Genes and Geography in Europe

In this exercise, the aim is to reproduce parts of the analysis of one of the seminal papers in human population genetics, "Genes Mirror Geography within Europe", Novembre et al 2018 Nature. We need to load two R packages for this:

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.0     v dplyr   1.0.4
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(vegan)
```

```
## Loading required package: permute
```

```
## Loading required package: lattice
```

```
## This is vegan 2.5-7
```

```
library(maps)
```

```
##
## Attaching package: 'maps'
```

```
## The following object is masked from 'package:purrr':
##
##     map
```

The dataset for reproducing the main figures is available from the Novembre lab Github repository. Here, we will use a minimally reformatted versions of the main PCA results table. There are two expected outcomes of this exercise:

Create a replication of the PCA plot in Figure 1, starting from the PCA results table

To achieve this, you should carry out the following steps:

- Read the dataset containing PCA coordinate and geographic locations, as well as the color palette table into R
- Rotate the original PCA coordinates to maximize the correlation between PCs and geographic locations. This can be achieved using the `procrustes()` function of the `vegan` R package. The function takes two matrix objects as input, the first one the target matrix (i.e. the geographic coordinates), and the second one the matrix to be rotated (i.e. the PCA result). More details on how to use the function can be found in the help page by typing `?procrustes`
- Visualize the resulting rotated PCA using `ggplot2`. Start by trying to replicate the overall PCA plot of samples (using `geom_text()` with the given population labels), with the color scale in the given

palette table (using `scale_colour_manual()`). Add population median positions using summary tables produced by `group_by()` and `summarise()`.

- Adding rotated axes and their labels requires a bit more processing, with one possible solution being rotating points at the ends of the original axes into the new coordinates by using the `predict()` function on the procrustes result object.

Predict the geographic location of samples from a chosee country using their PCA position

To achieve this, you can follow the linear regression approach from Novembre et al:

- Split your data into a test set containing only samples from the country of interest, and a training set of the remaining individuals.
- Create two linear regression models, one for longitude and one for latitude as outcomes. For each model, use both rotated PC1 and PC2, as well as their interaction as predictors. The syntax for such a model in R is `lm(response ~ predictor1 * predictor2, data = data)`.
- Obtain the predicted longitude and latitude for the test samples using the `predict()` function on both models.
- Plot the actual and predicted locations of the test samples on a map. You can find a simple example of how to plot spatial data in `ggplot2` towards the end of the chapter "Data visualization" of the "R for data science" book.

The following sections of this document show an example solution for these tasks

Preparing the dataset

First we'll set up the environment. We need the `tidyverse` packages, as well as the `vegan` package to perfrorm the procrustes analysis

```r
pca <- read_tsv("datasets/novembre_2008_pca.tsv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   sampleId = col_character(),
##   country = col_character(),
##   alabels = col_character(),
##   longitude = col_double(),
##   latitude = col_double(),
##   PC1 = col_double(),
##   PC2 = col_double()
## )
```

```r
colors <- read_tsv("datasets/novembre_2008_colours.tsv", col_names = c("country", "color"))
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   country = col_character(),
##   color = col_character()
## )
```

The variable `pca` contains the results of the principal component analysis of 1,387 European individuals from Novembre et al 2008. Let's have a look at the first few lines of the dataset:

sampleId	country	alabels	longitude	latitude	PC1	PC2
ind_474	Croatia	HR	16.10000	45.32000	0.0111	-0.0460
ind_660	Serbia and Montenegro	YG	20.61572	43.94949	-0.0300	-0.0443
ind_1890	Czech Republic	CZ	15.37698	49.73661	0.0137	-0.0444
ind_1923	Bosnia and Herzegovina	BA	17.86202	44.17588	0.0028	-0.0493
ind_2083	Serbia and Montenegro	YG	20.61572	43.94949	-0.0031	-0.0416
ind_2733	Serbia and Montenegro	YG	20.61572	43.94949	-0.0026	-0.0424

The color palette used for the countries is stored as a table in the variable `colors`:

country	color
Germany	gold1
Netherlands	orange
Austria	tan3
Luxembourg	yellow
Czech Republic	yellowgreen
Hungary	gold2

Rotating the PCA

In order to find the rotation for the PCA that aligns best with the geographic location of the samples, we will use the `procrustes()` function from the package `vegan`. We first need to convert both the principal components and geographic locations of the samples into matrix format.

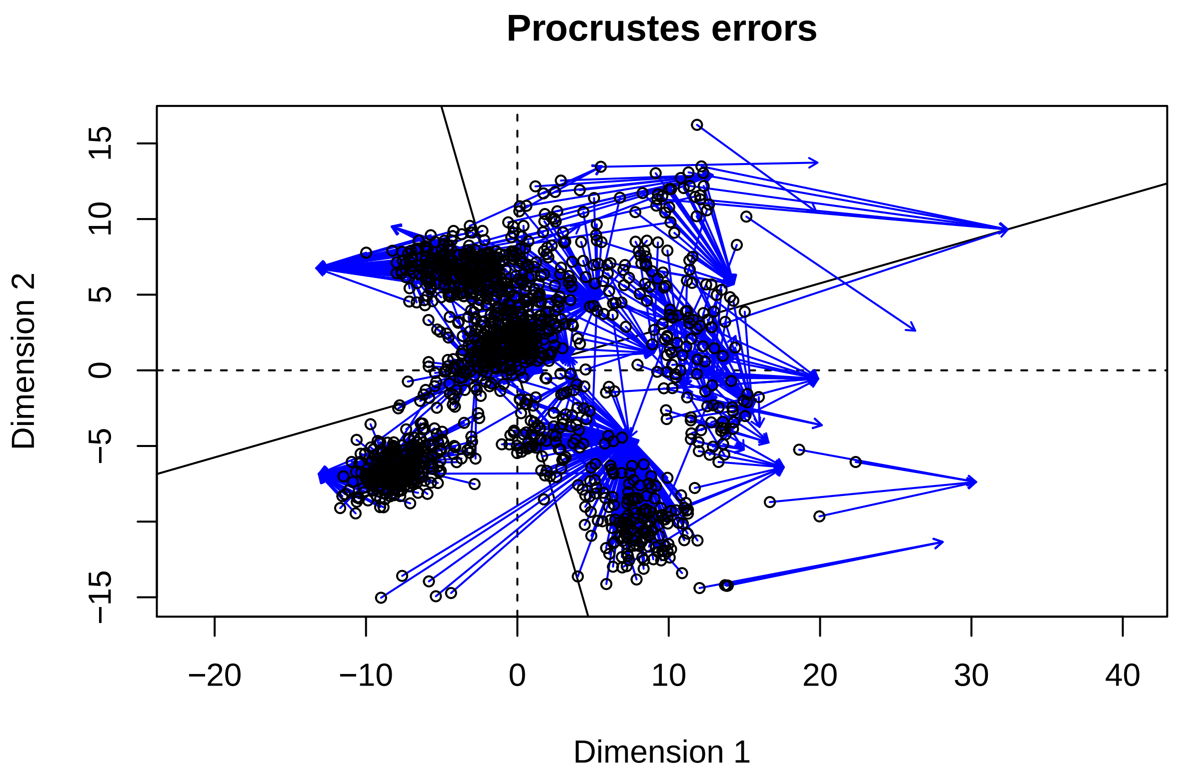
```
loc <- pca %>%
    select(longitude, latitude) %>%
    as.matrix()

pca1 <- pca %>%
    select(PC1, PC2) %>%
    as.matrix()
```

Now we carry out the procrustes analysis to find the optimal rotation

```
rot <- procrustes(loc, pca1)
```

We can examine the result by using base R `plot` function on the resulting object

```
plot(rot)
```



The rotated PC coordinates can be extracted from the `Yrot` element of the result object. Here we add two new columns for the rotated PCs to our `pca` object

```
pca <- pca %>%
    mutate(PC1_rot = rot$Yrot[,1], PC2_rot = rot$Yrot[,2])
```

Recreating Figure 1 of Novembre et al 2008

With the rotated PCA results calculated, everything is in place for creating the figure. First we set up the color scale for the countries

```
colorscale <- colors$color
names(colorscale) <- colors$country
```

Next, we calculate the median PC values for each country

```
pca_summary <- pca %>%
    group_by(country, alabels) %>%
    summarise_at(c("PC1_rot", "PC2_rot"), median)
```

To add rotated axes lines, we take the coordinates of the four points at the end of the axis ranges, and rotate them into the new coordinate space using the `predict` function on the procrustes output object

```
lim_pc1 <- extendrange(pca$PC1)
lim_pc2 <- extendrange(pca$PC2)

axis <- rbind(c(lim_pc1[1], 0), c(lim_pc1[2], 0), c(0, lim_pc2[1]), c(0, lim_pc2[2]))
axis_rot <- predict(rot, axis, truemean = FALSE)
axis_rot_df  <- tibble(x = axis_rot[c(1, 3),1], xend = axis_rot[c(2, 4),1], y = axis_rot[c(1, 3),2], ye
```

Similarly, we calculate rotated coordinates for two axis labels, in a position shifted towards the origin from the axis end point

```
labels <- rbind(c(lim_pc1[2] * 0.85, 0), c(0, lim_pc2[2] * 0.85))
labels_rot <- predict(rot, labels, truemean = FALSE) %>%
    as_tibble() %>%
    mutate(labels = c("PC1", "PC2"))
```

```
## Warning: The `x` argument of `as_tibble.matrix()` must have unique column names if `.name_repair` is
## Using compatibility `.name_repair`.
```

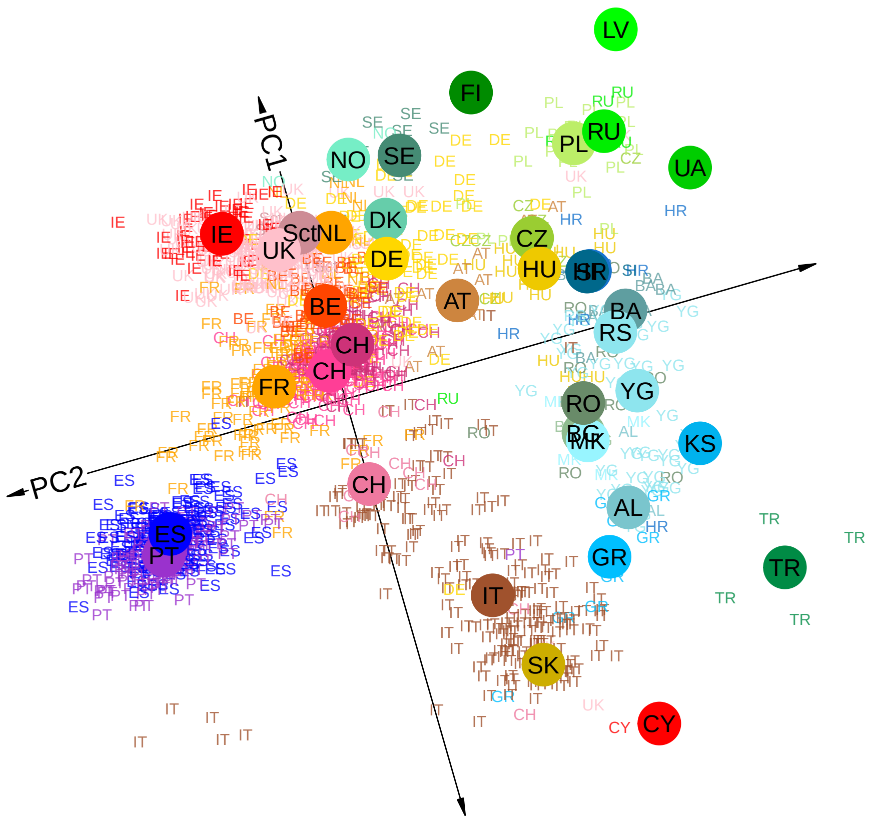
```
colnames(labels_rot)[1:2] <- c("PC1_rot", "PC2_rot")
```

Finally, we need to calculate the rotation angle from the rotation matrix for the axis label text

```
theta1 <- acos(rot$rotation[1,1]) * 180 / pi
```

With all objects in place we can now build the final plot

```
p <- ggplot(pca, aes(x = PC1_rot, y = PC2_rot))
p +
    geom_segment(aes(x = x, xend = xend, y = y, yend = yend), size = 0.25, arrow = arrow(ends = "both",
    geom_point(size = 10, colour = "white", data = labels_rot) +
    geom_text(aes(label = labels), angle = c(theta1 + 180, theta1 + 270), data = labels_rot) +
    geom_text(aes(label = alabels, colour = country), alpha = 0.8, size = 2) +
    geom_point(aes(colour = country), size = 7, data = pca_summary) +
    geom_text(aes(label = alabels), size = 3, data = pca_summary) +
    scale_color_manual(name = "Country", values = colorscale) +
    coord_equal() +
    theme_void() +
    theme(legend.position = "none")
```



Predicting location of origin from the PCA position

In this last section, we will follow the paper and build a predictor for an individual's geographic location from its PCA position, using linear regression. As test case, we take all individuals of a country of choice, remove them from the dataset and use the remaining individuals to build the model. Then, we use the built model to predict the location of the test individuals.

First we split our dataset into training and test data

```
test_country <- "Germany"
d_train <- filter(pca, country != test_country)
d_test <- filter(pca, country == test_country)
```

Next, we carry out two linear regressions, modelling both longitude and latitude as functions of the rotated PCs and including an interaction term

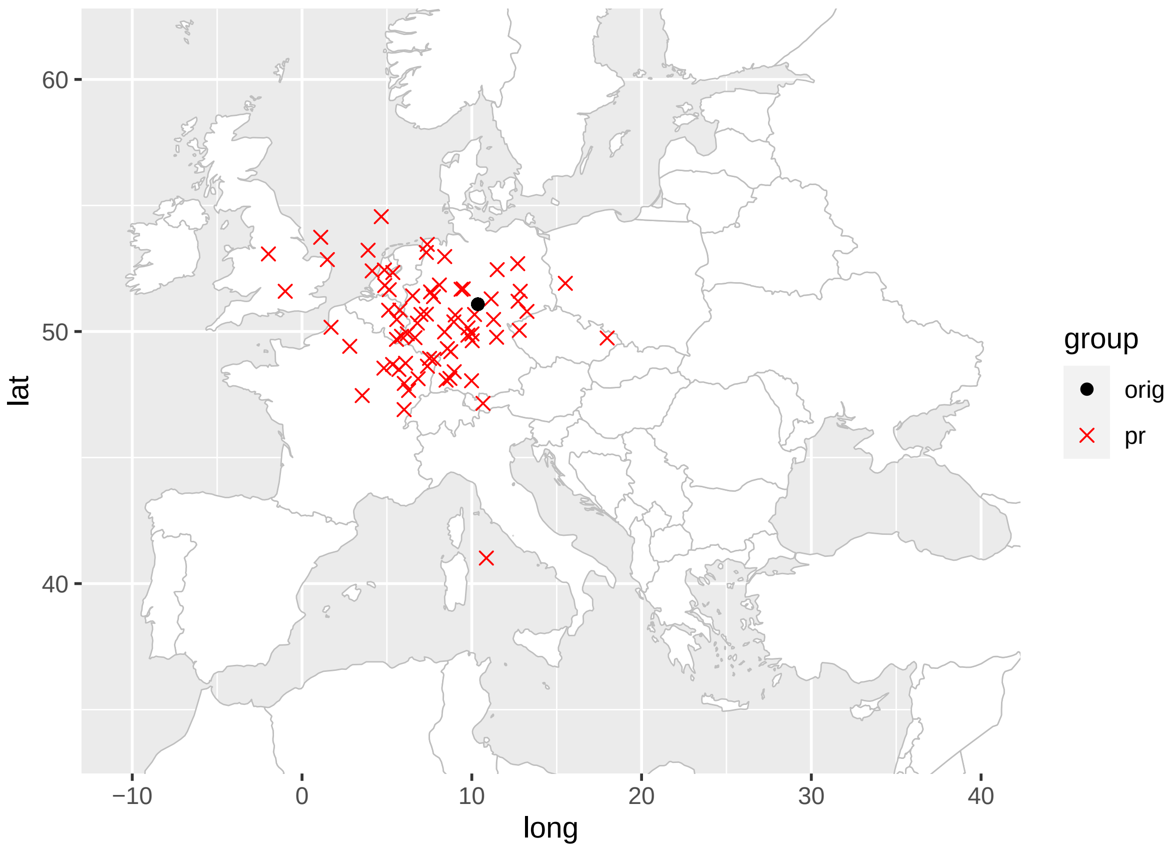
```
lm_long <- lm(longitude ~ PC1_rot * PC2_rot, data = d_train)
lm_lat <- lm(latitude ~ PC1_rot * PC2_rot, data = d_train)
```

Now we use the `predict()` function to infer the predicted geographic location for the test samples, and reshape the results into a format useful for plotting

```
d1 <- select(d_test, sampleId, longitude, latitude) %>%
    mutate(longitude_pr = predict(lm_long, d_test), latitude_pr = predict(lm_lat, d_test), longitude_or
    select(-longitude:-latitude) %>%
    pivot_longer(-sampleId) %>%
    separate(name, into = c("variable", "group")) %>%
    pivot_wider(names_from = variable, values_from = value)
```

Finally, we can plot the inferred versus actual sample locations using the basic map plotting functionality in `ggplot2`

```
w <- map_data("world")

lim_long <- extendrange(pca$longitude)
lim_lat <- extendrange(pca$latitude)

pal <- c("black", "red")
names(pal) <- c("orig", "pr")
sh <- c(16, 4)
names(sh) <- c("orig", "pr")

p <- ggplot(w)
p +
    geom_polygon(aes(long, lat, group = group), fill = "white", color = "grey", size = 0.25) +
    geom_point(aes(x = longitude, y = latitude, colour = group, shape = group), size = 2, data = d1) +
    scale_color_manual(values = pal) +
    scale_shape_manual(values = sh) +
    coord_quickmap(xlim = lim_long, ylim = lim_lat)
```



lat
60
50
40
long
−10
0
10
20
30
40
group
orig
pr