stat_ex_2

Gil Shiloh - 315440230, Dean Tesler - 316167105

26 4 2022

lab 2

```
# packages
library(ggplot2)
library(dplyr)
library(corrplot)
library(tidyr)
library(tidyverse)
library(psych)
library(zoo)
library(ggrepel)
library(mvtnorm)
library(gridExtra)
require(stats)
require(dendextend)
library(resample)
library(shiny)
```

1.1

1

a function that samples the first 10 coordinates of each μj

```
set.seed(123)

mu_i <- function(){mu_i <- rnorm(n = 10)
  return(mu_i)}
```

2

a function that samples a datasets of dimension 90 × p

```
mu_sim <- function(mu_1, mu_2, mu_3, p, sig){
  res <-as.data.frame(matrix(0, 90, p))
  for (i in seq(1:90)){
    if (i < 21){
      res[i,] = rmvnorm(n=1, mean = append(mu_1, rep(0, p-10)), sigma = diag(rep(sig, p)))}
    if((i >20) &(i < 51)){
      res[i,] = rmvnorm(n=1, mean = append(mu_2, rep(0, p-10)), sigma = diag(rep(sig, p)))}
    if (i > 50){
      res[i,] = rmvnorm(n=1, mean = append(mu_3, rep(0, p-10)), sigma = diag(rep(sig, p)))}}
  return(as.matrix(res))}
```

3

a function that computes the accuracy of a given clustering result based on the known components.

```
# from moodle tirgul 4 - pages 11-12
accuracy <- function(sample_mnist, mnist_kmeans){
  Mode <- function(x){
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]}

true_label <- unname(unlist((sample_mnist)))
cluster <- do.call(cbind, list(by(true_label, mnist_kmeans[["cluster"]], Mode)))
cluster <- cbind(rownames(cluster), cluster)
colnames(cluster) <- c("clus_center","cluster_label")
acc_table <- data.frame(true_label = true_label, clus_center = mnist_kmeans[["cluster"]])
acc_table <- merge(x = acc_table, y = cluster)
return(mean(acc_table$true_label == acc_table$cluster_label))}
```

4

a wrapper for the K-means algorithm that inputs a data-set and the set of true-labels, and outputs the accuracy and the run-time.

```
pred <- function(data,true_labels){
  b_time <- Sys.time()
  kmeans_res <- kmeans(data, 3)
  accuracy <- accuracy(true_labels, kmeans_res)
  t_time <- Sys.time() - b_time
  return(c(accuracy, t_time))}
```

1.2

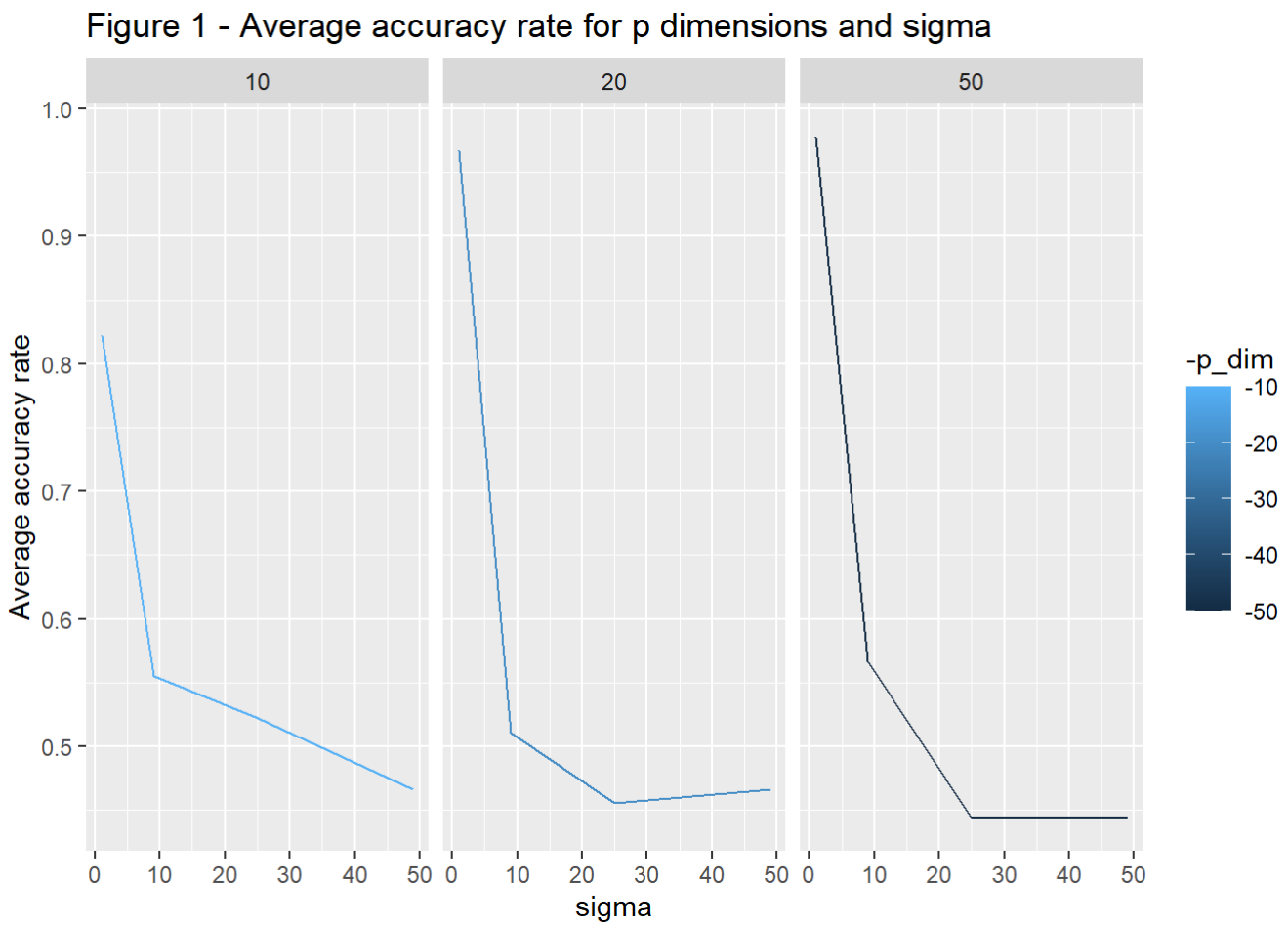
1

compute the average accuracy and the standard-error

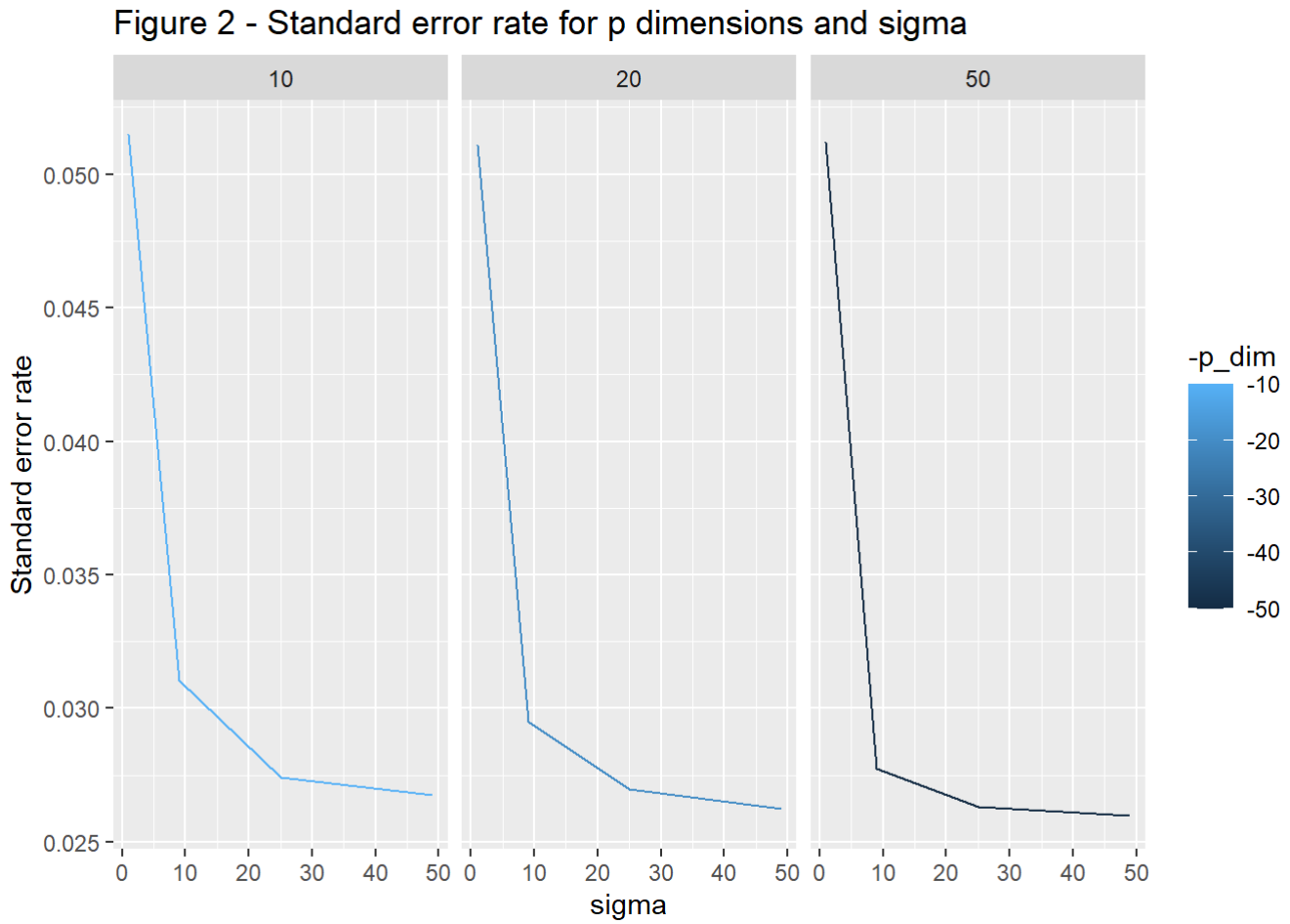
```
acc_fun <- data.frame(p_dim = NA, sigma = NA, ave_acc = NA, std_dev = NA)
data_time <- data.frame(p_dim = NA, sigma = NA, time = NA)

for (i in c(10, 20,50)){
  for (j in c(1,9,25,49)){
    temp = c()
    for (k in seq(80)){
      data <- mu_sim(mu_i(), mu_i(), mu_i(), i, j)
      temp <- append(temp,pred(data, rep(c(1,2,3), times = c(20,30,40))))
      data_time <- rbind(data_time,c(i, j, temp[2]))}
    acc_fun <- rbind(acc_fun,c(i, j,mean(temp[1]), sd(temp)/sqrt(80)))}}

acc_fun <- na.omit(acc_fun)
rownames(acc_fun) <- NULL
acc_fun
```

```
##    p_dim sigma   ave_acc    std_dev
## 1     10     1 0.8222222 0.05149594
## 2     10     9 0.5555556 0.03102757
## 3     10    25 0.5222222 0.02740865
## 4     10    49 0.4666667 0.02676590
## 5     20     1 0.9666667 0.05107082
## 6     20     9 0.5111111 0.02952209
## 7     20    25 0.4555556 0.02698841
## 8     20    49 0.4666667 0.02624876
## 9     50     1 0.9777778 0.05119989
## 10    50     9 0.5666667 0.02774330
## 11    50    25 0.4444444 0.02631305
## 12    50    49 0.4444444 0.02599900
```

```
ggplot(acc_fun, aes(x=sigma, y=ave_acc, colour = -p_dim))+
  facet_wrap(vars(p_dim)) +
  geom_line()+ ggtitle("Figure 1 - Average accuracy rate for p dimensions and sigma") +labs(
 x = "sigma", y = "Average accuracy rate")
```



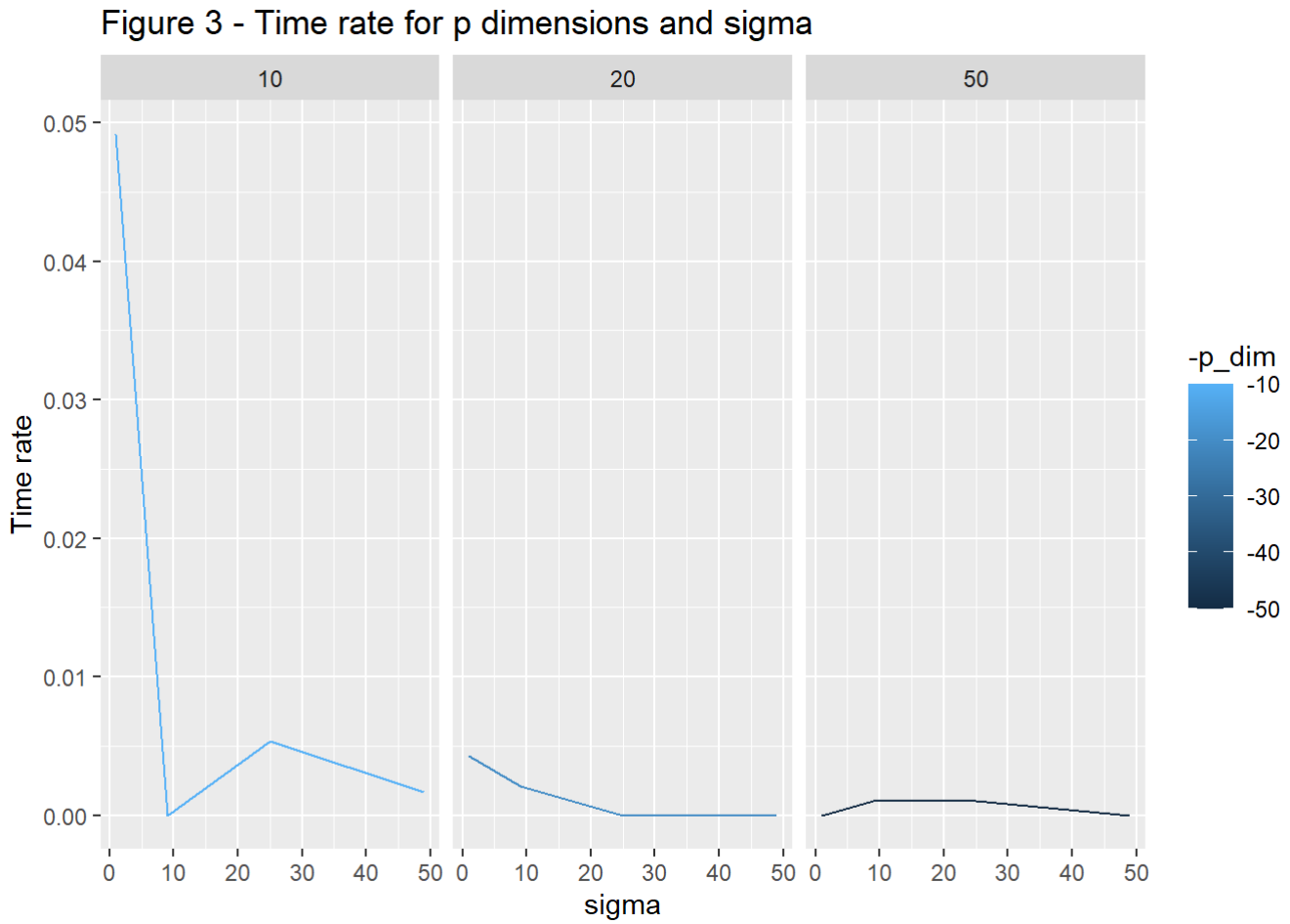

```
ggplot(acc_fun, aes(x=sigma, y=std_dev
, colour = -p_dim))+
  facet_wrap(vars(p_dim)) +
  geom_line()+ ggtitle("Figure 2 - Standard error rate for p dimensions and sigma") +labs( x
 = "sigma", y = "Standard error rate")
```



2

a figure describing run-time

```
data_time <- na.omit(data_time)
ggplot(data_time, aes(x=sigma, y=time, colour = -p_dim))+
  facet_wrap(vars(p_dim)) +
  geom_line()+ ggtitle("Figure 3 - Time rate for p dimensions and sigma") +labs( x = "sigma",
y = "Time rate")
```



3

Briefly, discuss the effect of increasing p (dim) and increasing sigma^2 on accuracy and run-time.

for accuracy: from figure 1 we can see that where the sigma is the lowest (1) the accuracy is increasing as the p dimensions is increasing. Further more we can see that Where the variance increasing we can see at all the dimensions that the accuracy of the prediction is decreasing. From figure 2 we can understand that the p dimension is not having a big impact on the standard error rate, so the sigma is the main focus here and we can see that as the sigma increasing the standard error rate is decreasing.

From figure 3 (rantime) we can see that when the variance is low and low dimensions the run time is the longest which is because of the stopping criterion of the K-means algorithm, and as the p dimension is increasing and the variance is increasing the runtime is getting shorter since its reaches the stopping criterion faster.

2

In this part we will explore how socio-economical similarity between cities relates to patterns in the spread and effect of the corona-virus. We will use the demographic dataset csb demographics.txt that is used to create the socio-economic ranking by the Israeli Statistical Bureau (ISB); each row is a town or “moatza mekomit”, and the variables represent some demographic property. We will compare the demographics to the Covid-19 statistics data-set by town found in covid towns.csv 1 . Both are on Moodle. For the Covid19 statistics, we processed the file to produce the monthly number of verified cases, recovered, deaths and diagnostic tests in each town.

1

Randomly choose a set of 20 cities described in the ISB (demographics) data sets. Identify these cities in the corona-virus data-sets.

```
data_demo <- read.delim("cbs_demographics.txt") # loading the data
rownames(data_demo) <- data_demo$village
data_covid <- read.csv("covid_towns.csv",encoding="UTF-8")
code_data <- read.csv("code_name_mapping.csv", encoding="UTF-8")
colnames(code_data)<- c("City_Code", "City_Name")
data_covid <- merge(code_data, data_covid, by = "City_Code") %>% select(-c(City_Name.y, City_
Code, X))
colnames(data_covid)[1] <- "City_Name"
rownames(data_covid) <- data_covid$village
```

```
set.seed(123)
# Sample 20 random rows
df_demogra_sample <- sample_n(data_demo[data_demo$village %in% data_covid$City_Name,], 20,)
# Identify and save these cities in the corona-virus data-sets
df_sample <- data_covid[data_covid$City_Name %in% df_demogra_sample$village,]
print(df_demogra_sample)
```

```
##                population median_age dependency_ratio pct_4family schoolyr_avrg
## GANNE TIQWA     14959.137         33         82.28767    9.270833     13.804204
## MESHHED          7594.093         22         98.60828   34.167386     11.083102
## YAVNE           36506.285         31         67.32890    9.166149     12.255597
## EFRAT            7911.939         22         99.85025   31.641554     14.094694
## KAFAR QARA      17095.179         24         87.06758   20.295056     11.882008
## HURA            18271.225         14        183.15947   55.314010      9.820403
## TAMRA           30854.946         23         90.46224   18.269231     10.844632
## RAME             7309.054         28         78.33417   12.489233     11.675845
## HURFEISH         5873.912         25         81.03449   13.403141     12.015693
## FASSUTA          2964.026         32         68.90269    5.896226     12.094105
## QIRYAT SHEMONA  23094.003         31         65.93100   10.539846     11.939518
## SEGEV-SHALOM     8493.434         15        168.33739   51.784232      7.807830
## BAT YAM        128561.475         39         74.24894    6.296852     10.741450
## GHAJAR           2360.635         22         93.51318   21.739130      9.956397
## OMER             7265.596         37         94.57827   22.469410     14.897199
## LEHAVIM          6282.022         31         72.56832   10.606061     15.423755
## QIRYAT YAM      38526.483         40         70.73617    5.452586     11.437799
## KUSEIFE         18076.250         15        160.10587   48.513674     10.174573
## EILABUN          5248.898         27         77.40674   12.722298     11.786926
## ELAT            48121.614         31         58.31920    7.449773     10.618951
##                pct_degree pct_earner pct_wmn_noincome pct_incometwice
## GANNE TIQWA     47.465805   70.39134         14.57178       26.823026
## MESHHED         12.855783   51.86348         59.57538        3.365005
## YAVNE           23.649483   72.34078         14.66966       12.778656
## EFRAT           55.743047   71.36446         17.15956       17.687434
## KAFAR QARA      20.845659   63.53974         38.15908        5.997028
## HURA            11.612254   45.25626         69.08133        3.469068
## TAMRA           12.961748   54.59750         50.75587        2.765054
## RAME            22.395179   55.76123         36.70485        6.921188
## HURFEISH        21.689965   56.03458         43.74261        8.947021
## FASSUTA         25.938028   60.93153         25.54109        7.735388
## QIRYAT SHEMONA  18.913144   69.84937         14.08263        7.045782
## SEGEV-SHALOM     6.911858   45.13709         62.38656        2.068349
## BAT YAM         18.735121   63.90879         16.42651        6.153100
## GHAJAR          14.984445   58.17945         60.15270        5.804722
## OMER            56.910330   66.82680         14.65625       35.682776
## LEHAVIM         65.194877   71.45907         10.87254       36.094879
## QIRYAT YAM      20.514313   62.00902         16.63597        7.968973
## KUSEIFE         12.175708   46.55063         68.09255        2.275004
## EILABUN         24.289709   64.14990         27.95902        7.161706
## ELAT            13.828558   77.94060         12.50546        8.207352
##                pct_belowminimum pct_recepients   income motorization avgvehicle
## GANNE TIQWA            29.98775      0.9562985 6498.960     43.43561   1412.290
## MESHHED                51.09259      3.3973772 1973.924     23.78569   1456.117
## YAVNE                  37.90127      3.1328214 4836.673     36.85593   1327.782
## EFRAT                  43.37458      0.3412564 4572.384     30.18084   1348.193
## KAFAR QARA             44.13115      1.7139335 2934.639     33.93993   1297.218
## HURA                   48.92396      4.4386734 1291.058     13.82616   1332.394
## TAMRA                  51.09772      5.3313981 2057.767     28.72439   1246.434
## RAME                   48.00254      4.0634532 2838.457     33.82650   1279.239
## HURFEISH               46.54071      1.8045894 2849.764     27.39770   1290.227
## FASSUTA                41.39848      3.3737893 3402.368     37.31772   1304.435
## QIRYAT SHEMONA         42.58721      5.0359394 3826.042     30.69897   1276.937
## SEGEV-SHALOM           50.32085      6.4284953 1228.270     14.32819   1261.311
```

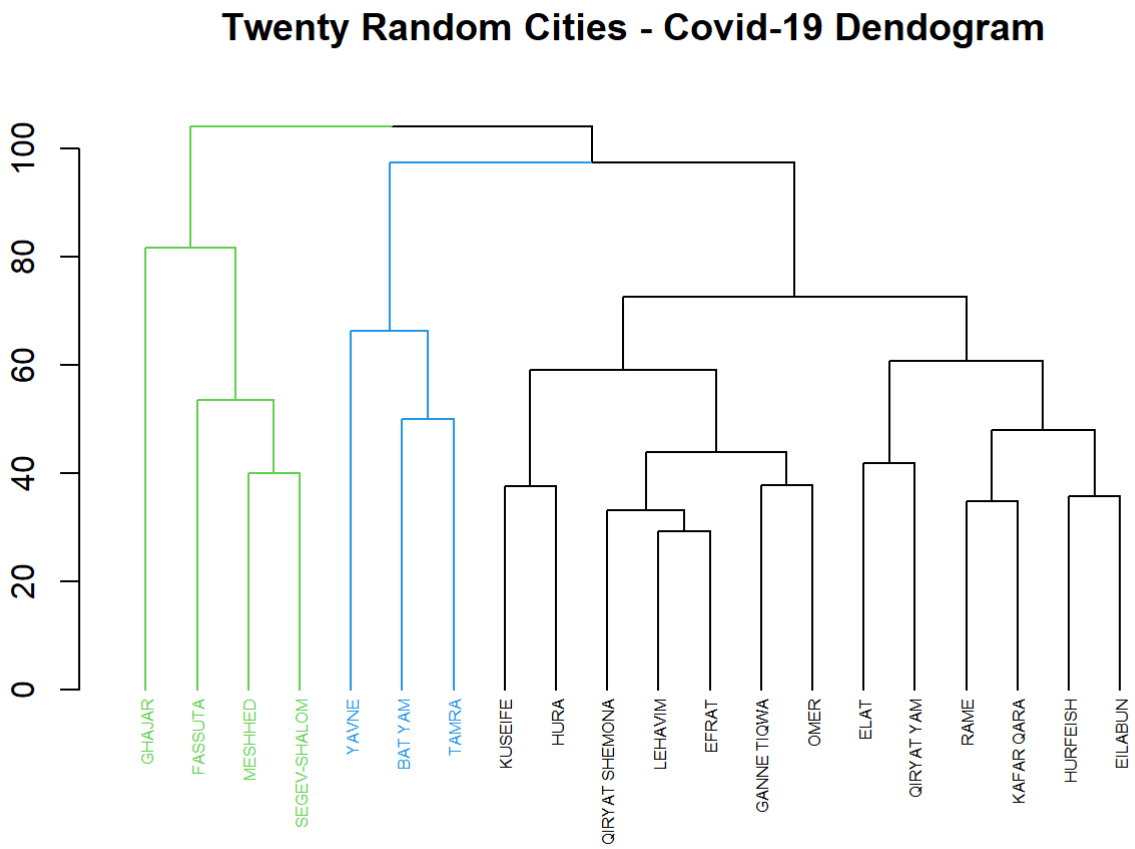
```
## BAT YAM                39.42566      7.7729179 3693.262     25.52662   1272.837
## GHAJAR                 46.19049      3.4312808 2414.657     20.57773   1453.617
## OMER                   28.18838      0.5230128 8485.050     53.73593   1570.388
## LEHAVIM                29.57186      0.2546950 9103.550     49.62189   1542.043
## QIRYAT YAM             41.41191      9.9386181 3788.054     27.66509   1283.060
## KUSEIFE                50.52504      5.0563584 1284.161     14.27361   1262.288
## EILABUN                46.41456      3.9627363 3309.721     35.64326   1313.537
## ELAT                   36.80978      3.1815226 4577.048     28.35299   1325.759
##                 avgabroad        village
## GANNE TIQWA     7.9682776    GANNE TIQWA
## MESHHED         0.5969880        MESHHED
## YAVNE           4.8102405          YAVNE
## EFRAT           9.2459547          EFRAT
## KAFAR QARA      1.4048299     KAFAR QARA
## HURA            0.2416370           HURA
## TAMRA           0.8722923          TAMRA
## RAME            2.0397056           RAME
## HURFEISH        1.0284285       HURFEISH
## FASSUTA         2.1553459        FASSUTA
## QIRYAT SHEMONA  2.9062120 QIRYAT SHEMONA
## SEGEV-SHALOM    0.2015613   SEGEV-SHALOM
## BAT YAM         5.1996365        BAT YAM
## GHAJAR          0.3807844         GHAJAR
## OMER           11.8539305           OMER
## LEHAVIM        11.0570641        LEHAVIM
## QIRYAT YAM      4.4298004     QIRYAT YAM
## KUSEIFE         0.1840262        KUSEIFE
## EILABUN         2.4689371        EILABUN
## ELAT            5.3252637           ELAT
```

2

Construct a hierarchical tree for the covid data

```
# Normalize the data by the size of the population
df_scale <- df_sample
rownames(df_scale) <- df_scale$City_Name
df_scale[c(2:100)] <- lapply(df_scale[c(2:100)], function(x) if(is.numeric(x)) c(x/df_demogra
_sample$population) else (x))
df_scale <- df_scale %>% select(-City_Name) %>% select(-c(101:105))

# by scale
demogra_scale <- df_demogra_sample
demogra_scale[c(2:15)] <- lapply(demogra_scale[c(2:15)], function(x) c(scale(x)))
demogra_scale <- demogra_scale %>% select(-village)
```

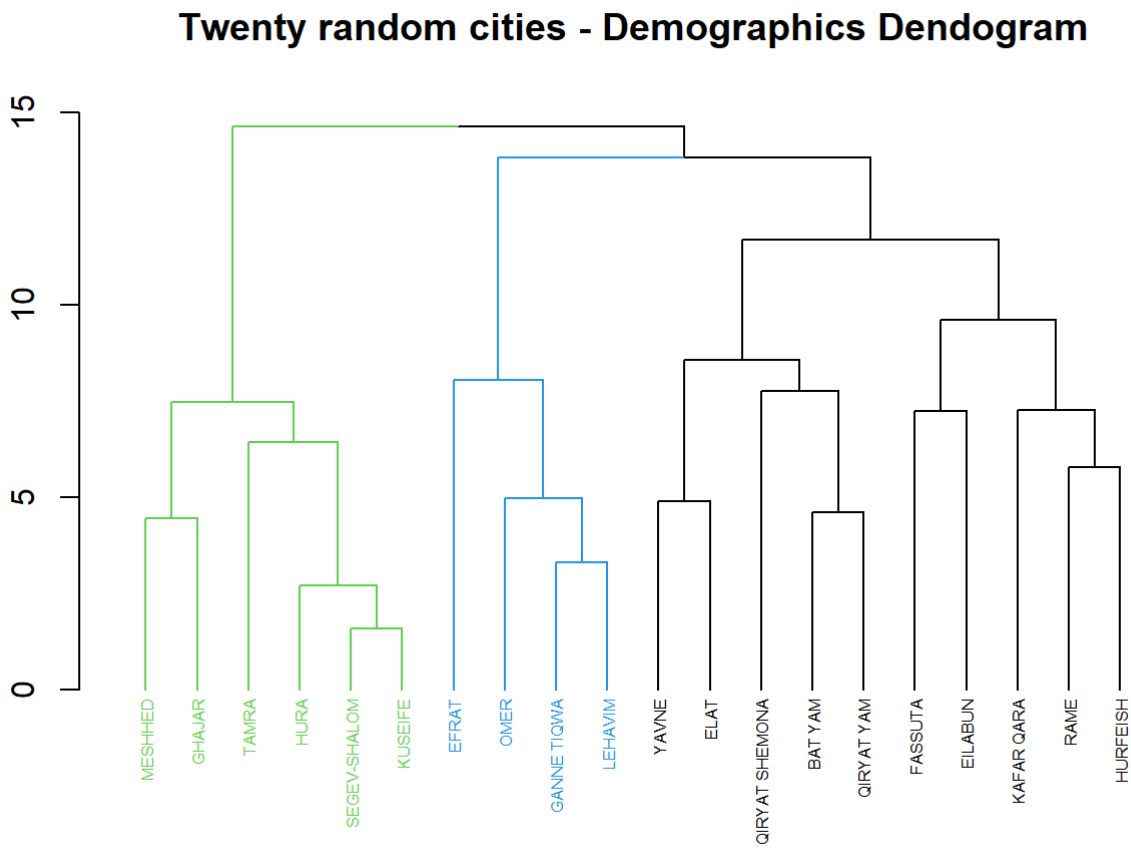
```
data_covid_samp_dist <- dist(df_scale, method ="canberra")
hir_clust_tree <- hclust(data_covid_samp_dist, method = "complete")
covid_dend <- as.dendrogram(hir_clust_tree)
covid_dend <- covid_dend %>% set("labels_cex", 0.5) %>% set("branches_k_color", value = c(3,4
,1), k = 3) %>% set("labels_col", value = c(3,4,1), k = 3)
plot(covid_dend, main = "Twenty Random Cities - Covid-19 Dendogram")
```



3

Construct a hierarchical tree for the demographic data.

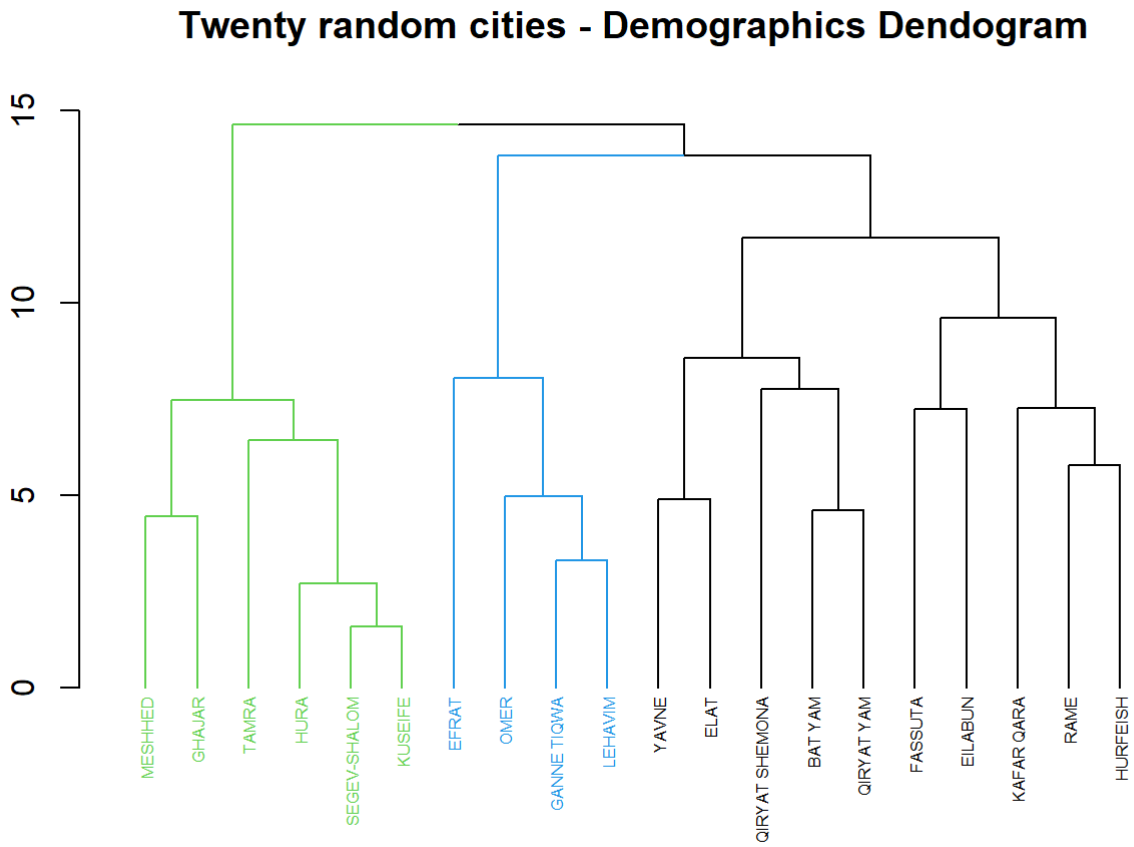
```
data_demo_samp_dist <- dist(demogra_scale, method ="canberra")
hir_clust_tree <- hclust(data_demo_samp_dist, method = "complete")
demo_dend <- as.dendrogram(hir_clust_tree)
demo_dend <- demo_dend %>% set("labels_cex", 0.5) %>% set("branches_k_color", value = c(3,4,1
), k = 3) %>% set("labels_col", value = c(3,4,1), k = 3)
plot(demo_dend, main = "Twenty random cities - Demographics Dendogram")
```



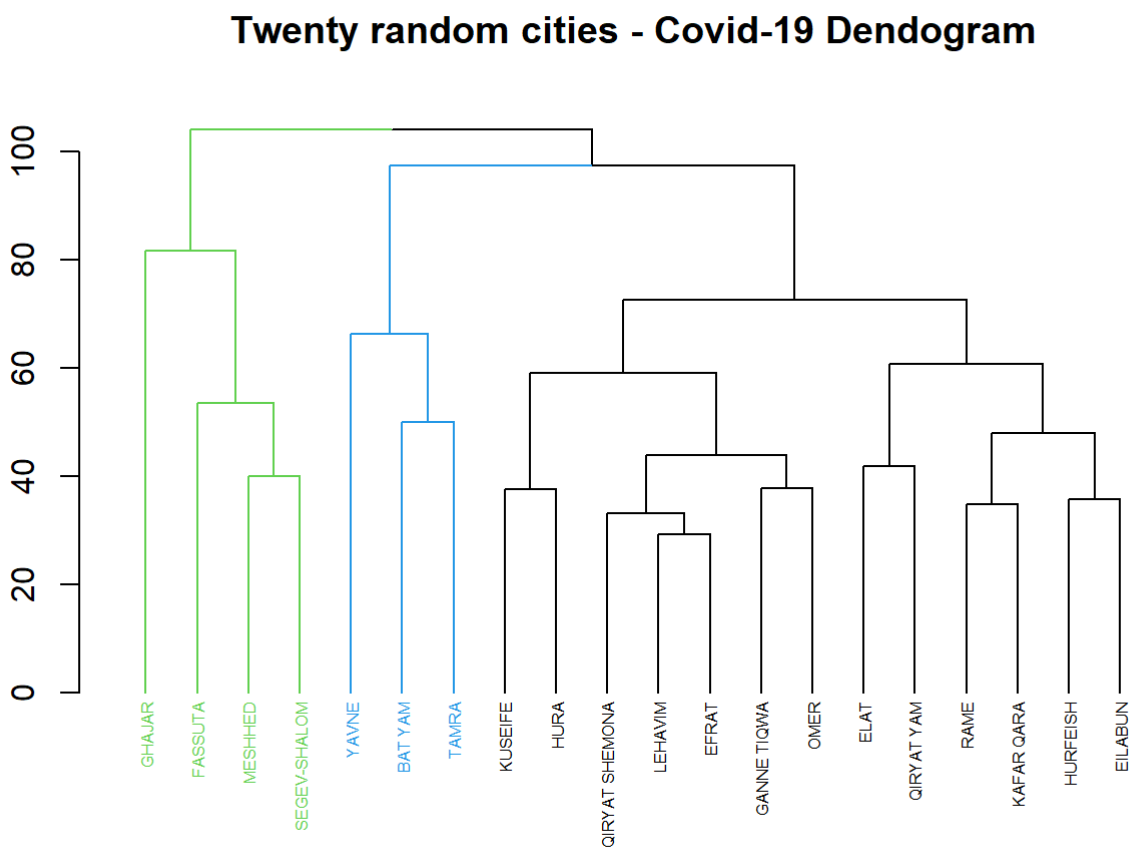
4

Compare the two hierarchies. Comment on similarities and differences.

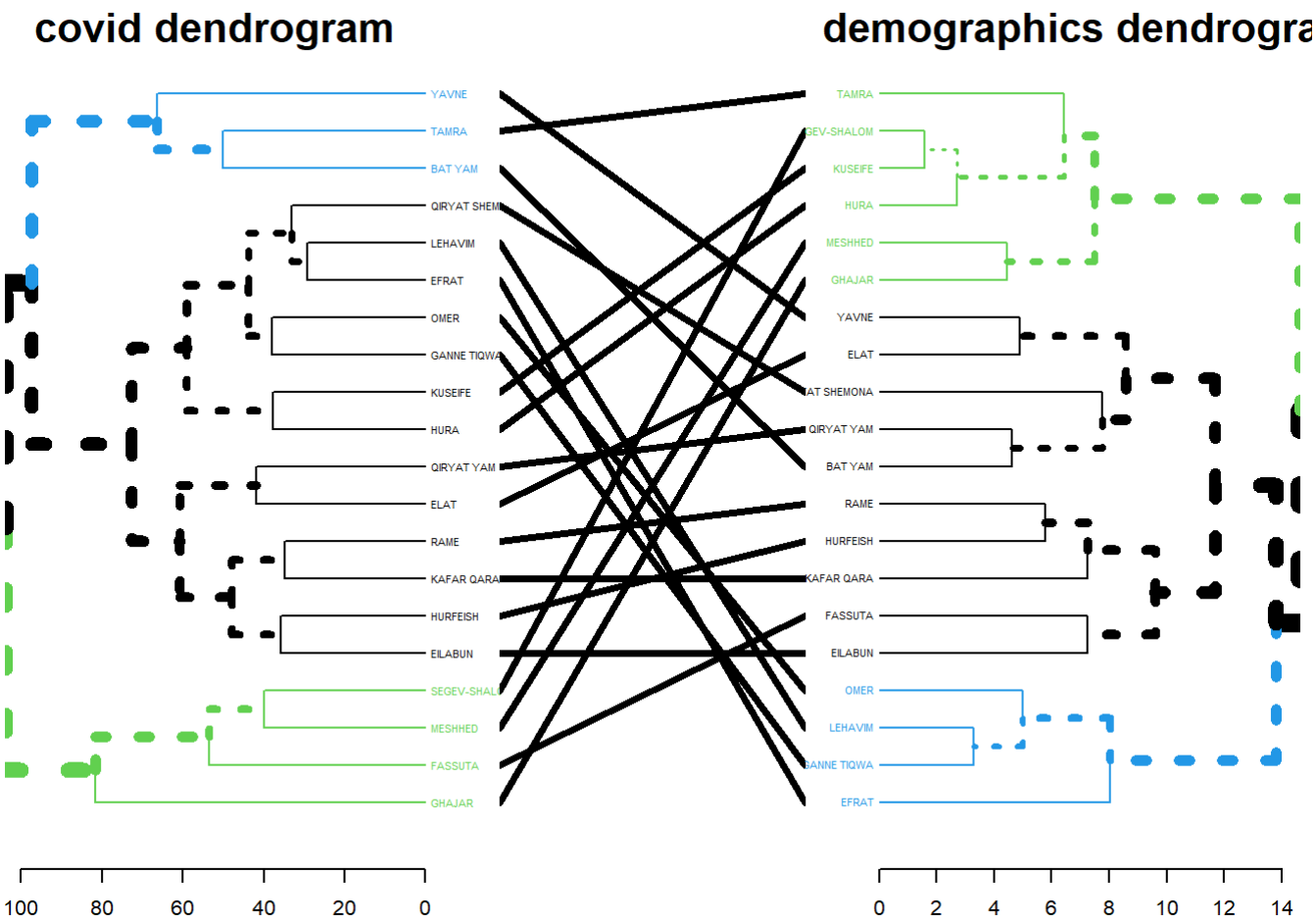
```
plot(demo_dend, main = "Twenty random cities - Demographics Dendogram")
```




```
plot(covid_dend, main = "Twenty random cities - Covid-19 Dendogram")
```



```
dl <- dendlist("covid dendrogram" = covid_dend, "demographics dendrogram" = demo_dend)
tanglegram(dl, sort = TRUE, common_subtrees_color_branches = TRUE)
```



by using a tanglegram plot we are getting comparison between 2 dendorograms with the same labes connected by lines. similarities between the two hierarchies: the main similarity is that both of them are with the same names Differences: As we can see each graph gives a different groups of cities, the sub-tree are different and the covid-19 dendorogram is more balanced compared to the demographics dendorogram.

5

Choose a similarity score for the two trees. You can base your score on one of the scores implemented in the dendextend package, including Baker's Gamma, the cophenetic correlation or the Fowlkes-Mallows (Bk) index.

```
print(paste('The Baker Index Score for the dendograms:',cor_bakers_gamma(demo_dend, covid_den
d)))
```

```
## [1] "The Baker Index Score for the dendograms: 0.243789161416997"
```

Baker's Gamma (see reference) is a measure of accosiation (similarity) between two trees of heirarchical clustering (dendrograms). It is calculated by taking two items, and see what is the heighst possible level of k (number of cluster groups created when cutting the tree) for which the two item still belongs to the same tree. That k is returned, and the same is done for these two items for the second tree. There are n over 2 combinations of such pairs of items from the items in the tree, and all of these numbers are calculated for each of the two trees. Then, these two sets of numbers (a set for the items in each tree) are paired according to the pairs of items compared, and a spearman correlation is calculated. The value can range between -1 to 1. With near 0 values meaning that the two trees are not statistically similar. For exact p-value one should result to a

permutation test. One such option will be to permute over the labels of one tree many times, and calculating the distriubtion under the null hypothesis (keeping the trees topologies constant). Notice that this measure is not affected by the height of a branch but only of its relative position compared with other branches.

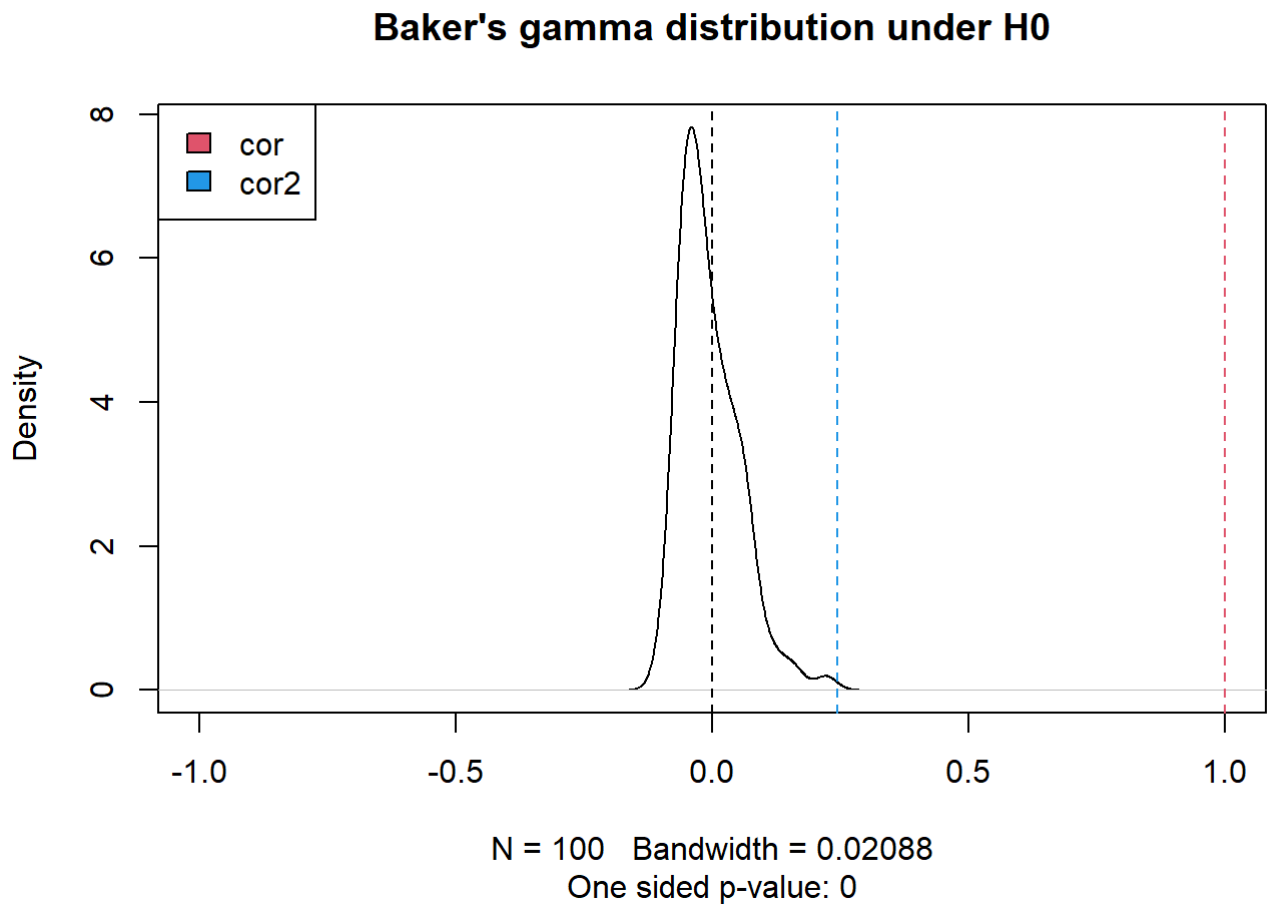
6

Find a background distribution for this score, assuming the labels of the trees are completely unrelated.

```
set.seed(23235)
the_cor <- cor_bakers_gamma(demo_dend, demo_dend)
the_cor2 <- cor_bakers_gamma(demo_dend, covid_dend)
R <- 100
cor_bakers_gamma_results <- numeric(R)
dend_mixed <- demo_dend
for(i in 1:R) {
   dend_mixed <- sample.dendrogram(dend_mixed, replace = FALSE)
   cor_bakers_gamma_results[i] <- cor_bakers_gamma(demo_dend, dend_mixed)
}
plot(density(cor_bakers_gamma_results),
     main = "Baker's gamma distribution under H0",
     xlim = c(-1,1))
abline(v = 0, lty = 2)
abline(v = the_cor, lty = 2, col = 2)
abline(v = the_cor2, lty = 2, col = 4)
legend("topleft", legend = c("cor", "cor2"), fill = c(2,4))
round(sum(the_cor2 < cor_bakers_gamma_results)/ R, 4)
```

```
## [1] 0
```

```
title(sub = paste("One sided p-value:",
                  sum(the_cor < cor_bakers_gamma_results)/ R
                  ))
```




```
print(paste('The one sided p-value:',sum(the_cor < cor_bakers_gamma_results)/ R))
```

```
## [1] "The one sided p-value: 0"
```

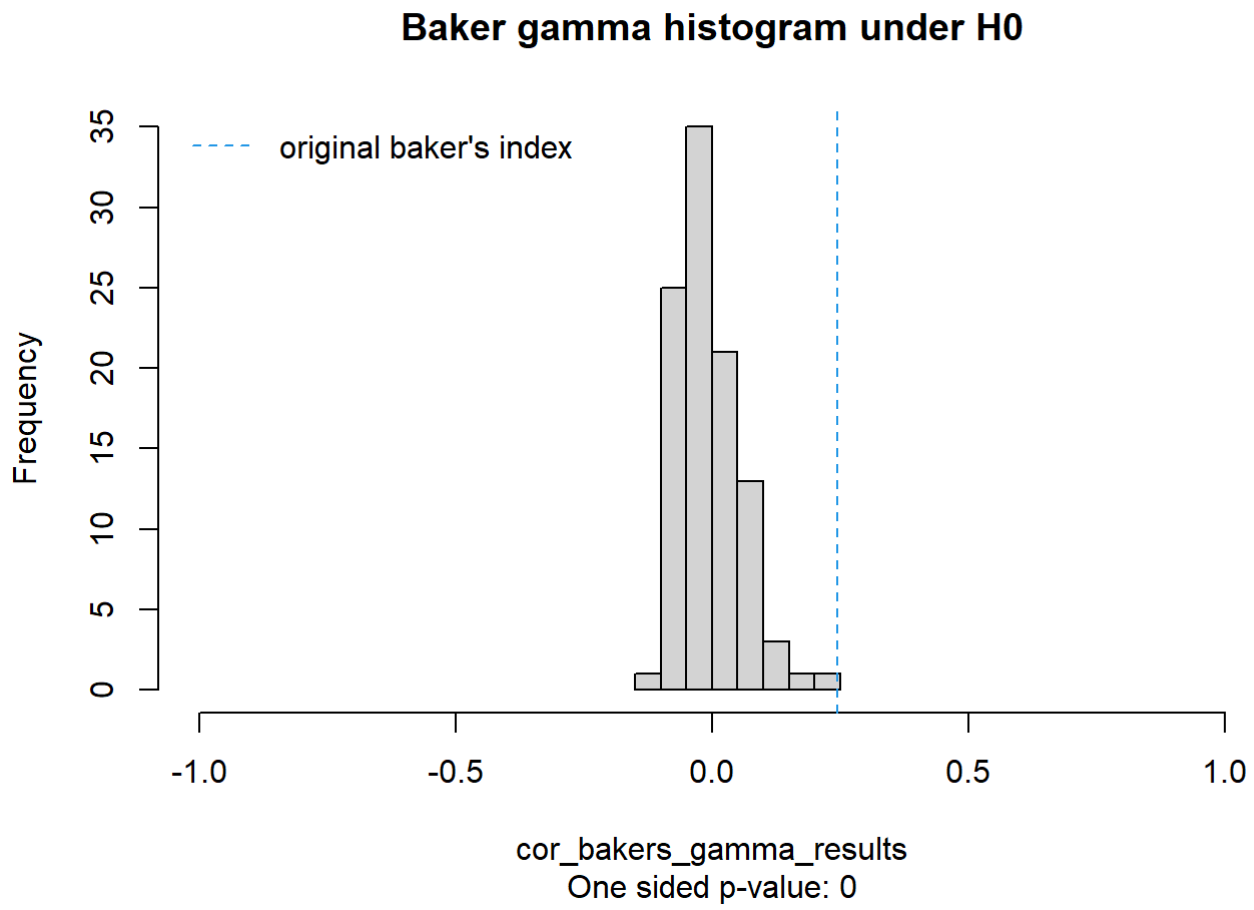
7

Display the results as a histogram approximating the null-distribution scores.

```
hist(cor_bakers_gamma_results,
     main = "Baker gamma histogram under H0",xlim = c(-1,1),breaks=6)
abline(v = the_cor2, lty = 2, col = 4)
legend("topleft",inset = 0.005, legend = "original baker's index", col = 4, lty = 2,     box.
lty = 0,xpd=T)
round(sum(the_cor2 < cor_bakers_gamma_results)/ R, 4)
```

```
## [1] 0
```

```
title(sub = paste("One sided p-value:",  round(sum(the_cor2 < cor_bakers_gamma_results)/ R, 4
)))
```



8

Explain your results in light of the null hypothesis you were testing.

Our null hypothesis was that the labels of the trees are completely unrelated. - H0: Baker Index = 0 | H1: Baker Index !=0

As from the pervious section and the permutation test the p-value is less than 0.05 (0) thus we reject H0. From that we can understand that the hierarchical trees match statistically. And the correlation is slightly proving the results.

3

We coded KMEANS code from scratch without using the k_means library algorithm. Then using the 'shiny' app we made an app so we could see the clusters and iterations in a visualize and interactive way.

```
gen_df <- read.delim("gtex.txt", skip = 2 ,row.names = c(1), header = TRUE)
log_gtex <- log(as.matrix(gen_df[2:54]) + 1)
sd_gtex <- as.data.frame(apply(log_gtex, 1, sd))
sd_gtex <- sd_gtex %>% top_n(200)
```

```
## Selecting by apply(log_gtex, 1, sd)
```

```
log_gtex <- as.data.frame(log_gtex[rownames(sd_gtex),])
k_means_df <- log_gtex
```

```
distance <- function(a,b) sqrt(sum((a-b)^2))
`%!in%` <- Negate(`%in%`)
centers <- function(data, k){
  data[sample(nrow(data), k),]
}


k_means = function (k_means_df, k, max_iter=100){

  df_cen <- centers(k_means_df, k)
  k_means_df$cluster <- NA
  k_means_df$error <- NA
  col_remove <- c("cluster", "error")
  wss <- c()
  for(iter in 1:max_iter){
    for (i in 1:nrow(k_means_df)){
      y <- c()
      for (j in 1:k){
         y <- c(y, distance(k_means_df[colnames(k_means_df) %!in% col_remove][i,], df_cen
[j,]))
      }
      k_means_df$cluster[i] = which(y == min(y))
      k_means_df$error[i] = min(y)
    }
    df_2_c <- aggregate(. ~ cluster, k_means_df, mean)
    df_2_c <- df_2_c[colnames(df_2_c) %!in% col_remove]
        wss = c(wss, sum(k_means_df$error))

    if(all(rowSums(df_2_c) == rowSums(df_cen))){
      break
    }

    df_cen <- df_2_c
  }

  return(list(df_cen, k_means_df, wss))

}

RUN <- k_means(log_gtex, 3, 10)
wss <- RUN[[3]]
final_df <- prcomp(x = k_means_df, center = T, scale. = T)
final_df <- as.data.frame(final_df$x[,c(1,2)])
```

```
ui <- fluidPage(
    titlePanel("K-means Gen App"),
    sidebarLayout(
        sidebarPanel(
            sliderInput("clusters",
                        "Number of clusters:",
                        min = 2,
                        max = 10,
                        value = 5),
            sliderInput("max_iter",
                        "Number of iterations:",
                        min = 1,
                        max = 100,
                        value = 10)
        ),
        mainPanel(
           plotOutput("distPlot")
        )
    )
)
server <- function(input, output) {

    output$distPlot <- renderPlot({

        res <- k_means(k_means_df, input$clusters, input$max_iter)

        clust <- res[[2]]
        plot(x=final_df$PC1, y=final_df$PC2, col = as.factor(unlist(clust$cluster)),
             cex=1.3, pch = 1, xlab = "PCA 1", ylab= "PCA 2",
             title(paste0("K Means Plot - Number of clusters: ",input$clusters),
             cex.main = 2, col.main= "blue"))
    })
}
shinyApp(ui = ui, server = server)
```

Shiny applications not supported in static R Markdown documents