stat428finalproject

2023-11-18

```
#Set seed for reproducibility
set.seed(123)
```

Question 1 Test Implementation (15 points)

```
#1. Nearest Neighbor Test
library(RANN)

#Calculate test statistic for nearest neigbor
NNT <- function(z, ix, sizes, R) {
  n1 <- sizes[1]
  n2 <- sizes[2]
  n <- n1 + n2

  #Create new permutated data
  z <- z[ix, ]

  #Find the nearest neighbors
  NN <- nn2(z, z, k=R+1)

  #Create blocks/groups
  block1 <- NN$nn.idx[1:n1, -1]
  block2 <- NN$nn.idx[(n1+1):n, -1]

  #Identify number of points in same group
  i1 <- sum(block1 < n1 + .5)
  i2 <- sum(block2 > n1 + .5)

  #Return calculated statistic
  return((i1 + i2) / (R * n))
}

#Return decision for nearest neighbor
NNT.perm <- function(z, sizes, R,alpha,B) {
  #Generate initial statistic
  T0=NNT(z,1:nrow(z),sizes, R)
  TPerm=rep(0,B)

  #Iterate through all permutations
  for (b in 1:B) {

    #Create permutations
    permindex=sample(1:nrow(z))

    #Input new statistic for permutated data
    TPerm[b]=NNT(z,permindex,sizes, R)
  }
  #Get p value
  P <- mean(c(T0, TPerm) >= T0)
  return(P<alpha)
}
```

```
#2. Energy Distance Test

#Calculate value of energy distance test statistic
EBT <- function(dst, ix, sizes) {
  n1 <- sizes[1]
  n2 <- sizes[2]

  #Create new permutated groups
  ii <- ix[1:n1]
  jj <- ix[(n1+1):(n1+n2)]

  #Calculate energy distance statistic
  w <- n1 * n2 / (n1 + n2)
  m11 <- sum(dst[ii, ii]) / (n1 * n1)
  m22 <- sum(dst[jj, jj]) / (n2 * n2)
  m12 <- sum(dst[ii, jj]) / (n1 * n2)
  e <- w * ((m12 + m12) - (m11 + m22))
  return (e)
}

#Return decision for energy distance test
EBT.perm <- function(dst, sizes,alpha,B) {
  #Generate initial statistic
  T0=EBT(dst,1:nrow(dst),sizes)
  TPerm=rep(0,B)
  for (b in 1:B) {
    #Create permutations
    permindex=sample(1:nrow(dst))

    #Input new statistic for permutated data
    TPerm[b]=EBT(dst,permindex,sizes)
  }

  #Get p value
  P <- mean(c(T0, TPerm) >= T0)
  return(P<alpha)
}
```

```
library(MASS)

#3. Hotelling T Square Test


# Function to calculate Hotelling's T-square test statistic
hotellings_t_square <- function(x, y) {
  n_x <- nrow(x)
  n_y <- nrow(y)

  # Calculate the mean vectors
  mean_x <- colMeans(x)
  mean_y <- colMeans(y)

  # Calculate the covariance matrices
  cov_x <- cov(x)
  cov_y <- cov(y)

  # Pooled covariance matrix
  pooled_cov <- ((n_x - 1) * cov_x + (n_y - 1) * cov_y) / (n_x + n_y - 2)
  # Calculate the Hotelling's T-square test statistic
  t_square <- n_x * n_y / (n_x + n_y) * t(mean_x - mean_y) %*% ginv(pooled_cov) %*% (
mean_x - mean_y)
  return(t_square)
}

#Calculate value of Hotelling T square test statistic
HTT <- function(z, ix, sizes) {
  n1 <- sizes[1]
  n2 <- sizes[2]
  n <- n1 + n2
  z <- z[ix, ]

  #Create permutated groups
  x <- z[1:n1, ]
  y <- z[(n1+1):n, ]

  #Calculate Hotelling's T-square test statistic between x and y
  hotelling_stat <- hotellings_t_square(x, y)
  return(hotelling_stat)
}


#Return decision for Hotelling T square test
HTT.perm <- function(z, sizes,alpha,B) {
  #Generate initial statistic
  T0=HTT(z,1:nrow(z),sizes)
  TPerm=rep(0,B)
  for (b in 1:B) {
    #Generate permutations
    permindex=sample(1:nrow(z))

    #Input new statistic for permutated data
    TPerm[b]=HTT(z,permindex,sizes)
  }
  #Create p-value
  P <- mean(c(T0, TPerm) >= c(T0))
  return(P<alpha)
}
```

```
#4. Graph based two sample test

#Returns value of graph test statistic

GST <- function(dst, ix, sizes, Q) {
  n1 <- sizes[1]
  n2 <- sizes[2]

  #Create permutated groups
  ii <- ix[1:n1]
  jj <- ix[(n1+1):(n1+n2)]
  n <- n1 + n2

  num_edges <- 0
  sum_I_e <- 0
  for(i in 1:n) {
    for (j in i:n) {
      #Check if there should be edge between i and j
      if(dst[i, j] != 0.0 & dst[i, j] <= Q) {
        num_edges <- num_edges + 1

        #Check if the two vertics have same label
        group1 <- (i %in% ii) & (j %in% ii)
        group2 <- (i %in% jj) & (j %in% jj)

        if (group1 | group2) {
          sum_I_e <- sum_I_e + 1
        }
      }
    }
  }

  #Calculate statistic
  R <- sum_I_e/num_edges
  return(R)
}



#Returns value of the test decision for graph based two sample test
```

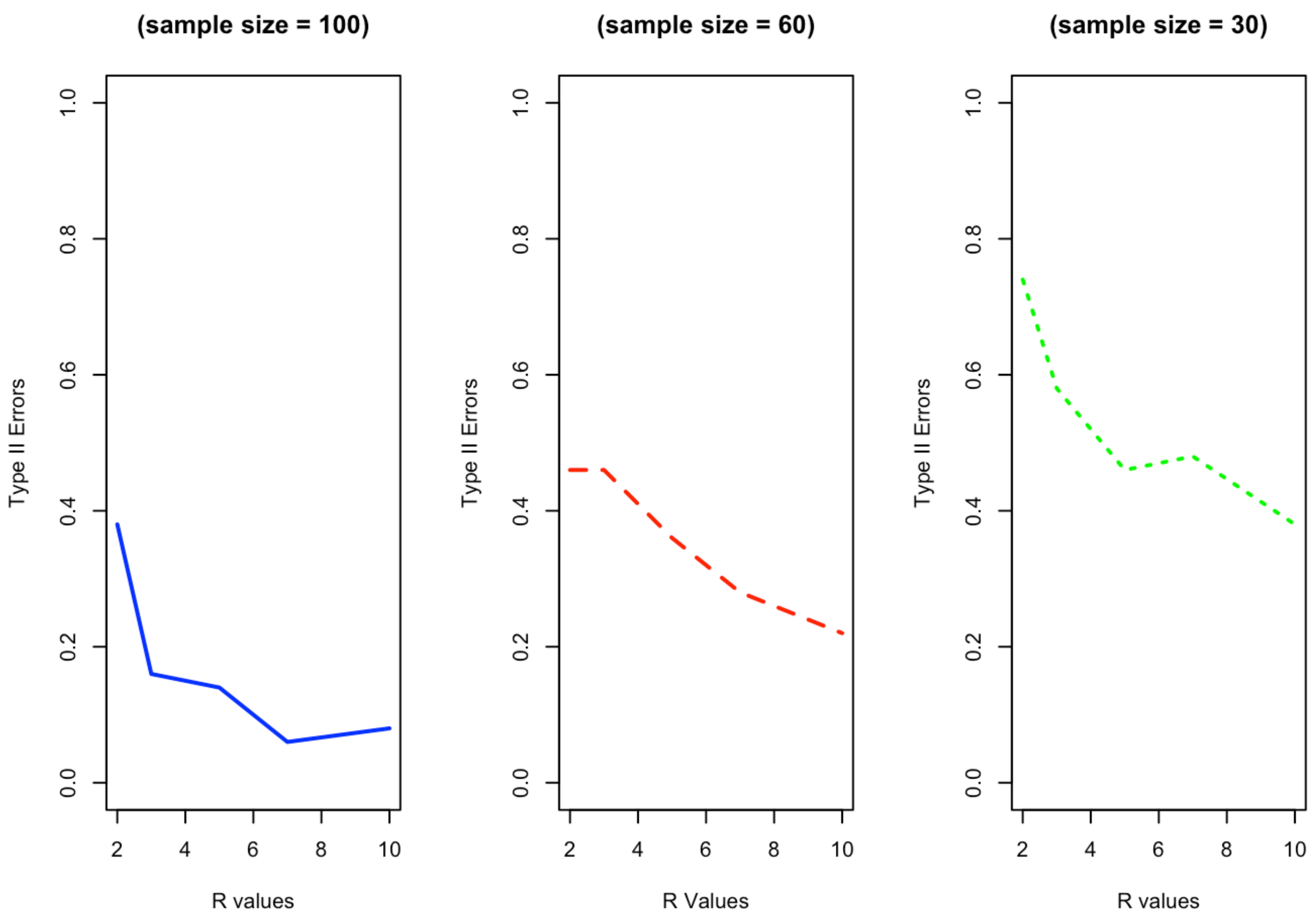
```
GST.perm <- function(dst, sizes,Q, alpha,B) {

  #Generate initial statistic
  T0=GST(dst,1:nrow(dst),sizes, Q)
  TPerm=rep(0,B)
  for (b in 1:B) {
    #Create permutations
    permindex=sample(1:nrow(dst))

    #Input new statistic for permutated data
    TPerm[b]=GST(dst,permindex,sizes, Q)
  }


  #Create p-value
  P <- mean(c(T0, TPerm) >= c(T0))
  return(P<alpha)
}
```

Question 2

In the nearest neighbor test, how should we choose the number of nearest neighbors R?

```r
n = 100; m = 100; d = 5
times <- 50
r_values <- c(2, 3, 5, 7, 10)
type_two_errors_100 <- c()
decision <- numeric(times)

for (r in r_values) {
  for (t in 1:times) {
    x <- matrix(rnorm(n*d), n, d)
    y <- matrix(rnorm(m*d,  mean = 0.4), m, d)
    z <- rbind(x, y)
    decision[t]=NNT.perm(z, c(n,m), R=r,alpha=0.05,B=200)
  }
  type_two_errors_100 <- append(type_two_errors_100, mean(1-decision))
}

type_two_errors_60 <- c()
decision <- numeric(times)
n = 60; m = 60; d = 5
for (r in r_values) {
  for (t in 1:times) {
    x <- matrix(rnorm(n*d), n, d)
    y <- matrix(rnorm(m*d,  mean = 0.4), m, d)
    z <- rbind(x, y)
    decision[t]=NNT.perm(z, c(n,m), R=r,alpha=0.05,B=200)
  }
  type_two_errors_60 <- append(type_two_errors_60, mean(1-decision))
}

type_two_errors_30 <- c()
decision <- numeric(times)
n = 30; m = 30; d = 5
for (r in r_values) {
  for (t in 1:times) {
    x <- matrix(rnorm(n*d), n, d)
    y <- matrix(rnorm(m*d,  mean = 0.4), m, d)
    z <- rbind(x, y)
    decision[t]=NNT.perm(z, c(n,m), R=r,alpha=0.05,B=200)
  }
  type_two_errors_30 <- append(type_two_errors_30, mean(1-decision))
}
```

```r
par(mfrow = c(1, 3))
plot(r_values, type_two_errors_100, type = "l", col = "blue", lty = 1, lwd = 2, ylim
= c(0, 1),
     xlab = "R values", ylab = "Type II Errors", main = "(sample size = 100)")
plot(r_values, type_two_errors_60, type = "l", col = "red", lty = 2, lwd = 2, ylim =
c(0, 1),
     xlab = "R Values", ylab = "Type II Errors", main = "(sample size = 60)")
plot(r_values, type_two_errors_30, type = "l", col = "green", lty = 3, lwd = 2, ylim
= c(0, 1),
     xlab = "R values", ylab = "Type II Errors", main = "(sample size = 30)")
```



From all three numerical experiments with differing sample sizes, we can see that in general, as the value of R increases, the type II error decreases. Therefore, we can make the conclusion that the desired type II error influences the choice of R. We should choose the number of nearest neighbors basd upon the desired type II error because we see that the type II error changes (decreases) as the R values changes (increases).

For sample size = 100, the R value of 7 minimizes the type II error as we can see from the plot above.

For sample size = 60, the R value of 10 minimizes the type II error.

For sample size = 30, the R value of 10 minimizes the type II error.

Additionally, sample size also influences the choice of nearest neighbors R. For example, from the plots, to obtain a type II error of 0.2, the test will sample size = 100 will need to have an R value of around 3. The test with sample size = 60 will need to have an R value of around 10. Finally, the test with sample size = 30 will need to have an R value of some integer greater than 10 (based on the trend). The conclusion is that to obtain a similar type II error across tests with different sample sizes, the value of R will also need to change.

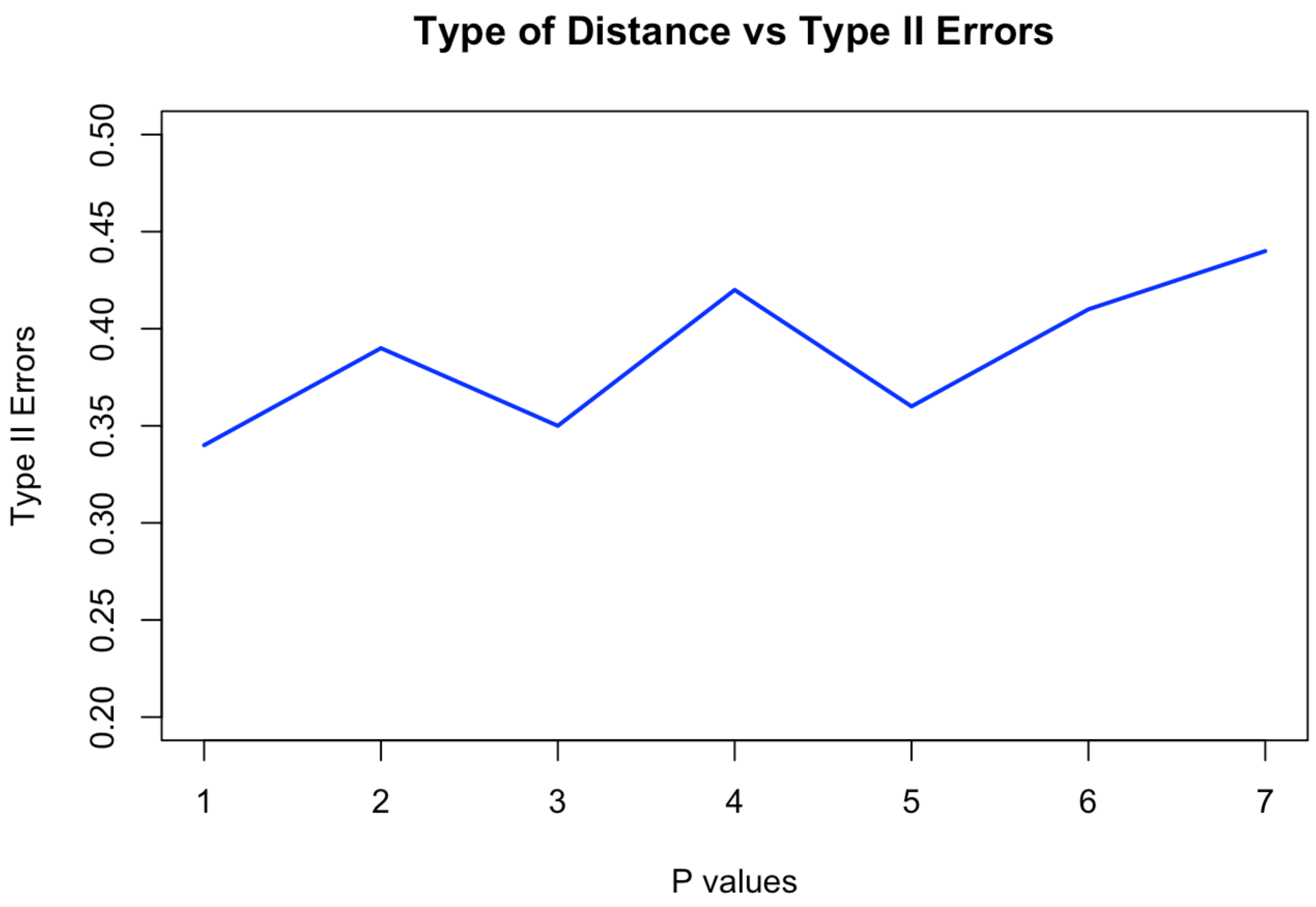
In conclusion, the desired type II error and the sample size affect the choice number of nearest neighbors R.

In the energy distance test, how should we choose the specific form of distance? For example, we may use Lp distance and what p should we use?

```r
#Define the distance function to be used in the energy test

#Parameter to adjust is p
distance_function <- function(z, p) {
  empty_matrix <- matrix(0, nrow = nrow(z), ncol = nrow(z))
  for (i in 1:nrow(z)){
    for(j in 1:nrow(z)){
      x <- z[i, ]
      y <- z[j, ]
      empty_matrix[i, j] <- sum(abs(x - y)^p)^(1/p)
    }
  }
  return(empty_matrix)
}
```

```r
p_values <- c(1, 2, 3, 4, 5, 6, 7)
n = 100; m = 100; d = 5

times = 100


type_two_errors2 <- c()
decision <- numeric(times)
for(p in p_values) {
  for (t in 1:times) {
    x <- matrix(rnorm(n*d), n, d)
    y <- matrix(rnorm(m*d, mean = 0.2), m, d)
    z <- rbind(x, y)
    dst <- as.matrix(distance_function(z, p))
    decision[t]=EBT.perm(dst, c(n,m),alpha=0.05,B=300)
  }
  type_two_errors2 <- append(type_two_errors2, mean(1-decision))
}
```

```
plot(p_values, type_two_errors2, type = "l", col = "blue", lty = 1, lwd = 2, ylim = c
(0.2, 0.5),
     xlab = "P values", ylab = "Type II Errors", main = "Type of Distance vs Type II
Errors")
```



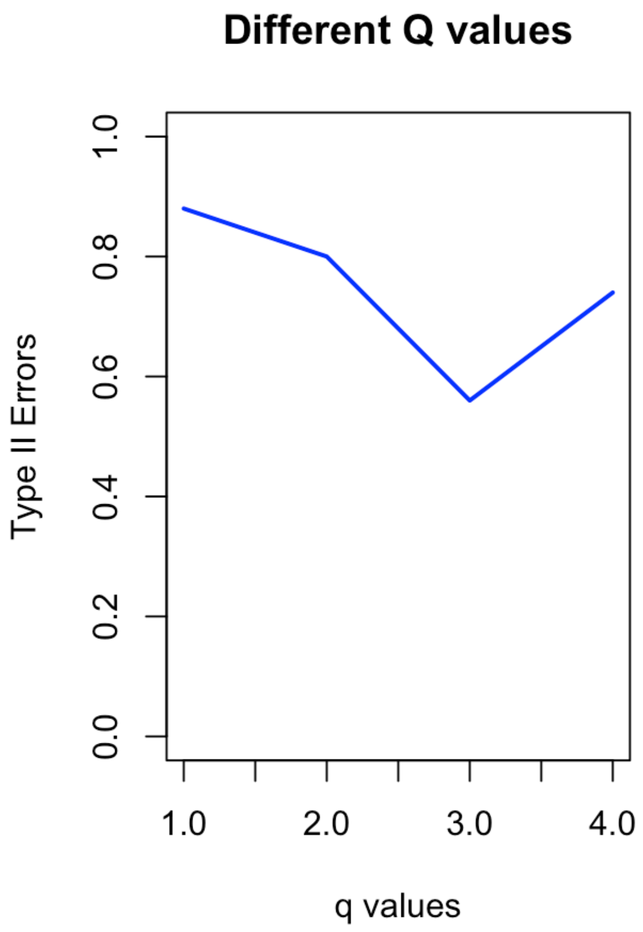
From our simulation results, we can see that as the p values or the form of distance changes, the type II error also changes. The change is not linear, but we can come to the conclusion that the desired type II error has an influence on the p value or the form of distance that should be used due to the fluctuations in Type II errors that is dependent on the form of distance. In this case, the recommended P value 1 distance form minimizes the type II error for sample size of 100 and dimensions = 5.

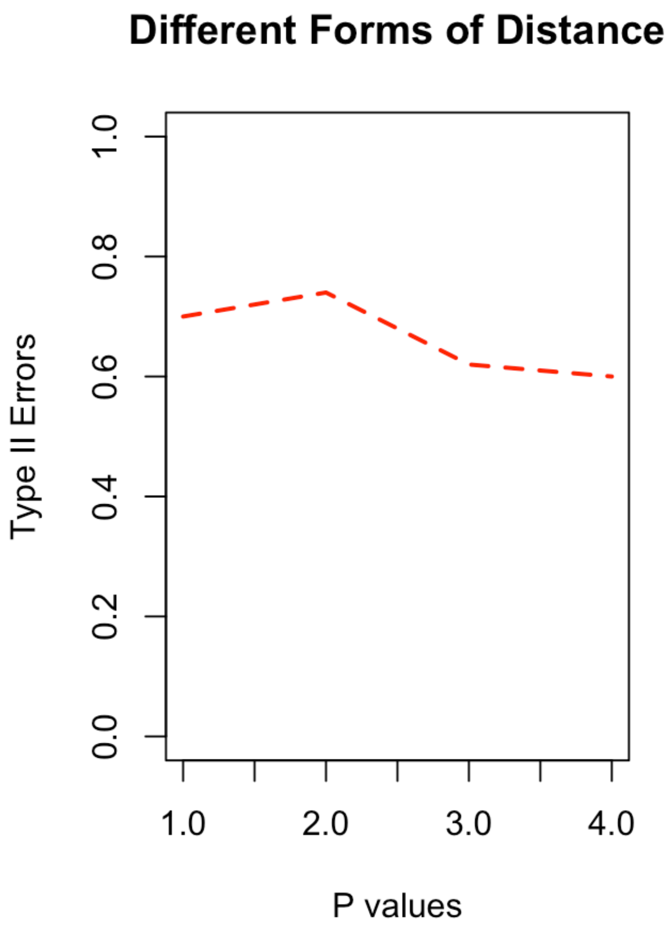
In the graph-based two-sample test, how should we choose the threshold Q and the specific form of distance?

```r
n = 50; m=50; d=2
times = 50

q_values <- c(1, 2, 3, 4)
decision <- numeric(times)
type_two_errors_q <- c()


for(q in q_values) {
  for (t in 1:times) {
    x <- matrix(rnorm(n*d), n, d)
    y <- matrix(rnorm(m*d, mean = 0.3), m, d)
    z <- rbind(x, y)
    dst <- as.matrix(dist(z))
    decision[t]=GST.perm(dst, c(n,m),Q = q, alpha=0.05,B=100)
  }
  type_two_errors_q <- append(type_two_errors_q, mean(1-decision))
}
```

```r
p_values <- c(1, 2, 3, 4)
type_two_errors_p <- c()
decision <- numeric(times)
for(p in p_values) {
  for (t in 1:times) {
    x <- matrix(rnorm(n*d), n, d)
    y <- matrix(rnorm(m*d, mean = 0.3), m, d)
    z <- rbind(x, y)
    dst <- distance_function(z, p)
    decision[t]=GST.perm(dst, c(n,m),Q = 2, alpha=0.05,B=100)
  }
  type_two_errors_p <- append(type_two_errors_p, mean(1-decision))
}
```

```r
par(mfrow = c(1, 2))


plot(q_values, type_two_errors_q, type = "l", col = "blue", lty = 1, lwd = 2, ylim =
c(0.0, 1),
     xlab = "q values", ylab = "Type II Errors", main = "Different Q values")
plot(p_values, type_two_errors_p, type = "l", col = "red", lty = 2, lwd = 2, ylim = c
(0.0, 1),
     xlab = "P values", ylab = "Type II Errors", main = "Different Forms of Distance"
)
```





The desired type II error determines the Q threshold and the form of distance to use (p value). In the simulation results above, we see that the type II error changes as the q threshold and the form of distance changes. In this case, the optimal q threshold that minimizes the type II error is q = 3 as seen from the plot (for sample size = 50 and dimensions = 2). Additionally, the optimal p value or form of distance that minimizes the type II error is p = 4 (for sample size = 50 and dimensions = 2) which can be seen from the plot.

Question 3

Which test is more suitable for low dimensional data set (i.e., d is small)? Which is better for high dimensional data set (i.e. d is large)?

```
n = 100; m=100; d=2
times <- 50
NNTdecision <- numeric(times)
EBTdecision <- numeric(times)
HTTdecision <- numeric(times)
graphdecision <- numeric(times)

#Run all tests for dimension = 2 and get type II error
for (t in 1:times) {
  x <- matrix(rnorm(n*d), n, d)
  y <- matrix(rnorm(m*d,  mean = 0.3), m, d)
  z <- rbind(x, y)
  NNTdecision[t]=NNT.perm(z, c(n,m), R=3,alpha=0.05,B=100)
  dst <- as.matrix(dist(z))
  EBTdecision[t]=EBT.perm(dst, c(n,m),alpha=0.05,B=100)
  HTTdecision[t]=HTT.perm(z, c(n,m), alpha=0.05,B=100)
  graphdecision[t]=GST.perm(dst, c(n,m),Q = 2, alpha=0.05,B=100)
}
```

```
lowdimensionalresults <- data.frame(Test = c("NNT", "EBT", "HTT", "Graph based"), typ
e_two_errors = c(mean(1 - NNTdecision), mean(1 - EBTdecision),


mean(1 - HTTdecision), mean(1 - graphdecision)))
lowdimensionalresults
```

```
##          Test type_two_errors
## 1         NNT            0.88
## 2         EBT            0.36
## 3         HTT            0.36
## 4 Graph based            0.40
```

```
n = 100; m=100; d=7
times <- 50
NNTdecision <- numeric(times)
EBTdecision <- numeric(times)
HTTdecision <- numeric(times)
graphdecision <- numeric(times)

for (t in 1:times) {
  x <- matrix(rnorm(n*d), n, d)
  y <- matrix(rnorm(m*d,  mean = 0.3), m, d)
  z <- rbind(x, y)
  NNTdecision[t]=NNT.perm(z, c(n,m), R=3,alpha=0.05,B=100)
  dst <- as.matrix(dist(z))
  EBTdecision[t]=EBT.perm(dst, c(n,m),alpha=0.05,B=100)
  HTTdecision[t]=HTT.perm(z, c(n,m), alpha=0.05,B=100)
  graphdecision[t]=GST.perm(dst, c(n,m),Q = 2, alpha=0.05,B=100)
}
```

```
highdimensionalresults <- data.frame(Test = c("NNT", "EBT", "HTT", "Graph based"), ty
pe_two_errors = c(mean(1 - NNTdecision), mean(1 - EBTdecision),


mean(1 - HTTdecision), mean(1 - graphdecision)))
highdimensionalresults
```

```
##          Test type_two_errors
## 1         NNT            0.44
## 2         EBT            0.00
## 3         HTT            0.00
## 4 Graph based            0.48
```

For the graph based test at lower dimensions (dimension = 2), the type II error is 0.40 and at higher dimensions (d = 7), the type II error increases to 0.48. For graph based test, type II error is less at lower dimensions than higher dimensions. This means that graph based test is more suitable for low dimensional data.

As seen from the tables above, NNT, EBT, and HTT all have lower type II errors at higher dimensional data than lower dimensional data, suggesting that those tests could be more suitable for higher dimensional data.

• Which test is more sensitive to different choices of the specific distribution of F or G?

We will measure the change in type II error from normal to exponential for all of the tests.

```
n = 100; m=100; d=2
times <- 50
NNTdecision <- numeric(times)
EBTdecision <- numeric(times)
HTTdecision <- numeric(times)
graphdecision <- numeric(times)

for (t in 1:times) {
  x <- matrix(rnorm(n*d), n, d)
  y <- matrix(rnorm(m*d), m, d)
  z <- rbind(x, y)
  NNTdecision[t]=NNT.perm(z, c(n,m), R=3,alpha=0.05,B=100)
  dst <- as.matrix(dist(z))
  EBTdecision[t]=EBT.perm(dst, c(n,m),alpha=0.05,B=100)
  HTTdecision[t]=HTT.perm(z, c(n,m), alpha=0.05,B=100)
  graphdecision[t]=GST.perm(dst, c(n,m),Q = 2, alpha=0.05,B=100)
}
```

```
expNNTdecision <- numeric(times)
expEBTdecision <- numeric(times)
expHTTdecision <- numeric(times)
expgraphdecision <- numeric(times)

for (t in 1:times) {
  x <- matrix(rexp(n*d), n, d)
  y <- matrix(rexp(m*d), m, d)
  z <- rbind(x, y)
  expNNTdecision[t]=NNT.perm(z, c(n,m), R=3,alpha=0.05,B=100)
  dst <- as.matrix(dist(z))
  expEBTdecision[t]=EBT.perm(dst, c(n,m),alpha=0.05,B=100)
  expHTTdecision[t]=HTT.perm(z, c(n,m), alpha=0.05,B=100)
  expgraphdecision[t]=GST.perm(dst, c(n,m),Q = 2, alpha=0.05,B=100)
}
```

```
sensitivityresults <- data.frame(Test = c("NNT change", "EBT change", "HTT change", "
Graph based change"), type_one_errors_difference = c(mean(NNTdecision) - mean(expNNTd
ecision), mean(EBTdecision) - mean(expEBTdecision),  mean(HTTdecision) - mean(expHTTd
ecision), mean(graphdecision) - mean(expgraphdecision)))
sensitivityresults
```

```
##                  Test type_one_errors_difference
## 1          NNT change                       0.06
## 2          EBT change                      -0.04
## 3          HTT change                      -0.02
## 4 Graph based change                      -0.02
```

To measure the sensitivity, we measured the difference in the type I errors for each of the test when the distribution was normal compared to when both distributions were exponential. The numerical results are shown above. It appears that NNT and EBT test are most sensitive to changes in the distributions from the results above as they had a greater difference in type II error when the distributions changed. HTT and graph based had a smaller difference in the type I errors when the distributions changed from normal to exponential.

• Are these tests able to control type I error?

```
n = 100; m=100; d=3
times <- 50
NNTdecision <- numeric(times)
EBTdecision <- numeric(times)
HTTdecision <- numeric(times)
graphdecision <- numeric(times)

for (t in 1:times) {
  x <- matrix(rnorm(n*d), n, d)
  y <- matrix(rnorm(m*d), m, d)
  z <- rbind(x, y)
  NNTdecision[t]=NNT.perm(z, c(n,m), R=3,alpha=0.05,B=100)
  dst <- as.matrix(dist(z))
  EBTdecision[t]=EBT.perm(dst, c(n,m),alpha=0.05,B=100)
  HTTdecision[t]=HTT.perm(z, c(n,m), alpha=0.05,B=100)
  graphdecision[t]=GST.perm(dst, c(n,m),Q = 2, alpha=0.05,B=100)
}
```

```
type_one_error_results <- data.frame(Test = c("NNT", "EBT", "HTT", "Graph based"), ty
pe_one_errors = c(mean(NNTdecision), mean(EBTdecision), mean(HTTdecision), mean(graph
decision)))
type_one_error_results
```

```
##           Test type_one_errors
## 1          NNT            0.04
## 2          EBT            0.02
## 3          HTT            0.02
## 4 Graph based            0.02
```

These tests are able to control type I error because as in the simulation results, the tests are able to achieve a result close to the alpha = significance level of 0.05 that served as input in the test. The type I errors from these tests are not far away from significance level = 0.05.

• Which test is more powerful?

```
n = 100; m=100; d=3
times <- 50
NNTdecision <- numeric(times)
EBTdecision <- numeric(times)
HTTdecision <- numeric(times)
graphdecision <- numeric(times)

for (t in 1:times) {
  x <- matrix(rnorm(n*d), n, d)
  y <- matrix(rnorm(m*d, mean = 0.3), m, d)
  z <- rbind(x, y)
  NNTdecision[t]=NNT.perm(z, c(n,m), R=3,alpha=0.05,B=100)
  dst <- as.matrix(dist(z))
  EBTdecision[t]=EBT.perm(dst, c(n,m),alpha=0.05,B=100)
  HTTdecision[t]=HTT.perm(z, c(n,m), alpha=0.05,B=100)
  graphdecision[t]=GST.perm(dst, c(n,m),Q = 2, alpha=0.05,B=100)
}
```

```
power_results <- data.frame(Test = c("NNT", "EBT", "HTT", "Graph based"), power = c(1
- mean(1-NNTdecision), 1-mean(1-EBTdecision), 1-mean(1-HTTdecision), 1-mean(1-graphde
cision)))
power_results
```

```
##          Test power
## 1         NNT  0.28
## 2         EBT  0.80
## 3         HTT  0.82
## 4 Graph based  0.56
```

The power of each test is shown above. This was done by subtracting the type II error from each test by 1. From our simulation results above, HTT is the most powerful test out of all 4 tests. EBT is also powerful while NNT is the least powerful test.

Which test is more computationally efficient?

```
n = 100; m=100; d=3



x <- matrix(rnorm(n*d), n, d)
y <- matrix(rnorm(m*d, mean = 0.3), m, d)
z <- rbind(x, y)

start_time <- proc.time()
NNT.perm(z, c(n,m), R=3,alpha=0.05,B=100)
```

```
## [1] TRUE
```

```
end_time <- proc.time()

NNT_time <- end_time[1] - start_time[1]

dst <- as.matrix(dist(z))

start_time <- proc.time()
EBT.perm(dst, c(n,m),alpha=0.05,B=100)
```

```
## [1] TRUE
```

```
end_time <- proc.time()

EBT_time <- end_time[1] - start_time[1]

start_time <- proc.time()
HTT.perm(z, c(n,m), alpha=0.05,B=100)
```

```
## [1] TRUE
```

```
end_time <- proc.time()
HTT_time <- end_time[1] - start_time[1]

start_time <- proc.time()
GST.perm(dst, c(n,m),Q = 2, alpha=0.05,B=100)
```

```
## [1] TRUE
```

```
end_time <- proc.time()
GST_time <- end_time[1]- start_time[1]
```

```
test_times <- data.frame(Test = c("NNT", "EBT", "HTT", "Graph based"), time = c(NNT_t
ime, EBT_time, HTT_time, GST_time))
test_times
```

```
##          Test  time
## 1         NNT 0.052
## 2         EBT 0.025
## 3         HTT 0.024
## 4 Graph based 8.904
```

From the simulation results above, HTT is the most computationally efficient test because it took the least amount of time to execute the code for the test. Next in terms of efficiency would be NNT and EBT. Finally, graph based test is the least efficient as it took by far the most amount of time for the test to execute.

Question 4.

```
pros.data = read.table("https://web.stanford.edu/~hastie/ElemStatLearn/datasets/prost
ate.data")
head(pros.data)
```

```
##       lcavol  lweight age      lbph svi       lcp gleason pgg45       lpsa
## 1 -0.5798185 2.769459  50 -1.386294   0 -1.386294       6     0 -0.4307829
## 2 -0.9942523 3.319626  58 -1.386294   0 -1.386294       6     0 -0.1625189
## 3 -0.5108256 2.691243  74 -1.386294   0 -1.386294       7    20 -0.1625189
## 4 -1.2039728 3.282789  58 -1.386294   0 -1.386294       6     0 -0.1625189
## 5  0.7514161 3.432373  62 -1.386294   0 -1.386294       6     0  0.3715636
## 6 -1.0498221 3.228826  50 -1.386294   0 -1.386294       6     0  0.7654678
##   train
## 1  TRUE
## 2  TRUE
## 3  TRUE
## 4  TRUE
## 5  TRUE
## 6  TRUE
```

Based on this data set, we are interested in if there is an overall difference between people older than 65 (age>65) and younger than 65 (age<=65). The variables of interest are lpsa, lweight, lcp and lbph. We can split the data set into two parts

```
X=pros.data[pros.data$age>65, c('lpsa','lweight','lcp','lbph')]
Y=pros.data[pros.data$age<=65, c('lpsa','lweight','lcp','lbph')]
```

Then, you can apply these four tests (with the best choice of tuning parameters and distances) to X and Y.What conclusion can you make?

X has 43 rows by 4 dimensions and Y has 54 rows by 4 dimensions. We want to conduct a hypothesis test on whether X is significantly different from Y.

Null Hypothesis: X is not significantly different than Y

Alternate Hypothesis: X is significantly different than Y

We will use a significance level of 0.05

```
combined <- rbind(X, Y)
NNT.perm(combined, c(43,54), R=3,alpha=0.05,B=100)
```

```
## [1] FALSE
```

```
HTT.perm(combined, c(43,54), alpha=0.05,B=100)
```

```
## [1] TRUE
```

```
dst <- as.matrix(dist(combined))


GST.perm(dst, c(43,54),Q = 2, alpha=0.05,B=100)
```

```
## [1] FALSE
```

```
EBT.perm(dst, c(43,54),alpha=0.05,B=100)
```

```
## [1] TRUE
```

HTT and EBT are the only tests in which the p value is less than 0.05 and suggests that we should reject the null hypothesis. More emphasis should be placed on the results of these tests than that of GST and NNT because HTT and EBT are more powerful tests than GST and NNT as seen from the simulation results in the previous question. Additionally, HTT and EBT have lower type II errors compared to the other tests for the same dimensions. As a result, we will use the results from the HTT and EBT tests.

Since these tests return TRUE, the p value is less than 0.05. There is enough evidence to reject the null hypothesis and conclude that X is significantly than Y. In other words, there is an overall statistically significant difference between people older than 65 (age>65) and younger than 65 (age<=65) from our two sample HTT and EBT tests.