Problem Set 1

Introduction to R
E63 Big Data Analytics

Andrew Caide
Harvard Extention School

September 7, 2017
Problem Set 1

Contents

Chapter 1

Problem Set Questions

1.1 Problem 1

Binomial distribution describes coin tosses with potentially doctored or altered coins. Value of p is the probability that head comes on top. If both the head and the tail have the same probability, p = 0.5. If the coin is doctored or altered, p could be larger or smaller. Plot on three separate graphs the binomial distribution for p = 0.3, p = 0.5 and p = 0.8 for the total number of trials n = 60 as a function of k, the number of successful (head up) trials. Subsequently, place all three curves on the same graph. For each value of p, determine 1st Quartile, median, mean, standard deviation and the 3rd Quartile. Present those values as a vertical box plot with the probability p on the horizontal axis. (15%)

1.2 Problem 2

Finish the plot of the correlation between waiting times and durations of Old Faithful data. Recreate the scatter plot of waiting vs. duration times. As we mentioned in class, the best linear assessment in the sense of the least squares fit of a relationship (proportionality) between two or many variables can be achieved with R function lm(). lm stands for the linear model. The first argument of lm() is called formula accepts a model which starts with the response variable, waiting in our case, followed by a tilde (symbol , read as 'modeled a' followed by the (so called Wilkinson-Rogers) model on the right. In our case we simply assume that waiting time is proportional to the duration time and that 'modelâĂİ reads: formula = waiting duration. The second argument of function lm() is called data and, in our case, will take value faithful, the data set containing our data. Store the result of function lm() in a variable. The name of that variable is not essential. Call it model. Print the variable. The first component of that variable is the intercept of calculated line with the

vertical axis (waiting, here) and the second is the slope of the line. Convince yourself that line with those parameters will truly lie on your graph. Function abline() adds a line to the previously created graph. Next, pass the variable model to the function abline(). Make that line somewhat thicker and blue. Use help(functionName) to find details about invocations of both lm() and abline() functions. (20%)

1.3 Problem 3

Calculate the covariance matrix of the faithful data. Determine the eigenvalues and eigenvectors of that matrix. Demonstrate that two eigenvectors are mutually orthogonal. Demonstrate that the eigenvector with the larger eigenvalue is parallel with line discovered by lm() function it the previous problem. (15%)

1.4 Problem 4

You noticed that eruptions clearly fall into two categories, short and long. Let us say that short eruptions are all which have duration shorter than 3.1 minute. Add a new column to data frame faithful called type, which would have value 'short'for all short eruptions and value 'long' for all long eruptions. Next use boxplot() function to provide your readers with some basic statistical measures for waiting. In a separate plot present the box plot for duration times. Please note that boxplot() function also accepts as its first argument a formula such as waiting type, where waiting is the numeric vector of data values to be split in groups according to the grouping variable type. The second argument of function boxplot() is called data, which in our case will take the name of our dataset, i.e. faithful. Find a way to add meaningful legends to your graphs. Subsequently, present both boxplots on one graph. (20%)

1.5 Problem 5

Create a matrix with 40 columns and 100 rows. Populate each column with random variable of the uniform distribution type. Make those distributions symmetric around zero. Let the distribution for each column appear like the one on slide 92 of the lecture note, except centered around zero. Present two distributions contained in any two randomly selected columns of your matrix on two separate plots. Convince yourself that generated distributions are (close to) uniform. (15

1.6 Problem 6

Start with your matrix from problem 5. Add yet another column to that matrix and populate that column with the sum of original 40 columns. Create a histogram of values in the new column showing that the distribution starts to resemble the Gaussian curve. Add a true, calculated, Gaussian curve to that diagram with the parameters you expect from the sum of 40 random variables of uniform distribution with values between -1 and 1. (15%)

1.7 Document History

- September 2 2017: first draft document

Chapter 2

Solutions

2.1 Problem 1

This question explores binomial distributions with different weights (or probabilities). A set of results that are binomially distributed have only two results, like a coin flip (heads or tails).
Firstly we explore what a weighted distribution looks like, with three weights: 0.3 (20% lower success rate), 0.5 (equally weighted), and 0.8 (30% higher success rate).

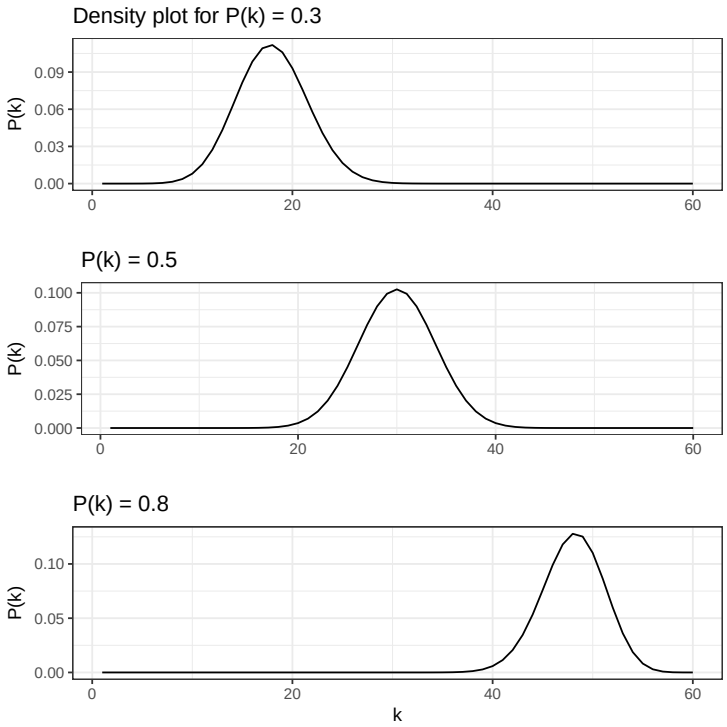

Figure 2.1: Question 1, Individual Density Plots of Binomial distributions of three trials where n = 60.

To confirm the distributions in Figure 2.1, we can recall the mean of a binomial distribution:

$$\mu_k = np$$

where μ_k is the mean (on k axis), with probability p, and n number of trials. Furthermore binomial distributions have a variance:

$$\sigma_k^2 = np(1-p)$$

where σ_k^2 is the standard deviation. It should be noted that as the probability increases the variance tightens as the probability increases. An eyeball test indicate our plots were spot on: $\mu_k\{0.3, 0.5, 0.8\} = 60 * \{0.3, 0.5, 0.8\} = \{18, 30, 48\}$.

Before moving on, I would like to use histograms for the next plot because coinflip measurements are discrete: there's no way somebody could have obtained 15.3 heads + 0.7 tails.The plot will have 60 bins, one per possible outcome (n = 60). It should also be noted that the probability values are going to appear greater - they should. The (continuous) density plots weigh the probability on all space, including invalid results. Now that the results have made discrete the likelihood of outcomes should look a lot more reasonable.

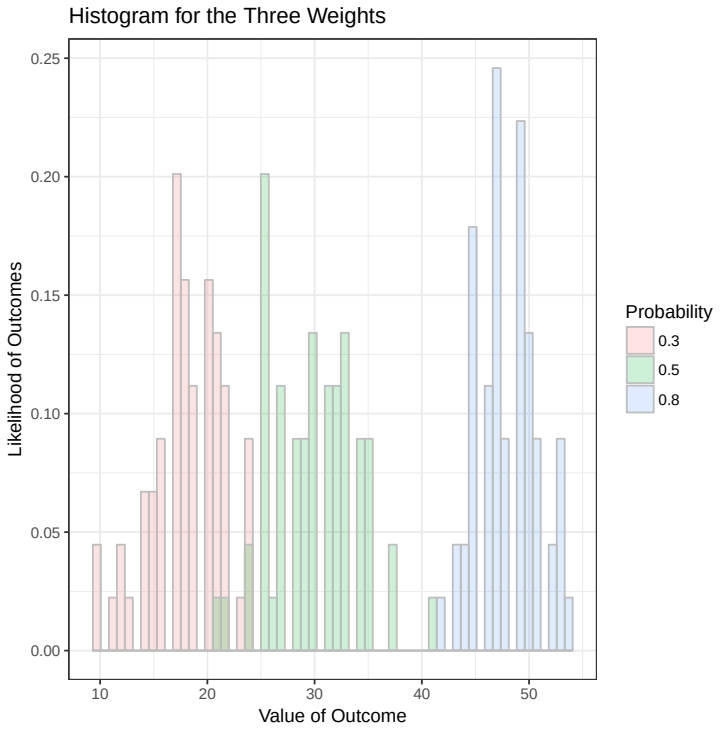

Figure 2.2: Question 1, Binomial distributions of three trials where n = 60.

The summary of the binomial statistics are outlined in the following table, and boxplots on figure 2.3 further illustrate the median, 25% and7% quartiles, and the max and min.

Table 2.1: Summary of Question 1 Data. Length of data set, 25 Percent quartile, Mean, Median, Standard Deviation, and 75 Percent quartile.

Probability	LowerQuartile	Median	Mean	SD	UpperQuartile
0.3	16	18	18	3	21
0.5	27	30	30	4	33
0.8	46	48	48	3	50

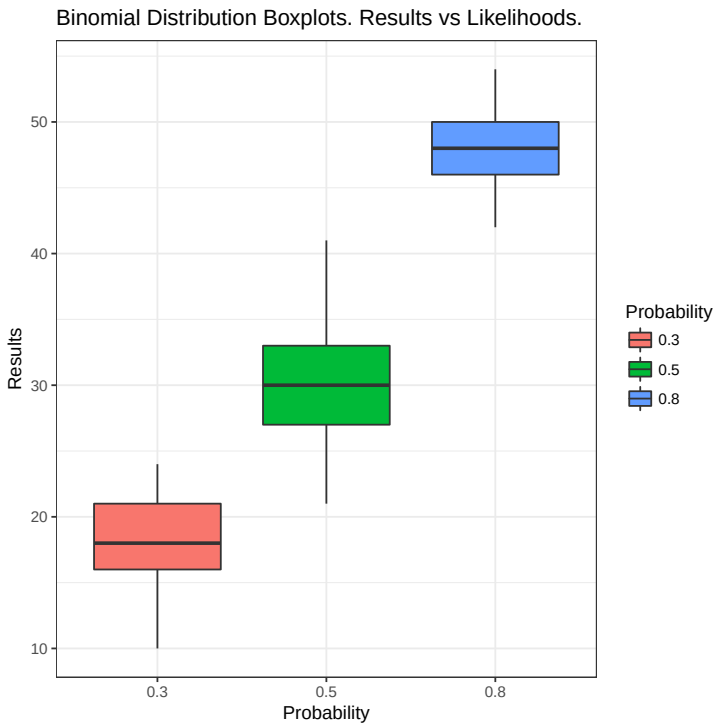

Figure 2.3: Question 1, Box Plots for the Binomial Distributions. Statistics outlined in aforementioned table.

2.2 Problem 2

This question investigates the relationship of two measures in the 'faithful' built in data-set. This data set comes from the Old Faithful Geyser in Yellowstone National Park, USA. Two measures will be examined in this data set: the time between geyser eruptions (in minutes), and the duration of the eruptions (in minutes). First we look at the data on a scatter-plot and see if we can apply a linear fit with a reasonable correlation.

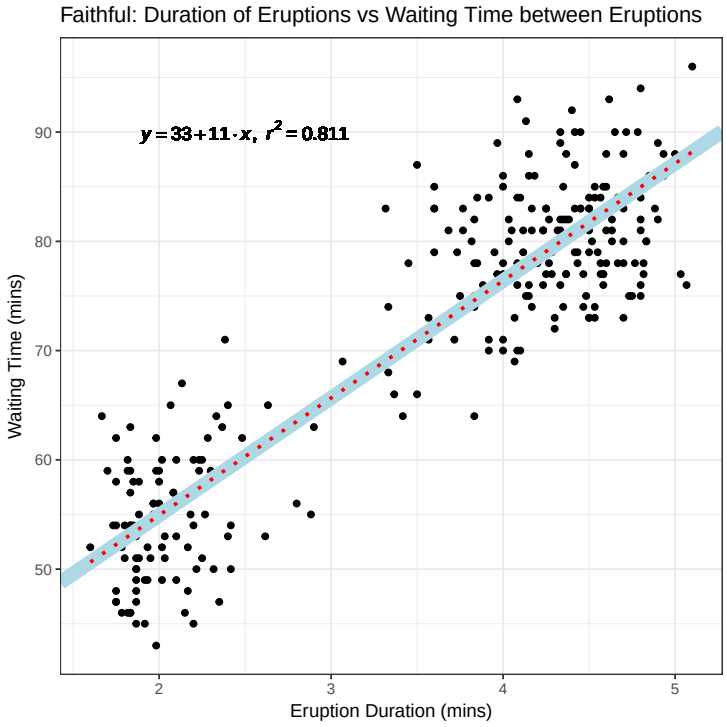

Figure 2.4: Question 2, Linear Fit on Faithful Data

Fhe linar fit constructed from the lm() function is shown in blue. To verify, I allowed ggplot to apply its fit (without additional input), indicated by the red dots. The correlation coefficient is weak at $r^2 = 0.811$.

2.3 Problem 3

This problem revisits the faithful data. In the previous question we found ourselves unsatisfied with the correlation coeficient from the linear plot, but the plot certainly indicates two populations of events. Let's take a more rigorous look.

First let's observe the covariance of the faithful data.

```
> df <- faithful
> covariance <- cov(df)
> covariance # See table
```

Table 2.2: Covariance result.

eruptions	waiting
1	14
14	185

The positive covariance between eruption durations and waiting times idicates a positive relationship between the two variables, which was already made fairly obvious. It is safe to ignore the diangonols in this matrix. Note: a covariance of 0 should indicate no relationsip, and the magnitude of the covariance indicates a stronger relationship. Therefore, for example, a covariance of -130 would indicate a tight correlation (or relatively stronger than our covariance of 13.98), but negatively trending relationsip.

From this we can calculate the eigen values and eigen vectors.

```
> eig.values <- eig$values
eigen.values

[1] 185.8818239   0.2442167

> eig.vectors <- eig$vectors
> eig.vectors

           [,1]       [,2]
[1,] -0.9885959 -0.1505924
[2,]  0.1505924 -0.9885959
```

We can also demonstrate orthogonality.

```
> eig.vectors * t(eig.vectors)

     [,1] [,2]
[1,]    1    0
```

```
[2,]    0    1
```

We can use the eigen vectors to calculate the slope of our data set! First we find which eigen value is largest - this is our principal component.

```
> i = which(eig.values == max(eig.values))
> i
```

```
1
```

The first vector shall be used to calculate our slope.

```
> slope.eig = eig.vectors[2,i]/eig.vectors[1,i] # This is just rise over run
> slope.eig
```

```
[1] 13.20515
```

This doesn't agree with our previous slope. This is because the lm() function uses OLM (ordinary least square (means)), which aims to minimize the error between the dependent variable and the model. The principal component minimizes the eerror orthogonal to the model line.

Note, I had difficulting accessing the intercept if I proceeded with this calculation. What I will do next is normalize the original data set, perform the same eigen calculations, and denormalize it by putting it back in terms of the faithful data.

```
> xyNorm <- cbind(x = x- mean(df[,1]), y = y-mean(df[,2]))
> xyEig <- eigen(cov(xyNorm))
> eigenValues <- xyEig$values
> eigenVectors <- xyEig$vectors

# Slope is calculated the same way we did before.
# Index should be the same
> slope.eig <- eigenVectors[2,i]/eigenVectors[1,i]


> xyNorm <- data.fram(xyNorm)
> p <- ggplot(data = xyNorm, aes(x = x, y = y))+
  geom_point() +
  geom_abline(slope = slope, color = "lightblue", size=2)+
  geom_smooth(method = "lm", se = FALSE, linetype="dotted", color = "red")+
    labs(title = "Faithful: Duration of Eruptions
                    vs Waiting Time between Eruptions",
          x = "Eruption Duration (mins)",
          y = "Waiting Time (mins)",
          color = "Linear Model")+
```

```
  theme_bw() +
  geom_abline( slope = slope.eig, color = "pink", size=2)+
  geom_text(x = -1, y = 20,
  label = as.character("Blue - original lm fit,
          Red - Principal Component fit."), parse = F)

> print(p)
```

Figure 2.5: Question 3, normalized data fit to linear model (blue) and principal component analysis model (red)

Below is the de-normalized data.

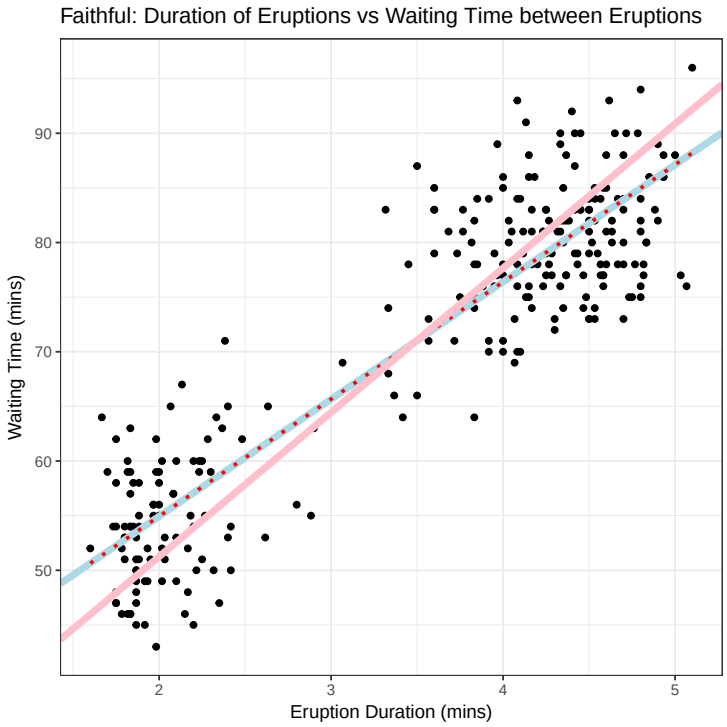

Figure 2.6: Question 3, data fit to linear model (blue) and principal component analysis model (red)

2.4 Problem 4

This chapter revisits the faithful data set. The goal of this problem is to subet the data to account for the two populations previously observed in our figures (according to length of eruptions). Data is presented below in boxplots.

Figure 2.7: Question 4, Faithful Boxplots

2.5 Problem 5

The following two question explores uniform distributions mapped form -1 to 1. Below are two randomly selected columns from a 40x100 matrix, with randomly generated points from the runif() function. The density is calculated by passing the results through dunif(). The two plots below explore the two outcomes plotted across -2 and 2 to illustrate the boundaries of the uniform distribution.

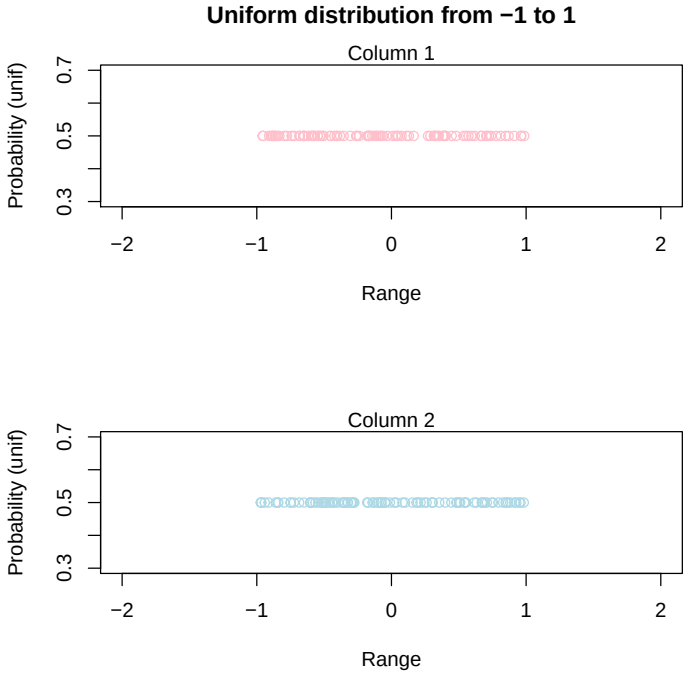

Figure 2.8: Question 5, Uniform Distribution

To further illustrate that the distributions are (close to) uniform, five boxplots are generated below from 5 randomly selected columns. The means should be centered around zero, and the spread should be from -1 to 1.

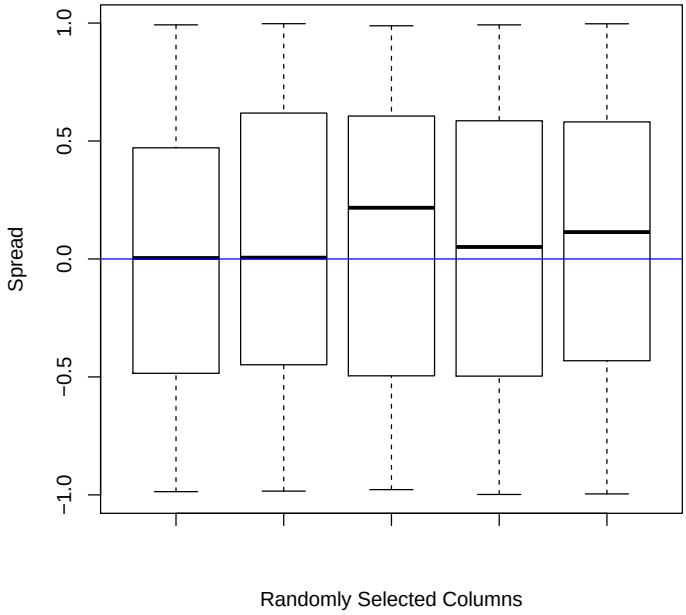

Figure 2.9: Question 5, Uniform Distribution, boxplots

2.6 Problem 6

The following problemis an exercize in data manipulation and visualization. Here we summed every column entry across the previous data table and fitted it to a gaussian curve.

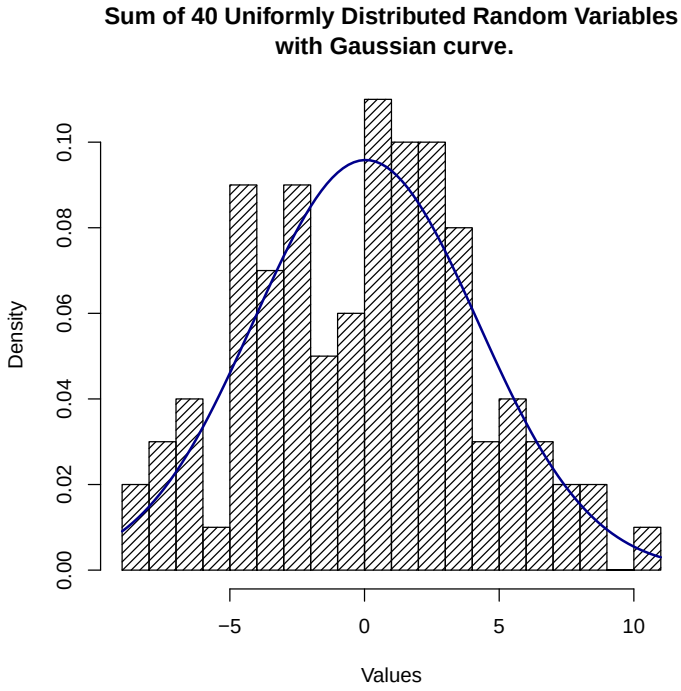

Figure 2.10: Question 6, Gaussian Distribution

Chapter 3

Code

3.1 Problem 1

```
> #####
> # Problem 1.
> #####
> # Code:
>
> set.seed(10111990)
> # Set conditions
> n <- 60
> P1 <- 0.3
> P2 <- 0.5
> P3 <- 0.8
> vector <- c(1:n)
> # For simple density plots - "d" for density
> binom1 <- dbinom(vector, n, P1)
> binom2 <- dbinom(vector, n, P2)
> binom3 <- dbinom(vector, n, P3)
> # For experimental trials - "r" for randomly generated values with given distribution
> values1 <- rbinom(vector, n, P1)
> values2 <- rbinom(vector, n, P2)
> values3 <- rbinom(vector, n, P3)
> # Form data frame and plot for simple density plot
> df <- data.frame("k" = vector, "Pk" = binom1, "Probability" = "0.3")
> df <- rbind(df, data.frame("k" = vector, "Pk" = binom2, "Probability" = "0.5"))
> df <- rbind(df, data.frame("k" = vector, "Pk" = binom3, "Probability" = "0.8"))
> # Individual Density Plots
> p1 <- ggplot(data=df[which(df$Probability==0.3),], aes(x=k, y=Pk))+
+     geom_line()+
+     labs(title = "Density plot for P(k) = 0.3", y = "P(k)", x="",
```

```
+        color = "0.3")+
+     theme_bw()         #turn off grey grid
> p2 <- ggplot(data=df[which(df$Probability==0.5),], aes(x=k, y=Pk))+
+    geom_line()+
+       labs(title = "P(k) = 0.5", y = "P(k)", x="",
+        color = "0.3")+
+   theme_bw()         #turn off grey grid
> p3 <- ggplot(data=df[which(df$Probability==0.8),], aes(x=k, y=Pk))+
+    geom_line()+
+       labs(title = "P(k) = 0.8", y = "P(k)", x="k",
+        color = "0.3")+
+   theme_bw()         #turn off grey grid
> grid.arrange(p1,p2,p3)
> # Histogram Plot/All three cumulatives.
> ## Make Bar-Plot values
> df2 <- data.frame("Value" = values1, "Probability" = "0.3")
> df2 <- rbind(df2,data.frame("Value" = values2, "Probability" = "0.5"))
> df2 <- rbind(df2,data.frame("Value" = values3, "Probability" = "0.8"))
> p <- ggplot(df2, aes(x=Value, fill=Probability)) +
+   geom_histogram(aes(y=..density..), alpha=0.2, position="identity", col = "grey", bins=60
+   labs(title = "Histogram for the Three Weights", x = "Value of Outcome",
+        y = "Likelihood of Outcomes",
+        color = "Weights\n")+
+   theme(panel.border = element_blank(), panel.grid.major = element_blank(),
+         panel.grid.minor = element_blank(), axis.line = element_line(colour = "black"))+
+   theme_bw()
> print(p)
> ##########################################
> #
> # Summary table of the bar-plot statistics!
>
> # This is very sloppy and lazy, but I can't figure a good "apply" alternative :\\
> quartile.bind <- quartile.compute(df2[which(df2$Probability=="0.3"),1])
> quartile.bind <- rbind(quartile.bind, quartile.compute(df2[which(df2$Probability=="0.5"),1
> quartile.bind <- rbind(quartile.bind, quartile.compute(df2[which(df2$Probability=="0.8"),1
> rownames(quartile.bind) <- c("0.3", "0.5", "0.8")
> # But for some reason this works???
> sum.tab <- ddply(df2,~Probability,summarise,Mean=mean(Value),SD=sd(Value), Median = median
> sum.tab <- cbind(sum.tab, quartile.bind)
> sum.tab <- sum.tab[,c(1,5,4,2,3,6)] #Cleaning
> xtab <- xtable(sum.tab, caption=paste("Summary of Question 1 Data. Length of data set, 25
+               align="|r|r|rrrrr|", digits = 0)
> print(xtab, file="q1.tex", table.placement = "h",
+       sanitize.text.function=my.sani, include.rownames = FALSE, floating = FALSE,
+                hline.after=c(-1,0,nrow(xtab)), tabular.environment="longtable",
+                add.to.row=addtorow, caption.placement="top")
```

```
> ##########################################
> #
> # Plot distribution barplots
> plot2 <- ggplot(data = df2, aes(y=Value, x = Probability, fill = Probability)) +
+   geom_boxplot() +
+   labs(title = "Binomial Distribution Boxplots. Results vs Likelihoods.", y = "Results",
+        color = "Probabilities\n")+
+   theme_bw()         #turn off grey grid
> print(plot2)
```

3.2 Problem 2

```
> #####
> # Problem 2.
> #####
> # Code:
>
> ## Found this on the web a long time ago. Very useful
> lm_eqn <- function(df){
+     m <- lm(y ~ x, df);
+     eq <- substitute(italic(y) == a + b %.% italic(x)*","~~italic(r)^2~"="~r2,
+          list(a = format(coef(m)[1], digits = 2),
+               b = format(coef(m)[2], digits = 2),
+              r2 = format(summary(m)$r.squared, digits = 3)))
+     as.character(as.expression(eq));
+ }
> ##
>
> # Select Data
> df <- faithful
> # Construct linear model and select coefficients
> lin.mod <- lm(data = df, formula = waiting ~ eruptions)
> intercept = lin.mod$coefficients[1]
> slope = lin.mod$coefficients[2]
> # Rename for convenience, plot
> colnames(df) <- c("x","y")
> p <- ggplot(data = df, aes(x = x, y = y))+ #+
+   geom_point() +
+   geom_abline(intercept = intercept, slope = slope, color = "lightblue", size=4)+
+   geom_smooth(method = "lm", se = FALSE, linetype="dotted", color = "red")+
+     labs(title = "Faithful: Duration of Eruptions vs Waiting Time between Eruptions",
+          x = "Eruption Duration (mins)", y = "Waiting Time (mins)",
+          color = "Linear Model")+
+   theme_bw() + geom_text(x = 2.5, y = 90, label = lm_eqn(df), parse = TRUE)
> print(p)
```

```
>
>
```

3.3 Problem 3

```
> #####
> # Problem 3.
> ####
> # Code:
>
> # Select Data
> df <- faithful
> # Compute covarience
> covariance <- cov(df)
> xtab <- xtable(covariance, caption=paste(" Covariance result.", sep=""),
+                align="rc|c", digits = 0)
> print(xtab, file="covar.tex", table.placement = "h",
+       sanitize.text.function=my.sani, include.rownames = FALSE, floating = FALSE,
+                hline.after=c(-1,0,1,nrow(xtab)), tabular.environment="longtable",
+                add.to.row=addtorow, caption.placement="top")
> # Calculate normalized data and eigen values
> xyNorm <- cbind(x = df[,1]- mean(df[,1]), y = df[,2]-mean(df[,2]))
> xyEig <- eigen(cov(xyNorm))
> eigenValues <- xyEig$values
> eigenVectors <- xyEig$vectors
> # Find greatest eigenvalue, retrieve corresponding vector
> i = which(eigenValues  == max(eigenValues))
> slope.eig <- eigenVectors[2,i]/eigenVectors[1,i]
> # Plot normalized data
> xyNorm <- data.frame(xyNorm)
> colnames(df) <- c("x","y")
> p <- ggplot(data = xyNorm, aes(x = x, y = y))+
+   geom_point() +
+   geom_abline(slope = slope, color = "lightblue", size=2)+
+   geom_smooth(method = "lm", se = FALSE, linetype="dotted", color = "red")+
+     labs(title = "Faithful: Duration of Eruptions vs Waiting Time between Eruptions",
+          x = "Eruption Duration (mins)", y = "Waiting Time (mins)",
+          color = "Linear Model")+
+   theme_bw() +
+   geom_abline( slope = slope.eig, color = "pink", size=2)+ geom_text(x = -1, y = 20,
+   label = as.character("Blue - original lm fit, Red - Principal Component fit."), parse =
> print(p)
> # Compute de-normalized data and plot
> points <- (slope.eig*xyNorm$x+mean(df$y))
> df <-cbind(df,points)
```

```
> newslope = lm(data=df, points~x)
> p <- ggplot(data = df, aes(x = x, y = y))+
+   geom_point() +
+   geom_abline(intercept = intercept, slope = slope, color = "lightblue", size=2)+
+   geom_smooth(method = "lm", se = FALSE, linetype="dotted", color = "red")+
+     labs(title = "Faithful: Duration of Eruptions vs Waiting Time between Eruptions",
+          x = "Eruption Duration (mins)", y = "Waiting Time (mins)",
+          color = "Linear Model")+
+   theme_bw() +
+   geom_abline(slope = newslope$coefficients[2], intercept =  newslope$coefficients[1], col
+   geom_text(x = -1, y = 20, label = as.character("Blue - original lm fit, Red - Principal
> print(p)
>
```

3.4 Problem 4

```
> #####
> # Problem 4.
> #####
> # Code:
>
> # Obtain data
> df <- faithful
> # Construct subsetting metrics, and apply labels
> df$Length <- NA
> df[which(df$eruptions<3.1),"Length"] <- "Short"
> df[which(df$eruptions>3.1),"Length"] <- "Long"
> #plot
> p1 <- ggplot(data=df, aes(x=Length, y = waiting))+
+   geom_boxplot(aes(fill =Length))+
+     labs(title = "Faithful: Waiting Time between Eruptions, Duration of Eruptions subset b
+          x = "", y = "Waiting Time (mins)",
+          color = "Linear Model")+
+   theme_bw()
> p2 <- ggplot(data=df, aes(x=Length, y = eruptions))+
+   geom_boxplot(aes(fill =Length))+
+     labs(title = "Faithful: Waiting Time between Eruptions, Duration of Eruptions subset b
+          x = "Duration of Eruption, subsets", y = "Duration of Eruption (mins)",
+          color = "Linear Model")+
+   theme_bw()
> grid.arrange(p1,p2)
>
>
>
```

3.5 Problem 5

```
> #####
> # Problem 5.
> #####
> # Code:
>
> #dunif gives the density,
> #punif gives the distribution function,
> #qunif gives the quantile function, and runif generates random deviates.
>
> # Set up data frame
> df <- matrix(nrow = 100, ncol=40)
> # Fill it in
> for(i in 1:ncol(df)){
+   df[,i] <-runif(nrow(df),min=-1,max=1)
+ }
> # Plot
> par(mfrow = c(2, 1))
> plot(df[,sample(1:ncol(df), 1)], dunif(df[,sample(1:ncol(df), 1)], min = -1, max = 1), xli
> mtext("Column 1")
> plot(df[,sample(1:ncol(df), 1)], dunif(df[,sample(1:ncol(df), 1)], min = -1, max = 1), xli
> mtext("Column 2")
> # Construct boxplots for proof
> par(mfrow = c(1, 1))
> boxplot(df[,sample(1:ncol(df), 1)],df[,sample(1:ncol(df), 1)],df[,sample(1:ncol(df), 1)],
+         df[,sample(1:ncol(df), 1)],df[,sample(1:ncol(df), 1)], xlab = "Randomly Selected C
> abline(h = 0, col = "blue")
>
```

3.6 Problem 6

```
> #####
> # Problem 6.
> #####
> # Code:
>
> # Same process as before
> df <- matrix(nrow = 100, ncol=40)
> for(i in 1:ncol(df)){
+   df[,i] <-runif(nrow(df),min=-1,max=1)
+ }
> # Throw everything into data frame, easier to work with
> df <- data.frame(df)
> # Sum the rows
```

```
> df$sum <- rowSums(df)
> # This renaming is for convenience
> g = df$sum
> m<-mean(g)
> std<-sqrt(var(g))
> # Plot histogram of values, "density" is for shading
> hist(g, density=20, breaks=20, prob=TRUE,
+      xlab="Values",
+      main="Sum of 40 Uniformly Distributed Random Variables\n with Gaussian curve.")
> # Add gaussian fit
> curve(dnorm(x, mean=m, sd=std),
+       col="darkblue", lwd=2, add=TRUE, yaxt="n")
>
```