STA6703 SML Take-Home Prelim, Fall 2022

Christopher Marais

Helping functions

```
myRound <- function(x, acc=3) {mult = 10^acc; round(x*mult)/mult}
```

Problem 1

Case 1

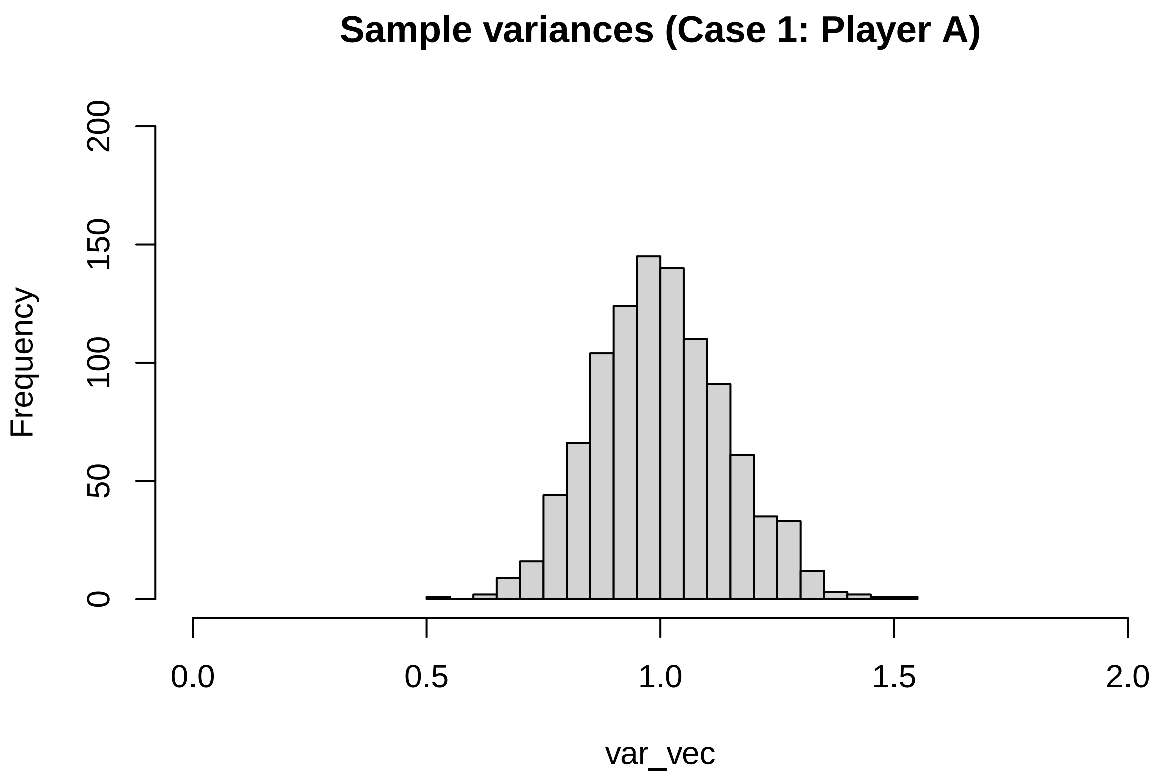
```
# Case 1
n = 100
m=1000
set.seed(0)
origData = rnorm(n) # case 1


# a) Player A
# generate 1000 data sets
set.seed(0)
z = rnorm(n*m)
mat = matrix(z,nrow=m)
#calculate variance for each row
var_vec = apply(mat,1,var)
samp_mean = mean(var_vec)
samp_var = var(var_vec)
print(paste("The sample mean is (Case 1: Player A): ", myRound(samp_mean)))
```

```
## [1] "The sample mean is (Case 1: Player A):  1.002"
```

```
print(paste("The sample variance is (Case 1: Player A): ", myRound(samp_var)))
```

```
## [1] "The sample variance is (Case 1: Player A):  0.02"
```

```
hist(var_vec,
     ylim=c(0,200),
     breaks=20,
     xlim=c(0,2),
     main="Sample variances (Case 1: Player A)")
```



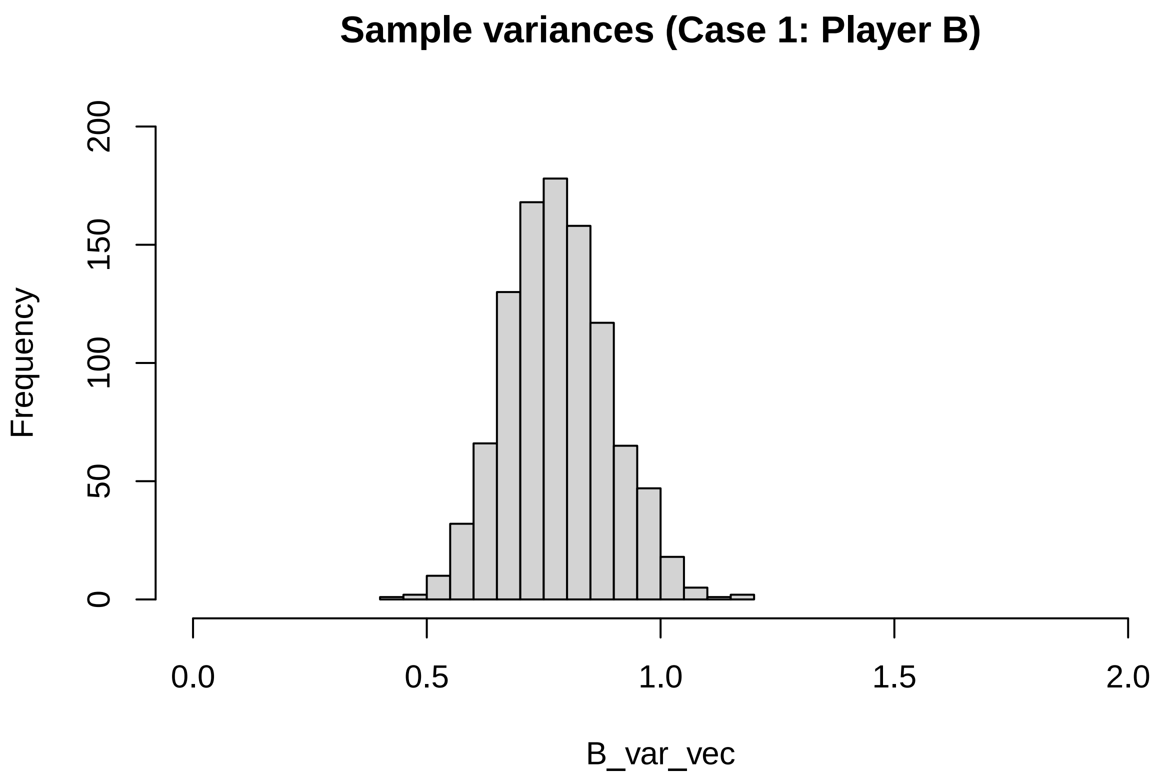

```
# b) Player B
# Estimate mean and variance from the sample data
B_var = var(origData)
B_mean = mean(origData)
# simulate new data with
set.seed(0)
B_mat = matrix(rnorm(n*m, mean = B_mean, sd = sqrt(B_var)),nrow=m)
B_var_vec = apply(B_mat,1,var)
B_samp_mean = mean(B_var_vec)
B_samp_var = var(B_var_vec)
print(paste("The sample mean is (Case 1: Player B): ", myRound(B_samp_mean)))
```

```
## [1] "The sample mean is (Case 1: Player B):  0.781"
```

```
print(paste("The sample variance is (Case 1: Player B): ", myRound(B_samp_var)))
```

```
## [1] "The sample variance is (Case 1: Player B):  0.012"
```

```
hist(B_var_vec,
     ylim=c(0,200),
     breaks=20,
     xlim=c(0,2),
     main="Sample variances (Case 1: Player B)")
```



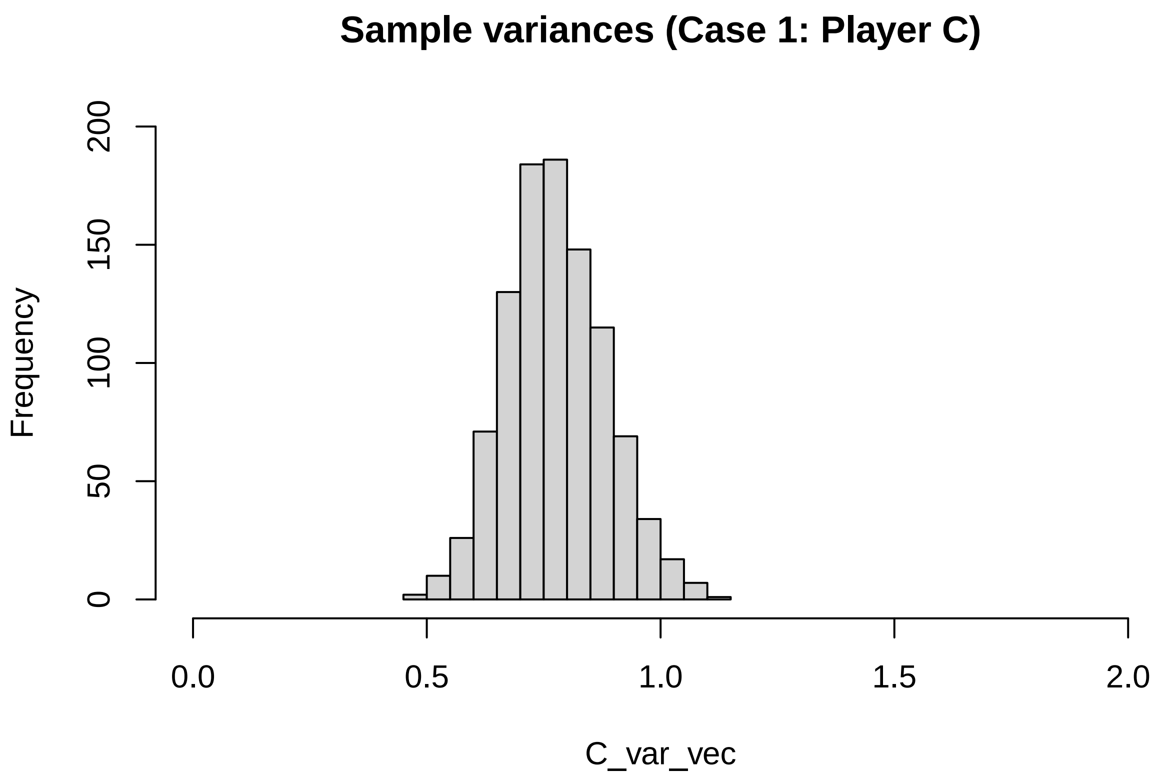

```r
# c) Player C
# sample new data
z=sample(x=origData, size=n*m, replace=TRUE)
C_mat = matrix(z,nrow=m)
C_var_vec = apply(C_mat,1,var)
C_samp_mean = mean(C_var_vec)
C_samp_var = var(C_var_vec)
print(paste("The sample mean is (Case 1: Player C): ", myRound(C_samp_mean)))
```

```
## [1] "The sample mean is (Case 1: Player C):  0.776"
```

```r
print(paste("The sample variance is (Case 1: Player C): ", myRound(C_samp_var)))
```

```
## [1] "The sample variance is (Case 1: Player C):  0.011"
```

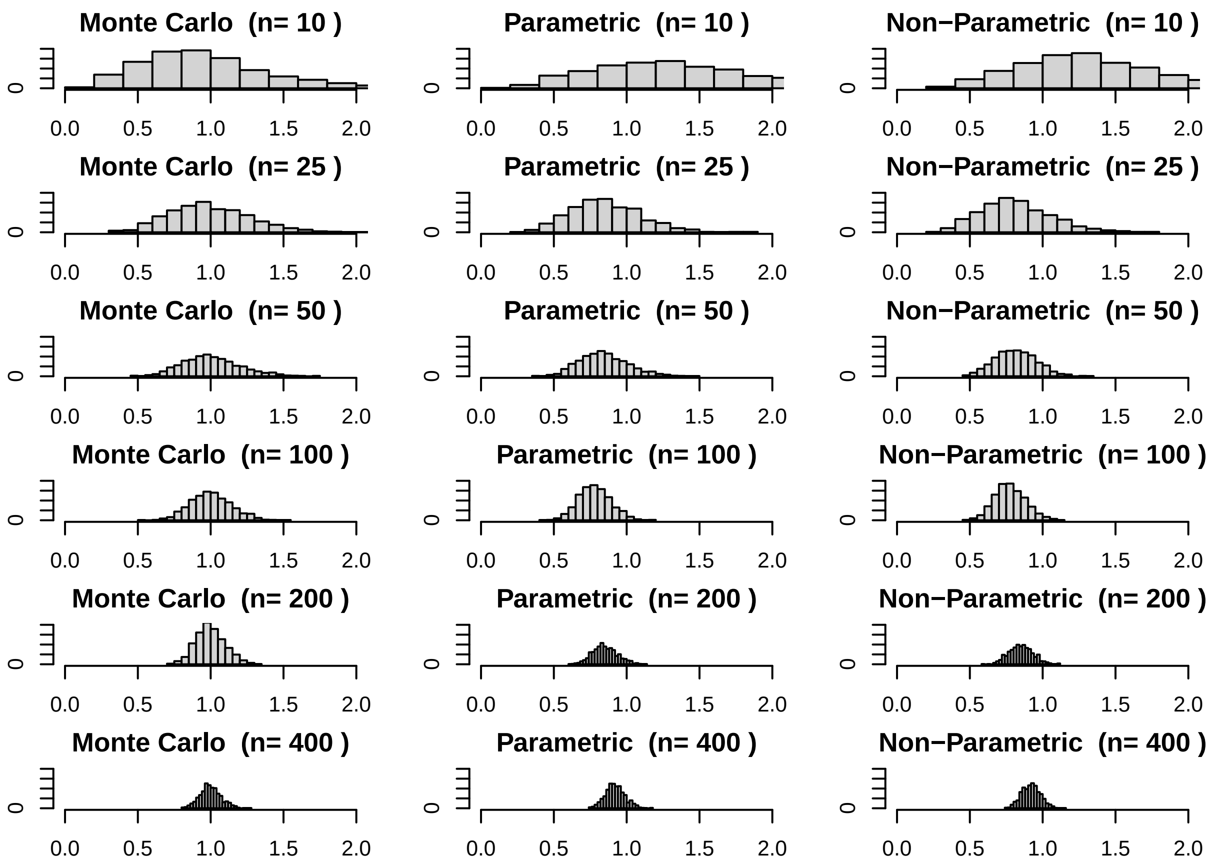
```r
hist(C_var_vec,
     ylim=c(0,200),
     breaks=20,
     xlim=c(0,2),
     main="Sample variances (Case 1: Player C)")
```




```r
# e)
n_vec=c(10, 25, 50, 100, 200, 400)
methods_vec = c("Monte Carlo", "Parametric", "Non-Parametric")
m=1000
set.seed(0)
data400 = rnorm(400)

# save results in tables
mean_res_df = data.frame(matrix(ncol=7, nrow=3))
var_res_df = data.frame(matrix(ncol=7, nrow=3))
colnames(mean_res_df) = c("Method", n_vec)
colnames(var_res_df) = c("Method", n_vec)
mean_res_df["Method"] = methods_vec
var_res_df["Method"] = methods_vec

par(mfrow=c(6,3),mar=c(2,2,2,2))
for(n in n_vec){
  mean_res_vec = c()
  var_res_vec = c()

  #Monte Carlo
  set.seed(0)
  z = rnorm(n*m)
  mat = matrix(z,nrow=m)
  var_vec = apply(mat,1,var)
```

```r
samp_mean = myRound(mean(var_vec))
samp_var = myRound(var(var_vec))
mean_res_vec = c(mean_res_vec, samp_mean)
var_res_vec = c(var_res_vec, samp_var)
hist(var_vec,
     ylim=c(0,200),
     breaks=20,
     xlim=c(0,2),
     main=paste("Monte Carlo ",
                "(n=",n,")"))

# Parametric
# Estimate mean and variance from the sample data
z = data400[1:n]
B_var = var(z)
B_mean = mean(z)
# simulate new data with
set.seed(0)
B_mat = matrix(rnorm(n*m, mean = B_mean, sd = sqrt(B_var)),nrow=m)
B_var_vec = apply(B_mat,1,var)
B_samp_mean = myRound(mean(B_var_vec))
B_samp_var = myRound(var(B_var_vec))
mean_res_vec = c(mean_res_vec, B_samp_mean)
var_res_vec = c(var_res_vec, B_samp_var)
hist(B_var_vec,
     ylim=c(0,200),
     breaks=20,
     xlim=c(0,2),
     main=paste("Parametric ",
                "(n=",n,")"))

# c) Player C
# sample new data
z=sample(x=data400[1:n], size=n*m, replace=TRUE)
C_mat = matrix(z,nrow=m)
C_var_vec = apply(C_mat,1,var)
C_samp_mean = myRound(mean(C_var_vec))
C_samp_var = myRound(var(C_var_vec))
mean_res_vec = c(mean_res_vec, C_samp_mean)
var_res_vec = c(var_res_vec, C_samp_var)
hist(C_var_vec,
     ylim=c(0,200),
     breaks=20,
     xlim=c(0,2),
     main=paste("Non-Parametric ",
                "(n=",n,")"))

mean_res_df[as.character(n)] = mean_res_vec
var_res_df[as.character(n)] = var_res_vec
```

```
}
```




```r
# display tables
knitr::kable(mean_res_df,
             format = "markdown",
             caption = "Mean for different n values (Case 1).")
```

Table 1: Mean for different n values (Case 1).

Method	10	25	50	100	200	400
Monte Carlo	0.987	0.995	1.005	1.002	0.998	0.998
Parametric	1.434	0.856	0.844	0.781	0.851	0.921
Non-Parametric	1.305	0.818	0.815	0.776	0.851	0.920

```r
knitr::kable(var_res_df,
             format = "markdown",
             caption = "Variance for different n values (Case 1).")
```

Table 2: Variance for different n values (Case 1).

Method	10	25	50	100	200	400
Monte Carlo	0.230	0.080	0.040	0.020	0.010	0.005
Parametric	0.486	0.059	0.028	0.012	0.007	0.004
Non-Parametric	0.225	0.061	0.020	0.011	0.007	0.004

(d) Carefully discuss your findings; specifically, accuracy (bias and variance) and merits/drawbacks of parametric vs nonparametric bootstrap. You can treat the results of the Monte Carlo-based inference (under simulation from the true distribution) as the golden standard, against which the parametric and nonparametric bootstrap results are compared.

Case 2

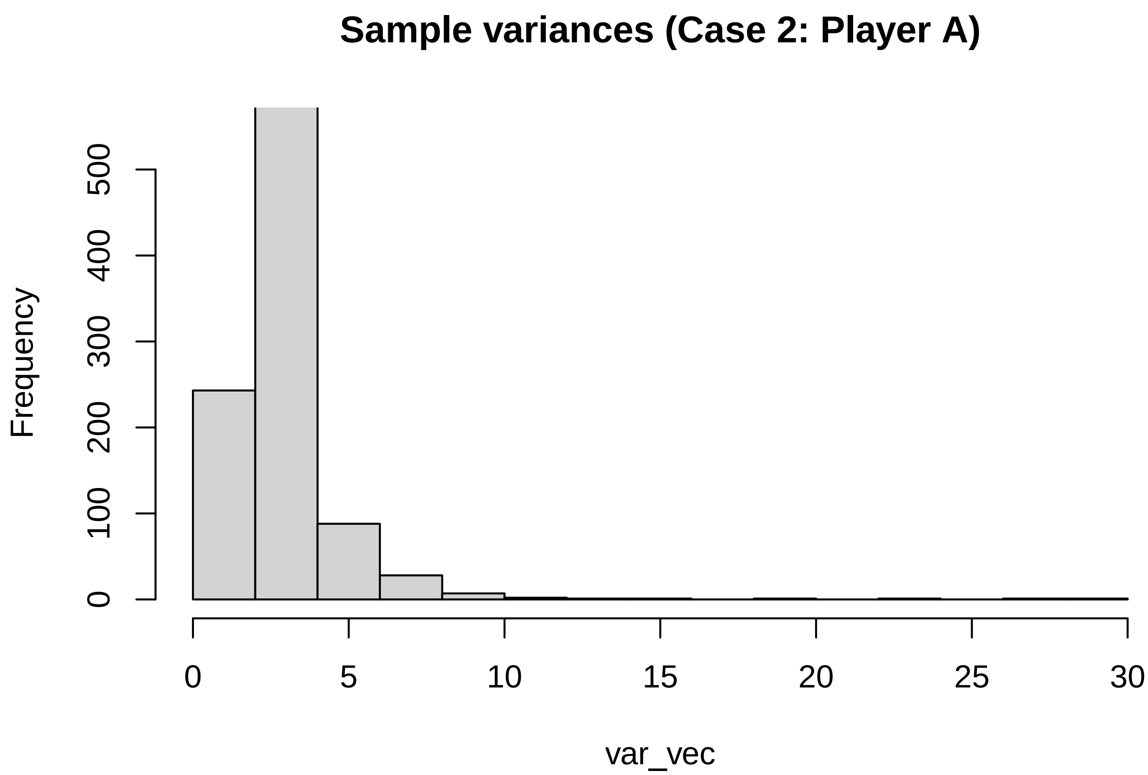
```
# Case 2
n = 100
m=1000
set.seed(0)
origData = rt(n,df=3); # case 2


# a) Player A
# generate 1000 data sets
set.seed(0)
z = rt(n*m,df=3)
mat = matrix(z,nrow=m)
#calculate variance for each row
var_vec = apply(mat,1,var)
samp_mean = mean(var_vec)
samp_var = var(var_vec)
print(paste("The sample mean is (Case 2: Player A): ", myRound(samp_mean)))
```

```
## [1] "The sample mean is (Case 2: Player A):  2.917"
```

```
print(paste("The sample variance is (Case 2: Player A): ", myRound(samp_var)))
```

```
## [1] "The sample variance is (Case 2: Player A):  3.897"
```

```
hist(var_vec,
     ylim=c(0,550),
     # breaks=20,
     xlim=c(0,30),
     main="Sample variances (Case 2: Player A)")
```



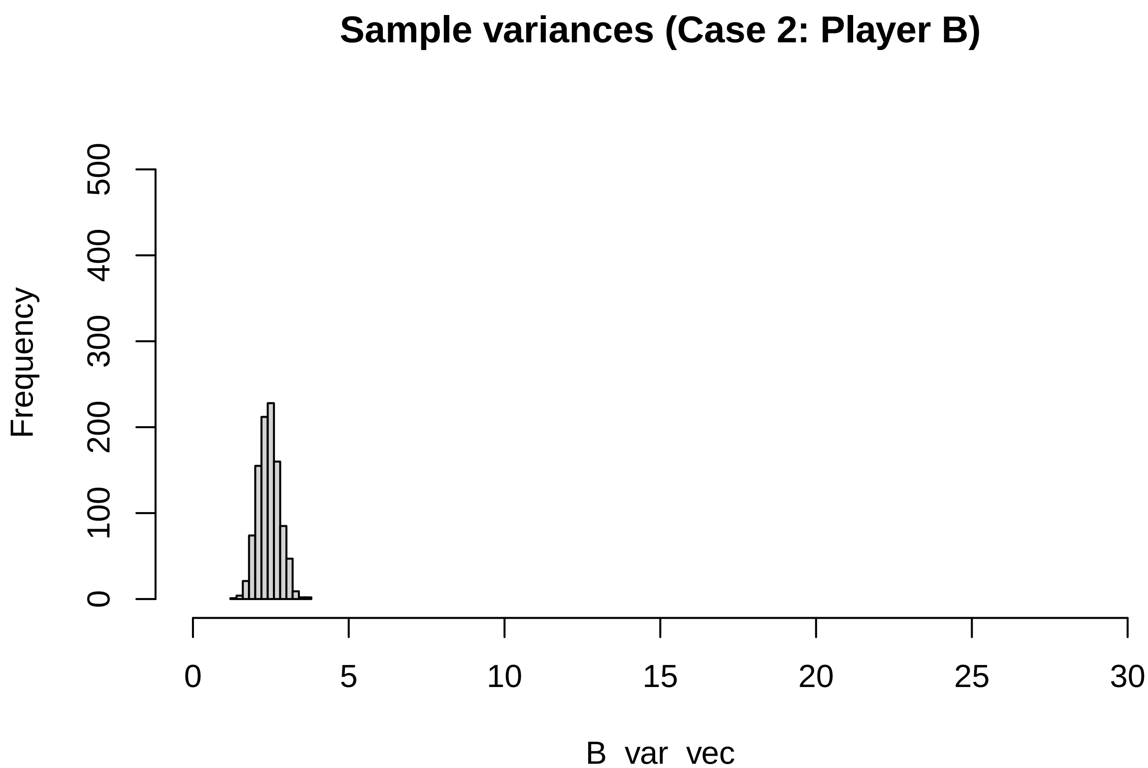

```
# b) Player B
# Estimate mean and variance from the sample data
B_var = var(origData)
B_mean = mean(origData)
# simulate new data with
set.seed(0)
B_mat = matrix(rnorm(n*m, mean = B_mean, sd = sqrt(B_var)),nrow=m)
B_var_vec = apply(B_mat,1,var)
B_samp_mean = mean(B_var_vec)
B_samp_var = var(B_var_vec)
print(paste("The sample mean is (Case 2: Player B): ", myRound(B_samp_mean)))
```

```
## [1] "The sample mean is (Case 2: Player B):  2.434"
```

```
print(paste("The sample variance is (Case 2: Player B): ", myRound(B_samp_var)))
```

```
## [1] "The sample variance is (Case 2: Player B):  0.118"
```

```
hist(B_var_vec,
     ylim=c(0,550),
     # breaks=20,
     xlim=c(0,30),
     main="Sample variances (Case 2: Player B)")
```



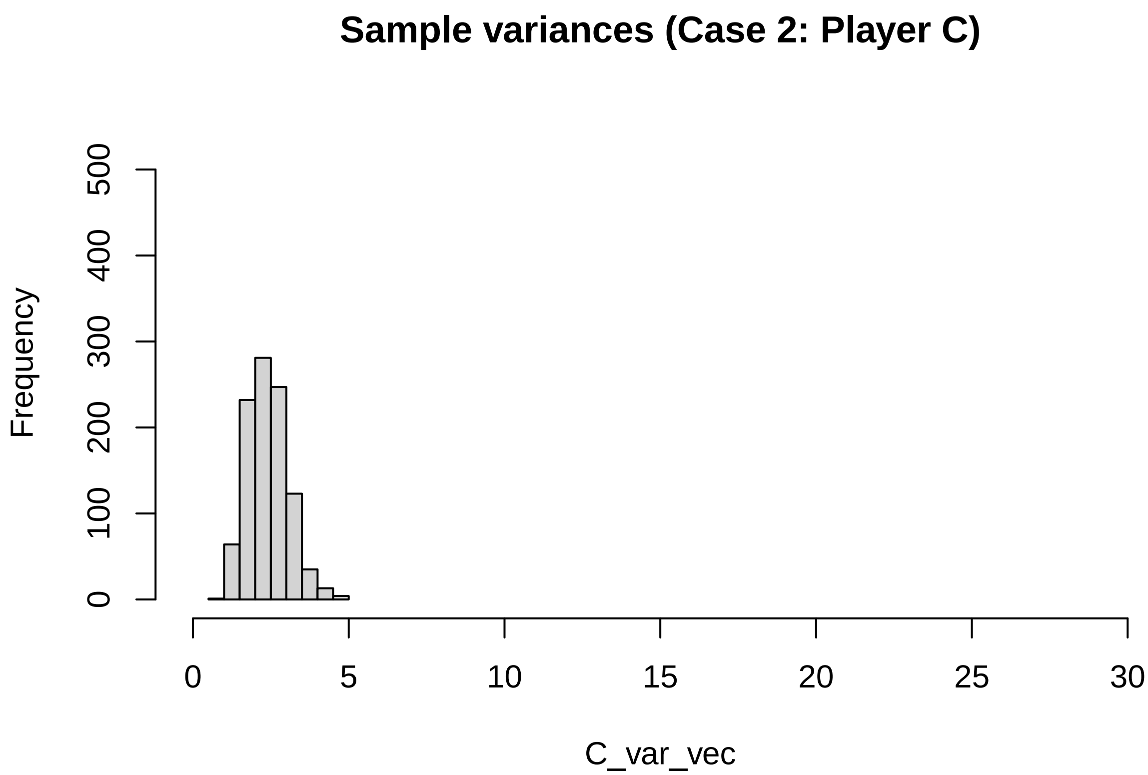

```r
# c) Player C
# sample new data
z=sample(x=origData, size=n*m, replace=TRUE)
C_mat = matrix(z,nrow=m)
C_var_vec = apply(C_mat,1,var)
C_samp_mean = mean(C_var_vec)
C_samp_var = var(C_var_vec)
print(paste("The sample mean is (Case 2: Player C): ", myRound(C_samp_mean)))
```

```
## [1] "The sample mean is (Case 2: Player C):  2.41"
```

```r
print(paste("The sample variance is (Case 2: Player C): ", myRound(C_samp_var)))
```

```
## [1] "The sample variance is (Case 2: Player C):  0.422"
```

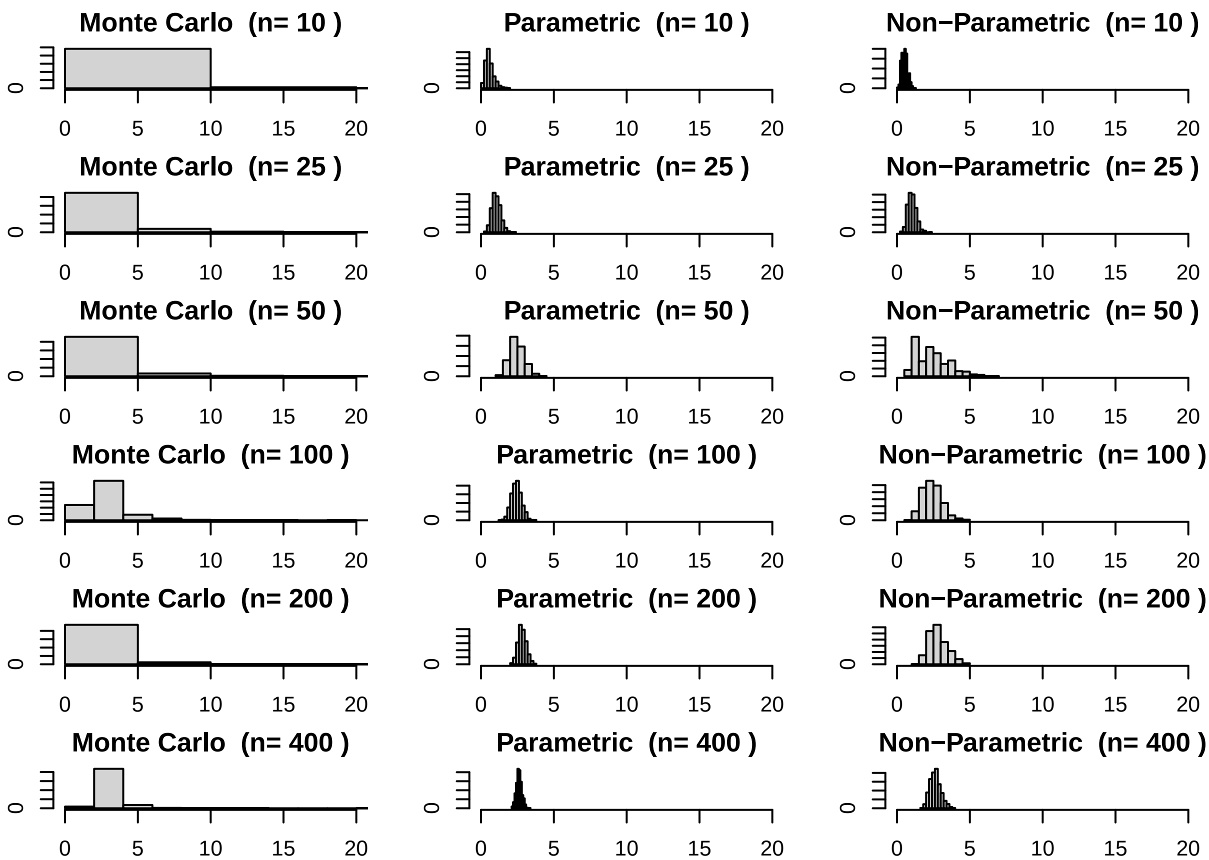
```r
hist(C_var_vec,
     ylim=c(0,550),
     # breaks=20,
     xlim=c(0,30),
     main="Sample variances (Case 2: Player C)")
```




```r
# e)
n_vec=c(10, 25, 50, 100, 200, 400)
methods_vec = c("Monte Carlo", "Parametric", "Non-Parametric")
m=1000
set.seed(0)
data400 = rt(400,df=3)

# save results in tables
mean_res_df = data.frame(matrix(ncol=7, nrow=3))
var_res_df = data.frame(matrix(ncol=7, nrow=3))
colnames(mean_res_df) = c("Method", n_vec)
colnames(var_res_df) = c("Method", n_vec)
mean_res_df["Method"] = methods_vec
var_res_df["Method"] = methods_vec

par(mfrow=c(6,3),mar=c(2,2,2,2))
for(n in n_vec){
  mean_res_vec = c()
  var_res_vec = c()

  #Monte Carlo
  set.seed(0)
  z = rt(n*m, df=3)
  mat = matrix(z,nrow=m)
  var_vec = apply(mat,1,var)
```

```
samp_mean = myRound(mean(var_vec))
samp_var = myRound(var(var_vec))
mean_res_vec = c(mean_res_vec, samp_mean)
var_res_vec = c(var_res_vec, samp_var)
hist(var_vec,
     # ylim=c(0,500),
     # breaks=20,
     xlim=c(0,20),
     main=paste("Monte Carlo ",
                "(n=",n,")"))

# Parametric
# Estimate mean and variance from the sample data
z = data400[1:n]
B_var = var(z)
B_mean = mean(z)
# simulate new data with
set.seed(0)
B_mat = matrix(rnorm(n*m, mean = B_mean, sd = sqrt(B_var)),nrow=m)
B_var_vec = apply(B_mat,1,var)
B_samp_mean = myRound(mean(B_var_vec))
B_samp_var = myRound(var(B_var_vec))
mean_res_vec = c(mean_res_vec, B_samp_mean)
var_res_vec = c(var_res_vec, B_samp_var)
hist(B_var_vec,
     # ylim=c(0,500),
     # breaks=20,
     xlim=c(0,20),
     main=paste("Parametric ",
                "(n=",n,")"))

# c) Player C
# sample new data
z=sample(x=data400[1:n], size=n*m, replace=TRUE)
C_mat = matrix(z,nrow=m)
C_var_vec = apply(C_mat,1,var)
C_samp_mean = myRound(mean(C_var_vec))
C_samp_var = myRound(var(C_var_vec))
mean_res_vec = c(mean_res_vec, C_samp_mean)
var_res_vec = c(var_res_vec, C_samp_var)
hist(C_var_vec,
     # ylim=c(0,500),
     # breaks=20,
     xlim=c(0,20),
     main=paste("Non-Parametric ",
                "(n=",n,")"))

mean_res_df[as.character(n)] = mean_res_vec
var_res_df[as.character(n)] = var_res_vec
```

```
}
```




```
# display tables
knitr::kable(mean_res_df,
             format = "markdown",
             caption = "Mean for different n values (Case 2).")
```

Table 3: Mean for different n values (Case 2).

Method	10	25	50	100	200	400
Monte Carlo	3.061	3.021	2.940	2.917	3.004	2.976
Parametric	0.583	1.050	2.471	2.434	2.828	2.614
Non-Parametric	0.528	1.031	2.407	2.410	2.816	2.633

```
knitr::kable(var_res_df,
             format = "markdown",
             caption = "Variance for different n values (Case 2).")
```

Table 4: Variance for different n values (Case 2).

Method	10	25	50	100	200	400
Monte Carlo	41.988	12.041	8.706	3.897	3.778	2.057
Parametric	0.080	0.089	0.242	0.118	0.079	0.036
Non-Parametric	0.044	0.079	1.281	0.422	0.409	0.136

Case 3

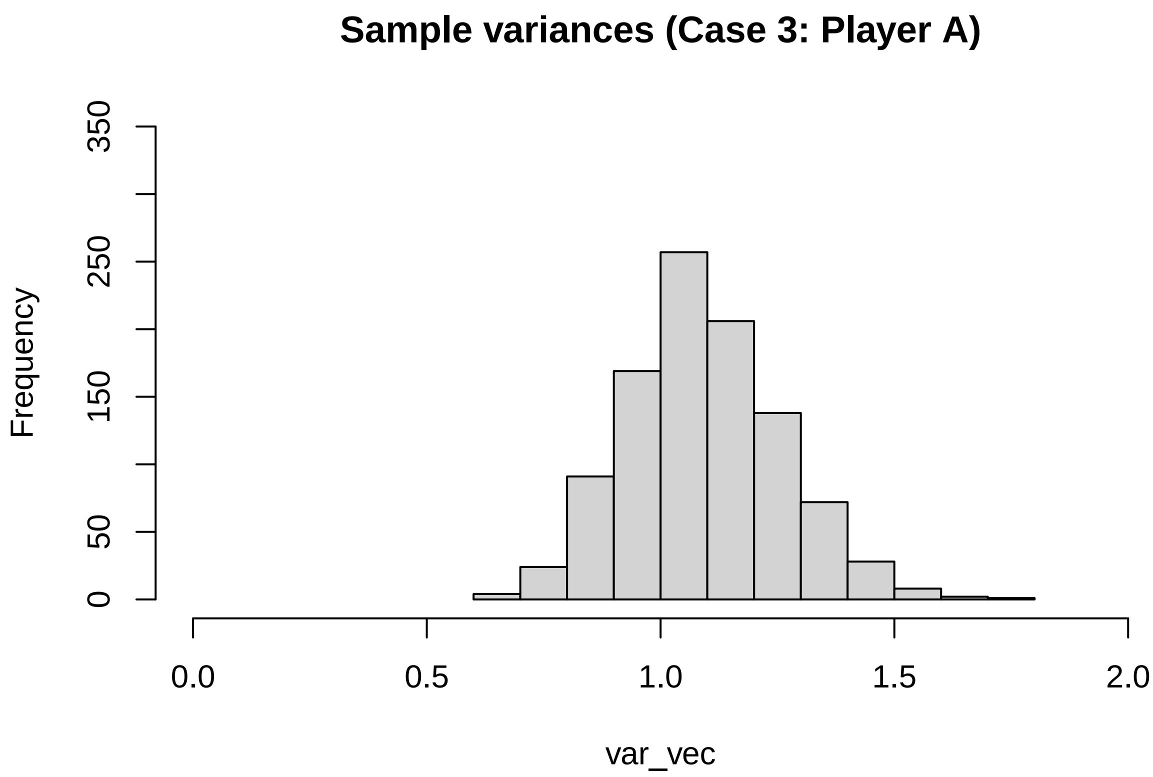
```
# Case 3
n = 100
m=1000
set.seed(0)
origData = rt(n,df=25); # case 3


# a) Player A
# generate 1000 data sets
set.seed(0)
z = rt(n*m,df=25)
mat = matrix(z,nrow=m)
#calculate variance for each row
var_vec = apply(mat,1,var)
samp_mean = mean(var_vec)
samp_var = var(var_vec)
print(paste("The sample mean is (Case 3: Player A): ", myRound(samp_mean)))
```

```
## [1] "The sample mean is (Case 3: Player A):  1.094"
```

```
print(paste("The sample variance is (Case 3: Player A): ", myRound(samp_var)))
```

```
## [1] "The sample variance is (Case 3: Player A):  0.027"
```

```
hist(var_vec,
     ylim=c(0,350),
     # breaks=20,
     xlim=c(0,2),
     main="Sample variances (Case 3: Player A)")
```



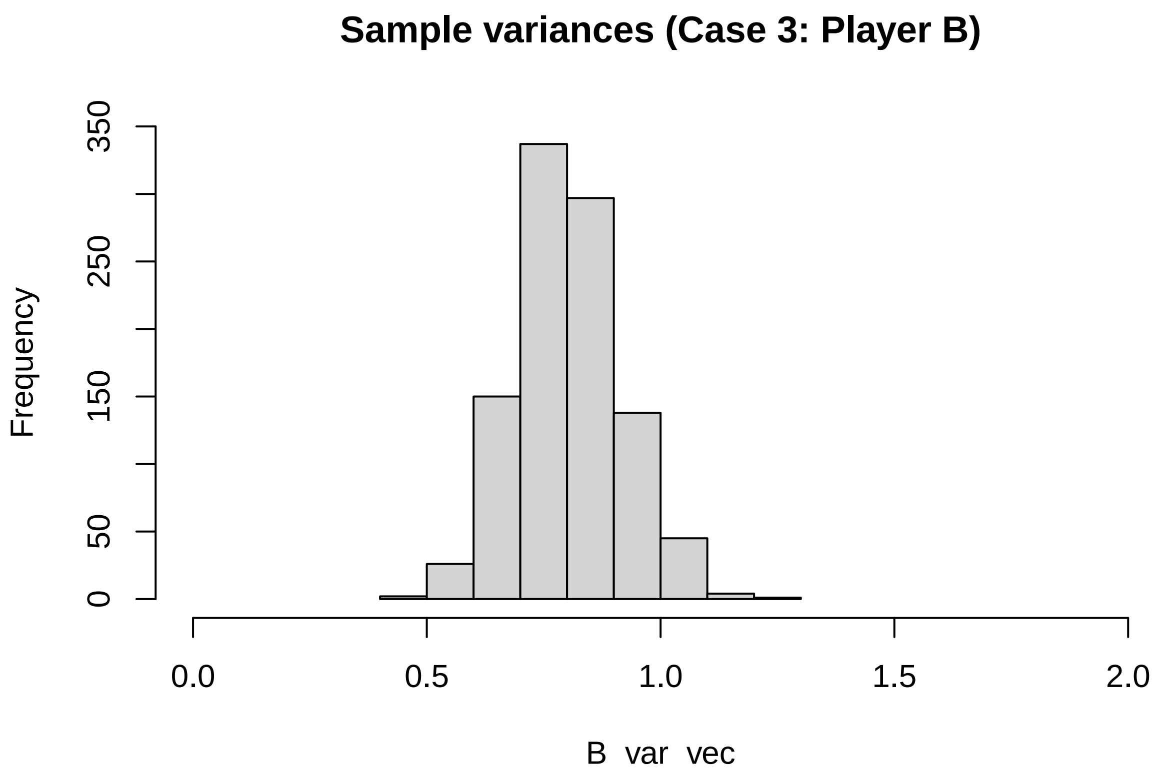

```r
# b) Player B
# Estimate mean and variance from the sample data
B_var = var(origData)
B_mean = mean(origData)
# simulate new data with
set.seed(0)
B_mat = matrix(rnorm(n*m, mean = B_mean, sd = sqrt(B_var)),nrow=m)
B_var_vec = apply(B_mat,1,var)
B_samp_mean = mean(B_var_vec)
B_samp_var = var(B_var_vec)
print(paste("The sample mean is (Case 3: Player B): ", myRound(B_samp_mean)))
```

```
## [1] "The sample mean is (Case 3: Player B):  0.801"
```

```r
print(paste("The sample variance is (Case 3: Player B): ", myRound(B_samp_var)))
```

```
## [1] "The sample variance is (Case 3: Player B):  0.013"
```

```r
hist(B_var_vec,
     ylim=c(0,350),
     # breaks=20,
     xlim=c(0,2),
     main="Sample variances (Case 3: Player B)")
```



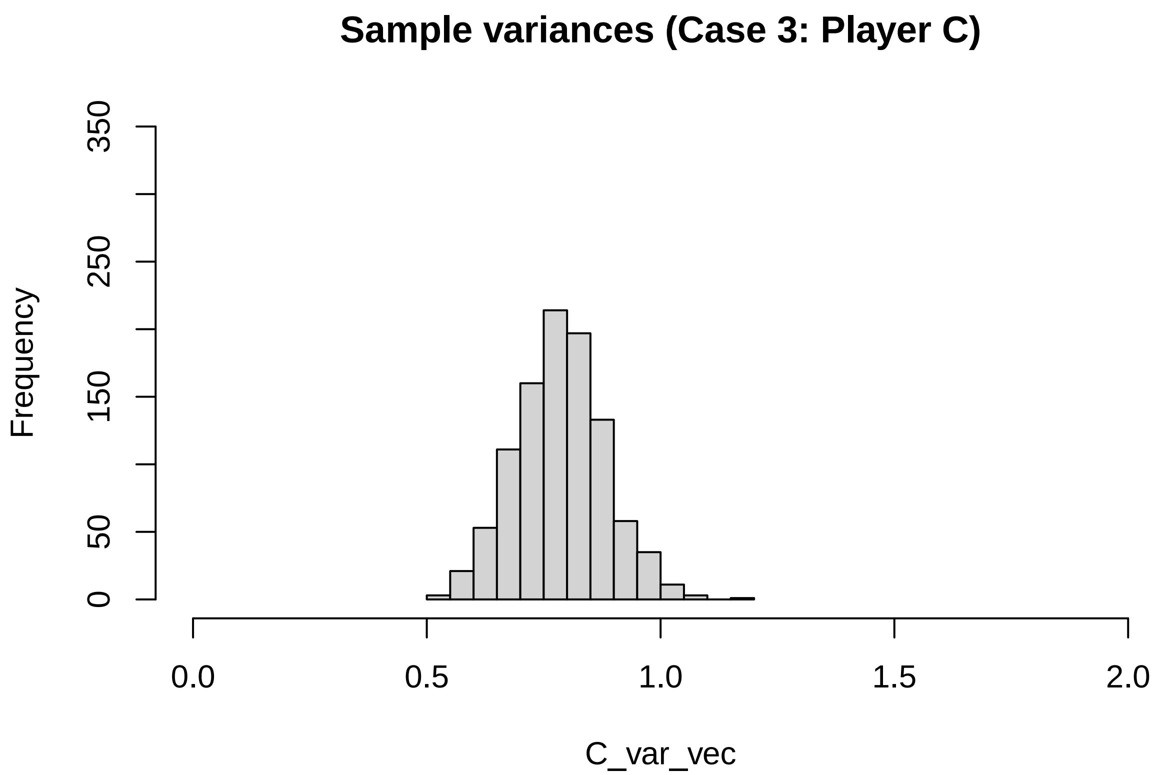

```
# c) Player C
# sample new data
z=sample(x=origData, size=n*m, replace=TRUE)
C_mat = matrix(z,nrow=m)
C_var_vec = apply(C_mat,1,var)
C_samp_mean = mean(C_var_vec)
C_samp_var = var(C_var_vec)
print(paste("The sample mean is (Case 3: Player C): ", myRound(C_samp_mean)))
```

```
## [1] "The sample mean is (Case 3: Player C):  0.786"
```

```
print(paste("The sample variance is (Case 3: Player C): ", myRound(C_samp_var)))
```

```
## [1] "The sample variance is (Case 3: Player C):  0.009"
```

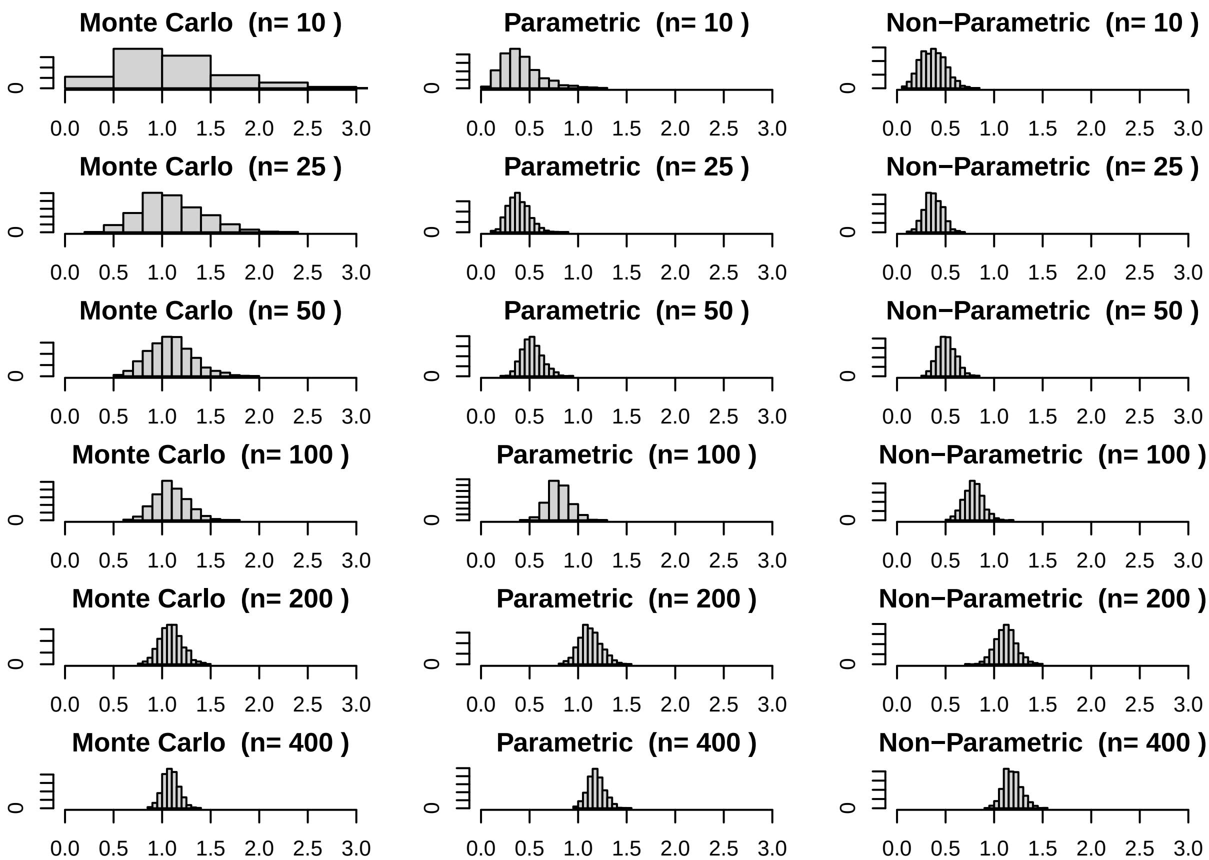
```
hist(C_var_vec,
     ylim=c(0,350),
     # breaks=20,
     xlim=c(0,2),
     main="Sample variances (Case 3: Player C)")
```




```
# e)
n_vec=c(10, 25, 50, 100, 200, 400)
methods_vec = c("Monte Carlo", "Parametric", "Non-Parametric")
m=1000
set.seed(0)
data400 = rt(400,df=25)

# save results in tables
mean_res_df = data.frame(matrix(ncol=7, nrow=3))
var_res_df = data.frame(matrix(ncol=7, nrow=3))
colnames(mean_res_df) = c("Method", n_vec)
colnames(var_res_df) = c("Method", n_vec)
mean_res_df["Method"] = methods_vec
var_res_df["Method"] = methods_vec

par(mfrow=c(6,3),mar=c(2,2,2,2))
for(n in n_vec){
  mean_res_vec = c()
  var_res_vec = c()

  #Monte Carlo
  set.seed(0)
  z = rt(n*m, df=25)
  mat = matrix(z,nrow=m)
  var_vec = apply(mat,1,var)
```

```r
samp_mean = myRound(mean(var_vec))
samp_var = myRound(var(var_vec))
mean_res_vec = c(mean_res_vec, samp_mean)
var_res_vec = c(var_res_vec, samp_var)
hist(var_vec,
     # ylim=c(0,500),
     # breaks=20,
     xlim=c(0,3),
     main=paste("Monte Carlo ",
                "(n=",n,")"))

# Parametric
# Estimate mean and variance from the sample data
z = data400[1:n]
B_var = var(z)
B_mean = mean(z)
# simulate new data with
set.seed(0)
B_mat = matrix(rnorm(n*m, mean = B_mean, sd = sqrt(B_var)),nrow=m)
B_var_vec = apply(B_mat,1,var)
B_samp_mean = myRound(mean(B_var_vec))
B_samp_var = myRound(var(B_var_vec))
mean_res_vec = c(mean_res_vec, B_samp_mean)
var_res_vec = c(var_res_vec, B_samp_var)
hist(B_var_vec,
     # ylim=c(0,500),
     # breaks=20,
     xlim=c(0,3),
     main=paste("Parametric ",
                "(n=",n,")"))

# c) Player C
# sample new data
z=sample(x=data400[1:n], size=n*m, replace=TRUE)
C_mat = matrix(z,nrow=m)
C_var_vec = apply(C_mat,1,var)
C_samp_mean = myRound(mean(C_var_vec))
C_samp_var = myRound(var(C_var_vec))
mean_res_vec = c(mean_res_vec, C_samp_mean)
var_res_vec = c(var_res_vec, C_samp_var)
hist(C_var_vec,
     # ylim=c(0,500),
     # breaks=20,
     xlim=c(0,3),
     main=paste("Non-Parametric ",
                "(n=",n,")"))

mean_res_df[as.character(n)] = mean_res_vec
var_res_df[as.character(n)] = var_res_vec
```

```
}
```




```r
# display tables
knitr::kable(mean_res_df,
             format = "markdown",
             caption = "Mean for different n values (Case 3).")
```

Table 5: Mean for different n values (Case 3).

Method	10	25	50	100	200	400
Monte Carlo	1.089	1.095	1.090	1.094	1.091	1.088
Parametric	0.409	0.387	0.527	0.801	1.124	1.184
Non-Parametric	0.370	0.376	0.511	0.786	1.121	1.187

```r
knitr::kable(var_res_df,
             format = "markdown",
             caption = "Variance for different n values (Case 3).")
```

Table 6: Variance for different n values (Case 3).

Method	10	25	50	100	200	400
Monte Carlo	0.283	0.103	0.051	0.027	0.013	0.007
Parametric	0.040	0.012	0.011	0.013	0.012	0.007
Non-Parametric	0.016	0.008	0.009	0.009	0.011	0.008

Problem 2

```
myCVids <- function(n, K, seed=0) {
# balanced subsets generation (subset sizes differ by at most 1)
# n is the number of observations/rows in the training set
# K is the desired number of folds (e.g., 5 or 10)
set.seed(seed);
t = floor(n/K); r = n-t*K;
id0 = rep((1:K),times=t)
ids = sample(id0,t*K)
if (r > 0) {ids = c(ids, sample(K,r))}
ids
}

# two-sample t test
column_t_test <- function(features, target){
  tests = lapply(seq(1,ncol(features)),function(x){t.test(features[,x]~target)})
  pval_lst = lapply(seq(1,length(tests)),function(x){tests[[x]]$p.value})
  return(list(tests, pval_lst))
}


# keep features with p-value <= 0.05
top_features <- function(pval_lst, p_val, p_min){
  bool_lst = lapply(seq(1,length(pval_lst)),function(x){pval_lst[[x]] < p_val})
  if(sum(as.integer(bool_lst))<=p_min){
    pval_df = t(data.frame(pval_lst))
    row.names(pval_df) <- NULL
    bool_df=data.frame(pval_sig=matrix(unlist(bool_lst), nrow=length(bool_lst), byrow=TRUE))
    bool_df$p_val <- pval_df[,1]
    selected_df = bool_df[bool_df$pval_sig==TRUE,]
    feat_index_vec = as.numeric(rownames(selected_df))
    return(feat_index_vec)
  }else{
    pval_df = t(data.frame(pval_lst))
    row.names(pval_df) <- NULL
    bool_df=data.frame(pval_sig=matrix(unlist(bool_lst), nrow=length(bool_lst), byrow=TRUE))
    bool_df$p_val <- pval_df[,1]
    bool_df$p_top = FALSE
    sorted_res = sort(bool_df$p_val, index.return=TRUE)
```

```
    bool_df[head(sorted_res$ix,p_min),]$p_top = TRUE
    bool_df$p_select = as.logical(bool_df$pval_sig*bool_df$p_top)
    selected_df = bool_df[bool_df$p_select==TRUE,]
    feat_index_vec = as.numeric(rownames(selected_df))
    return(feat_index_vec)
  }


}

# Mis-classification ratio calculation
MCR <- function(true_vals, pred_probs, threshold=0.5){
  if(length(true_vals)!=length(pred_probs)){
    print("ERROR: predictions and true values not of same shape")
  }else{
    pred_vals = as.integer((pred_probs > threshold))
    mcr = sum(pred_vals != true_vals)/length(true_vals)
    return(mcr)
  }
}
```

```
# Generate data
set.seed(0) # set seed
n = 25 # number of samples in each class
nr = n*2 # total number of samples
Y = c(rep(1,n),rep(0,n)) # target values
k=10 # k value for cross validation
p_star_vec = c(5,10,20,40) # p* values to select the top features in the data
i_vec = seq(5) #different values of i for different sized data sets

# create matrix where results will be stored
mcr_i_pstar_df = data.frame(matrix(0,
                                   nrow = length(p_star_vec),
                                   ncol = length(i_vec)+1))
mcr_i_pstar_df[,1] = p_star_vec # add p* values to results table

# loop over values of i to make different data sets
for(i in i_vec){
  nc = 200*2^i # nc = 6400
  M = matrix(rnorm(nr*nc),nrow=nr)
  X = M[,1:nc] # features of data
  # calculate cross validation indexes
  # this is applied only after feature selection
  ids = myCVids(n=nr, K=k, seed=0)
  # feature selection
  # apply two-sample t-test
  t_test_results = column_t_test(features=X, target=Y)
  # get all features according to p-value at differing values of p_star
  # create vector to store mean mcr values for each p* subset
```

```
  mean_pstar_mcr_vec = c()

  # loop over p* values to create a subset of selected features
  for(p_star in p_star_vec){
    feat_index_vec = c(top_features(pval_lst=t_test_results[[2]],
                                    p_val=0.05,
                                    p_min=p_star))

    X_pstar = X[,feat_index_vec]
    k_mcr_vec = c() # store all MCR values for each k

    # loop over k to calculate the MCR for each fold
    for( k in seq(k)){
      isk = (ids == k) # k varies from 1 to K
      valid.k = which(isk) # test data index
      train.k = which(!isk) # train data index
      # get all training data in single data frame
      train_df = data.frame(Y=Y[train.k], X_pstar[train.k,])
      # get all testing data in single data frame
      val_df = data.frame(Y=Y[valid.k], X_pstar[valid.k,])

      # train a Logistic regression model
      LR = glm(Y ~ .,
              data=train_df,
              family="binomial")

      # get estimated probabilities on the test data
      LR_probs = data.frame(
                    predict(LR,
                    val_df,
                    type ="response"
                    )
                  )

      # use the probabilities to calculate the MCR
      mcr = MCR(
              true_vals=val_df$Y,
              pred_probs=LR_probs[,1],
              threshold=0.5)

      k_mcr_vec = c(k_mcr_vec, mcr)
    }
    mean_pstar_mcr = mean(k_mcr_vec) # get the mean MCR for all values of k

    mean_pstar_mcr_vec = c(mean_pstar_mcr_vec, mean_pstar_mcr)
  }
  mcr_i_pstar_df[,i+1] = mean_pstar_mcr_vec
}
# add all data to a data frame and transform to be in the correct format
```

```r
mcr_i_pstar_df = t(mcr_i_pstar_df)
rownames(mcr_i_pstar_df) = c("p*", i_vec)
knitr::kable(mcr_i_pstar_df, format = "markdown") # display table
```

p*	5.00	10.00	20.00	40.00
1	0.30	0.26	0.26	0.26
2	0.24	0.16	0.20	0.32
3	0.16	0.10	0.10	0.26
4	0.18	0.06	0.04	0.18
5	0.16	0.02	0.04	0.30

Problem 3

```r
# Generate data
set.seed(0) # set seed
n = 25 # number of samples in each class
nr = n*2 # total number of samples
Y = c(rep(1,n),rep(0,n)) # target values
k=10 # k value for cross validation
p_star_vec = c(5,10,20,40) # p* values to select the top features in the data
i_vec = seq(5) #different values of i for different sized data sets

# create matrix where results will be stored
mcr_i_pstar_df = data.frame(matrix(0,
                                   nrow = length(p_star_vec),
                                   ncol = length(i_vec)+1))
mcr_i_pstar_df[,1] = p_star_vec # add p* values to results table
progress=0
# loop over values of i to make different data sets
for(i in i_vec){
  nc = 200*2^i # nc = 6400
  M = matrix(rnorm(nr*nc),nrow=nr)
  X = M[,1:nc] # features of data
  # calculate cross validation indexes
  # this is applied only after feature selection
  ids = myCVids(n=nr, K=k, seed=0)

  # get all features according to p-value at differing values of p_star
  # create vector to store mean mcr values for each p* subset
  mean_pstar_mcr_vec = c()

  # loop over p* values to create a subset of selected features
  for(p_star in p_star_vec){
    k_mcr_vec = c() # store all mcr values for each k

    # k-fold cross validation
```

```
    # loop over k to calculate the MCR for each fold
    for( k in seq(k)){
      progress=progress+1
      # print(myRound(progress/200)*100)
      isk = (ids == k) # k varies from 1 to K
      valid.k = which(isk) # test data index
      train.k = which(!isk) # train data index

      # feature selection
      # apply two-sample t-test
      t_test_results = column_t_test(features=X[train.k,], target=Y[train.k])
      feat_index_vec = c(top_features(pval_lst=t_test_results[[2]],
                                 p_val=0.05,
                                 p_min=p_star))
      X_pstar = X[,feat_index_vec]

      # get all training data in single data frame
      train_df = data.frame(Y=Y[train.k], X_pstar[train.k,])
      # get all testing data in single data frame
      val_df = data.frame(Y=Y[valid.k], X_pstar[valid.k,])

      # train a Logistic regression model
      LR = glm(Y ~ .,
              data=train_df,
              family="binomial")

      # get estimated probabilities on the test data
      LR_probs = data.frame(
                    predict(LR,
                    val_df,
                    type ="response"
                    )
                  )

      # use the probabilities to calculate the MCR
      mcr = MCR(
              true_vals=val_df$Y,
              pred_probs=LR_probs[,1],
              threshold=0.5)

      k_mcr_vec = c(k_mcr_vec, mcr)
    }
    mean_pstar_mcr = mean(k_mcr_vec) # get the mean MCR for all values of k

    mean_pstar_mcr_vec = c(mean_pstar_mcr_vec, mean_pstar_mcr)
  }
  mcr_i_pstar_df[,i+1] = mean_pstar_mcr_vec
}
# add all data to a data frame and transform to be in the correct format
```

```r
mcr_i_pstar_df = t(mcr_i_pstar_df)
rownames(mcr_i_pstar_df) = c("p*", i_vec)
knitr::kable(mcr_i_pstar_df, format = "markdown") # display table
```

p*	5.00	10.00	20.00	40.00
1	0.46	0.48	0.48	0.46
2	0.48	0.48	0.52	0.62
3	0.38	0.36	0.32	0.48
4	0.38	0.42	0.32	0.42
5	0.38	0.48	0.56	0.48

Note: it takes much longer this way… like waayyy longer more data = worse generalization with problem 2 but not 3.

Problem 4

```r
# functions
myCVids <- function(n, K, seed=0) {
# balanced subsets generation (subset sizes differ by at most 1)
# n is the number of observations/rows in the training set
# K is the desired number of folds (e.g., 5 or 10)
set.seed(seed);
t = floor(n/K); r = n-t*K;
id0 = rep((1:K),times=t)
ids = sample(id0,t*K)
if (r > 0) {ids = c(ids, sample(K,r))}
ids
}

genData <- function(n, seed=0) {
set.seed(seed)
x = seq(-1,1,length.out=n)
y = x - x^2 + 2*rnorm(n) # true sigma = 2;
out.df = data.frame(x=x, y=y)
out.df
}
```

4.1

```r
# parameters
set.seed(100)
d_vec = seq(0,4)
# generate data
train.df = genData(n=200,seed=100)
```

```r
test.df = genData(400)

# create k-fold indexes
k=5
inds.part = myCVids(n=nrow(train.df),K=k)
# create a data frame to save results into
M = matrix(0, nrow = (length(d_vec)), ncol = k+1)
KFOLD_df = data.frame(M)
KFOLD_df[,1] = d_vec # add degrees to results table
colnames(KFOLD_df) = c("Degree", seq(k))

# loop over k-folds
for( k in seq(k)){
  isk = (inds.part == k) # k varies from 1 to K
  valid.k = which(isk) # test data index
  train.k = which(!isk)
  # split data
  train_sub_df = train.df[train.k,]
  valid_df = train.df[valid.k,]

  d_mse_vec = c()
  for(d in d_vec){
    if(d==0){
      # train a polynomial model
      PR = lm(y ~ 1, data=train_sub_df)
    }else{
     # train a polynomial model
      PR = lm(y ~ poly(x, d, raw = TRUE), data=train_sub_df)
    }
    # get predicted values to calculate MSE
    pred_val = predict(PR, valid_df, type="response")
    pred_true_val_df = data.frame(pred = pred_val, actual = valid_df$y)
    #calculate MSE
    mse = mean((pred_true_val_df$actual - pred_true_val_df$pred)^2)
    d_mse_vec = c(d_mse_vec, mse)
  }
  KFOLD_df[,k+1] = d_mse_vec
}

# tally of MSE
KFOLD_df$Total_MSE = rowSums(KFOLD_df)
knitr::kable(KFOLD_df, format = "markdown") # display table
```

Degree	1	2	3	4	5	Total_MSE
0	2.533138	2.984333	4.243087	4.513501	4.206231	18.48029
1	2.228435	2.853997	3.653187	4.190807	3.876836	17.80326
2	2.204849	2.797685	3.639950	4.126926	3.967901	18.73731
3	2.250757	2.872660	3.635394	4.190780	3.961687	19.91128

Degree	1	2	3	4	5	Total_MSE
4	2.241189	2.894667	3.626206	4.490320	3.955490	21.20787

```
# print best d value
print(paste("The best K-fold degree is: ",
            as.character(KFOLD_df[which.min(KFOLD_df$Total_MSE),]$Degree)))
```

```
## [1] "The best K-fold degree is:  1"
```

4.2

```
# create k-fold indexes
k=nrow(train.df)
inds.part = myCVids(n=nrow(train.df),K=k)
# create a data frame to save results into
M = matrix(0, nrow = (length(d_vec)), ncol = k+1)
LOOCV_df = data.frame(M)
LOOCV_df[,1] = d_vec # add degrees to results table
colnames(LOOCV_df) = c("Degree", seq(k))

# loop over k-folds
for( k in seq(k)){
  isk = (inds.part == k) # k varies from 1 to K
  valid.k = which(isk) # test data index
  train.k = which(!isk)
  # split data
  train_sub_df = train.df[train.k,]
  valid_df = train.df[valid.k,]

  d_mse_vec = c()
  for(d in d_vec){
    if(d==0){
      # train a polynomial model
      PR = lm(y ~ 1, data=train_sub_df)
    }else{
     # train a polynomial model
      PR = lm(y ~ poly(x, d, raw = TRUE), data=train_sub_df)
    }
    # get predicted values to calculate MSE
    pred_val = predict(PR, valid_df, type="response")
    pred_true_val_df = data.frame(pred = pred_val, actual = valid_df$y)
    #calculate MSE
    mse = mean((pred_true_val_df$actual - pred_true_val_df$pred)^2)
    d_mse_vec = c(d_mse_vec, mse)
  }
  LOOCV_df[,k+1] = d_mse_vec
}
```

```r
# tally of MSE
LOOCV_df$Total_MSE = rowSums(LOOCV_df)
knitr::kable(LOOCV_df[,c(1,k+2)], format = "markdown") # display table
```

Degree	Total_MSE
0	740.7244
1	677.5361
2	679.7304
3	686.7229
4	693.2117

```r
# print best d value
print(paste("The best LOOCV degree is: ",
            as.character(LOOCV_df[which.min(LOOCV_df$Total_MSE),]$Degree)))
```

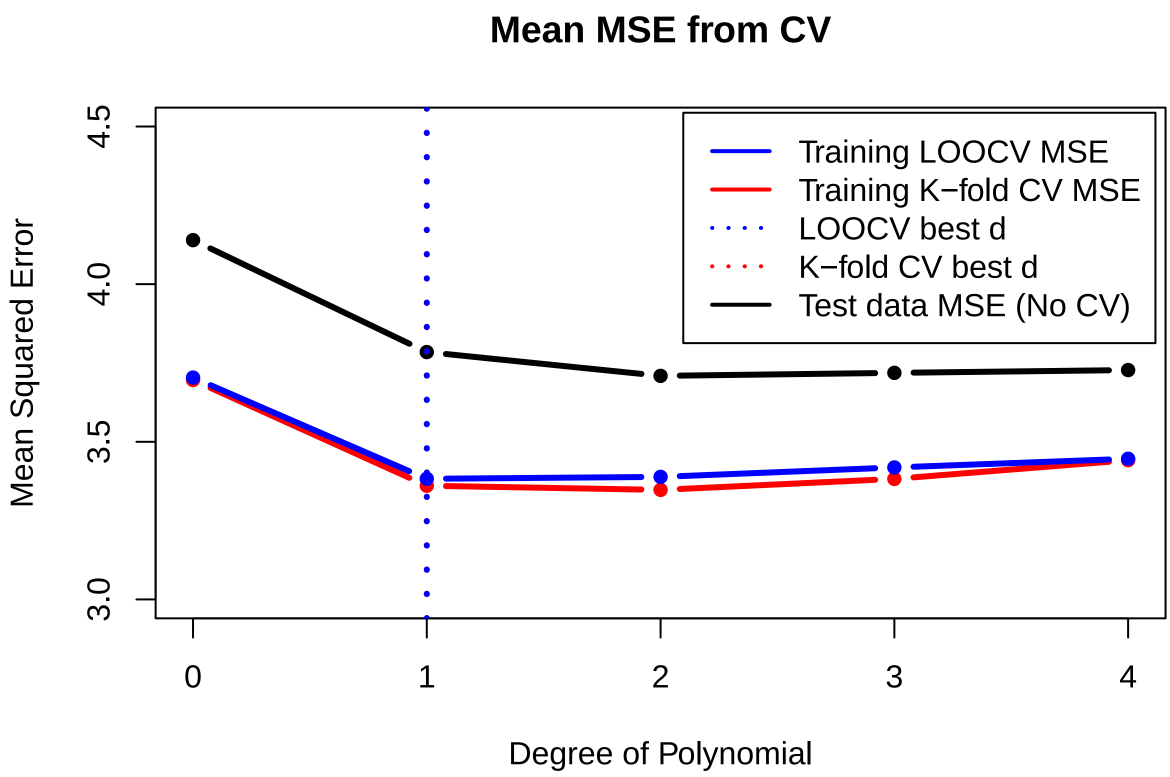
```
## [1] "The best LOOCV degree is:  1"
```

There is a part in lectures where it says that LOOCV is the same as k-fold for polynomial regression as a special case.page 179. 183

4.3

```r
# estimate test data MSE
test_d_vec = c()
for(d in d_vec){
  if(d==0){
    # train a polynomial model
    PR = lm(y ~ 1, data=train.df)
  }else{
   # train a polynomial model
    PR = lm(y ~ poly(x, d, raw = TRUE), data=train.df)
  }
  # get predicted values to calculate MSE
  pred_val = predict(PR, test.df, type="response")
  pred_true_val_df = data.frame(pred = pred_val, actual = test.df$y)
  #calculate MSE
  mse = mean((pred_true_val_df$actual - pred_true_val_df$pred)^2)
  test_d_vec = c(test_d_vec, mse)
}

# visualize LOOCV and K-fold CV
# calculate max and min limits of data KFOLD
kfold_max_vec = apply(KFOLD_df[, 2:(ncol(KFOLD_df)-1)], 1, max)
kfold_min_vec = apply(KFOLD_df[, 2:(ncol(KFOLD_df)-1)], 1, min)
kfold_mean_vec = apply(KFOLD_df[, 2:(ncol(KFOLD_df)-1)], 1, mean)
```

```r
# calculate max and min limits of data for LOOCV
loocv_max_vec = apply(LOOCV_df[, 2:(ncol(LOOCV_df)-1)], 1, max)
loocv_min_vec = apply(LOOCV_df[, 2:(ncol(LOOCV_df)-1)], 1, min)
loocv_mean_vec = apply(LOOCV_df[, 2:(ncol(LOOCV_df)-1)], 1, mean)

{plot(x=KFOLD_df$Degree,
     y=kfold_mean_vec,
     ylab="Mean Squared Error",
     main="Mean MSE from CV",
     xlab="Degree of Polynomial",
     type='b',
     col='red',
     pch = 16,
     lwd=3,
     ylim=c(3,4.5))
lines(x=LOOCV_df$Degree,
     y=loocv_mean_vec,
     type='b',
     col='blue',
     pch = 16,
     lwd=3)
lines(x=LOOCV_df$Degree,
     y=test_d_vec,
     type='b',
     col='black',
     pch = 16,
     lwd=3)
abline(v=KFOLD_df[which.min(KFOLD_df$Total_MSE),]$Degree,
     col='red',
     pch = 16,
     lty=3,
     lwd=3)
abline(v=LOOCV_df[which.min(LOOCV_df$Total_MSE),]$Degree,
     col='blue',
     pch = 16,
     lty=3,
     lwd=3)
legend("topright",
       inset = 0.01,
       legend = c("Training LOOCV MSE", "Training K-fold CV MSE", "LOOCV best d", "K-fold CV best
       lty = c(1,1,3,3),
       col = c("blue", "red","blue", "red", "black"),
       lwd = 2)}
```



4.4

```
# parameters
set.seed(100)
d_vec = seq(0,4)
# generate data
train.df = genData(n=400,seed=100)
test.df = genData(400)

# create k-fold indexes
k=5
inds.part = myCVids(n=nrow(train.df),K=k)
# create a data frame to save results into
M = matrix(0, nrow = (length(d_vec)), ncol = k+1)
KFOLD_df = data.frame(M)
KFOLD_df[,1] = d_vec # add degrees to results table
colnames(KFOLD_df) = c("Degree", seq(k))

# loop over k-folds
for( k in seq(k)){
  isk = (inds.part == k) # k varies from 1 to K
  valid.k = which(isk) # test data index
  train.k = which(!isk)
```

```r
  # split data
  train_sub_df = train.df[train.k,]
  valid_df = train.df[valid.k,]

  d_mse_vec = c()
  for(d in d_vec){
    if(d==0){
      # train a polynomial model
      PR = lm(y ~ 1, data=train_sub_df)
    }else{
     # train a polynomial model
      PR = lm(y ~ poly(x, d, raw = TRUE), data=train_sub_df)
    }
    # get predicted values to calculate MSE
    pred_val = predict(PR, valid_df, type="response")
    pred_true_val_df = data.frame(pred = pred_val, actual = valid_df$y)
    #calculate MSE
    mse = mean((pred_true_val_df$actual - pred_true_val_df$pred)^2)
    d_mse_vec = c(d_mse_vec, mse)
  }
  KFOLD_df[,k+1] = d_mse_vec
}

# tally of MSE
KFOLD_df$Total_MSE = rowSums(KFOLD_df)
knitr::kable(KFOLD_df, format = "markdown") # display table
```

Degree	1	2	3	4	5	Total_MSE
0	3.438751	4.942601	3.672744	4.779153	4.645821	21.47907
1	3.584766	4.483129	3.614016	4.226312	4.321004	21.22923
2	3.427265	4.326183	3.499535	4.504805	4.064654	21.82244
3	3.381529	4.333047	3.556816	4.469806	4.060612	22.80181
4	3.375970	4.375542	3.547353	4.476218	4.128512	23.90359

```r
# print best d value
print(paste("The best K-fold degree is: ",
            as.character(KFOLD_df[which.min(KFOLD_df$Total_MSE),]$Degree)))
```

```
## [1] "The best K-fold degree is:  1"
```

```r
# create k-fold indexes
k=nrow(train.df)
inds.part = myCVids(n=nrow(train.df),K=k)
# create a data frame to save results into
M = matrix(0, nrow = (length(d_vec)), ncol = k+1)
LOOCV_df = data.frame(M)
LOOCV_df[,1] = d_vec # add degrees to results table
```

```r
colnames(LOOCV_df) = c("Degree", seq(k))

# loop over k-folds
for( k in seq(k)){
  isk = (inds.part == k) # k varies from 1 to K
  valid.k = which(isk) # test data index
  train.k = which(!isk)
  # split data
  train_sub_df = train.df[train.k,]
  valid_df = train.df[valid.k,]

  d_mse_vec = c()
  for(d in d_vec){
    if(d==0){
      # train a polynomial model
      PR = lm(y ~ 1, data=train_sub_df)
    }else{
     # train a polynomial model
      PR = lm(y ~ poly(x, d, raw = TRUE), data=train_sub_df)
    }
    # get predicted values to calculate MSE
    pred_val = predict(PR, valid_df, type="response")
    pred_true_val_df = data.frame(pred = pred_val, actual = valid_df$y)
    #calculate MSE
    mse = mean((pred_true_val_df$actual - pred_true_val_df$pred)^2)
    d_mse_vec = c(d_mse_vec, mse)
  }
  LOOCV_df[,k+1] = d_mse_vec
}

# tally of MSE
LOOCV_df$Total_MSE = rowSums(LOOCV_df)
knitr::kable(LOOCV_df[,c(1,k+2)], format = "markdown") # display table
```

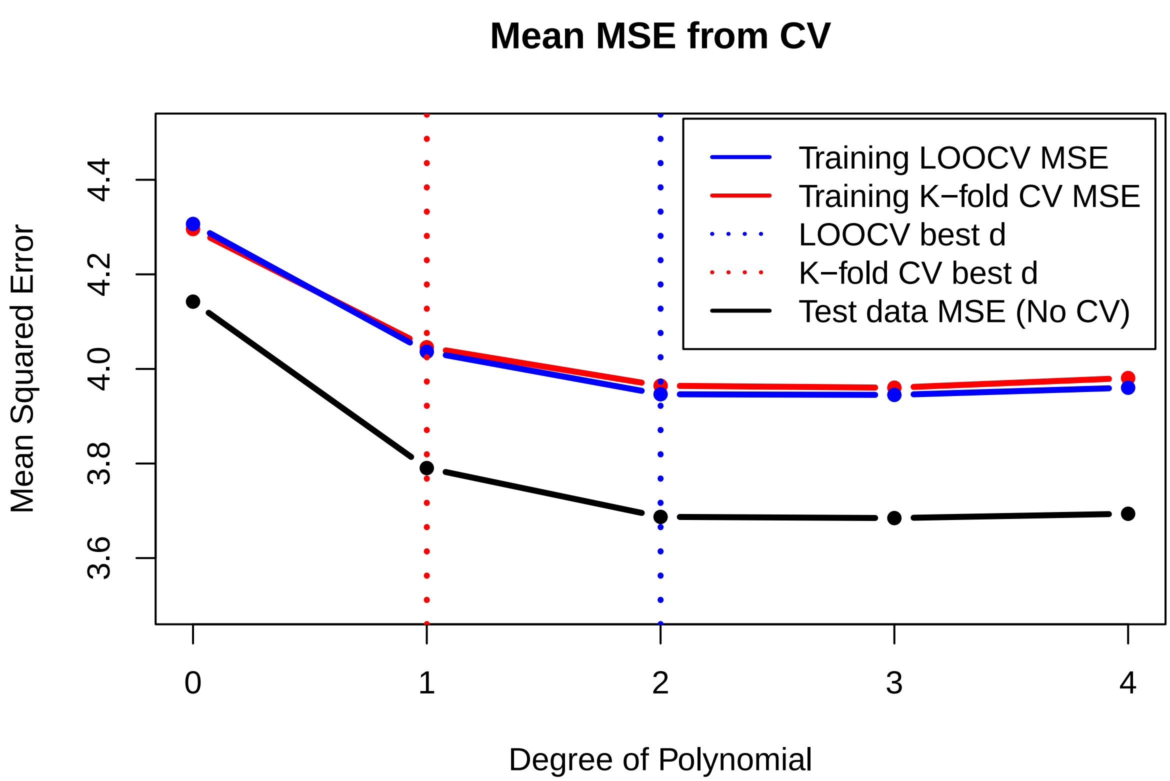
Degree	Total_MSE
0	1722.641
1	1615.560
2	1580.588
3	1581.030
4	1588.124

```r
# print best d value
print(paste("The best LOOCV degree is: ",
            as.character(LOOCV_df[which.min(LOOCV_df$Total_MSE),]$Degree)))
```

```
## [1] "The best LOOCV degree is:  2"
```

```r
# estimate test data MSE
test_d_vec = c()
for(d in d_vec){
  if(d==0){
    # train a polynomial model
    PR = lm(y ~ 1, data=train.df)
  }else{
   # train a polynomial model
    PR = lm(y ~ poly(x, d, raw = TRUE), data=train.df)
  }
  # get predicted values to calculate MSE
  pred_val = predict(PR, test.df, type="response")
  pred_true_val_df = data.frame(pred = pred_val, actual = test.df$y)
  #calculate MSE
  mse = mean((pred_true_val_df$actual - pred_true_val_df$pred)^2)
  test_d_vec = c(test_d_vec, mse)
}

# visualize LOOCV and K-fold CV
# calculate max and min limits of data KFOLD
kfold_max_vec = apply(KFOLD_df[, 2:(ncol(KFOLD_df)-1)], 1, max)
kfold_min_vec = apply(KFOLD_df[, 2:(ncol(KFOLD_df)-1)], 1, min)
kfold_mean_vec = apply(KFOLD_df[, 2:(ncol(KFOLD_df)-1)], 1, mean)
# calculate max and min limits of data for LOOCV
loocv_max_vec = apply(LOOCV_df[, 2:(ncol(LOOCV_df)-1)], 1, max)
loocv_min_vec = apply(LOOCV_df[, 2:(ncol(LOOCV_df)-1)], 1, min)
loocv_mean_vec = apply(LOOCV_df[, 2:(ncol(LOOCV_df)-1)], 1, mean)

{plot(x=KFOLD_df$Degree,
     y=kfold_mean_vec,
     ylab="Mean Squared Error",
     main="Mean MSE from CV",
     xlab="Degree of Polynomial",
     type='b',
     col='red',
     pch = 16,
     lwd=3,
     ylim=c(3.5,4.5))
lines(x=LOOCV_df$Degree,
     y=loocv_mean_vec,
     type='b',
     col='blue',
     pch = 16,
     lwd=3)
lines(x=LOOCV_df$Degree,
     y=test_d_vec,
     type='b',
     col='black',
     pch = 16,
```

```
      lwd=3)
abline(v=KFOLD_df[which.min(KFOLD_df$Total_MSE),]$Degree,
      col='red',
      pch = 16,
      lty=3,
      lwd=3)
abline(v=LOOCV_df[which.min(LOOCV_df$Total_MSE),]$Degree,
      col='blue',
      pch = 16,
      lty=3,
      lwd=3)
legend("topright",
        inset = 0.01,
        legend = c("Training LOOCV MSE", "Training K-fold CV MSE", "LOOCV best d", "K-fold CV best
        lty = c(1,1,3,3),
        col = c("blue", "red","blue", "red", "black"),
        lwd = 2)}
```

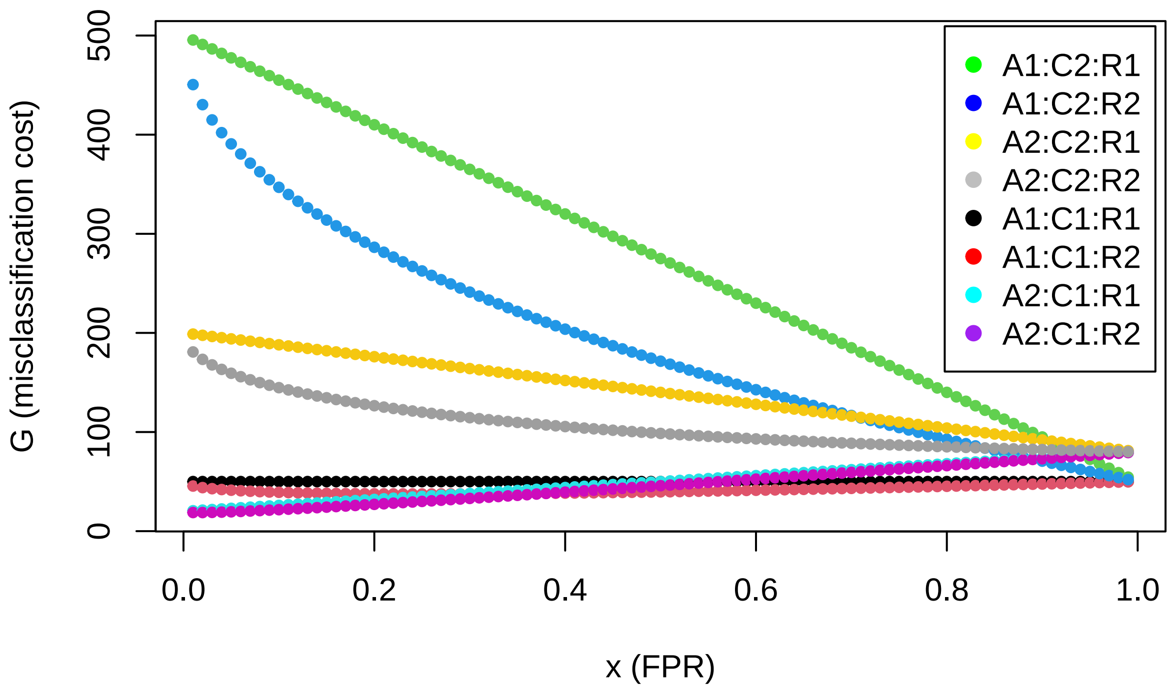


Problem 5

```
# assume values for x and cfp
# x = 0.25
```

```r
cfp=1
n=100


x_vec=c()
A_vec=c()
C_vec=c()
R_vec=c()
FPR_vec=c()
TPR_vec=c()
G_vec=c()
for (x in seq(0.01,0.99,0.01)) {
  A=c(0.5, 0.2)
  C=c(cfp, 10*cfp)
  R=c(x, sqrt(x))

  Ai=0
  for(q in A){
    Ai = Ai+1
    P=n*q
    N=n*(1-q)

    Ri=0
    for(tpr in R){
      Ri=Ri+1
      # calculate TP, FP, TN, and FN with regards to x
      TP = tpr*P
      FN = P-TP
      FP = x*N
      TN = N-FP
      FPR = x
      TPR = tpr

      Ci=0
      for(cfn in C){
        Ci=Ci+1
        G = cfn*FN + cfp*FP

        x_vec=c(x_vec,x)
        A_vec=c(A_vec,Ai)
        C_vec=c(C_vec,Ci)
        R_vec=c(R_vec,Ri)
        FPR_vec=c(FPR_vec,FPR)
        TPR_vec=c(TPR_vec,TPR)
        G_vec=c(G_vec,G)

        # print(paste("(A:",as.character(Ai),")"))
        # print(paste("(C:",as.character(Ci),")"))
        # print(paste("(R:",as.character(Ri),")"))
```

```
        # print(paste("FPR = ", FPR))
        # print(paste("TPR = ", TPR))
        # print(paste("G = ", G))
        # print("----------------------------")


      }
    }

  }
}

results_df = data.frame(x_vec,
                          A_vec,
                          C_vec,
                          R_vec,
                          FPR_vec,
                          TPR_vec,
                          G_vec)

results_df$design_id <- paste(results_df$A_vec,
                              results_df$C_vec,
                              results_df$R_vec)

# Objective function score visualization
{plot(x=results_df$x_vec,
     y=results_df$G_vec,
     col=factor(results_df$design_id),
     ylab="G (misclassification cost)",
     xlab="x (FPR)",
     pch=20)
legend("topright",
     inset = 0.01,
     pch=c(19,19,19,19),
     legend = c("A1:C2:R1", "A1:C2:R2", "A2:C2:R1", "A2:C2:R2", "A1:C1:R1", "A1:C1:R2", "A2:C1:R1
     col = c("green", "blue","yellow", "grey", "black","red", "cyan", "purple"))}
```




```
knitr::kable(results_df, format = "markdown")
```

x_vec	A_vec	C_vec	R_vec	FPR_vec	TPR_vec	G_vec	design_id
0.01	1	1	1	0.01	0.0100000	50.00000	1 1 1
0.01	1	2	1	0.01	0.0100000	495.50000	1 2 1
0.01	1	1	2	0.01	0.1000000	45.50000	1 1 2
0.01	1	2	2	0.01	0.1000000	450.50000	1 2 2
0.01	2	1	1	0.01	0.0100000	20.60000	2 1 1
0.01	2	2	1	0.01	0.0100000	198.80000	2 2 1
0.01	2	1	2	0.01	0.1000000	18.80000	2 1 2
0.01	2	2	2	0.01	0.1000000	180.80000	2 2 2
0.02	1	1	1	0.02	0.0200000	50.00000	1 1 1
0.02	1	2	1	0.02	0.0200000	491.00000	1 2 1
0.02	1	1	2	0.02	0.1414214	43.92893	1 1 2
0.02	1	2	2	0.02	0.1414214	430.28932	1 2 2
0.02	2	1	1	0.02	0.0200000	21.20000	2 1 1
0.02	2	2	1	0.02	0.0200000	197.60000	2 2 1
0.02	2	1	2	0.02	0.1414214	18.77157	2 1 2
0.02	2	2	2	0.02	0.1414214	173.31573	2 2 2
0.03	1	1	1	0.03	0.0300000	50.00000	1 1 1
0.03	1	2	1	0.03	0.0300000	486.50000	1 2 1
0.03	1	1	2	0.03	0.1732051	42.83975	1 1 2
0.03	1	2	2	0.03	0.1732051	414.89746	1 2 2

x_vec	A_vec	C_vec	R_vec	FPR_vec	TPR_vec	G_vec	design_id
0.03	2	1	1	0.03	0.0300000	21.80000	2 1 1
0.03	2	2	1	0.03	0.0300000	196.40000	2 2 1
0.03	2	1	2	0.03	0.1732051	18.93590	2 1 2
0.03	2	2	2	0.03	0.1732051	167.75898	2 2 2
0.04	1	1	1	0.04	0.0400000	50.00000	1 1 1
0.04	1	2	1	0.04	0.0400000	482.00000	1 2 1
0.04	1	1	2	0.04	0.2000000	42.00000	1 1 2
0.04	1	2	2	0.04	0.2000000	402.00000	1 2 2
0.04	2	1	1	0.04	0.0400000	22.40000	2 1 1
0.04	2	2	1	0.04	0.0400000	195.20000	2 2 1
0.04	2	1	2	0.04	0.2000000	19.20000	2 1 2
0.04	2	2	2	0.04	0.2000000	163.20000	2 2 2
0.05	1	1	1	0.05	0.0500000	50.00000	1 1 1
0.05	1	2	1	0.05	0.0500000	477.50000	1 2 1
0.05	1	1	2	0.05	0.2236068	41.31966	1 1 2
0.05	1	2	2	0.05	0.2236068	390.69660	1 2 2
0.05	2	1	1	0.05	0.0500000	23.00000	2 1 1
0.05	2	2	1	0.05	0.0500000	194.00000	2 2 1
0.05	2	1	2	0.05	0.2236068	19.52786	2 1 2
0.05	2	2	2	0.05	0.2236068	159.27864	2 2 2
0.06	1	1	1	0.06	0.0600000	50.00000	1 1 1
0.06	1	2	1	0.06	0.0600000	473.00000	1 2 1
0.06	1	1	2	0.06	0.2449490	40.75255	1 1 2
0.06	1	2	2	0.06	0.2449490	380.52551	1 2 2
0.06	2	1	1	0.06	0.0600000	23.60000	2 1 1
0.06	2	2	1	0.06	0.0600000	192.80000	2 2 1
0.06	2	1	2	0.06	0.2449490	19.90102	2 1 2
0.06	2	2	2	0.06	0.2449490	155.81021	2 2 2
0.07	1	1	1	0.07	0.0700000	50.00000	1 1 1
0.07	1	2	1	0.07	0.0700000	468.50000	1 2 1
0.07	1	1	2	0.07	0.2645751	40.27124	1 1 2
0.07	1	2	2	0.07	0.2645751	371.21243	1 2 2
0.07	2	1	1	0.07	0.0700000	24.20000	2 1 1
0.07	2	2	1	0.07	0.0700000	191.60000	2 2 1
0.07	2	1	2	0.07	0.2645751	20.30850	2 1 2
0.07	2	2	2	0.07	0.2645751	152.68497	2 2 2
0.08	1	1	1	0.08	0.0800000	50.00000	1 1 1
0.08	1	2	1	0.08	0.0800000	464.00000	1 2 1
0.08	1	1	2	0.08	0.2828427	39.85786	1 1 2
0.08	1	2	2	0.08	0.2828427	362.57864	1 2 2
0.08	2	1	1	0.08	0.0800000	24.80000	2 1 1
0.08	2	2	1	0.08	0.0800000	190.40000	2 2 1
0.08	2	1	2	0.08	0.2828427	20.74315	2 1 2
0.08	2	2	2	0.08	0.2828427	149.83146	2 2 2
0.09	1	1	1	0.09	0.0900000	50.00000	1 1 1
0.09	1	2	1	0.09	0.0900000	459.50000	1 2 1
0.09	1	1	2	0.09	0.3000000	39.50000	1 1 2
0.09	1	2	2	0.09	0.3000000	354.50000	1 2 2

x_vec	A_vec	C_vec	R_vec	FPR_vec	TPR_vec	G_vec	design_id
0.09	2	1	1	0.09	0.0900000	25.40000	2 1 1
0.09	2	2	1	0.09	0.0900000	189.20000	2 2 1
0.09	2	1	2	0.09	0.3000000	21.20000	2 1 2
0.09	2	2	2	0.09	0.3000000	147.20000	2 2 2
0.10	1	1	1	0.10	0.1000000	50.00000	1 1 1
0.10	1	2	1	0.10	0.1000000	455.00000	1 2 1
0.10	1	1	2	0.10	0.3162278	39.18861	1 1 2
0.10	1	2	2	0.10	0.3162278	346.88612	1 2 2
0.10	2	1	1	0.10	0.1000000	26.00000	2 1 1
0.10	2	2	1	0.10	0.1000000	188.00000	2 2 1
0.10	2	1	2	0.10	0.3162278	21.67544	2 1 2
0.10	2	2	2	0.10	0.3162278	144.75445	2 2 2
0.11	1	1	1	0.11	0.1100000	50.00000	1 1 1
0.11	1	2	1	0.11	0.1100000	450.50000	1 2 1
0.11	1	1	2	0.11	0.3316625	38.91688	1 1 2
0.11	1	2	2	0.11	0.3316625	339.66876	1 2 2
0.11	2	1	1	0.11	0.1100000	26.60000	2 1 1
0.11	2	2	1	0.11	0.1100000	186.80000	2 2 1
0.11	2	1	2	0.11	0.3316625	22.16675	2 1 2
0.11	2	2	2	0.11	0.3316625	142.46750	2 2 2
0.12	1	1	1	0.12	0.1200000	50.00000	1 1 1
0.12	1	2	1	0.12	0.1200000	446.00000	1 2 1
0.12	1	1	2	0.12	0.3464102	38.67949	1 1 2
0.12	1	2	2	0.12	0.3464102	332.79492	1 2 2
0.12	2	1	1	0.12	0.1200000	27.20000	2 1 1
0.12	2	2	1	0.12	0.1200000	185.60000	2 2 1
0.12	2	1	2	0.12	0.3464102	22.67180	2 1 2
0.12	2	2	2	0.12	0.3464102	140.31797	2 2 2
0.13	1	1	1	0.13	0.1300000	50.00000	1 1 1
0.13	1	2	1	0.13	0.1300000	441.50000	1 2 1
0.13	1	1	2	0.13	0.3605551	38.47224	1 1 2
0.13	1	2	2	0.13	0.3605551	326.22244	1 2 2
0.13	2	1	1	0.13	0.1300000	27.80000	2 1 1
0.13	2	2	1	0.13	0.1300000	184.40000	2 2 1
0.13	2	1	2	0.13	0.3605551	23.18890	2 1 2
0.13	2	2	2	0.13	0.3605551	138.28897	2 2 2
0.14	1	1	1	0.14	0.1400000	50.00000	1 1 1
0.14	1	2	1	0.14	0.1400000	437.00000	1 2 1
0.14	1	1	2	0.14	0.3741657	38.29171	1 1 2
0.14	1	2	2	0.14	0.3741657	319.91713	1 2 2
0.14	2	1	1	0.14	0.1400000	28.40000	2 1 1
0.14	2	2	1	0.14	0.1400000	183.20000	2 2 1
0.14	2	1	2	0.14	0.3741657	23.71669	2 1 2
0.14	2	2	2	0.14	0.3741657	136.36685	2 2 2
0.15	1	1	1	0.15	0.1500000	50.00000	1 1 1
0.15	1	2	1	0.15	0.1500000	432.50000	1 2 1
0.15	1	1	2	0.15	0.3872983	38.13508	1 1 2
0.15	1	2	2	0.15	0.3872983	313.85083	1 2 2

x_vec	A_vec	C_vec	R_vec	FPR_vec	TPR_vec	G_vec	design_id
0.15	2	1	1	0.15	0.1500000	29.00000	2 1 1
0.15	2	2	1	0.15	0.1500000	182.00000	2 2 1
0.15	2	1	2	0.15	0.3872983	24.25403	2 1 2
0.15	2	2	2	0.15	0.3872983	134.54033	2 2 2
0.16	1	1	1	0.16	0.1600000	50.00000	1 1 1
0.16	1	2	1	0.16	0.1600000	428.00000	1 2 1
0.16	1	1	2	0.16	0.4000000	38.00000	1 1 2
0.16	1	2	2	0.16	0.4000000	308.00000	1 2 2
0.16	2	1	1	0.16	0.1600000	29.60000	2 1 1
0.16	2	2	1	0.16	0.1600000	180.80000	2 2 1
0.16	2	1	2	0.16	0.4000000	24.80000	2 1 2
0.16	2	2	2	0.16	0.4000000	132.80000	2 2 2
0.17	1	1	1	0.17	0.1700000	50.00000	1 1 1
0.17	1	2	1	0.17	0.1700000	423.50000	1 2 1
0.17	1	1	2	0.17	0.4123106	37.88447	1 1 2
0.17	1	2	2	0.17	0.4123106	302.34472	1 2 2
0.17	2	1	1	0.17	0.1700000	30.20000	2 1 1
0.17	2	2	1	0.17	0.1700000	179.60000	2 2 1
0.17	2	1	2	0.17	0.4123106	25.35379	2 1 2
0.17	2	2	2	0.17	0.4123106	131.13789	2 2 2
0.18	1	1	1	0.18	0.1800000	50.00000	1 1 1
0.18	1	2	1	0.18	0.1800000	419.00000	1 2 1
0.18	1	1	2	0.18	0.4242641	37.78680	1 1 2
0.18	1	2	2	0.18	0.4242641	296.86797	1 2 2
0.18	2	1	1	0.18	0.1800000	30.80000	2 1 1
0.18	2	2	1	0.18	0.1800000	178.40000	2 2 1
0.18	2	1	2	0.18	0.4242641	25.91472	2 1 2
0.18	2	2	2	0.18	0.4242641	129.54719	2 2 2
0.19	1	1	1	0.19	0.1900000	50.00000	1 1 1
0.19	1	2	1	0.19	0.1900000	414.50000	1 2 1
0.19	1	1	2	0.19	0.4358899	37.70551	1 1 2
0.19	1	2	2	0.19	0.4358899	291.55505	1 2 2
0.19	2	1	1	0.19	0.1900000	31.40000	2 1 1
0.19	2	2	1	0.19	0.1900000	177.20000	2 2 1
0.19	2	1	2	0.19	0.4358899	26.48220	2 1 2
0.19	2	2	2	0.19	0.4358899	128.02202	2 2 2
0.20	1	1	1	0.20	0.2000000	50.00000	1 1 1
0.20	1	2	1	0.20	0.2000000	410.00000	1 2 1
0.20	1	1	2	0.20	0.4472136	37.63932	1 1 2
0.20	1	2	2	0.20	0.4472136	286.39320	1 2 2
0.20	2	1	1	0.20	0.2000000	32.00000	2 1 1
0.20	2	2	1	0.20	0.2000000	176.00000	2 2 1
0.20	2	1	2	0.20	0.4472136	27.05573	2 1 2
0.20	2	2	2	0.20	0.4472136	126.55728	2 2 2
0.21	1	1	1	0.21	0.2100000	50.00000	1 1 1
0.21	1	2	1	0.21	0.2100000	405.50000	1 2 1
0.21	1	1	2	0.21	0.4582576	37.58712	1 1 2
0.21	1	2	2	0.21	0.4582576	281.37122	1 2 2

x_vec	A_vec	C_vec	R_vec	FPR_vec	TPR_vec	G_vec	design_id
0.21	2	1	1	0.21	0.2100000	32.60000	2 1 1
0.21	2	2	1	0.21	0.2100000	174.80000	2 2 1
0.21	2	1	2	0.21	0.4582576	27.63485	2 1 2
0.21	2	2	2	0.21	0.4582576	125.14849	2 2 2
0.22	1	1	1	0.22	0.2200000	50.00000	1 1 1
0.22	1	2	1	0.22	0.2200000	401.00000	1 2 1
0.22	1	1	2	0.22	0.4690416	37.54792	1 1 2
0.22	1	2	2	0.22	0.4690416	276.47921	1 2 2
0.22	2	1	1	0.22	0.2200000	33.20000	2 1 1
0.22	2	2	1	0.22	0.2200000	173.60000	2 2 1
0.22	2	1	2	0.22	0.4690416	28.21917	2 1 2
0.22	2	2	2	0.22	0.4690416	123.79168	2 2 2
0.23	1	1	1	0.23	0.2300000	50.00000	1 1 1
0.23	1	2	1	0.23	0.2300000	396.50000	1 2 1
0.23	1	1	2	0.23	0.4795832	37.52084	1 1 2
0.23	1	2	2	0.23	0.4795832	271.70842	1 2 2
0.23	2	1	1	0.23	0.2300000	33.80000	2 1 1
0.23	2	2	1	0.23	0.2300000	172.40000	2 2 1
0.23	2	1	2	0.23	0.4795832	28.80834	2 1 2
0.23	2	2	2	0.23	0.4795832	122.48337	2 2 2
0.24	1	1	1	0.24	0.2400000	50.00000	1 1 1
0.24	1	2	1	0.24	0.2400000	392.00000	1 2 1
0.24	1	1	2	0.24	0.4898979	37.50510	1 1 2
0.24	1	2	2	0.24	0.4898979	267.05103	1 2 2
0.24	2	1	1	0.24	0.2400000	34.40000	2 1 1
0.24	2	2	1	0.24	0.2400000	171.20000	2 2 1
0.24	2	1	2	0.24	0.4898979	29.40204	2 1 2
0.24	2	2	2	0.24	0.4898979	121.22041	2 2 2
0.25	1	1	1	0.25	0.2500000	50.00000	1 1 1
0.25	1	2	1	0.25	0.2500000	387.50000	1 2 1
0.25	1	1	2	0.25	0.5000000	37.50000	1 1 2
0.25	1	2	2	0.25	0.5000000	262.50000	1 2 2
0.25	2	1	1	0.25	0.2500000	35.00000	2 1 1
0.25	2	2	1	0.25	0.2500000	170.00000	2 2 1
0.25	2	1	2	0.25	0.5000000	30.00000	2 1 2
0.25	2	2	2	0.25	0.5000000	120.00000	2 2 2
0.26	1	1	1	0.26	0.2600000	50.00000	1 1 1
0.26	1	2	1	0.26	0.2600000	383.00000	1 2 1
0.26	1	1	2	0.26	0.5099020	37.50490	1 1 2
0.26	1	2	2	0.26	0.5099020	258.04902	1 2 2
0.26	2	1	1	0.26	0.2600000	35.60000	2 1 1
0.26	2	2	1	0.26	0.2600000	168.80000	2 2 1
0.26	2	1	2	0.26	0.5099020	30.60196	2 1 2
0.26	2	2	2	0.26	0.5099020	118.81961	2 2 2
0.27	1	1	1	0.27	0.2700000	50.00000	1 1 1
0.27	1	2	1	0.27	0.2700000	378.50000	1 2 1
0.27	1	1	2	0.27	0.5196152	37.51924	1 1 2
0.27	1	2	2	0.27	0.5196152	253.69238	1 2 2

x_vec	A_vec	C_vec	R_vec	FPR_vec	TPR_vec	G_vec	design_id
0.27	2	1	1	0.27	0.2700000	36.20000	2 1 1
0.27	2	2	1	0.27	0.2700000	167.60000	2 2 1
0.27	2	1	2	0.27	0.5196152	31.20770	2 1 2
0.27	2	2	2	0.27	0.5196152	117.67695	2 2 2
0.28	1	1	1	0.28	0.2800000	50.00000	1 1 1
0.28	1	2	1	0.28	0.2800000	374.00000	1 2 1
0.28	1	1	2	0.28	0.5291503	37.54249	1 1 2
0.28	1	2	2	0.28	0.5291503	249.42487	1 2 2
0.28	2	1	1	0.28	0.2800000	36.80000	2 1 1
0.28	2	2	1	0.28	0.2800000	166.40000	2 2 1
0.28	2	1	2	0.28	0.5291503	31.81699	2 1 2
0.28	2	2	2	0.28	0.5291503	116.56995	2 2 2
0.29	1	1	1	0.29	0.2900000	50.00000	1 1 1
0.29	1	2	1	0.29	0.2900000	369.50000	1 2 1
0.29	1	1	2	0.29	0.5385165	37.57418	1 1 2
0.29	1	2	2	0.29	0.5385165	245.24176	1 2 2
0.29	2	1	1	0.29	0.2900000	37.40000	2 1 1
0.29	2	2	1	0.29	0.2900000	165.20000	2 2 1
0.29	2	1	2	0.29	0.5385165	32.42967	2 1 2
0.29	2	2	2	0.29	0.5385165	115.49670	2 2 2
0.30	1	1	1	0.30	0.3000000	50.00000	1 1 1
0.30	1	2	1	0.30	0.3000000	365.00000	1 2 1
0.30	1	1	2	0.30	0.5477226	37.61387	1 1 2
0.30	1	2	2	0.30	0.5477226	241.13872	1 2 2
0.30	2	1	1	0.30	0.3000000	38.00000	2 1 1
0.30	2	2	1	0.30	0.3000000	164.00000	2 2 1
0.30	2	1	2	0.30	0.5477226	33.04555	2 1 2
0.30	2	2	2	0.30	0.5477226	114.45549	2 2 2
0.31	1	1	1	0.31	0.3100000	50.00000	1 1 1
0.31	1	2	1	0.31	0.3100000	360.50000	1 2 1
0.31	1	1	2	0.31	0.5567764	37.66118	1 1 2
0.31	1	2	2	0.31	0.5567764	237.11178	1 2 2
0.31	2	1	1	0.31	0.3100000	38.60000	2 1 1
0.31	2	2	1	0.31	0.3100000	162.80000	2 2 1
0.31	2	1	2	0.31	0.5567764	33.66447	2 1 2
0.31	2	2	2	0.31	0.5567764	113.44471	2 2 2
0.32	1	1	1	0.32	0.3200000	50.00000	1 1 1
0.32	1	2	1	0.32	0.3200000	356.00000	1 2 1
0.32	1	1	2	0.32	0.5656854	37.71573	1 1 2
0.32	1	2	2	0.32	0.5656854	233.15729	1 2 2
0.32	2	1	1	0.32	0.3200000	39.20000	2 1 1
0.32	2	2	1	0.32	0.3200000	161.60000	2 2 1
0.32	2	1	2	0.32	0.5656854	34.28629	2 1 2
0.32	2	2	2	0.32	0.5656854	112.46291	2 2 2
0.33	1	1	1	0.33	0.3300000	50.00000	1 1 1
0.33	1	2	1	0.33	0.3300000	351.50000	1 2 1
0.33	1	1	2	0.33	0.5744563	37.77719	1 1 2
0.33	1	2	2	0.33	0.5744563	229.27187	1 2 2

x_vec	A_vec	C_vec	R_vec	FPR_vec	TPR_vec	G_vec	design_id
0.33	2	1	1	0.33	0.3300000	39.80000	2 1 1
0.33	2	2	1	0.33	0.3300000	160.40000	2 2 1
0.33	2	1	2	0.33	0.5744563	34.91087	2 1 2
0.33	2	2	2	0.33	0.5744563	111.50875	2 2 2
0.34	1	1	1	0.34	0.3400000	50.00000	1 1 1
0.34	1	2	1	0.34	0.3400000	347.00000	1 2 1
0.34	1	1	2	0.34	0.5830952	37.84524	1 1 2
0.34	1	2	2	0.34	0.5830952	225.45241	1 2 2
0.34	2	1	1	0.34	0.3400000	40.40000	2 1 1
0.34	2	2	1	0.34	0.3400000	159.20000	2 2 1
0.34	2	1	2	0.34	0.5830952	35.53810	2 1 2
0.34	2	2	2	0.34	0.5830952	110.58096	2 2 2
0.35	1	1	1	0.35	0.3500000	50.00000	1 1 1
0.35	1	2	1	0.35	0.3500000	342.50000	1 2 1
0.35	1	1	2	0.35	0.5916080	37.91960	1 1 2
0.35	1	2	2	0.35	0.5916080	221.69601	1 2 2
0.35	2	1	1	0.35	0.3500000	41.00000	2 1 1
0.35	2	2	1	0.35	0.3500000	158.00000	2 2 1
0.35	2	1	2	0.35	0.5916080	36.16784	2 1 2
0.35	2	2	2	0.35	0.5916080	109.67840	2 2 2
0.36	1	1	1	0.36	0.3600000	50.00000	1 1 1
0.36	1	2	1	0.36	0.3600000	338.00000	1 2 1
0.36	1	1	2	0.36	0.6000000	38.00000	1 1 2
0.36	1	2	2	0.36	0.6000000	218.00000	1 2 2
0.36	2	1	1	0.36	0.3600000	41.60000	2 1 1
0.36	2	2	1	0.36	0.3600000	156.80000	2 2 1
0.36	2	1	2	0.36	0.6000000	36.80000	2 1 2
0.36	2	2	2	0.36	0.6000000	108.80000	2 2 2
0.37	1	1	1	0.37	0.3700000	50.00000	1 1 1
0.37	1	2	1	0.37	0.3700000	333.50000	1 2 1
0.37	1	1	2	0.37	0.6082763	38.08619	1 1 2
0.37	1	2	2	0.37	0.6082763	214.36187	1 2 2
0.37	2	1	1	0.37	0.3700000	42.20000	2 1 1
0.37	2	2	1	0.37	0.3700000	155.60000	2 2 1
0.37	2	1	2	0.37	0.6082763	37.43447	2 1 2
0.37	2	2	2	0.37	0.6082763	107.94475	2 2 2
0.38	1	1	1	0.38	0.3800000	50.00000	1 1 1
0.38	1	2	1	0.38	0.3800000	329.00000	1 2 1
0.38	1	1	2	0.38	0.6164414	38.17793	1 1 2
0.38	1	2	2	0.38	0.6164414	210.77930	1 2 2
0.38	2	1	1	0.38	0.3800000	42.80000	2 1 1
0.38	2	2	1	0.38	0.3800000	154.40000	2 2 1
0.38	2	1	2	0.38	0.6164414	38.07117	2 1 2
0.38	2	2	2	0.38	0.6164414	107.11172	2 2 2
0.39	1	1	1	0.39	0.3900000	50.00000	1 1 1
0.39	1	2	1	0.39	0.3900000	324.50000	1 2 1
0.39	1	1	2	0.39	0.6244998	38.27501	1 1 2
0.39	1	2	2	0.39	0.6244998	207.25010	1 2 2

x_vec	A_vec	C_vec	R_vec	FPR_vec	TPR_vec	G_vec	design_id
0.39	2	1	1	0.39	0.3900000	43.40000	2 1 1
0.39	2	2	1	0.39	0.3900000	153.20000	2 2 1
0.39	2	1	2	0.39	0.6244998	38.71000	2 1 2
0.39	2	2	2	0.39	0.6244998	106.30004	2 2 2
0.40	1	1	1	0.40	0.4000000	50.00000	1 1 1
0.40	1	2	1	0.40	0.4000000	320.00000	1 2 1
0.40	1	1	2	0.40	0.6324555	38.37722	1 1 2
0.40	1	2	2	0.40	0.6324555	203.77223	1 2 2
0.40	2	1	1	0.40	0.4000000	44.00000	2 1 1
0.40	2	2	1	0.40	0.4000000	152.00000	2 2 1
0.40	2	1	2	0.40	0.6324555	39.35089	2 1 2
0.40	2	2	2	0.40	0.6324555	105.50889	2 2 2
0.41	1	1	1	0.41	0.4100000	50.00000	1 1 1
0.41	1	2	1	0.41	0.4100000	315.50000	1 2 1
0.41	1	1	2	0.41	0.6403124	38.48438	1 1 2
0.41	1	2	2	0.41	0.6403124	200.34379	1 2 2
0.41	2	1	1	0.41	0.4100000	44.60000	2 1 1
0.41	2	2	1	0.41	0.4100000	150.80000	2 2 1
0.41	2	1	2	0.41	0.6403124	39.99375	2 1 2
0.41	2	2	2	0.41	0.6403124	104.73752	2 2 2
0.42	1	1	1	0.42	0.4200000	50.00000	1 1 1
0.42	1	2	1	0.42	0.4200000	311.00000	1 2 1
0.42	1	1	2	0.42	0.6480741	38.59630	1 1 2
0.42	1	2	2	0.42	0.6480741	196.96297	1 2 2
0.42	2	1	1	0.42	0.4200000	45.20000	2 1 1
0.42	2	2	1	0.42	0.4200000	149.60000	2 2 1
0.42	2	1	2	0.42	0.6480741	40.63852	2 1 2
0.42	2	2	2	0.42	0.6480741	103.98519	2 2 2
0.43	1	1	1	0.43	0.4300000	50.00000	1 1 1
0.43	1	2	1	0.43	0.4300000	306.50000	1 2 1
0.43	1	1	2	0.43	0.6557439	38.71281	1 1 2
0.43	1	2	2	0.43	0.6557439	193.62807	1 2 2
0.43	2	1	1	0.43	0.4300000	45.80000	2 1 1
0.43	2	2	1	0.43	0.4300000	148.40000	2 2 1
0.43	2	1	2	0.43	0.6557439	41.28512	2 1 2
0.43	2	2	2	0.43	0.6557439	103.25123	2 2 2
0.44	1	1	1	0.44	0.4400000	50.00000	1 1 1
0.44	1	2	1	0.44	0.4400000	302.00000	1 2 1
0.44	1	1	2	0.44	0.6633250	38.83375	1 1 2
0.44	1	2	2	0.44	0.6633250	190.33752	1 2 2
0.44	2	1	1	0.44	0.4400000	46.40000	2 1 1
0.44	2	2	1	0.44	0.4400000	147.20000	2 2 1
0.44	2	1	2	0.44	0.6633250	41.93350	2 1 2
0.44	2	2	2	0.44	0.6633250	102.53501	2 2 2
0.45	1	1	1	0.45	0.4500000	50.00000	1 1 1
0.45	1	2	1	0.45	0.4500000	297.50000	1 2 1
0.45	1	1	2	0.45	0.6708204	38.95898	1 1 2
0.45	1	2	2	0.45	0.6708204	187.08980	1 2 2

x_vec	A_vec	C_vec	R_vec	FPR_vec	TPR_vec	G_vec	design_id
0.45	2	1	1	0.45	0.4500000	47.00000	2 1 1
0.45	2	2	1	0.45	0.4500000	146.00000	2 2 1
0.45	2	1	2	0.45	0.6708204	42.58359	2 1 2
0.45	2	2	2	0.45	0.6708204	101.83592	2 2 2
0.46	1	1	1	0.46	0.4600000	50.00000	1 1 1
0.46	1	2	1	0.46	0.4600000	293.00000	1 2 1
0.46	1	1	2	0.46	0.6782330	39.08835	1 1 2
0.46	1	2	2	0.46	0.6782330	183.88350	1 2 2
0.46	2	1	1	0.46	0.4600000	47.60000	2 1 1
0.46	2	2	1	0.46	0.4600000	144.80000	2 2 1
0.46	2	1	2	0.46	0.6782330	43.23534	2 1 2
0.46	2	2	2	0.46	0.6782330	101.15340	2 2 2
0.47	1	1	1	0.47	0.4700000	50.00000	1 1 1
0.47	1	2	1	0.47	0.4700000	288.50000	1 2 1
0.47	1	1	2	0.47	0.6855655	39.22173	1 1 2
0.47	1	2	2	0.47	0.6855655	180.71727	1 2 2
0.47	2	1	1	0.47	0.4700000	48.20000	2 1 1
0.47	2	2	1	0.47	0.4700000	143.60000	2 2 1
0.47	2	1	2	0.47	0.6855655	43.88869	2 1 2
0.47	2	2	2	0.47	0.6855655	100.48691	2 2 2
0.48	1	1	1	0.48	0.4800000	50.00000	1 1 1
0.48	1	2	1	0.48	0.4800000	284.00000	1 2 1
0.48	1	1	2	0.48	0.6928203	39.35898	1 1 2
0.48	1	2	2	0.48	0.6928203	177.58984	1 2 2
0.48	2	1	1	0.48	0.4800000	48.80000	2 1 1
0.48	2	2	1	0.48	0.4800000	142.40000	2 2 1
0.48	2	1	2	0.48	0.6928203	44.54359	2 1 2
0.48	2	2	2	0.48	0.6928203	99.83594	2 2 2
0.49	1	1	1	0.49	0.4900000	50.00000	1 1 1
0.49	1	2	1	0.49	0.4900000	279.50000	1 2 1
0.49	1	1	2	0.49	0.7000000	39.50000	1 1 2
0.49	1	2	2	0.49	0.7000000	174.50000	1 2 2
0.49	2	1	1	0.49	0.4900000	49.40000	2 1 1
0.49	2	2	1	0.49	0.4900000	141.20000	2 2 1
0.49	2	1	2	0.49	0.7000000	45.20000	2 1 2
0.49	2	2	2	0.49	0.7000000	99.20000	2 2 2
0.50	1	1	1	0.50	0.5000000	50.00000	1 1 1
0.50	1	2	1	0.50	0.5000000	275.00000	1 2 1
0.50	1	1	2	0.50	0.7071068	39.64466	1 1 2
0.50	1	2	2	0.50	0.7071068	171.44661	1 2 2
0.50	2	1	1	0.50	0.5000000	50.00000	2 1 1
0.50	2	2	1	0.50	0.5000000	140.00000	2 2 1
0.50	2	1	2	0.50	0.7071068	45.85786	2 1 2
0.50	2	2	2	0.50	0.7071068	98.57864	2 2 2
0.51	1	1	1	0.51	0.5100000	50.00000	1 1 1
0.51	1	2	1	0.51	0.5100000	270.50000	1 2 1
0.51	1	1	2	0.51	0.7141428	39.79286	1 1 2
0.51	1	2	2	0.51	0.7141428	168.42858	1 2 2

x_vec	A_vec	C_vec	R_vec	FPR_vec	TPR_vec	G_vec	design_id
0.51	2	1	1	0.51	0.5100000	50.60000	2 1 1
0.51	2	2	1	0.51	0.5100000	138.80000	2 2 1
0.51	2	1	2	0.51	0.7141428	46.51714	2 1 2
0.51	2	2	2	0.51	0.7141428	97.97143	2 2 2
0.52	1	1	1	0.52	0.5200000	50.00000	1 1 1
0.52	1	2	1	0.52	0.5200000	266.00000	1 2 1
0.52	1	1	2	0.52	0.7211103	39.94449	1 1 2
0.52	1	2	2	0.52	0.7211103	165.44487	1 2 2
0.52	2	1	1	0.52	0.5200000	51.20000	2 1 1
0.52	2	2	1	0.52	0.5200000	137.60000	2 2 1
0.52	2	1	2	0.52	0.7211103	47.17779	2 1 2
0.52	2	2	2	0.52	0.7211103	97.37795	2 2 2
0.53	1	1	1	0.53	0.5300000	50.00000	1 1 1
0.53	1	2	1	0.53	0.5300000	261.50000	1 2 1
0.53	1	1	2	0.53	0.7280110	40.09945	1 1 2
0.53	1	2	2	0.53	0.7280110	162.49451	1 2 2
0.53	2	1	1	0.53	0.5300000	51.80000	2 1 1
0.53	2	2	1	0.53	0.5300000	136.40000	2 2 1
0.53	2	1	2	0.53	0.7280110	47.83978	2 1 2
0.53	2	2	2	0.53	0.7280110	96.79780	2 2 2
0.54	1	1	1	0.54	0.5400000	50.00000	1 1 1
0.54	1	2	1	0.54	0.5400000	257.00000	1 2 1
0.54	1	1	2	0.54	0.7348469	40.25765	1 1 2
0.54	1	2	2	0.54	0.7348469	159.57654	1 2 2
0.54	2	1	1	0.54	0.5400000	52.40000	2 1 1
0.54	2	2	1	0.54	0.5400000	135.20000	2 2 1
0.54	2	1	2	0.54	0.7348469	48.50306	2 1 2
0.54	2	2	2	0.54	0.7348469	96.23062	2 2 2
0.55	1	1	1	0.55	0.5500000	50.00000	1 1 1
0.55	1	2	1	0.55	0.5500000	252.50000	1 2 1
0.55	1	1	2	0.55	0.7416198	40.41901	1 1 2
0.55	1	2	2	0.55	0.7416198	156.69008	1 2 2
0.55	2	1	1	0.55	0.5500000	53.00000	2 1 1
0.55	2	2	1	0.55	0.5500000	134.00000	2 2 1
0.55	2	1	2	0.55	0.7416198	49.16760	2 1 2
0.55	2	2	2	0.55	0.7416198	95.67603	2 2 2
0.56	1	1	1	0.56	0.5600000	50.00000	1 1 1
0.56	1	2	1	0.56	0.5600000	248.00000	1 2 1
0.56	1	1	2	0.56	0.7483315	40.58343	1 1 2
0.56	1	2	2	0.56	0.7483315	153.83426	1 2 2
0.56	2	1	1	0.56	0.5600000	53.60000	2 1 1
0.56	2	2	1	0.56	0.5600000	132.80000	2 2 1
0.56	2	1	2	0.56	0.7483315	49.83337	2 1 2
0.56	2	2	2	0.56	0.7483315	95.13370	2 2 2
0.57	1	1	1	0.57	0.5700000	50.00000	1 1 1
0.57	1	2	1	0.57	0.5700000	243.50000	1 2 1
0.57	1	1	2	0.57	0.7549834	40.75083	1 1 2
0.57	1	2	2	0.57	0.7549834	151.00828	1 2 2

x_vec	A_vec	C_vec	R_vec	FPR_vec	TPR_vec	G_vec	design_id
0.57	2	1	1	0.57	0.5700000	54.20000	2 1 1
0.57	2	2	1	0.57	0.5700000	131.60000	2 2 1
0.57	2	1	2	0.57	0.7549834	50.50033	2 1 2
0.57	2	2	2	0.57	0.7549834	94.60331	2 2 2
0.58	1	1	1	0.58	0.5800000	50.00000	1 1 1
0.58	1	2	1	0.58	0.5800000	239.00000	1 2 1
0.58	1	1	2	0.58	0.7615773	40.92113	1 1 2
0.58	1	2	2	0.58	0.7615773	148.21134	1 2 2
0.58	2	1	1	0.58	0.5800000	54.80000	2 1 1
0.58	2	2	1	0.58	0.5800000	130.40000	2 2 1
0.58	2	1	2	0.58	0.7615773	51.16845	2 1 2
0.58	2	2	2	0.58	0.7615773	94.08454	2 2 2
0.59	1	1	1	0.59	0.5900000	50.00000	1 1 1
0.59	1	2	1	0.59	0.5900000	234.50000	1 2 1
0.59	1	1	2	0.59	0.7681146	41.09427	1 1 2
0.59	1	2	2	0.59	0.7681146	145.44271	1 2 2
0.59	2	1	1	0.59	0.5900000	55.40000	2 1 1
0.59	2	2	1	0.59	0.5900000	129.20000	2 2 1
0.59	2	1	2	0.59	0.7681146	51.83771	2 1 2
0.59	2	2	2	0.59	0.7681146	93.57708	2 2 2
0.60	1	1	1	0.60	0.6000000	50.00000	1 1 1
0.60	1	2	1	0.60	0.6000000	230.00000	1 2 1
0.60	1	1	2	0.60	0.7745967	41.27017	1 1 2
0.60	1	2	2	0.60	0.7745967	142.70167	1 2 2
0.60	2	1	1	0.60	0.6000000	56.00000	2 1 1
0.60	2	2	1	0.60	0.6000000	128.00000	2 2 1
0.60	2	1	2	0.60	0.7745967	52.50807	2 1 2
0.60	2	2	2	0.60	0.7745967	93.08067	2 2 2
0.61	1	1	1	0.61	0.6100000	50.00000	1 1 1
0.61	1	2	1	0.61	0.6100000	225.50000	1 2 1
0.61	1	1	2	0.61	0.7810250	41.44875	1 1 2
0.61	1	2	2	0.61	0.7810250	139.98752	1 2 2
0.61	2	1	1	0.61	0.6100000	56.60000	2 1 1
0.61	2	2	1	0.61	0.6100000	126.80000	2 2 1
0.61	2	1	2	0.61	0.7810250	53.17950	2 1 2
0.61	2	2	2	0.61	0.7810250	92.59501	2 2 2
0.62	1	1	1	0.62	0.6200000	50.00000	1 1 1
0.62	1	2	1	0.62	0.6200000	221.00000	1 2 1
0.62	1	1	2	0.62	0.7874008	41.62996	1 1 2
0.62	1	2	2	0.62	0.7874008	137.29961	1 2 2
0.62	2	1	1	0.62	0.6200000	57.20000	2 1 1
0.62	2	2	1	0.62	0.6200000	125.60000	2 2 1
0.62	2	1	2	0.62	0.7874008	53.85198	2 1 2
0.62	2	2	2	0.62	0.7874008	92.11984	2 2 2
0.63	1	1	1	0.63	0.6300000	50.00000	1 1 1
0.63	1	2	1	0.63	0.6300000	216.50000	1 2 1
0.63	1	1	2	0.63	0.7937254	41.81373	1 1 2
0.63	1	2	2	0.63	0.7937254	134.63730	1 2 2

x_vec	A_vec	C_vec	R_vec	FPR_vec	TPR_vec	G_vec	design_id
0.63	2	1	1	0.63	0.6300000	57.80000	2 1 1
0.63	2	2	1	0.63	0.6300000	124.40000	2 2 1
0.63	2	1	2	0.63	0.7937254	54.52549	2 1 2
0.63	2	2	2	0.63	0.7937254	91.65492	2 2 2
0.64	1	1	1	0.64	0.6400000	50.00000	1 1 1
0.64	1	2	1	0.64	0.6400000	212.00000	1 2 1
0.64	1	1	2	0.64	0.8000000	42.00000	1 1 2
0.64	1	2	2	0.64	0.8000000	132.00000	1 2 2
0.64	2	1	1	0.64	0.6400000	58.40000	2 1 1
0.64	2	2	1	0.64	0.6400000	123.20000	2 2 1
0.64	2	1	2	0.64	0.8000000	55.20000	2 1 2
0.64	2	2	2	0.64	0.8000000	91.20000	2 2 2
0.65	1	1	1	0.65	0.6500000	50.00000	1 1 1
0.65	1	2	1	0.65	0.6500000	207.50000	1 2 1
0.65	1	1	2	0.65	0.8062258	42.18871	1 1 2
0.65	1	2	2	0.65	0.8062258	129.38711	1 2 2
0.65	2	1	1	0.65	0.6500000	59.00000	2 1 1
0.65	2	2	1	0.65	0.6500000	122.00000	2 2 1
0.65	2	1	2	0.65	0.8062258	55.87548	2 1 2
0.65	2	2	2	0.65	0.8062258	90.75485	2 2 2
0.66	1	1	1	0.66	0.6600000	50.00000	1 1 1
0.66	1	2	1	0.66	0.6600000	203.00000	1 2 1
0.66	1	1	2	0.66	0.8124038	42.37981	1 1 2
0.66	1	2	2	0.66	0.8124038	126.79808	1 2 2
0.66	2	1	1	0.66	0.6600000	59.60000	2 1 1
0.66	2	2	1	0.66	0.6600000	120.80000	2 2 1
0.66	2	1	2	0.66	0.8124038	56.55192	2 1 2
0.66	2	2	2	0.66	0.8124038	90.31923	2 2 2
0.67	1	1	1	0.67	0.6700000	50.00000	1 1 1
0.67	1	2	1	0.67	0.6700000	198.50000	1 2 1
0.67	1	1	2	0.67	0.8185353	42.57324	1 1 2
0.67	1	2	2	0.67	0.8185353	124.23236	1 2 2
0.67	2	1	1	0.67	0.6700000	60.20000	2 1 1
0.67	2	2	1	0.67	0.6700000	119.60000	2 2 1
0.67	2	1	2	0.67	0.8185353	57.22929	2 1 2
0.67	2	2	2	0.67	0.8185353	89.89294	2 2 2
0.68	1	1	1	0.68	0.6800000	50.00000	1 1 1
0.68	1	2	1	0.68	0.6800000	194.00000	1 2 1
0.68	1	1	2	0.68	0.8246211	42.76894	1 1 2
0.68	1	2	2	0.68	0.8246211	121.68944	1 2 2
0.68	2	1	1	0.68	0.6800000	60.80000	2 1 1
0.68	2	2	1	0.68	0.6800000	118.40000	2 2 1
0.68	2	1	2	0.68	0.8246211	57.90758	2 1 2
0.68	2	2	2	0.68	0.8246211	89.47577	2 2 2
0.69	1	1	1	0.69	0.6900000	50.00000	1 1 1
0.69	1	2	1	0.69	0.6900000	189.50000	1 2 1
0.69	1	1	2	0.69	0.8306624	42.96688	1 1 2
0.69	1	2	2	0.69	0.8306624	119.16881	1 2 2

x_vec	A_vec	C_vec	R_vec	FPR_vec	TPR_vec	G_vec	design_id
0.69	2	1	1	0.69	0.6900000	61.40000	2 1 1
0.69	2	2	1	0.69	0.6900000	117.20000	2 2 1
0.69	2	1	2	0.69	0.8306624	58.58675	2 1 2
0.69	2	2	2	0.69	0.8306624	89.06752	2 2 2
0.70	1	1	1	0.70	0.7000000	50.00000	1 1 1
0.70	1	2	1	0.70	0.7000000	185.00000	1 2 1
0.70	1	1	2	0.70	0.8366600	43.16700	1 1 2
0.70	1	2	2	0.70	0.8366600	116.66999	1 2 2
0.70	2	1	1	0.70	0.7000000	62.00000	2 1 1
0.70	2	2	1	0.70	0.7000000	116.00000	2 2 1
0.70	2	1	2	0.70	0.8366600	59.26680	2 1 2
0.70	2	2	2	0.70	0.8366600	88.66799	2 2 2
0.71	1	1	1	0.71	0.7100000	50.00000	1 1 1
0.71	1	2	1	0.71	0.7100000	180.50000	1 2 1
0.71	1	1	2	0.71	0.8426150	43.36925	1 1 2
0.71	1	2	2	0.71	0.8426150	114.19251	1 2 2
0.71	2	1	1	0.71	0.7100000	62.60000	2 1 1
0.71	2	2	1	0.71	0.7100000	114.80000	2 2 1
0.71	2	1	2	0.71	0.8426150	59.94770	2 1 2
0.71	2	2	2	0.71	0.8426150	88.27700	2 2 2
0.72	1	1	1	0.72	0.7200000	50.00000	1 1 1
0.72	1	2	1	0.72	0.7200000	176.00000	1 2 1
0.72	1	1	2	0.72	0.8485281	43.57359	1 1 2
0.72	1	2	2	0.72	0.8485281	111.73593	1 2 2
0.72	2	1	1	0.72	0.7200000	63.20000	2 1 1
0.72	2	2	1	0.72	0.7200000	113.60000	2 2 1
0.72	2	1	2	0.72	0.8485281	60.62944	2 1 2
0.72	2	2	2	0.72	0.8485281	87.89437	2 2 2
0.73	1	1	1	0.73	0.7300000	50.00000	1 1 1
0.73	1	2	1	0.73	0.7300000	171.50000	1 2 1
0.73	1	1	2	0.73	0.8544004	43.77998	1 1 2
0.73	1	2	2	0.73	0.8544004	109.29981	1 2 2
0.73	2	1	1	0.73	0.7300000	63.80000	2 1 1
0.73	2	2	1	0.73	0.7300000	112.40000	2 2 1
0.73	2	1	2	0.73	0.8544004	61.31199	2 1 2
0.73	2	2	2	0.73	0.8544004	87.51993	2 2 2
0.74	1	1	1	0.74	0.7400000	50.00000	1 1 1
0.74	1	2	1	0.74	0.7400000	167.00000	1 2 1
0.74	1	1	2	0.74	0.8602325	43.98837	1 1 2
0.74	1	2	2	0.74	0.8602325	106.88374	1 2 2
0.74	2	1	1	0.74	0.7400000	64.40000	2 1 1
0.74	2	2	1	0.74	0.7400000	111.20000	2 2 1
0.74	2	1	2	0.74	0.8602325	61.99535	2 1 2
0.74	2	2	2	0.74	0.8602325	87.15349	2 2 2
0.75	1	1	1	0.75	0.7500000	50.00000	1 1 1
0.75	1	2	1	0.75	0.7500000	162.50000	1 2 1
0.75	1	1	2	0.75	0.8660254	44.19873	1 1 2
0.75	1	2	2	0.75	0.8660254	104.48730	1 2 2

x_vec	A_vec	C_vec	R_vec	FPR_vec	TPR_vec	G_vec	design_id
0.75	2	1	1	0.75	0.7500000	65.00000	2 1 1
0.75	2	2	1	0.75	0.7500000	110.00000	2 2 1
0.75	2	1	2	0.75	0.8660254	62.67949	2 1 2
0.75	2	2	2	0.75	0.8660254	86.79492	2 2 2
0.76	1	1	1	0.76	0.7600000	50.00000	1 1 1
0.76	1	2	1	0.76	0.7600000	158.00000	1 2 1
0.76	1	1	2	0.76	0.8717798	44.41101	1 1 2
0.76	1	2	2	0.76	0.8717798	102.11011	1 2 2
0.76	2	1	1	0.76	0.7600000	65.60000	2 1 1
0.76	2	2	1	0.76	0.7600000	108.80000	2 2 1
0.76	2	1	2	0.76	0.8717798	63.36440	2 1 2
0.76	2	2	2	0.76	0.8717798	86.44404	2 2 2
0.77	1	1	1	0.77	0.7700000	50.00000	1 1 1
0.77	1	2	1	0.77	0.7700000	153.50000	1 2 1
0.77	1	1	2	0.77	0.8774964	44.62518	1 1 2
0.77	1	2	2	0.77	0.8774964	99.75178	1 2 2
0.77	2	1	1	0.77	0.7700000	66.20000	2 1 1
0.77	2	2	1	0.77	0.7700000	107.60000	2 2 1
0.77	2	1	2	0.77	0.8774964	64.05007	2 1 2
0.77	2	2	2	0.77	0.8774964	86.10071	2 2 2
0.78	1	1	1	0.78	0.7800000	50.00000	1 1 1
0.78	1	2	1	0.78	0.7800000	149.00000	1 2 1
0.78	1	1	2	0.78	0.8831761	44.84120	1 1 2
0.78	1	2	2	0.78	0.8831761	97.41196	1 2 2
0.78	2	1	1	0.78	0.7800000	66.80000	2 1 1
0.78	2	2	1	0.78	0.7800000	106.40000	2 2 1
0.78	2	1	2	0.78	0.8831761	64.73648	2 1 2
0.78	2	2	2	0.78	0.8831761	85.76478	2 2 2
0.79	1	1	1	0.79	0.7900000	50.00000	1 1 1
0.79	1	2	1	0.79	0.7900000	144.50000	1 2 1
0.79	1	1	2	0.79	0.8888194	45.05903	1 1 2
0.79	1	2	2	0.79	0.8888194	95.09028	1 2 2
0.79	2	1	1	0.79	0.7900000	67.40000	2 1 1
0.79	2	2	1	0.79	0.7900000	105.20000	2 2 1
0.79	2	1	2	0.79	0.8888194	65.42361	2 1 2
0.79	2	2	2	0.79	0.8888194	85.43611	2 2 2
0.80	1	1	1	0.80	0.8000000	50.00000	1 1 1
0.80	1	2	1	0.80	0.8000000	140.00000	1 2 1
0.80	1	1	2	0.80	0.8944272	45.27864	1 1 2
0.80	1	2	2	0.80	0.8944272	92.78640	1 2 2
0.80	2	1	1	0.80	0.8000000	68.00000	2 1 1
0.80	2	2	1	0.80	0.8000000	104.00000	2 2 1
0.80	2	1	2	0.80	0.8944272	66.11146	2 1 2
0.80	2	2	2	0.80	0.8944272	85.11456	2 2 2
0.81	1	1	1	0.81	0.8100000	50.00000	1 1 1
0.81	1	2	1	0.81	0.8100000	135.50000	1 2 1
0.81	1	1	2	0.81	0.9000000	45.50000	1 1 2
0.81	1	2	2	0.81	0.9000000	90.50000	1 2 2

x_vec	A_vec	C_vec	R_vec	FPR_vec	TPR_vec	G_vec	design_id
0.81	2	1	1	0.81	0.8100000	68.60000	2 1 1
0.81	2	2	1	0.81	0.8100000	102.80000	2 2 1
0.81	2	1	2	0.81	0.9000000	66.80000	2 1 2
0.81	2	2	2	0.81	0.9000000	84.80000	2 2 2
0.82	1	1	1	0.82	0.8200000	50.00000	1 1 1
0.82	1	2	1	0.82	0.8200000	131.00000	1 2 1
0.82	1	1	2	0.82	0.9055385	45.72307	1 1 2
0.82	1	2	2	0.82	0.9055385	88.23074	1 2 2
0.82	2	1	1	0.82	0.8200000	69.20000	2 1 1
0.82	2	2	1	0.82	0.8200000	101.60000	2 2 1
0.82	2	1	2	0.82	0.9055385	67.48923	2 1 2
0.82	2	2	2	0.82	0.9055385	84.49230	2 2 2
0.83	1	1	1	0.83	0.8300000	50.00000	1 1 1
0.83	1	2	1	0.83	0.8300000	126.50000	1 2 1
0.83	1	1	2	0.83	0.9110434	45.94783	1 1 2
0.83	1	2	2	0.83	0.9110434	85.97832	1 2 2
0.83	2	1	1	0.83	0.8300000	69.80000	2 1 1
0.83	2	2	1	0.83	0.8300000	100.40000	2 2 1
0.83	2	1	2	0.83	0.9110434	68.17913	2 1 2
0.83	2	2	2	0.83	0.9110434	84.19133	2 2 2
0.84	1	1	1	0.84	0.8400000	50.00000	1 1 1
0.84	1	2	1	0.84	0.8400000	122.00000	1 2 1
0.84	1	1	2	0.84	0.9165151	46.17424	1 1 2
0.84	1	2	2	0.84	0.9165151	83.74243	1 2 2
0.84	2	1	1	0.84	0.8400000	70.40000	2 1 1
0.84	2	2	1	0.84	0.8400000	99.20000	2 2 1
0.84	2	1	2	0.84	0.9165151	68.86970	2 1 2
0.84	2	2	2	0.84	0.9165151	83.89697	2 2 2
0.85	1	1	1	0.85	0.8500000	50.00000	1 1 1
0.85	1	2	1	0.85	0.8500000	117.50000	1 2 1
0.85	1	1	2	0.85	0.9219544	46.40228	1 1 2
0.85	1	2	2	0.85	0.9219544	81.52278	1 2 2
0.85	2	1	1	0.85	0.8500000	71.00000	2 1 1
0.85	2	2	1	0.85	0.8500000	98.00000	2 2 1
0.85	2	1	2	0.85	0.9219544	69.56091	2 1 2
0.85	2	2	2	0.85	0.9219544	83.60911	2 2 2
0.86	1	1	1	0.86	0.8600000	50.00000	1 1 1
0.86	1	2	1	0.86	0.8600000	113.00000	1 2 1
0.86	1	1	2	0.86	0.9273618	46.63191	1 1 2
0.86	1	2	2	0.86	0.9273618	79.31908	1 2 2
0.86	2	1	1	0.86	0.8600000	71.60000	2 1 1
0.86	2	2	1	0.86	0.8600000	96.80000	2 2 1
0.86	2	1	2	0.86	0.9273618	70.25276	2 1 2
0.86	2	2	2	0.86	0.9273618	83.32763	2 2 2
0.87	1	1	1	0.87	0.8700000	50.00000	1 1 1
0.87	1	2	1	0.87	0.8700000	108.50000	1 2 1
0.87	1	1	2	0.87	0.9327379	46.86310	1 1 2
0.87	1	2	2	0.87	0.9327379	77.13105	1 2 2

x_vec	A_vec	C_vec	R_vec	FPR_vec	TPR_vec	G_vec	design_id
0.87	2	1	1	0.87	0.8700000	72.20000	2 1 1
0.87	2	2	1	0.87	0.8700000	95.60000	2 2 1
0.87	2	1	2	0.87	0.9327379	70.94524	2 1 2
0.87	2	2	2	0.87	0.9327379	83.05242	2 2 2
0.88	1	1	1	0.88	0.8800000	50.00000	1 1 1
0.88	1	2	1	0.88	0.8800000	104.00000	1 2 1
0.88	1	1	2	0.88	0.9380832	47.09584	1 1 2
0.88	1	2	2	0.88	0.9380832	74.95842	1 2 2
0.88	2	1	1	0.88	0.8800000	72.80000	2 1 1
0.88	2	2	1	0.88	0.8800000	94.40000	2 2 1
0.88	2	1	2	0.88	0.9380832	71.63834	2 1 2
0.88	2	2	2	0.88	0.9380832	82.78337	2 2 2
0.89	1	1	1	0.89	0.8900000	50.00000	1 1 1
0.89	1	2	1	0.89	0.8900000	99.50000	1 2 1
0.89	1	1	2	0.89	0.9433981	47.33009	1 1 2
0.89	1	2	2	0.89	0.9433981	72.80094	1 2 2
0.89	2	1	1	0.89	0.8900000	73.40000	2 1 1
0.89	2	2	1	0.89	0.8900000	93.20000	2 2 1
0.89	2	1	2	0.89	0.9433981	72.33204	2 1 2
0.89	2	2	2	0.89	0.9433981	82.52038	2 2 2
0.90	1	1	1	0.90	0.9000000	50.00000	1 1 1
0.90	1	2	1	0.90	0.9000000	95.00000	1 2 1
0.90	1	1	2	0.90	0.9486833	47.56584	1 1 2
0.90	1	2	2	0.90	0.9486833	70.65835	1 2 2
0.90	2	1	1	0.90	0.9000000	74.00000	2 1 1
0.90	2	2	1	0.90	0.9000000	92.00000	2 2 1
0.90	2	1	2	0.90	0.9486833	73.02633	2 1 2
0.90	2	2	2	0.90	0.9486833	82.26334	2 2 2
0.91	1	1	1	0.91	0.9100000	50.00000	1 1 1
0.91	1	2	1	0.91	0.9100000	90.50000	1 2 1
0.91	1	1	2	0.91	0.9539392	47.80304	1 1 2
0.91	1	2	2	0.91	0.9539392	68.53040	1 2 2
0.91	2	1	1	0.91	0.9100000	74.60000	2 1 1
0.91	2	2	1	0.91	0.9100000	90.80000	2 2 1
0.91	2	1	2	0.91	0.9539392	73.72122	2 1 2
0.91	2	2	2	0.91	0.9539392	82.01216	2 2 2
0.92	1	1	1	0.92	0.9200000	50.00000	1 1 1
0.92	1	2	1	0.92	0.9200000	86.00000	1 2 1
0.92	1	1	2	0.92	0.9591663	48.04168	1 1 2
0.92	1	2	2	0.92	0.9591663	66.41685	1 2 2
0.92	2	1	1	0.92	0.9200000	75.20000	2 1 1
0.92	2	2	1	0.92	0.9200000	89.60000	2 2 1
0.92	2	1	2	0.92	0.9591663	74.41667	2 1 2
0.92	2	2	2	0.92	0.9591663	81.76674	2 2 2
0.93	1	1	1	0.93	0.9300000	50.00000	1 1 1
0.93	1	2	1	0.93	0.9300000	81.50000	1 2 1
0.93	1	1	2	0.93	0.9643651	48.28175	1 1 2
0.93	1	2	2	0.93	0.9643651	64.31746	1 2 2

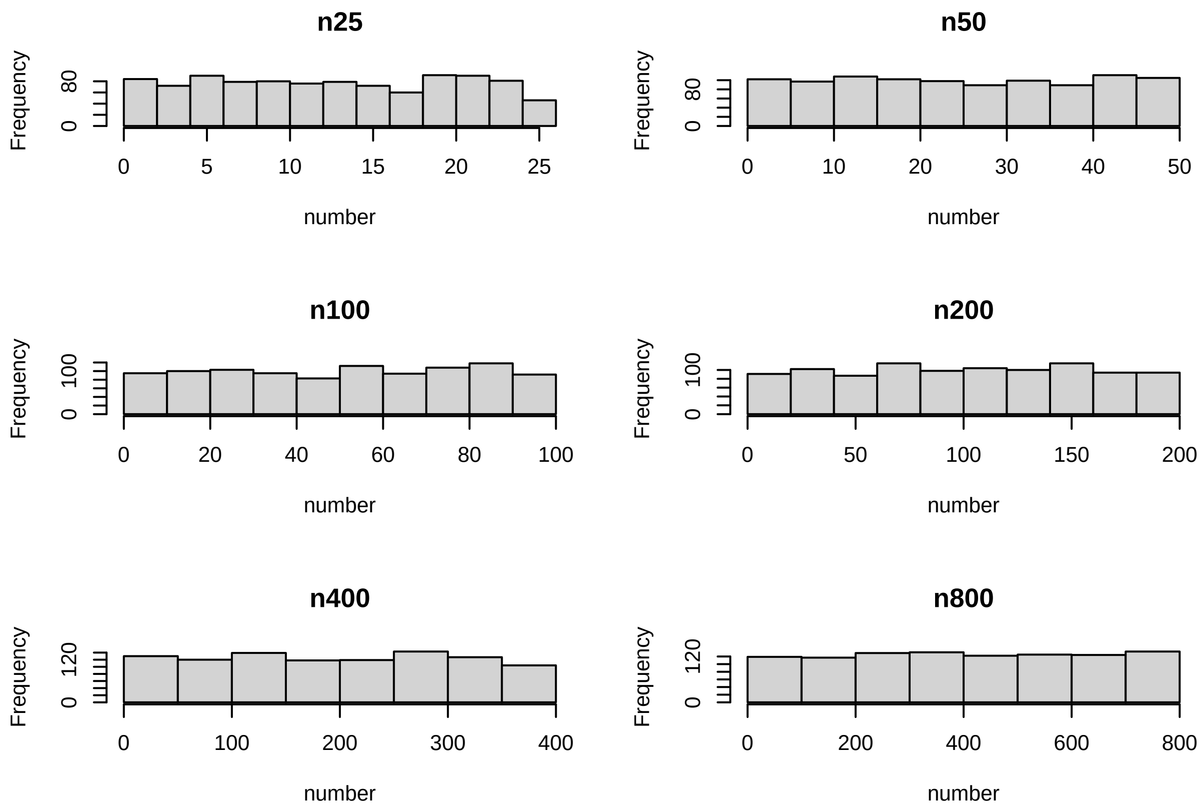
x_vec	A_vec	C_vec	R_vec	FPR_vec	TPR_vec	G_vec	design_id
0.93	2	1	1	0.93	0.9300000	75.80000	2 1 1
0.93	2	2	1	0.93	0.9300000	88.40000	2 2 1
0.93	2	1	2	0.93	0.9643651	75.11270	2 1 2
0.93	2	2	2	0.93	0.9643651	81.52698	2 2 2
0.94	1	1	1	0.94	0.9400000	50.00000	1 1 1
0.94	1	2	1	0.94	0.9400000	77.00000	1 2 1
0.94	1	1	2	0.94	0.9695360	48.52320	1 1 2
0.94	1	2	2	0.94	0.9695360	62.23201	1 2 2
0.94	2	1	1	0.94	0.9400000	76.40000	2 1 1
0.94	2	2	1	0.94	0.9400000	87.20000	2 2 1
0.94	2	1	2	0.94	0.9695360	75.80928	2 1 2
0.94	2	2	2	0.94	0.9695360	81.29281	2 2 2
0.95	1	1	1	0.95	0.9500000	50.00000	1 1 1
0.95	1	2	1	0.95	0.9500000	72.50000	1 2 1
0.95	1	1	2	0.95	0.9746794	48.76603	1 1 2
0.95	1	2	2	0.95	0.9746794	60.16028	1 2 2
0.95	2	1	1	0.95	0.9500000	77.00000	2 1 1
0.95	2	2	1	0.95	0.9500000	86.00000	2 2 1
0.95	2	1	2	0.95	0.9746794	76.50641	2 1 2
0.95	2	2	2	0.95	0.9746794	81.06411	2 2 2
0.96	1	1	1	0.96	0.9600000	50.00000	1 1 1
0.96	1	2	1	0.96	0.9600000	68.00000	1 2 1
0.96	1	1	2	0.96	0.9797959	49.01021	1 1 2
0.96	1	2	2	0.96	0.9797959	58.10205	1 2 2
0.96	2	1	1	0.96	0.9600000	77.60000	2 1 1
0.96	2	2	1	0.96	0.9600000	84.80000	2 2 1
0.96	2	1	2	0.96	0.9797959	77.20408	2 1 2
0.96	2	2	2	0.96	0.9797959	80.84082	2 2 2
0.97	1	1	1	0.97	0.9700000	50.00000	1 1 1
0.97	1	2	1	0.97	0.9700000	63.50000	1 2 1
0.97	1	1	2	0.97	0.9848858	49.25571	1 1 2
0.97	1	2	2	0.97	0.9848858	56.05711	1 2 2
0.97	2	1	1	0.97	0.9700000	78.20000	2 1 1
0.97	2	2	1	0.97	0.9700000	83.60000	2 2 1
0.97	2	1	2	0.97	0.9848858	77.90228	2 1 2
0.97	2	2	2	0.97	0.9848858	80.62284	2 2 2
0.98	1	1	1	0.98	0.9800000	50.00000	1 1 1
0.98	1	2	1	0.98	0.9800000	59.00000	1 2 1
0.98	1	1	2	0.98	0.9899495	49.50253	1 1 2
0.98	1	2	2	0.98	0.9899495	54.02525	1 2 2
0.98	2	1	1	0.98	0.9800000	78.80000	2 1 1
0.98	2	2	1	0.98	0.9800000	82.40000	2 2 1
0.98	2	1	2	0.98	0.9899495	78.60101	2 1 2
0.98	2	2	2	0.98	0.9899495	80.41010	2 2 2
0.99	1	1	1	0.99	0.9900000	50.00000	1 1 1
0.99	1	2	1	0.99	0.9900000	54.50000	1 2 1
0.99	1	1	2	0.99	0.9949874	49.75063	1 1 2
0.99	1	2	2	0.99	0.9949874	52.00628	1 2 2

x_vec	A_vec	C_vec	R_vec	FPR_vec	TPR_vec	G_vec	design_id
0.99	2	1	1	0.99	0.9900000	79.40000	2 1 1
0.99	2	2	1	0.99	0.9900000	81.20000	2 2 1
0.99	2	1	2	0.99	0.9949874	79.30025	2 1 2
0.99	2	2	2	0.99	0.9949874	80.20251	2 2 2

The best classifier is the one that minimizes the objective function the best over values of x. In this case it was one with the design combinations of (A:1 C1: R:2) or (A:2 C1: R:2). For higher values of x an unbalanced population gives higher values from the objective function. A1 is thus better with even populations when. This is because with an uneven class ratio the mis-classification rate increases for higher values of x because a higher x value means a higher false positive count. Even when the false negative count is low. THis classifier is therefore not stable for all values of x and we choose the (A:1 C1: R:2) combination over the (A:2 C1: R:2) combination. Even though the (A:2 C1: R:2) combination clearly performs better at lower values of x. This in turn increased the mis-classifications calculated in the objective function. All classifiers that had a cfn that was 10 times the cfp resulted in an objective function that was orders of magnitude larger than the other classifiers. This makes intuitive senses as it dramatically increases the net cost of mis-classifications. With a TPR that is square rooted the objective function consistently returns a lower mis-classifications score. R2 is thus better. This is because, when the square root of the TPR is used to derive True positives they are higher than than when the TPR is not square rooted. This in turn decreases the amount of mis-classifications.

Problem 6

```
# Generate data
set.seed(0)
n_vec = c(25, 50, 100, 200, 400, 800)
m = 1000
df = data.frame(m=1:m)
table_df = data.frame(n=n_vec) # results table
mean_uniq_vec = c()
sd_uniq_vec = c()
for (n in n_vec){
  I = seq(n)
  samp_vec = c(sample(x=I, size=m ,replace=TRUE))
  df[paste("n",n , sep="")] = samp_vec

  mean_uniq_vec = c(mean_uniq_vec, mean(unique(samp_vec)))
  sd_uniq_vec = c(sd_uniq_vec, sd(unique(samp_vec)))
}

#visualize data
par(mfrow=c(3,2))
for(i in names(df)[2:7]){
  hist(df[[i]],
       xlab = "number",
       main=i)
}
```




```r
# make table of results
table_df$Mean_Unique = myRound(mean_uniq_vec, acc=2)
table_df$SD_Unique = myRound(sd_uniq_vec, acc=2)
table_df = data.frame(t(table_df))

knitr::kable(table_df, format = "markdown")
```

	X1	X2	X3	X4	X5	X6
n	25.00	50.00	100.00	200.00	400.00	800.00
Mean_Unique	13.00	25.50	50.50	100.75	200.02	408.84
SD_Unique	7.36	14.58	29.01	57.92	116.29	230.78