#56B1F7") +
  ggtitle("Histogram of BMI") +
  xlab("BMI") + ylab("Frequency")
```



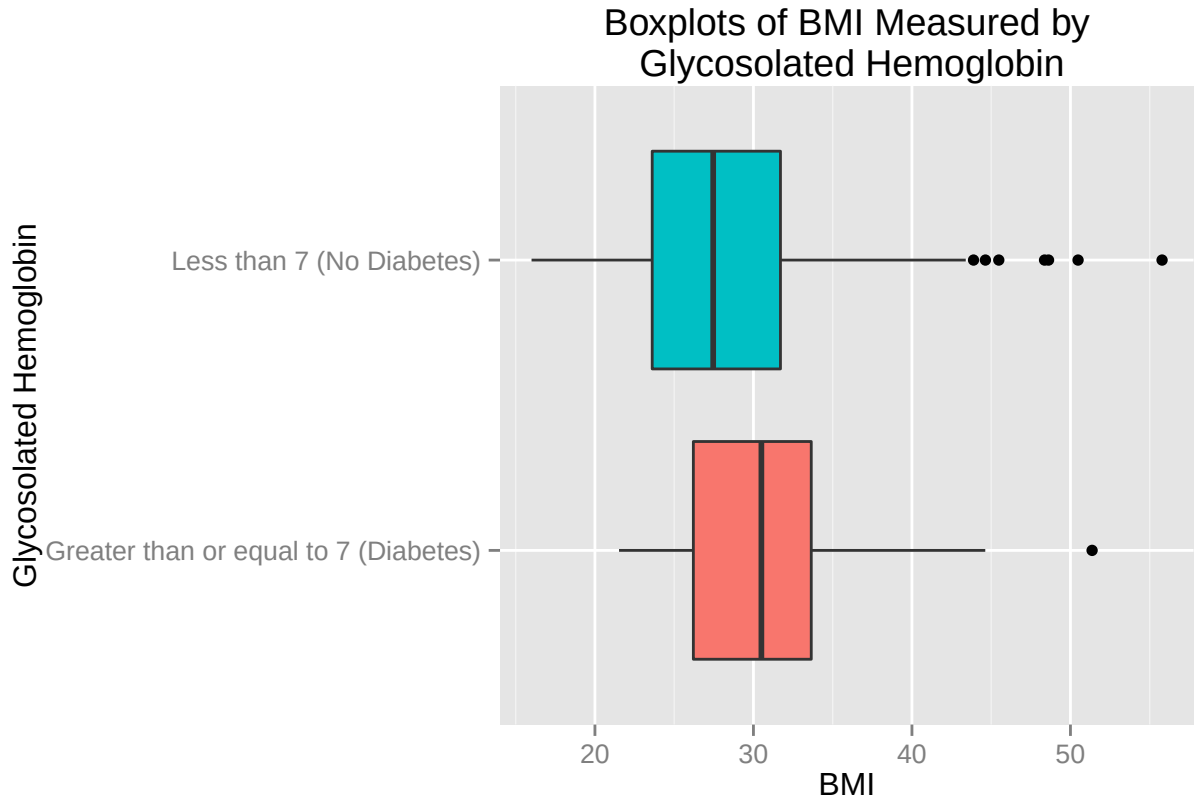
Now let's make one for WHR.

```
ggplot(data) + geom_histogram(aes(x = WHR), binwidth = 0.03, col = "white", fill = "#56B1F7") +
  ggtitle("Histogram of WHR") +
  xlab("WHR") + ylab("Frequency")
```



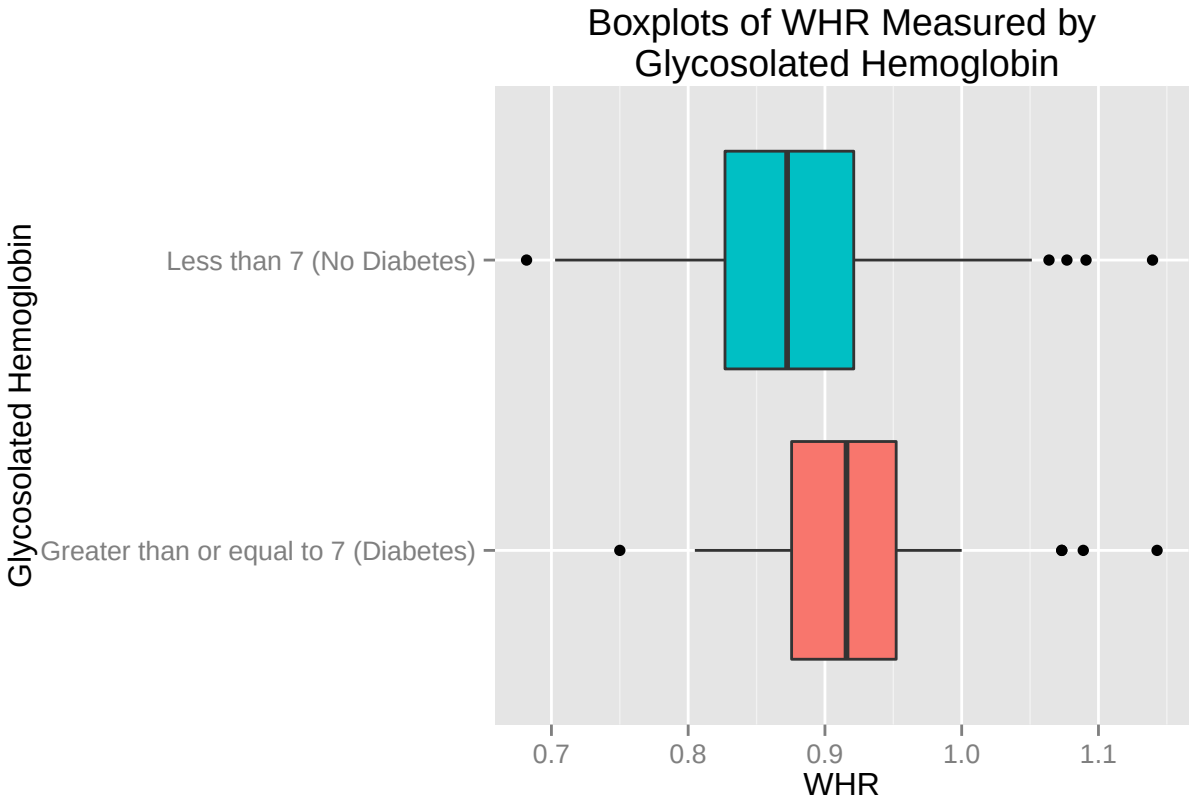
Now let's compute their conditional boxplots of glyhb. Let's make one for BMI first.

```
ggplot(data, aes(x = diabetes, y = BMI)) +
  geom_boxplot(aes(fill = diabetes)) +
  coord_flip() +
  theme(legend.position="none") +
  ylab("BMI") +
  xlab("Glycosolated Hemoglobin") +
  ggtitle("Boxplots of BMI Measured by\n Glycosolated Hemoglobin")
```



Now let's make one for WHR.

```
ggplot(data, aes(x = diabetes, y = WHR)) +
  geom_boxplot(aes(fill = diabetes)) +
  coord_flip() +
  theme(legend.position="none") +
  ylab("WHR") +
  xlab("Glycosolated Hemoglobin") +
  ggtitle("Boxplots of WHR Measured by\n Glycosolated Hemoglobin")
```



Question 6

There does seem to be a difference in cholesterol level between those who have diabetes and those who do not. Though this feature looks like it is related to the presence of type-II diabetes, it might not be the best predictor we have.

The difference in stabilized glucose level between those who have and do not have diabetes seems very significant. Not only are the distributions centered at different values, but their variances differ greatly as well.

HDL is similar to cholesterol in the sense that there does seem to be a difference between people who have and do not have diabetes. Once again, this might not be the most telling indicator, but there does seem to be a relationship.

The ratio of cholesterol to HDL seems lower for those who do not have diabetes. Overall, ratio seems like it has potential to be related to type-II diabetes but we would have to analyze the data more to know for sure.

Age seems very related to the presence of type-II diabetes. The average age for those with diabetes seems much higher than the average age for those without. The variance of the ages of those without diabetes is higher than the variance of those without.

Height seems almost identical among both groups. It does not seem very relevant to the presence of diabetes. Since type-II diabetes is related to obesity, we should pay more attention to the ratio of height to weight instead of height alone.

It does seem that those with diabetes have a higher weight than those without. However, once again, a ratio of height to weight should be a better indicator. Someone might have a very high weight but also be very tall, meaning that they are not necessarily unhealthy.

First systolic blood pressure is higher for those with diabetes than those without. The distribution of first systolic blood pressure for those without diabetes takes on a larger range. It seems that this feature is significant.

The boxplots of first diastolic blood pressure seem to overlap a lot so the feature does not seem that related to the presence of type-II diabetes.

At a first glance, waist size seems quite relevant to diabetes. At the same time, looking at a ratio of waist to hip size is more informative and this might be a better indicator.

The boxplots of hip size for each group once again overlap a lot. Hip circumference on its own might not be the most telling feature.

The distributions of the value of glycosolated hemoglobin level are significantly different among frame sizes. However, the majority of the data for glycosolated hemoglobin for each frame size is below seven. In other words, frame size is not a good indicator of diabetes.

The boxplots of postprandial time for the two groups are close to identical. This feature is not related to the presence of type-II diabetes.

Location does not seem relevant to whether or not a person has type-II diabetes.

Gender does not seem related to the presence of type-II diabetes. The distributions of glycosolated hemoglobin level are similar for both genders.

BMI seems different between those who have diabetes and those who do not. A higher BMI is related to the presence of diabetes. BMI is a ratio, so it gives us information about both height and weight.

WHR also looks relevant to type-II diabetes. The boxplots seem fairly distinct. Because this is a ratio of two seemingly predictive features, it seems like a good indicator.

Part 3: Parametric Inference

Question 1:

Let's fit BMI to a Gamma distribution. There are two parameterizations of this. We will use the one with alpha and beta.

First we begin by finding the expected values of the first and second moments.

```
bmi = data$BMI
e1 <- mean(bmi)
e1
```

```
## [1] 28.84369
```

```
e2 <- mean(bmi^2)
e2
```

```
## [1] 877.1068
```

Next, we will use these values to find alpha and beta by setting them equal to the true moments.

```
b <- e1/(e2 - e1^2)
b
```

```
## [1] 0.6388641
```

```
a <- e1^2/(e2 - e1^2)
a
```

```
## [1] 18.4272
```

Now we will use a non-parametric bootstrap to find confidence intervals around alpha and beta. Let's first find all our values of alpha and beta.

```
gamma.est <- matrix(0, 1000, 2)
for (i in 1:1000) {
  boot <- sample(bmi, 1000, replace = T)
  e1 <- mean(boot)
  e2 <- mean(boot^2)
  gamma.est[i,1] <- e1^2/(e2 - e1^2)
  gamma.est[i,2] <- e1/(e2 - e1^2)
}
```

Now we can compute our 95% confidence intervals for each parameter. We will compute a percentile confidence interval.

For alpha, our 95% CI is:

```
pboot.perc.a <- quantile(gamma.est[,1], probs = c(0.025, 0.975))
pboot.perc.a
```

```
##     2.5%    97.5%
## 16.91242 20.07369
```

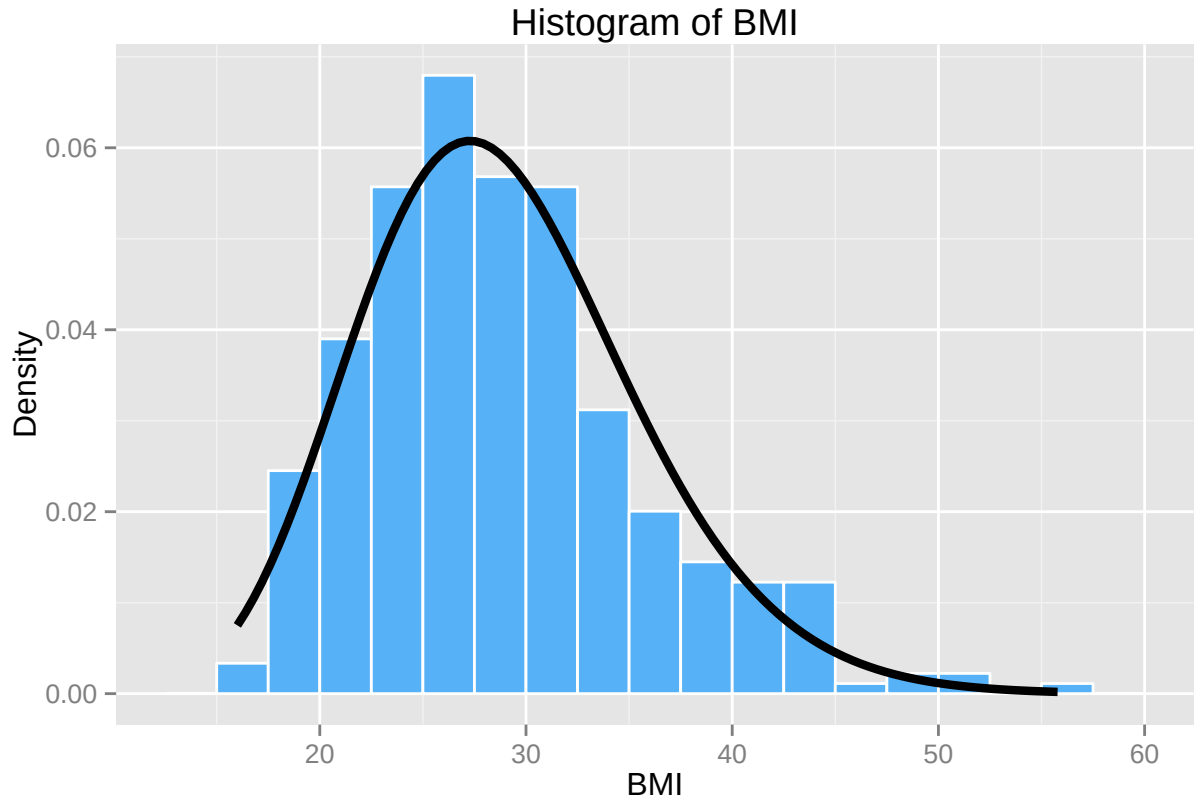
For beta, our 95% CI is:

```
pboot.perc.b <- quantile(gamma.est[,2], probs = c(0.025, 0.975))
pboot.perc.b
```

```
##      2.5%     97.5%
## 0.5840511 0.7018724
```

Lastly, let's fit our Gamma distribution to our empirical histogram.

```
ggplot(data) + geom_histogram(aes(x = BMI, y = ..density..), binwidth = 2.5, col = "white", fill = "#56
  stat_function(fun = dgamma, args = list(shape = a, rate = b), size = 1.5) +
  ggtitle("Histogram of BMI") +
  xlab("BMI") + ylab("Density")
```



Question 2

Let's fit WHR to a Gaussian distribution.

First let's solve for the MLE of the mean and the variance.

```
whr = data$WHR
mle.u <- mean(whr)
mle.u
```

```
## [1] 0.8821109
```

```
mle.sig <- sqrt(mean((whr - mle.u)^2))
mle.sig
```

```
## [1] 0.07373207
```

Now let's use nonparametric bootstrap to generate our means and variances.

```
whr.est <- matrix(0, 1000, 2)
for (i in 1:1000) {
boot <- sample(whr, 1000, replace = T)
      whr.est[i,1] <- mean(boot)
     whr.est[i,2] <- var(boot)
}
```

Now let's find our 95% CI for the mean and variance.

```
pboot.perc.mean <- quantile(whr.est[,1], probs = c(0.025, 0.975))
pboot.perc.mean
```

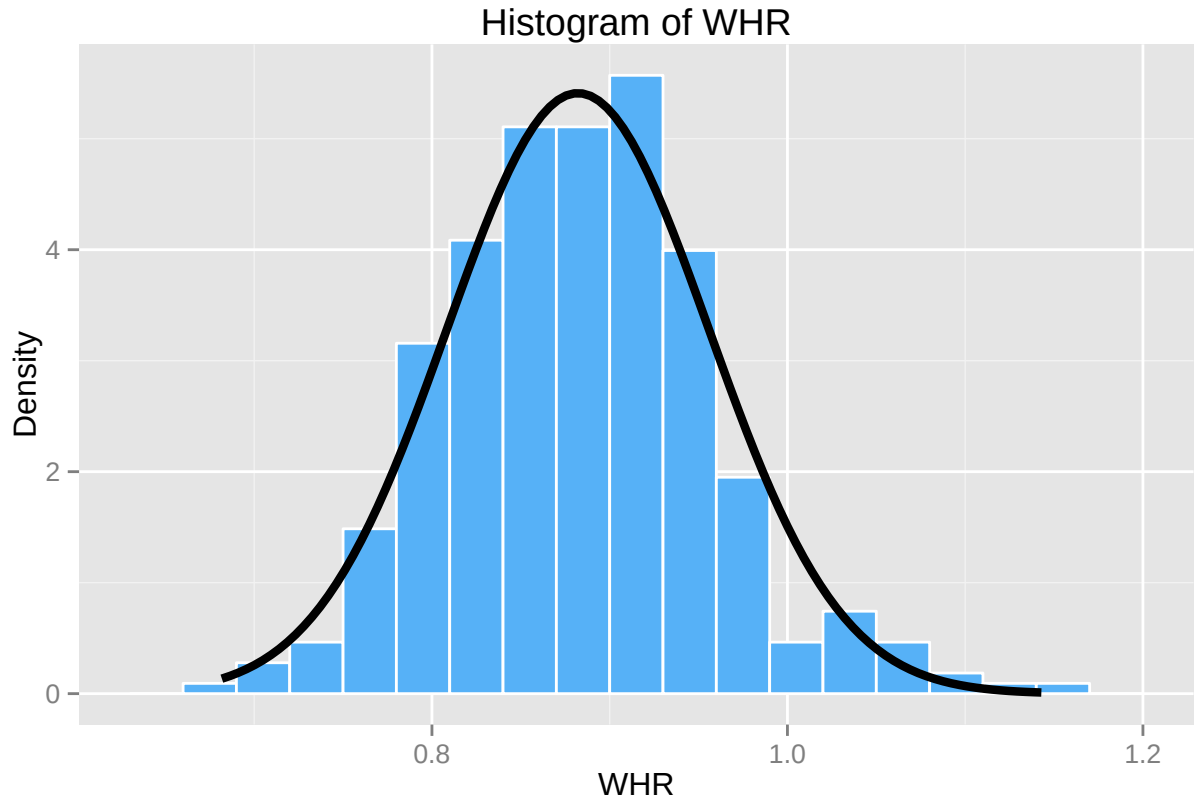
```
##      2.5%     97.5%
## 0.8775953 0.8864241
```

```
pboot.perc.var <- quantile(whr.est[,2], probs = c(0.025, 0.975))
pboot.perc.var
```

```
##       2.5%      97.5%
## 0.00490237 0.00603367
```

Lastly, let's fit our Gaussian distribution to our empirical histogram.

```
ggplot(data) + geom_histogram(aes(x = WHR, y = ..density..), binwidth = 0.03, col = "white", fill = "#5
  stat_function(fun = dnorm, args = list(mean = mle.u, sd = mle.sig), size = 1.5) +
  ggtitle("Histogram of WHR") +
  xlab("WHR") + ylab("Density")
```



Question 3

First let's deal with the case of diabetic males.

We will use nonparametric bootstrap to generate BMI and WHR values for this population.

```
est.maleND <- matrix(0, 1000, 4)
for (i in 1:1000) {
  boot <- sample(bmi[data$glyhb >= 7 & data$gender == "male"], 1000, replace = T)
  e1 <- mean(boot)
  e2 <- mean(boot^2)
  est.maleND[i,1] <- e1^2/(e2 - e1^2)
  est.maleND[i,2] <- e1/(e2 - e1^2)

 boot <- sample(whr[data$glyhb >= 7 & data$gender == "male"], 1000, replace = T)
  est.maleND[i,3] <- mean(boot)
  est.maleND[i,4] <- var(boot)
}
```

Now let's find 95% Confidence Intervals for the alpha & beta parameters of BMI and the mean & variance parameters of WHR.

```
pboot.perc.a.maleD <- quantile(est.maleND[,1], probs = c(0.025, 0.975))
pboot.perc.a.maleD
```

```
##     2.5%    97.5%
## 27.11879 33.24216
```

```
pboot.perc.b.maleD <- quantile(est.maleND[,2], probs = c(0.025, 0.975))
pboot.perc.b.maleD
```

```
##      2.5%     97.5%
## 0.9386298 1.1594966
```

```
pboot.perc.mu.maleD <- quantile(est.maleND[,3], probs = c(0.025, 0.975))
pboot.perc.mu.maleD
```

```
##      2.5%     97.5%
## 0.9442435 0.9536929
```

```
pboot.perc.var.maleD <- quantile(est.maleND[,4], probs = c(0.025, 0.975))
pboot.perc.var.maleD
```

```
##        2.5%       97.5%
## 0.005478385 0.006844812
```

Now let's deal with the case of non-diabetic males.

Let's boostrap for all parameters.

```
est.maleND <- matrix(0, 1000, 4)
for (i in 1:1000) {
  boot <- sample(bmi[data$glyhb < 7 & data$gender == "male"], 1000, replace = T)
  e1 <- mean(boot)
  e2 <- mean(boot^2)
  est.maleND[i,1] <- e1^2/(e2 - e1^2)
  est.maleND[i,2] <- e1/(e2 - e1^2)

boot <- sample(whr[data$glyhb < 7 & data$gender == "male"], 1000, replace = T)
  est.maleND[i,3] <- mean(boot)
  est.maleND[i,4] <- var(boot)
}
```

Now let's find our 95% confidence intervals.

```
pboot.perc.a.maleND <- quantile(est.maleND[,1], probs = c(0.025, 0.975))
pboot.perc.a.maleND
```

```
##     2.5%    97.5%
## 20.77789 24.69890
```

```
pboot.perc.b.maleND <- quantile(est.maleND[,2], probs = c(0.025, 0.975))
pboot.perc.b.maleND
```

```
##      2.5%     97.5%
## 0.7813978 0.9392171
```

```
pboot.perc.mu.maleND <- quantile(est.maleND[,3], probs = c(0.025, 0.975))
pboot.perc.mu.maleND
```

```
##      2.5%     97.5%
## 0.9020969 0.9097333
```

```
pboot.perc.var.maleND <- quantile(est.maleND[,4], probs = c(0.025, 0.975))
pboot.perc.var.maleND
```

```
##        2.5%       97.5%
## 0.003671955 0.004383052
```

Now let's deal with the case of diabetic females.

Let's boostrap for all parameters.

```
est.femaleD <- matrix(0, 1000, 4)
for (i in 1:1000) {
  boot <- sample(bmi[data$glyhb >= 7 & data$gender == "female"], 1000, replace = T)
  e1 <- mean(boot)
  e2 <- mean(boot^2)
  est.femaleD[i,1] <- e1^2/(e2 - e1^2)
  est.femaleD[i,2] <- e1/(e2 - e1^2)

boot <- sample(whr[data$glyhb >= 7 & data$gender == "female"], 1000, replace = T)
  est.femaleD[i,3] <- mean(boot)
  est.femaleD[i,4] <- var(boot)
}
```

Now let's find our 95% confidence intervals.

```
pboot.perc.a.femaleD <- quantile(est.femaleD[,1], probs = c(0.025, 0.975))
pboot.perc.a.femaleD
```

```
##     2.5%    97.5%
## 23.72559 27.87962
```

```
pboot.perc.b.femaleD <- quantile(est.femaleD[,2], probs = c(0.025, 0.975))
pboot.perc.b.femaleD
```

```
##      2.5%     97.5%
## 0.7110028 0.8438933
```

```
pboot.perc.mu.femaleD <- quantile(est.femaleD[,3], probs = c(0.025, 0.975))
pboot.perc.mu.femaleD
```

```
##      2.5%     97.5%
## 0.8903318 0.8971341
```

```
pboot.perc.var.femaleD <- quantile(est.femaleD[,4], probs = c(0.025, 0.975))
pboot.perc.var.femaleD
```

```
##        2.5%       97.5%
## 0.003048528 0.003784395
```

Lastly, let's deal with the case of non-diabetic females.

Let's boostrap for all parameters.

```
est.femaleND <- matrix(0, 1000, 4)
for (i in 1:1000) {
  boot <- sample(bmi[data$glyhb < 7 & data$gender == "female"], 1000, replace = T)
  e1 <- mean(boot)
  e2 <- mean(boot^2)
  est.femaleND[i,1] <- e1^2/(e2 - e1^2)
  est.femaleND[i,2] <- e1/(e2 - e1^2)

boot <- sample(whr[data$glyhb < 7 & data$gender == "female"], 1000, replace = T)
  est.femaleND[i,3] <- mean(boot)
  est.femaleND[i,4] <- var(boot)
}
```

Now let's find our 95% confidence intervals.

```
pboot.perc.a.femaleND <- quantile(est.femaleND[,1], probs = c(0.025, 0.975))
pboot.perc.a.femaleND
```

```
##     2.5%    97.5%
## 16.44322 19.60141
```

```
pboot.perc.b.femaleND <- quantile(est.femaleND[,2], probs = c(0.025, 0.975))
pboot.perc.b.femaleND
```

```
##      2.5%     97.5%
## 0.5502240 0.6594617
```

```
pboot.perc.mu.femaleND <- quantile(est.femaleND[,3], probs = c(0.025, 0.975))
pboot.perc.mu.femaleND
```

```
##      2.5%     97.5%
## 0.8506494 0.8591685
```

```r
pboot.perc.var.femaleND <- quantile(est.femaleND[,4], probs = c(0.025, 0.975))
pboot.perc.var.femaleND
```

```
##        2.5%       97.5%
## 0.004378061 0.005496694
```

Let's create a nice table to compare all our confidence intervals.

```r
all_CI = list(pboot.perc.a.maleD, pboot.perc.b.maleD, pboot.perc.mu.maleD, pboot.perc.var.maleD,
           pboot.perc.a.maleND, pboot.perc.b.maleND, pboot.perc.mu.maleND, pboot.perc.var.maleD,
           pboot.perc.a.femaleD, pboot.perc.b.femaleD, pboot.perc.mu.femaleD, pboot.perc.var.femaleD,
           pboot.perc.a.femaleND, pboot.perc.b.femaleND, pboot.perc.mu.femaleND, pboot.perc.var.femaleNI

# Apply this to convert into desired string
pretty_CI = function(x) {
  paste(toString(round(x[1], digits = 3)), "-", toString(round(x[2], digits = 3)), sep = " ")
}

simple_CI = sapply(all_CI, pretty_CI)
CI_matrix = matrix(simple_CI, nrow = 4, byrow = TRUE)
colnames(CI_matrix) = c("alpha", "beta", "mean", "variance")
rownames(CI_matrix) = c("Diabetic Male", "Non-diabetic Male", "Diabetic Female", "Non-diabetic Female")
CI_matrix
```

```
##                     alpha             beta            mean
## Diabetic Male       "27.119 - 33.242" "0.939 - 1.159" "0.944 - 0.954"
## Non-diabetic Male   "20.778 - 24.699" "0.781 - 0.939" "0.902 - 0.91"
## Diabetic Female     "23.726 - 27.88"  "0.711 - 0.844" "0.89 - 0.897"
## Non-diabetic Female "16.443 - 19.601" "0.55 - 0.659"  "0.851 - 0.859"
##                     variance
## Diabetic Male       "0.005 - 0.007"
## Non-diabetic Male   "0.005 - 0.007"
## Diabetic Female     "0.003 - 0.004"
## Non-diabetic Female "0.004 - 0.005"
```

From the table above, we see that alpha is typically higher among diabetics than non-diabetics. We see that the same occurs for beta and the mean. However, the variance seems the same among all groups.

Part 4: Testing

Question 1:

There are two good ways to test if males and females are equally exposed to Type II diabetes. The first way is the chi-square test of homogeneity.

To do this test, let's first split our data into their 4 respective cells.

```
female.diabetics = length(which(data$gender=="female" & data$glyhb >= 7))
female.nondiabetics = length(which(data$gender=="female" & data$glyhb < 7))

male.diabetics = length(which(data$gender=="male" & data$glyhb >= 7))
male.nondiabetics = length(which(data$gender=="male" & data$glyhb < 7))

gender.mat = matrix(c(male.diabetics, male.nondiabetics, female.diabetics,
                      female.nondiabetics), nrow=2, dimnames=list(c("Diabetic",
                      "Not Diabetic"), c("Male", "Female")))
gender.mat
```

```
##              Male Female
## Diabetic       24     30
## Not Diabetic  125    180
```

Now let's run our chi-square test.

```
chisq.test(gender.mat)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  gender.mat
## X-squared = 0.1062, df = 1, p-value = 0.7445
```

With one degree of freedom and a chi-square statistic of 0.1062, we get a p-value of 0.7445 and fail to reject the null hypothesis that males and females are equally exposed to type-II diabetes.

Now we will use our second way which is an equal means test of 2 proportions.

First we will need to convert data sets into 0s and 1s:

```
female.df = data[data$gender=="female",]
female.diab.status = vector()
  for (i in 1:nrow(female.df)){
  if (female.df$glyhb[i]>=7) {female.diab.status[i]=1
  } else female.diab.status[i]=0
}

male.df = data[data$gender=="male",]
male.diab.status = vector()
for (i in 1:nrow(male.df)){
  if (male.df$glyhb[i]>=7) {male.diab.status[i]=1
  } else male.diab.status[i]=0
}
```

Now we will run the t-test.

```
t.test(female.diab.status, male.diab.status, alternative="two.sided", mu=0,
       paired=FALSE, var.equal=FALSE, conf.level=0.95)
```

```
##
##  Welch Two Sample t-test
##
## data:  female.diab.status and male.diab.status
## t = -0.4705, df = 308.83, p-value = 0.6383
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.09439683  0.05796347
## sample estimates:
## mean of x mean of y
## 0.1428571 0.1610738
```

Using a two-sample t-test for proportions, we get a p-value of 0.6383 and fail to reject the null hypothesis that males and females are equally exposed to type-II diabetes.

So, both of our tests failed to reject the null.

Question 2:

Our five favorite features are: stabilized glucose, age, first systolic blood pressure, BMI, and WHR.

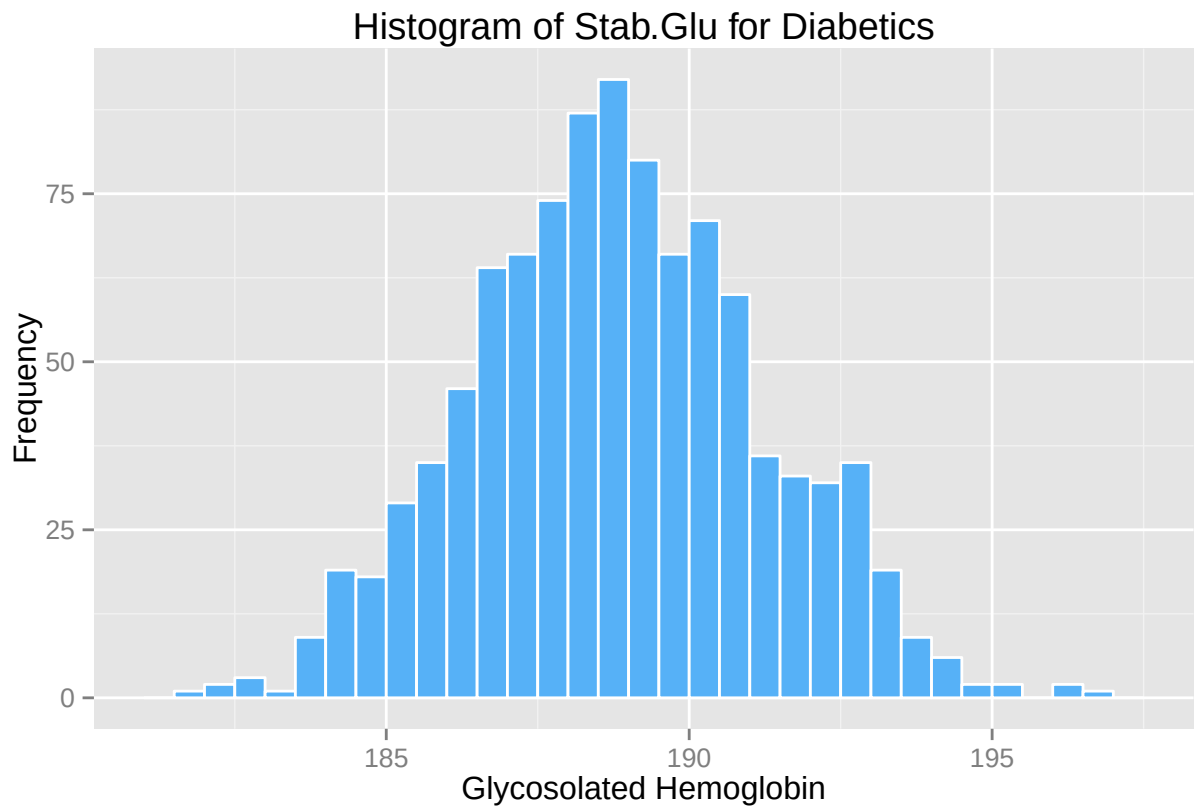
First, let's make vectors subsetted by our features and store them in a list.

```
groupList = list()

# Stabilized glucose level:
stab.gluc.diabetic = data[data$glyhb>=7, 4]
stab.gluc.nondiabetic = data[data$glyhb<7, 4]

groupList[[1]] = stab.gluc.diabetic
groupList[[2]] = stab.gluc.nondiabetic

# Age:
age.diabetic = data[data$glyhb>=7, 9]
age.nondiabetic = data[data$glyhb<7, 9]

groupList[[3]] = age.diabetic
groupList[[4]] = age.nondiabetic

# First systolic blood pressure:
bp.1s.diabetic = data[data$glyhb>=7, 14]
bp.1s.nondiabetic = data[data$glyhb<7, 14]

groupList[[5]] = bp.1s.diabetic
groupList[[6]] = bp.1s.nondiabetic

# BMI:
bmi.diabetic = data[data$glyhb>=7, 20]
bmi.nondiabetic = data[data$glyhb<7, 20]

groupList[[7]] = bmi.diabetic
```

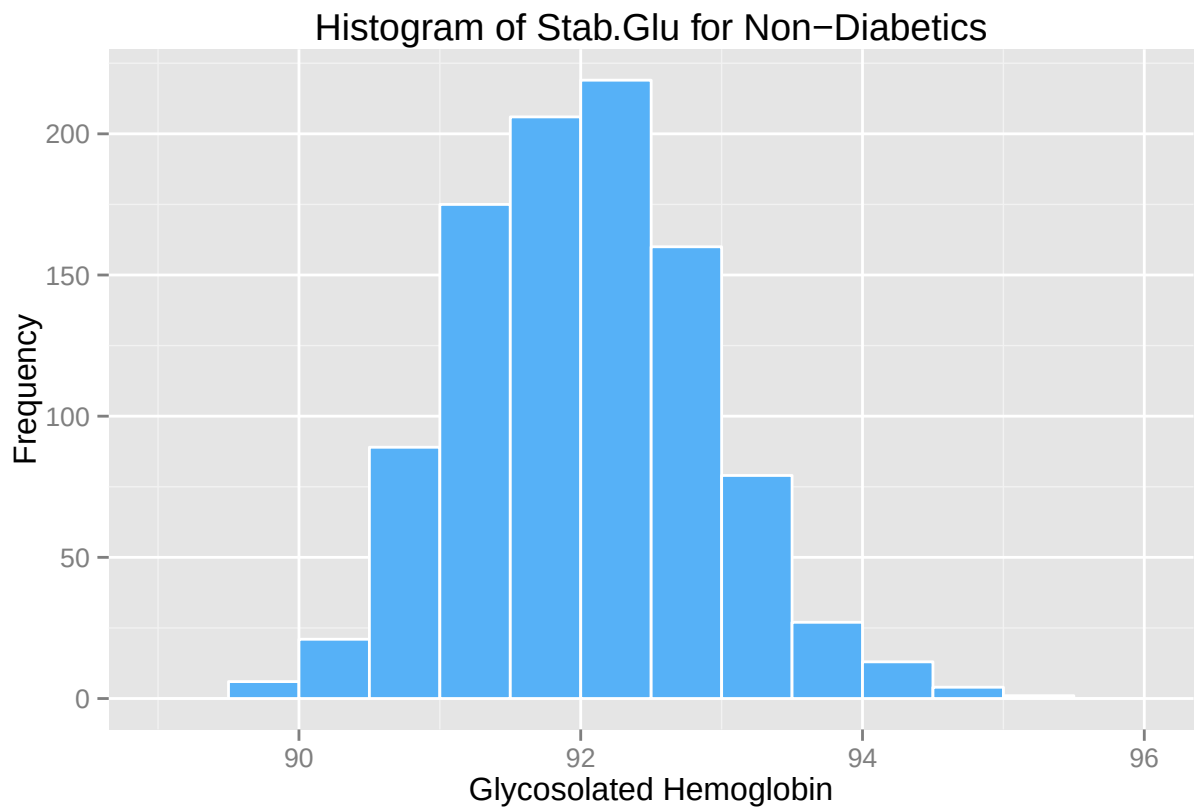
```
groupList[[8]] = bmi.nondiabetic

# WHR:
whr.diabetic = data[data$glyhb>=7, 21]
whr.nondiabetic = data[data$glyhb<7, 20]

groupList[[9]] = whr.diabetic
groupList[[10]] = whr.nondiabetic
```

Let's check for normality of samples means for each group to see if parametric t-tests are appropriate.

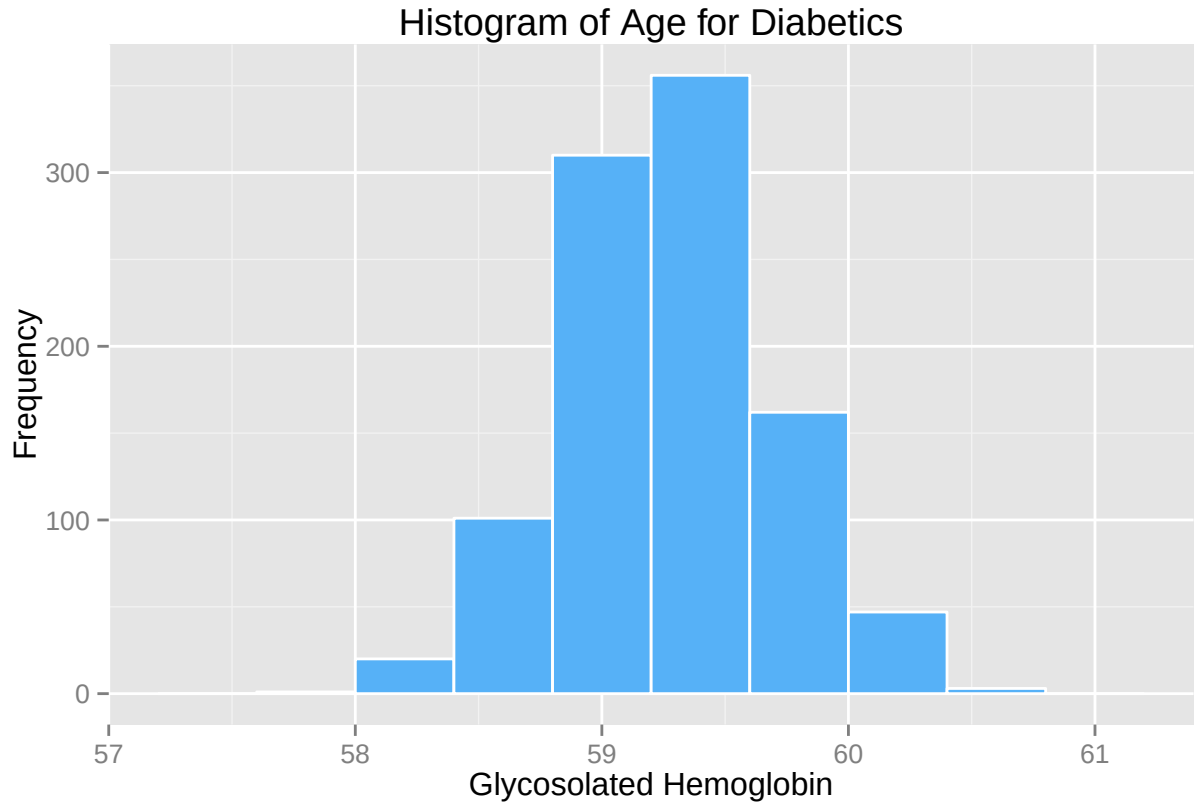
```
plot.boot.hist = function(x){
  boot.pop = replicate(1000, sample(x, 1000, replace=TRUE))
  boot.pop.means = apply(boot.pop, 2, mean)
  return(boot.pop.means)
}

pop_means = data.frame(sapply(groupList, plot.boot.hist))

# Stab.glu of diabetics
ggplot(pop_means) + geom_histogram(aes(x = pop_means[ , 1]), binwidth = 0.5, col = "white", fill = "#56
  ggtitle("Histogram of Stab.Glu for Diabetics") +
  xlab("Glycosolated Hemoglobin") + ylab("Frequency")
```



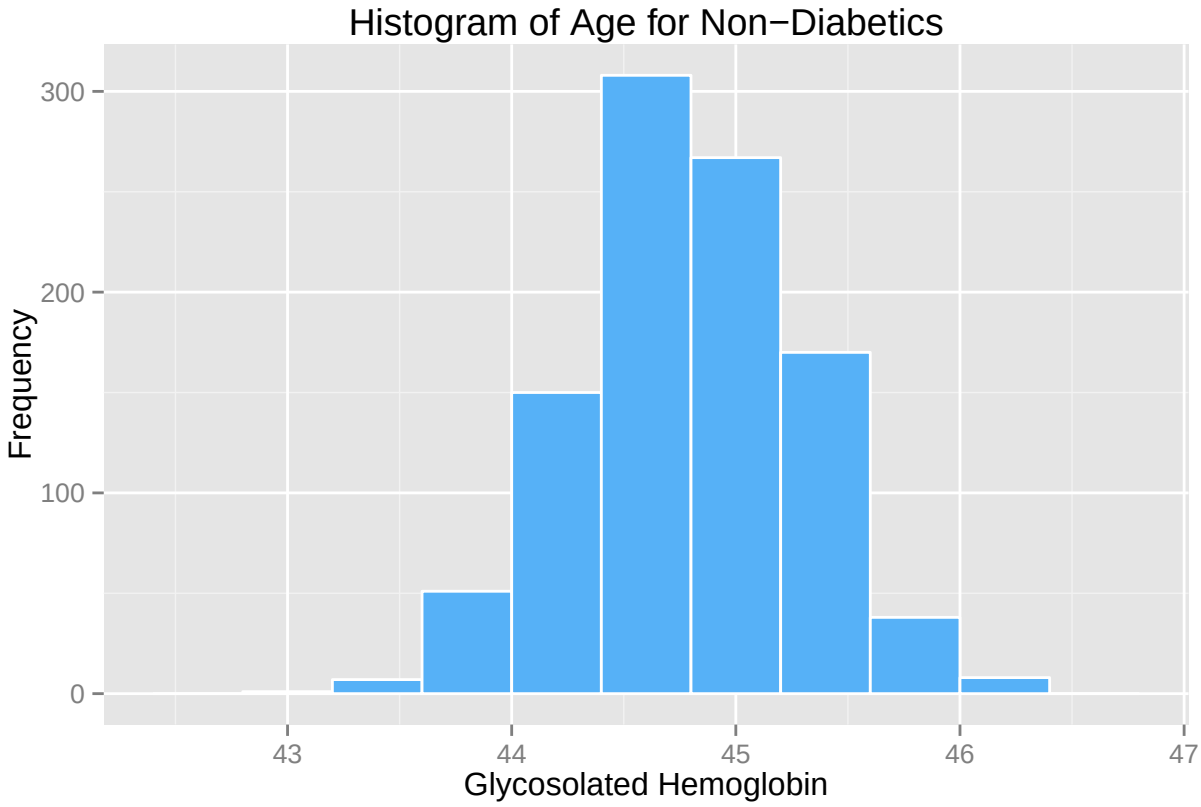
```
# Stab.glu of non-diabetics
ggplot(pop_means) + geom_histogram(aes(x = pop_means[ , 2]), binwidth = 0.5, col = "white", fill = "#56
  ggtitle("Histogram of Stab.Glu for Non-Diabetics") +
  xlab("Glycosolated Hemoglobin") + ylab("Frequency")
```



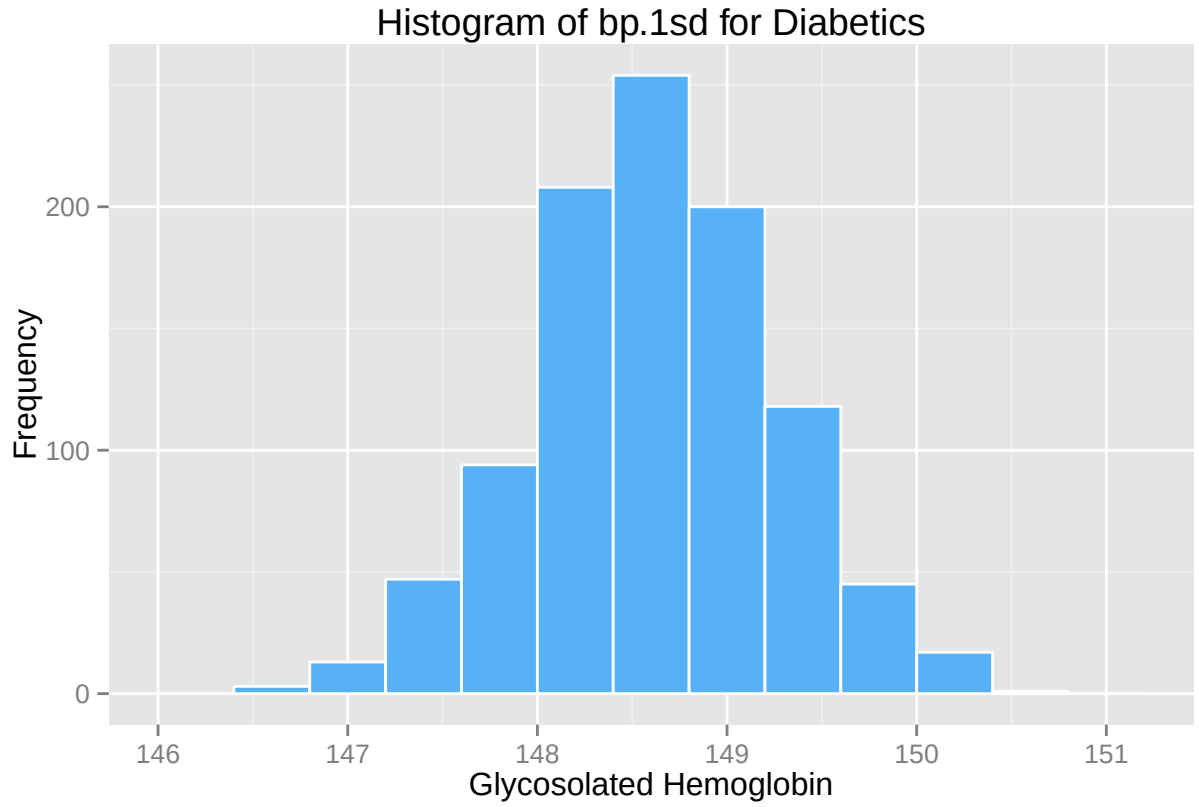

```
# Age of diabetics
ggplot(pop_means) + geom_histogram(aes(x = pop_means[ , 3]), binwidth = 0.4, col = "white", fill = "#56
  ggtitle("Histogram of Age for Diabetics") +
  xlab("Glycosolated Hemoglobin") + ylab("Frequency")
```



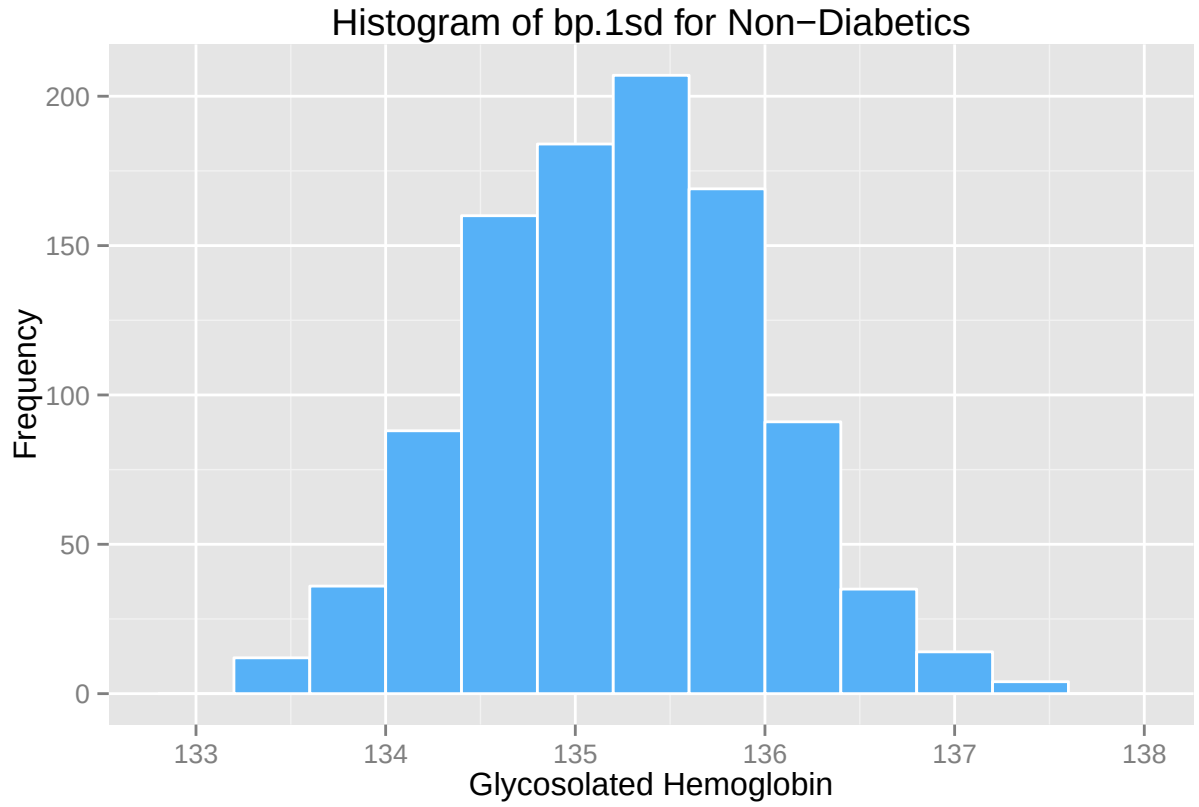

```
# Age of non-diabetics
ggplot(pop_means) + geom_histogram(aes(x = pop_means[ , 4]), binwidth = 0.4, col = "white", fill = "#56
  ggtitle("Histogram of Age for Non-Diabetics") +
  xlab("Glycosolated Hemoglobin") + ylab("Frequency")
```



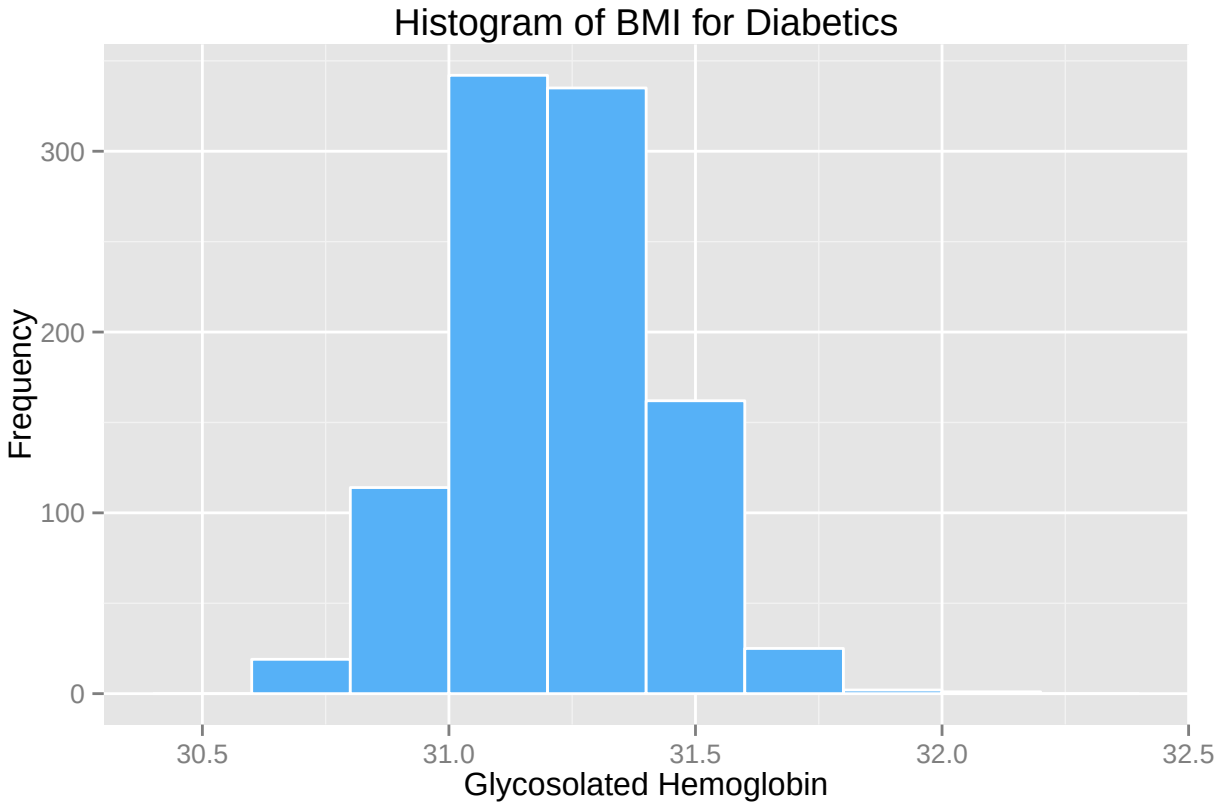

```
# bp.1sd of diabetics
ggplot(pop_means) + geom_histogram(aes(x = pop_means[ , 5]), binwidth = 0.4, col = "white", fill = "#56
  ggtitle("Histogram of bp.1sd for Diabetics") +
  xlab("Glycosolated Hemoglobin") + ylab("Frequency")
```



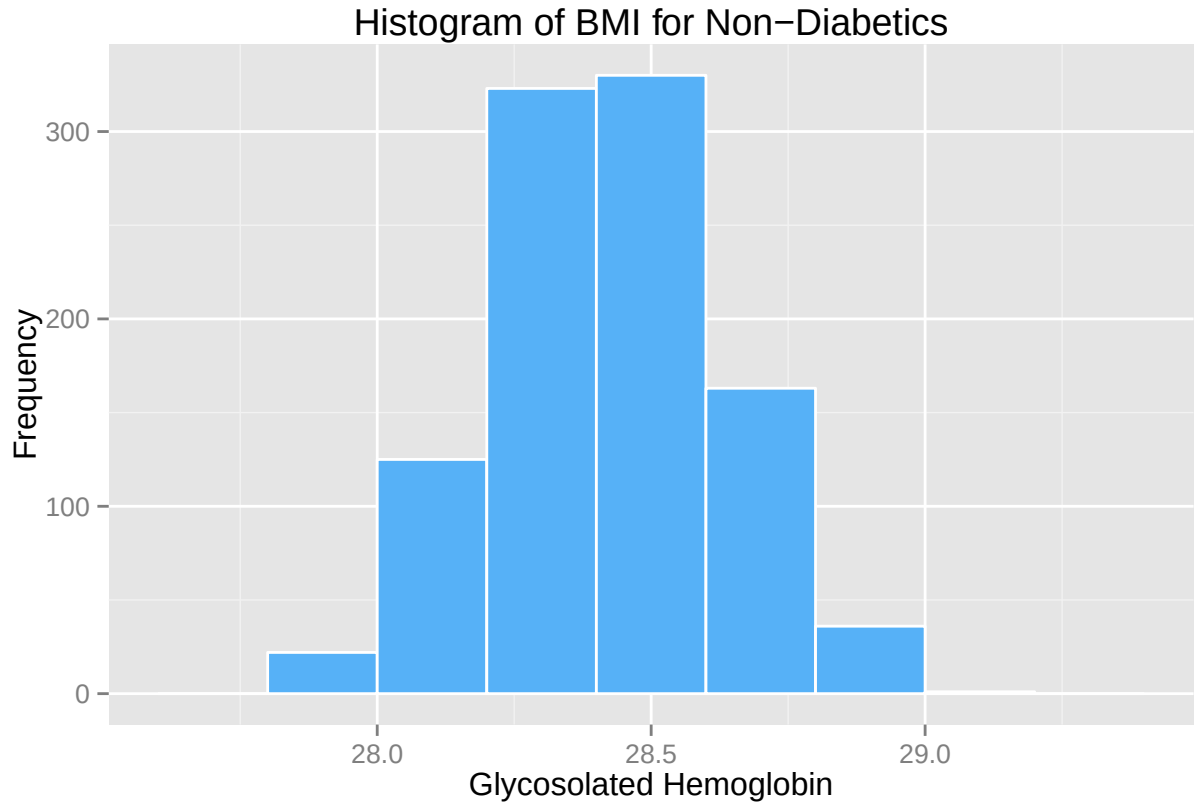

```
# bp.1sd of non-diabetics
ggplot(pop_means) + geom_histogram(aes(x = pop_means[ , 6]), binwidth = 0.4, col = "white", fill = "#56
  ggtitle("Histogram of bp.1sd for Non-Diabetics") +
  xlab("Glycosolated Hemoglobin") + ylab("Frequency")
```



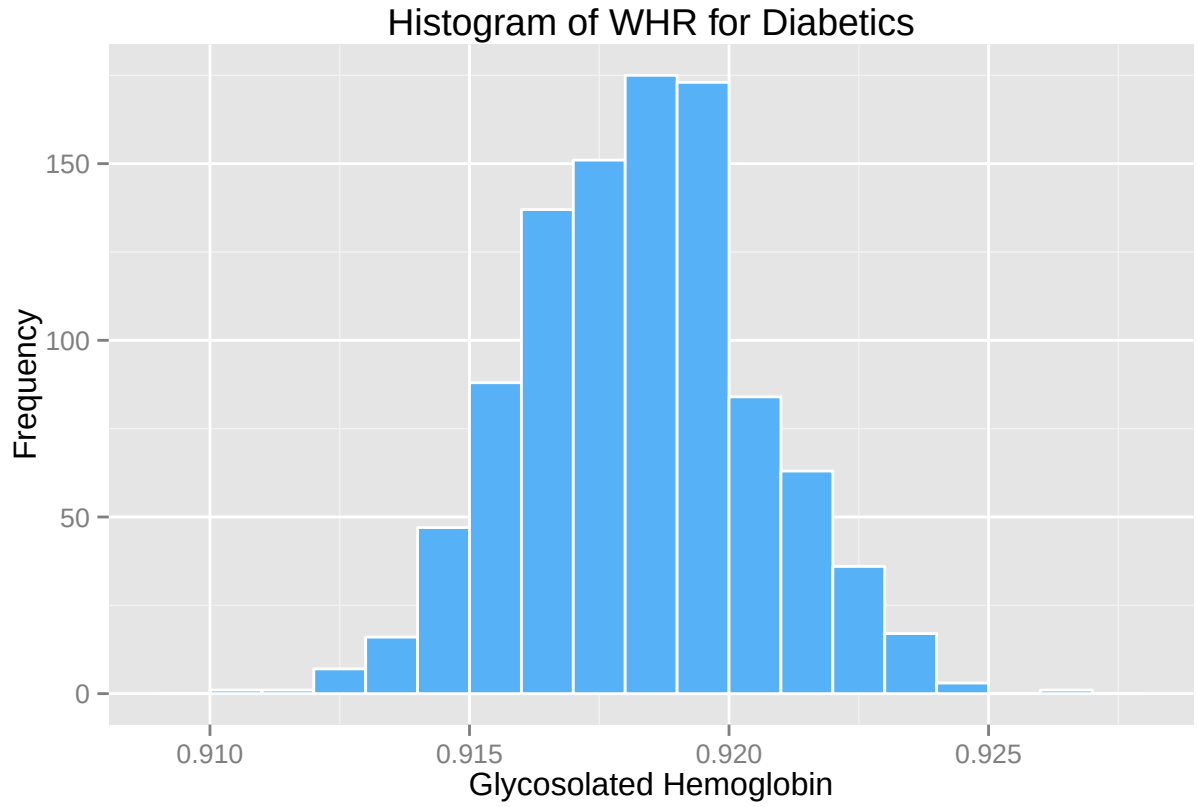

```
# BMI of diabetics
ggplot(pop_means) + geom_histogram(aes(x = pop_means[ , 7]), binwidth = 0.2, col = "white", fill = "#56
  ggtitle("Histogram of BMI for Diabetics") +
  xlab("Glycosolated Hemoglobin") + ylab("Frequency")
```



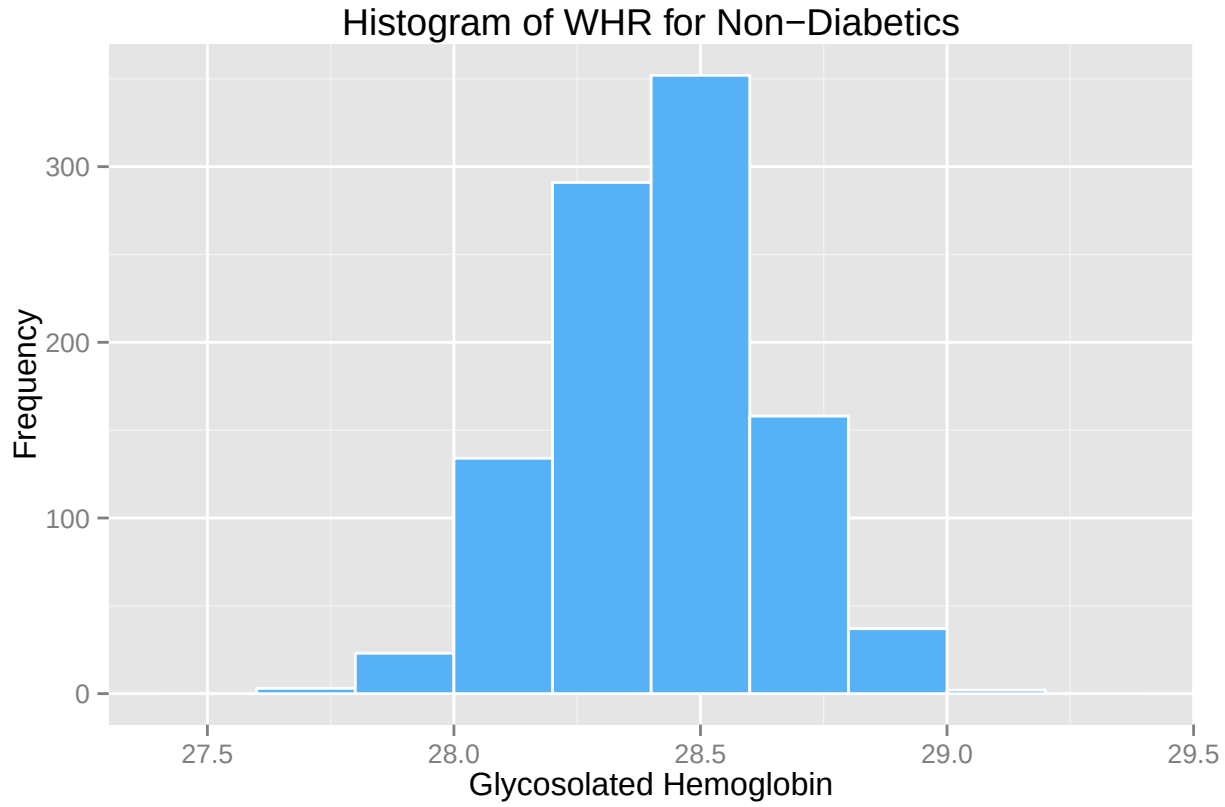

```
# BMI of non-diabetics
ggplot(pop_means) + geom_histogram(aes(x = pop_means[ , 8]), binwidth = 0.2, col = "white", fill = "#56
  ggtitle("Histogram of BMI for Non-Diabetics") +
  xlab("Glycosolated Hemoglobin") + ylab("Frequency")
```




```
# WHR of diabetics
ggplot(pop_means) + geom_histogram(aes(x = pop_means[ , 9]), binwidth = 0.001, col = "white", fill = "#
  ggtitle("Histogram of WHR for Diabetics") +
  xlab("Glycosolated Hemoglobin") + ylab("Frequency")
```




```
# WHR of non-diabetics
ggplot(pop_means) + geom_histogram(aes(x = pop_means[ , 10]), binwidth = 0.20, col = "white", fill = "#
  ggtitle("Histogram of WHR for Non-Diabetics") +
  xlab("Glycosolated Hemoglobin") + ylab("Frequency")
```



All of these histograms look approximately normal, so we can use t-tests.

```
t.test(stab.gluc.diabetic, stab.gluc.nondiabetic, alternative="two.sided", mu=0,
       paired=FALSE, var.equal=FALSE, conf.level=0.95)
```

```
##
##  Welch Two Sample t-test
##
## data:  stab.gluc.diabetic and stab.gluc.nondiabetic
## t = 9.3067, df = 55.536, p-value = 6.257e-13
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   75.99568 117.69455
## sample estimates:
## mean of x mean of y
##  188.9074   92.0623
```

```
t.test(age.diabetic, age.nondiabetic, alternative="two.sided", mu=0,
       paired=FALSE, var.equal=FALSE, conf.level=0.95)
```

```
##
##  Welch Two Sample t-test
##
## data:  age.diabetic and age.nondiabetic
## t = 7.1981, df = 84.628, p-value = 2.302e-10
```

```
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  10.49031 18.49803
## sample estimates:
## mean of x mean of y
##  59.27778  44.78361
```

```
t.test(bp.1s.diabetic, bp.1s.nondiabetic, alternative="two.sided", mu=0,
       paired=FALSE, var.equal=FALSE, conf.level=0.95)
```

```
##
##  Welch Two Sample t-test
##
## data:  bp.1s.diabetic and bp.1s.nondiabetic
## t = 4.3315, df = 77.595, p-value = 4.384e-05
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   7.24591 19.57400
## sample estimates:
## mean of x mean of y
##  148.6296  135.2197
```

```
t.test(bmi.diabetic, bmi.nondiabetic, alternative="two.sided", mu=0,
       paired=FALSE, var.equal=FALSE, conf.level=0.95)
```

```
##
##  Welch Two Sample t-test
##
## data:  bmi.diabetic and bmi.nondiabetic
## t = 2.919, df = 74.596, p-value = 0.00464
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.8891927 4.7122889
## sample estimates:
## mean of x mean of y
##  31.22315  28.42241
```

```
t.test(whr.diabetic, whr.nondiabetic, alternative="two.sided", mu=0,
       paired=FALSE, var.equal=FALSE, conf.level=0.95)
```

```
##
##  Welch Two Sample t-test
##
## data:  whr.diabetic and whr.nondiabetic
## t = -71.6995, df = 304.418, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -28.25904 -26.74934
## sample estimates:
##  mean of x  mean of y
##  0.9182154 28.4224085
```

For all of these features, we reject the null hypothesis that the means for those with and for those without diabetes are equal.

Question 3:

First let's focus on pi_bmi and find it from our data.

```
X <- data$BMI[data$gender == "male" & data$glyhb <7]
Y <- data$BMI[data$gender == "male" & data$glyhb >=7]
n <- length(X)
m <- length(Y)
all.ranks <- rank(c(Y, X))
R.prime <- sum(all.ranks[1:length(Y)])
pi.bmi <- 1/(n*m)*(R.prime - m*(m + 1)/2)
pi.bmi
```

```
## [1] 0.6335
```

Here's the built-in way to do it in R.

```
data$diff <- data$gender == "male" & data$glyhb <7
wilcox.test(bmi ~ diff, data=data)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  bmi by diff
## W = 19420.5, p-value = 3.075e-07
## alternative hypothesis: true location shift is not equal to 0
```

Now let's use nonparametric bootstrap to generate values of pi_bmi.

```
boot <- 1:1000
for (i in 1:1000) {
  smp <- sample(data$BMI[data$gender == "male" & data$glyhb <7], 100, replace =T)
  smp2 <- sample(data$BMI[data$gender == "male" & data$glyhb >=7], 100, replace =T)
  all.ranks <- rank(c(smp2, smp))
  R.prime <- sum(all.ranks[1:length(smp2)])
  boot[i] <- 1/(100*100)*(R.prime - 100*(100 + 1)/2)
}
```

Now let's make our 95% confidence interval.

```
pboot.perc.pi.bmi <- quantile(boot, probs = c(0.025, 0.975))
pboot.perc.pi.bmi
```

```
##      2.5%     97.5%
## 0.5570938 0.7152513
```

Now let's focus on pi_whr. Let's first find it from our data.

```
X <- data$WHR[data$gender == "male" & data$glyhb <7]
Y <- data$WHR[data$gender == "male" & data$glyhb >=7]
n <- length(X)
m <- length(Y)
all.ranks <- rank(c(Y, X))
R.prime <- sum(all.ranks[1:length(Y)])
pi.whr <- 1/(n*m)*(R.prime - m*(m + 1)/2)
pi.whr
```

```
## [1] 0.6885
```

Now let's use nonparametric bootstrap to generate values of pi_whr.

```
boot <- 1:1000
for (i in 1:1000) {
  smp <- sample(data$WHR[data$gender == "male" & data$glyhb <7], 100, replace =T)
  smp2 <- sample(data$WHR[data$gender == "male" & data$glyhb >=7], 100, replace =T)
  all.ranks <- rank(c(smp2, smp))
  R.prime <- sum(all.ranks[1:length(smp2)])
  boot[i] <- 1/(100*100)*(R.prime - 100*(100 + 1)/2)
}
```

Lastly, let's make our 95% confidence interval.

```
pboot.perc.pi.whr <- quantile(boot, probs = c(0.025, 0.975))
pboot.perc.pi.whr
```

```
##      2.5%     97.5%
## 0.6099438 0.7580137
```

Question 4:

We have two distributions, one of WHR for glyhb >= 7 and the other of WHR for glyhb < 7. The first one will be our null hypothesis H0 and the second will be our alternative hypothesis H1.

We are given the the WHR value for a new male patient. Our test will look something of the form: "If the man's WHR >= threshold, then the man is diabetic." If alpha = 0.05, P_H0 (test rejects) = 0.05. So, in other words, P_H0 (wHR_man > threshold) = 0.05. This is simple a matter of finding a quantile now. Namely, the 95% quantile.

So first, we must find our vector for the null hypothesis.

```
male.WHRs.null = male.df[male.df$glyhb < 7,]$waist/male.df[male.df$glyhb < 7,]$hip
male.WHRs.null
```

```
##   [1] 0.8684211 0.8571429 0.9387755 0.8500000 0.9000000 0.7948718 0.8823529
##   [8] 1.0243902 0.9782609 0.9148936 0.9487179 0.8285714 0.9411765 0.8857143
##  [15] 0.8611111 0.9523810 0.9024390 0.9268293 0.8500000 1.0512821 0.8857143
##  [22] 0.9000000 0.9761905 0.9268293 0.9024390 0.8780488 0.8500000 0.8372093
```

```
##  [29] 0.8717949 0.9000000 0.9756098 0.9318182 0.8157895 0.8717949 0.7968750
##  [36] 0.7750000 0.9142857 0.7948718 0.8000000 0.8947368 0.8837209 0.8461538
##  [43] 0.8787879 1.0408163 0.8055556 0.8604651 0.9512195 0.8974359 0.8888889
##  [50] 0.8965517 0.9500000 1.0270270 1.0454545 0.9782609 0.8292683 0.8409091
##  [57] 1.0476190 0.8604651 0.8372093 0.8604651 1.0638298 0.9696970 0.9375000
##  [64] 0.9705882 0.8787879 0.8684211 0.8285714 0.8571429 0.9268293 0.8974359
##  [71] 0.9302326 0.9024390 0.8809524 0.9459459 0.9047619 0.9743590 0.9756098
##  [78] 0.9795918 0.8974359 0.9230769 0.8888889 0.9230769 0.9024390 0.8431373
##  [85] 1.0263158 1.0769231 1.0000000 1.0204082 0.9512195 0.9183673 0.9047619
##  [92] 0.8717949 0.9000000 0.7948718 0.9250000 0.9318182 0.9361702 0.8421053
##  [99] 0.8684211 0.8717949 0.8378378 0.8823529 0.8684211 0.8974359 0.8536585
## [106] 0.9210526 0.9756098 0.9189189 0.8863636 0.9512195 0.8648649 0.9411765
## [113] 0.9000000 0.7857143 0.9347826 0.9210526 0.9761905 0.9791667 0.9523810
## [120] 0.8918919 0.8108108 0.8837209 0.8181818 0.9090909 0.8750000
```

Now we will find the "threshold."

```
threshold = quantile(male.WHRs.null, prob=0.95)
threshold
```

```
##      95%
## 1.026885
```

Now we focus on finding the power of our test. Power is simply equal to 1 - beta. So we must find beta first. Beta is simply beta = P_H1 (WHR_man < threshold). We can simply plug in the threshold we found above, and easily find beta.

```
male.WHRs.alt = male.df[male.df$glyhb >= 7,]$waist/male.df[male.df$glyhb >= 7,]$hip
male.WHRs.alt
```

```
##  [1] 1.0731707 0.9512195 0.9375000 0.9767442 0.9767442 0.9512195 0.9361702
##  [8] 1.0888889 0.9250000 1.1428571 0.9777778 0.9230769 0.9024390 1.0000000
## [15] 0.8936170 0.9743590 0.7500000 0.9750000 0.8809524 0.8775510 0.8461538
## [22] 0.9111111 0.9555556 0.9473684
```

```
beta = length(which(male.WHRs.alt < threshold))/length(male.WHRs.alt)
beta
```

```
## [1] 0.875
```

Now let's plug in to find the power.

```
power = 1 - beta
power
```

```
## [1] 0.125
```

Question 5:

First let's split our data by the interval specified by Table 1.

```
res <- 1:359
i = 1
for (bm in data$BMI) {
  if (bm < 18.5) {
    res[i] = 1
  } else if (bm >= 18.5 & bm < 25) {
    res[i] = 2
  } else if (bm >= 25 & bm < 30) {
    res[i] = 3
  } else if (bm >= 30 & bm < 35) {
    res[i] = 4
  } else if (bm >= 35 & bm < 40) {
    res[i] = 5
  } else {
    res[i] = 6
  }
  i = i + 1
}
data$bmicat <- res
bmicat.mat <- matrix(0, 2, 6)

for (i in 1:359) {
  bmicat.mat[as.numeric(data$gender)[i] - 1, data$bmicat[i]] = bmicat.mat[as.numeric(data$gender)[i] - 
}
```

Now, let's run a chi-square test of homogeneity.

```
chisq.test(bmicat.mat)
```

```
## Warning in chisq.test(bmicat.mat): Chi-squared approximation may be
## incorrect

##
##  Pearson's Chi-squared test
##
## data:  bmicat.mat
## X-squared = 25.3523, df = 5, p-value = 0.0001191
```

Our p-value, 0.0001191, is much less than 0.01. So, our results our very significant and we reject the null hypothesis which states that the distributions of BMI among the genders are the same.

Now let's split our data according to Table 2.

```
res <- 1:359
i = 1
for (wh in data$WHR) {
  ag = data$age[i]
  if (
      ((data$gender[i] == "male") & ((wh < 0.83 & ag < 30) | (wh < 0.84 & ag < 40 & ag >= 30) | (wh < 0
      ((data$gender[i] == "female") & ((wh < 0.71 & ag < 30) | (wh < 0.72 & ag < 40 & ag >= 30) | (wh <
    )
  {
    res[i] = 1
```

```
  } else if (
      ((data$gender[i] == "male") & ((wh >= 0.83 & wh < 0.89 & ag < 30) | (wh >= 0.84 & wh < 0.92 & ag <
      ((data$gender[i] == "female") & ((wh >= 0.71 & wh < 0.78 & ag < 30) | (wh >= 0.72 & wh < 0.79 & ag
    )
  {
    res[i] = 2
  } else if (
    ((data$gender[i] == "male") & ((wh > 0.94 & ag < 30) | (wh > 0.96 & ag < 40 & ag >= 30) | (wh > 1.0
    ((data$gender[i] == "female") & ((wh > 0.82 & ag < 30) | (wh > 0.84 & ag < 40 & ag >= 30) | (wh > 0
  )
  {
    res[i] = 4
  } else {
    res[i] = 3
  }
  i = i + 1
}
data$whrcat <- res
whrcat.mat <- as.matrix(table(data$gender, data$whrcat)[2:3, ])
```

Now we can run another chi-square test of homogeneity.

```
chisq.test(whrcat.mat)
```

```
##
##  Pearson's Chi-squared test
##
## data:  whrcat.mat
## X-squared = 125.1513, df = 3, p-value < 2.2e-16
```

Our p-value, whic is $< 2.2e\text{-}16$, is much less than 0.01. So, our results our very significant and we reject the null hypothesis which states that the distributions of WHR among the genders are the same.

Question 6:

To test both features, and to see if they interact, let's run a Two-Way ANOVA test.

```
results = aov(glyhb ~ bmicat * whrcat, data = data)
summary(results)
```

```
##                Df Sum Sq Mean Sq F value Pr(>F)
## bmicat          1   24.5  24.459   5.427 0.0204 *
## whrcat          1    1.0   0.965   0.214 0.6438
## bmicat:whrcat   1    0.1   0.096   0.021 0.8838
## Residuals     355 1600.0   4.507
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

With a p-value of 0.8838, we fail to reject the null which states that the features do not interact. BMI was significant while WHR was not. BMI is more sensitive to glyhb because it has more variability across the groups.

This is consistent with the answer of 3.3. If we look at the confidence intervals in 3.3, we observe that there is a greater difference in the ranges that we get for diabetics and non-diabetics for BMI than for WHR.

Part 5: Regression

Question 1

First let's create our linear model. The most relevant features we selected were: Stabilized Glucose (stab.glu), Age (age), First Systolic Blood Pressure (bp.1s), Waist to Hip Ratio (WHR), & Body Mass Index (BMI).

```
lm.yhat = lm(glyhb ~ stab.glu + age + bp.1s + WHR + BMI, data = data)
```

Now let's use the predict function to find our predicted values of glyhb.

```
features_data = data[ , c("stab.glu", "age", "bp.1s", "WHR", "BMI")]
y_hat = predict(lm.yhat, data = features_data)
```

Let's convert our predictions and our actual glyhb values over to logical vectors. This will help us compute the predicted error rate.

Let's first convert the predicted values.

```
y_hat_diabetes = c()
y_hat_diabetes[y_hat >= 7] = TRUE
y_hat_diabetes[y_hat < 7] = FALSE
data$y_hat_diabetes = y_hat_diabetes
```

Now let's convert the actual values.

```
y_actual_diabetes = c()
y_actual_diabetes[glyhb >= 7] = TRUE
y_actual_diabetes[glyhb < 7] = FALSE
data$y_actual_diabetes = y_actual_diabetes
```

Lastly, let's compute our predictive error rate.

```
error_table = table(y_hat_diabetes, y_actual_diabetes)
error_table
```

```
##               y_actual_diabetes
## y_hat_diabetes FALSE TRUE
##          FALSE   297   19
##          TRUE      8   35
```

```
pred_error = (error_table[2, 1] + error_table[1, 2]) / sum(error_table)
pred_error
```

```
## [1] 0.07520891
```

Question 2

Let's first compute the False Positives Rate. we can use our error_table variable to easily do this.

```
false_positive = error_table[2, 1] / sum(error_table[ , 1])
false_positive
```

```
## [1] 0.02622951
```

Now let's compute the False Negatives Rate.

```
false_negative = error_table[1, 2] / sum(error_table[ , 2])
false_negative
```

```
## [1] 0.3518519
```

To modify lambda such that our False Negatives Rate is at most 10%, we need to find the threshold such that P(y_hat <= threshold | glyhb >= 7) <= 0.1. The threshold which fulfills this will be our modified lambda and is simply the 10th quantile.

First let's find the y_hat for the set of people who truly have diabetes.

```
all = y_hat[glyhb >= 7]
```

Now let's find the 10th quantile of this to find our threshold, and thereby find our new lambda.

```
lambda_new = quantile(all, probs = 0.1)
lambda_new
```

```
##      10%
## 5.862622
```

Now let's find our False Positive Rate with this new lambda value.

```
FPR_new = mean(y_hat[glyhb < 7] >= lambda_new)
FPR_new
```

```
## [1] 0.1311475
```

Our False Positives Rate now increases. This makes sense. In essense, our False Positive Rate is our Type I error, and our False Negative Rate is our Type II error. When we decreased our False Negative Rate, we were decreasing our Type II Error, which in turn increases our Type I error, which increases our False Positive Rate.

Question 3

First let's find the features with the largest influence. To do this, we fill first need to put all our features on the same scale, so that large magnitudes across the features all follow the same distribution. We will standardize each feature so that each one follows a N(0,1) distribution.

```
standard_features = cbind(1, apply(features_data, 2, scale))
```

To test for the largest influence, we will run a t-test testing that each beta coefficient is zero. To do this, we need to first find sigma hat squared.

```
X = standard_features
Y = glyhb
XTX = t(X)%*%X
XTX.inv = solve(XTX)
RSS = t(Y)%*%Y - t(Y) %*% X %*% XTX.inv %*% t(X) %*%Y
RSS
```

```
##          [,1]
## [1,] 739.1836
```

```
n = nrow(data)
p = 6
sigma2.hat = RSS/(n-p)
sigma2.hat
```

```
##          [,1]
## [1,] 2.094004
```

Now that we have sigma hat squared, we can find the t statistics for each beta hat coefficient.

```
beta.hat = XTX.inv%*%t(X)%*%Y
beta.hat
```

```
##               [,1]
##          5.5695265
## stab.glu 1.3965997
## age      0.2743039
## bp.1s    0.0835899
## WHR      0.1031767
## BMI      0.1008851
```

```
t_stat = c()
for (i in 1:5){
  t_stat[i] = abs(beta.hat[i + 1] - 0)/(sqrt(sigma2.hat * XTX.inv[i + 1, i + 1]))
}
```

Lastly, we can find our p-values for each coefficient.

```r
p_vals = 2*(1 - pt(t_stat, n-p))
p_vals
```

```
## [1] 0.000000000 0.003033552 0.336520758 0.204955625 0.199432521
```

Our first two features, stab.glu and age, are significant and for each we reject the null which states that that particular feature has no effect. In other words, stab.glu and age are two features that we should absolutely keep in our model!

For the other three features, bp.1s, WHR, & BMI, we fail to reject the null and cannot say that b_feature != 0.

Question 4

We did not conclude that both ratios interact. If we had, we could use this information to improve our regression. We would do so by using an asterisk instead of a plus sign between BMI and WHR in our lm function, which would add a term to the regression for the interaction effect.

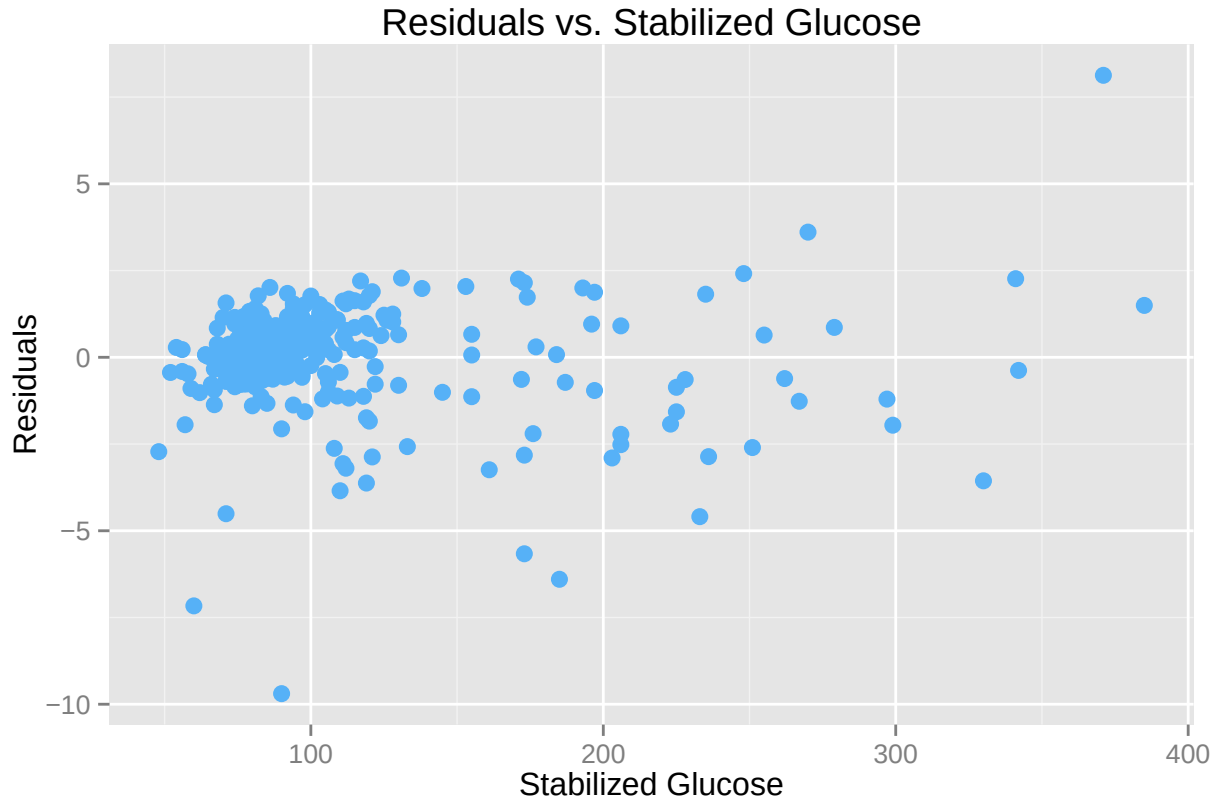
Question 5

First let's find the residuals.

```r
res = y_hat - glyhb
```

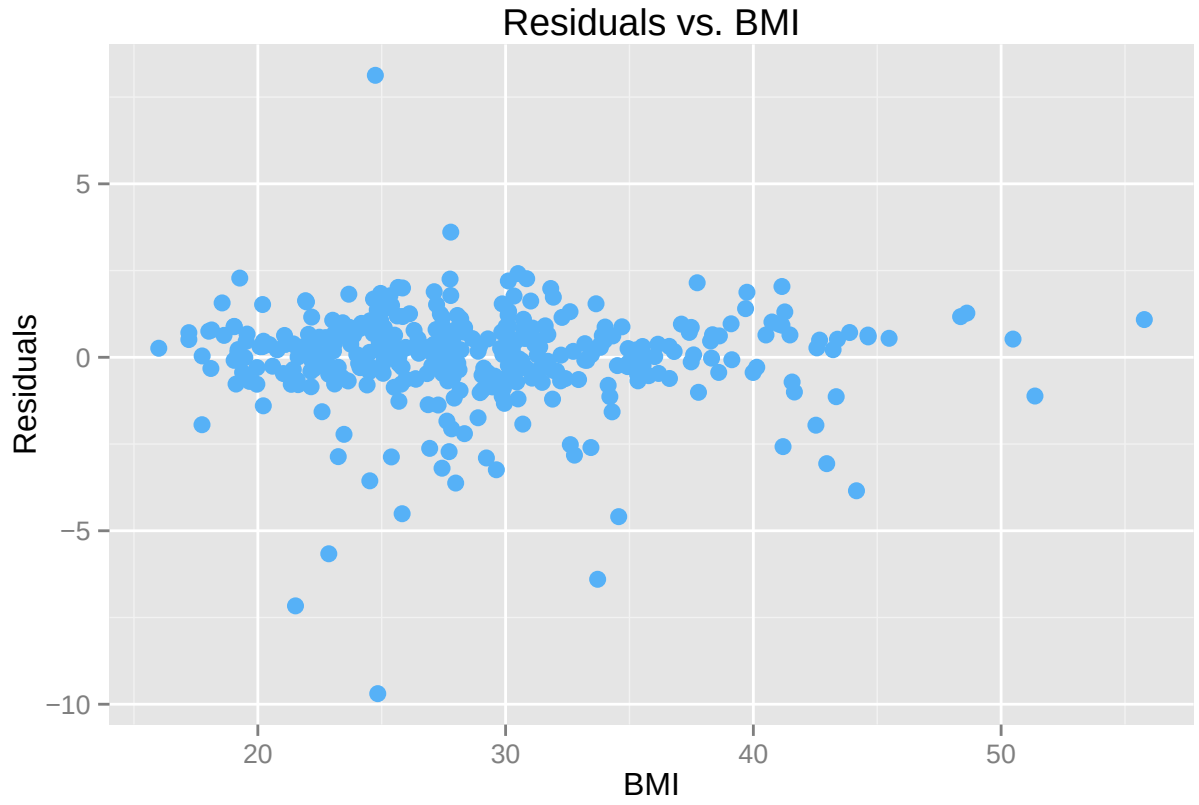
Now let's make our three plots. First let's plot the residuals as a function of stab.glu.

```r
res_data = cbind(res = res, features_data)
ggplot(res_data, aes(x = stab.glu, y = res)) +
    geom_point(shape = 16, col = "#56B1F7", size = 3) +
    xlab("Stabilized Glucose") +
    ylab("Residuals") +
    ggtitle("Residuals vs. Stabilized Glucose")
```



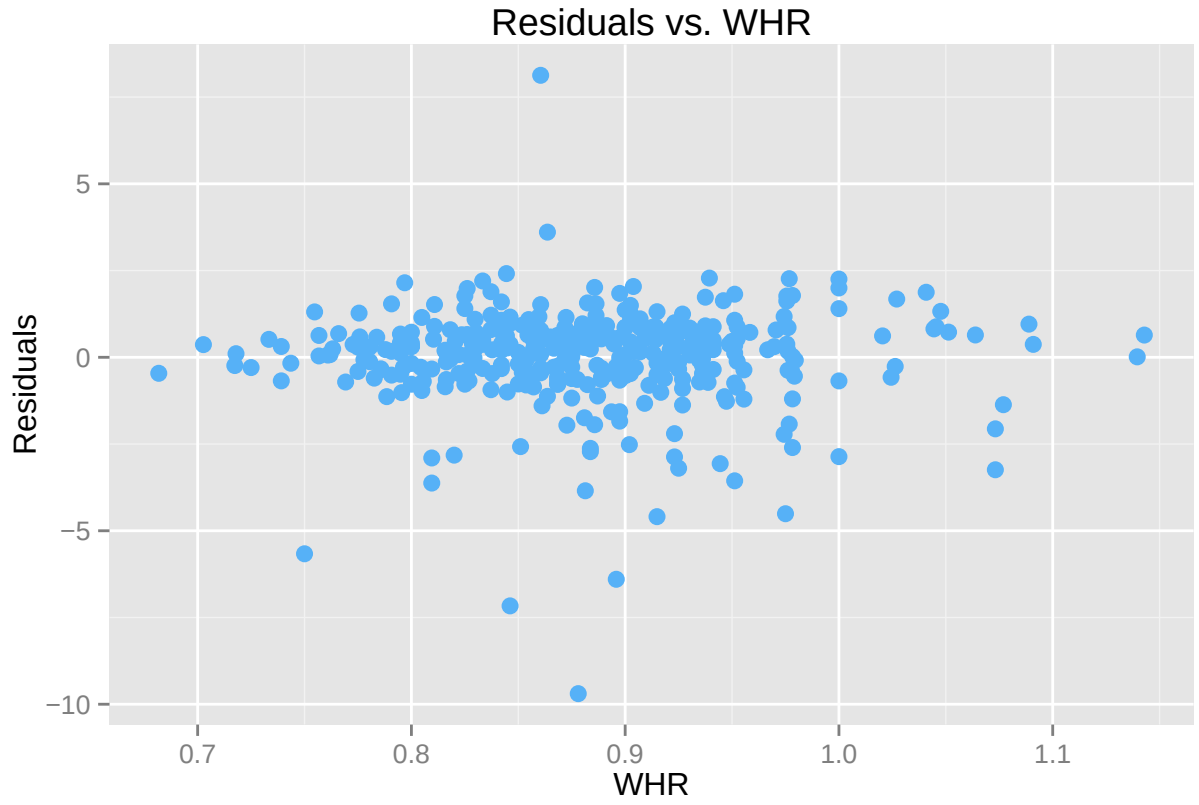
Now let's plot the residuals as a function of BMI.

```
ggplot(res_data, aes(x = BMI, y = res)) +
    geom_point(shape = 16, col = "#56B1F7", size = 3) +
    xlab("BMI") +
    ylab("Residuals") +
    ggtitle("Residuals vs. BMI")
```



Lastly, let's plot the residuals as a function of WHR.

```
ggplot(res_data, aes(x = WHR, y = res)) +
    geom_point(shape = 16, col = "#56B1F7", size = 3) +
    xlab("WHR") +
    ylab("Residuals") +
    ggtitle("Residuals vs. WHR")
```



For the first residual plot, our residuals seem a bit clustered, but then expand farther away as we move to the right on the x axis.

The second and third residual plots seem to have random scatter, and a somewhat constant variance.

However, maybe a log transformation will help stabilize the variance more and help it become constant.

Let's build our log linear model.

```
lm.yhat.log = lm(log(glyhb) ~ log(stab.glu) + log(age) + log(bp.1s) + log(WHR) + log(BMI), data = data)
```

Now let's use the predict function to find our predicted values of glyhb.

```
features_data = data[ , c("stab.glu", "age", "bp.1s", "WHR", "BMI")]
y_hat = exp(predict(lm.yhat.log, data = features_data))
y_hat
```

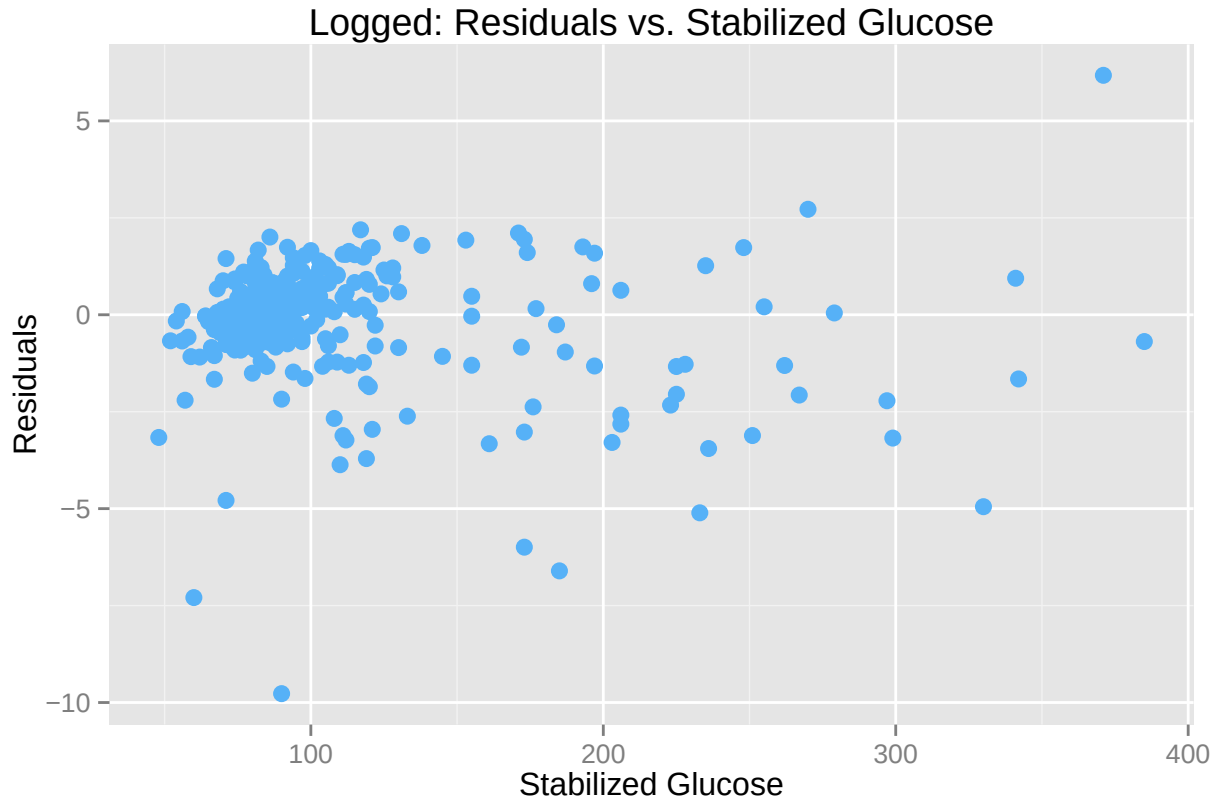
```
##         1         2         3         4         5         6         7
##  4.521701  5.116011  5.638496  5.128257  5.543418  4.909106  4.971664
##         8         9        10        11        12        13        14
##  4.913444  5.212364  4.927719  5.947505  4.309928  4.403137  4.627878
##        15        16        17        18        19        20        21
##  4.745432  4.806757  4.470288  8.419311  5.797727  4.493885  9.560162
##        22        23        24        25        26        27        28
##  5.804367  3.764457  4.348418  5.023796  4.703380  4.574936  5.415323
##        29        30        31        32        33        34        35
##  4.786087 10.570526  5.473715  4.464577  4.852034  4.683789  4.703135
```

```
##        36        37        38        39        40        41        42
##  8.540742  4.232926  4.558824  4.845743  5.047362  3.738403  7.889899
##        43        44        45        46        47        48        49
##  8.570796  4.807213  4.739586  5.195445  4.751455  5.734646  5.092157
##        50        51        52        53        54        55        56
##  6.273079  5.131750  6.407333  6.100533  8.592179  9.621572  4.419988
##        57        58        59        60        61        62        63
##  7.705122  4.227713  4.006703  8.117116  6.533593  8.141321  5.276117
##        64        65        66        67        68        69        70
##  4.656899  3.926398  4.848022  3.906121  6.137977  5.548388  5.586695
##        71        72        73        74        75        76        77
##  4.394420  5.482542  4.960217  5.657576  4.873878  4.942759  5.050519
##        78        79        80        81        82        83        84
##  5.145651  4.640238  5.413706  4.685458  4.805336  7.848863  5.304046
##        85        86        87        88        89        90        91
##  6.223822  3.749321  4.130785  4.618293  5.934773  5.350415 11.091783
##        92        93        94        95        96        97        98
##  4.192235  4.561438  6.030931  5.124270  7.394328  5.449848  4.757117
##        99       100       101       102       103       104       105
##  5.907568  9.135571  4.645492  4.260101  5.230731  3.989125  4.854597
##       106       107       108       109       110       111       112
##  7.294649  4.593084  4.805393  5.776656  4.096725  6.050678  3.985712
##       113       114       115       116       117       118       119
##  4.767360  4.716016  4.520179  6.531117  7.688387  5.390853  4.498473
##       120       121       122       123       124       125       126
##  5.387933  9.230389  5.868470  4.766550  5.614795  6.220260  8.312382
##       127       128       129       130       131       132       133
##  3.805732  4.658237  4.607282  4.884407  5.949860  4.736266  5.161134
##       134       135       136       137       138       139       140
##  5.794143  7.927789  4.546423  3.742174  9.785000  4.875223  6.171581
##       141       142       143       144       145       146       147
##  5.873265  5.139524 11.314449  5.503066  4.736881  5.209143  4.884928
##       148       149       150       151       152       153       154
##  4.860293  6.303405  4.830726  5.035852  4.298801  6.294825  4.413743
##       155       156       157       158       159       160       161
##  4.436292  4.933911  4.033382  5.952590  5.249656 10.899344  4.692354
##       162       163       164       165       166       167       168
##  6.074686  8.872430  6.194518  6.163171  6.405953  6.071005  4.459098
##       169       170       171       172       173       174       175
##  5.029472  5.734607  5.291308  5.772209  4.870489  4.819068  5.579529
##       176       177       178       179       180       181       182
##  4.300624  5.434086  5.690437  5.586413 10.985414  5.395487  5.797550
##       183       184       185       186       187       188       189
##  7.177278  5.186550  5.398563  4.779470  7.774132  5.458751  5.102159
##       190       191       192       193       194       195       196
##  5.074691  4.333320  5.235169  5.080013  8.755698  4.077231  4.578311
##       197       198       199       200       201       202       203
##  9.555024  8.620538  3.591842  3.633888  5.927190  5.556361  3.913893
##       204       205       206       207       208       209       210
##  4.397610  4.060356  5.377539  5.245248  4.700988  5.038983  4.559246
##       211       212       213       214       215       216       217
##  6.922846  4.935127  4.912321  4.099768  5.889387  7.609062  5.199807
##       218       219       220       221       222       223       224
##  4.673685  4.431320  5.834614  4.840622  4.554919  4.892771  4.777546
```

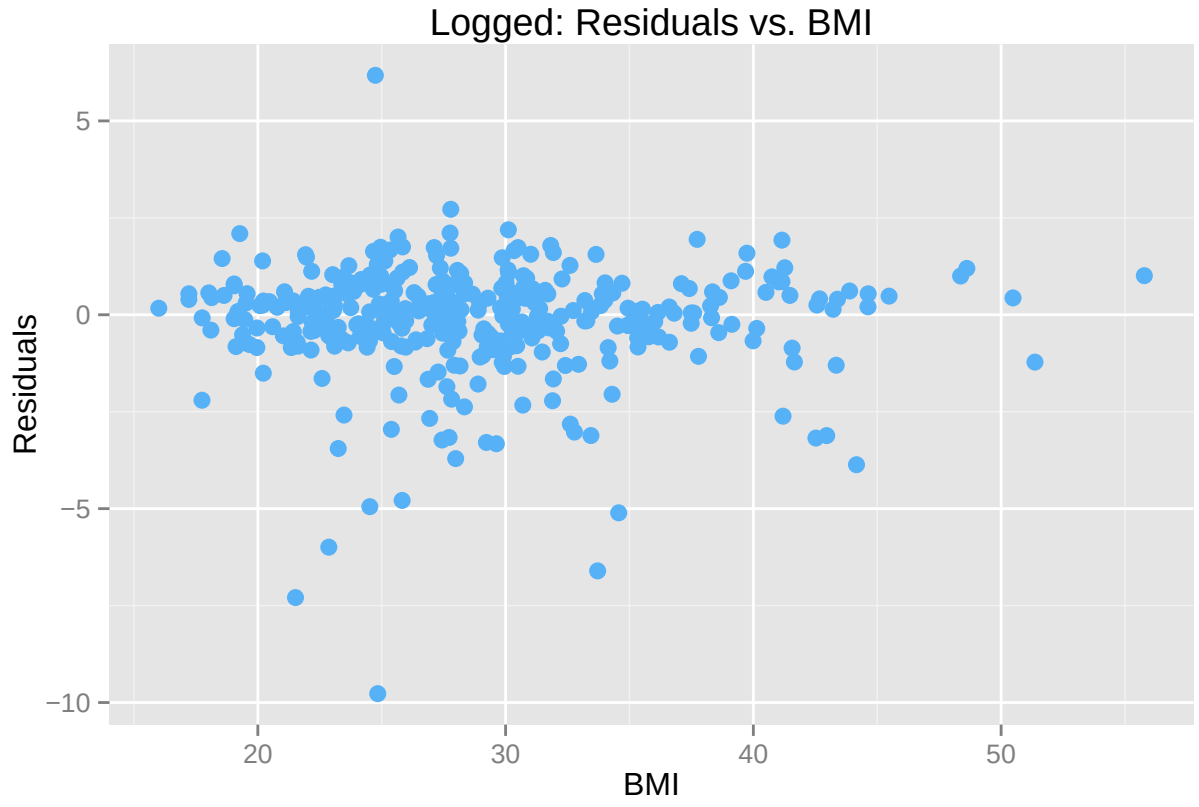
```
##       225       226       227       228       229       230       231
##  5.674633  5.387296  4.383335  4.268467  4.922402  6.278856  4.780747
##       232       233       234       235       236       237       238
##  4.056437  4.627015  4.873174  4.764007  3.956368  4.183411  6.259363
##       239       240       241       242       243       244       245
##  5.329165  6.664834  4.849405  8.419535  5.708484  5.347417  5.781273
##       246       247       248       249       250       251       252
##  4.756438  5.336189  4.027326  7.787933  4.837124  4.726653  4.927631
##       253       254       255       256       257       258       259
##  4.297987  4.276621  3.913864  4.641437  4.948272  5.244452  5.063546
##       260       261       262       263       264       265       266
##  5.591236  5.702416  4.390673  5.219291  5.465951  5.942287  5.203707
##       267       268       269       270       271       272       273
##  4.132244  4.261740  5.469097  5.095301  4.637066  5.545951  5.769261
##       274       275       276       277       278       279       280
##  8.162791  4.876912  5.222519  3.939168  4.698988  5.877315  4.059769
##       281       282       283       284       285       286       287
##  8.002578  4.423844  4.506699  4.518436  7.066600  4.323995  4.416065
##       288       289       290       291       292       293       294
##  5.759789  4.957873  4.553007  5.605532  4.832207  4.394468  5.722691
##       295       296       297       298       299       300       301
##  4.730255  5.853108  4.970519  4.788583  4.299659  4.286309  4.129691
##       302       303       304       305       306       307       308
##  5.330481  7.394008  4.224563  6.758837  4.668637  4.893459  4.865347
##       309       310       311       312       313       314       315
##  5.070694  3.676270  4.462241  6.387092  5.565371  3.923582  4.966184
##       316       317       318       319       320       321       322
##  4.764203  4.621898  4.149940  4.757321  4.917317  4.173655  4.588641
##       323       324       325       326       327       328       329
##  5.549476  4.600580  6.373775  4.507312  4.403611  5.240044  4.873308
##       330       331       332       333       334       335       336
##  4.970465  4.737080  5.982531  4.936894  9.942812  5.100884  5.433929
##       337       338       339       340       341       342       343
##  4.744804  5.168439  7.134848 10.116546  4.088678  5.730015  5.724336
##       344       345       346       347       348       349       350
##  7.854012  4.022874  4.775549  5.583156  4.010032  4.824292  4.773057
##       351       352       353       354       355       356       357
##  4.887169  5.126103  4.591879  4.830989  5.126856  4.733462  9.339341
##       358       359
##  4.893686  5.055075
```

Now, let's plot all three residual plots again.

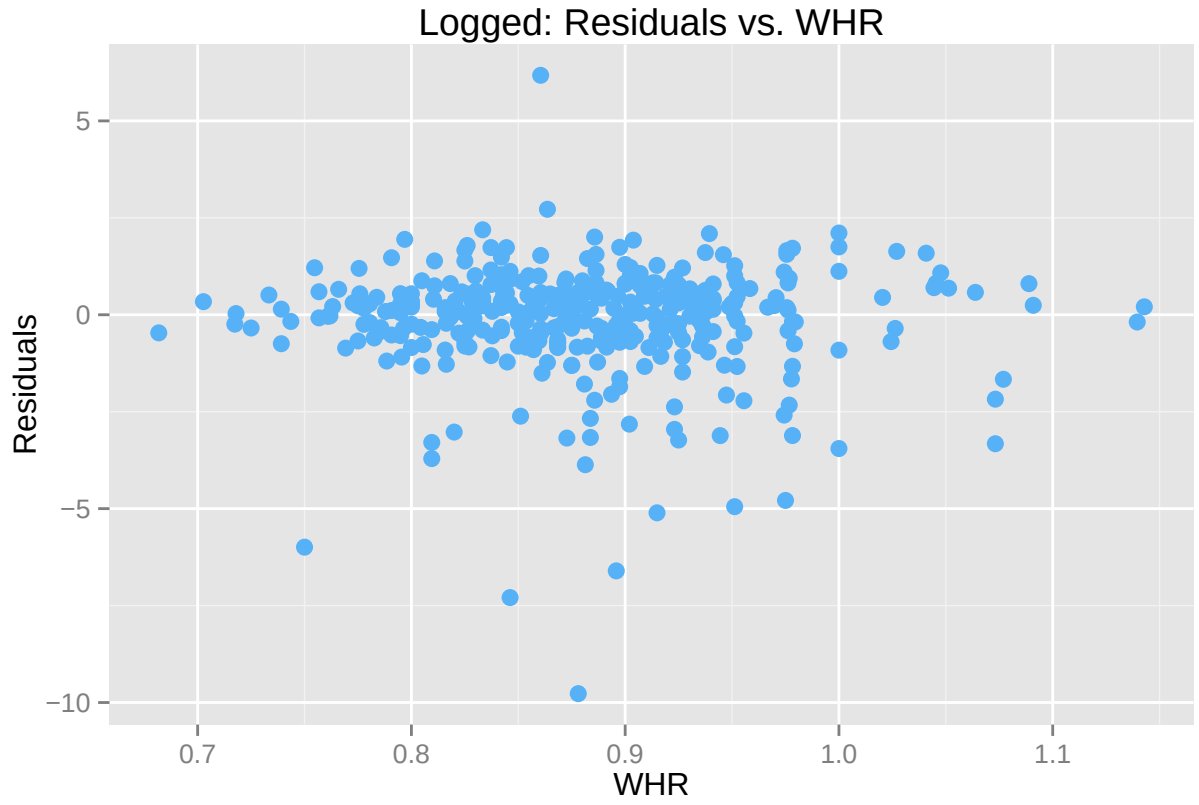
```
res.log = y_hat - glyhb
res_data_log = cbind(res = res.log, features_data)
ggplot(res_data_log, aes(x = stab.glu, y = res)) +
    geom_point(shape = 16, col = "#56B1F7", size = 3) +
    xlab("Stabilized Glucose") +
    ylab("Residuals") +
    ggtitle("Logged: Residuals vs. Stabilized Glucose")
```




```
ggplot(res_data_log, aes(x = BMI, y = res)) +
    geom_point(shape = 16, col = "#56B1F7", size = 3) +
    xlab("BMI") +
    ylab("Residuals") +
    ggtitle("Logged: Residuals vs. BMI")
```




```
ggplot(res_data_log, aes(x = WHR, y = res)) +
    geom_point(shape = 16, col = "#56B1F7", size = 3) +
    xlab("WHR") +
    ylab("Residuals") +
    ggtitle("Logged: Residuals vs. WHR")
```



For the first variable, stab.glu, our residuals shrunk down a lot on the y-axis scale, and are therefore more uniformly distributed. Thus, the log transformation helped.

For both BMI and WHR, our residuals become more uniform as well as the ranges of values that our residuals take on become narrower. Thus, the log transformation helped.

Question 6

To perform logistic regression, we will first create a logical vector of whether patients have diabetes or not.

```
diabetic.binary <- as.numeric(glyhb >= 7)
```

Now we can use the built-in glm function in R to help us create a logistic model. We will need to be sure the pass in binomial(logit) as the argument for family, since our model is logistic and our outcome is binary.

```
standard_features = data.frame(standard_features)
glm.yhat <- glm(diabetic.binary ~ stab.glu + age + bp.1s + WHR + BMI, data = standard_features, family=
```

Now we can predict the patients' glyhb values with our logistic model.

```
y_hat2 = predict(glm.yhat, type="response")
```

Now that we have the probabilities for each patient, we need to find a threshold to classify our patients by. In other words, we need a threshold such that a patient with a probability greater than our threshold is classified as having diabetes, and a patient with a probability less than or equal to our threshold is classified as not having diabetes.

Well, our first instinct was to use 0.5 as our threshold and find our Predictive Error Rate, False Positive Rate, and False Negative Rate.

```
TN = sum(y_hat2 > 0.5 & diabetic.binary) # True negative
FN = sum(y_hat2 < 0.5 & diabetic.binary) # False negative
TP = sum(y_hat2 < 0.5 & !diabetic.binary) # True positive
FP = sum(y_hat2 > 0.5 & !diabetic.binary) # False positive

PER = (FN + FP) / sum(TN + FN + TP + FP)
PER
```

```
## [1] 0.08077994
```

```
FPR = FP / (FP + TP)
FPR
```

```
## [1] 0.02295082
```

```
FNR = FN / (FN + TN)
FNR
```

```
## [1] 0.4074074
```

Because we are dealing with a medical condition here, we must be extra sensitive towards our False Negative Rate as telling a patient that they don't have diabetes when they really do is more harmful than telling a patient they have diabetes when they really don't.

So, now let's plot each error rate as a function of probability to help us choose which threshold value works best.

The following function will help us plot these graphs.

```
errs = function(prob){
TN = sum(y_hat2 > prob & diabetic.binary) # True negative
FN = sum(y_hat2 < prob & diabetic.binary) # False negative
TP = sum(y_hat2 < prob & !diabetic.binary) # True positive
FP = sum(y_hat2 > prob & !diabetic.binary) # False positive

PER = (FN + FP) / (TN + FN + TP + FP)
FPR = FP / (FP + TP)
FNR = FN / (FN + TN)

return(c(PER, FPR, FNR))
}
```

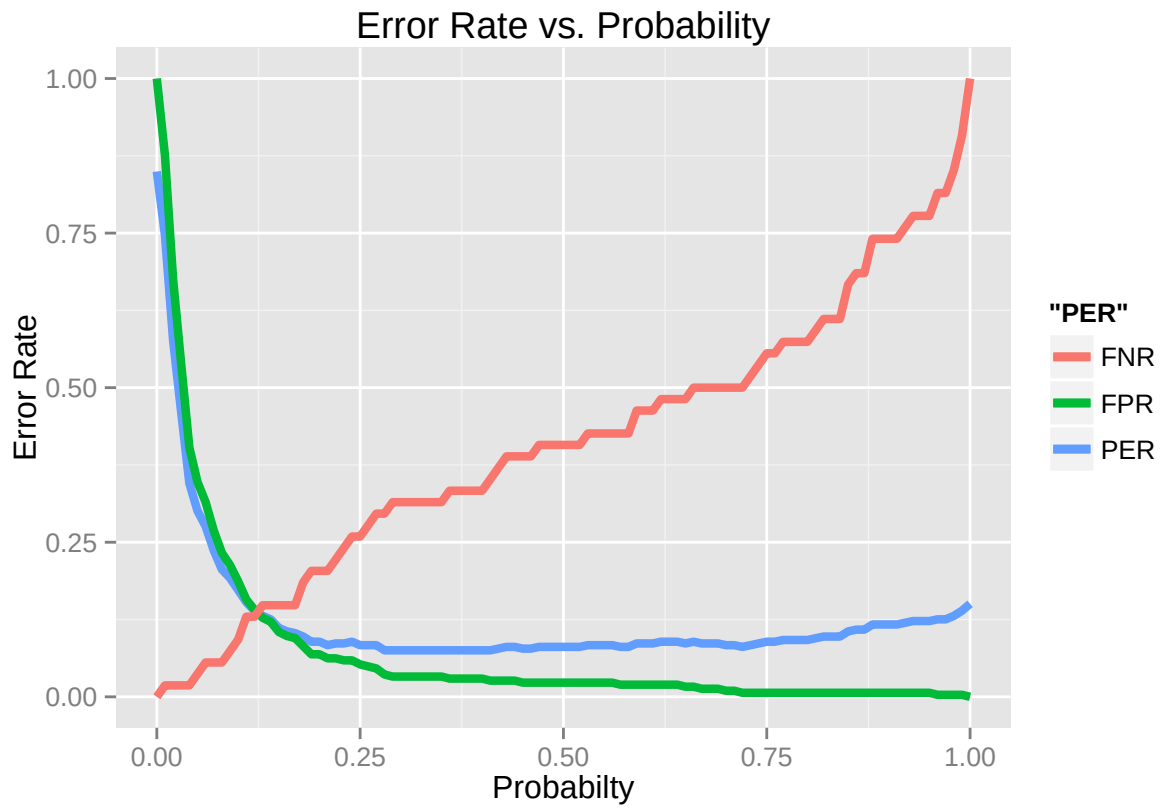
Now let's make a data frame to input in ggplot to help us create these three plots.

```
x = seq(0, 1, by = 0.01)
y = t(sapply(x, errs))
colnames(y) = c("PER", "FPR", "FNR")
error_data = data.frame(cbind(x = x, y))
head(error_data)
```

```
##      x       PER       FPR        FNR
## 1 0.00 0.8495822 1.0000000 0.00000000
## 2 0.01 0.7465181 0.8754098 0.01851852
## 3 0.02 0.5766017 0.6754098 0.01851852
## 4 0.03 0.4596100 0.5377049 0.01851852
## 5 0.04 0.3454039 0.4032787 0.01851852
## 6 0.05 0.3008357 0.3475410 0.03703704
```

Now let's plot all three curves on one plot to get a nice visual of what's happening.

```
ggplot(error_data, aes(x)) +
  geom_line(aes(y = PER, colour = "PER"), size = 1.5) +
  geom_line(aes(y = FPR, colour = "FPR"), size = 1.5) +
  geom_line(aes(y = FNR, colour = "FNR"), size = 1.5) +
  xlab("Probabilty") +
  ylab("Error Rate") +
  ggtitle("Error Rate vs. Probability")
```



From our visual above, it seems as if a probability of 0.125 will fit best for all three errors in one consideration. This is simply the x value at which all three lines intercept.

So, for our logistic model, we state that if a predicted probability is greater than 0.125, then the patient has diabetes. Otherwise, they do not have diabetes.

In Section 5.2 we had the following Error Rates: False Positive = 0.02622951 and False Negative = 0.3518519. For our logistic model, using a threshold of 0.124, we have the following Error Rates: False Positive = 0.1344262 and False Negative = 0.1296296.

With logistic regression, we were able to adjust and make our model more stable. By this, we mean that both error rates and close and we don't sacrifice too much of one for the other.

Question 7

First let's run our linear model on the test set. We will simply repeat the steps from above.

```
data_test = read.csv("diabetes_test.csv")
data_test$BMI = 703 * (data_test$weight / (data_test$height)^2)
data_test$WHR = data_test$waist / data_test$hip
features_data2 = data_test[ , c("stab.glu", "age", "bp.1s", "WHR", "BMI")]
y_hat2 = predict(lm.yhat, newdata = features_data2)

y_hat_diabetes = c()
y_hat_diabetes[y_hat2 >= 7] = TRUE
y_hat_diabetes[y_hat2 < 7] = FALSE
data_test$y_hat_diabetes = y_hat_diabetes

y_actual_diabetes = c()
y_actual_diabetes[data_test$glyhb >= 7] = TRUE
y_actual_diabetes[data_test$glyhb < 7] = FALSE
data_test$y_actual_diabetes = y_actual_diabetes

error_table = table(y_hat_diabetes, y_actual_diabetes)
error_table
```

```
##               y_actual_diabetes
## y_hat_diabetes FALSE TRUE
##          FALSE    12    1
##          TRUE      0    3
```

```
pred_error = (error_table[2, 1] + error_table[1, 2]) / sum(error_table)
pred_error
```

```
## [1] 0.0625
```

```
FPR.test = error_table[2, 1] / sum(error_table[ , 1])
FPR.test
```

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

```
FNR.test = error_table[1, 2] / sum(error_table[ , 2])
FNR.test
```

```
## [1] 0.25
```

Using our linear regression model leads to an overall error prediction rate of 0.0625. The False Positives Rate is 0, and the False Negatives Rate is 0.25.

Now let's use our logistic model where our threshold is 0.125.

```
standard_features2 = data.frame(cbind(1, apply(features_data2, 2, scale)))
y_hat2 = predict(glm.yhat, newdata = standard_features2, type="response")
```

Now let's use our errs function from above to find the error rates. This will work as we will chang the diabetic.binary vector to the current 16 patients outcome of diabetes and rename the y_hat2 vector to its current values of probabilities.

```
diabetic.binary = as.numeric(data_test$glyhb >= 7)
errs(0.125)
```

```
## [1] 0.0625 0.0000 0.2500
```

So we see that our logistic regression model leads to an overall error prediction rate of 0.0625. The False Positives Rates is 0 and the False Negatives Rates is 0.25. These are the exact error rates we got from our linear model.

Thus, neither model did better than the other when applied to our test set. Thus, we do obtain similary performance. This is probably because we used the same training set and because the test set is very small. With only 16 patients in the test set, it is hard to see variance between the two models. Or rather, a difference in any error rate.