Data Science Methods - Assignment 1

M. Alberti, s.n. 2020162 N. Ceschin, s.n. 344510

February 21, 2020

Question 1

First we upload all relevant libraries:

```
library(readxl)
library(ggplot2)
library(ggfortify)
library(dplyr)
library(tidyr)
library(RCurl)
library(ggrepel)
```

Upload dataset:

```
setwd("C:/Users/Mr Nobody/Desktop/Uni/EME/Data science Methods/Assignments")
#setwd("~/Tilburg/Courses/Data Science Methods/Assignment1/DATA-SCIENCE-ASSIGNMENTS")
data<-read_excel("env_air_emis.xls")
urlRemote<-"https://raw.githubusercontent.com/"
pathGithub<-"AlbertiMarco/DATA-SCIENCE-ASSIGNMENTS/master/EU%20labels.csv"
x <- getURL(paste0(urlRemote, pathGithub))      #country tags to make plots more readable
EU_labels<- read.csv(text = x, header = FALSE ,sep=";")
rownames(EU_labels)<-EU_labels[[1]]
```

After a quick glimpse of the data we realized that information for the five pollutant are presented in separated consecutive tables, the separation contains some information in the first column and NA cells in the rest. To be sure not to drop NAs in the middle of the dataset, we first proceed to drop all raws containing at least 5 NA values and we assign to *df*:

```
dim(data)
df<-data[rowSums(is.na(data))<length(data)-5,]
```

Given the data structure and the subpoints a-c requests, we decided that the optimal approach would be looping over the chunks of data containing information for each pollutant, producing without repeting the code the outputs all in one step. Unfortunately, given the canvas announcement this was not necessary, but since by that time we already had the code for looping over a-c requests, we decided to keep it.

First we create some variables that will be used in the loop:

```
#build 'index' for your loop
interval<-c(1,30,59,88,117)                #first row of each datset
pollutants<-c("ammonia","nmvoc","smallpart","largepart","sulphur")
index<-data.frame(interval,pollutants)

PC1<-data.frame(matrix(ncol=5,nrow=28))    #dataframes to be filled with PC1-2
PC2<-data.frame(matrix(ncol=5,nrow=28))
```

Then we run the loop that:

1) Selects the chunk of the dataframe corresponding to one pollutant and puts it into shape
2) Runs PCA on the reduced dataframe and produces the plots of the first two PCs
3) Produces a scree plot
4) Computes the Bayesian Information Criteria and prints the optimal number of PCs according to this method
5) Stores plots and loadings for future use

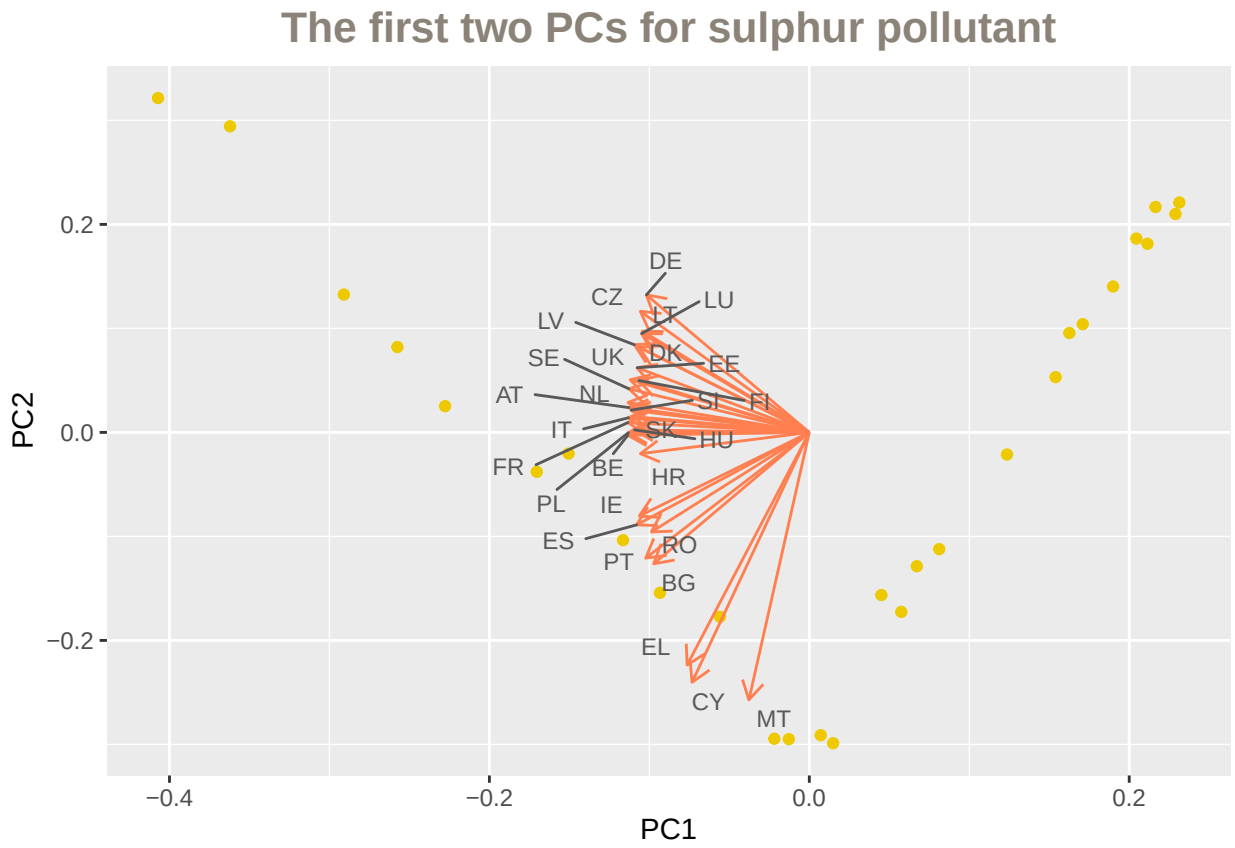
```
mytheme <- theme(plot.title= element_text(face="bold",
  colour = "antiquewhite4",size = (16),hjust = 0.5))     #setting theme for plots

for (i in 1:5){
  #Data chunk preparation#
  begin<-index[i,1]                          #select the right range to cut main dataset
  end<-index[i,1]+28
  dfx<-df[begin:end,]
  dfx<-as.data.frame(dfx)
  colnames(dfx)<-dfx[1,]
  rownames(dfx)<-dfx[,1]
  dfx<-dfx[c(2:29),c(2:29)]                  #drop first column and row
  if (sum(mapply(grepl,rownames(EU_labels),rownames(dfx)))==length(dfx)) {
  rownames(dfx)<- EU_labels[[2]]
    }                                        #substitute name with short labels
  dfx<-as.data.frame(t(dfx))                 #convert factor columns into numeric
  indx <- sapply(dfx, is.factor)
  dfx[indx] <- lapply(dfx[indx], function(x) as.numeric(as.character(x)))


  #Principal Component Analysis
  pr.out<-prcomp(dfx, scale=TRUE)
  graph<-autoplot(pr.out,variance_percentage=FALSE,loadings=TRUE,
           loadings.label=TRUE,loadings.label.repel=TRUE,loadings.colour="coral",
           loadings.label.size=3,loadings.label.colour="grey35",scale=TRUE,colour="gold2")+
           ggtitle(paste("The first two PCs for",toString(index[i,2]),"pollutant"))+
           mytheme

  #Screeplot#
  pve= 100* (pr.out$sdev ^2)/ sum(pr.out$sdev ^2)
  pvedf<-as.data.frame(pve)
  pvedf["PC"]<-c(1:28)
  scree<-ggplot(pvedf, aes(x=factor(PC),y=pve, group=1))+
    geom_point(size =2.25,color="blue")+geom_line(size = 1,color="blue")+
    labs(x="Number of principal components",y="Proportion of variance explained")+
    ggtitle(paste("Screeplot for PCs of ",toString(index[i,2]))) + mytheme

  #Compute vector of BIC for first 27 principal components#
  BIC<-c(1:27)
  for (j in 1:27) {
    f<-pr.out$x[,1:j]%*%t(pr.out$rotation[,1:j]) #compute aF in X=aF+e
    res_mat<-scale(dfx)-f                        #compute matrix of residuals
    res_mat_sq<-res_mat*res_mat
    if (j==1){
      res<-(sum(res_mat_sq)/(dim(dfx)[1]*dim(dfx)[2]))
    } else{
```

```
    res<-(sum(rowSums(res_mat_sq))/(dim(dfx)[1]*dim(dfx)[2]))   #residuals sum of squares
    }
    k<-j
    BICk<-log(res)+k*(log(28^2))/(28^2)
    BIC[j]<-BICk                                  #fill BIC vector at each iteration
    }
  min<-min(BIC)
  num_pc<-match(min,BIC)                          #number of PC selected by BIC


  #Save first two PC in separate dataset for point d)#
  PC1[i]<-pr.out$x[,1]
  colnames(PC1)[i]<-as.character(index[i,2])
  PC2[i]<-pr.out$x[,2]
  colnames(PC2)[i]<-as.character(index[i,2])

  #save relevant objects with their respective name for each pollutant#
  assign(paste0("BIC_", index[i,2]), BIC)        #vector of BIC
  assign(paste0("df_", index[i,2]), dfx)         #dataset
  assign(paste0("prcomp_",index[i,2]),pr.out)    #output of prcomp
  assign(paste0("Screeplot_",index[i,2]),scree) #scree plot
  assign(paste0("PC1-PC2_",index[i,2]),graph)    #plot of first two PCs
  assign(paste0("num_pc_", index[i,2]), num_pc) #number of PC selected by BIC

  #remove non relevant objects
  rm(dfx)
  rm(BIC)
  rm(pr.out)
  rm(num_pc)
  rm(scree)
}
```

Now to answer a)-d) questions we just call the relevant stored objects.

a)

```
print(`PC1-PC2_sulphur`)
```



Were we to disregard the statistical pitfalls of this exercise, we would interpret the results from this plot as follows. To ease our analysis, we will present as well some basic descriptive statistics about the distribution of the first two principal components' loadings.

```
summary(prcomp_sulphur$rotation[,1])
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.20549 -0.20321 -0.19402 -0.18680 -0.18810 -0.06845
```

At a first glance, we notice that the loadings on the first principal component are extremly stable in their magnitude. They range from -0.205 to -0.068, but 75% of them lies between -0.20 and -0.18. In line with the interpretation perspective presented in class as well as in the book *An Introduction to Statistical Learning*, this result doesn't help to draw any meaningful conclusion. According to this view, a principal component "represents" the subset of variables for which its loadings are higher. However, in this case there is no variable whose corresponing loading stands out for its particular magnitude (relative to the others). Hence, we cannot say whether this component "corresponds" to a particular subset of variables.

```
summary(prcomp_sulphur$rotation[,2])
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.46580 -0.14947  0.02658 -0.02133  0.09684  0.23947
```

The second vector of loadings displays a completely different situation. The range of variability is definitely larger (from -0.46 to 0.23), with a vector of loadings with both positive and negative values (although sign cannot be identified). Also, these coefficients are more uniformely distributed along this range. In line with

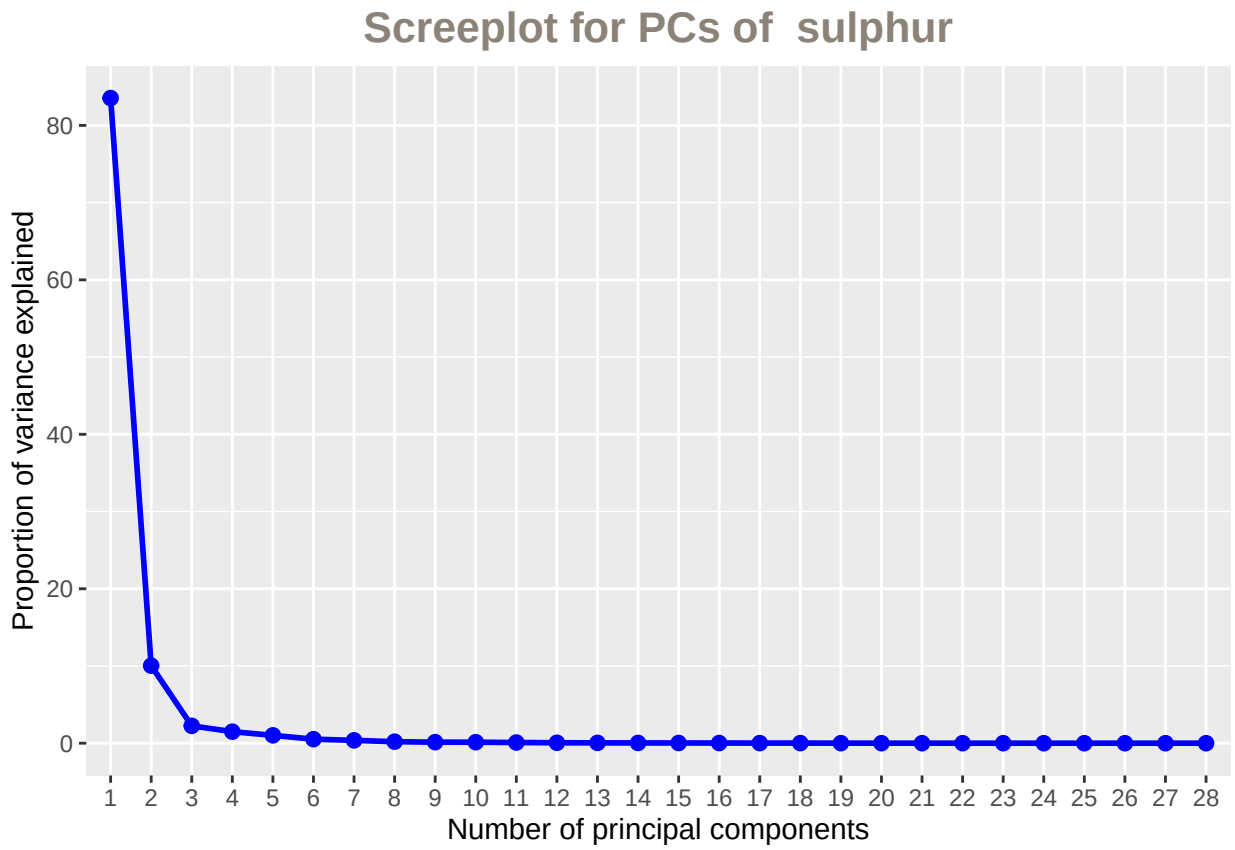
the previous interpretation perspective, we can conlude that the second principal component "corresponds" to the set of countries whose loadings are the highest. At a first glance, the countries which display higher values are all from Northern Europe, while those displaying lower values are all from Southern Europe. Needless to say, this is just a rough visual analysis and futher investigation would be necessary in order to understand which is the real driver of these values.

The problem which arises when we interpret principal components' scores and loadings is that they are usually identified only up to a rotation. In view of a more general "factor model" as $X = F\Lambda^{'} + e$, under a set of assumption laid down in Bai & Ng (2002) (which we present in Question 2) principal components analysis estimates doesn't esimate F and Λ, but FH (a rotation of F) and $\Lambda H^{'-1}$ (a rotation of Λ). This means that it is not possible to interpret straightforwardly signs and magnitudes of the estimated factor and factors loadings. Bai and Ng (2012), with a later paper, provides a set of assumptions under which this is possible. Indeed, were these assumptions to hold for F anf Λ, then the corrisponding rotation matrix would be asymptotically an identity matrix. Hence, we would achieve exact identification. However, due to the very tiny dimension of the datset under analysis, we reasonably exclude asymptotic results to hold in this PC analysis.

In this case PCA aims to find the linear combination of variables able to explain a large part of the variance in the data. Meaning that here it is used as a dimension's reduction technique. For this specific case we are trying to find a linear combination of countries that are able to explain most of the variation in the air pollutant over years.

b)

```
print(Screeplot_sulphur)
```



Based on the screeplot, it would be straightforward to choose only one principal component. This derives from the fact that the first principal component explains more than 80% of the total variance of all the principal components. The variance explained by the second one amounts only to 10% while that explained by the rest of principal components is trivial. However, this conclusion may not be reliable. Indeed, it is worth to remember that these values are random variables and they can change were other samples to be drawn. It is possible to implement some statistical tests in order to understand, for example, whether two principal components explain the same proportion of the total variance or not, but these kind of tests are usually extremly complicated, also due to the fact that the distribution of these random variables are difficult to handle.

c)

```
cat("According to the BIC criterion, the optimal number of principal components is ",
    num_pc_sulphur)
```

```
## According to the BIC criterion, the optimal number of principal components is  27
```

The conclusion we draw by using the BIC criterion is completely at odds with that we drawn from the analysis of the screeplot. The BIC picks the maximum number of principal components available (given the restriction $k < p$), while in the previous point we appreciated that all the principapl components from 2 to 27 explain a trivial part of the total variance. By looking more closely at how the BIC is computed in this case, we notice that this result is due to the fact that while the "fit" part decreases rapidly and monotonically as the number of principal components increases, the "penalty" part is rather stable. Hence, the model

with the maximum number of principal components presents the lowest BIC. This fact is common to all the pollutants. A first hypothesis to try to rationalize this result is that we know that the BIC criterion is consistent only asymptotically(for both p and n). Namely, we know that the probablilty that the number of principal components selected by the BIC is the true nuumber of principal components of the model approaches 1 only for very large numbers of observations and variables. The datasets provided clearly don't meet this condition, having dimension 28×28. We can then conclude that in our opinion the reason why the BIC criterion yields such a strange result in this case is that a fundamental assumption is violated, and thus the results come from small sample bias.

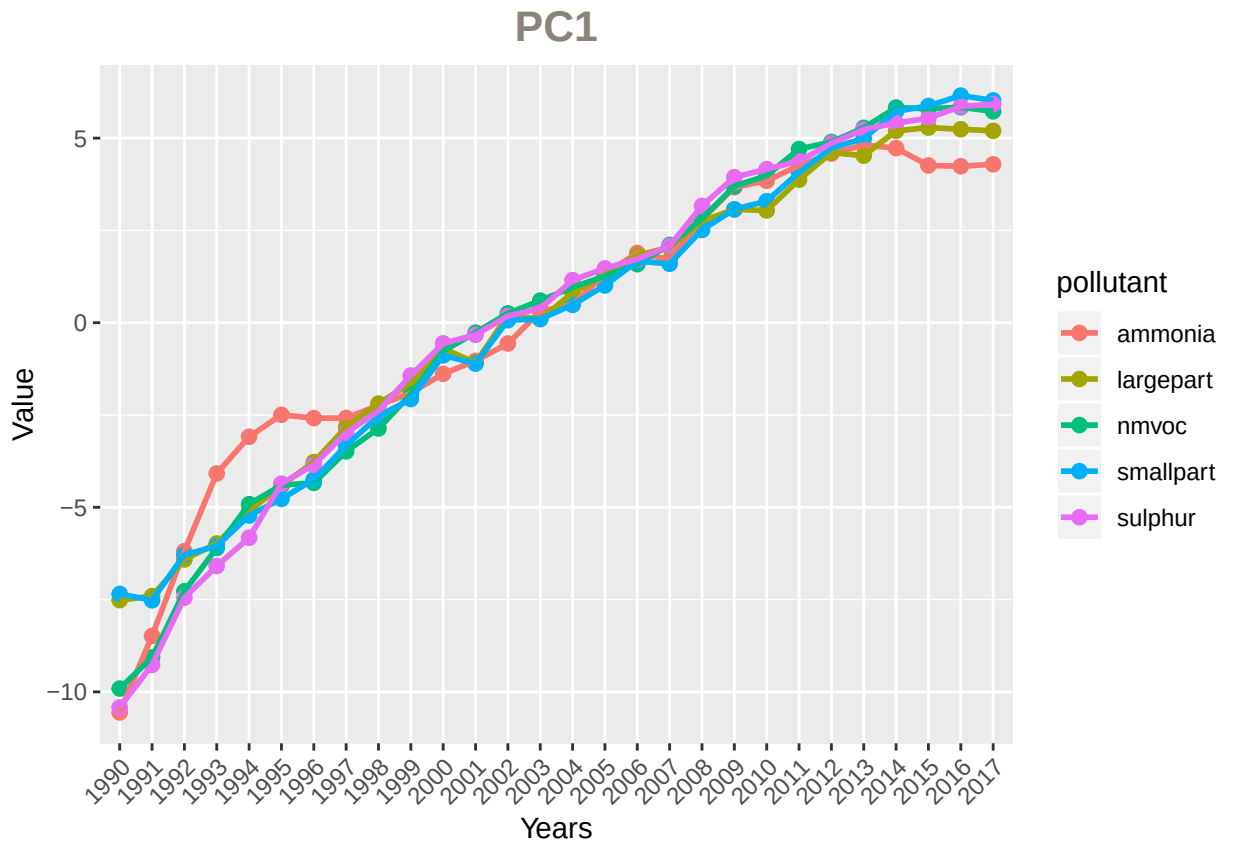
d)

We plot the first 2 principal components for all the five pollutants over the time interval.

Code to produce the plots:

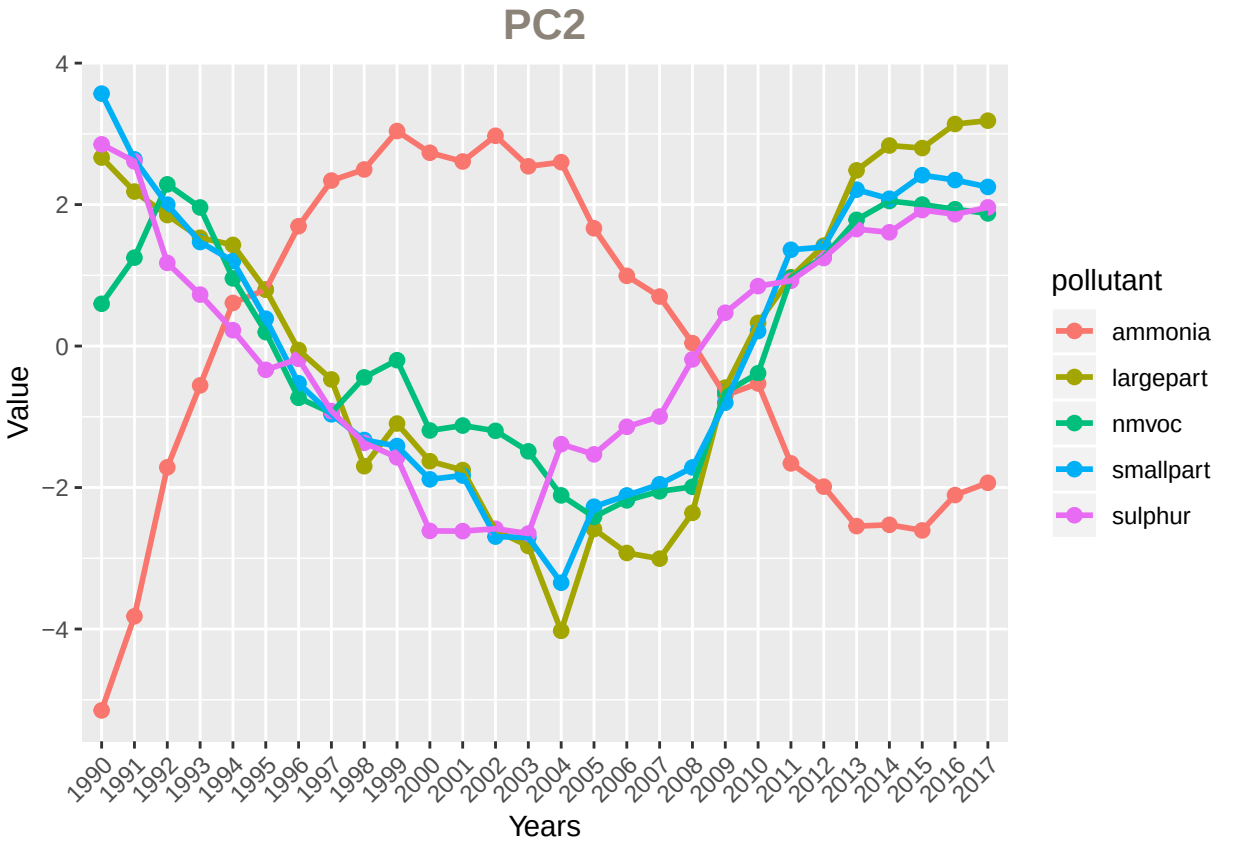
```
theme2<-theme(axis.text.x = element_text(angle = 45, hjust=1))

PC1["years"]<-c(1990:2017)
PC1<-gather(PC1, `ammonia`, `largepart`, `nmvoc`,
            `smallpart`, `sulphur`, key = "pollutant", value = "value")
PC1_plot<-ggplot(PC1,aes(x=factor(years),y=value, group=pollutant,color=pollutant))+
  geom_point(size = 2.25) +  geom_line(size = 1) +theme2 +mytheme+
  labs(title = "PC1",x="Years",y="Value")

PC2["years"]<-c(1990:2017)
PC2<-gather(PC2, `ammonia`, `largepart`, `nmvoc`,
            `smallpart`, `sulphur`, key = "pollutant", value = "value")
PC2_plot<-ggplot(PC2,aes(x=factor(years),y=value, group=pollutant,color=pollutant))+
  geom_point(size = 2.25) +  geom_line(size = 1) +theme2 +mytheme+
  labs(title = "PC2",x="Years",y="Value")
```

```
print(PC1_plot)
```



By plotting the values of the first principal component for all the pollutants, we can immediately appreciate how the trends of all of them are extremly similar, increasing monotonically and linearly. The fact that they are alike even in levels arises probably from having rescaled all the five datasets before running the PCA.

```
print(PC2_plot)
```



Looking at the second principal component, we notice a different situation: while the trend and the level of four pollutants are again extremly similar (more u-shaped this time) that of ammonia behaves in a completely opposite way. Indeed, it appears almost exaclty complementary to that of the other four pollutants.

Question 2

A factor model is defined as:

$$X_{ij} = \alpha_{j1} f_{i1} + \alpha_{j2} f_{i2} + ... + \alpha_{jr} f_{ir} + \epsilon_{ij} = \alpha_j^T f_i + \epsilon_{ij}$$

with i=1,...,n ; j=1,...,p and r number of factors $f_i = \begin{bmatrix} f_{i1} \\ f_{i2} \\ \vdots \\ f_{ir} \end{bmatrix}$ and $\alpha_j = \begin{bmatrix} \alpha_{j1} & .. & \alpha_{jr} \end{bmatrix}$

In matrix form we can write:

$$\underset{n\times p}{X} = \underset{n\times r}{F}\ \underset{r\times p}{B^T} + \underset{n\times p}{\epsilon}$$

In general the expression above is not identified since we don't observe neither B nor F. If we consider a particular rotation matrix H, with $H^T H = I_r$ then we can rewrite the equation above as:

$$X = \underbrace{FH}_{F^*}\underbrace{H^T B^T}_{B^*} + \epsilon$$

In the particular case of PCA, the NIPALS algorithm can find a unique solution because PCA imposes a specific rotation matrix H. In particular, the rotation matrix imposes restrictions on B, on one hand $\frac{r(1-r)}{2}$ restrictions of orthogonality of the loadings and r on the norm of the loadings (normalization):

(i) $a_i a_j = 0$ when $a_i \neq a_j$

(ii) $a_i^T a_i = 1$

Of course the PCA solution is just one of the possibles, any restrictions that identify the model can lead to a unique solution(for example restrictions on the factors instead).

In a factor model we believe that observed variables (Xs) are linear combinations of a limited number of underlying and unique factors (Zs); whereas in PCA component scores (Zs) are a linear combination of the observed variables (Xs) weighted by eigenvectors. The relationship between the two can be summarized by:

$$XH = UDH^T H = UD = Z$$

with $U^T H = I_p$, $H^T H = HH^T = I_p$, $D = diag(d)$ of size $p \times p$. This shows that there is a transformation matrix (inverse of H, which in the case of orthogonal H is equal to H^T) for which the explanatory variables are linear combinations of the unobserved unique factors Z, namely $X = ZH^T$. This is exactly what NIPALS exploits in order to estimate consistently the PC solution. Indeed, in NIPALS we are repeatedly regressing our matix of data on estimates of loadings and principal components which get recursively updated. Thus, NIPALS is assuming exactly that we can express our data as linear combinations of the principal components. To appreciate how this algorithm exploits this relationship, it is worth to summarize its main steps. NIPALS starts assuming a guess for the first principal component, Z_1^I. Then, it runs a linear regression of each column (variable) of our dataset on this initial prinicipal component. It is possible to obtain unbiased estimates of the coefficients on Z_1^I since we assume the error term in this regression to be independent of the regressor (exogeneity holds), as principal components are supposed to be independent among them. After having standardized these first coefficients, we exploit again the relationship we mentioned above by running a regression of each row of the datset on the vector of coefficients we estimated before. Again, the loadings in PCA are supposed to be independent among them, hence we assume exogenity holds in this regression. We obtain then new estimates of the principal component Z_1^I, Z_1^{II}, that we use as new initial value for the algorithm. NIPALS stops when the distance between Z_1^I and Z_1^{II} within an iteration is sufficienlty small. Also, it can be set to compute as many principal components as desirable.

Consistency of NIPALS Algorithm and factor analysis

In class we have seen how the NIPALS algorithm can be used to consistently estimate the scores in a principal components analysis (PCA) when the traditional optimization approach fails or is inefficient given the large number of parameters (p). In particular, we limited ourselves to the case in which $n > p$ and the matrix D of eigenvalues has rank p.
The starting point is

$$X = F^* B^* + \epsilon$$

Under the assumptions:

1. ϵ_{ij} are iid over i and j (can be relaxed to weakly dependent)
2. $$E||f_i||^4 < M$$
$$\frac{1}{n}\sum_{i=1}^{\infty} f_i f_i^T \xrightarrow{p} \Sigma_F$$
which is a positive definite matrix of constants. This assumption is related to saying that the matrix D of eigenvalues has rank p, since positive definiteness for a symmetric matrix can be defined as having a diagonal of non-zero elements
3. f_{ij}, ϵ_{ij} are independent for all j (Exogeneity Assumption)

4. $\underset{r\times r}{H}$ is defined as above (r are restrictions)

Given these assumptions we can consistently estimate $F^* = FH$ by $\hat{F}$ meaning that we have:

1. $\hat{f}_i \xrightarrow{p} Hf_i$ pointwise consistency
2. $min(\sqrt{n}, \sqrt{p})[\hat{f}_i - Hf_i] \xrightarrow{d} N(0, HV(f_i)H^T)$ for $min(\sqrt{n}, \sqrt{p}) \to \infty$

If the assumptions 1-4 are met, also NIPALS method with the PCA restrictions can identify a factor model up to a rotation matrix H. Notice that for consistency to hold, we need both the number of observations n and the number of variables p to be large.

Question 3

a) $p_1(x) = \frac{Pr(Y=1)Pr(X=x|Y=1)}{\sum_{l=1}^{K} Pr(Y=l)Pr(X=x|Y=1)}$

In this case we have that:

- $\pi_k = Pr(Y = k)$ is prior probability for class k
- $X|Y = k \sim f_k(x) = N(\mu_k, \sigma^2)$ is the distribution of X for each class k

Using the Bayes rule to rewrite:

$$p_1(x) = \frac{\pi_1 f_1(x)}{\pi_1 f_1(x) + \pi_2 f_2(x)}$$

In order to ease calculations in the next step, it is convenient to keep implicit the likelihood functions for the moment.

b) Plugging in the posterior probability of class 1 obtained in the previous point, we obtain:

$$\log\left(\frac{p_1(x)}{1-p_1(x)}\right) = \log\left(\frac{\frac{\pi_1 f_1(x)}{\pi_1 f_1(x)+\pi_2 f_2(x)}}{1-\frac{\pi_1 f_1(x)}{\pi_1 f_1(x)+\pi_2 f_2(x)}}\right) = \log\left(\frac{\pi_1 f_1(x)}{\pi_2 f_2(x)}\right)$$

We can now explicit the likelihood functions and simplify conveniently

$$\log\left(\frac{p_1(x)}{1-p_1(x)}\right) = \log\left(\frac{\pi_1}{\pi_2}\right) + \log\left(\frac{f_1(X)}{f_2(x)}\right) = \log\left(\frac{\pi_1}{\pi_2}\right) + \log\left(\exp\left[-\frac{1}{2\sigma^2}\left((x-\mu_1)^2 - (x-\mu_2)^2\right)\right]\right)$$

$$= \log\left(\frac{\pi_1}{\pi_2}\right) - \frac{1}{2\sigma^2}(-2x\mu_1 + \mu_2^2 + 2x\mu_2 - \mu_2^2) = \left[\log\left(\frac{\pi_1}{\pi_2}\right) + \frac{\mu_2^2 - \mu_1^2}{2\sigma^2}\right] + x\left[\frac{\mu_1 + \mu_2}{\sigma^2}\right] =$$

Therefore,

- $c_0 = \log\left(\frac{\pi_1}{\pi_2}\right) + \frac{\mu_2^2 - \mu_1^2}{2\sigma^2}$
- $c_1 = \frac{\mu_1 + \mu_2}{\sigma^2}$

c) This derivation helps us appreciating how these two classfication methods are closely related. In particular, the log-odds ratio is linear in x in both cases. In the logit model, the parameters estimated by maximum-likelihood are exactly the intercept and the coefficient on the x, while in LDA c_0 and c_1 are functions on the prior probabilities and of the paramters of the likelihood functions μ_1, μ_2, σ^2. We could conclude that the parameters estimated by the logit model are somehow *reduced forms* of those of LDA.

d) Knowing that the Bayes classifier assigns an observation $X = x$ to the class k for which the posterior probability $P(Y = k|X = x)$ is the largest, we start by calculating the posterior probability of beloning to a generic class:

$$p_k(x) = \frac{Pr(Y=k)Pr(X=x|Y=k)}{\sum_{l=1}^{K} Pr(Y=l)Pr(X=x|Y=1)} = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}$$

Therefore, the Bayes classifier assign $X = x$ to class 1 if $p_1(x) > p_2(x)$. We can restate this rule as

$$\frac{\pi_1 f_1(x)}{\sum_{l=1}^{K} \pi_l f_l(x)} > \frac{\pi_2 f_2(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}$$
$$\iff$$
$$\pi_1 f_1(x) > \pi_2 f_2(x)$$

By making the density function explicit, the constant terms simplifies. By taking logs on both sides, we are left with

$$-\frac{1}{2}\left(x^T\Sigma^{-1}x - 2x^T\Sigma^{-1}\mu_1 + \mu_1^T\Sigma^{-1}\mu_1\right) + \log(\pi_1) > -\frac{1}{2}\left(x^T\Sigma^{-1}x - 2x^T\Sigma^{-1}\mu_2 + \mu_2^T\Sigma^{-1}\mu_2\right) + \log(\pi_2)$$
$$\iff$$
$$-\frac{1}{2}(x^T\Sigma^{-1}x) + x^T\Sigma^{-1}\mu_1 - \frac{1}{2}(\mu_1^T\Sigma^{-1}\mu_1) + \log(\pi_1) > -\frac{1}{2}(x^T\Sigma^{-1}x) + x^T\Sigma^{-1}\mu_2 - \frac{1}{2}(\mu_2^T\Sigma^{-1}\mu_2) + \log(\pi_2)$$
$$\iff$$
$$x^T\Sigma^{-1}\mu_1 - \frac{1}{2}(\mu_1^T\Sigma^{-1}\mu_1) + \log(\pi_1) > x^T\Sigma^{-1}\mu_2 - \frac{1}{2}(\mu_2^T\Sigma^{-1}\mu_2) + \log(\pi_2)$$

Which corresponds exactly to $\delta_1(x) > \delta_2(x)$.