Exercise_2 solution

Tristan Koning, Isabelle Cretton, Ali Movasati

Sept. 30th, 2024

```
#install.packages("ggrepel")
#install.packages("ggplot2")
#install.packages("ggbiplot")
library(ggplot2)
library(ggrepel)
library(ggbiplot)
require(maps)
```

```
## Loading required package: maps
```

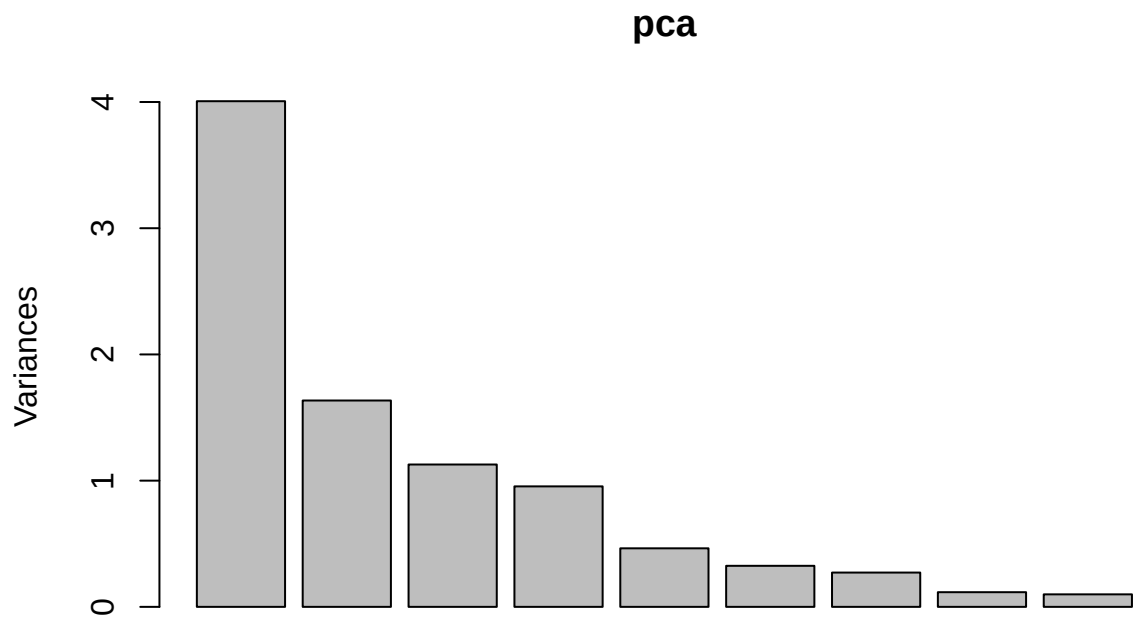
```
require(fields)
```

```
## Loading required package: fields

## Loading required package: spam

## Spam version 2.10-0 (2023-10-23) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.

##
## Attaching package: 'spam'

## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve

## Loading required package: viridisLite

##
## Try help(fields) to get started.
```

Problem 1

1.A)

```r
# Setup
data <-  read.csv("data/protein.txt", sep = "\t", header = TRUE)

# Perform PCA on data, except country names
pca <- prcomp(data[, -1], scale = TRUE)

# Variance explained by each PC
screeplot(pca)
```

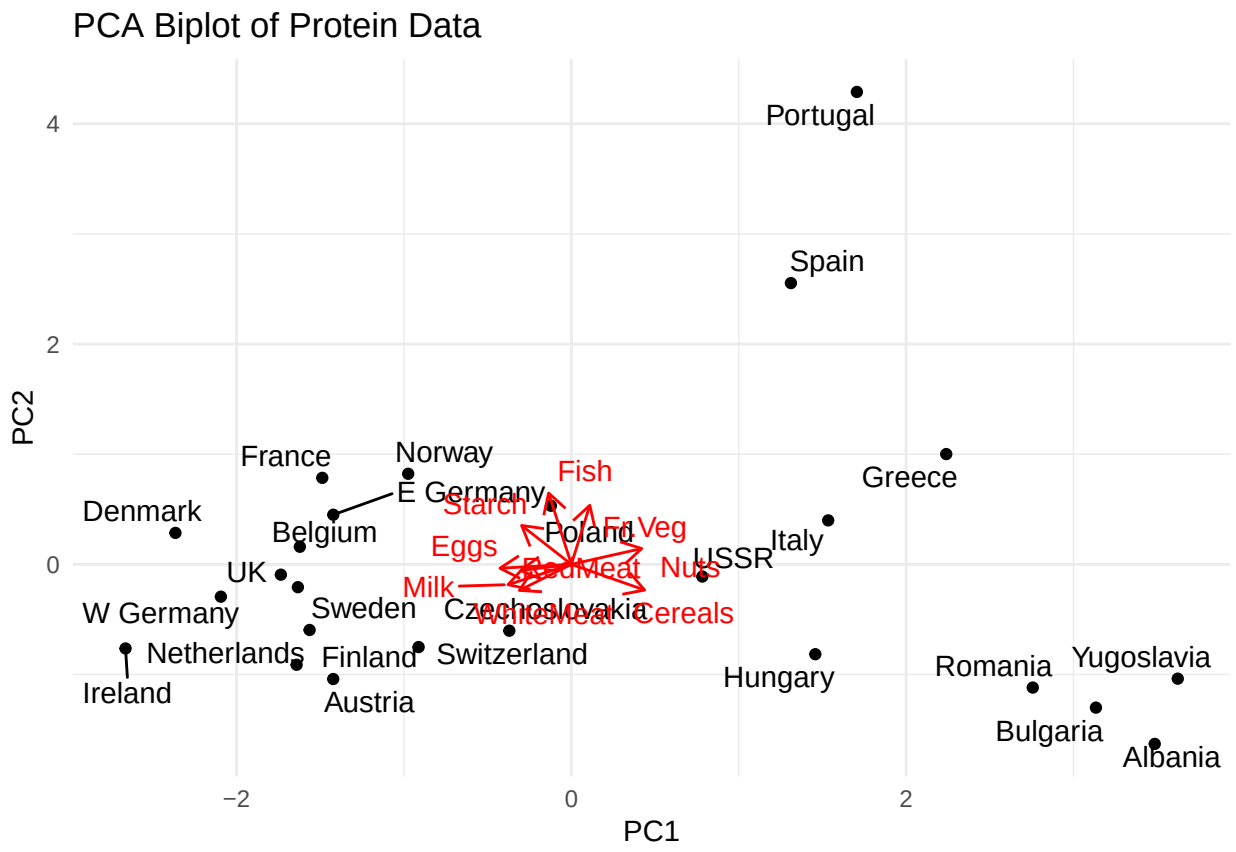



```r
round(cumsum(pca$sdev^2 / sum(pca$sdev^2)), 3)
```

```
## [1] 0.445 0.627 0.752 0.858 0.910 0.946 0.976 0.989 1.000
```

```r
# Get loadings of PC
loadings <- pca$rotation
loadings_data <- data.frame(Variable = rownames(loadings), PC1 = loadings[, 1], PC2 = loadings[, 2])

# Perform Dimensionality reduction
scores <- pca$x
pca_data <- data.frame(Country = data$Country, PC1 = scores[, 1], PC2 = scores[, 2])

ggplot() +
  geom_point(data = pca_data, aes(x = PC1, y = PC2)) +
  geom_text_repel(data = pca_data, aes(x = PC1, y = PC2, label = Country)) +
```

```
  geom_segment(data = loadings_data, aes(x = 0, y = 0, xend = PC1, yend = PC2),
               arrow = arrow(length = unit(0.3, "cm")), color = "red") +
  geom_text_repel(data = loadings_data, aes(x = PC1, y = PC2, label = Variable), color = "red") +
  labs(title = "PCA Biplot of Protein Data", x = "PC1", y = "PC2") +
  theme_minimal()
```



Looking at the Biplot of the PCA together with its Loadings, we can observe that Eastern European Countries consume their protein more along PC1, which primarily includes Nuts and Cereals. On the other Hand, Western European and Scandinavian Countries consume their protein via Meat, Eggs and Milk. Mediterranean Latin Countries such as Portugal, Spain and Greece consume theirs along both PC1 and 2, which includs Fruites, Vegetables and Fish additionally to what has already been mentioned along PC1.

1.B)

Given the high cumulative variance explained by the first 4 components, and diminishing returns from including more components, it would be reasonable to focus on the first 4 components for interpreting and understanding the data. Breakdown of what each component represents: - PC1 (44.52%): likely captures the general protein consumption pattern, with higher loadings on food groups such as Fish and Fr&Veg (Fruits & Vegetables). Countries like Portugal and Spain have high scores on PC1 due to higher fish consumption, while countries like Romania consume more cereals and have lower PC1 scores.

- PC2 (18.17%): may reflect meat consumption patterns, particularly distinguishing countries that consume more Red Meat (like the UK and France) from those with lower consumption of these food groups.

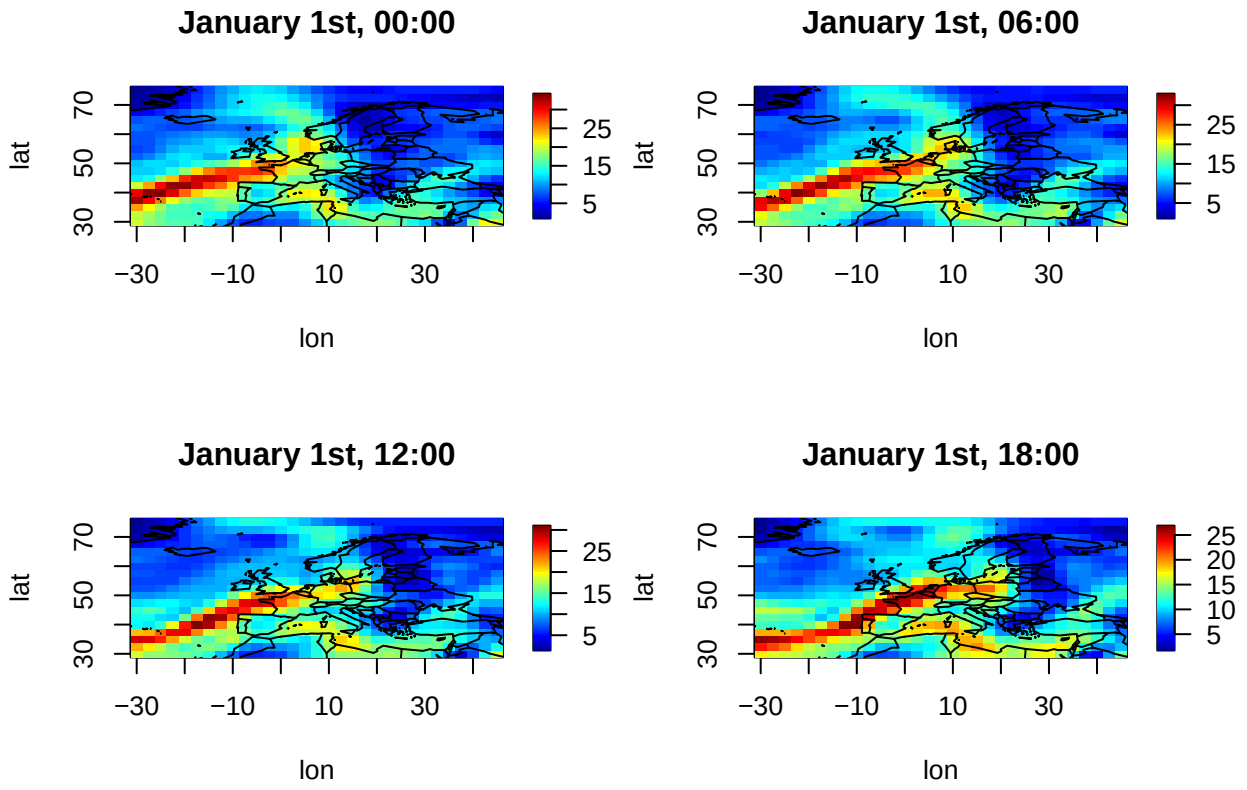
- PC3 (12.53%): could represent a trade-off between White Meat and Dairy (Milk) consumption, differentiating countries like Ireland and W Germany from those with lower dairy consumption.
- PC4 (10.61%): might capture more specific dietary differences related to nuts, eggs, and starch, distinguishing countries like Greece and Hungary from others.

Problem 2

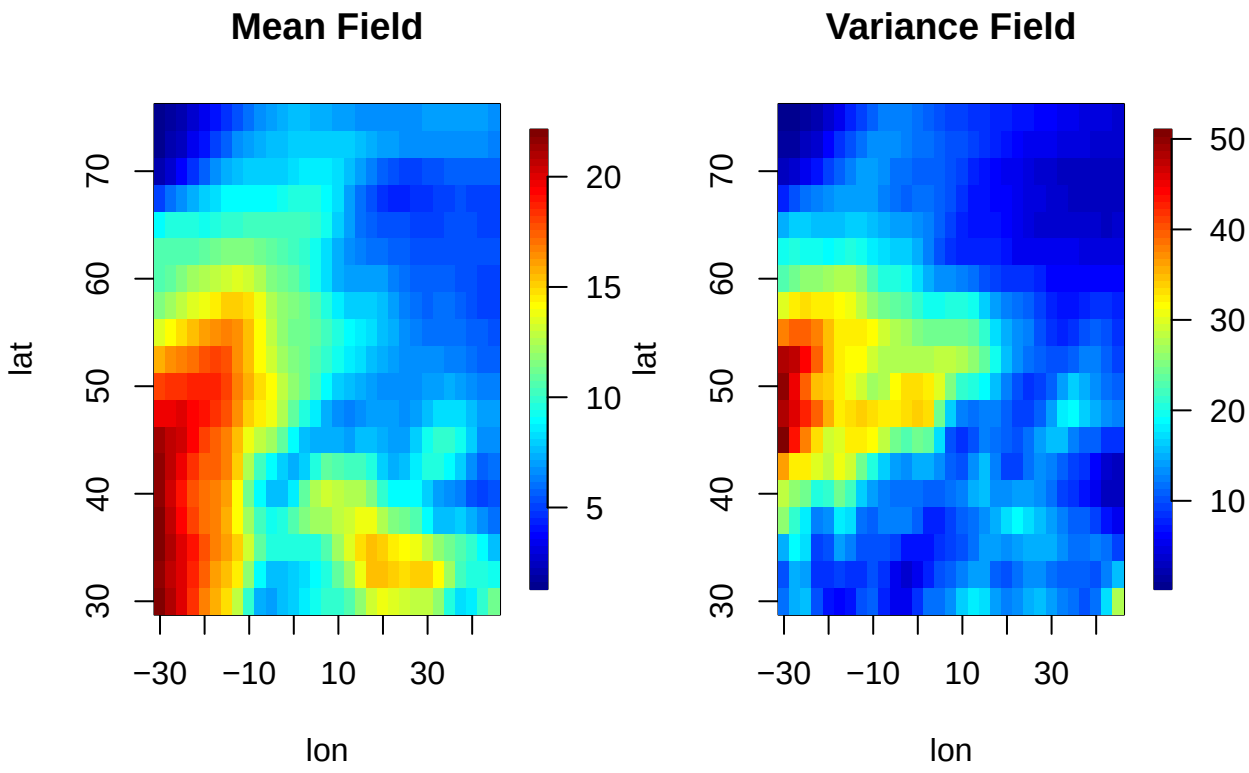
```
# load the data
loaded_obj <- load("data/prec_jan_feb.RData")
```

2.A)

```
par(mfcol=c(2,2))

times <- c("00:00", "06:00", "12:00", "18:00")

for (i in 1:4){
    time <- times[i]
    image.plot(lon, lat, pre[, , i], main = paste0("January 1st, ", time))
    map("world", add = T)
}
```

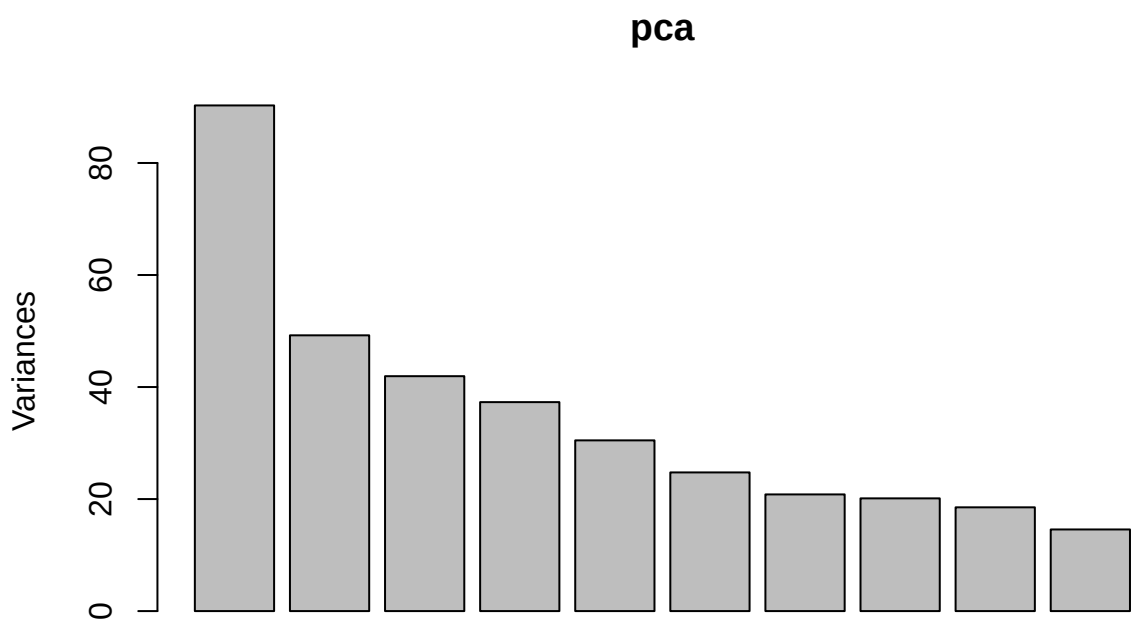


```r
par(mfcol=c(1,2))

mean_values <- apply(pre, c(1, 2), mean)

image.plot(lon, lat, mean_values, main = paste0("Mean Field"))


variance_values <- apply(pre, c(1, 2), var)

image.plot(lon, lat, variance_values, main = paste0("Variance Field"))
```



2.B)

```r
# Data matrix: 31 x 19 spatial locations, 240 observations
pre2 <- t(array(pre, c(dim(pre)[1] * dim(pre)[2], dim(pre)[3])))

# Perform PCA
pca <- prcomp(pre2, scale = TRUE)

# Variance explained by each PC
screeplot(pca)
```

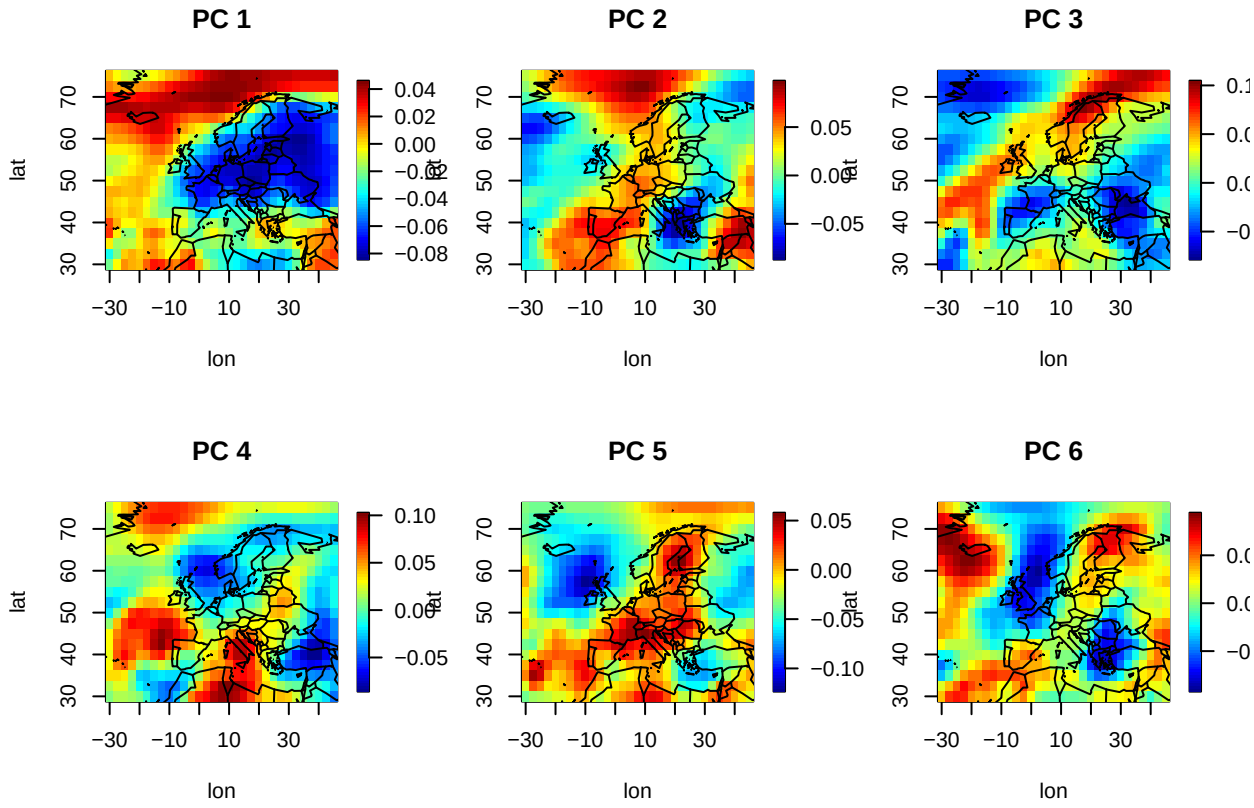



```
round(cumsum(pca$sdev^2 / sum(pca$sdev^2)), 3)
```

```
##   [1] 0.153 0.237 0.308 0.371 0.423 0.465 0.501 0.535 0.566 0.591 0.614 0.637
##  [13] 0.656 0.674 0.690 0.705 0.721 0.735 0.748 0.760 0.771 0.783 0.793 0.802
##  [25] 0.811 0.819 0.827 0.834 0.840 0.847 0.853 0.859 0.865 0.870 0.875 0.880
##  [37] 0.884 0.889 0.893 0.897 0.901 0.905 0.908 0.912 0.915 0.918 0.921 0.924
##  [49] 0.927 0.929 0.932 0.934 0.936 0.939 0.941 0.943 0.945 0.947 0.949 0.950
##  [61] 0.952 0.953 0.955 0.956 0.958 0.959 0.961 0.962 0.963 0.964 0.965 0.966
##  [73] 0.967 0.968 0.970 0.971 0.971 0.972 0.973 0.974 0.975 0.976 0.977 0.977
##  [85] 0.978 0.979 0.980 0.980 0.981 0.982 0.982 0.983 0.983 0.984 0.984 0.985
##  [97] 0.985 0.986 0.986 0.987 0.987 0.988 0.988 0.988 0.989 0.989 0.990 0.990
## [109] 0.990 0.991 0.991 0.991 0.992 0.992 0.992 0.992 0.993 0.993 0.993 0.993
## [121] 0.994 0.994 0.994 0.994 0.994 0.995 0.995 0.995 0.995 0.995 0.996 0.996
## [133] 0.996 0.996 0.996 0.996 0.996 0.997 0.997 0.997 0.997 0.997 0.997 0.997
## [145] 0.997 0.998 0.998 0.998 0.998 0.998 0.998 0.998 0.998 0.998 0.998 0.998
## [157] 0.998 0.998 0.999 0.999 0.999 0.999 0.999 0.999 0.999 0.999 0.999 0.999
## [169] 0.999 0.999 0.999 0.999 0.999 0.999 0.999 0.999 0.999 0.999 0.999 0.999
## [181] 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
## [193] 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
## [205] 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
## [217] 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
## [229] 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
```

```
# Get loadings of top 6 PCs that explain the most variance
loadings <- pca$rotation[, 1:6]
```

```r
# Plot for each loading
par(mfrow = c(2, 3))
for (i in 1:6) {
  image.plot(lon, lat, matrix(loadings[, i], nrow = 31), main = paste("PC", i))
  map("world", add = TRUE)
}
```



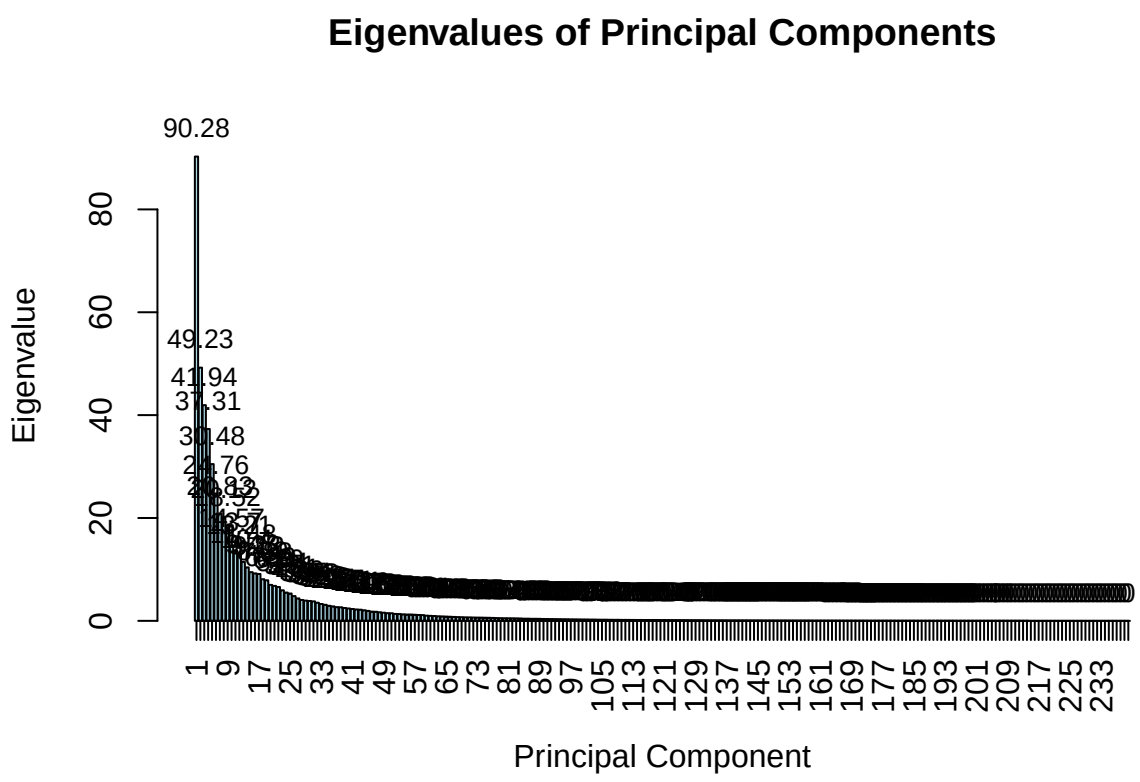
« Comments »

- First we needed to bring the data in the right format to perform PCA. We have 31*19=589 locations which are our variables. We have 240 time points that are our observations. Therefore, we rearrange the 3 dimensional matrix into a 2D matrix where the columns are the locations (variables or features) and the rows are time points (observations).
- We have spatial locations (variables)
- in this scenario we do not scale the data since they are all in the same unit and we are interested in the difference in variance that exist in the data.

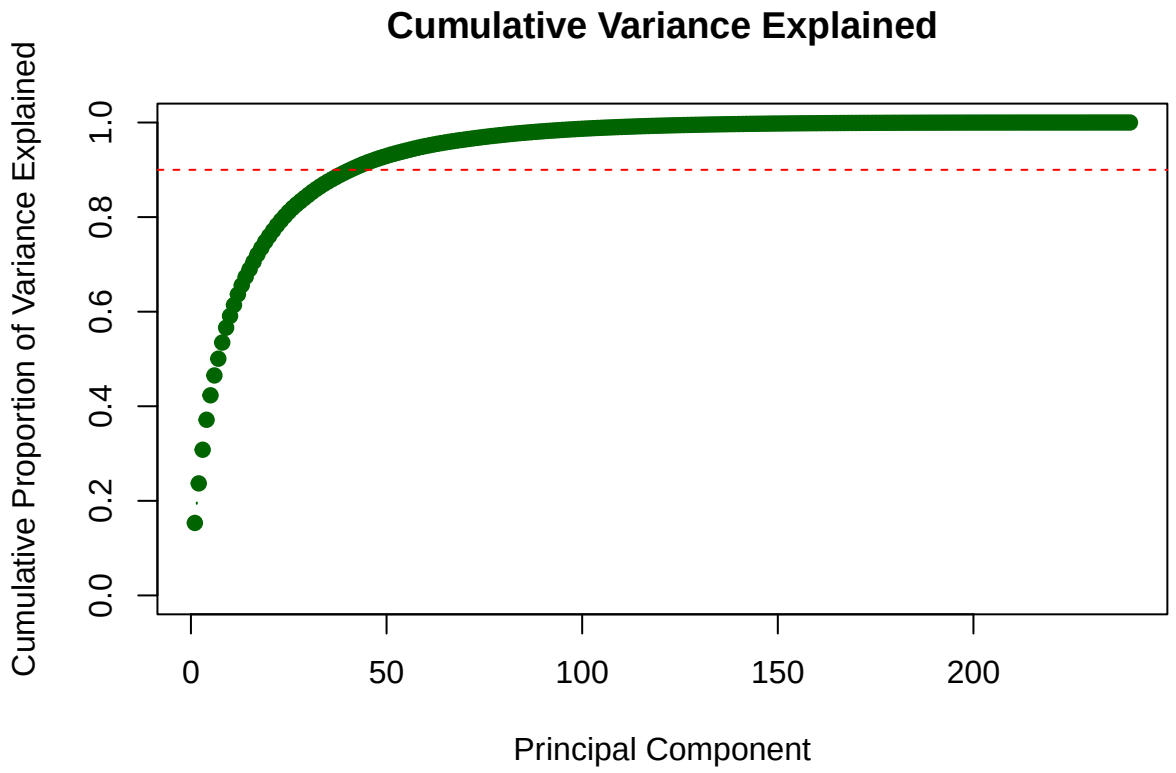
2.C)

```r
eigenvalues <- pca$sdev^2

par(mfrow = c(1, 1))  # Ensure single plot layout
```

```r
# Create a bar plot and store the bar positions
bar_positions <- barplot(
  eigenvalues,
  main = "Eigenvalues of Principal Components",
  xlab = "Principal Component",
  ylab = "Eigenvalue",
  col = "lightblue",  # Use a distinct color for bars
  border = "black",   # Black border around bars
  ylim = c(0, max(eigenvalues) * 1.1)  # Add some space at the top for labels
)

#Add eigenvalue labels on top of each bar
text(
  bar_positions,
  eigenvalues,
  labels = round(eigenvalues, 2),  # Round to 2 decimal places
  pos = 3,  # Position the text above the bars
  cex = 0.8,  # Text size
  col = "black"  # Text color
)
#Add axis labels with integer principal component indices
axis(1, at = bar_positions, labels = 1:length(eigenvalues), las = 2)  # X-axis labels
```



```r
cumulative_variance <- cumsum(eigenvalues) / sum(eigenvalues)
plot(cumulative_variance, type = "b", pch = 19, col = "darkgreen",
     xlab = "Principal Component", ylab = "Cumulative Proportion of Variance Explained",
     main = "Cumulative Variance Explained",
     ylim = c(0, 1))
abline(h = 0.9, col = "red", lty = 2)  # Reference line for 90% variance explained
```



Based on both plots, approximately 15 to 20 principal components are relevant, as they explain around 90% of the variance. After that point, the additional components contribute minimally to the overall variance, making them less important for the analysis.

```r
## let's apply North 'rule of thumb' to the result of the pca analysis to determine how many PCs should

eigen_vals <- pca$sdev^2

for (i in 1:(length(eigen_vals)-1)){
    if ((eigen_vals[i] - eigen_vals[i+1])/eigen_vals[i] < sqrt(2/length(eigen_vals))) {
        pc_opt <- i
        print(paste0("According to North's rule of thumb the first ", i, " PCs should be kept and the r
        break
    }
}
```

```
## [1] "According to North's rule of thumb the first 7 PCs should be kept and the rest should be trunca
```

```r
print(paste0("With ", i, " PCs we can explain ", round(cumsum(pca$sdev^2)/sum( pca$sdev^2), 3)[pc_opt]
```

```
## [1] "With 7 PCs we can explain 0.501 of variance in the data!"
```

Problem 3

3.A)

```r
# Load necessary library
library(MASS)

# Define the mean vector and covariance matrix
mu <- c(2, 5)
Sigma <- matrix(c(1, 1/2, 1/2, 1), nrow = 2)

# Calculate eigenvalues and eigenvectors
eigen_decomp <- eigen(Sigma)
eigenvalues <- eigen_decomp$values
eigenvectors <- eigen_decomp$vectors

# Output the eigenvalues and eigenvectors
print("Eigenvalues:")
```

```
## [1] "Eigenvalues:"
```

```r
print(eigenvalues)
```

```
## [1] 1.5 0.5
```

```r
print("Eigenvectors:")
```

```
## [1] "Eigenvectors:"
```

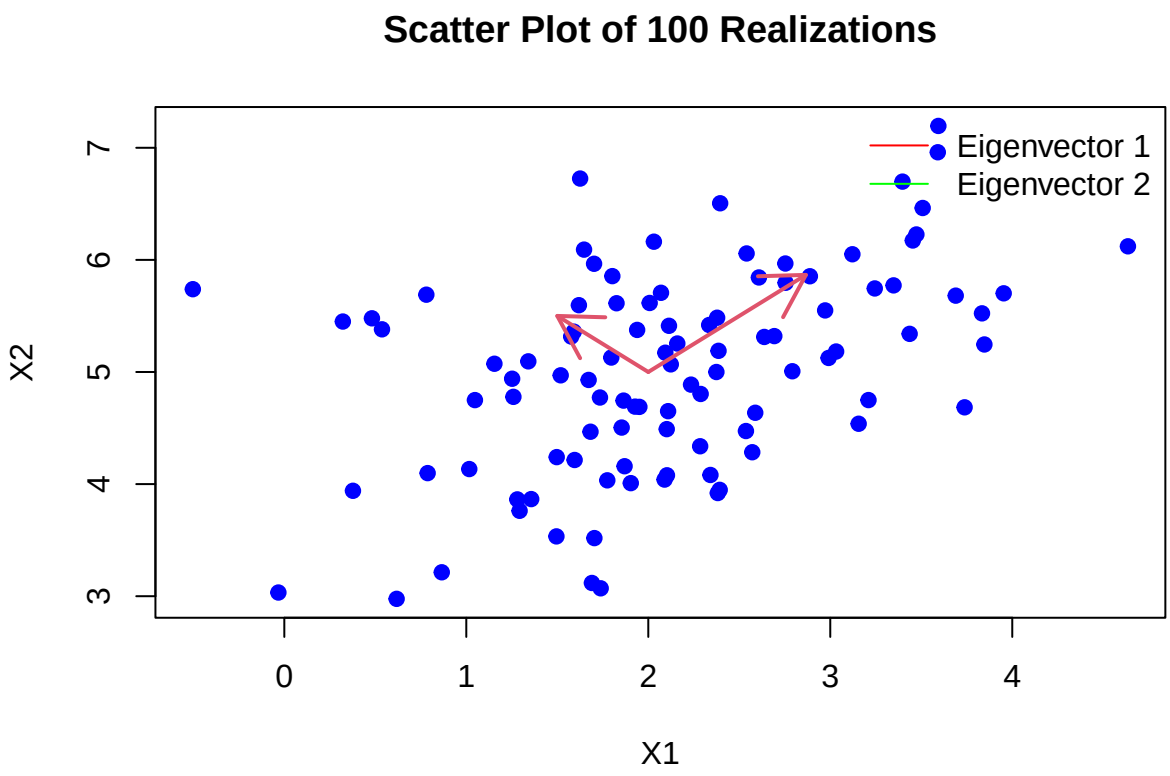
```r
print(eigenvectors)
```

```
##           [,1]       [,2]
## [1,] 0.7071068 -0.7071068
## [2,] 0.7071068  0.7071068
```

3.B & 3.C)

```r
set.seed(123)  # Set seed for reproducibility
n <- 100
data <- mvrnorm(n, mu, Sigma)
```

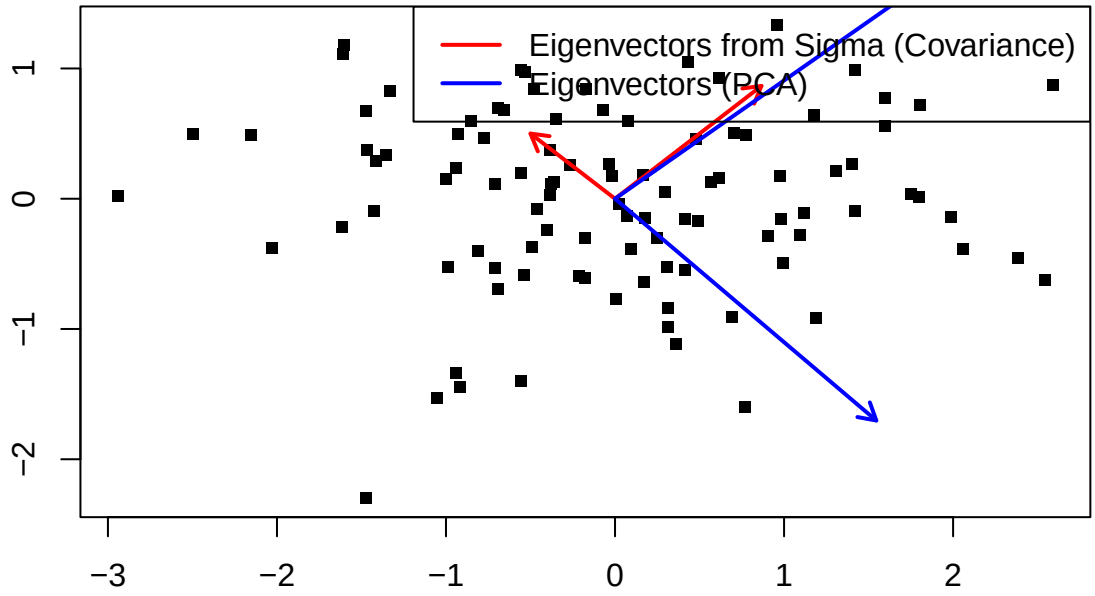
```r
plot(data, main = "Scatter Plot of 100 Realizations", xlab = "X1", ylab = "X2", pch = 19, col = "blue")

arrows(2, 5, 2+sqrt(eigenvalues)*eigenvectors[1,], 5+sqrt(eigenvalues)*eigenvectors[2,], col = 2, lwd =

legend("topright", legend = c("Eigenvector 1", "Eigenvector 2"), col = c("red", "green"), lty = 1, bty =
```



3.D)

```r
set.seed(1)
# perform pcs
pca <- prcomp(data, scale=FALSE)



plot(pca$x, pch='.', xlab='', ylab='', cex = 6)



arrows(0, 0, eigenvectors[1, 1] * sqrt(eigenvalues[1]), eigenvectors[2, 1] * sqrt(eigenvalues[1]),
       col="red", lwd=2, length=0.1)
arrows(0, 0, eigenvectors[1, 2] * sqrt(eigenvalues[2]), eigenvectors[2, 2] * sqrt(eigenvalues[2]),
       col="red", lwd=2, length=0.1)
```

```r
loadings <- pca$rotation  # Get PCA loadings (eigenvectors)

arrows(0, 0, loadings[1, 1] * max(abs(pca$x[, 1])), loadings[2, 1] * max(abs(pca$x[, 1])),
       col="blue", lwd=2, length=0.1)
arrows(0, 0, loadings[1, 2] * max(abs(pca$x[, 2])), loadings[2, 2] * max(abs(pca$x[, 2])),
       col="blue", lwd=2, length=0.1)


legend("topright", legend=c("Eigenvectors from Sigma (Covariance)", "Eigenvectors (PCA)"), col=c("red",
```



« Comments »

- the descrepency may arise from two sources:

1. Data Variability: PCA uses the empirical covariance of the data, which may vary slightly from the theoretical covariance defined in sigma due to the randomness in the generated data.
2. Scaling and Centering: PCA works with centered data (mean-subtracted), while the eigenvalues of sigma represent the theoretical model, which assumes the data is centered.