STAT545 HW1

Yi Yang

8/28/2018

Problem 1: A little text-mining.

1. Install the tm package for Text-mining.

```
library(tm)
```

2. Read source files into a variable.

```
dir_of_corpus = file.path('.','data')
dir(dir_of_corpus)
```

```
## [1] "Machine_learning_wiki.txt" "Optimal_control_wiki.txt"
```

```
corpus <- Corpus(DirSource(dir_of_corpus))
```

3. Look at the type of variable corpus, differentiate double and single brackets.

```
typeof(corpus)
```

```
## [1] "list"
```

[ ] is used to return a list of elements indexed by the parameter in the bracket. Instead, [[ ]] is used to return a single element indexed by the parameter in the bracket. For exampel, given an example list *example_* and we use it to illustrate the difference.

```
example_ <- list(1, 2, 3, 'a', 'b', FALSE)
typeof(example_[1])
```

```
## [1] "list"
```

```
typeof(example_[4])
```

```
## [1] "list"
```

```
typeof(example_[[1]])
```

```
## [1] "double"
```

```
typeof(example_[[4]])
```

```
## [1] "character"
```

```
typeof(example_[[6]])
```

```
## [1] "logical"
```

4. Transformation, words pre-processing

```
# replace '/', '@', and '|' with whitespace
toSpace <- content_transformer(function(x,pattern) gsub(pattern,' ',x))
corpus <- tm_map(corpus, toSpace, '/')
corpus <- tm_map(corpus, toSpace, '@')
corpus <- tm_map(corpus, toSpace, '\\|')
# conversion to lower case
corpus <- tm_map(corpus, content_transformer(tolower))
# remove numbers
corpus <- tm_map(corpus, removeNumbers)
# remove punctuation
corpus <- tm_map(corpus, removePunctuation)
# remove English stop words
corpus <- tm_map(corpus, removeWords, stopwords('english'))
# remove own stop words
corpus <- tm_map(corpus, removeWords, c('eat','breakfast','teacher'))
# strip whitespace
corpus <- tm_map(corpus, stripWhitespace)
# specific transformations to be cared about
toAbbrev <- content_transformer(function(x, from, to) gsub(from, to, x))
corpus <- tm_map(corpus, toAbbrev, 'Massachusetts Institute of Technology', 'MIT')
corpus <- tm_map(corpus, toAbbrev, 'University of California at Berkeley', 'UCB')
```

The code we tried here has implemented following functions:

- remove '/', '@,'|' and replace them with whitespace
- make all letters lower case
- remove all numbers
- remove all punctuations
- remove English stop words
- remove self-defined stop words
- strip whitespaces
- abbreviate several common words

5. Convert corpus into a document term matrix

```
dtm_of_corpus <- DocumentTermMatrix(corpus)
inspect(dtm_of_corpus[1:2, c('control', 'algorithm')])
```

```
## <<DocumentTermMatrix (documents: 2, terms: 2)>>
## Non-/sparse entries: 3/1
## Sparsity           : 25%
## Maximal term length: 9
## Weighting          : term frequency (tf)
## Sample             :
##                              Terms
## Docs                           algorithm control
##   Machine_learning_wiki.txt        14       1
##   Optimal_control_wiki.txt          0      87
```

6. Frequency analysis

- calculate frequency for each document and the overall document

```
# calculate the frequency for each document and all the document
dtm_of_corpus1 <- DocumentTermMatrix(corpus[1])
dtm_of_corpus2 <- DocumentTermMatrix(corpus[2])
freq1 <- colSums(as.matrix(dtm_of_corpus1))
freq2 <- colSums(as.matrix(dtm_of_corpus2))
freq_tol <- colSums(as.matrix(dtm_of_corpus))
length(freq1)
```

```
## [1] 1846
```

```
length(freq2)
```

```
## [1] 778
```

```
length(freq_tol)
```

```
## [1] 2346
```

- sort term frequency and get the top 15

```
# calculate the relatvie frequency for each document and all the document
rel_freq1 <- freq1 / sum(freq1)
rel_freq2 <- freq2 / sum(freq2)
rel_freq_tol <- freq_tol / sum(freq_tol)
# sort frequency in descending order
ord_of_freq1 <- order(freq1, decreasing = TRUE)
ord_of_freq2 <- order(freq2, decreasing = TRUE)
ord_of_freq_tol <- order(freq_tol, decreasing = TRUE)
# list top 15 words
top15_of_doc1 <- freq1[ord_of_freq1[1:15]]
top15_of_doc1
```

```
##   learning    machine       data algorithms      https  retrieved
##        189        113         54         29         29         26
##      model          -   training artificial        can        set
##         25         22         21         20         20         18
##    systems     neural       also
##         17         16         15
```

```
top15_of_doc2 <- freq2[ord_of_freq2[1:15]]
top15_of_doc2
```

```
##      control      optimal      problem         time optimization
##           87           62           35           22           20
##     problems       method       direct          can      methods
##           20           18           17           16           16
##        using         cost     solution     software            -
##           16           15           13           12           12
```

```
top15_of_docs <- freq_tol[ord_of_freq_tol[1:15]]
top15_of_docs
```

```
##   learning    machine    control    optimal       data    problem
##        189        113         88         62         54         43
##        can          -      https    methods       time algorithms
```

```
##         36          34          33          30          30          29
##     method    problems   retrieved
##         28          27          27
```

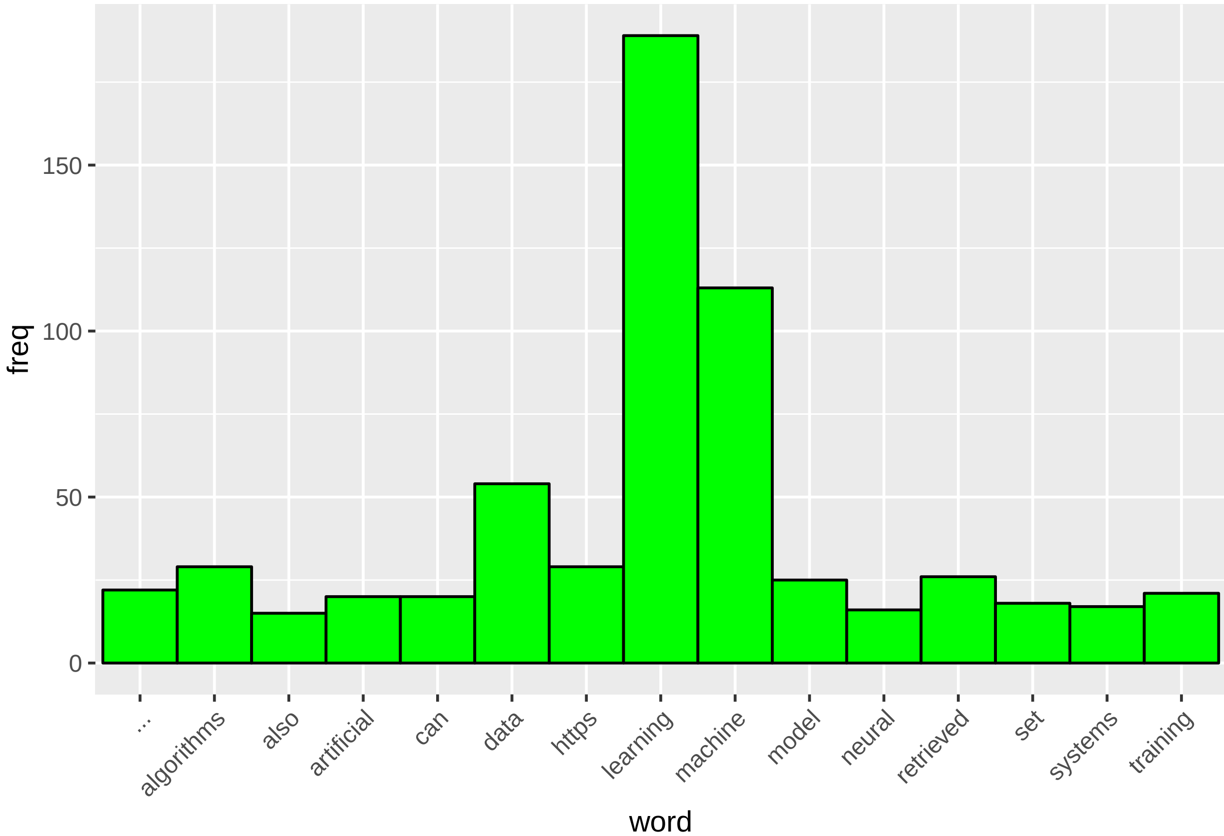
- plot the histogram of the top 15

```r
library(ggplot2)
```

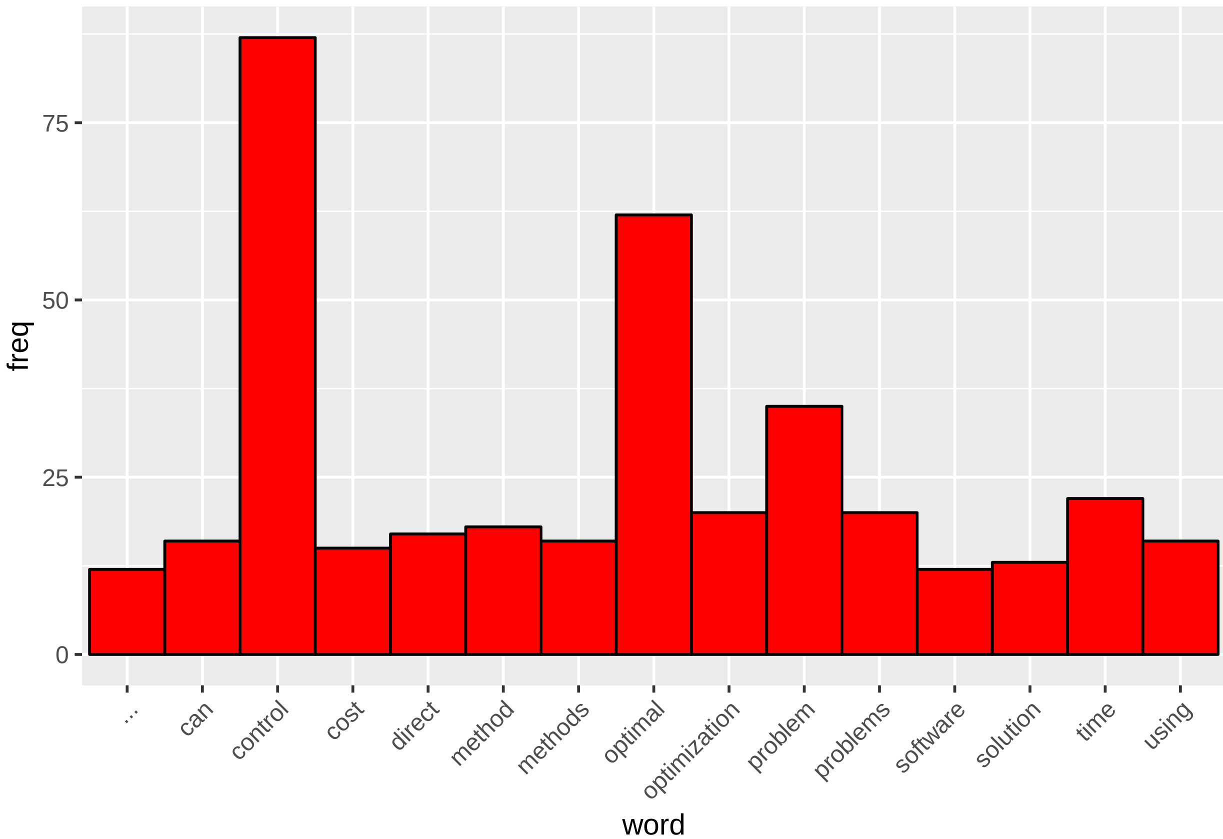
```
##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:NLP':
##
##     annotate
```

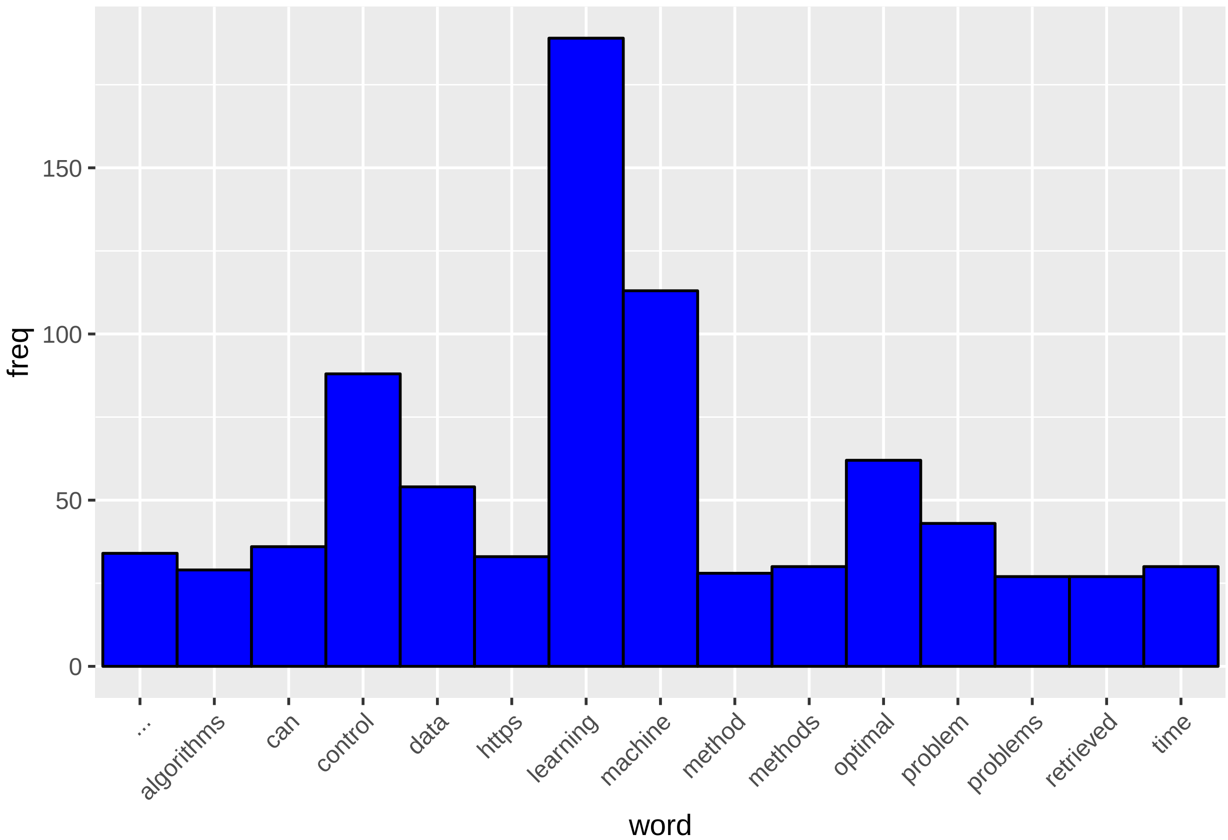
```r
# plot the histogram for the first document
wf1 <- data.frame(word=names(top15_of_doc1), freq=top15_of_doc1)
#head(wf1)
ggplot(data=wf1, aes(word, freq)) + geom_bar(stat = 'identity', width = 1, color = 'black', fill = 'gree
  theme(axis.text.x=element_text(angle = 45, hjust=1))
```




```r
# plot the histogram for the second document
wf2 <- data.frame(word=names(top15_of_doc2), freq=top15_of_doc2)
#head(wf2)
ggplot(data=wf2, aes(word, freq)) + geom_bar(stat = 'identity', width = 1, color = 'black', fill = 'red
  theme(axis.text.x=element_text(angle = 45, hjust=1))
```




```
# plot the histogram for all the documents
wf <- data.frame(word=names(top15_of_docs), freq=top15_of_docs)
#head(wf)
ggplot(data=wf, aes(word, freq)) + geom_bar(stat = 'identity', width = 1, color = 'black', fill = 'blue
  theme(axis.text.x=element_text(angle = 45, hjust=1))
```



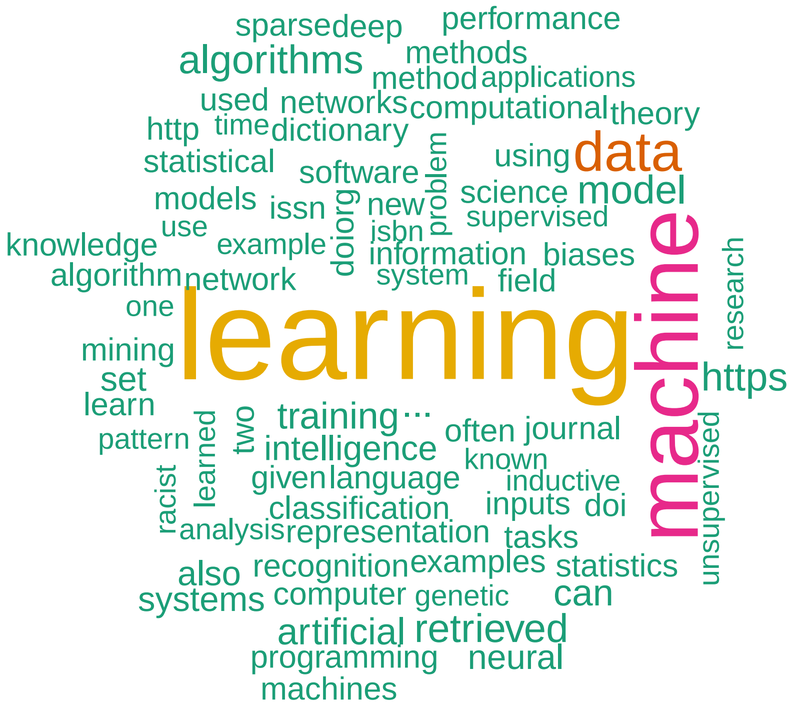
7. Produce wordclouds

- word cloud for the first document

```
# word cloud for the first document
library(wordcloud)
```

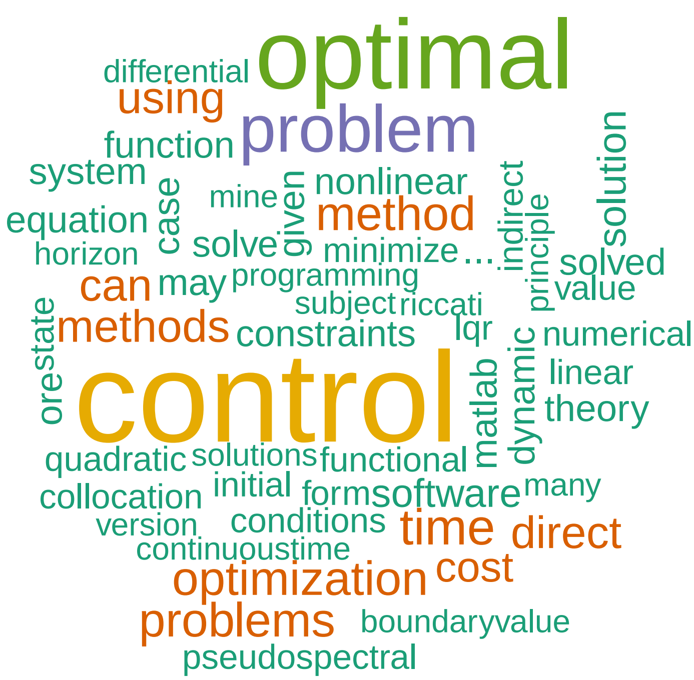
```
## Loading required package: RColorBrewer
```

```
set.seed(132)
wordcloud(names(freq1), freq1, max.words=200, min.freq=8, scale = c(4,.8) , colors = brewer.pal(6,'Dark2
```



- word cloud for the second document

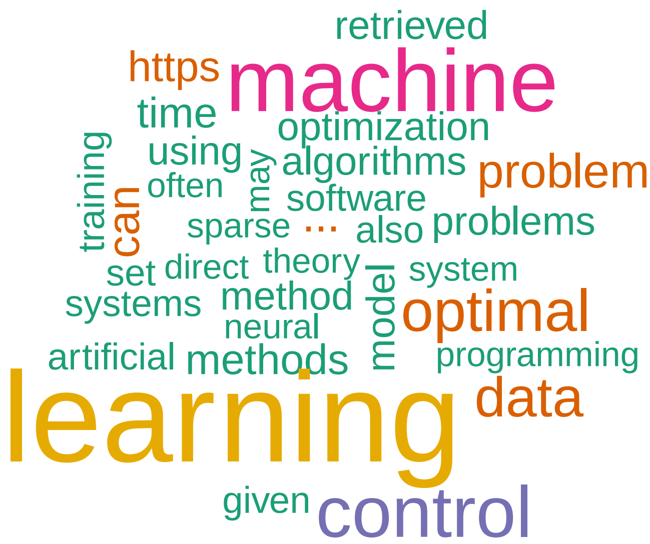
```
# word cloud for the second document
library(wordcloud)
set.seed(142)
wordcloud(names(freq2), freq2, max.words=200, min.freq=6, scale = c(4,.8) , colors = brewer.pal(6,'Dark
```



- word cloud for the whole documents

```
# word cloud for the whole documents
library(wordcloud)
```

```
set.seed(162)
wordcloud(names(freq_tol), freq_tol, max.words=200, min.freq=16, scale = c(4,.8) , colors = brewer.pal(
```



Problem 2: Checking the central limit theorem.

1. Write a function to evaluate the central limit theorem.

Central limit theorem (CLT) states that: Let $X_1, X_2, \ldots, X_n$ be a random sample from a distribution with mean μ and variance σ^2. Then if n is sufficiently large, $\bar{X}$ has approximately a normal distribution with $\mu_{\bar{X}} = \mu$ and $\sigma^2_{\bar{X}} = \sigma^2/n$, and $T_0 = X_1 + X_2 + \cdots + X_n$ also has approximately a normal distribution with $\mu_{T_0} = n\mu$ and $\sigma^2_{T_0} = n\sigma^2$. The larger the value of n, the better the approximation.

For this reason, if $X_i \sim Unif(-1, 1)$ with $\mu_{X_i} = 0$ and $\sigma^2_{X_i} = \frac{1}{3}$, we add a scalar a such that $Y = a\sqrt{n}T_0 = a\sqrt{n}\sum_{i=1}^{n} X_i \sim N(0, \frac{1}{3}n^2a^2)$. In order to make the new random variable Y has variance of unity, the scalar $a = \frac{\sqrt{3}}{n}$.

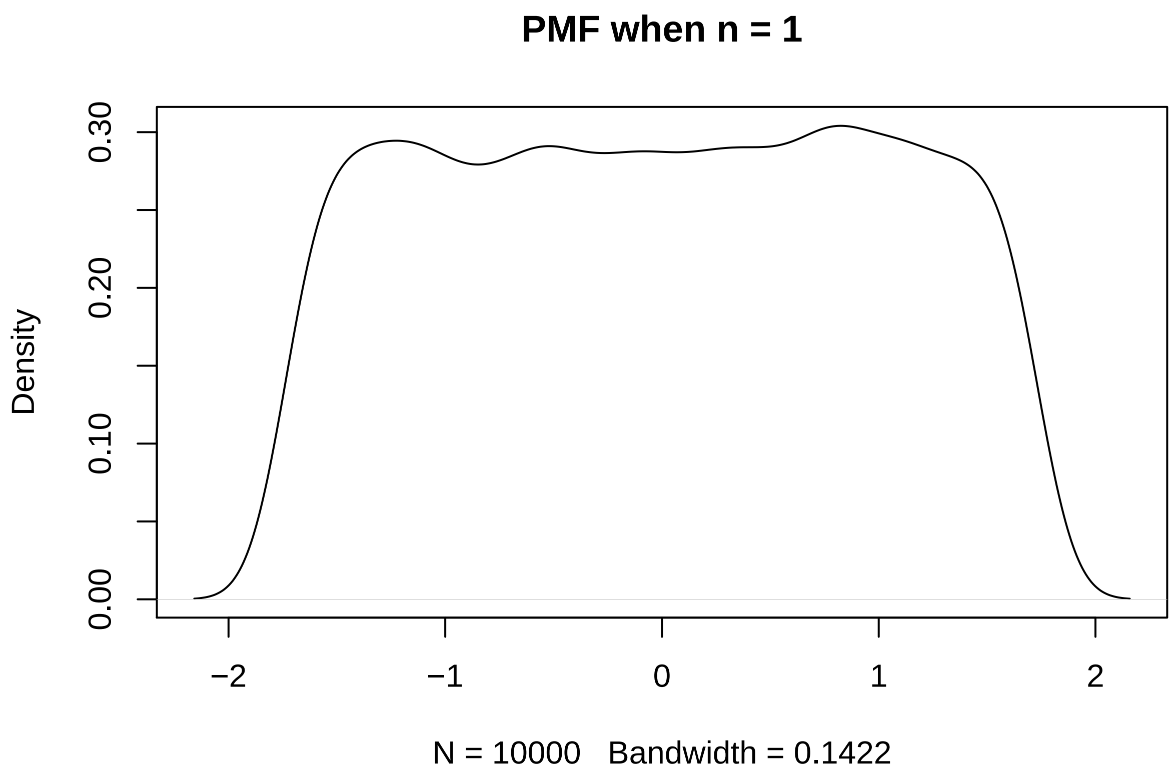
- random varible generating function

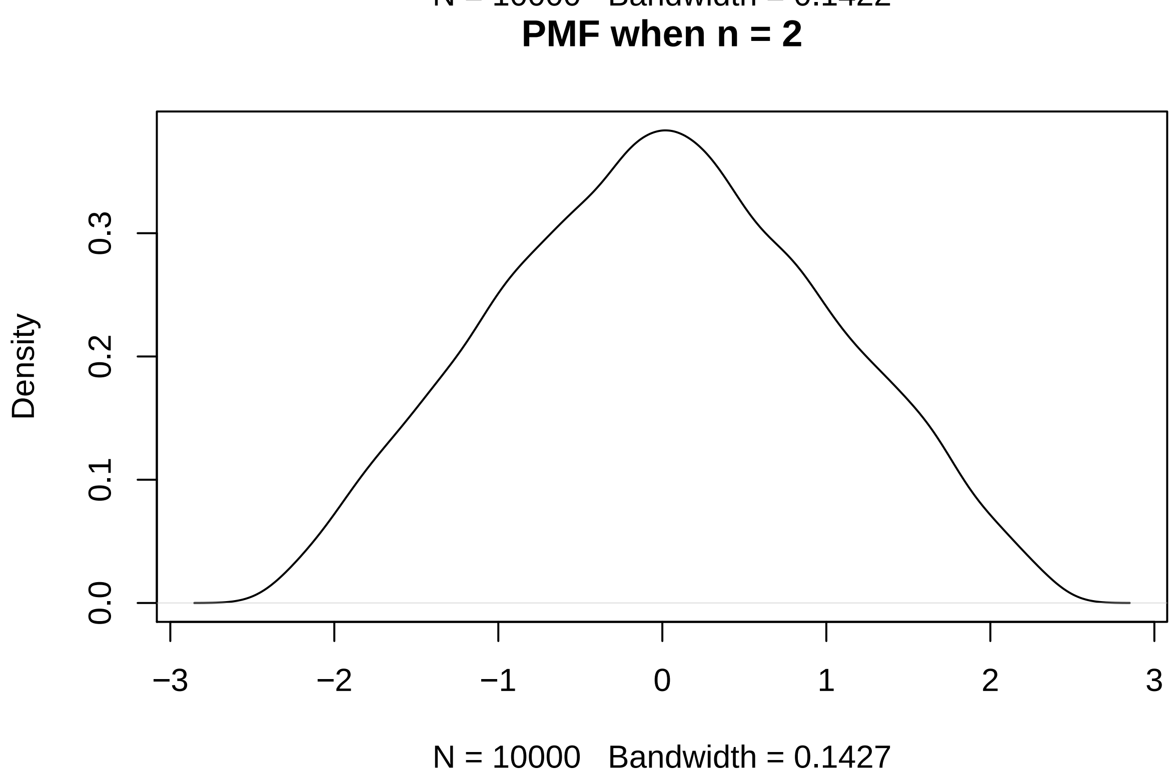
```
my_CLT <- function(n, m){
  a <- sqrt(3) / n
  unif_data <- replicate(m, runif(n, -1, 1))
  if (is.null(dim(unif_data))){
    new_rvs <- unif_data
  } else{
    new_rvs <- colSums(unif_data)
  }
  scaled_new_rvs <- new_rvs * a * sqrt(n)
  return(scaled_new_rvs)
}
```

2. For m = 10000, plot the density of Y for n = 1,2,3,5,10,15 and 20.

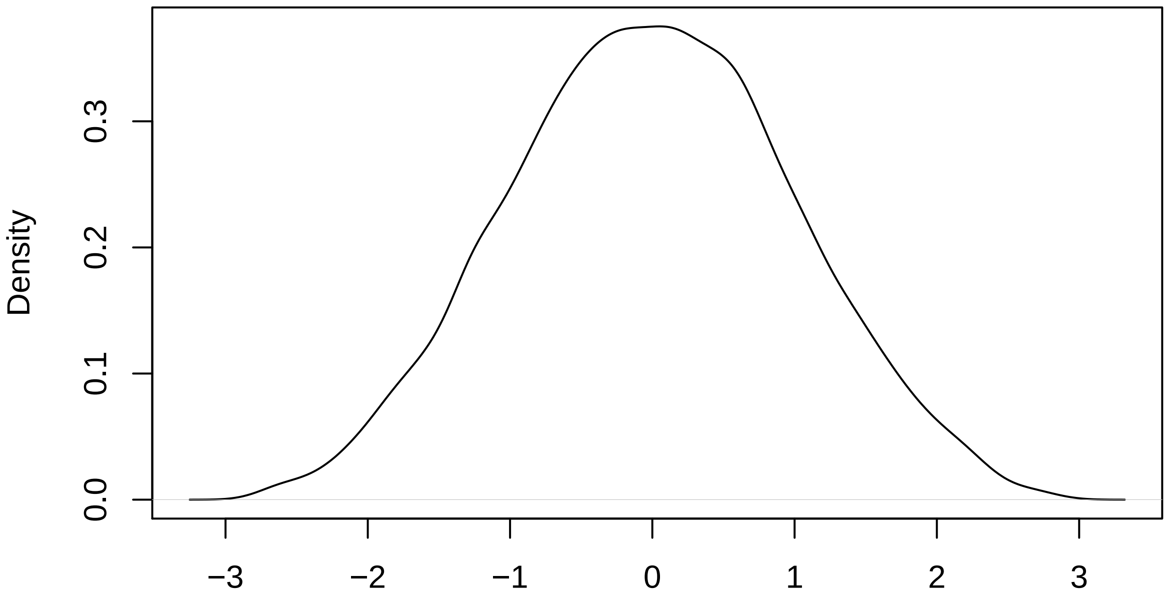
```
m <- 10000
N <- c(1, 2, 3, 5, 10, 15, 20)
```

```
for (n in N){
  rvs <- my_CLT(n, m)
  plot(density(rvs), main = paste0('PMF when n = ', n))
}
```



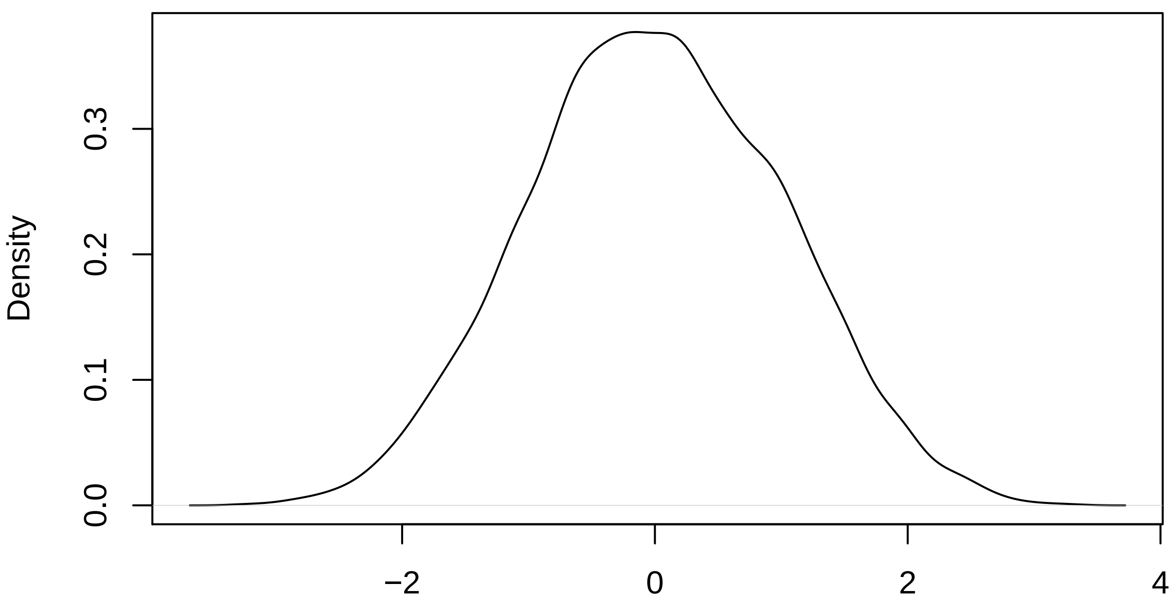


PMF when n = 3

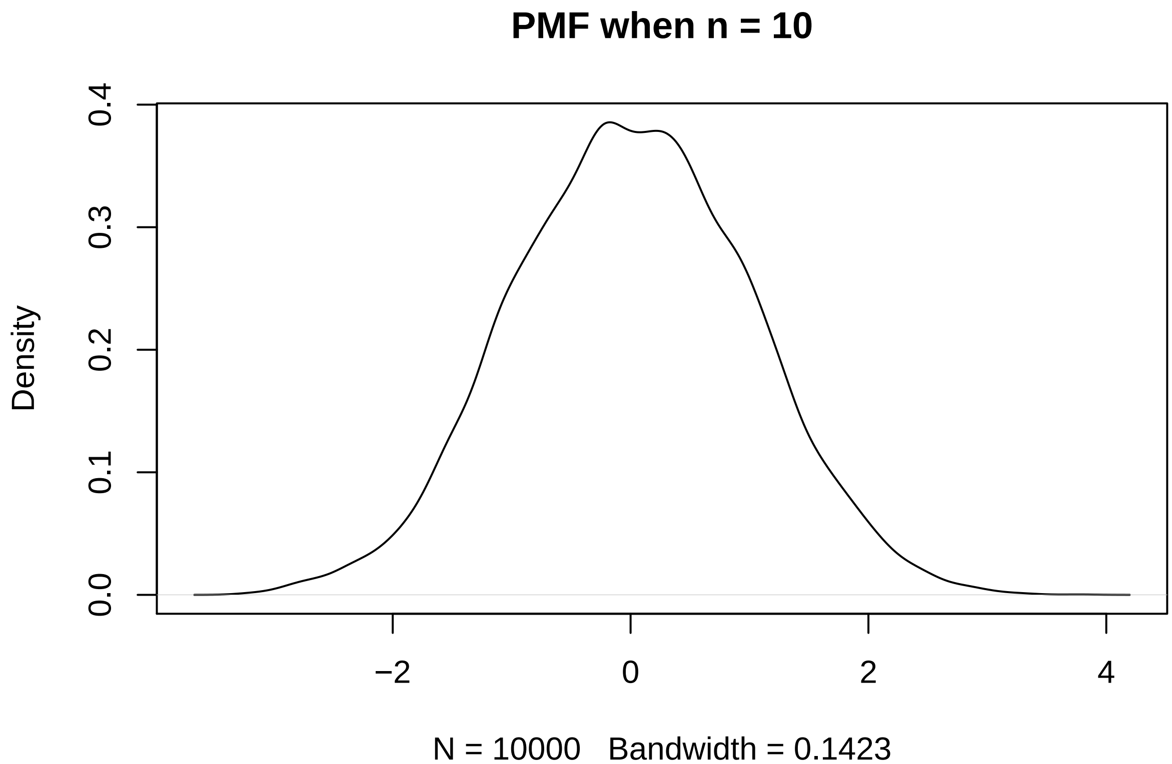


N = 10000   Bandwidth = 0.1418

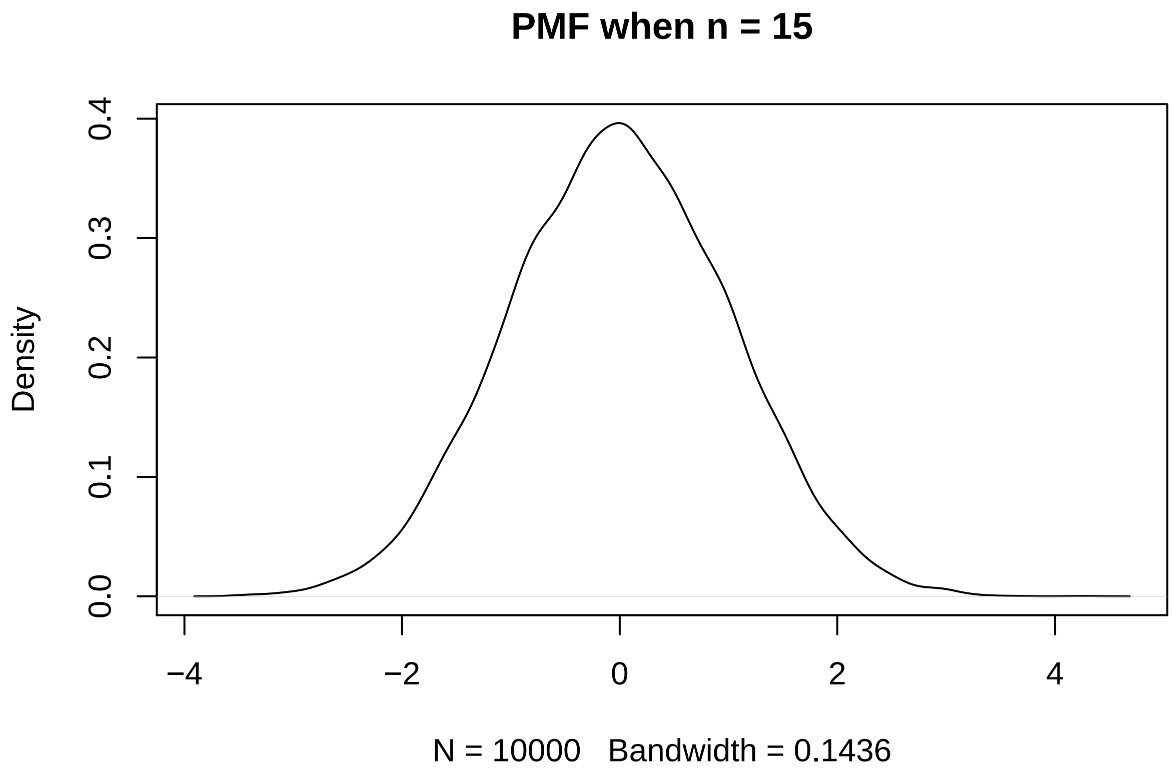
PMF when n = 5



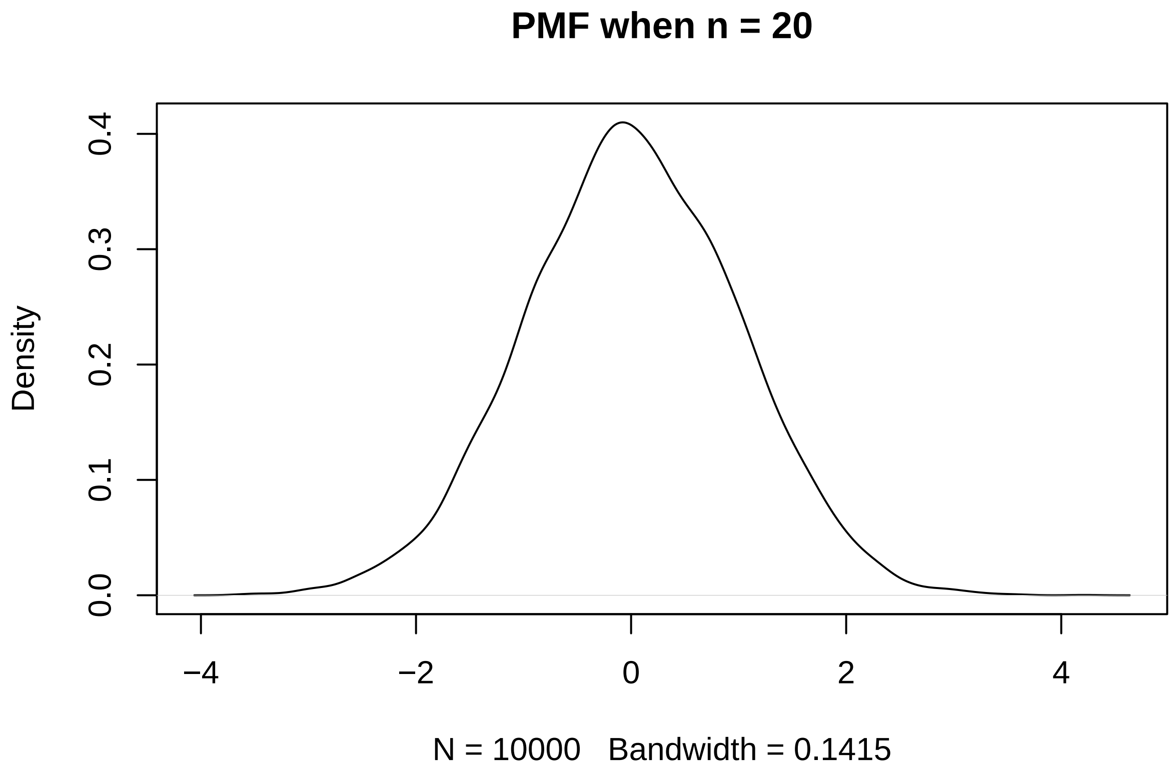
N = 10000   Bandwidth = 0.1426



PMF when n = 10
Density
0.0
0.1
0.2
0.3
0.4
−2
0
2
4
N = 10000 Bandwidth = 0.1423



PMF when n = 15
Density
0.0
0.1
0.2
0.3
0.4
−4
−2
0
2
4
N = 10000 Bandwidth = 0.1436



Problem 3: Permutation test

1. Calculate the difference in the average response time of the two groups.

```
reaction_times <- read.table('./data1/reaction_times.txt', header = TRUE, sep = ',')
avg_restime1 <- mean(reaction_times$ReactionTime[1:100])
avg_restime1
```

```
## [1] 1.03047
```

```
avg_restime2 <- mean(reaction_times$ReactionTime[101:200])
avg_restime2
```

```
## [1] 1.28203
```

```
diff_avg <- avg_restime1 - avg_restime2
diff_avg
```

```
## [1] -0.25156
```

2. Randomize users in two groups and compare the average response time.

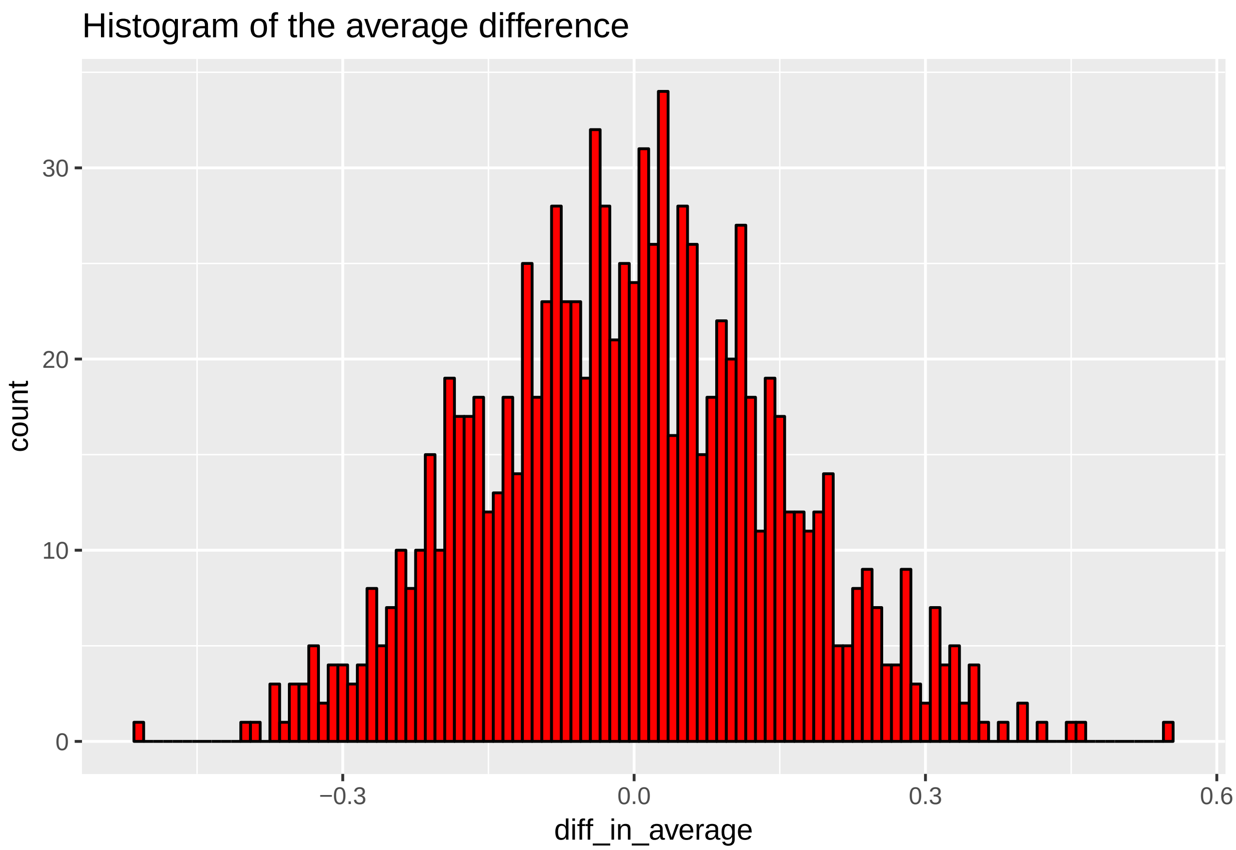
```
# randomly assign users to two groups with the same size 100
shuffled_UserID <- sample(reaction_times$UserID)
shuffled_rts <- reaction_times[shuffled_UserID,]
shuffled_avg_restime1 <- mean(shuffled_rts$ReactionTime[1:100])
shuffled_avg_restime2 <- mean(shuffled_rts$ReactionTime[101:200])
```

```
diff_avg_shuffled <- shuffled_avg_restime1 - shuffled_avg_restime2
diff_avg_shuffled
```

```
## [1] -0.03503293
```

3. Repeat for 1000 times

```
diffs_avg_shuffled <- c()
for (id in c(1:1000)){
  shuffled_UserID <- sample(reaction_times$UserID)
  shuffled_rts <- reaction_times[shuffled_UserID,]
  shuffled_avg_restime1 <- mean(shuffled_rts$ReactionTime[1:100])
  shuffled_avg_restime2 <- mean(shuffled_rts$ReactionTime[101:200])
  diff_avg_shuffled <- shuffled_avg_restime1 - shuffled_avg_restime2
  diffs_avg_shuffled <- c(diffs_avg_shuffled, diff_avg_shuffled)
}
diffs_avg_shuffled_df <- data.frame(diff_in_average=diffs_avg_shuffled)
ggplot(data = diffs_avg_shuffled_df, aes(x=diff_in_average)) + geom_histogram(binwidth=0.01, color='bla
```



Let us consider a two-sided test with significance level to be $\alpha = 0.05$ with large number samples, in which case the test statiscs should satisfy a standard normal distribution.

```
# calculate the test statistics
# sample size
n <-  length(diffs_avg_shuffled)
# test statistics
Ts <- (mean(diffs_avg_shuffled) - 0) / (sd(diffs_avg_shuffled) / sqrt(n))
```

```r
qnorm(0.975)
```

```
## [1] 1.959964
```

```r
pvalue <- (1 - pnorm(abs(Ts)))*2
pvalue
```

```
## [1] 0.3434277
```

since *pvalue* is greater than α, it means we can not reject the null hypothesis which states there is no group difference. We can use another r function to do the hypothesis testing.

```r
library(TeachingDemos)
z.test(diffs_avg_shuffled,stdev = sd(diffs_avg_shuffled), alternative = 'two.sided')
```

```
##
##  One Sample z-test
##
## data:  diffs_avg_shuffled
## z = -0.94741, n = 1.0000e+03, Std. Dev. = 1.5446e-01, Std. Dev. of
## the sample mean = 4.8844e-03, p-value = 0.3434
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.014200877  0.004945726
## sample estimates:
## mean of diffs_avg_shuffled
##               -0.004627576
```

Porblem 4: The power method

1. Genrate a symmetric matrix and solve for its eigenvalues.

- generate a symmetric matrix

```r
set.seed(13)
upper_element <- runif(15, min = -1, max = 1)
A <- matrix(0, nrow = 5, ncol = 5)
A[upper.tri(A, diag = TRUE)] <- upper_element
A[lower.tri(A, diag = FALSE)] <- t(A)[lower.tri(A, diag = FALSE)]
A
```

```
##            [,1]       [,2]       [,3]       [,4]      [,5]
## [1,]  0.4206449 -0.5077254 -0.8172327  0.1485904 0.3222432
## [2,] -0.5077254 -0.2207311  0.9241291  0.5287960 0.7567417
## [3,] -0.8172327  0.9241291 -0.9781333  0.7467646 0.7811181
## [4,]  0.1485904  0.5287960  0.7467646 -0.9178733 0.1325609
## [5,]  0.3222432  0.7567417  0.7811181  0.1325609 0.1870947
```

- print out the eigenvalues and eigenvectors

```r
eigen(A)
```

```
## eigen() decomposition
## $values
## [1]  1.7788857  0.6812455 -0.6353990 -1.2442201 -2.0895104
##
```

```
## $vectors
##            [,1]        [,2]       [,3]       [,4]        [,5]
## [1,] -0.3785588  0.80695629 -0.3019174 -0.1498579 -0.30315548
## [2,]  0.5706018  0.05423601 -0.2457639 -0.7792633  0.06181275
## [3,]  0.5030766 -0.04275255 -0.1341220  0.3458687 -0.77940507
## [4,]  0.2530744  0.12053825 -0.6991825  0.4520970  0.47767773
## [5,]  0.4625734  0.57404072  0.5844665  0.2151148  0.26196879
```

2. Implement the power method to solve the first eigenvector and its eigenvalue

```r
# uisng power method to compute the dominant eigenvalue and its eigenvector
power_method <- function(A){
  x0 <- rep(1, ncol(A))
  iter <- 0
  iter_max <- 100
  error <- 1
  while (iter < iter_max && error > 1e-3) {
    x1 <- A %*% x0
    x1_normed <- x1 / norm(x1, type='2')
    error <- max(abs(x1_normed - x0))
    x0 <- x1_normed
    iter <- iter + 1
  }
  eigenVector <- x0
  eigenValue <- sum(A %*% eigenVector * eigenVector) / sum(eigenVector^2)
  return(list(eigen_value = eigenValue, eigen_vector = eigenVector))
}

eigenResults <- power_method(A)
eigenResults
```

```
## $eigen_value
## [1] -2.08951
##
## $eigen_vector
##             [,1]
## [1,]  0.30315529
## [2,] -0.06181246
## [3,]  0.77940533
## [4,] -0.47767760
## [5,] -0.26196855
```

3. Apply the power function to compute the second dominant eigenvalue and eigenvector.

Let us see the intuition of the theorem: Let A be an $n \times n$ matrix with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ and associated eigenvectors $v_1, v_2, \ldots, v_n$, and let x be any n-vector for which $x^T v_1 = 1$. Then the matrix

$$B = A - \lambda_1 v_1 x^T$$

has eigenvalues $0, \lambda_2, \lambda_3, \ldots, \lambda_n$ with assoicated eigenvectors $v_1, u_2, u_3, \ldots, u_n$ where for $i = 2, 3, \ldots, n$,

$$v_i = (\lambda_i - \lambda_1)u_i + \lambda_1(x^T u_i)v_i$$

Let us give a proof for the case in which A is a symmetric real matrix, in whih case the eigenvectors are orthogonal associated with distinct eigenvalues are orthogonal to each other. which means $Bv_i = Av_i - \lambda_1 v_1 x^T v_i = Av_i = \lambda_i v_i$.

```
lam1 <- eigenResults$eigen_value
vec1 <- eigenResults$eigen_vector
B <- A - lam1 * vec1 %*% t(vec1)
power_method(B)
```

```
## $eigen_value
## [1] 1.778885
##
## $eigen_vector
##            [,1]
## [1,] -0.3786171
## [2,]  0.5702908
## [3,]  0.5032150
## [4,]  0.2532549
## [5,]  0.4626597
```

Problem 5: Briefly answer the following questions

1. What is the positive definite matrix?

In linear algebra, a hermitian matrix ($A^* = A$) in complex domain or a symmetric matrix ($A^T = A$) is said to be positive definite if for any nonzero vectors $\mathbf{v} \in \mathbb{C}$ or $\mathbf{v} \in \mathbb{R}$, we always have $\mathbf{v}^* A \mathbf{v} > 0$ or $\mathbf{v}^T A \mathbf{v} > 0$. Any nonsymmetric or non-Hermitian matrix can be symmetrize and hermitialize to test the positive definiteness. The easiest way is to test the positive definiteness is to see if all the leading principal minors of the symmetrized matrix is positve.

2. What is Hessian of a function? What is a convex function? Plot a convex function.

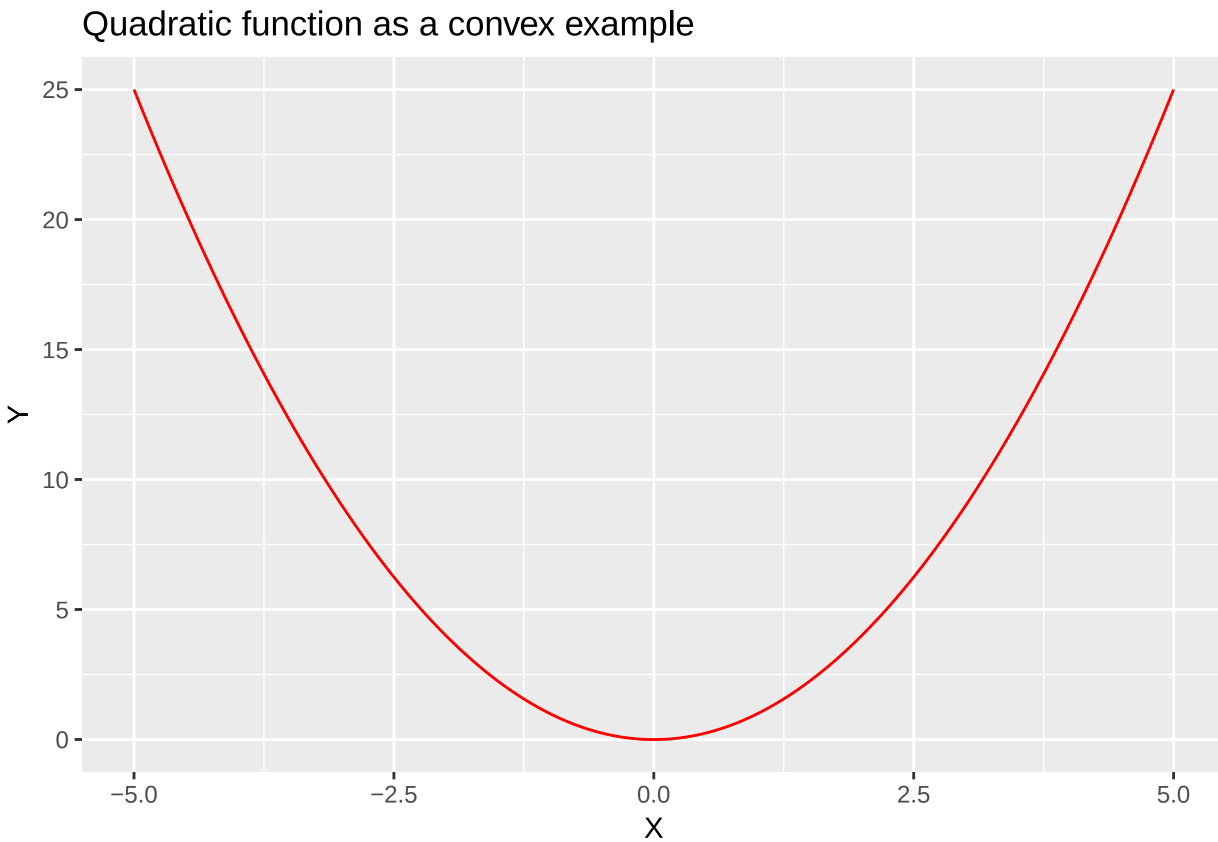
- Hessian matrix For a function $f : \mathbb{R}^n \to \mathbb{R}$, if all the second partial derivatives exist and continous over the domain, then the Hessian matrix of $f(\mathbf{x})$ is defined by

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_n} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

- Convex function Let $\mathbf{X}$ be a convex set in a real vector space and let $f : \mathbf{X} \to \mathbb{R}$ be a function. f is called convex if $\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathbf{X}, \forall t \in [0,1]$, we have $f(t\mathbf{x}_1 + (1-t)\mathbf{x}_2) \leq tf(\mathbf{x}_1) + (1-t)f(\mathbf{x}_2)$. Let us plot a convex function: $f(x) = x^2$.

```
quadraticFunc <- function(x){
  return(x^2)
}
X <- seq(-5,5,0.01)
```

```r
Y <- quadraticFunc(X)
ggplot(data = data.frame(x=X,y=Y), aes(X, Y)) +
  geom_line(color='red') + ggtitle(paste0('Quadratic function as a convex example'))
```



3. Write down the probability density of the multivariate Guassian distribution?

A random vector $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}$ is said to have the multivariate Guassian distribution if every linear combination of its components is normally distributed. For a non-degenerate multivariate Gaussian distribution, the pdf can be given by

$$f_{\mathbf{X}}(x_1, x_2, \ldots, x_n) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu))$$

4. What is the law of large numbers? What is the central limit theorem?

The law of large number states that the result of performing the same experiment a large number of times, the average of the resuls obtained from a large number of trials should be close to the expected value, and will tend to become closer as more trials are performed. Mathematically speaking, the law is

$$\lim_{N \to \infty} \bar{X}_{X_1, \ldots, X_N} = \lim_{N \to \infty} \frac{X_1 + \cdots + X_N}{N} = \mu_{X_i}$$

Central limit theorem (CLT) states that: Let $X_1, X_2, \ldots, X_n$ be a random sample from a distribution with mean μ and variance σ^2. Then if n is sufficiently large, $\bar{X}$ has approximately a normal distribution with $\mu_{\bar{X}} = \mu$ and $\sigma^2_{\bar{X}} = \sigma^2/n$, and $T_0 = X_1 + X_2 + \cdots + X_n$ also has approximately a normal distribution with $\mu_{T_0} = n\mu$ and $\sigma^2_{T_0} = n\sigma^2$. The larger the value of n, the better the approximation.