Homework 1: Statistical Analysis

E-63 Big Data Analytics
Harvard University, Autumn 2017

Tim Hagmann

September 09, 2017

Contents

Initialize

In the following code chunk all the necessary setup for the modelling environment is done.

```
## Options
options(scipen = 10)                          # Disable scientific notation
update_package <- FALSE                       # Use old status of packages

## Init files (always execute, eta: 10s)
source("scripts/01_init.R")                   # Helper functions to load packages
source("scripts/02_packages.R")               # Load all necessary packages
source("scripts/03_functions.R")              # Load project specific functions
```

Problem 1 (15%)

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.

i) Three Graphs

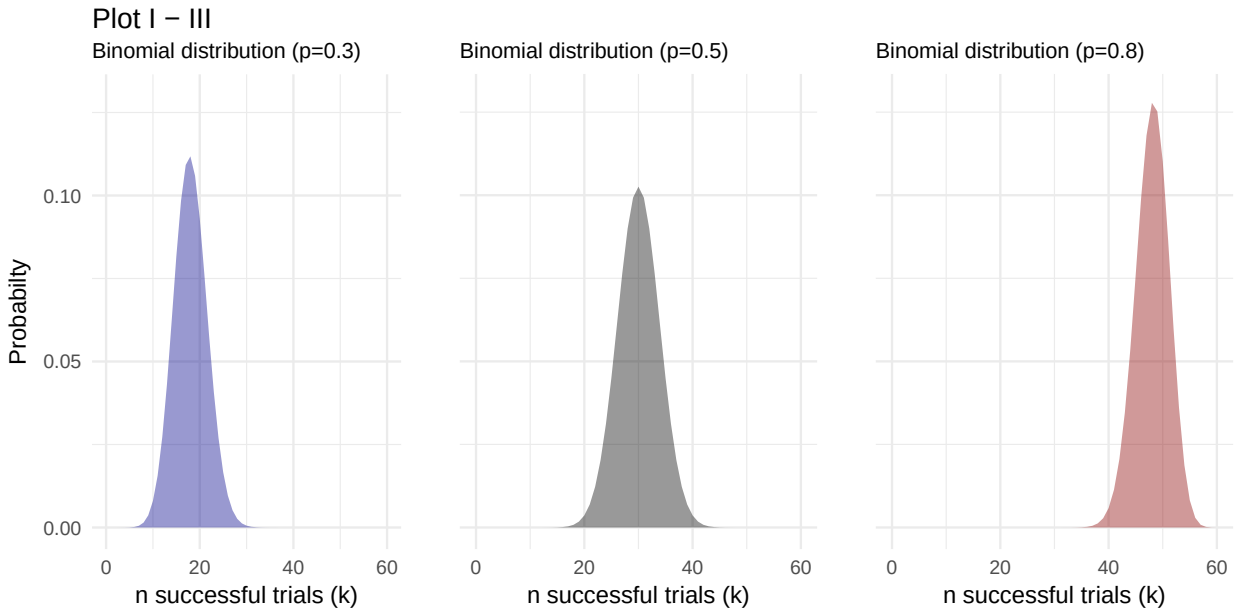
Plot on three separate graphs the binomial distribution for $p = 0.3$, $p = 0.5$ and $p = 0.8$ for the total number of trials $n = 60$ as a function of k, the number of successful (head up) trials.

Simulate distributions

```
set.seed(123) # Set random seed
trials <- 0:60
df_dist <- data.frame(trials=trials,
                      binomial_p_03=dbinom(x=trials, size=60, prob=0.3),
                      binomial_p_05=dbinom(x=trials, size=60, prob=0.5),
                      binomial_p_08=dbinom(x=trials, size=60, prob=0.8))
```

Visualize

```
# p=0.3 Distribution
g1 <- ggplot(data=df_dist, aes(x=trials)) +
  geom_area(aes(y=df_dist$binomial_p_03), fill="darkblue", alpha=0.4) +
  labs(title="Plot I - III",
       subtitle="Binomial distribution (p=0.3)") +
  theme_minimal() +
  ylab("Probabilty") +
  ylim(0, 0.13) +
  xlab("n successful trials (k)")

# p=0.5 Distribution
g2 <- ggplot(data=df_dist, aes(x=trials)) +
  geom_area(aes(y=df_dist$binomial_p_05), fill="black", alpha=0.4) +
  labs(title="",
       subtitle="Binomial distribution (p=0.5)") +
  theme_minimal() +
  theme(axis.text.y = element_blank()) +
  ylab("") +
  ylim(0, 0.13) +
  xlab("n successful trials (k)")

# p=0.8 Distribution
g3 <- ggplot(data=df_dist, aes(x=trials)) +
  geom_area(aes(y=df_dist$binomial_p_08), fill="darkred", alpha=0.4) +
  labs(title="",
       subtitle="Binomial distribution (p=0.8)") +
  theme_minimal() +
  theme(axis.text.y = element_blank()) +
  ylab("") +
  ylim(0, 0.13) +
  xlab("n successful trials (k)")

# Plot grid
grid.arrange(g1, g2, g3, nrow=1, ncol=3)
```

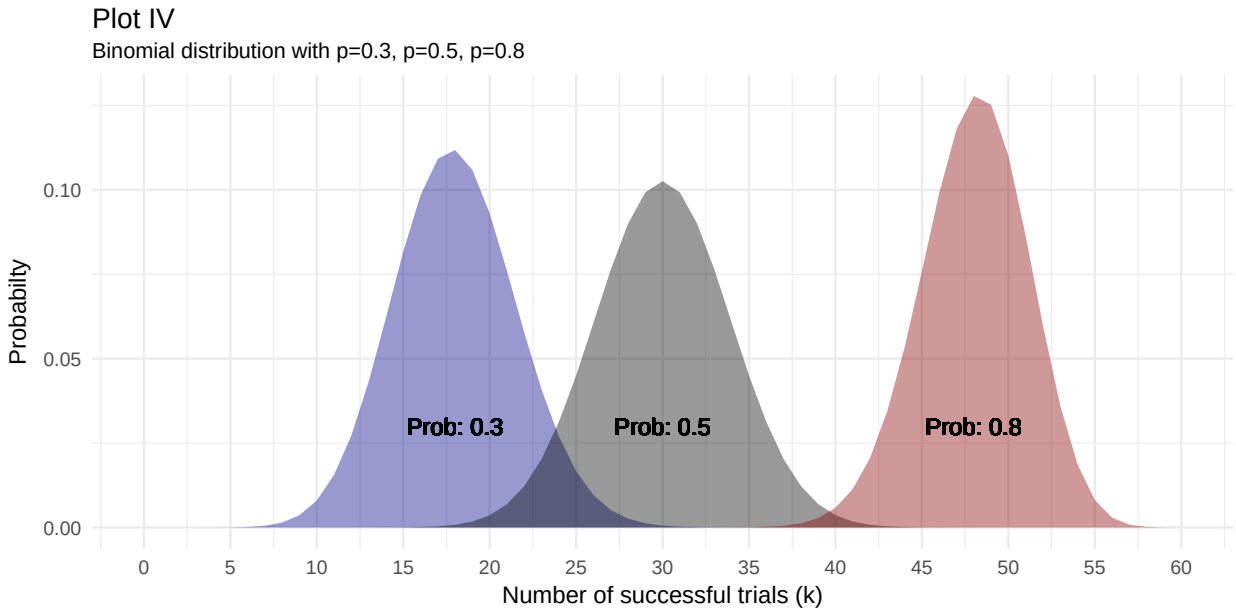


ii) One graph

Subsequently, place all three curves on the same graph.

Visualize

```
ggplot(data=df_dist, aes(x=trials)) +
  geom_area(aes(y=df_dist$binomial_p_03), fill="darkblue", alpha=0.4) +
  geom_area(aes(y=df_dist$binomial_p_05), fill="black", alpha=0.4) +
  geom_area(aes(y=df_dist$binomial_p_08), fill="darkred", alpha=0.4) +
  labs(title="Plot IV",
       subtitle="Binomial distribution with p=0.3, p=0.5, p=0.8") +
  theme_minimal() +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
  ylab("Probabilty") +
  xlab("Number of successful trials (k)") +
  geom_text(x=df_dist$trials[df_dist$binomial_p_03==max(df_dist$binomial_p_03)],
            y=0.03, label="Prob: 0.3", size=4) +
  geom_text(x=df_dist$trials[df_dist$binomial_p_05==max(df_dist$binomial_p_05)],
            y=0.03, label="Prob: 0.5", size=4) +
  geom_text(x=df_dist$trials[df_dist$binomial_p_08==max(df_dist$binomial_p_08)],
            y=0.03, label="Prob: 0.8", size=4)
```



iii) Summary statistics

For each value of p, determine 1st Quartile, median, mean, standard deviation and the 3rd Quartile.

Function to calculate summary statistics

```
summaryStat <- function(n, p){
  Prob <- p
  Quart_1 <- qbinom(0.25, size=n, prob=p)
  Median <- qbinom(0.5, size=n, prob=p)
  Mean <- n * p
  Std <- sqrt(n * p * (1 - p))
  Quart_3 <- qbinom(0.75, size=n, prob=p)
  data.frame(Prob, Quart_1, Median, Mean, Std, Quart_3)
}
```

Table I: Summary Statistics

```
pander(rbind(summaryStat(60, 0.3), summaryStat(60, 0.5), summaryStat(60, 0.8)))
```

Prob	Quart_1	Median	Mean	Std	Quart_3
0.3	16	18	18	3.55	20
0.5	27	30	30	3.873	33
0.8	46	48	48	3.098	50

The above table shows how the values change alonge the different p values.

iv) Boxplot

Present those values as a vertical box plot with the probability p on the horizontal axis.

In order to generate a boxplot the easiest way is to simulate the data with different probabilty values (p). A number of 100'000 trials should be sufficient so that the real values lie close to simulated ones. *Note:* The central limit theorem (CLT) shows, that from a n>30 the mean of the values distribute themself normally around the true value.

Generate data

```
# Simulation
set.seed(123) # Set random seed
df_sim <- data.frame(Prob_0.3=rbinom(100000, size = 60, prob = 0.3),
                     Prob_0.5=rbinom(100000, size = 60, prob = 0.5),
                     Prob_0.8=rbinom(100000, size = 60, prob = 0.8))
df_sim2 <- melt(df_sim)
```

Table II: Simulated data

```
pander(summary(df_sim))
```

Prob_0.3	Prob_0.5	Prob_0.8
Min. : 5.00	Min. :13.00	Min. :33.00
1st Qu.:16.00	1st Qu.:27.00	1st Qu.:46.00
Median :18.00	Median :30.00	Median :48.00
Mean :17.99	Mean :30.03	Mean :48.02
3rd Qu.:20.00	3rd Qu.:33.00	3rd Qu.:50.00
Max. :34.00	Max. :46.00	Max. :59.00

The above table shows, that the summary statistics of the simulated data gives the same numbers as seen before. We can therefore safely assume, that the numbers are correct.

Summary statistics

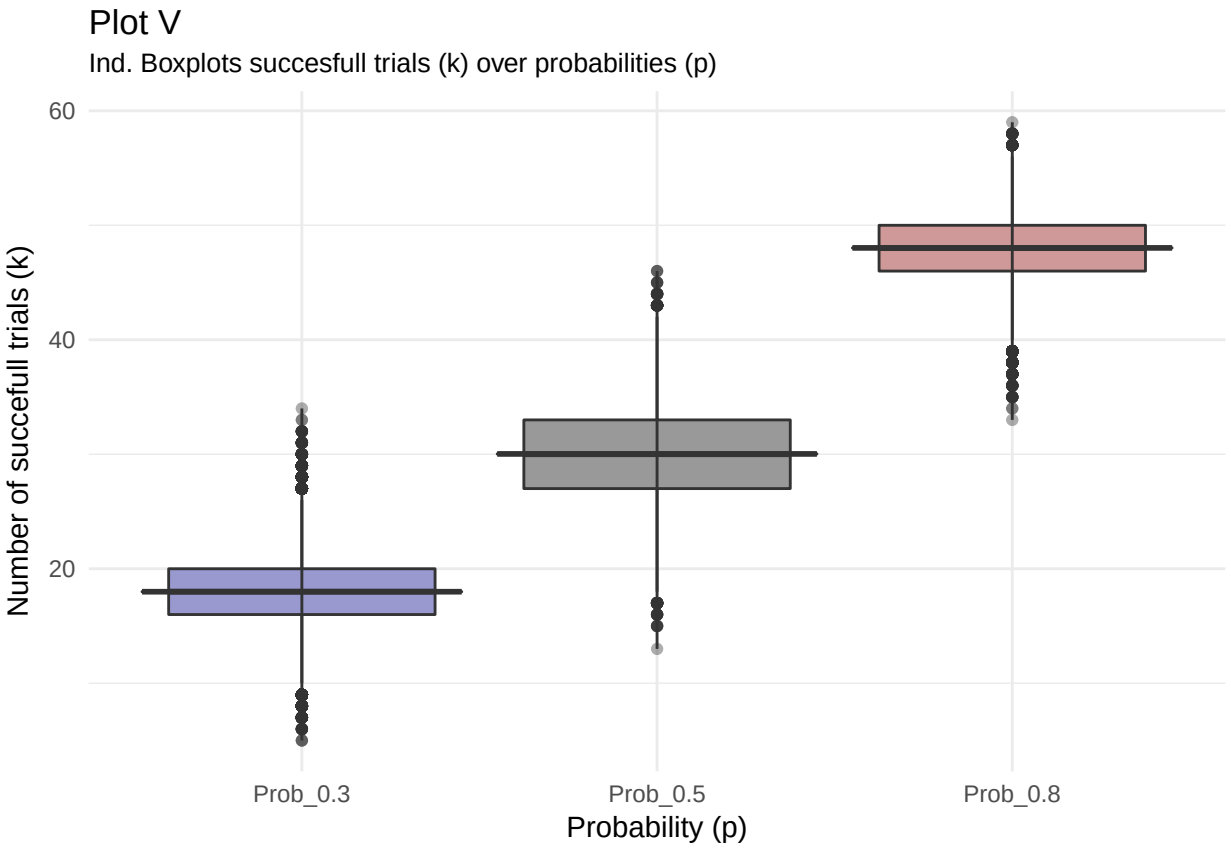
In order to visualize the summary values in the boxplot format a function that compute the mean-1SE, and Mean+1SE has to be written.

```
newSummaryStat <- function(x) {
  vec_var <- c(min(x), mean(x) - sd(x)/sqrt(length(x)), mean(x),
               mean(x) + sd(x)/sqrt(length(x)), max(x))
  names(vec_var) <- c("ymin", "lower", "middle", "upper", "ymax")
  vec_var
}
```

Visualize

```
ggplot(data=df_sim2, aes(x=variable, y=value)) +
  geom_boxplot(fill=c("darkblue", "black", "darkred"), alpha=0.4) +
```

```
  labs(title="Plot V",
       subtitle="Ind. Boxplots succesfull trials (k) over probabilities (p)") +
  stat_summary(fun.data=newSummaryStat, geom="boxplot") +
  theme_minimal() +
  ylab("Number of succefull trials (k)") +
  xlab("Probability (p)")
```



Problem 2 (20%)

i) Correlation Plot I

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 "is modeled as") 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 "model" 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.

Load data

```
df_faith <- data.frame(faithful)
```

Table III: OLS Model

```
fit_lm <- lm(waiting ~ eruptions, data=df_faith)
pander(summary(fit_lm), add.significance.stars=TRUE, digits=3, split.table=Inf)
```

	Estimate	Std. Error	t value	Pr(>|t|)	
eruptions	10.7	0.315	34.1	8.13e-100	* * *
(Intercept)	33.5	1.15	29	7.14e-85	* * *

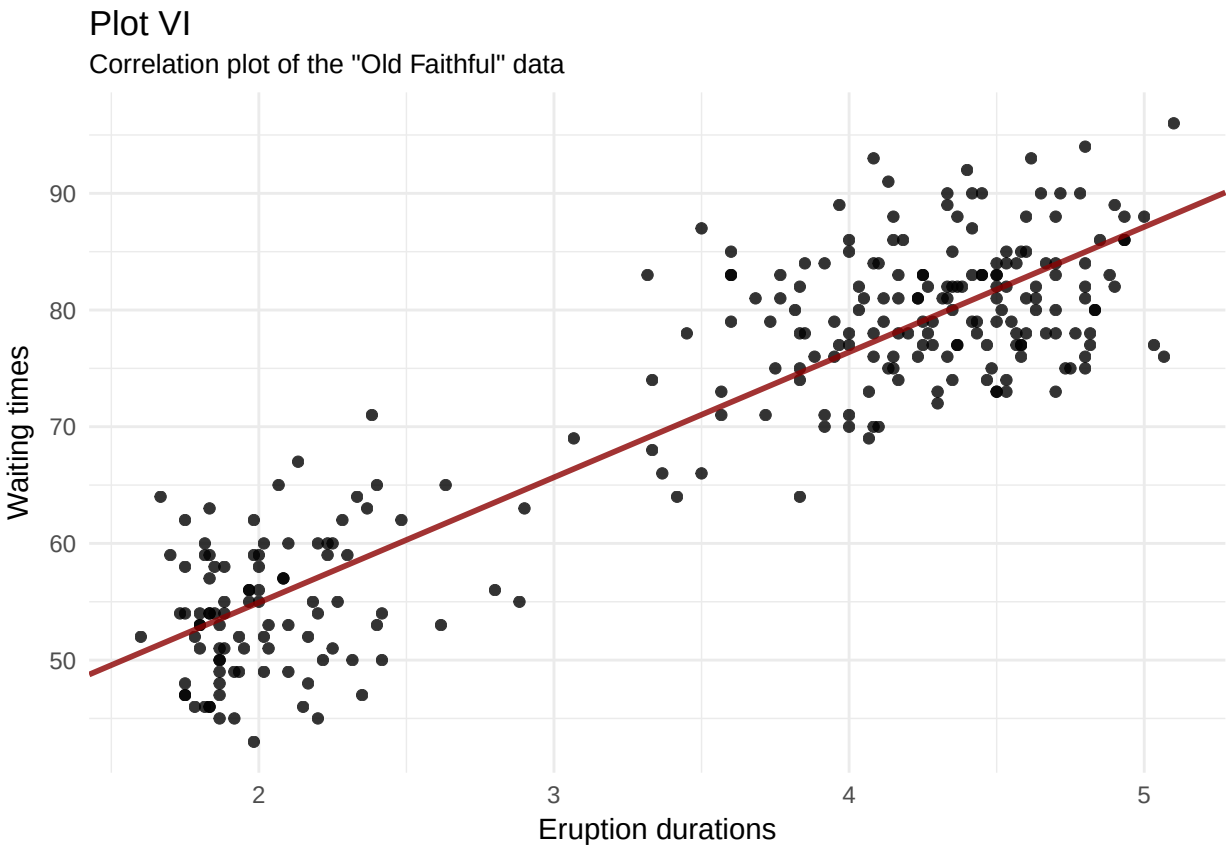
Table 4: Fitting linear model: waiting ~ eruptions

Observations	Residual Std. Error	R^2	Adjusted R^2
272	5.914	0.8115	0.8108

The above output table shows, that the intercept term lies at around 33.5 and the slope is at 10.7.

Visualize

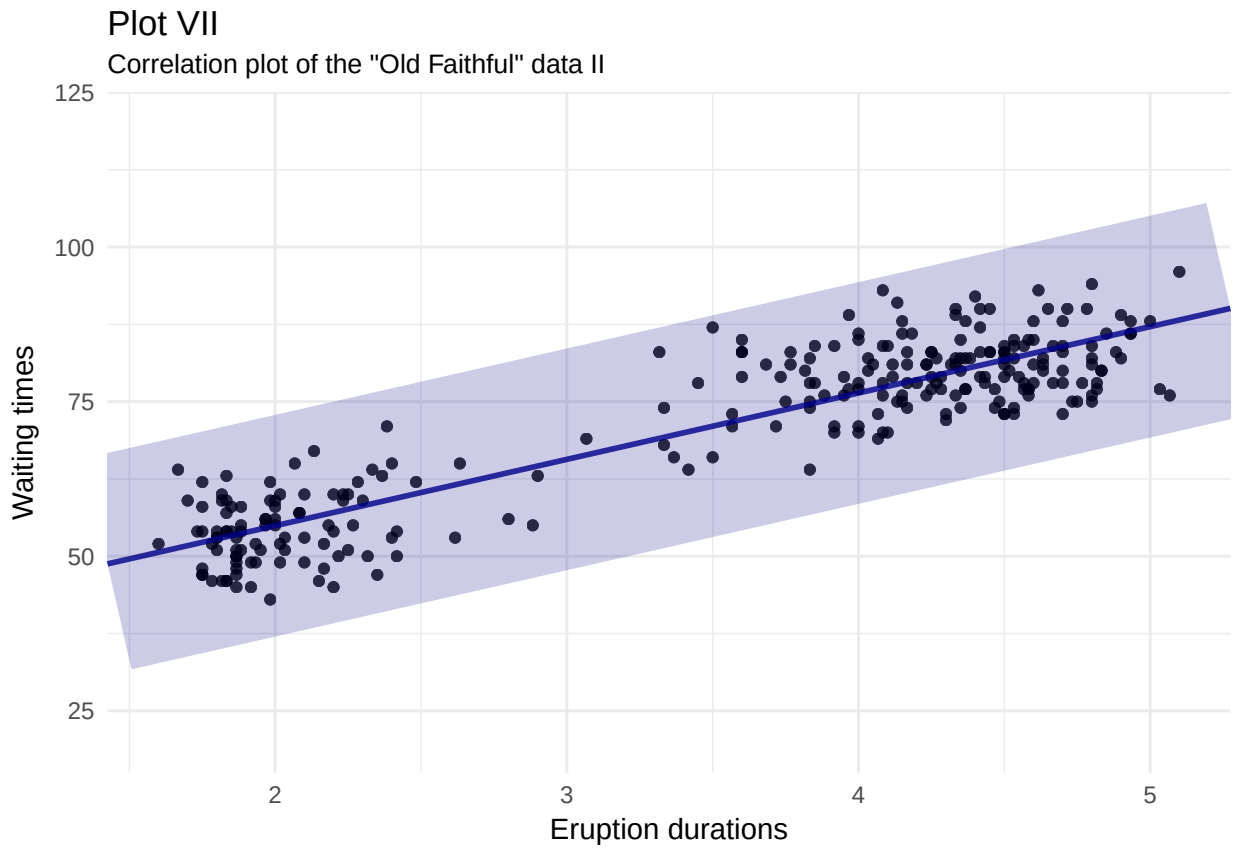
```
ggplot(data=df_faith, aes(x=eruptions, y=waiting)) +
  geom_point(alpha=0.8, shape = 20, color="black",
             fill = "black", size = 3, stroke = 0) +
  labs(title="Plot VI",
       subtitle='Correlation plot of the "Old Faithful" data') +
  theme_minimal() +
  xlab("Eruption durations") +
  ylab("Waiting times") +
  geom_abline(intercept = fit_lm$coefficients[1],
              slope=fit_lm$coefficients[2],
              size=1,
              color="darkred",
              alpha=0.8)
```



ii) Correlation Plot II

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.

```
ggplot(data=df_faith, aes(x=eruptions, y=waiting)) +
  geom_point(alpha=0.8, shape = 20, color="black",
             fill = "black", size = 3, stroke = 0) +
  labs(title="Plot VII",
       subtitle='Correlation plot of the "Old Faithful" data II') +
  theme_minimal() +
  xlab("Eruption durations") +
  ylab("Waiting times") +
  geom_abline(intercept = fit_lm$coefficients[1],
              slope=fit_lm$coefficients[2],
              size=1,
              color="darkblue",
              alpha=0.8) +
  geom_abline(intercept = fit_lm$coefficients[1],
              slope=fit_lm$coefficients[2],
              size=38,
              color="darkblue",
              alpha=0.2) +
  ylim(20, 120)
```



Problem 3 (15%)

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.

Table IV: Covariance matrix

```
# Calculate covariance matrix
mat_cov <- cov(df_faith)
pander(mat_cov)
```

	eruptions	waiting
eruptions	1.303	13.98
waiting	13.98	184.8

The above table shows the co-variance matrix between eruptions and waiting time. The numbers indicate that variance of the eruptions is 1.3, the variance in the waiting time is 184.8 and the covariance between eruptions and watining is 13.98.

Calculate slopes

```
eigen <- eigen(mat_cov)
eigen$slopes[1] <- eigen$vectors[1,1]/eigen$vectors[2,1]  # calc slopes as ratios
eigen$slopes[2] <- eigen$vectors[1,1]/eigen$vectors[1,2]  # calc slopes as ratios
eigen$slopes
```

```
## [1]  0.07572801 -0.07572801
```

As can be seen above, the slopes of the two vectors are orthogonal to each other. Next we can compare all the different values (eigenvector and lm) visually. Before we do this we normalize the data in order to compare like with like.

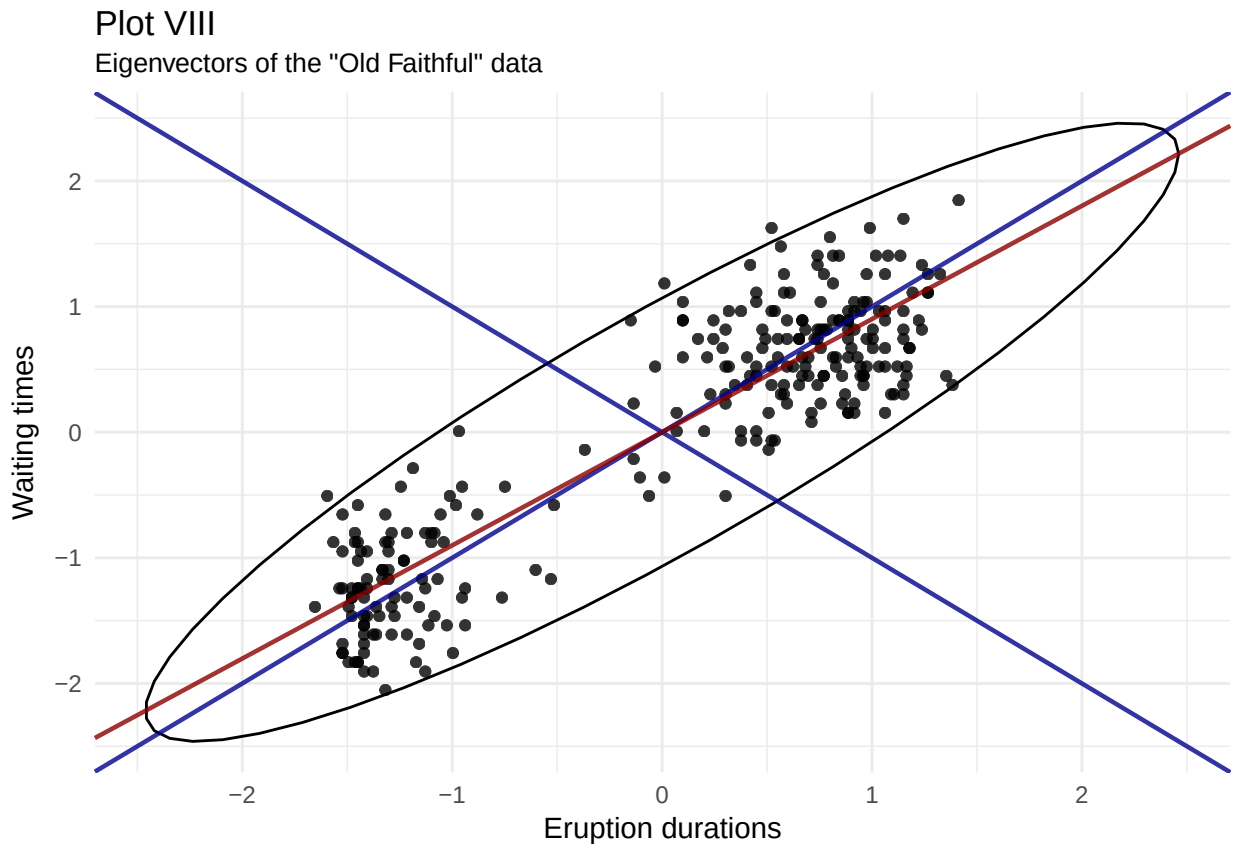
Normalize data and recompute

```
# Normalize
df_faith$eruptions_norm <- (df_faith$eruptions - mean(df_faith$eruptions)) /
  sqrt(var(df_faith$eruptions))
df_faith$waiting_norm <- (df_faith$waiting - mean(df_faith$waiting)) /
  sqrt(var(df_faith$waiting))

# Calculate covariance matrix
cov_mat <- cov(df_faith[, 3:4])
eigen <- eigen(cov_mat)  # calculate eigenvectors and values

# Calculate slopes
eigen$slopes[1] <- eigen$vectors[1,1]/eigen$vectors[2,1]
eigen$slopes[2] <- eigen$vectors[1,1]/eigen$vectors[1,2]

# lm
fit_lm <- lm(waiting_norm ~ eruptions_norm, data=df_faith)
```

Visualize

```
ggplot(data=df_faith, aes(x=eruptions_norm, y=waiting_norm)) +
  labs(title="Plot VIII",
       subtitle='Eigenvectors of the "Old Faithful" data') +
  theme_minimal() +
  xlab("Eruption durations") +
  ylab("Waiting times") +
  geom_point(alpha=0.8, shape = 20, color="black",
             fill = "black", size = 3, stroke = 0) +
  stat_ellipse(type="norm") +
  geom_abline(intercept=0,
              slope=eigen$slopes[1],
              size=0.8,
              alpha=0.8,
              colour = "darkblue") +
  geom_abline(intercept=0,
              slope=eigen$slopes[2],
              size=0.8,
```

```
              alpha=0.8,
              colour = "darkblue") +
  geom_abline(intercept=0,
              slope=fit_lm$coefficients[2],
              size=0.8,
              alpha=0.8,
              colour = "darkred")
```



The above plot shows, that the lm and eigenvectors are not the same. This is because the errors of the ordinary least squares (OLS) procedure and the ordering of the parameters in the formula (x ~ y vs. y ~ x). Using a covariance matrix and its eigenvectors, we're looking at the errors directly orthogonal from each point, lm on the other looks at the points horizontally or vertically. That's why the slopes we get are different.

Problem 4 (20%)

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.

Add a column

```
# Add type
df_faith$type <- ifelse(df_faith$eruptions < 3.1, "short", "long")
```

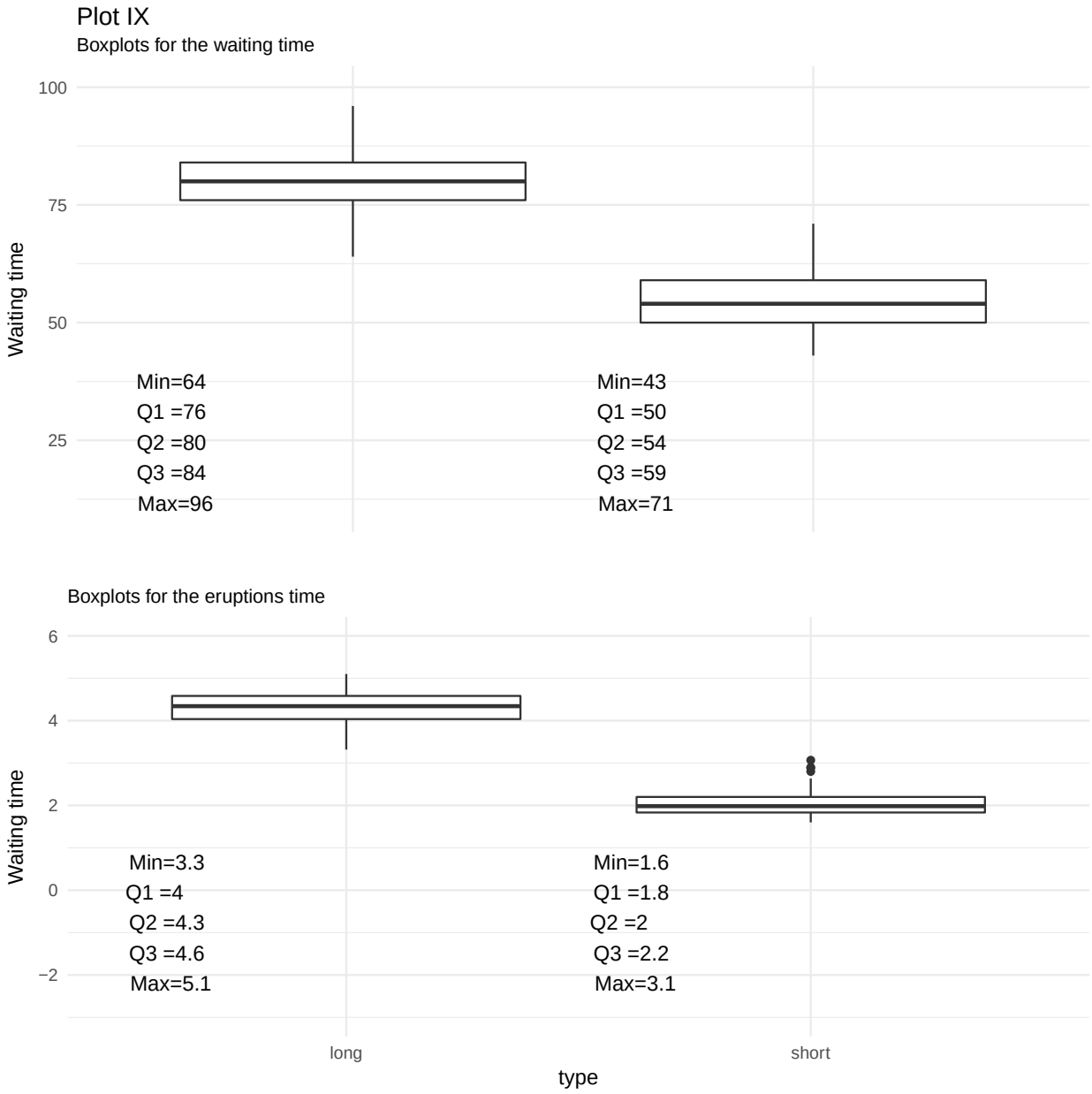
Summary statistics

```
# Get statistics
summaryStat2 <- function(x){
  Min <- min(x)
  Quart_1 <- as.numeric(quantile(x, 0.25))
  Median <- as.numeric(quantile(x, 0.5))
  Quart_3 <- as.numeric(quantile(x, 0.75))
  Max <- max(x)
  data.frame(Min, Quart_1, Median, Quart_3, Max)
}
```

Visualize

```
df_long <- summaryStat2(df_faith$waiting[df_faith$type == "long"])
df_short <- summaryStat2(df_faith$waiting[df_faith$type == "short"])

g1 <- ggplot(data=df_faith, aes(x=type, y=waiting)) +
  geom_boxplot() +
  labs(title="Plot IX",
       subtitle="Boxplots for the waiting time") +
  theme_minimal() +
  theme(axis.text.x = element_blank()) +
  ylab("Waiting time") +
  xlab(NULL) +
  ylim(10, 100) +
  geom_text(data=df_long, aes(label=paste0("Min=", Min, "\n",
                                           "Q1 =", Quart_1, "\n",
                                           "Q2 =", Median, "\n",
                                           "Q3 =", Quart_3, "\n",
                                           "Max=", Max)),
            x=0.5, y=45, hjust=-0.2, vjust=1.2) +
  geom_text(data=df_short, aes(label=paste0("Min=", Min, "\n",
                                           "Q1 =", Quart_1, "\n",
                                           "Q2 =", Median, "\n",
                                           "Q3 =", Quart_3, "\n",
                                           "Max=", Max)),
            x=1.5, y=45, hjust=-0.2, vjust=1.2)


df_long <- round(summaryStat2(df_faith$eruptions[df_faith$type == "long"]), 1)
df_short <- round(summaryStat2(df_faith$eruptions[df_faith$type == "short"]), 1)

g2 <- ggplot(data=df_faith, aes(x=type, y=eruptions)) +
  geom_boxplot() +
  labs(title="",
```

```
       subtitle="Boxplots for the eruptions time") +
  theme_minimal() +
  ylab("Waiting time") +
  ylim(-3, 6) +
  geom_text(data=df_long, aes(label=paste0("Min=", Min, "\n",
                                           "Q1 =", Quart_1, "\n",
                                           "Q2 =", Median, "\n",
                                           "Q3 =", Quart_3, "\n",
                                           "Max=", Max)),
            x=0.5, y=1.5, hjust=-0.2, vjust=1.2) +
  geom_text(data=df_short, aes(label=paste0("Min=", Min, "\n",
                                           "Q1 =", Quart_1, "\n",
                                           "Q2 =", Median, "\n",
                                           "Q3 =", Quart_3, "\n",
                                           "Max=", Max)),
            x=1.5, y=1.5, hjust=-0.2, vjust=1.2)

# Plot grid
grid.arrange(g1, g2, nrow=2, ncol=1)
```



Problem 5 (15%)

Create a matrix with 40 columns and 100 rows. Populate each column with random variable of the uniform distribution with values between -1 and 1 (symmetric around zero).

i) Random uniform distribution

Let the distribution for each column appear like the one on slide 92 of the lecture note, except centered around zero.

Table V: Random uniform matrix

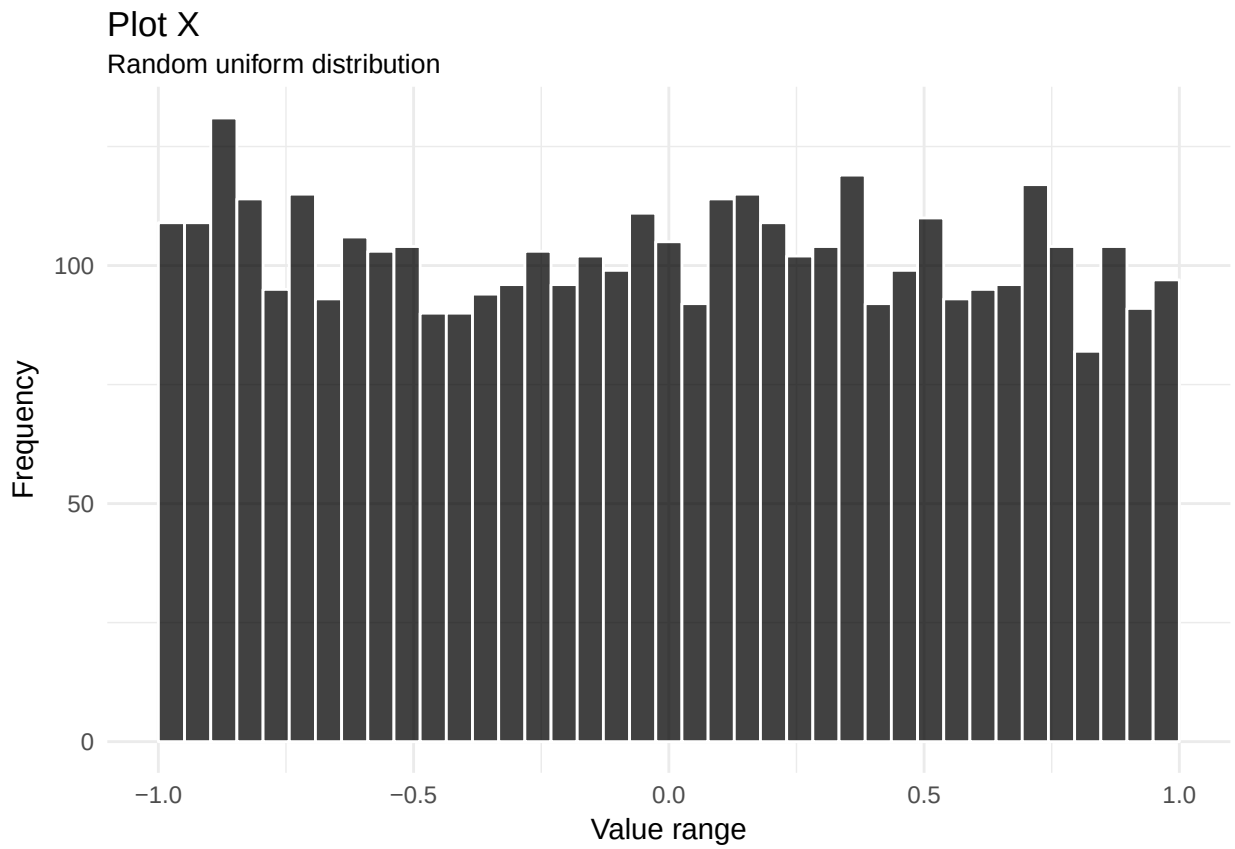
```
set.seed(555)
mat <- matrix(runif(4000, min = -1, max = 1), nrow = 100, ncol = 40)
pander(c(summary(as.vector(mat)), Var=var(as.vector(mat)), Std=sqrt(var(as.vector(mat)))))
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	Var	Std
-0.9992	-0.5259	0.001723	-0.01375	0.4828	0.9998	0.3349	0.5787

The above table shows, that the variables are distributed around 0 and are spanning between -1 and 1.

Visualize

```
ggplot(data=NULL, aes(x=as.vector(mat))) +
  geom_histogram(color="white", fill="black", alpha=0.75, bins=40) +
  labs(title="Plot X",
       subtitle="Random uniform distribution") +
  theme_minimal() +
  xlim(-1, 1) +
  ylab("Frequency") +
  xlab("Value range")
```



ii.) Random columns

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.

Pick two columns

```
# pick two random columns
set.seed(123)
samp_col <- sample(ncol(mat), 2, replace = FALSE)

# create two vectors using above columns
vec1 <- mat[,samp_col[1]]
vec2 <- mat[,samp_col[2]]
```

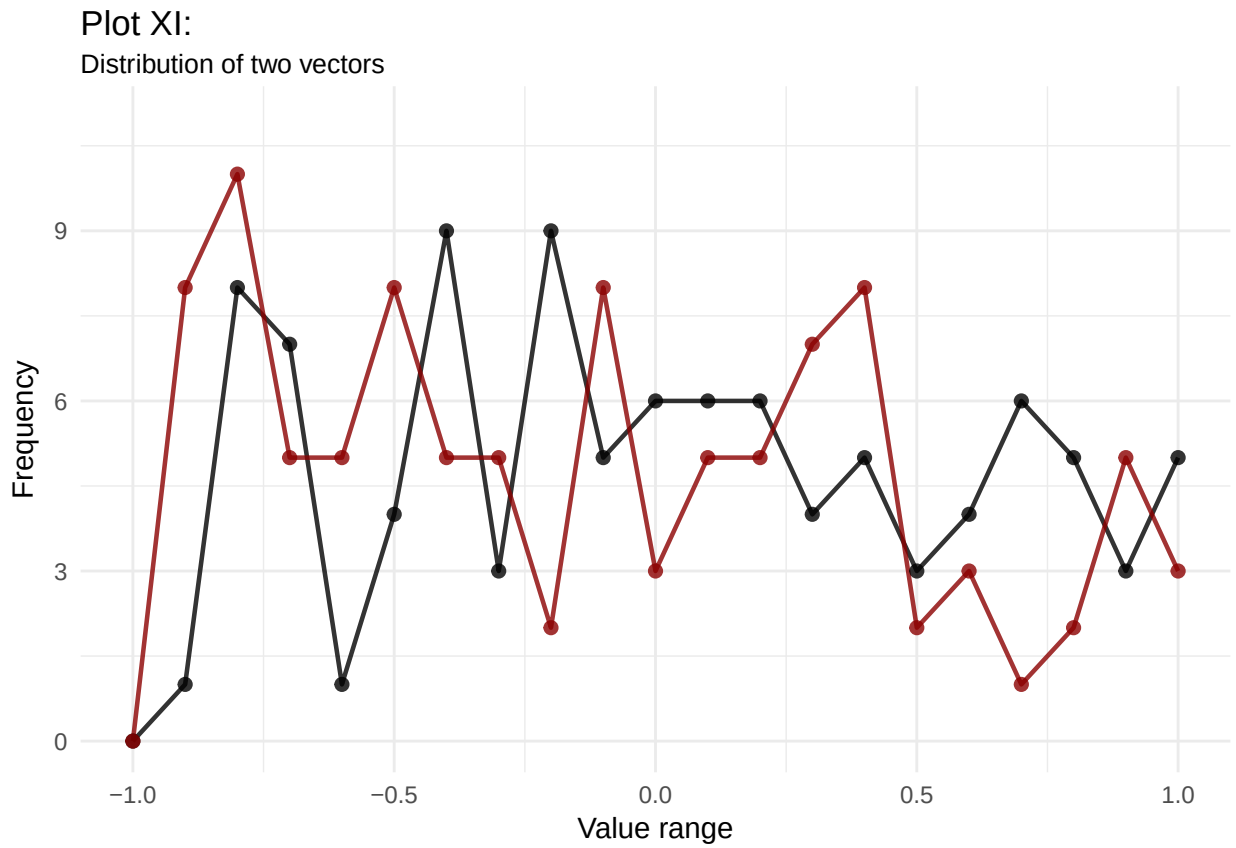
From that a distribution for each vector can be extracted and applied to the two vectors from the previous step.

Get the distributions

```
# Function for the distribution for the two vectors
getDist <- function(vector, breaks){
  vector_cut = cut(vector, breaks, right=FALSE)
  vector_freq = table(vector_cut)
  vector_freq = c(0, vector_freq)

  return(vector_freq)
}

# Getting the distritibutions
breaks = seq(-1, 1, by=0.1)
dist1 <- getDist(vec1, breaks)
dist2 <- getDist(vec2, breaks)

df_dist <- data.frame(breaks, dist1, dist2)
```

Visualization

```
# plot distribution for vector 1
ggplot(data=df_dist, aes(x=breaks)) +
  geom_line(y=dist1, color="black", alpha=0.8, size=0.8) +
  geom_line(y=dist2, color="darkred", alpha=0.8, size=0.8) +
  geom_point(y=dist1, color="black", alpha=0.8, size=2) +
  geom_point(y=dist2, color="darkred", alpha=0.8, size=2) +
  ylim(0, 11) +
  theme_minimal() +
  labs(title="Plot XI:",
       subtitle="Distribution of two vectors") +
  ylab("Frequency") +
  xlab("Value range")
```



The above graph shows that there is no particular structure present in the two vectors. The distribution appears to be random with no particular mean or median beeing visible.

Problem 6 (15%)

i) Calculate row sum

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.

```
mat2 <- cbind(mat, rowSums(mat))
```

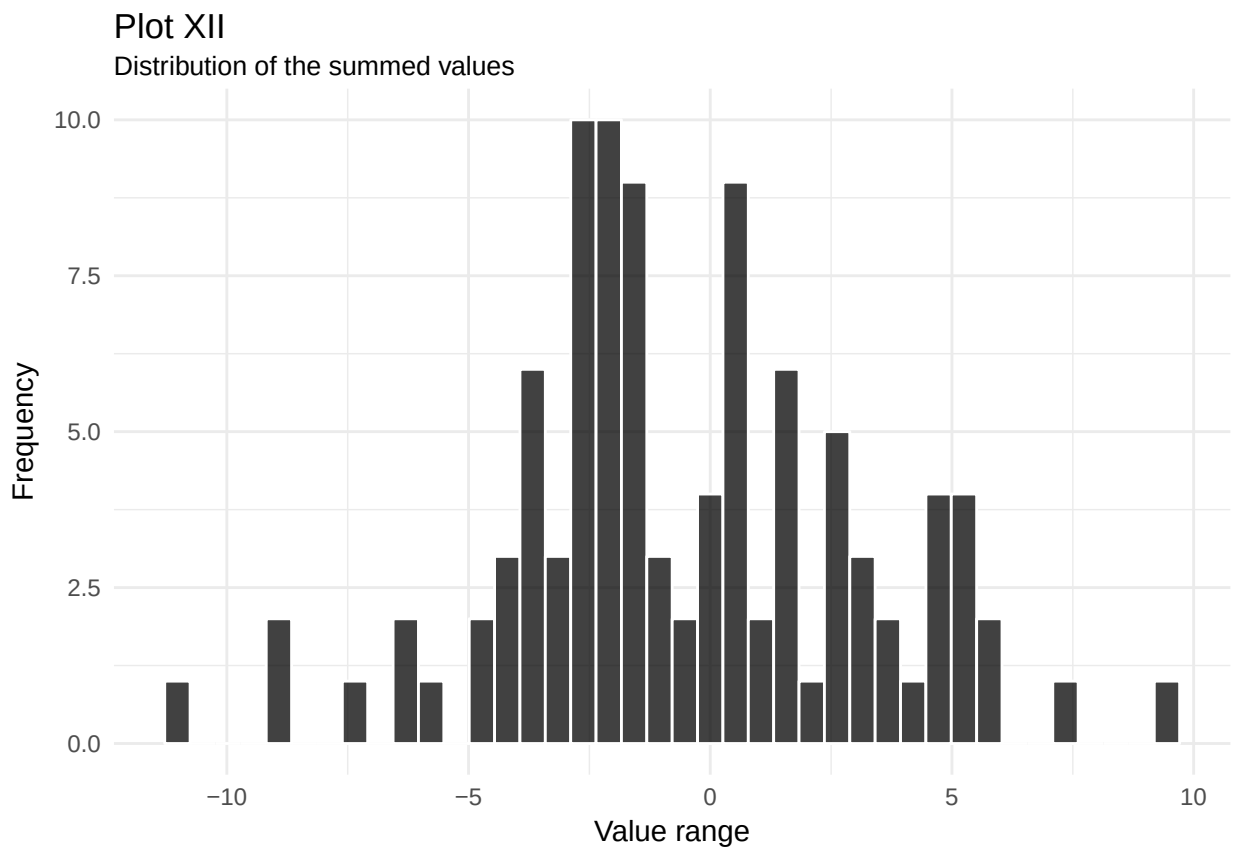
ii) Distribution of the sums

Create a histogram of values in the new column showing that the distribution resembles the Gaussian curve.

Visualize

```
ggplot(data=NULL, aes(x=mat2[, 41])) +
  geom_histogram(color="white", fill="black", alpha=0.75, bins=40) +
  labs(title="Plot XII",
       subtitle="Distribution of the summed values") +
  theme_minimal() +
```

```
  ylab("Frequency") +
  xlab("Value range")
```



iii) Gaussian distribution

Add a true, calculated, Gaussian curve to that diagram with the parameters you expect from the sum of 40 random variables of uniform distribution.

Calculate expected values

From problem 5 we now that x is uniformly distributed, i.e., $X_i \sim U(-1, 1)$ That means we can calculate the first and second *moment* (mean and variance) as follows:

$$\mu_i = \frac{1}{2}(a + b) = \frac{1}{2}(1 - 1) = 0$$

and,

$$\sigma_i^2 = \frac{1}{12}(b - a)^2 = \frac{1}{12}(-1 - 1)^2 = \frac{4}{12} = \frac{1}{4}$$

according to the CLT

$$X_N = \frac{1}{N} * \sum_{i=1}^{n} X_i \sim N(\mu_x,\ \sigma_x^2)$$
$$\mu_x = \mu_i$$
$$\sigma_x^2 = \frac{\sigma_i^2}{N}$$

When we take the column sums it follows:

$$\sum_{i=1}^{n} X_i = N * X_N \sim N(N\mu_x,\ N\sigma_x^2)$$

That means

$$\sum_{i=1}^{n} X_i \sim N(N * \mu_i,\ N * \frac{\sigma_i^2}{N})$$

$$\sum_{i=1}^{n} X_i \sim N(0,\ \frac{1}{4})$$

The above calculation shows, that the expected mean is zero and the expected variance is 1/4 (i.e, the expected standart devation lies at 0.5)

Calculate the sums of each row and plot on a histogram

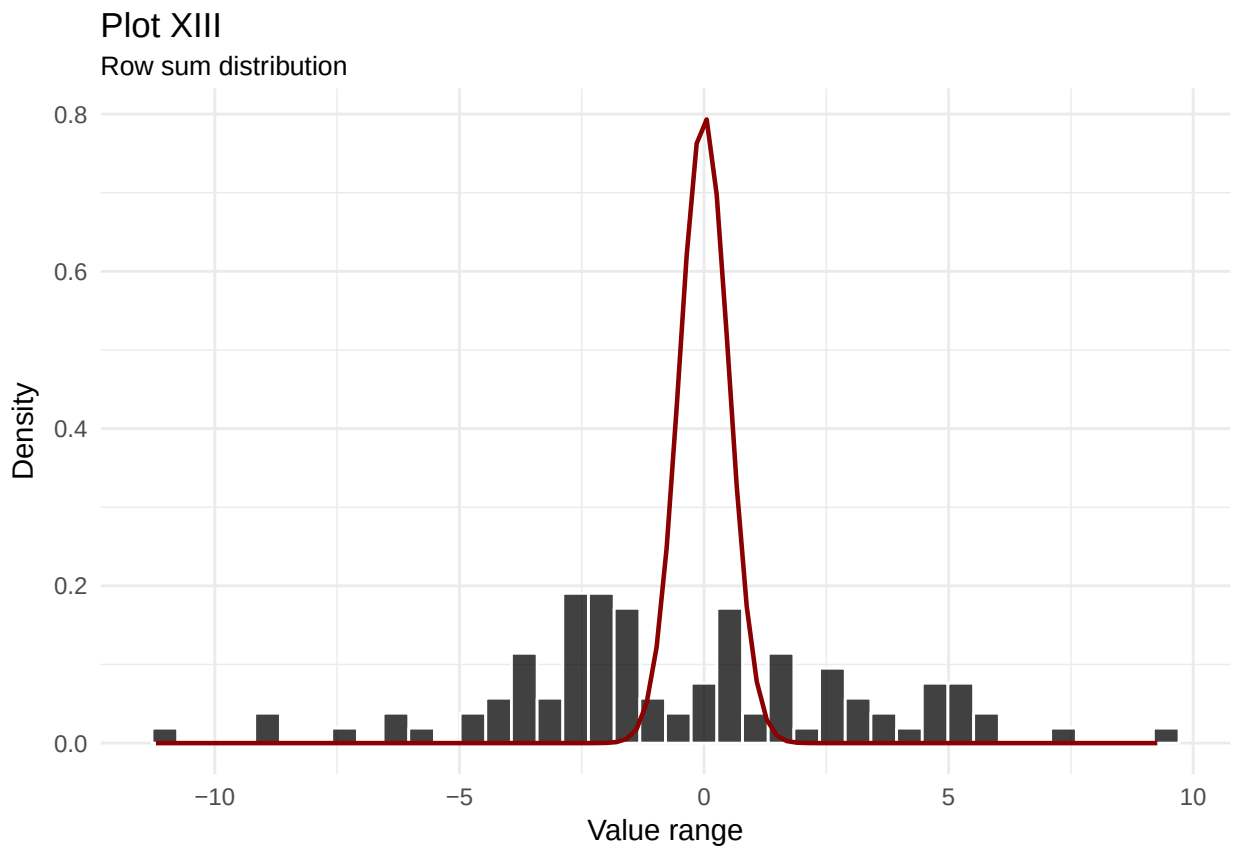
```
mean(x=mat2[, 41])
```

```
## [1] -0.5500212
```

```
sqrt(var(x=mat2[, 41]))
```

```
## [1] 3.62521
```

```
ggplot(data=NULL, aes(x=mat2[, 41])) +
  geom_histogram(color="white", fill="black",
                 alpha=0.75, bins=40,
                 aes(y=..density..)) +
  labs(title="Plot XIII",
       subtitle="Row sum distribution") +
  theme_minimal() +
  xlab("Value range") +
  ylab("Density") +
  stat_function(fun = dnorm,
                args = list(mean = 0, sd = 0.5),
                lwd = 0.8,
                col = 'darkred')
```



The above plot shows that the sum of the random uniform values are starting to appear normally distributed. However, the amount of data doesn't appear to be sufficient yet, to follow the theoretical density curve.