Final_Exam

Maxime Grossman

6.21.2021

Problem 1: readr

Read in the dataframe **results.csv** using the relevant function from the **readr** package. Identify the dimension of this tibble and display all of the variable names. Note that **results.csv** includes the number of goals scored in numerous soccer games based on whether or not the team is playing at their home field.

```
library(readr)

setwd("C:/Users/Maxime/Desktop/Statistical Computing & Intro to Data Science/FINAL")
results <- read_csv("results.csv", col_names = TRUE)
```

```
## Parsed with column specification:
## cols(
##   date = col_date(format = ""),
##   home_team = col_character(),
##   away_team = col_character(),
##   home_score = col_double(),
##   away_score = col_double(),
##   tournament = col_character(),
##   city = col_character(),
##   country = col_character(),
##   neutral = col_logical()
## )
```

```
head(results)
```

```
## # A tibble: 6 x 9
##   date       home_team away_team home_score away_score tournament city  country
##   <date>     <chr>     <chr>           <dbl>      <dbl> <chr>      <chr> <chr>
## 1 1872-11-30 Scotland  England             0          0 Friendly   Glas~ Scotla~
## 2 1873-03-08 England   Scotland            4          2 Friendly   Lond~ England
## 3 1874-03-07 Scotland  England             2          1 Friendly   Glas~ Scotla~
## 4 1875-03-06 England   Scotland            2          2 Friendly   Lond~ England
## 5 1876-03-04 Scotland  England             3          0 Friendly   Glas~ Scotla~
## 6 1876-03-25 Scotland  Wales               4          0 Friendly   Glas~ Scotla~
## # ... with 1 more variable: neutral <lgl>
```

```
dim(results)
```

```
## [1] 42105     9
```

```
ls(results)
```

```
## [1] "away_score" "away_team"  "city"       "country"    "date"
## [6] "home_score" "home_team"  "neutral"    "tournament"
```

Problem 2: dplyr and magrittr

Using functions from the **dplyr** package, build a **magrittr** pipe that removes the years 2020, 2021 and also removes all of the rows that have ties, i.e., remove cases where the home score equals the away score. Also create a new variable called **prop_home_win** that computes the proportion:

$$prop\_home\_win = \frac{home\_score}{home\_score + away\_score}$$

The **prop_home_win** variable should be constructed after the years and ties are removed. Report the dimension of the new tible.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(magrittr)

newresults <- results %>%
  filter(home_score != away_score)%>%
  subset(date<"2020-01-01")

tail(newresults)
```

```
## # A tibble: 6 x 9
##   date       home_team away_team home_score away_score tournament city  country
##   <date>     <chr>     <chr>          <dbl>      <dbl> <chr>      <chr> <chr>
## 1 2019-12-10 China PR  Japan              1          2 EAFF Cham~ Busan South ~
## 2 2019-12-11 South Ko~ Hong Kong          2          0 EAFF Cham~ Busan South ~
## 3 2019-12-14 Japan     Hong Kong          5          0 EAFF Cham~ Busan South ~
## 4 2019-12-15 South Ko~ China PR           1          0 EAFF Cham~ Busan South ~
## 5 2019-12-18 Hong Kong China PR           0          2 EAFF Cham~ Busan South ~
## 6 2019-12-18 South Ko~ Japan              1          0 EAFF Cham~ Busan South ~
## # ... with 1 more variable: neutral <lgl>
```

```
newresults2 <- newresults %>%
  mutate(prop_home_win = home_score/(home_score + away_score))

dim(newresults2)
```

```
## [1] 32028    10
```

Problem 3: Basic data science task

Using functions from the **tidyverse** package or **base R**, compute the average of **prop_home_win** split by country and plot a histogram of your resulting vector. To solve this task, use the same filtered data from problem 2. Based on this plot, do you believe that on average, countries tend to have a home-field advantage? Also display the 5 countries with the highest average **prop_home_win**.

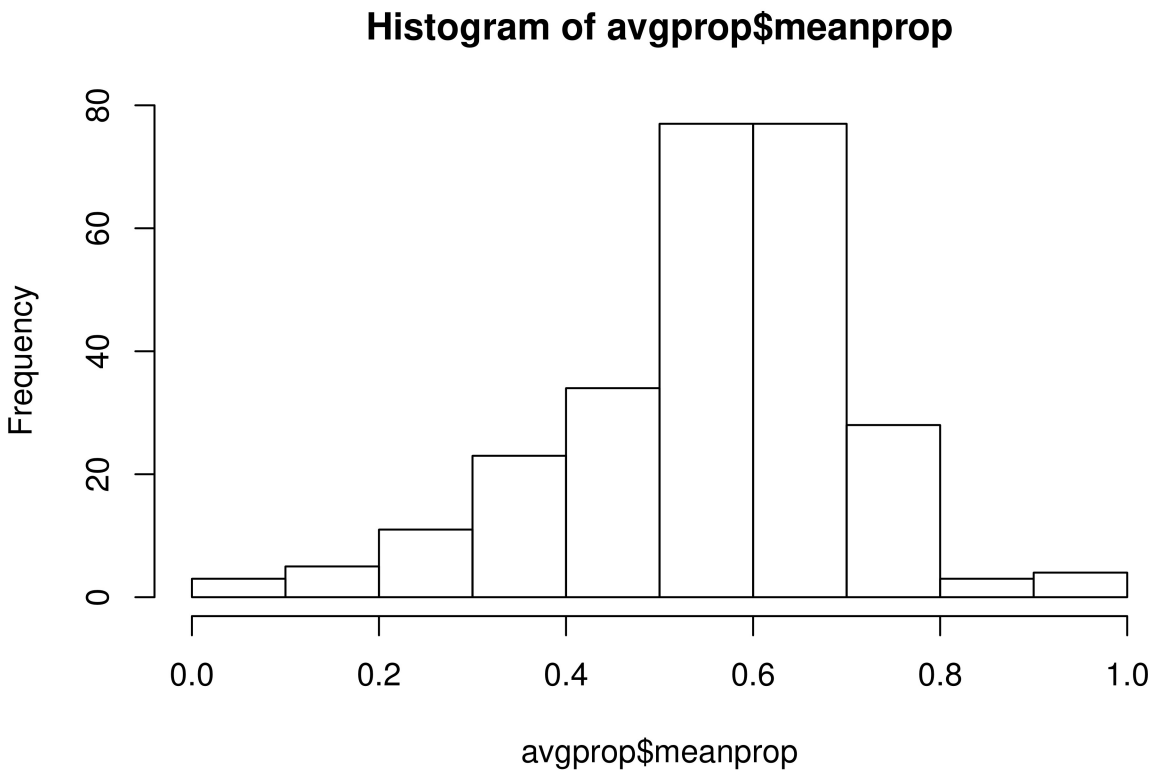
```r
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  2.1.3     v stringr 1.4.0
## v tidyr   1.0.2     v forcats 0.5.0
## Warning: package 'ggplot2' was built under R version 3.6.3
## Warning: package 'forcats' was built under R version 3.6.3
## -- Conflicts ------------------------------------------------------------------------------------
## x tidyr::extract()   masks magrittr::extract()
## x dplyr::filter()    masks stats::filter()
## x dplyr::lag()       masks stats::lag()
## x purrr::set_names() masks magrittr::set_names()
```

```r
avgprop <- newresults2 %>%
  group_by(country)%>%
  summarise(meanprop = mean(prop_home_win))

head(avgprop)
```

```
## # A tibble: 6 x 2
##   country     meanprop
##   <chr>          <dbl>
## 1 Afghanistan    0.375
## 2 Albania        0.472
## 3 Algeria        0.756
## 4 Andorra        0.147
## 5 Angola         0.716
## 6 Anguilla       0.194
```

```r
hist(avgprop$meanprop)
```




```
## The results seem to be concentrated around 0.5 to 0.7
## There is a slight tendency to be over 50%, so yes, home field advantage


# Top 5 meanprop:
tail(avgprop[order(avgprop$meanprop),],5)
```

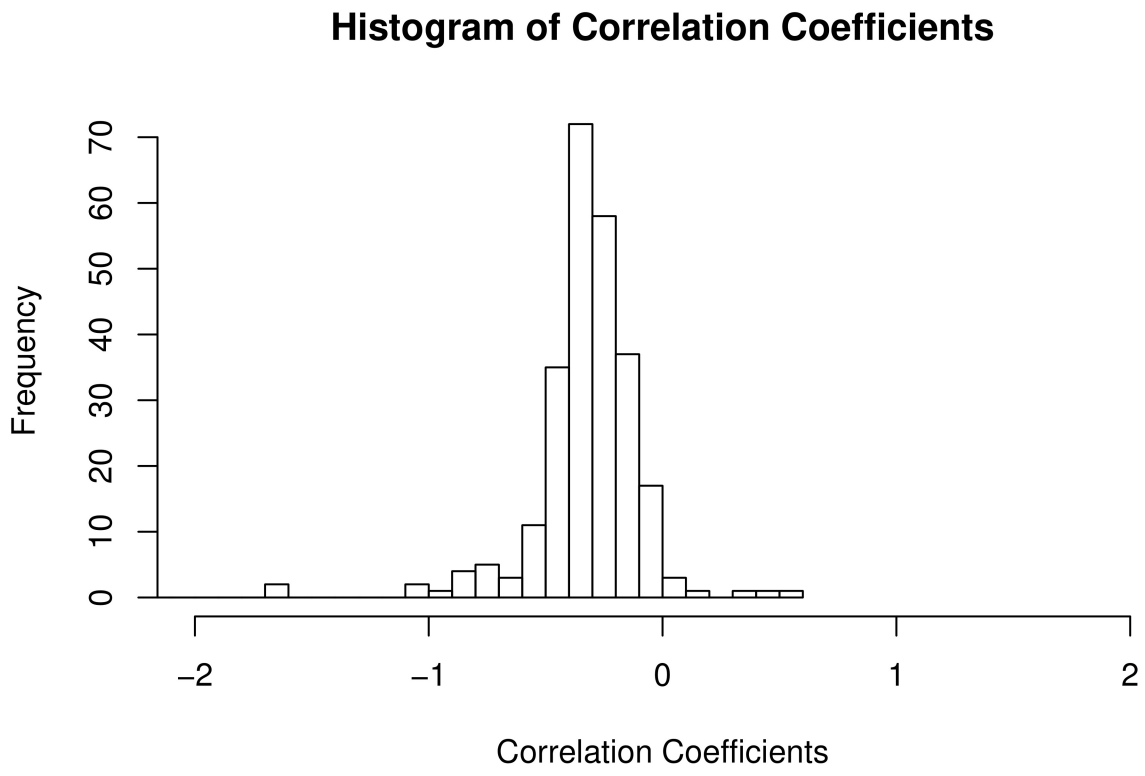
```
## # A tibble: 5 x 2
##   country           meanprop
##   <chr>                <dbl>
## 1 Nigeria              0.813
## 2 Gold Coast           0.929
## 3 Lautoka              1
## 4 Micronesia           1
## 5 Portuguese Guinea    1
```

Problem 4: Another basic data science task

Using functions from the **tidyverse** package or **base R**, compute the correlation coefficient of **home_score** (Y) versus **away_score** (X), split by country. To solve this task, use the same filtered data from problem 2 and problem 3. Plot a histogram of your correlation coefficients and discuss if the results look consistent with problem 3.

```
lm4 <- newresults2 %>%
  split(.$country) %>%
  map(~lm(home_score ~ away_score, data=.)) %>%
  map(coef)
```

```r
lm4 <- unlist(lm4)

emp <- NULL

for(i in 1:265){
  emp[i] <- lm4[2*i]
}

hist(emp, main="Histogram of Correlation Coefficients", breaks=60, xlim=c(-2,2), xlab="Correlation Coefi
```




```r
## It seems the concentration of correlation coefficients are centered around -0.5 to -0.7
## This is consistent with our results; slightly over 50%
```

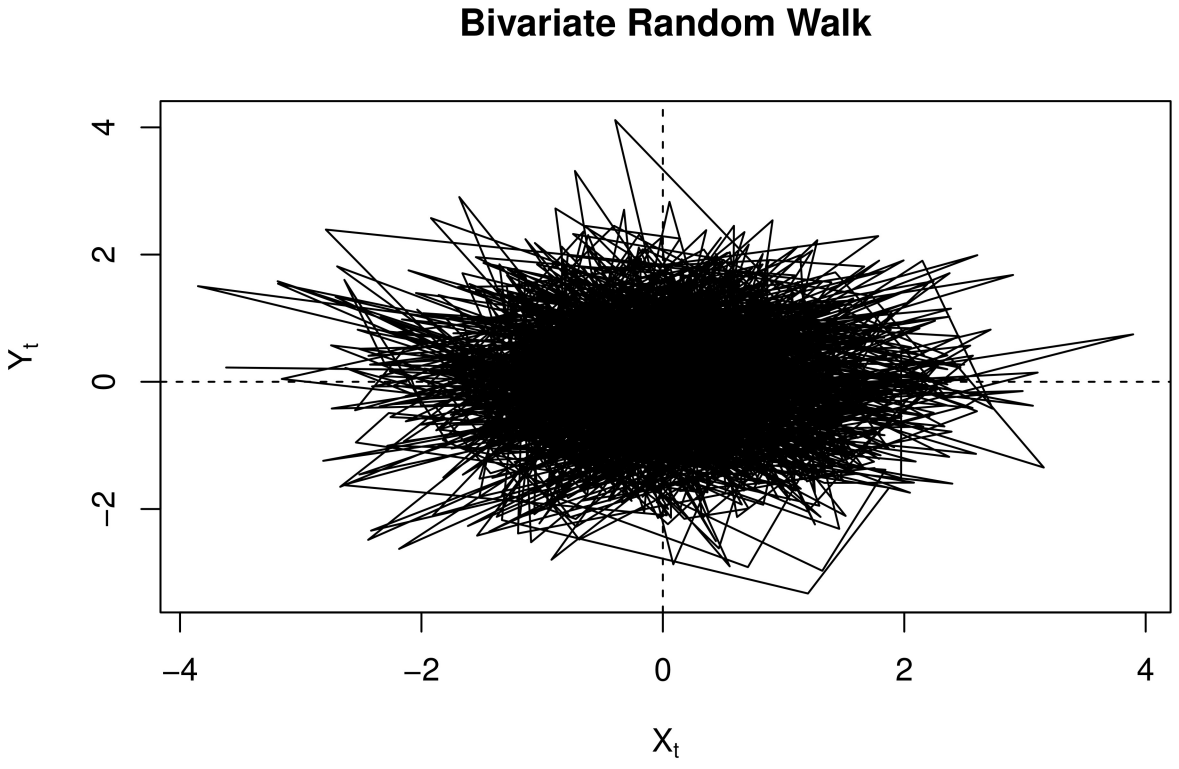
Problem 5: Simulating a bivariate random walk

Define a function in **R** called **biv_rw()** that computes a realization of the bivariate random walk up to time $t = n$. Your function should return a matrix or dataframe of dimension $(n \times 2)$. Graph a single realization of your random walk with $n = 2000$. Below shows most of the plotting code.

Note that your final random walk will look like a particle, randomly wandering through 2-dimensional space.

```r
biv_rw <- function(n){

sims <- data.frame(matrix(NA, nrow=n, ncol=2))    #create empty df
colnames(sims) <-  c("X","Y")
```

```r
x  <- 0                # x_0 = 0
y <- 0                 # y_0 = 0

sims[1,1] <- x
sims[1,2] <- y

for(i in 2:n){

    sims[i,1] <- x + rnorm(1, mean = 0, sd=1)
    sims[i,2] <- y + rnorm(1, mean = 0, sd = 1)

  }
return(sims)
}


my_sim <- biv_rw(2000)


plot(my_sim[,1],my_sim[,2],type="l",xlab=expression(X[t]),ylab=expression(Y[t]), main="Bivariate Random
abline(h=0,lty=2)
abline(v=0,lty=2)
```



Problem 6: Simulating a stopping time

Define a new random walk function named **biv_rw_new** that computes the total number of iterations your simulated random walk takes to initially escape a circle of radius $r > 0$. In this case, you should only have the single argument **r** (radius) because you do not know how many iterations your random walk will take until it exceeds r. Your function should output the final random walk sequence and should also output the number of iterations required to escape the circle. Plot two realizations of your function, showing both the realized random walk and a circle of radius $r = 50$. Place the number of iterations in the title of your plot. Again, some plotting code is provided below for efficiency.

```
biv_rw_new <- function(radius){

sims <- (data.frame(matrix(nrow=(1000000), ncol=2))) #create empty df

colnames(sims) <-  c("X","Y")

n <- 1

x  <- 0              # x_0 = 0
y <- 0               # y_0 = 0

sims[1,1] <- x
sims[1,2] <- y

while(sqrt((sims[n,1])^2 + (sims[n,2])^2) < radius){
    n <- n + 1
    sims[n,1] <- x + rnorm(1, mean = 0, sd=1)
    sims[n,2] <- y + rnorm(1, mean = 0, sd = 1)
    }
return(n)
  }


## Note: I am using radius = 3 because radius = 50 takes too long to run.
## Please consider my code and not the fact that I did not input 50.


biv_rw_new(3)
```

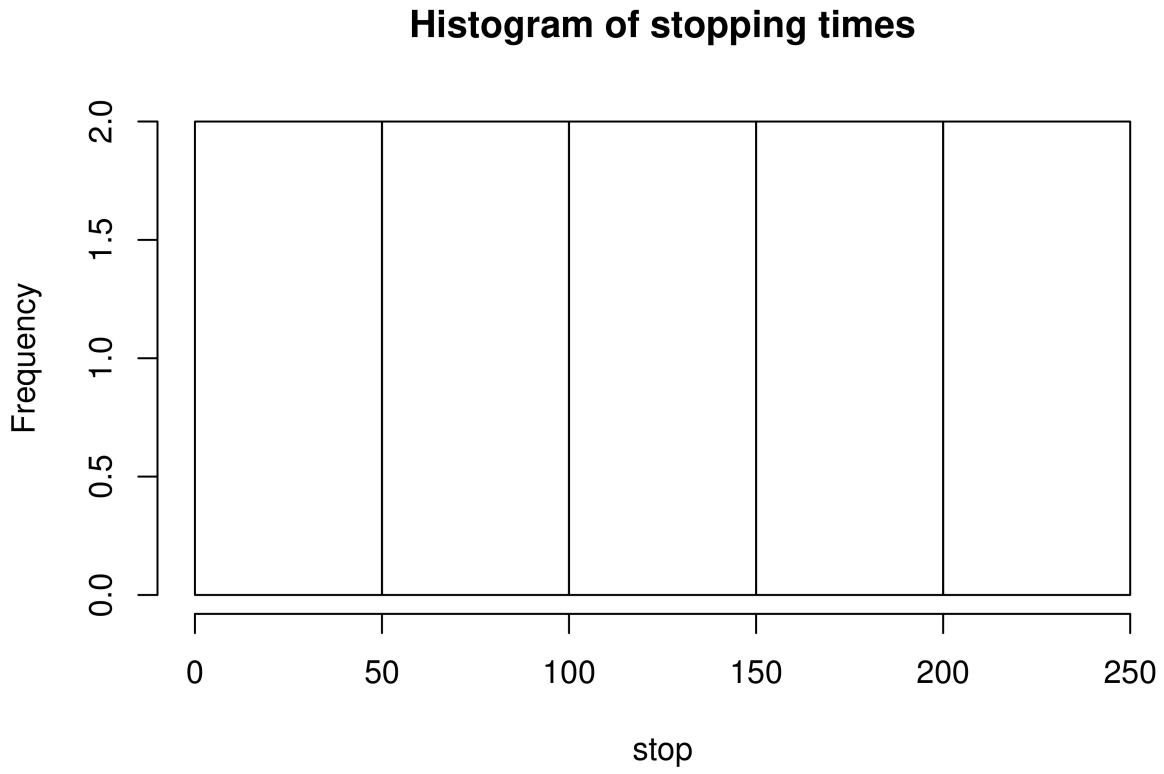
```
## [1] 26
```

Problem 7: Simulating the distribution of the stopping time

Simulate 10,000 stopping times using $r = 50$, i.e., run the function **biv_rw_new()** 10,000 times and extract the number of iterations required to escape the circle for each simulated random walk. Note that your simulation might take at least 2 minutes to run. After running the simulation, plot a histogram and compute the mean of your simulated stopping times. Based on your results, do you believe that a theoretical average stopping time exists for this scenario?

```
## Note: I am using radius = 3 because radius = 50 runs indefinitely.
## I am also running it 10 times because otherwise it takes a really long time.


stopping <- function(k){
score <- NULL
```

```
for(i in 1:k){
  score[i] <- biv_rw_new(3)

}
return(score)
}


stop <- stopping(10)

hist(stop, main="Histogram of stopping times")
```




```
mean(stop)
```

```
## [1] 115.1
```

```
## The mean running iterations for radius of 3 is about 50-100 runs.
## The average stopping time exists for r = 3 but it is very volatile each time.
```

Problem 8: Approximating the distribution of the stopping time

Approximate the distribution of stopping times with a gamma. Note that we are using a continuous distribution to approximate a discrete random variable but let's assume this is legitimate for this exercise. To accomplish this task, use method of moments to estimate the shape and scale parameters of the gamma distribution. Display the estimated shape and scale parameters.

Hint: You can use the **gamma.MMest()** from lecture 10 directly:

```r
gamma.MMest <- function(v){

  m <- mean(v)
  v <- var(v)
  a <-  (m^2)/v
  s = (v)/m
  return(c(a,s))
  }


gamma.MMest(stop)
```

```
## [1]  1.983931 58.016121
```

Problem 9: Computing some probabilities

Based on your estimated gamma distribution, compute the probability that a bivariate random walk takes more than 1,000 iterations to exit a circle of radius $r = 50$. Compare this answer to the empirical proportion, i.e., **mean(stopping_times>5000)**.

Similarly, compute the probability that a bivariate random walk takes more than 5,000 and 10,000 iterations to exit a circle or radius $r = 50$. Compare these answers to the empirical proportions.

Problem 10: Monte Carlo integration

Use Monte Carlo integration to approximate the normalizing constant $d > 0$. Note for comparison, the true normalizing constant is $6/7$.

```r
my.kernel <- function(x,y) {
  ifelse(x > 1 | x < 0 | y < 0 | y > 2, 0, x^2 + (x*y)/2)    # let x2 = y
}

mc.sim1 <-runif(100000,0,1)

mc.sim2 <-runif(100000,0,2)

mean(my.kernel(mc.sim1, mc.sim2))
```

```
## [1] 0.5839647
```

Part 11.i

Write a function called **my_sim_fun** that simulates **n.samps** draws from $f(x_1, x_2)$. Your function should use the accept-reject algorithm to simulate draws from f. To test your function, simulate 5 draws from f and show all cases. Your final result will be a matrix of dimension 5×2 or two vectors, each of length 5.

Some notes: Before solving this problem, you should redefine f so that $d = 6/7$, i.e., f should be a valid pdf. Also note that the proposal density can be easily simulated as two independent uniform random variables.

```r
my_sim_fun <- function(x){
  return( ifelse(x > 1 | x < 0 | y < 0 | y > 2, 0, 6/7*(x^2 + (x*y)/2)))

}

# We can see by inspection that the function max is when x1 = 1 and x2 = 2
```

```r
f.max <- (6/7)*(2^2 + 1)



#x <- seq(0, 1, length = 100)
#plot(x, f(x), type = "l", ylab = "f(x)")


e <- function(x){
  return(ifelse((x > 1 | x < 0 | y < 0 | y > 2), Inf, f.max))   ### f.max is the x plugged into f(x) wh
}


my_target <- function(n.samps=5) {
  n.samps <- 5
  n       <- 0
  samps   <- numeric(n.samps)
  samps2  <- numeric(n.samps)
  while (n < n.samps) {
    y <- runif(1,min=0,max=2)
    u <- runif(1)
    if (u < my_sim_fun(y)/e(y)) {
      n        <- n + 1
      samps[n] <- y
      samps2[n] <- u
      }

  }
  return(samps)
  return(samps2)
}



#x <- seq(0, 1, length = 100)
#hist(samps, prob = T, ylab = "f(x)", xlab = "x",
#     main = "Histogram of draws from Beta(4,3)")
#lines(x, dbeta(x, 4, 3), lty = 2)
```

Part 11.ii

Simulate 10,000 draws from $f(x_1, x_2)$ using your function **my_sim_fun()**. Use a heat map to visualize your sampled distribution. For reference, please see the below code showing how to plot a simple bivariate distribution using a heatmap. The below example shows two independent standard normals represented as a bivariate random vector $Z = (Z_1, Z_2)$.

```r
# my_sim_fun(10000)
```

Problem 12: MLE of a normal distribution

Suppose that $X_1, X_2, \ldots X_{200}$ are drawn from a normal population with both μ and σ^2 unknown. The dataset of interest is **normal_mle.csv**. Define two functions in R: 1) the likelihood and 2) the negative log-likelihood. Test both functions at the point $(\mu, \sigma^2) = (1, 1)$. Note that the likelihood will yield a very small positive number.

```r
# Likelihood

data1 <- read.csv("normal_mle.csv", header = TRUE)

data_normal <- data1$x

norm.ll <- function(mean, sd, data){
  return(sum(dnorm(data, mean, sd, log=FALSE)))
}

norm.ll(1,1,data_normal)
```

```
## [1] 43.21385
```

```r
# Log likelihood

neg.norm.ll <- function(mean, sd, data){
  return(-1*sum(dnorm(data, mean, sd, log=TRUE)))
}

neg.norm.ll(1,1,data_normal)
```

```
## [1] 380.0774
```

Problem 13: MLE and nlm()

Use the **nlm()** function to compute the maximum likelihood estimates of μ and σ^2. Also check your answer with the analytic formula $\hat{\mu} = \bar{x}$ and $\hat{\sigma}^2 = \sum_{i=1}^{n}(x_i - \bar{x})^2/n$.

```r
nlm(neg.norm.ll, 1,1, data=data_normal)
```

```
## $minimum
## [1] 353.4331
##
## $estimate
## [1] 0.4838192
##
## $gradient
## [1] 0.0001025455
##
## $code
## [1] 1
##
## $iterations
## [1] 1
```

Problem 14: MCMC

For this exercise, we consider the same dataset as problem 12 and problem 13. Based on numerous pilot studies, suppose that the true mean μ is always between $[0, 1]$ and the true variance is between $[0, 2]$. Further, suppose that the distribution $f(\cdot, \cdot)$ from problem 10 is a suitable prior for (μ, σ^2).

Run the Metropolis-Hastings algorithm to estimate the joint posterior of(μ, σ^2). Some notes follow:

- Run 20000 iterations. I.e., simulate 20000 draws of $\theta_t = (\mu_t, \sigma_t^2)$.

- Proposal distribution $f(\mu, \sigma^2)$, same as the prior.
- Independence chain with Metropolis-Hastings ratio:

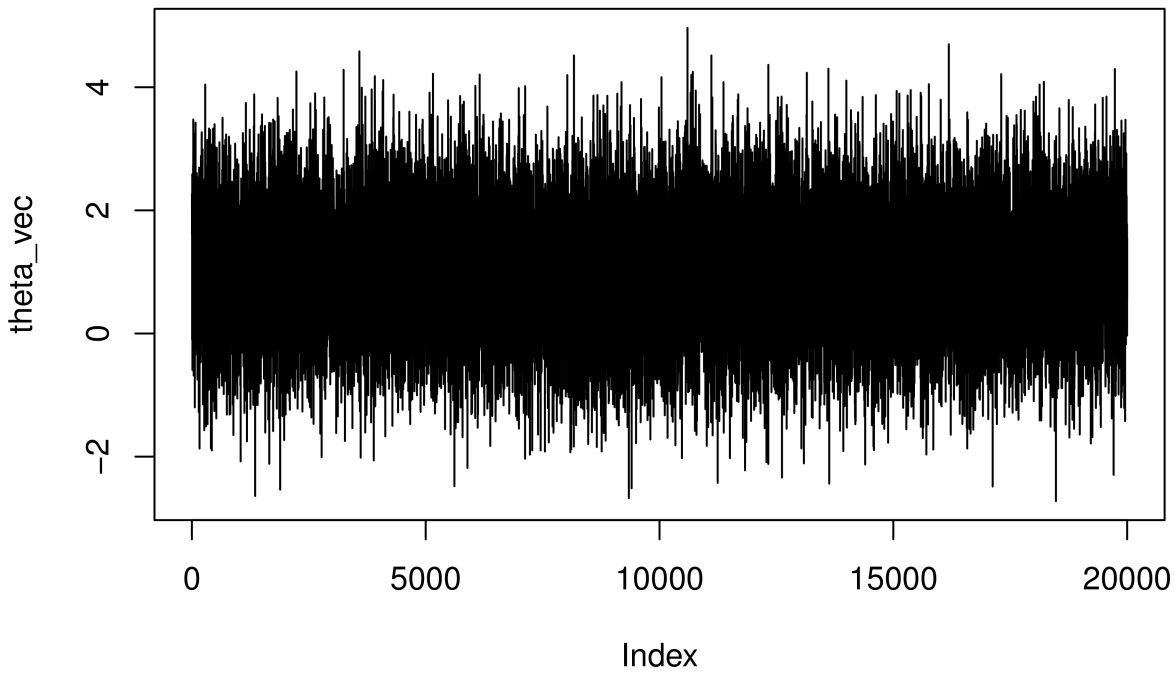
$$R(\theta_t, \theta^*) = \frac{L(\theta^*|x_1, \ldots, x_{200})}{L(\theta_t|x_1, \ldots, x_{200})}$$

Display traceplots of the simulated marginal posteriors μ_t and σ_t^2. Also plot a heatmap for the full posterior of (μ, σ^2).

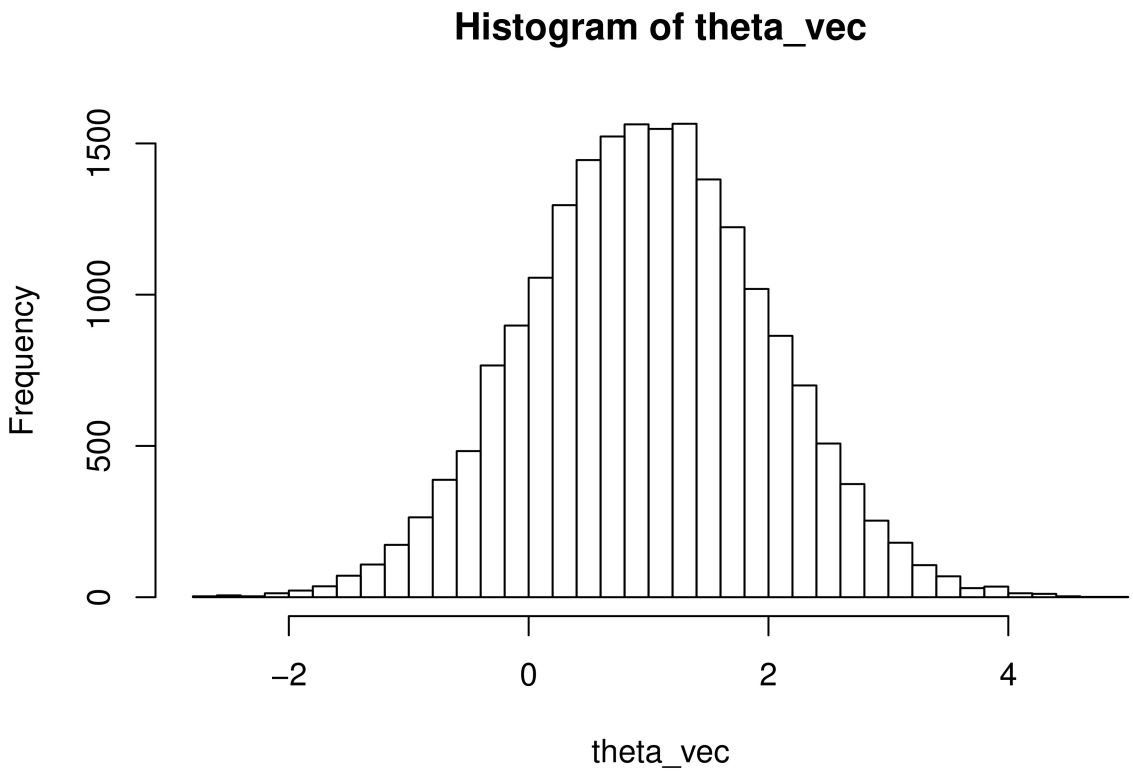
```
n<-50
X<-runif(n)
mean(X)
```

```
## [1] 0.4322347
```

```
# Draw case for t=0 from proposal

theta_1 <- rnorm(1,1,1)
n.samps <- 20000
theta_vec <- rep(NA,(n.samps+1))
theta_vec[1] <- theta_1

# MCMC loop
for (t in 1:n.samps) {
  theta_star <- rnorm(1,1,1)
  theta_t <- theta_vec[t]
  MH_ratio <- prod(dunif(X))/prod(dunif(X))

  prob_vec <- c(min(MH_ratio,1),1-min(MH_ratio,1))
  theta_vec[t+1] <- sample(c(theta_star,theta_t),1,prob = prob_vec)
}

plot(theta_vec,type="l")
```




```
hist(theta_vec,breaks=30)
```


Histogram of theta_vec
Frequency
0
500
1000
1500
−2
0
2
4
theta_vec