MDS Class Exercise (Q1 and Q2)

Roger Bukuru

2024-02-20

Question 1

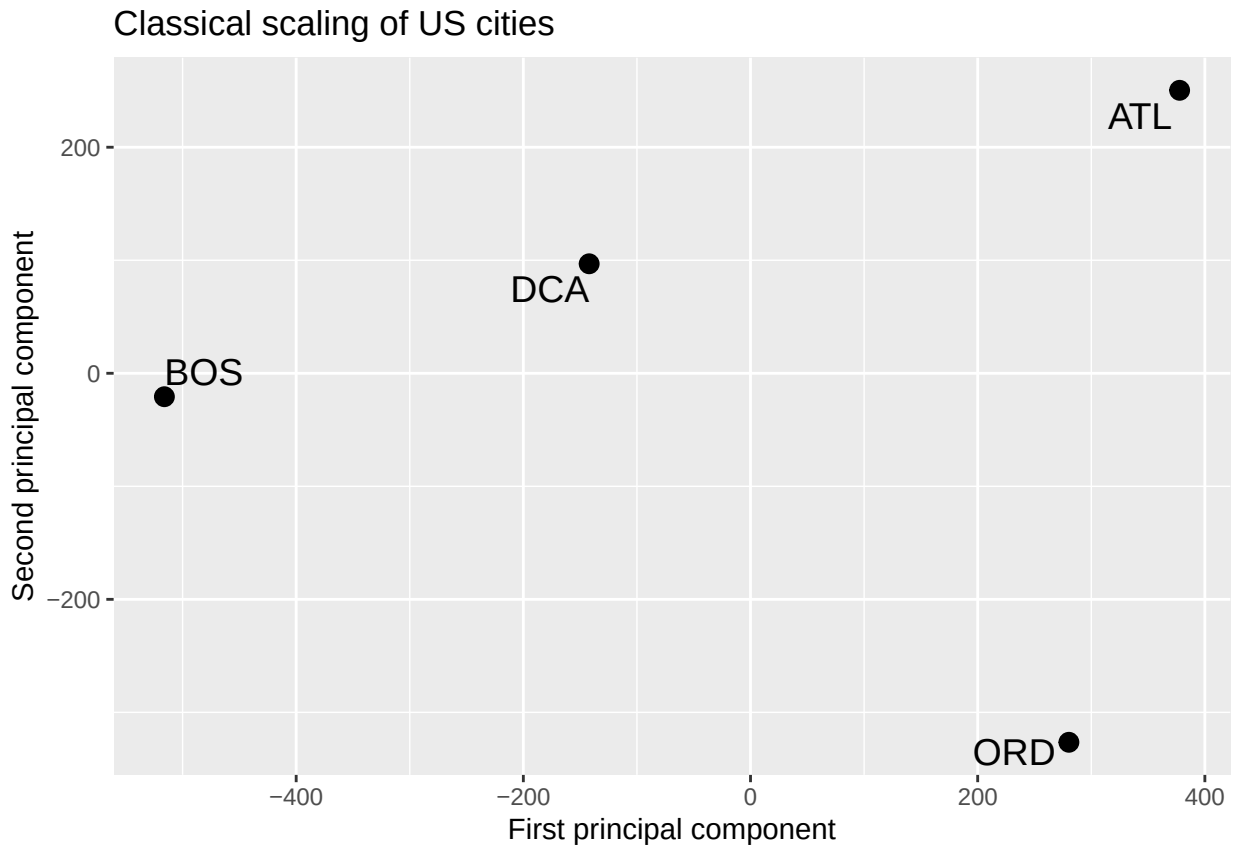
For this exercise we will be using the cities data that is found in the psych package in R. Run the following lines of code to obtain the dissimilarity matrix that we will work with.

a) Using the first 4 (four) cities, ATL, BOS, ORD and DCA. Perform the classical scaling algorithm by first principals to obtain the principals coordinates. Plot the principal coordinates in 2 dimensions. You can use R to create matrices and perform the Eigen value decomposition.

```
rm(list = ls())
library(psych)
library(psychTools)
library(tibble)
library(dplyr)
library(ggplot2)
library(ggrepel)
data(cities)

classical_scaling_first_principals = function(noOfCities = 4,
    rotateMap = FALSE) {
    rotateFactor = -1
    if (!rotateMap) {
        rotateFactor = 1
    }
    proximity_matrix = as.matrix(cities[1:noOfCities, 1:noOfCities])

    # A = -0.5 * sqrt(proximity_matrix)
    A = (-1/2) * proximity_matrix^2

    # B = HAH H = I-1/n11^1
    n = nrow(A)
    H = diag(1, nrow = n) - 1/n * matrix(1, nrow = n, ncol = ncol(A))
    B = H %*% A %*% H

    # Objective function Y = Eigenvectors %*%
    # diag(eigenvalues)
    evd = eigen(B)
    Y = evd$vectors %*% diag(sqrt(evd$values))  # Principal components

    principal_comp = Y %>%
        as.tibble() %>%
```

```
        mutate(city = colnames(proximity_matrix))
    # First two components, represent comps with the most
    # variance in representing the data with comp 1 having
    # the most variance etc.
    two_dim_plot = ggplot(principal_comp, mapping = aes(rotateFactor *
        V1, rotateFactor * V2)) + geom_point(size = 3) + geom_text_repel(aes(label = city),
        size = 5) + labs(x = "First principal component", y = "Second principal component",
        title = "Classical scaling of US cities")
    coord_equal()

    two_dim_plot

}
```

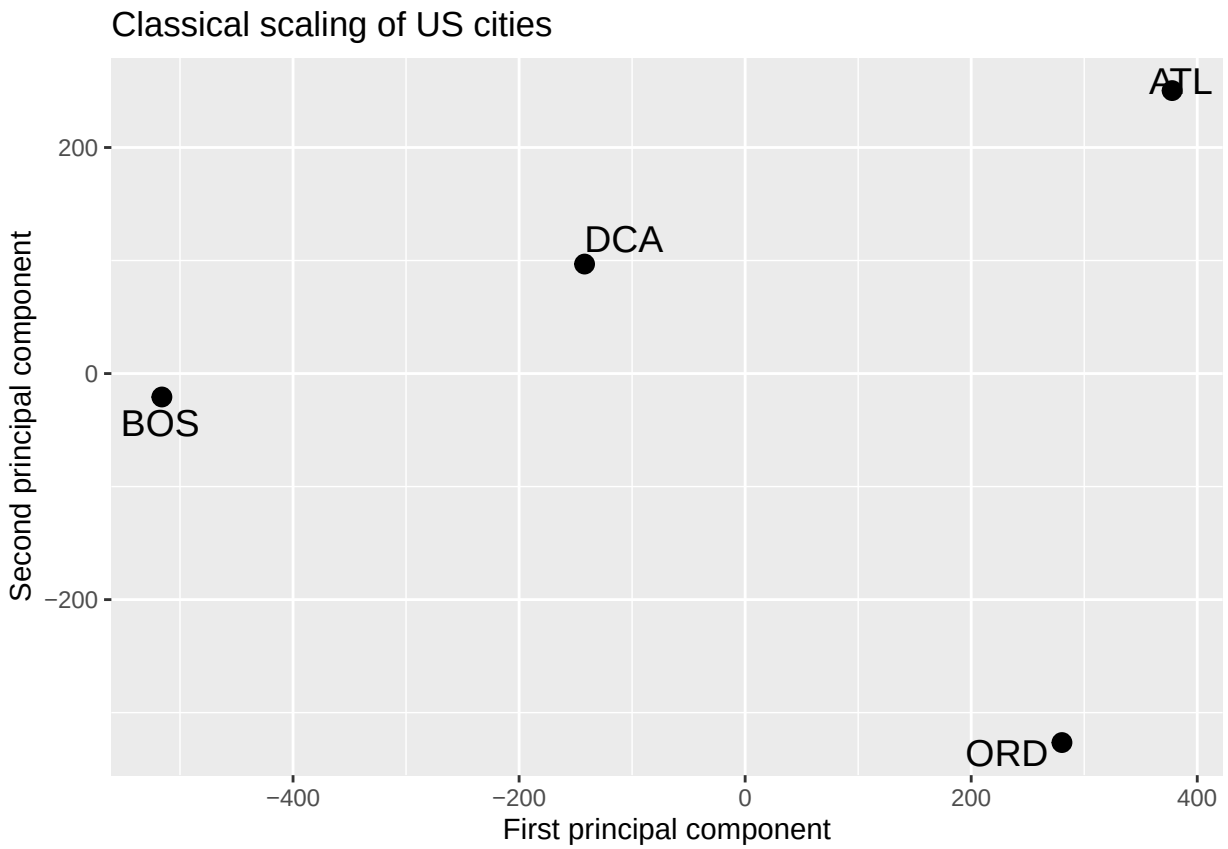
```
classical_scaling_first_principals()
```



b) Use the function cmdscale in R to check check your answer in part (a).

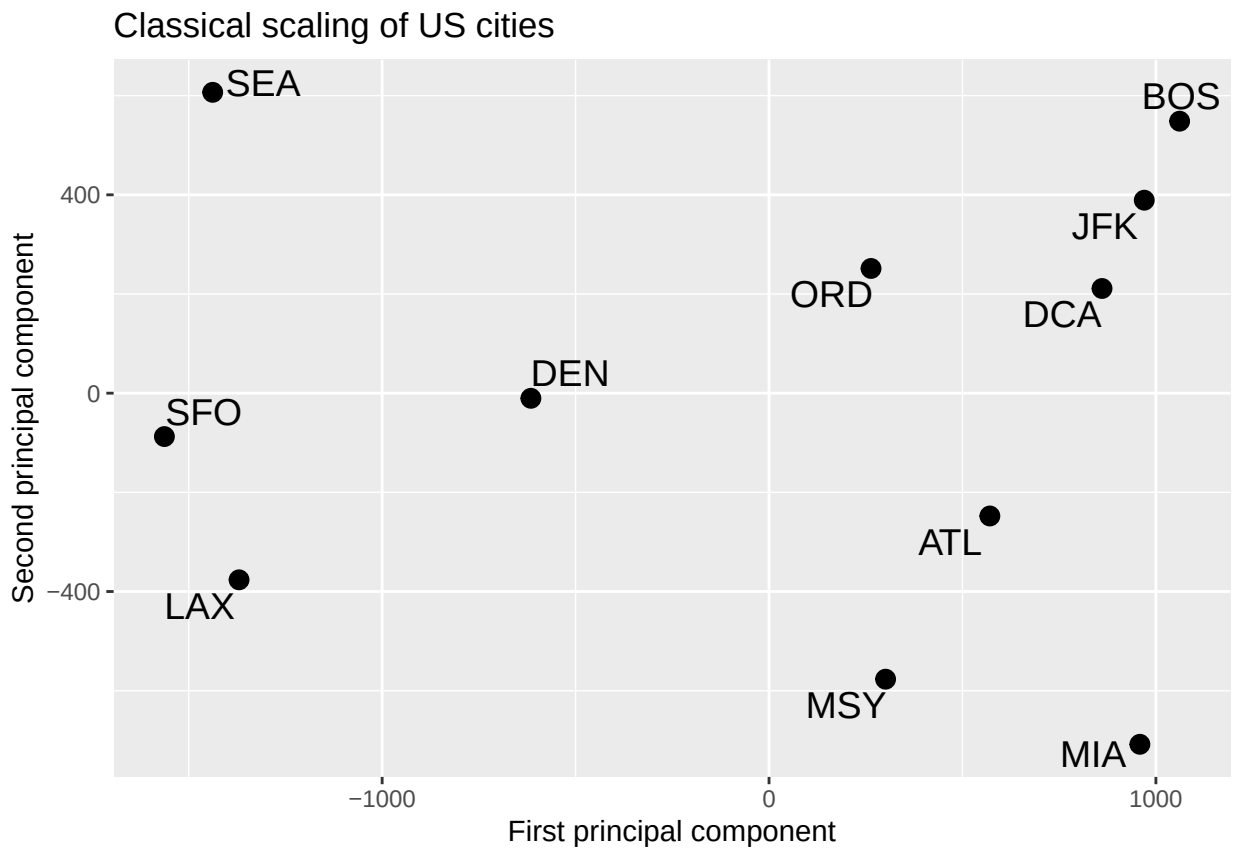
```
proximity_matrix = as.matrix(cities[1:4, 1:4])
evd = cmdscale(proximity_matrix, eig = TRUE, list. = TRUE)
Y = evd$points
principal_comp = Y %>%
    as.tibble() %>%
    mutate(city = colnames(proximity_matrix))
```

```
ggplot(principal_comp, aes(V1, V2)) + geom_point(size = 3) +
    geom_text_repel(aes(label = city), size = 5) + labs(x = "First principal component",
    y = "Second principal component", title = "Classical scaling of US cities") +
    coord_equal()
```



c) Perform the classical scaling algorithm to the entire dissimilarity matrix and obtain the principal coordinates. Plot the resulting coordinates in 2 dimensions.

```
classical_scaling_first_principals(noOfCities = 11, rotateMap = TRUE)
```



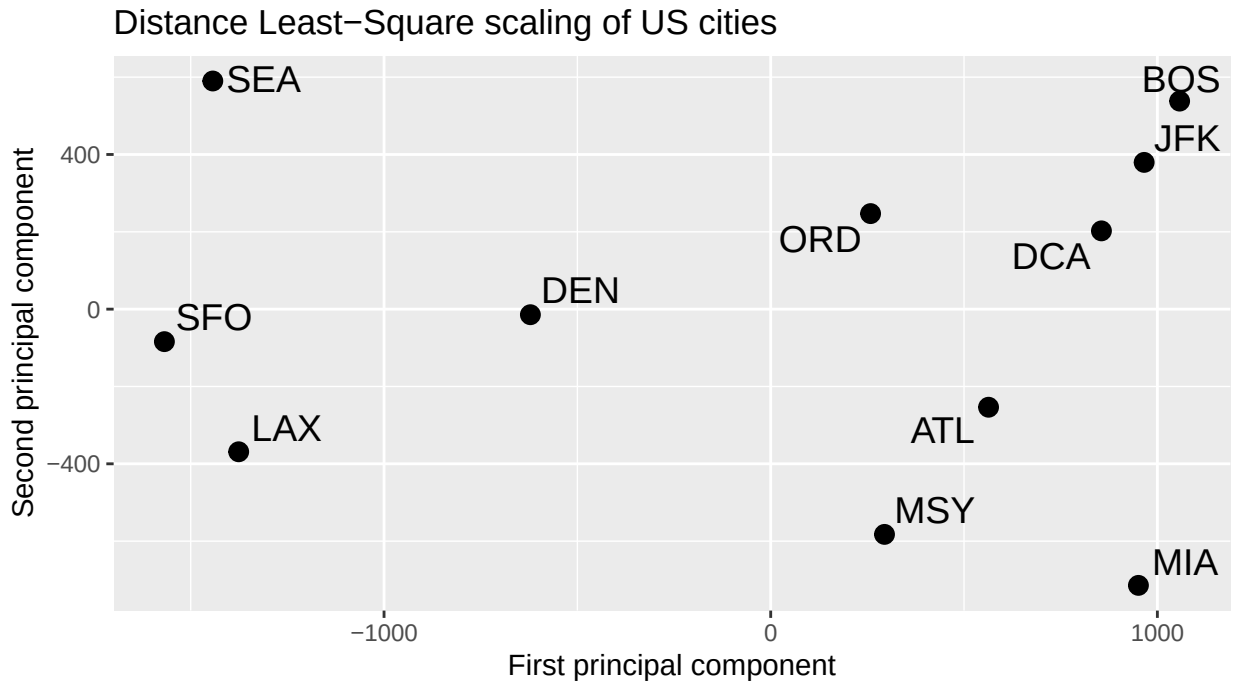
Question 2

a) (i) Use the mds() function – in the smacof package – to perform metric multidimensional scaling. Locate the cities in t=2 dimensions, starting with a classical scaling solution as the initial configuration.

From from first principles

```
# Step 1: Dissimilarity matrix
proximity_matrix = as.matrix(cities)
# Step 2: Initial config points y1....yn using classical
# scaling
low_dim_points = cmdscale(proximity_matrix)


stress_fun = function(y, dissimilarity_matrix, principalComp = 2) {
    y_mat = matrix(y, ncol = principalComp)  # points in lower-dim
    d_mat = as.matrix(dist(y_mat))  # distance of points in lower-dim
    # high dim transform fun = identity function i.e
    # maintains the dissimilarities from higher-dim as
    # distances
    weight_mat = 1/sum(dissimilarity_matrix[lower.tri(dissimilarity_matrix)]^2)
    error_mat = ((d_mat - dissimilarity_matrix)^2)[lower.tri(dissimilarity_matrix)]
```

```
    weight_mat * sum(error_mat)
}

min_stress_fun = optim(low_dim_points, stress_fun, dissimilarity_matrix = proximity_matrix)

optimal_distance_scaling_points = matrix(min_stress_fun$par,
    ncol = 2)

fitted_configs = optimal_distance_scaling_points %>%
    as.tibble() %>%
    mutate(city = colnames(proximity_matrix))


ggplot(fitted_configs, aes(-V1, -V2), alpha = 0.5, color = "blue",
    size = 10) + geom_point(size = 3) + geom_text_repel(aes(label = city),
    size = 5) + labs(x = "First principal component", y = "Second principal component",
    title = "Distance Least-Square scaling of US cities") + coord_equal()
```

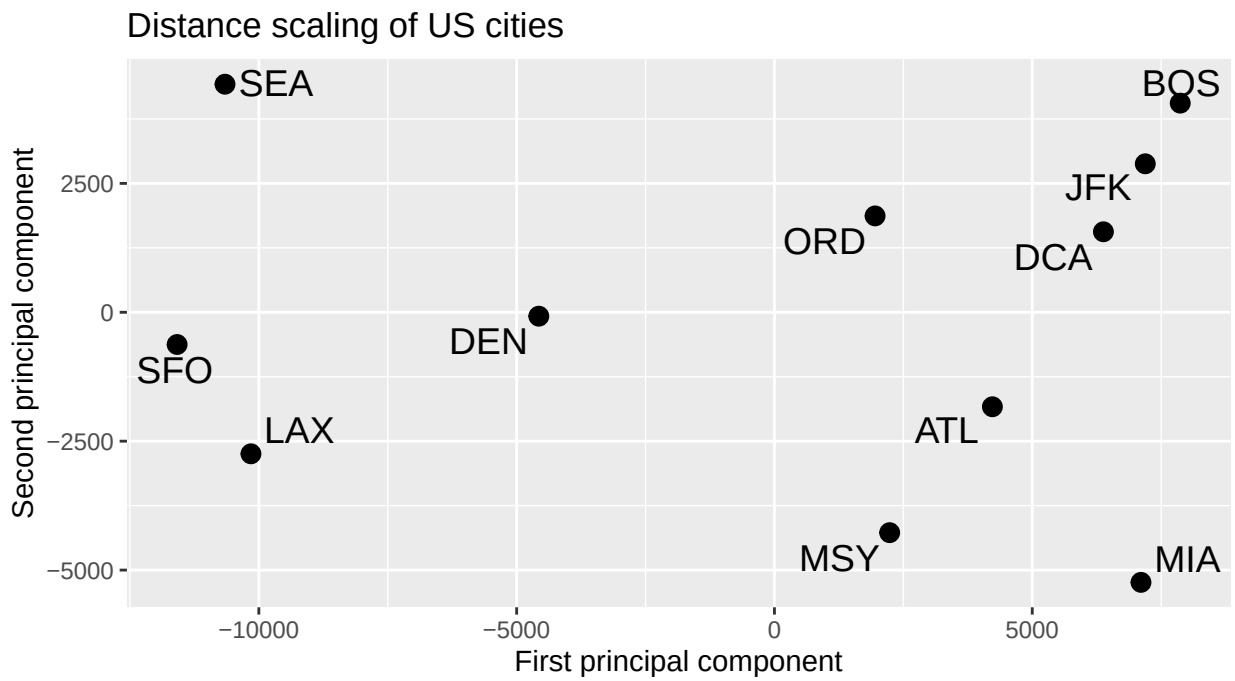



```
min_stress_fun$value
```

```
## [1] 3.331485e-06
```

Using mds function

```
library(smacof)

weight_mat = matrix(0, ncol = 11, nrow = 11)
constant_weight = 1/sum(proximity_matrix[lower.tri(proximity_matrix)]^2)
weight_mat[lower.tri(weight_mat)] = constant_weight


metric_mds = mds(delta = proximity_matrix, ndim = 2, init = "torgerson",
    type = "ratio", weightmat = weight_mat)

fitted_configs_2 = metric_mds$conf %>%
    as.tibble() %>%
    mutate(city = colnames(proximity_matrix))

ggplot(fitted_configs_2, aes(-D1, -D2), alpha = 0.5, color = "blue",
    size = 10) + geom_point(size = 3) + geom_text_repel(aes(label = city),
    size = 5) + labs(x = "First principal component", y = "Second principal component",
    title = "Distance scaling of US cities") + coord_equal()
```



(ii) Which of the 11 cities has the poorest fit? Explain your answer.

Per Point Stress Points

```
# Overall Stress and Stress Per Point
metric_mds$stress
```

```
## [1] 0.00206447
```

```
metric_mds$spp
```

```
##        ATL        BOS        ORD        DCA        DEN        LAX        MIA
##  2.6440777  7.9040554  1.7018267  0.7010979  1.5993119 20.1502872 12.2196378
##        JFK        SEA        SFO        MSY
##  2.3776837 33.4778124 13.6394028  3.5848066
```

Observing the per point stress points we note that SEA is the city with the poorest fit as it has the highest stress point of 33.48%, this means that it has been matched the poorest relative to it's original dissimilarities as it has the highest contribution to the stress value of 0.00206447. Could indicate an outlier or it's dissimilarities with the other cities might be inconsistent with the rest of the data

(iii) Plot the 2D multidimensional scaling configuration and compare this the locations of the cities on a map from the atlas.

From the map below we observe that the the low-dimensional map is a good representation of the Atlas mapping indicating the actual positions of the cities. We further support this with the relavtively low stress value of 0.00206447, indicating that the points in the low-dim have been matched relatively well with their original dissimilarities.

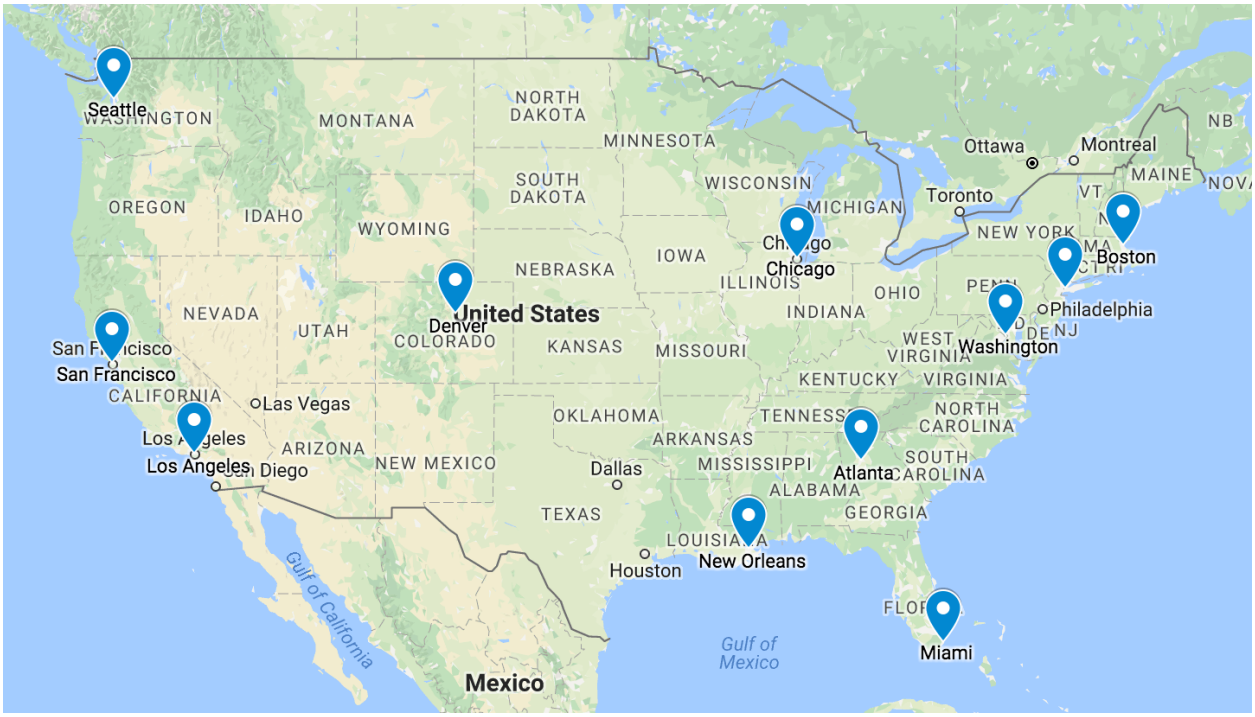
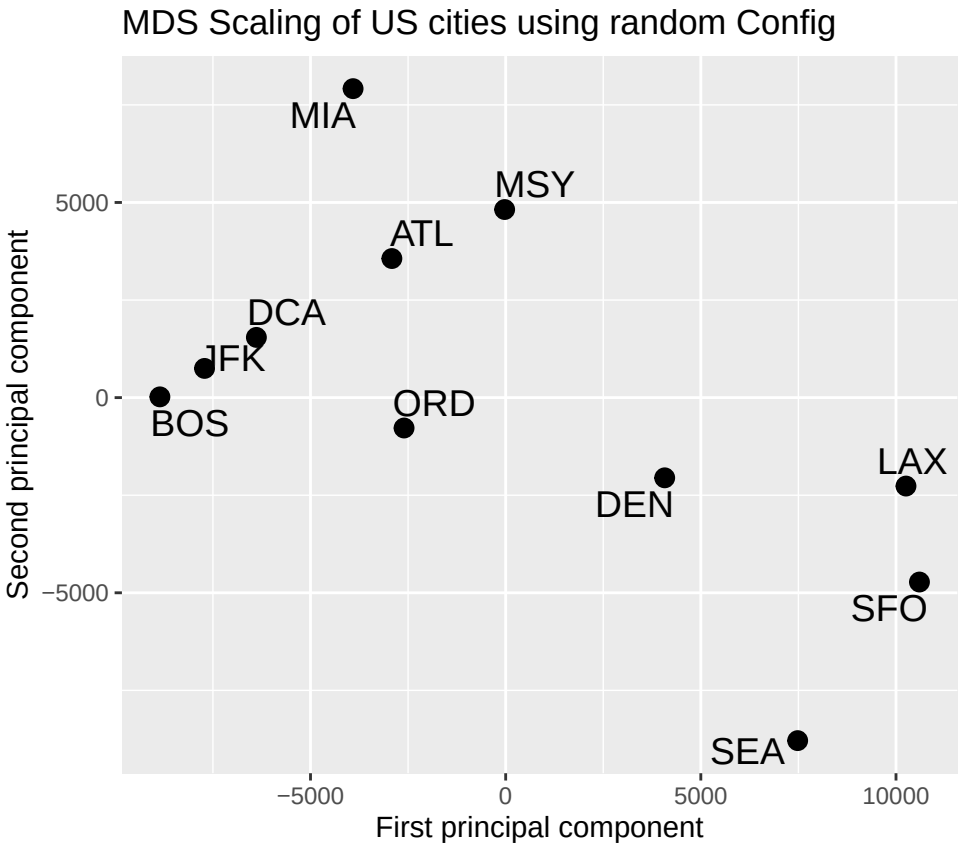


Figure 1: US Cities Map

b)Repeat the analysis in part (a) using a random starting solution. How does the different initial configuration change the location of the cities in the 2D configuration?

```
metric_mds_random = mds(delta = proximity_matrix, ndim = 2, init = "random",
    type = "ratio", weightmat = weight_mat)


fitted_configs_random = metric_mds_random$conf %>%
    as.tibble() %>%
    mutate(city = colnames(proximity_matrix))
ggplot(fitted_configs_random, aes(-D1, -D2), alpha = 0.5, color = "blue",
    size = 10) + geom_point(size = 3) + geom_text_repel(aes(label = city),
    size = 5) + labs(x = "First principal component", y = "Second principal component",
    title = "MDS Scaling of US cities using random Config") +
    coord_equal()
```




```
# Overal Stress and Stress Per Point
metric_mds_random$stress
```

```
## [1] 0.002150198
```

```
metric_mds_random$spp
```

```
##       ATL       BOS       ORD       DCA       DEN       LAX       MIA       JFK
```

```
##  2.564000 13.999352  1.868120  1.925221  3.829775 13.464993 10.624828  5.736687
##       SEA       SFO       MSY
## 30.948804 10.666566  4.371654
```

Using a random starting point we note that the resulting fit is quite poor. SEA has the poorest fit with a as it has the highest stress per point contribution to the stress. At times it's in the wrong direction and closer to LAX and SFO then it should be, however varies due to the randomness of the configuration. The cities on the east are all poorly represented as well. This shows that a random starting point is not effective.