hw3

Khoa Vu 705600710

Problem 1: Dimension Reduction Methods (40 pts) In this problem, please perform principal component analysis (PCA), multidimensional scaling (MDS), and t-SNE on the USArrests data set, which is part of the base R package. The rows of the data set contain the fifty states, in alphabetical order. The columns of the data set contain the four variables.

```
row.names(USArrests)
```

```
##  [1] "Alabama"        "Alaska"         "Arizona"        "Arkansas"
##  [5] "California"     "Colorado"       "Connecticut"    "Delaware"
##  [9] "Florida"        "Georgia"        "Hawaii"         "Idaho"
## [13] "Illinois"       "Indiana"        "Iowa"           "Kansas"
## [17] "Kentucky"       "Louisiana"      "Maine"          "Maryland"
## [21] "Massachusetts"  "Michigan"       "Minnesota"      "Mississippi"
## [25] "Missouri"       "Montana"        "Nebraska"       "Nevada"
## [29] "New Hampshire"  "New Jersey"     "New Mexico"     "New York"
## [33] "North Carolina" "North Dakota"   "Ohio"           "Oklahoma"
## [37] "Oregon"         "Pennsylvania"   "Rhode Island"   "South Carolina"
## [41] "South Dakota"   "Tennessee"      "Texas"          "Utah"
## [45] "Vermont"        "Virginia"       "Washington"     "West Virginia"
## [49] "Wisconsin"      "Wyoming"
```

```
names(USArrests)
```

```
## [1] "Murder"   "Assault"  "UrbanPop" "Rape"
```

1. Perform PCA using the prcomp() function to find principal components as linear combinations of the four columns, with each column centered at 0 and scaled to have standard deviation 1. Print the loadings of principal components 1-4.

Solution:

```
#Using prcomp() to perform PCA, we center the data at mean 0 and scale it to
#have unit variance by setting those parameters to be TRUE
#If we want to find the prinicpal components as linear combinations
#of the four columns, we define the response variable formula:
#~Murder + Assault + UrbanPop + Rape, equivalently, we use ~. to
#include all columns
USArrests_PCA = prcomp(~.,
                       data = USArrests,
                       center = TRUE,
                       scale = TRUE)

#We print the loadings of prinicpal components 1-4
loadings <- USArrests_PCA$rotation
loadings
```

```
##                 PC1        PC2        PC3         PC4
```

```
## Murder   -0.5358995 -0.4181809  0.3412327  0.64922780
## Assault  -0.5831836 -0.1879856  0.2681484 -0.74340748
## UrbanPop -0.2781909  0.8728062  0.3780158  0.13387773
## Rape     -0.5434321  0.1673186 -0.8177779  0.08902432
```

2. Obtain a summary of the proportion of variance explained (PVE) of the principal components using the summary() function on the prcomp object.

Solution:

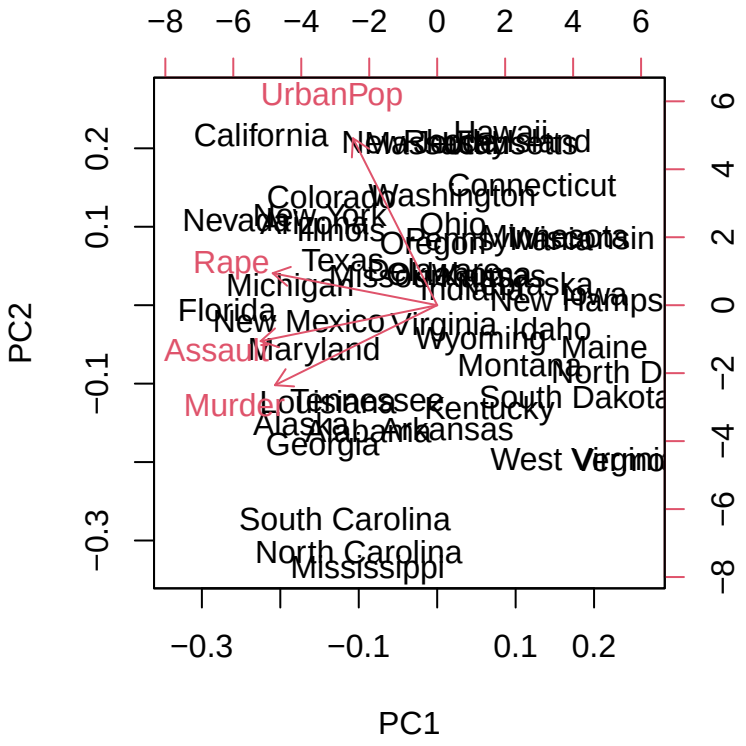
```
#Using the summary() function, we obtain a summary of the proportion of variance
#explained (PVE) of the principal components
summary(USArrests_PCA)$importance
```

```
##                             PC1       PC2       PC3       PC4
## Standard deviation     1.574878 0.9948694 0.5971291 0.4164494
## Proportion of Variance 0.620060 0.2474400 0.0891400 0.0433600
## Cumulative Proportion  0.620060 0.8675000 0.9566400 1.0000000
```

3. Use the biplot() function on the prcomp object to plot the scores of the 50 states in the first and second principal components along with the loadings of these two principal components shown as arrows in one plot. Explain the plot. What messages do you learn from this plot?

Solution:

```
#Plotting the scores of 50 states in the first and second principal components
biplot(USArrests_PCA)
```



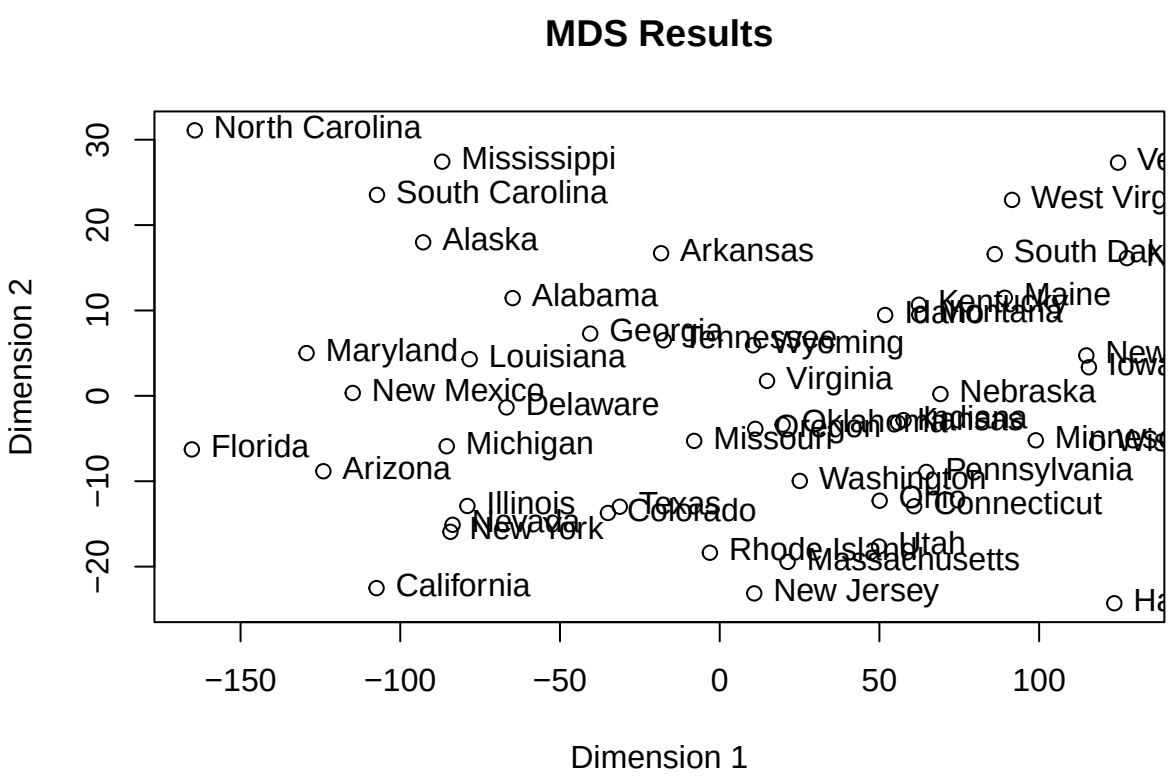
The loadings of each principal component demonstrate the contributions of each variable to that principal component. In PC1, we see that most variables have a negative loading, indicating that the absence of

crimes and a slight absence of urban population are associated with PC1. Thus, in PC1, states with lower crime rates and urban populations closer to zero will have higher positive PC1 scores, while states with higher crime rates and larger populations will have high negative PC1 scores. In PC2, we see that urban population and rape have large and small positive loadings, respectively, while murder and assault have medium and small negative loadings. These loadings mean that states with a large population, high rape incidences, and lower murder/assault incidences will have a higher positive PC2 score, and states with a lower population, low rape incidences, and higher murder/assault incidences will have a higher negative PC2 score. For example, California has a very high positive PC2 score and a slightly negative PC1 score. These scores mean California is a very populous state but has slightly higher incidences of crime. On the other hand, West Virginia has a somewhat negative PC2 score and a very positive PC1 score. These scores mean the state is not populated but has lower incidences of crime.

4. Calculate the pairwise Euclidean distance between the 50 states. Use the R function cmdscale() to perform MDS and visualize the result. Compare the result with the PCA result you visualize in Problem 5.3.

Solution:

```r
#Calculating the pairwise distances between each of the 50 states
pairwise_distance_USArrests <- dist(USArrests)
#Performing MDS using cmdscale() function
USArrests_MDS <- cmdscale(pairwise_distance_USArrests)

plot(USArrests_MDS,
     xlab = "Dimension 1",
     ylab = "Dimension 2",
     main = "MDS Results"
     )
text(USArrests_MDS, labels = row.names(USArrests_MDS), pos = 4)
```



Opposite to PCA, states with high populations negatively contribute along the second dimension of MDS. States with high populations, such as California and New York, have negative scores along the second dimension and states with low populations, such as West Virginia and Maine, have positive scores along the second dimension. Similar to PCA, crime rates seem to have a negative contribution along the first dimension of MDS. States with higher incidences of crime have a negative score along the first dimension, such as Florida, and states with lower incidences of crime, such as West Virginia, have a positive score along the first dimension. Thus, a state with a positive score along the first and second MDS dimensions has a low crime rate and population.

5. Apply t-SNE and UMAP to this data set using the R package M3C. Visualize the result, and compare it with the PCA and MDS results.

Solution:

```
#eval|echo=FALSE
#Installing the required packages
if (!require("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("M3C")
```

```
## 'getOption("repos")' replaces Bioconductor standard repositories, see
## 'help("repositories", package = "BiocManager")' for details.
## Replacement repositories:
##     CRAN: https://cloud.r-project.org
```

```
## Bioconductor version 3.20 (BiocManager 1.30.25), R 4.4.2 (2024-10-31)
```

```
## Warning: package(s) not installed when version(s) same as or greater than current; use
```

```
##   `force = TRUE` to re-install: 'M3C'
```

```
## Installation paths not writeable, unable to update packages
##   path: /usr/lib/R/library
##   packages:
##     boot, class, codetools, foreign, KernSmooth, lattice, MASS, Matrix, nlme,
##     nnet, rpart, spatial, survival
```

```
## Old packages: 'cluster', 'duckdb', 'httr2', 'nloptr', 'quantreg', 'RcppTOML',
##   'readxl', 'reticulate', 'rhdf5filters', 'rstan', 'spatstat.utils', 'tzdb',
##   'V8', 'xml2'
```

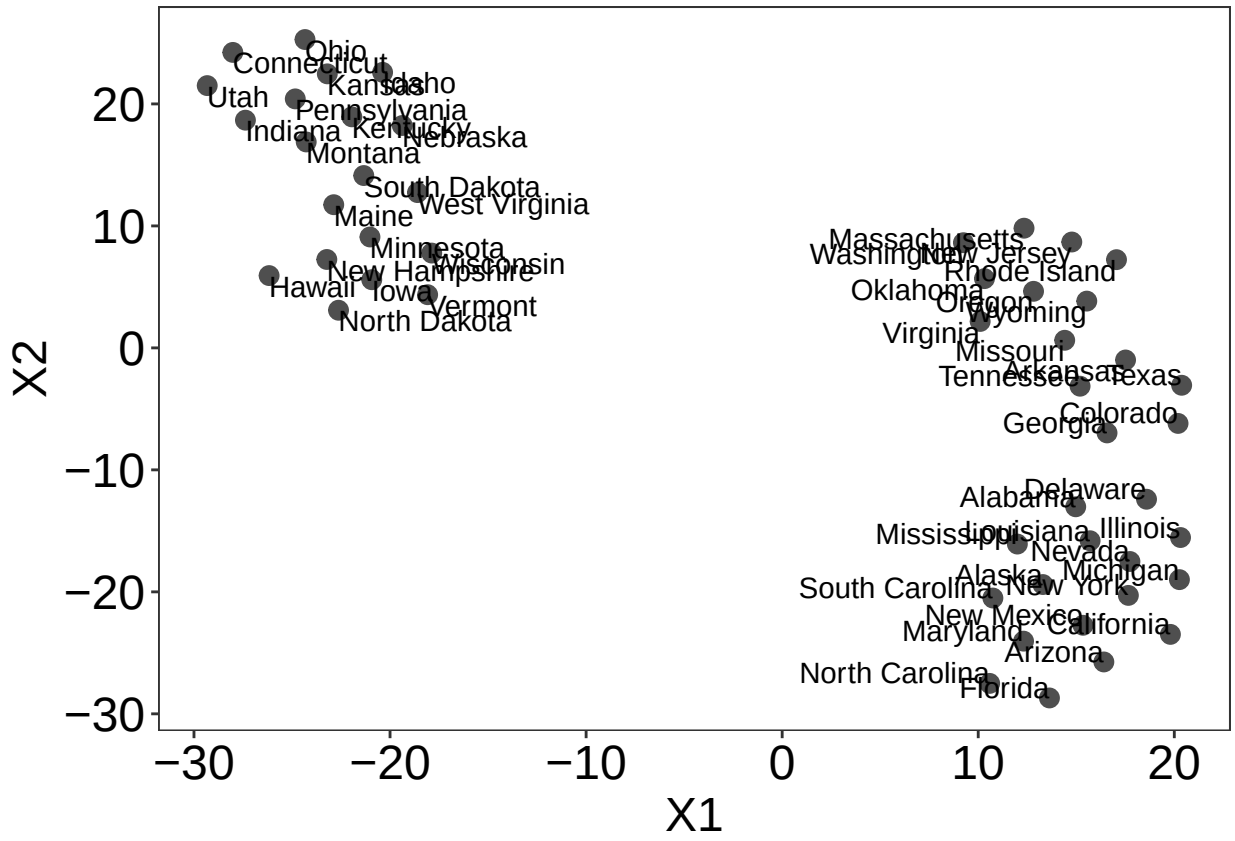
```
#Loading the required packages
library(M3C)

#Applying t-SNE, we set seed to 1 for reproducibility

#We need to transpose the data since tsne() expects samples as columns (states)
#and rows as features
USArrests_tsne <- tsne(t(USArrests), seed = 1,
                       dotsize = 3,
                       textlabelsize = 4,
                       colvec = "grey31",
                       text = row.names(USArrests))
```

```
## ***t-SNE wrapper function***
```

```
## running...
```

```
## done.
```

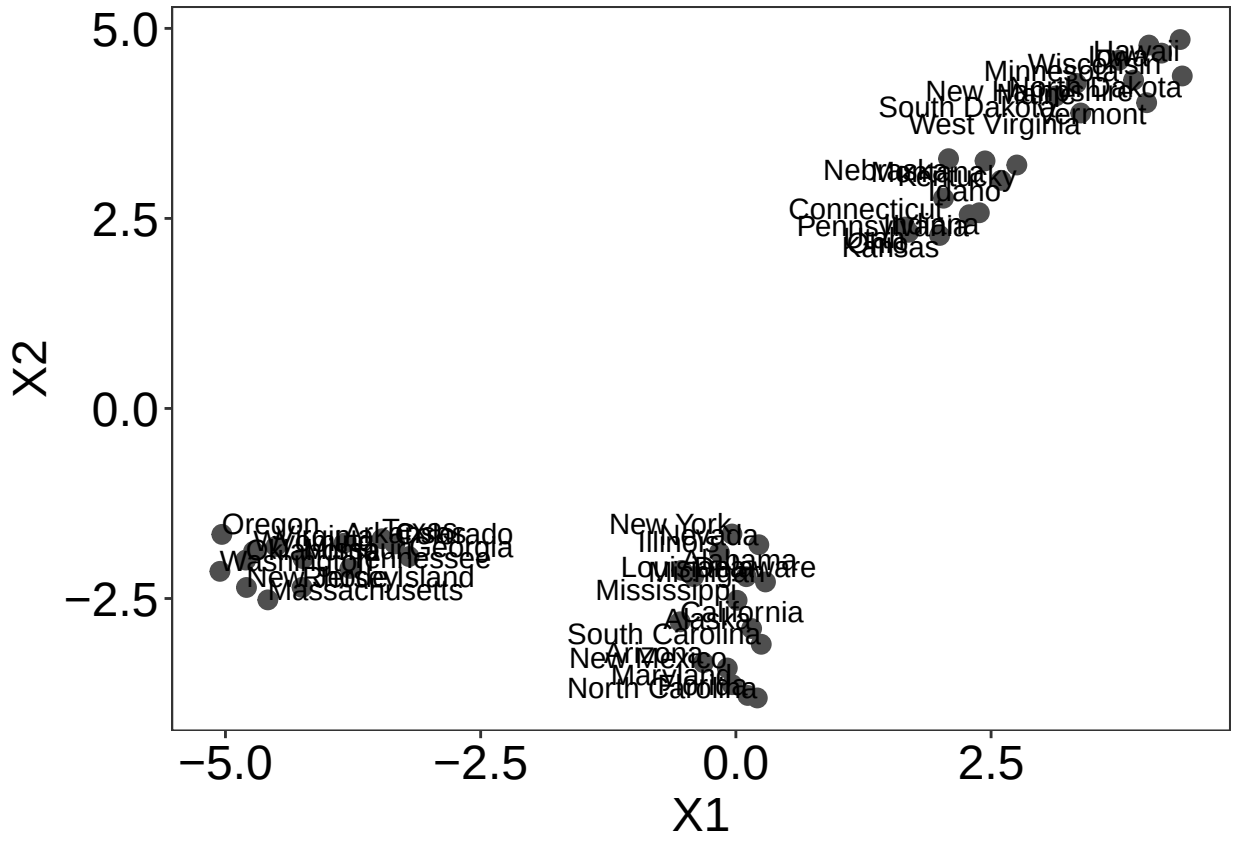
```
#Visualizing the t-SNE result
plot(USArrests_tsne)
```




```
#Applying UMAP, we set seed to 1 for reproducibility
USArrests_umap <- umap(t(USArrests), seed = 1,
                       dotsize = 3,
                       textlabelsize = 4,
                       colvec = "grey31",
                       text = row.names(USArrests))
```

```
## ***UMAP wrapper function***

## running...

## done.
```

```
plot(USArrests_umap)
```



The t-SNE and UMAP results show a well-defined clustering of the states for three clusters (for UMAP, this clustering is much more well-defined) in a 2D plot. However, because we don't see the loadings as to what contributes to each dimension, the clustering criteria of the states are less clear. However, Florida and California cluster together, so the cluster containing them seems to have high-population states with higher crime incidences. Meanwhile, the cluster with Maine and West Virginia seemingly contains states with small populations and low crime. The last cluster, with Wyoming and Rhode Island, seems to be states with moderate populations and crime rates.

Problem 2: Compare Measures of Association (40 pts)

Please use the following simulation study to compare different measures of association we covered in the class.

Assuming that X and Y are N(0, 1) variables, we would like to compare the power of different measures of association as test statistics for testing the following hypotheses.

Null hypothesis H_0: X and Y are independent;

vs.

Alternative hypothesis H_1: Y = X + ;

or

Alternative hypothesis H_2: $Y = X^2 +$;

or

Alternative hypothesis H_3: Y = sin(X) + ;

or

Alternative hypothesis H_4: $Y = \text{sign}(X) + \ ,$

where $\text{sign}(x) = 1$ if $x > 0$, $\text{sign}(x) = 0$ if $x = 0$ and $\text{sign}(x) = -1$ if $x < 0$;

or

Alternative hypothesis H_5: $Y = (-1)^Z X + \ $, where Z Bernoulli(0.5) and is independent of X and .

Suppose the error term $N(0, \ ^2)$. Calculate the power of measures of association for (1) H_0 vs. H_1, (2) H_0 vs. H_2, (3) H_0 vs. H_3, (4) H_0 vs. H_4, and (5) H_0 vs. H_5, using the following approach:

1. For B = 1, 000 times, simulate a sample of size n = 100 pairs of $(x_i \ , \ y_i)$, i = 1, . . . , n, under H_0. This gives B null data sets each with n data points.
2. For B = 1, 000 times, simulate a sample of size n = 100 pairs of $(x_i, \ y_i)$, i = 1, . . . , n, under the alternative hypothesis. This gives B alternative data sets each with n data points.
3. Apply the measure of association (e.g., Pearson correlation) to each of the B null data sets, resulting in B null values.
4. Apply the measure of association to each of the B alternative data sets, resulting in B alternative values.
5. At the significance level = 0.05, find the 95% percentile of the null values, using this percentile as the threshold.
6. Apply the threshold to the alternative values, calculate the percentage of the alternative values that are equal or above the threshold. Record this percentage as the power estimate.

Using the above approach, you will have one power estimate per ^{2} per measure. Vary ^{2} using values {.1, .3, .5, .7, .9, 1.1, 1.3, 1.5} and compare the following measures:

• Pearson correlation (function cor()) • Spearman rank correlation (function cor()) • Kendell's tau (function cor()) • maximal correlation (package acepack) • distance correlation (package energy) • maximal information coefficient (package minerva) • Chatterjee's correlation (package XICOR)

For each of tests (1)-(5), please summarize the power estimates using a line plot, whose horizontal axis is the ^{2} value (i.e, noise level) and vertical axis is the power estimate. This will give you a total of five plots. Use different colors for different measures. Label your plots clearly.

Solution:

```
#eval|echo=FALSE
#Installing the required packages
install.packages("acepack")
```

```
## Installing package into '/home/kvu1702/R/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
install.packages("energy")
```

```
## Installing package into '/home/kvu1702/R/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
install.packages("minerva")
```

```
## Installing package into '/home/kvu1702/R/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
install.packages("XICOR")
```

```
## Installing package into '/home/kvu1702/R/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
devtools::install_github("lijy03/gR2",ref="RcppVersion")
```

```
## Skipping install of 'gR2' from a github remote, the SHA1 (40670faf) has not changed since last instal
##   Use `force = TRUE` to force installation
```

```
#Importing the required packages
library(acepack)
library(energy)
library(minerva)
library(XICOR)
library(gR2)
library(ggplot2)
```

(1) Pearson correlation (function cor())

```
#Setting the seed for reproducibility
set.seed(1)

#Set the number of simulations
B = 1000
n = 100
#Defining the alpha levels
alpha <- 0.05

#Setting the sigma^2 values
sigma_squared_list = c(.1, .3, .5, .7, .9, 1.1, 1.3, 1.5)

#Defining the lists to store our power estimates
power_estimate_H0_vs_H1_pearson <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H2_pearson <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H3_pearson <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H4_pearson <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H5_pearson <- numeric(length(sigma_squared_list))

#Creating the null and alternative distribution dataset for each sigma^2 value
for (i in seq_len(length(sigma_squared_list))) {

  #Creating the null distribution dataset
  #We create B replicates of n normally distributed x_i, y_i pairs
  x_H0 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H0 <- replicate(B, rnorm(n, mean = 0, sd = 1))

  #Finding the null values using Pearson correlation (function cor())
  #and finding the threshold values
  null_values <- numeric(B)
  for (j in seq_len(B)) {
    null_values[j] <- cor(x_H0[, j], y_H0[, j], method = "pearson")
  }
  null_threshold <- quantile(null_values, 1 - alpha)

  #We create B replicates of n x_i, y_i pairs under the alternative hypothesis
  #Note: Epsilon ~ N(0, sigma^2)
  #We sample 1 value of epsilon B times under the distribution above
  epsilon <- replicate(B, rnorm(n, mean = 0, sd = sqrt(sigma_squared_list[i])))
```

```r
#H_1: Y = X + epsilon
x_H1 <- replicate(B, rnorm(n, mean = 0, sd = 1))
y_H1 <- x_H1 + epsilon
#We calculate the alternative values using Pearson correlation for each of the
#B alternative datasets
alternative_values_H1 <- numeric(B)
for (j in seq_len(B)) {
  alternative_values_H1[j] <- cor(x_H1[, j], y_H1[, j], method = "pearson")
}
#We calculate the power estimate by finding the percentage of values greater
#than or equal to the null threshold, then we divide by the total number of
#values for that calculation (equivalently taking the mean)
power_estimate_H0_vs_H1_pearson[i] <- mean(
  alternative_values_H1 >= null_threshold)

#H_2: Y = X^2 + epsilon
x_H2 <- replicate(B, rnorm(n, mean = 0, sd = 1))
y_H2 <- x_H2^2 + epsilon
alternative_values_H2 <- numeric(B)
for (j in seq_len(B)) {
  alternative_values_H2[j] <- cor(x_H2[, j], y_H2[, j], method = "pearson")
}
power_estimate_H0_vs_H2_pearson[i] <-
  mean(alternative_values_H2 >= null_threshold)

#H_3: Y = sin(X) + epsilon
x_H3 <- replicate(B, rnorm(n, mean = 0, sd = 1))
y_H3 <- sin(x_H3) + epsilon
alternative_values_H3 <- numeric(B)
for (j in seq_len(B)) {
  alternative_values_H3[j] <- cor(x_H3[, j], y_H3[, j], method = "pearson")
}
power_estimate_H0_vs_H3_pearson[i] <- mean(
  alternative_values_H3 >= null_threshold)

#H_4: Y = sign(X) + epsilon
x_H4 <- replicate(B, rnorm(n, mean = 0, sd = 1))
y_H4 <- sign(x_H4) + epsilon
alternative_values_H4 <- numeric(B)
for (j in seq_len(B)) {
  alternative_values_H4[j] <- cor(x_H4[, j], y_H4[, j], method = "pearson")
}
power_estimate_H0_vs_H4_pearson[i] <- mean(
  alternative_values_H4 >= null_threshold)

#H_5: Y = (-1)^Z*X + epsilon
#Sampling Z~Bernoulli(0.5)
Z <- replicate(B, rbinom(1, 1, 0.5))
x_H5 <- replicate(B, rnorm(n, mean = 0, sd = 1))
y_H5 <- (-1)^Z*x_H5 + epsilon
alternative_values_H5 <- numeric(B)
for (j in seq_len(B)) {
  alternative_values_H5[j] <- cor(x_H5[, j], y_H5[, j], method = "pearson")
```

```
  }
  power_estimate_H0_vs_H5_pearson[i] <- mean(
    alternative_values_H5 >= null_threshold)

}
```

(2) Spearman rank correlation (function cor())

```
#Setting the seed for reproducibility
set.seed(1)

#Set the number of simulations
B = 1000
n = 100
#Defining the alpha levels
alpha <- 0.05

#Setting the sigma^2 values
sigma_squared_list = c(.1, .3, .5, .7, .9, 1.1, 1.3, 1.5)

#Defining the lists to store our power estimates
power_estimate_H0_vs_H1_spearman <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H2_spearman <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H3_spearman <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H4_spearman <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H5_spearman <- numeric(length(sigma_squared_list))



#Creating the null alternative distribution dataset for each sigma^2 value
for (i in seq_len(length(sigma_squared_list))) {

  x_H0 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H0 <- replicate(B, rnorm(n, mean = 0, sd = 1))

  #Finding the null values using spearman correlation (function cor())
  #and finding the threshold values

  null_values <- numeric(B)
  for (j in seq_len(B)) {
    null_values[j] <- cor(x_H0[, j], y_H0[, j], method = "spearman")
  }
  null_threshold <- quantile(null_values, 1 - alpha)

  #We sample 1 value of epsilon B times under Epsilon ~ N(0, sigma^2)
  epsilon <- replicate(B, rnorm(n, mean = 0, sd = sqrt(sigma_squared_list[i])))

  #H_1: Y = X + epsilon
  x_H1 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H1 <- x_H1 + epsilon
  alternative_values_H1 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H1[j] <- cor(x_H1[, j], y_H1[, j], method = "spearman")
  }
```

```
  power_estimate_H0_vs_H1_spearman[i] <- mean(
    alternative_values_H1 >= null_threshold)

  #H_2: Y = X + epsilon
  x_H2 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H2 <- x_H2^2 + epsilon
  alternative_values_H2 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H2[j] <- cor(x_H2[, j], y_H2[, j], method = "spearman")
  }
  power_estimate_H0_vs_H2_spearman[i] <- mean(
    alternative_values_H2 >= null_threshold)

  #H_3: Y = sin(X) + epsilon
  x_H3 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H3 <- sin(x_H3) + epsilon
  alternative_values_H3 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H3[j] <- cor(x_H3[, j], y_H3[, j], method = "spearman")
  }
  power_estimate_H0_vs_H3_spearman[i] <- mean(
    alternative_values_H3 >= null_threshold)

  #H_4: Y = sign(X) + epsilon
  x_H4 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H4 <- sign(x_H4) + epsilon
  alternative_values_H4 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H4[j] <- cor(x_H4[, j], y_H4[, j], method = "spearman")
  }
  power_estimate_H0_vs_H4_spearman[i] <- mean(
    alternative_values_H4 >= null_threshold)

  #H_5: Y = (-1)^Z*X + epsilon
  #Sampling Z~Bernoulli(0.5)
  Z <- replicate(B, rbinom(1, 1, 0.5))
  x_H5 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H5 <- (-1)^Z*x_H5 + epsilon
  alternative_values_H5 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H5[j] <- cor(x_H5[, j], y_H5[, j], method = "spearman")
  }
  power_estimate_H0_vs_H5_spearman[i] <- mean(
    alternative_values_H5 >= null_threshold)

}
```

(3) Kendell's tau (function cor())

```
#Setting the seed for reproducibility
set.seed(1)

#Set the number of simulations
B = 1000
```

```r
n = 100
#Defining the alpha levels
alpha <- 0.05

#Setting the sigma^2 values
sigma_squared_list = c(.1, .3, .5, .7, .9, 1.1, 1.3, 1.5)

#Defining the lists to store our power estimates
power_estimate_H0_vs_H1_kendall <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H2_kendall <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H3_kendall <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H4_kendall <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H5_kendall <- numeric(length(sigma_squared_list))


#Creating the null and alternative distribution dataset for each sigma^2 value
for (i in seq_len(length(sigma_squared_list))) {

  x_H0 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H0 <- replicate(B, rnorm(n, mean = 0, sd = 1))

  #Finding the null values using Kendell's tau (function cor())
  #and finding the threshold values
  null_values <- numeric(B)
  for (j in seq_len(B)) {
    null_values[j] <- cor(x_H0[, j], y_H0[, j], method = "kendall")
  }
  null_threshold <- quantile(null_values, 1 - alpha)

  #We sample 1 value of epsilon B times under Epsilon ~ N(0, sigma^2)
  epsilon <- replicate(B, rnorm(1, mean = 0, sd = sqrt(sigma_squared_list[i])))

  #H_1: Y = X + epsilon
  x_H1 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H1 <- x_H1 + epsilon
  alternative_values_H1 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H1[j] <- cor(x_H1[, j], y_H1[, j], method = "kendall")
  }
  power_estimate_H0_vs_H1_kendall[i] <- mean(
    alternative_values_H1 >= null_threshold)

  #H_2: Y = X + epsilon
  x_H2 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H2 <- x_H2^2 + epsilon
  alternative_values_H2 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H2[j] <- cor(x_H2[, j], y_H2[, j], method = "kendall")
  }
  power_estimate_H0_vs_H2_kendall[i] <- mean(
    alternative_values_H2 >= null_threshold)

  #H_3: Y = sin(X) + epsilon
```

```
  x_H3 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H3 <- sin(x_H3) + epsilon
  alternative_values_H3 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H3[j] <- cor(x_H3[, j], y_H3[, j], method = "kendall")
  }
  power_estimate_H0_vs_H3_kendall[i] <- mean(
    alternative_values_H3 >= null_threshold)

  #H_4: Y = sign(X) + epsilon
  x_H4 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H4 <- sign(x_H4) + epsilon
  alternative_values_H4 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H4[j] <- cor(x_H4[, j], y_H4[, j], method = "kendall")
  }
  power_estimate_H0_vs_H4_kendall[i] <- mean(
    alternative_values_H4 >= null_threshold)

  #H_5: Y = (-1)^Z*X + epsilon
  #Sampling Z~Bernoulli(0.5)
  Z <- replicate(B, rbinom(1, 1, 0.5))
  x_H5 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H5 <- (-1)^Z*x_H5 + epsilon
  alternative_values_H5 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H5[j] <- cor(x_H5[, j], y_H5[, j], method = "kendall")
  }
  power_estimate_H0_vs_H5_kendall[i] <- mean(
    alternative_values_H5 >= null_threshold)

}
```

(4) maximal correlation (package acepack)

```
#Setting the seed for reproducibility
set.seed(1)

#Set the number of simulations
B = 1000
n = 100
#Defining the alpha levels
alpha <- 0.05

#Setting the sigma^2 values
sigma_squared_list = c(.1, .3, .5, .7, .9, 1.1, 1.3, 1.5)

#Defining the lists to store our power estimates
power_estimate_H0_vs_H1_max_corr <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H2_max_corr <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H3_max_corr <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H4_max_corr <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H5_max_corr <- numeric(length(sigma_squared_list))
```

```r
#Creating the alternative distribution dataset for each sigma^2 value
for (i in seq_len(length(sigma_squared_list))) {
  x_H0 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H0 <- replicate(B, rnorm(n, mean = 0, sd = 1))

  #Finding the null values using maximal correlation (package acepack)
  #and finding the threshold values
  null_values <- numeric(B)
  for (j in seq_len(B)) {
    null_values[j] <- ace(x_H0[, j], y_H0[, j])$rsq
  }
  null_threshold <- quantile(null_values, 1 - alpha)

  #We sample 1 value of epsilon B times under Epsilon ~ N(0, sigma^2)
  epsilon <- replicate(B, rnorm(1, mean = 0, sd = sqrt(sigma_squared_list[i])))

  #H_1: Y = X + epsilon
  x_H1 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H1 <- x_H1 + epsilon
  alternative_values_H1 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H1[j] <- ace(x_H1[, j], y_H1[, j])$rsq
  }
  power_estimate_H0_vs_H1_max_corr[i] <- mean(
    alternative_values_H1 >= null_threshold)

  #H_2: Y = X + epsilon
  x_H2 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H2 <- x_H2^2 + epsilon
  alternative_values_H2 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H2[j] <- ace(x_H2[, j], y_H2[, j])$rsq
  }
  power_estimate_H0_vs_H2_max_corr[i] <- mean(
    alternative_values_H2 >= null_threshold)

  #H_3: Y = sin(X) + epsilon
  x_H3 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H3 <- sin(x_H3) + epsilon
  alternative_values_H3 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H3[j] <- ace(x_H3[, j], y_H3[, j])$rsq
  }
  power_estimate_H0_vs_H3_max_corr[i] <- mean(
    alternative_values_H3 >= null_threshold)

  #H_4: Y = sign(X) + epsilon
  x_H4 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H4 <- sign(x_H4) + epsilon
  alternative_values_H4 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H4[j] <- ace(x_H4[, j], y_H4[, j])$rsq
  }
```

```
  power_estimate_H0_vs_H4_max_corr[i] <- mean(
    alternative_values_H4 >= null_threshold)

  #H_5: Y = (-1)^Z*X + epsilon
  #Sampling Z~Bernoulli(0.5)
  Z <- replicate(B, rbinom(1, 1, 0.5))
  x_H5 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H5 <- (-1)^Z*x_H5 + epsilon
  alternative_values_H5 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H5[j] <- ace(x_H5[, j], y_H5[, j])$rsq
  }
  power_estimate_H0_vs_H5_max_corr[i] <- mean(
    alternative_values_H5 >= null_threshold)

}
```

(5) distance correlation (package energy)

```
#Setting the seed for reproducibility
set.seed(1)

#Set the number of simulations
B = 1000
n = 100
#Defining the alpha levels
alpha <- 0.05

#Setting the sigma^2 values
sigma_squared_list = c(.1, .3, .5, .7, .9, 1.1, 1.3, 1.5)

#Defining the lists to store our power estimates
power_estimate_H0_vs_H1_dist_corr <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H2_dist_corr <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H3_dist_corr <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H4_dist_corr <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H5_dist_corr <- numeric(length(sigma_squared_list))

#Creating the alternative distribution dataset for each sigma^2 value
for (i in seq_len(length(sigma_squared_list))) {
  x_H0 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H0 <- replicate(B, rnorm(n, mean = 0, sd = 1))

  #Finding the null values using distance correlation (package energy)
  #and finding the threshold values
  null_values <- numeric(B)
  for (j in seq_len(B)) {
    null_values[j] <- dcor(x_H0[, j], y_H0[, j])
  }
  null_threshold <- quantile(null_values, 1 - alpha)

  #We sample 1 value of epsilon B times under Epsilon ~ N(0, sigma^2)
  epsilon <- replicate(B, rnorm(1, mean = 0, sd = sqrt(sigma_squared_list[i])))
```

```
  #H_1: Y = X + epsilon
  x_H1 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H1 <- x_H1 + epsilon
  alternative_values_H1 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H1[j] <- dcor(x_H1[, j], y_H1[, j])
  }
  power_estimate_H0_vs_H1_dist_corr[i] <- mean(
    alternative_values_H1 >= null_threshold)

  #H_2: Y = X + epsilon
  x_H2 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H2 <- x_H2^2 + epsilon
  alternative_values_H2 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H2[j] <- dcor(x_H2[, j], y_H2[, j])
  }
  power_estimate_H0_vs_H2_dist_corr[i] <- mean(
    alternative_values_H2 >= null_threshold)

  #H_3: Y = sin(X) + epsilon
  x_H3 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H3 <- sin(x_H3) + epsilon
  alternative_values_H3 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H3[j] <- dcor(x_H3[, j], y_H3[, j])
  }
  power_estimate_H0_vs_H3_dist_corr[i] <- mean(
    alternative_values_H3 >= null_threshold)

  #H_4: Y = sign(X) + epsilon
  x_H4 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H4 <- sign(x_H4) + epsilon
  alternative_values_H4 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H4[j] <- dcor(x_H4[, j], y_H4[, j])
  }
  power_estimate_H0_vs_H4_dist_corr[i] <- mean(
    alternative_values_H4 >= null_threshold)

  #H_5: Y = (-1)^Z*X + epsilon
  #Sampling Z~Bernoulli(0.5)
  Z <- replicate(B, rbinom(1, 1, 0.5))
  x_H5 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H5 <- (-1)^Z*x_H5 + epsilon
  alternative_values_H5 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H5[j] <- dcor(x_H5[, j], y_H5[, j])
  }
  power_estimate_H0_vs_H5_dist_corr[i] <- mean(
    alternative_values_H5 >= null_threshold)

}
```

(6) maximal information coefficient (package minerva)

```
#Setting the seed for reproducibility
set.seed(1)

#Set the number of simulations
B = 1000
n = 100
#Defining the alpha levels
alpha <- 0.05

#Setting the sigma^2 values
sigma_squared_list = c(.1, .3, .5, .7, .9, 1.1, 1.3, 1.5)

#Defining the lists to store our power estimates
power_estimate_H0_vs_H1_MIC <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H2_MIC <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H3_MIC <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H4_MIC <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H5_MIC <- numeric(length(sigma_squared_list))

#Creating the alternative distribution dataset for each sigma^2 value
for (i in seq_len(length(sigma_squared_list))) {
  x_H0 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H0 <- replicate(B, rnorm(n, mean = 0, sd = 1))

  #Finding the null values using distance correlation (package energy)
  #and finding the threshold values
  null_values <- numeric(B)
  for (j in seq_len(B)) {
    null_values[j] <- cstats(matrix(x_H0[, j]), matrix(y_H0[, j]))[3]
  }
  null_threshold <- quantile(null_values, 1 - alpha)

  #We sample 1 value of epsilon B times under Epsilon ~ N(0, sigma^2)
  epsilon <- replicate(B, rnorm(1, mean = 0, sd = sqrt(sigma_squared_list[i])))

  #H_1: Y = X + epsilon
  x_H1 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H1 <- x_H1 + epsilon
  alternative_values_H1 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H1[j] <- cstats(matrix(x_H1[, j]), matrix(y_H1[, j]))[3]
  }
  power_estimate_H0_vs_H1_MIC[i] <- mean(
    alternative_values_H1 >= null_threshold)

  #H_2: Y = X + epsilon
  x_H2 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H2 <- x_H2^2 + epsilon
  alternative_values_H2 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H2[j] <- cstats(matrix(x_H2[, j]), matrix(y_H2[, j]))[3]
  }
```

```
  power_estimate_H0_vs_H2_MIC[i] <- mean(
    alternative_values_H2 >= null_threshold)

  #H_3: Y = sin(X) + epsilon
  x_H3 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H3 <- sin(x_H3) + epsilon
  alternative_values_H3 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H3[j] <- cstats(matrix(x_H3[, j]), matrix(y_H3[, j]))[3]
  }
  power_estimate_H0_vs_H3_MIC[i] <- mean(
    alternative_values_H3 >= null_threshold)

  #H_4: Y = sign(X) + epsilon
  x_H4 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H4 <- sign(x_H4) + epsilon
  alternative_values_H4 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H4[j] <- cstats(matrix(x_H4[, j]), matrix(y_H4[, j]))[3]
  }
  power_estimate_H0_vs_H4_MIC[i] <- mean(
    alternative_values_H4 >= null_threshold)

  #H_5: Y = (-1)^Z*X + epsilon
  #Sampling Z~Bernoulli(0.5)
  Z <- replicate(B, rbinom(1, 1, 0.5))
  x_H5 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H5 <- (-1)^Z*x_H5 + epsilon
  alternative_values_H5 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H5[j] <- cstats(matrix(x_H5[, j]), matrix(y_H5[, j]))[3]
  }
  power_estimate_H0_vs_H5_MIC[i] <- mean(
    alternative_values_H5 >= null_threshold)

}
```

(7) Chatterjee's correlation (package XICOR)

```
#Setting the seed for reproducibility
set.seed(1)

#Set the number of simulations
B = 1000
n = 100
#Defining the alpha levels
alpha <- 0.05

#Setting the sigma^2 values
sigma_squared_list = c(.1, .3, .5, .7, .9, 1.1, 1.3, 1.5)

#Defining the lists to store our power estimates
power_estimate_H0_vs_H1_XICOR <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H2_XICOR <- numeric(length(sigma_squared_list))
```

```r
power_estimate_H0_vs_H3_XICOR <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H4_XICOR <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H5_XICOR <- numeric(length(sigma_squared_list))

#Creating the alternative distribution dataset for each sigma^2 value
for (i in seq_len(length(sigma_squared_list))) {
  x_H0 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H0 <- replicate(B, rnorm(n, mean = 0, sd = 1))

  #Finding the null values using distance correlation (package energy)
  #and finding the threshold values
  null_values <- numeric(B)
  for (j in seq_len(B)) {
    null_values[j] <- calculateXI(x_H0[, j], y_H0[, j])
  }
  null_threshold <- quantile(null_values, 1 - alpha)

  #We sample 1 value of epsilon B times under Epsilon ~ N(0, sigma^2)
  epsilon <- replicate(B, rnorm(1, mean = 0, sd = sqrt(sigma_squared_list[i])))

  #H_1: Y = X + epsilon
  x_H1 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H1 <- x_H1 + epsilon
  alternative_values_H1 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H1[j] <- calculateXI(x_H1[, j], y_H1[, j])
  }
  power_estimate_H0_vs_H1_XICOR[i] <- mean(
    alternative_values_H1 >= null_threshold)

  #H_2: Y = X + epsilon
  x_H2 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H2 <- x_H2^2 + epsilon
  alternative_values_H2 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H2[j] <- calculateXI(x_H2[, j], y_H2[, j])
  }
  power_estimate_H0_vs_H2_XICOR[i] <- mean(
    alternative_values_H2 >= null_threshold)

  #H_3: Y = sin(X) + epsilon
  x_H3 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H3 <- sin(x_H3) + epsilon
  alternative_values_H3 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H3[j] <- calculateXI(x_H3[, j], y_H3[, j])
  }
  power_estimate_H0_vs_H3_XICOR[i] <- mean(
    alternative_values_H3 >= null_threshold)

  #H_4: Y = sign(X) + epsilon
  x_H4 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H4 <- sign(x_H4) + epsilon
```

```
  alternative_values_H4 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H4[j] <- calculateXI(x_H4[, j], y_H4[, j])
  }
  power_estimate_H0_vs_H4_XICOR[i] <- mean(
    alternative_values_H4 >= null_threshold)

  #H_5: Y = (-1)^Z*X + epsilon
  #Sampling Z~Bernoulli(0.5)
  Z <- replicate(B, rbinom(1, 1, 0.5))
  x_H5 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H5 <- (-1)^Z*x_H5 + epsilon
  alternative_values_H5 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H5[j] <- calculateXI(x_H5[, j], y_H5[, j])
  }
  power_estimate_H0_vs_H5_XICOR[i] <- mean(
    alternative_values_H5 >= null_threshold)

}
```

(8) generalized Pearson correlation squares (package gR2 available at https://github.com/lijy03/gR2)

```
#Setting the seed for reproducibility
RNGkind("L'Ecuyer-CMRG")
set.seed(1)

#Set the number of simulations
B = 1000
n = 100
#Defining the alpha levels
alpha <- 0.05

#Setting the sigma^2 values
sigma_squared_list = c(.1, .3, .5, .7, .9, 1.1, 1.3, 1.5)

#Defining the lists to store our power estimates
power_estimate_H0_vs_H1_gR2 <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H2_gR2 <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H3_gR2 <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H4_gR2 <- numeric(length(sigma_squared_list))
power_estimate_H0_vs_H5_gR2 <- numeric(length(sigma_squared_list))

#Creating the alternative distribution dataset for each sigma^2 value
for (i in seq_len(length(sigma_squared_list))) {
  x_H0 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H0 <- replicate(B, rnorm(n, mean = 0, sd = 1))

  #Finding the null values using distance correlation (package energy)
  #and finding the threshold values
  null_values <- numeric(B)
  for (j in seq_len(B)) {
    null_values[j] <- gR2(x_H0[, j], y_H0[, j], mc.cores = 1, inference = TRUE,
                          verbose = FALSE)$estimate
```

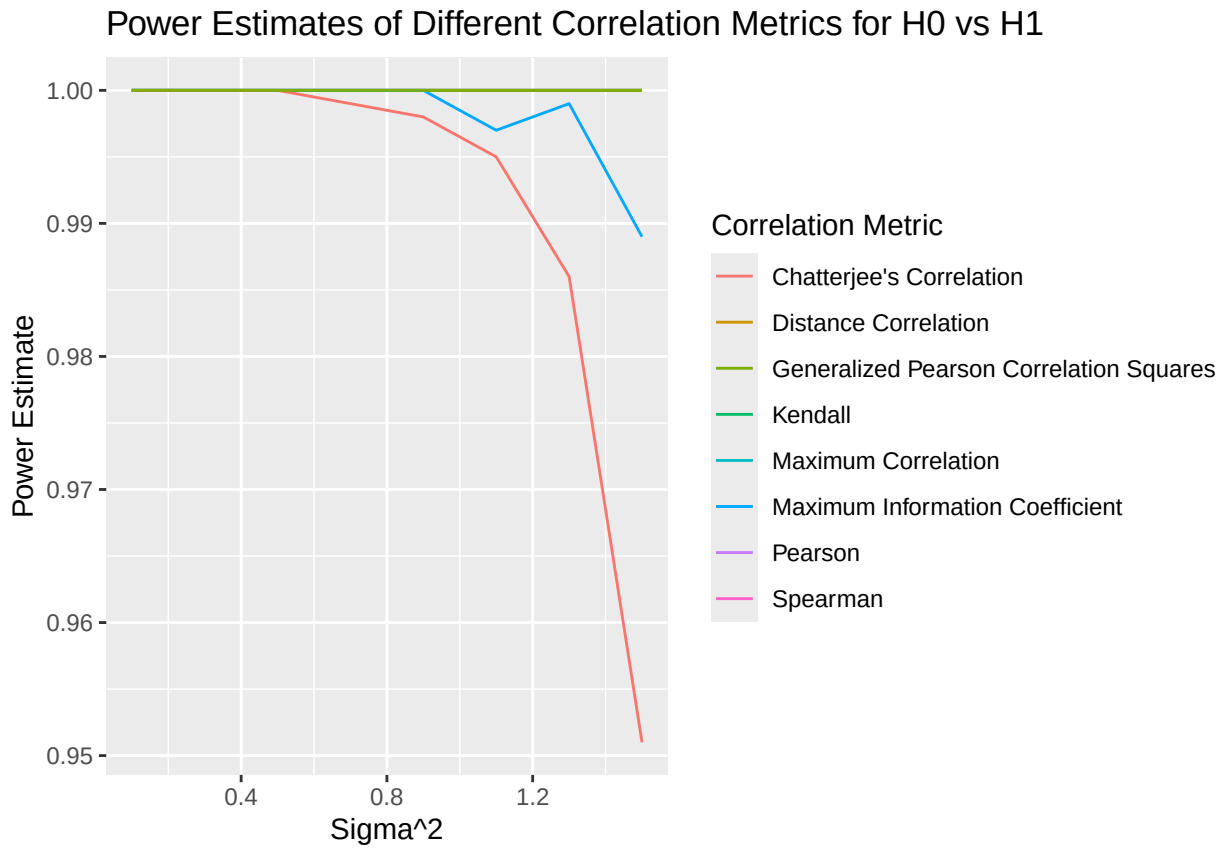
```
}
null_threshold <- quantile(null_values, 1 - alpha)

#We sample 1 value of epsilon B times under Epsilon ~ N(0, sigma^2)
epsilon <- replicate(B, rnorm(1, mean = 0, sd = sqrt(sigma_squared_list[i])))

#H_1: Y = X + epsilon
x_H1 <- replicate(B, rnorm(n, mean = 0, sd = 1))
y_H1 <- x_H1 + epsilon
alternative_values_H1 <- numeric(B)
for (j in seq_len(B)) {
  alternative_values_H1[j] <- gR2(x_H1[, j], y_H1[, j], mc.cores = 1,
                                  inference = TRUE,
                                  verbose = FALSE)$estimate
}
power_estimate_H0_vs_H1_gR2[i] <- mean(
  alternative_values_H1 >= null_threshold)

#H_2: Y = X + epsilon
x_H2 <- replicate(B, rnorm(n, mean = 0, sd = 1))
y_H2 <- x_H2^2 + epsilon
alternative_values_H2 <- numeric(B)
for (j in seq_len(B)) {
  alternative_values_H2[j] <- gR2(x_H2[, j], y_H2[, j], mc.cores = 1,
                                  verbose = FALSE)$estimate
}
power_estimate_H0_vs_H2_gR2[i] <- mean(
  alternative_values_H2 >= null_threshold)

#H_3: Y = sin(X) + epsilon
x_H3 <- replicate(B, rnorm(n, mean = 0, sd = 1))
y_H3 <- sin(x_H3) + epsilon
alternative_values_H3 <- numeric(B)
for (j in seq_len(B)) {
  alternative_values_H3[j] <- gR2(x_H3[, j], y_H3[, j], mc.cores = 1,
                                  verbose = FALSE)$estimate
}
power_estimate_H0_vs_H3_gR2[i] <- mean(
  alternative_values_H3 >= null_threshold)

#H_4: Y = sign(X) + epsilon
x_H4 <- replicate(B, rnorm(n, mean = 0, sd = 1))
y_H4 <- sign(x_H4) + epsilon
alternative_values_H4 <- numeric(B)
for (j in seq_len(B)) {
  alternative_values_H4[j] <- gR2(x_H4[, j], y_H4[, j], mc.cores = 1,
                                  verbose = FALSE)$estimate
}
power_estimate_H0_vs_H4_gR2[i] <- mean(
  alternative_values_H4 >= null_threshold)

#H_5: Y = (-1)^Z*X + epsilon
#Sampling Z~Bernoulli(0.5)
```

```
  Z <- replicate(B, rbinom(1, 1, 0.5))
  x_H5 <- replicate(B, rnorm(n, mean = 0, sd = 1))
  y_H5 <- (-1)^Z*x_H5 + epsilon
  alternative_values_H5 <- numeric(B)
  for (j in seq_len(B)) {
    alternative_values_H5[j] <- gR2(x_H5[, j], y_H5[, j], mc.cores = 1,
                                    verbose = FALSE)$estimate
  }
  power_estimate_H0_vs_H5_gR2[i] <- mean(
    alternative_values_H5 >= null_threshold)

}
```

Summarizing the power estimates using a line plot, whose horizontal axis is the ^{2} value (i.e, noise level) and vertical axis is the power estimate.

(1) HO vs H1

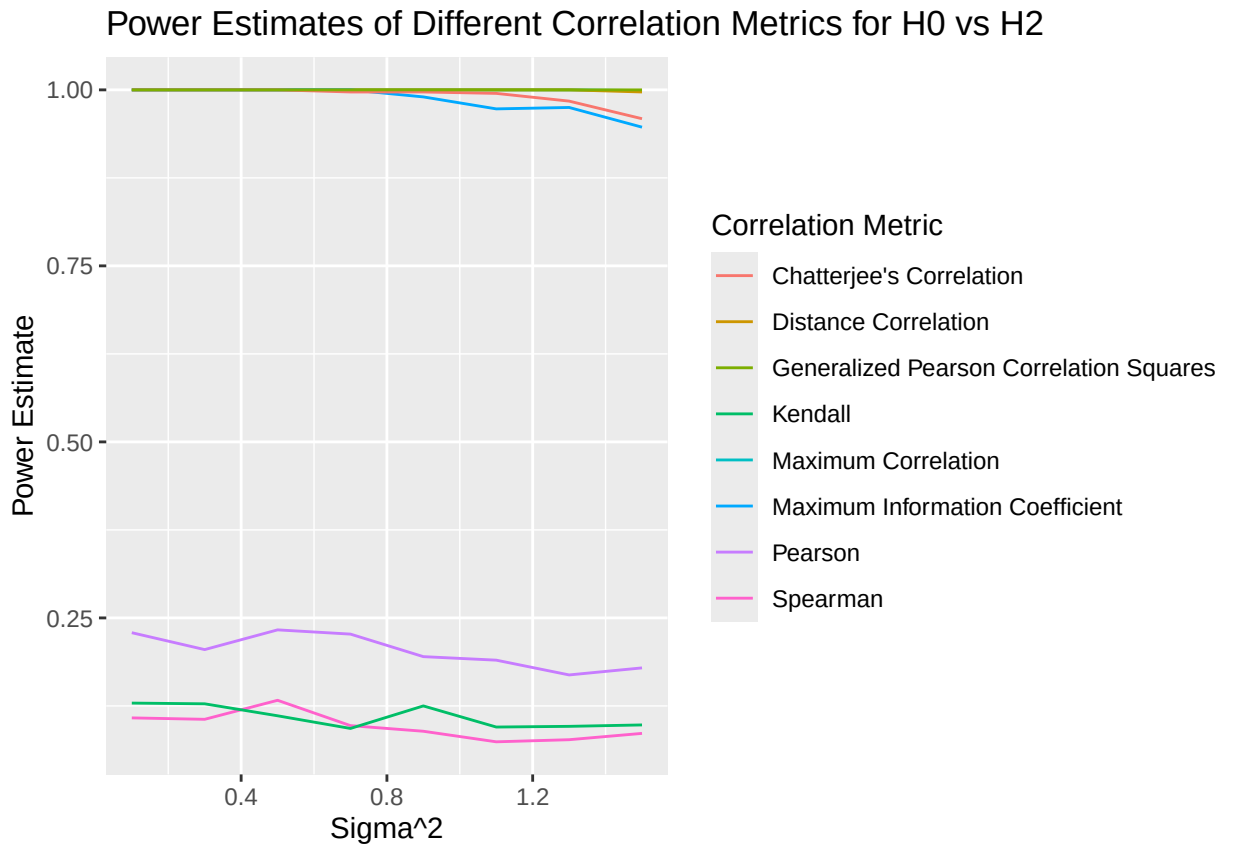
```
power_estimates_H0_H1 <- data.frame(sigma_squared_list,
                                    power_estimate_H0_vs_H1_pearson,
                                    power_estimate_H0_vs_H1_spearman,
                                    power_estimate_H0_vs_H1_kendall,
                                    power_estimate_H0_vs_H1_max_corr,
                                    power_estimate_H0_vs_H1_dist_corr,
                                    power_estimate_H0_vs_H1_MIC,
                                    power_estimate_H0_vs_H1_XICOR,
                                    power_estimate_H0_vs_H1_gR2)
colnames(power_estimates_H0_H1) <- c("sigma_squared", "pearson",
                                     "spearman", "kendall", "max_corr",
                                     "dist_corr", "MIC", "XICOR", "gR2")

ggplot(power_estimates_H0_H1, aes(x = sigma_squared)) +
  geom_line(aes(y = pearson, color = "Pearson")) +
  geom_line(aes(y = spearman, color = "Spearman")) +
  geom_line(aes(y = kendall, color = "Kendall")) +
  geom_line(aes(y = max_corr, color = "Maximum Correlation")) +
  geom_line(aes(y = dist_corr, color = "Distance Correlation")) +
  geom_line(aes(y = MIC, color = "Maximum Information Coefficient")) +
  geom_line(aes(y = XICOR, color = "Chatterjee's Correlation")) +
  geom_line(aes(y = gR2, color = "Generalized Pearson Correlation Squares")) +
  xlab("Sigma^2") +
  ylab("Power Estimate") +
  labs(color = "Correlation Metric") +
  ggtitle("Power Estimates of Different Correlation Metrics for H0 vs H1")
```



(2) HO vs H2

```
power_estimates_H0_H2 <- data.frame(sigma_squared_list,
                                    power_estimate_H0_vs_H2_pearson,
                                    power_estimate_H0_vs_H2_spearman,
                                    power_estimate_H0_vs_H2_kendall,
                                    power_estimate_H0_vs_H2_max_corr,
                                    power_estimate_H0_vs_H2_dist_corr,
                                    power_estimate_H0_vs_H2_MIC,
                                    power_estimate_H0_vs_H2_XICOR,
                                    power_estimate_H0_vs_H2_gR2)
colnames(power_estimates_H0_H2) <- c("sigma_squared", "pearson",
                                     "spearman", "kendall", "max_corr",
                                     "dist_corr", "MIC", "XICOR", "gR2")

ggplot(power_estimates_H0_H2, aes(x = sigma_squared)) +
  geom_line(aes(y = pearson, color = "Pearson")) +
  geom_line(aes(y = spearman, color = "Spearman")) +
  geom_line(aes(y = kendall, color = "Kendall")) +
  geom_line(aes(y = max_corr, color = "Maximum Correlation")) +
  geom_line(aes(y = dist_corr, color = "Distance Correlation")) +
  geom_line(aes(y = MIC, color = "Maximum Information Coefficient")) +
  geom_line(aes(y = XICOR, color = "Chatterjee's Correlation")) +
  geom_line(aes(y = gR2, color = "Generalized Pearson Correlation Squares")) +
  xlab("Sigma^2") +
  ylab("Power Estimate") +
  labs(color = "Correlation Metric") +
```

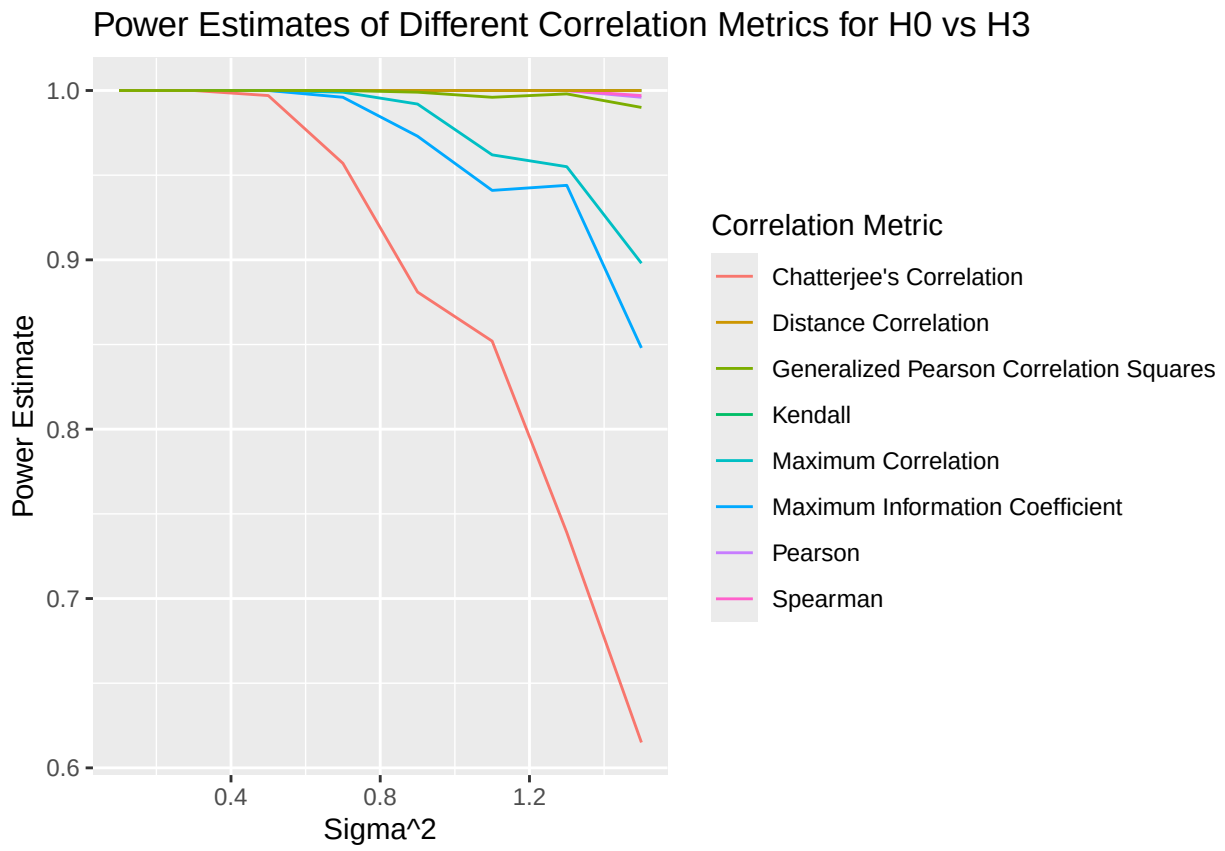
```
  ggtitle("Power Estimates of Different Correlation Metrics for H0 vs H2")
```



(3) HO vs H3

```
power_estimates_H0_H3 <- data.frame(sigma_squared_list,
                                    power_estimate_H0_vs_H3_pearson,
                                    power_estimate_H0_vs_H3_spearman,
                                    power_estimate_H0_vs_H3_kendall,
                                    power_estimate_H0_vs_H3_max_corr,
                                    power_estimate_H0_vs_H3_dist_corr,
                                    power_estimate_H0_vs_H3_MIC,
                                    power_estimate_H0_vs_H3_XICOR,
                                    power_estimate_H0_vs_H3_gR2)
colnames(power_estimates_H0_H3) <- c("sigma_squared", "pearson",
                                     "spearman", "kendall", "max_corr",
                                     "dist_corr", "MIC", "XICOR", "gR2")

ggplot(power_estimates_H0_H3, aes(x = sigma_squared)) +
  geom_line(aes(y = pearson, color = "Pearson")) +
  geom_line(aes(y = spearman, color = "Spearman")) +
  geom_line(aes(y = kendall, color = "Kendall")) +
  geom_line(aes(y = max_corr, color = "Maximum Correlation")) +
  geom_line(aes(y = dist_corr, color = "Distance Correlation")) +
  geom_line(aes(y = MIC, color = "Maximum Information Coefficient")) +
  geom_line(aes(y = XICOR, color = "Chatterjee's Correlation")) +
  geom_line(aes(y = gR2, color = "Generalized Pearson Correlation Squares")) +
```

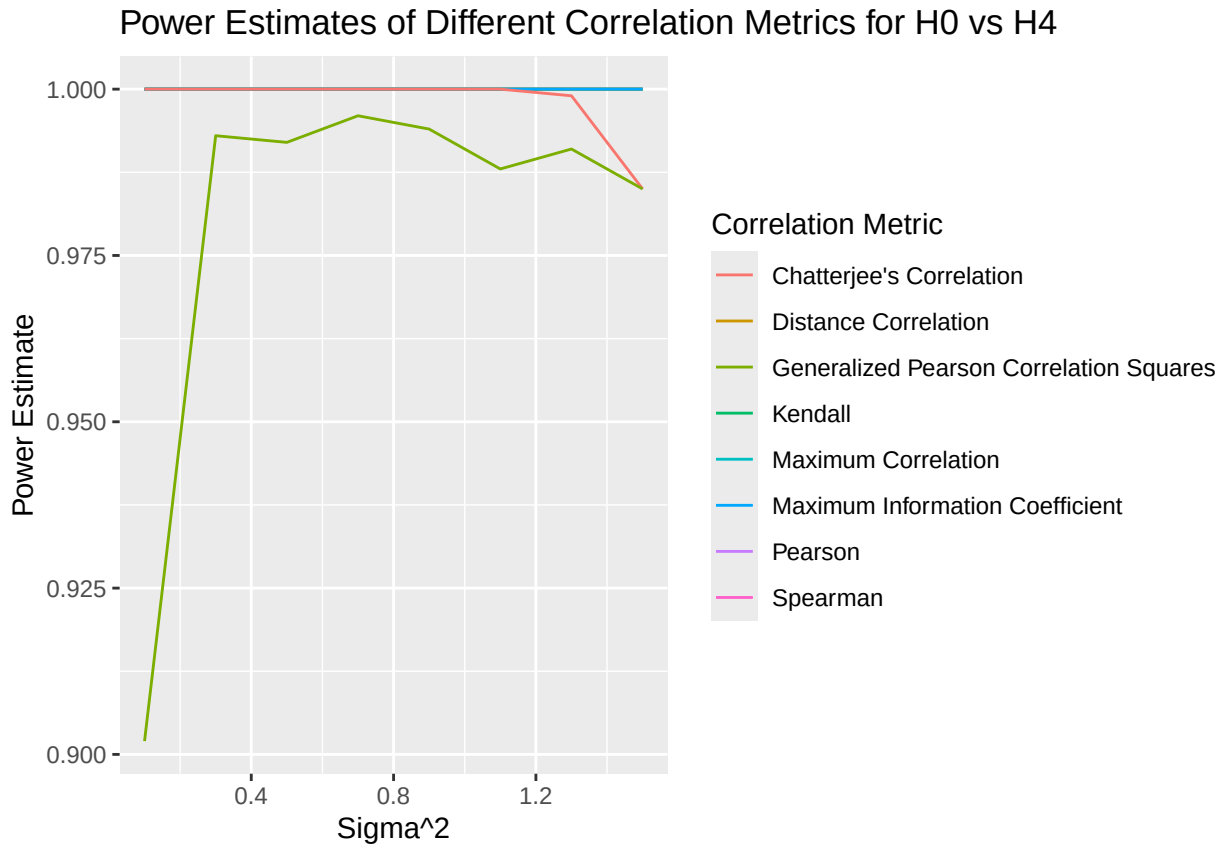
```
  xlab("Sigma^2") +
  ylab("Power Estimate") +
  labs(color = "Correlation Metric") +
  ggtitle("Power Estimates of Different Correlation Metrics for H0 vs H3")
```



(4) HO vs H4

```
power_estimates_H0_H4 <- data.frame(sigma_squared_list,
                                    power_estimate_H0_vs_H4_pearson,
                                    power_estimate_H0_vs_H4_spearman,
                                    power_estimate_H0_vs_H4_kendall,
                                    power_estimate_H0_vs_H4_max_corr,
                                    power_estimate_H0_vs_H4_dist_corr,
                                    power_estimate_H0_vs_H4_MIC,
                                    power_estimate_H0_vs_H4_XICOR,
                                    power_estimate_H0_vs_H4_gR2)
colnames(power_estimates_H0_H4) <- c("sigma_squared", "pearson",
                                     "spearman", "kendall", "max_corr",
                                     "dist_corr", "MIC", "XICOR", "gR2")

ggplot(power_estimates_H0_H4, aes(x = sigma_squared)) +
  geom_line(aes(y = pearson, color = "Pearson")) +
  geom_line(aes(y = spearman, color = "Spearman")) +
  geom_line(aes(y = kendall, color = "Kendall")) +
  geom_line(aes(y = max_corr, color = "Maximum Correlation")) +
  geom_line(aes(y = dist_corr, color = "Distance Correlation")) +
```

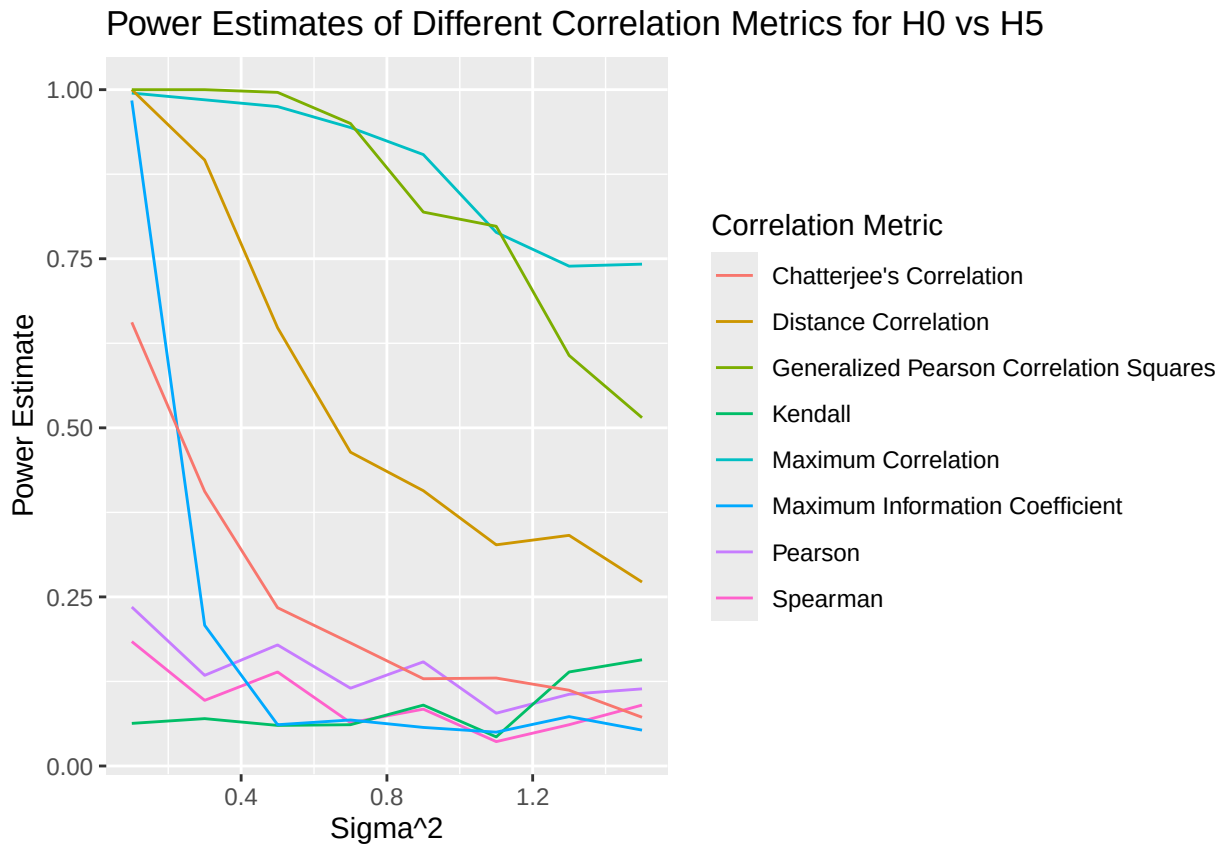
```
  geom_line(aes(y = MIC, color = "Maximum Information Coefficient")) +
  geom_line(aes(y = XICOR, color = "Chatterjee's Correlation")) +
  geom_line(aes(y = gR2, color = "Generalized Pearson Correlation Squares")) +
  xlab("Sigma^2") +
  ylab("Power Estimate") +
  labs(color = "Correlation Metric") +
  ggtitle("Power Estimates of Different Correlation Metrics for H0 vs H4")
```



(5) HO vs H5

```
power_estimates_H0_H5 <- data.frame(sigma_squared_list,
                                    power_estimate_H0_vs_H5_pearson,
                                    power_estimate_H0_vs_H5_spearman,
                                    power_estimate_H0_vs_H5_kendall,
                                    power_estimate_H0_vs_H5_max_corr,
                                    power_estimate_H0_vs_H5_dist_corr,
                                    power_estimate_H0_vs_H5_MIC,
                                    power_estimate_H0_vs_H5_XICOR,
                                    power_estimate_H0_vs_H5_gR2)

colnames(power_estimates_H0_H5) <- c("sigma_squared", "pearson",
                                     "spearman", "kendall", "max_corr",
                                     "dist_corr", "MIC", "XICOR", "gR2")

ggplot(power_estimates_H0_H5, aes(x = sigma_squared)) +
  geom_line(aes(y = pearson, color = "Pearson")) +
```

```
  geom_line(aes(y = spearman, color = "Spearman")) +
  geom_line(aes(y = kendall, color = "Kendall")) +
  geom_line(aes(y = max_corr, color = "Maximum Correlation")) +
  geom_line(aes(y = dist_corr, color = "Distance Correlation")) +
  geom_line(aes(y = MIC, color = "Maximum Information Coefficient")) +
  geom_line(aes(y = XICOR, color = "Chatterjee's Correlation")) +
  geom_line(aes(y = gR2, color = "Generalized Pearson Correlation Squares")) +
  xlab("Sigma^2") +
  ylab("Power Estimate") +
  labs(color = "Correlation Metric") +
  ggtitle("Power Estimates of Different Correlation Metrics for H0 vs H5")
```



Problem 3: Galton's Peas (20 pts) Following Section 3 of Chatterjee's paper (https://doi.org/10.1080/01621459.2020.1758115) and implement the eight correlations in Problem 2 on the peas dataset in R package psych. Can you reproduce the results in Chatterjee's paper? What conclusions can you draw from the other seven measures?

```
#Installing required packages
install.packages("psych")
```

```
## Installing package into '/home/kvu1702/R/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```
install.packages("psychTools")
```

```
## Installing package into '/home/kvu1702/R/x86_64-pc-linux-gnu-library/4.4'
## (as 'lib' is unspecified)
```

```r
#Loading required libraries and setting the seed
library(psych)
```

```
##
## Attaching package: 'psych'
```

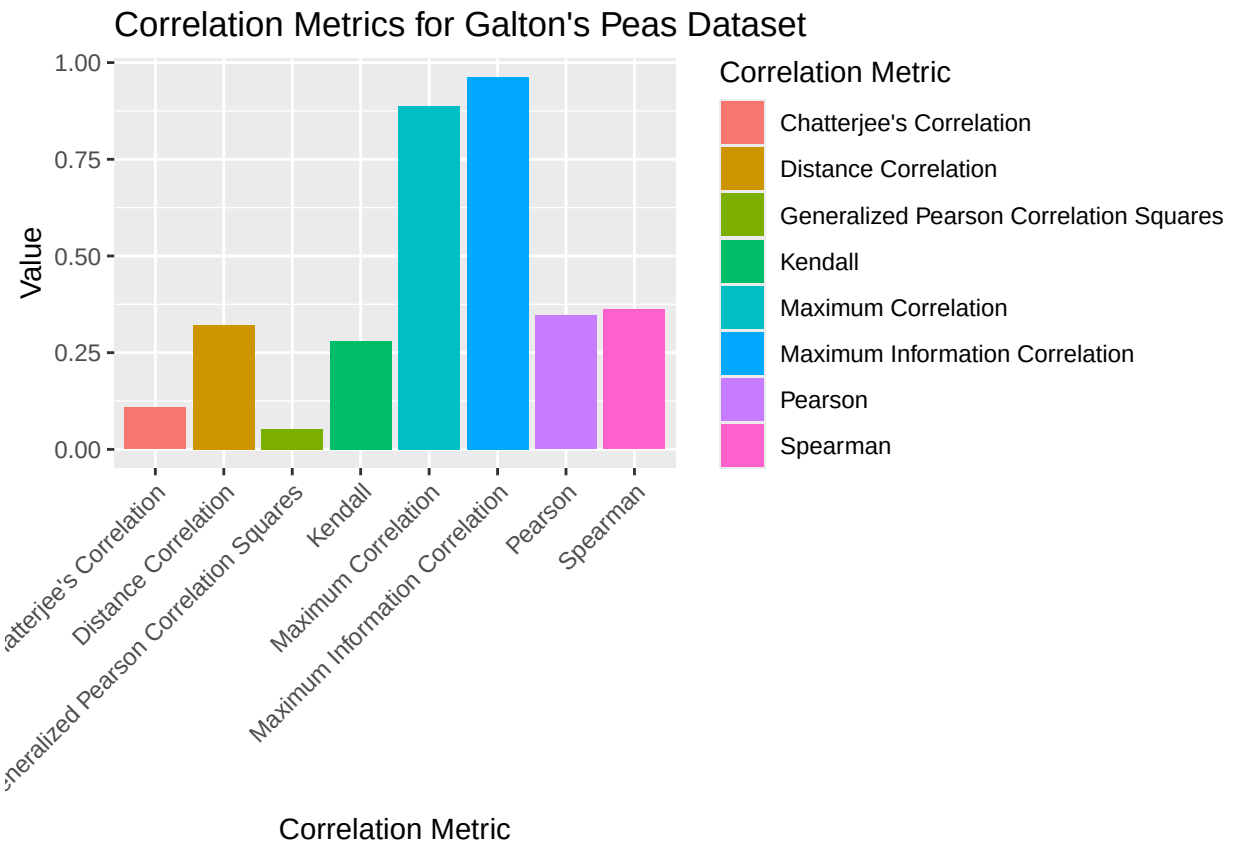
```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
## The following object is masked from 'package:M3C':
##
##     pca
```

```r
library(psychTools)
```

```r
set.seed(1)

#Loading the peas dataset
data(peas)

#Pearson correlation
peas_pearson <- cor(peas[, 1], peas[, 2], method = "pearson")

#Spearman correlation
peas_spearman <- cor(peas[, 1], peas[, 2], method = "spearman")

#Kendall correlation
peas_kendall <- cor(peas[, 1], peas[, 2], method = "kendall")

#Maximal correlation
peas_max_corr <- ace(peas[, 1], peas[, 2])$rsq

#Distance correlation
peas_dist_corr <- dcor(peas[, 1], peas[, 2])

#Maximal information coefficient
peas_MIC <- cstats(matrix(peas[, 1]), matrix(peas[, 2]))[3]

#Chatterjee's correlation
peas_XICOR <- calculateXI(peas[, 1], peas[, 2])

#Generalized Pearson correlation squares

#Deduplicating data in the dataset that are causing errors
peas_unique <- peas[!duplicated(peas), ]
peas_gR2 <- gR2(peas_unique[, 1], peas_unique[, 2], inference = TRUE,
                mc.cores = 1, verbose = FALSE)$estimate

#Plotting the correlation metrics
peas_correlation <- data.frame(correlation = c("Pearson", "Spearman",
                                               "Kendall", "Maximum Correlation",
                                               "Distance Correlation",
                                               "Maximum Information Correlation",
                                               "Chatterjee's Correlation",
```

```
                                                "Generalized Pearson Correlation Squares"),
                                value = c(peas_pearson, peas_spearman,
                                          peas_kendall, peas_max_corr,
                                          peas_dist_corr, peas_MIC,
                                          peas_XICOR, peas_gR2))
ggplot(peas_correlation, aes(x = correlation, y = value,
                             fill = correlation)) +
  geom_bar(stat = "identity") +
  xlab("Correlation Metric") +
  ylab("Value") +
  ggtitle("Correlation Metrics for Galton's Peas Dataset") +
  labs(fill = "Correlation Metric") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
peas_correlation
```

```
##                                correlation      value
## 1                                  Pearson 0.34633188
## 2                                 Spearman 0.36159547
## 3                                  Kendall 0.27936432
## 4                      Maximum Correlation 0.88738330
## 5                     Distance Correlation 0.32164449
## 6          Maximum Information Correlation 0.96372979
## 7                 Chatterjee's Correlation 0.10800856
## 8 Generalized Pearson Correlation Squares 0.05265166
```