Table of Contents

Unconstrained ordination 1
Principal Coordinate Analysis (PCoA) 1
Example 1 - PCoA on the matrix of distances between European cities 1

Unconstrained ordination

Principal Coordinate Analysis (PCoA)

Theory R functions **Examples** Exercise 🔒

Example 1 - PCoA on the matrix of distances between European cities

We use data from the variable `eurodist`, which is available in R (you don't need to install any library, just type `euro disc`). This distance matrix contains real geographical distances among big European cities (driving distance, in km). We will use this matrix to calculate PCoA and draw the PCoA ordination diagram, and also a screeplot of eigenvalues for individual PCoA axes.

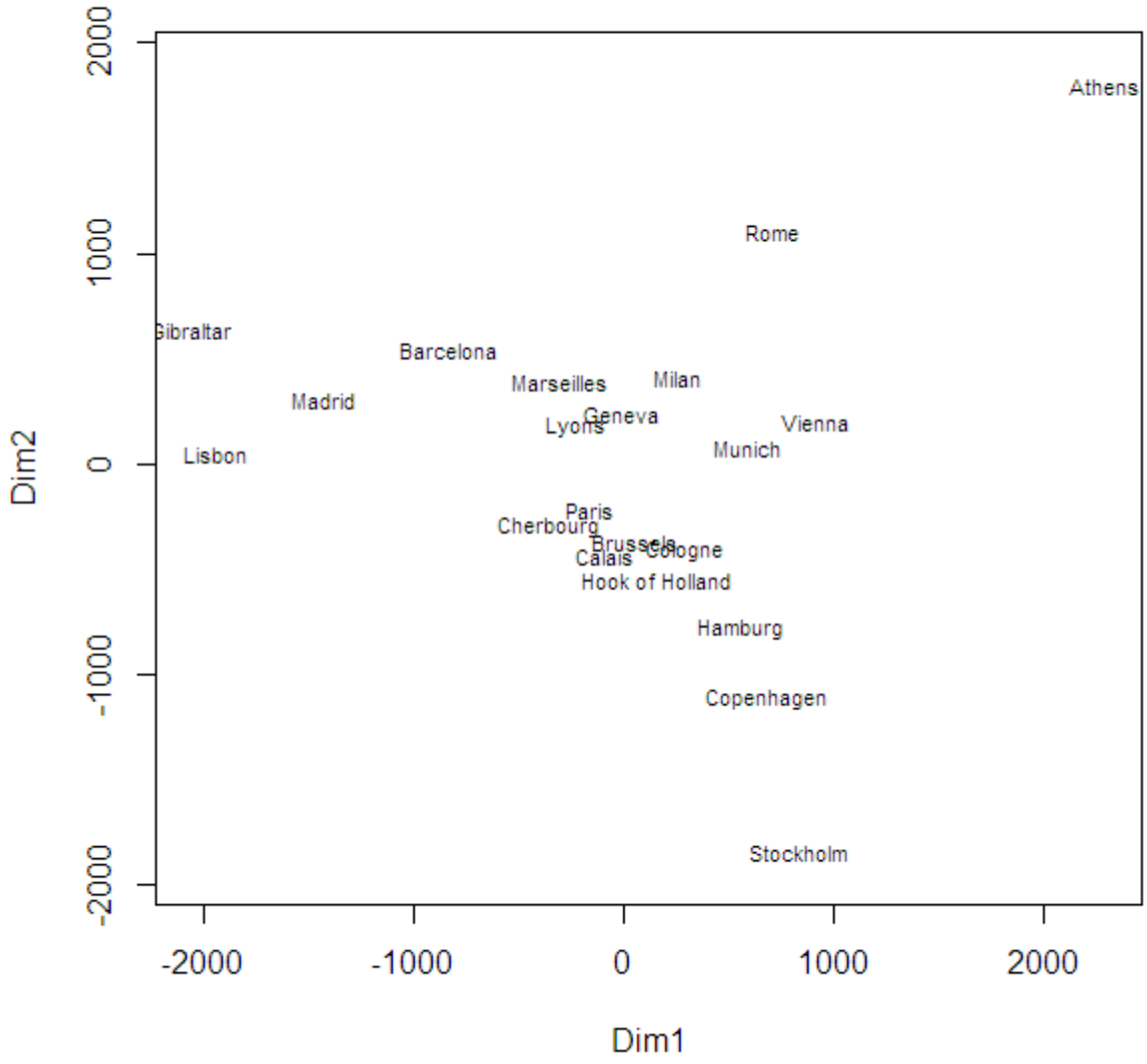
To calculate PCoA, use the base R function `cmdscale` (note that `vegan` contains the function `wcmdscale`, which in default setting is doing the same):

```
pcoa <- cmdscale (eurodist, eig = TRUE)
```

Note that I set up the argument `eig = TRUE` - in this way, the `cmdscale` function returns also the eigenvalues for individual axes (in the default setting this argument is set to FALSE and the function returns only the data frame of sample scores on individual PCoA axes).

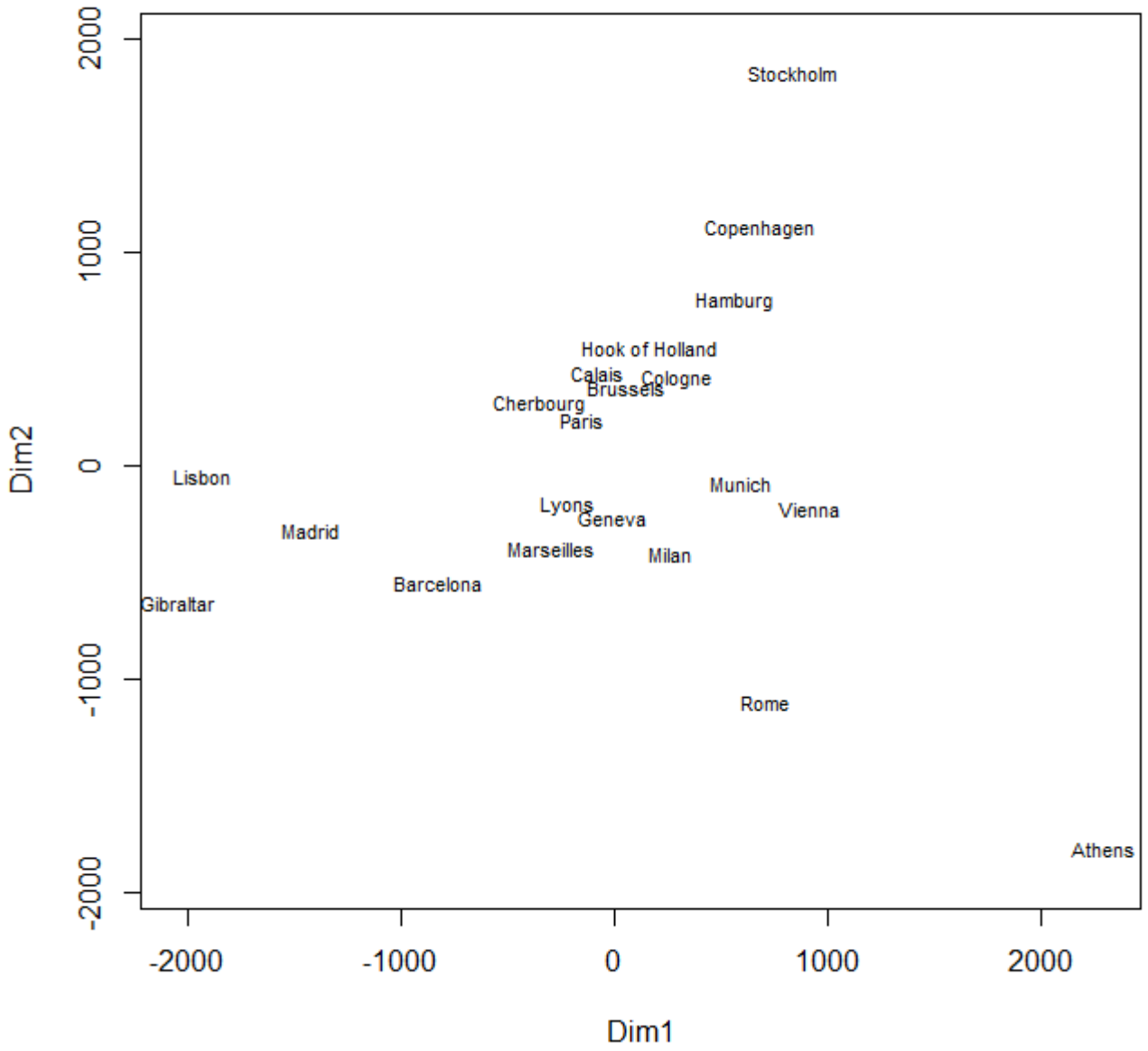
Now we can draw the ordination diagram of the cities:

```
library (vegan)
ordiplot (pcoa, display = 'sites', type = 'text')
```



You can see that the distances between cities make intuitive sense (Athens are far from Stockholm, for example), and it almost looks like the map of Europe, except that Athens are at north and Stockholm at south. Let's flip the y-axis (the second axis of PCoA) and draw the ordination diagram again. The sample scores in PCoA ordination are in the object `pcoa`, in the component `points`[1] (use `str (pcoa)` if you wish to see the structure of `pcoa` object).

```
pcoa[,2] <- -pcoa[,2]
ordiplot (pcoa, display = 'sites', type = 't')
```

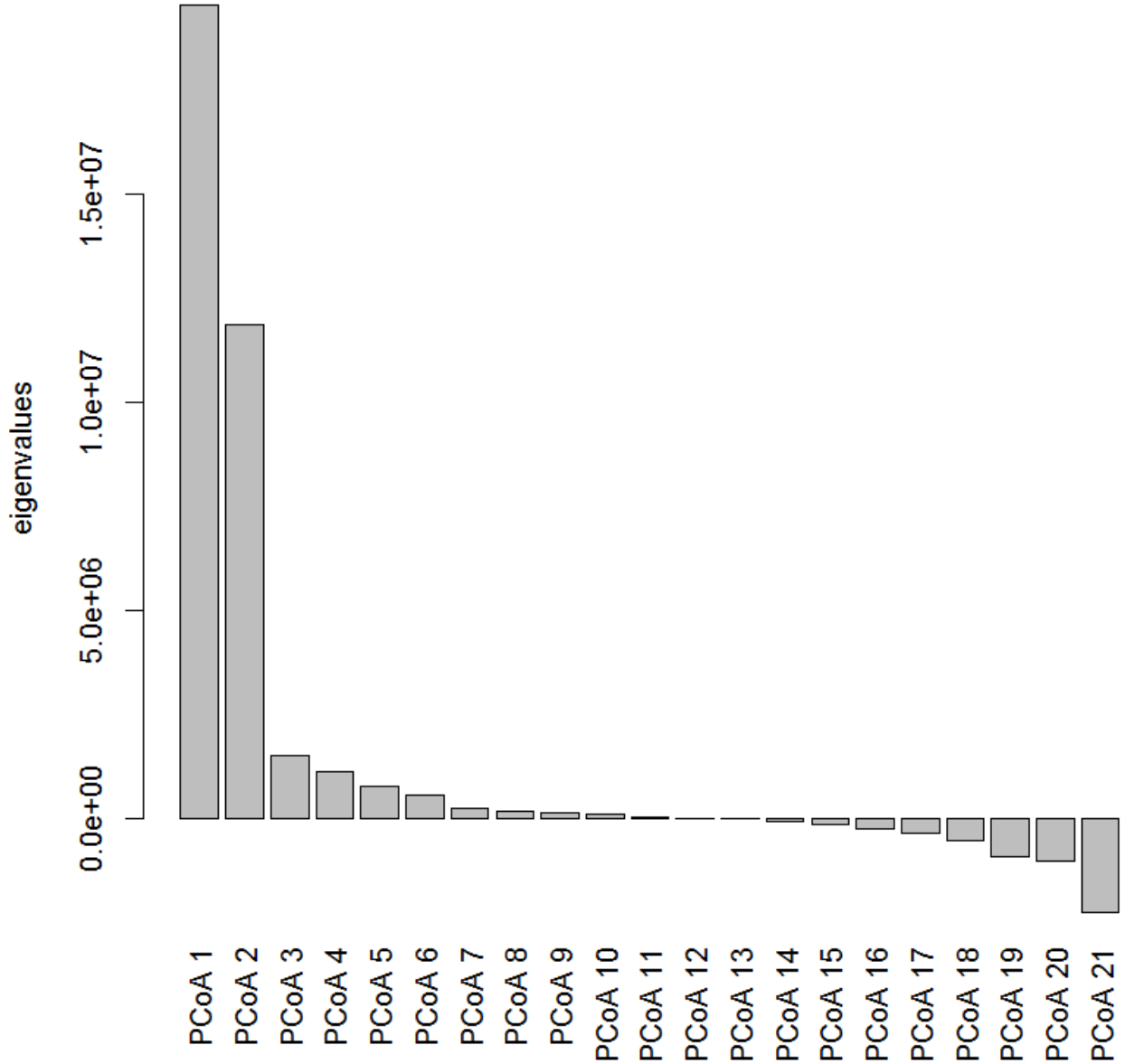


Now the distribution of cities make better geographical sense!

Finally, let's draw the screeplot with eigenvalues for individual axes; these eigenvalues are stored in the component `$eig` of the object `pcoa`:

```
barplot (pcoa$eig, names = paste ('PCoA', 1:21), las = 3, ylab =
'eigenvalues')
```

Note that `names` argument adds the names to tickmarks on horizontal axis, `las` argument influences rotation of labels on both x and y axis (see `?par` for explanation) and `ylab` adds the label to the



[1)]

If we have calculated `cmdscale` with `eig = FALSE`, the structure of `pcoa` object would be simpler, it would be just a data frame with sample scores; with `eig = TRUE` the object became a list with the score data frame nested inside within the component `$points`.