Air Pollution Data Analysis

Library import

```r
suppressWarnings({
  library(tree)
  library(knitr)
}) # supressing warning
```

Part I – Supervised Learning (Classification)

Task 1: Data Pre-processing

1.i) Import the dataset and convert qualitative variables into a factor variables.

Importing dataset and viewing first few rows:

```r
glass = read.csv('C:/Users/rosmi/OneDrive/Desktop/Sem 2/Intro to DS/glass(2).csv')
head(glass)
```

```
##        RI    Na   Mg   Al    Si    K   Fe   Type
## 1 1.52101 13.64 4.49 1.10 71.78 0.06 0.00 Window
## 2 1.51761 13.89 3.60 1.36 72.73 0.48 0.00 Window
## 3 1.51618 13.53 3.55 1.54 72.99 0.39 0.00 Window
## 4 1.51766 13.21 3.69 1.29 72.61 0.57 0.00 Window
## 5 1.51742 13.27 3.62 1.24 73.08 0.55 0.00 Window
## 6 1.51596 12.79 3.61 1.62 72.97 0.64 0.26 Window
```

```r
attach(glass)
```

Converting qualitative variables into factor variables is a crucial preprocessing step in supervised learning classification. It ensures machine learning models to effectively utilize categorical information, leading to better model performance, compatibility with algorithms, and interpretability. That's why question has asked us to convert qualitative variables into a factor variables. So we examine the type of variables in our dataset with the help of str() function and convert those qualitative variables into a factor variables:

```r
names(glass)
```

```
## [1] "RI"   "Na"   "Mg"   "Al"   "Si"   "K"    "Fe"   "Type"
```

```
str(glass)
```

```
## 'data.frame':    214 obs. of  8 variables:
##  $ RI  : num  1.52 1.52 1.52 1.52 1.52 ...
##  $ Na  : num  13.6 13.9 13.5 13.2 13.3 ...
##  $ Mg  : num  4.49 3.6 3.55 3.69 3.62 3.61 3.6 3.61 3.58 3.6 ...
##  $ Al  : num  1.1 1.36 1.54 1.29 1.24 1.62 1.14 1.05 1.37 1.36 ...
##  $ Si  : num  71.8 72.7 73 72.6 73.1 ...
##  $ K   : num  0.06 0.48 0.39 0.57 0.55 0.64 0.58 0.57 0.56 0.57 ...
##  $ Fe  : num  0 0 0 0 0 0.26 0 0 0 0.11 ...
##  $ Type: chr  "Window" "Window" "Window" "Window" ...
```

From the output above we can see that the variable "Type" is qualitative variable. So, now we change it to factor variable. To change it to factor variable we use as.factor() function:

```
glass$Type = as.factor(glass$Type)
```

Now lets check the type of "Type" variable:

```
str(glass)
```

```
## 'data.frame':    214 obs. of  8 variables:
##  $ RI  : num  1.52 1.52 1.52 1.52 1.52 ...
##  $ Na  : num  13.6 13.9 13.5 13.2 13.3 ...
##  $ Mg  : num  4.49 3.6 3.55 3.69 3.62 3.61 3.6 3.61 3.58 3.6 ...
##  $ Al  : num  1.1 1.36 1.54 1.29 1.24 1.62 1.14 1.05 1.37 1.36 ...
##  $ Si  : num  71.8 72.7 73 72.6 73.1 ...
##  $ K   : num  0.06 0.48 0.39 0.57 0.55 0.64 0.58 0.57 0.56 0.57 ...
##  $ Fe  : num  0 0 0 0 0 0.26 0 0 0 0.11 ...
##  $ Type: Factor w/ 2 levels "Non-window","Window": 2 2 2 2 2 2 2 2 2 2 ...
```

We can see the output above, the "type" variable has now converted into a factor variable. And it exhibits two levels: 'Non-window' and 'Window'.

1.ii) Divide the dataset into two sets randomly assigning 70 percent of the observations for training set and 30 percent for testing set (Set the seed 1).

To divide the dataset into two sets randomly, we first set the seed so that it would ensure the result reproducibility for our analysis. After seeding seed we divide our dataset into two sets randomly assigning 70 percent of the observations for training set and 30 percent for testing set:

```
set.seed(1)
tr1 = sample(1:nrow(glass), round(nrow(glass)*0.7))
trainset = glass[tr1, ]
testset = glass[-tr1, ]
```

Now lets check the dimensions of our datasets to confirm that we got the right split as asked by the question:

```
dim(glass)
```

```
## [1] 214   8
```

```
dim(trainset)
```

```
## [1] 150   8
```

```
dim(testset)
```

```
## [1] 64  8
```

It looks like we've done right split as asked by our question, i.e, 70 percent of obs for training set and 30 percent for the testing set.

Now that we've made our dataset ready, we can proceed further to our analysis:

Task 2: Logistic Regression Model

2.i) Fit a logistic regression model for the training dataset (in question 1) to classify the type of glass using the other variables as predictors. Give the summary output.

As asked by the question, we make use of the training dataset and calculate the logistic regression model:

```
logistic_model1 <- glm(Type ~ ., data = trainset, family = binomial)
summary(logistic_model1)
```

```
##
## Call:
## glm(formula = Type ~ ., family = binomial, data = trainset)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.04430  -0.00076   0.06911   0.18183   1.98821
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2176.4605   738.9922   2.945 0.003228 **
## RI          -1152.5424   403.4869  -2.856 0.004284 **
## Na             -3.3758     1.0022  -3.368 0.000757 ***
## Mg              0.7661     0.4672   1.640 0.101060
## Al             -9.8661     2.6384  -3.739 0.000184 ***
## Si             -5.0429     1.6474  -3.061 0.002206 **
## K              -2.2528     1.9035  -1.184 0.236602
## Fe             12.1241     4.8915   2.479 0.013190 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 171.917  on 149  degrees of freedom
## Residual deviance:  41.464  on 142  degrees of freedom
## AIC: 57.464
##
## Number of Fisher Scoring iterations: 8
```

From the output above, we can see that few variables are not that significant to our model. So in next step we will remove them to improve our model.

2.ii) Improve the above model by clearly indicating reasons.

In above model summary, we can clearly see that the variable "Mg" and "K" have relatively high p-values, (i.e, greater than 0.05, considering 5 percent significance level), which indicates that these variables are not statistically significant in predicting the glass type. Thus, using hypothesis testing, we can consider removing these variables to improve our model.

Hence, the improved model:

```
logistic_model2 <- glm(Type ~ RI + Na + Al + Si + Fe, data = trainset, family = binomial )
summary(logistic_model2)
```

```
##
## Call:
## glm(formula = Type ~ RI + Na + Al + Si + Fe, family = binomial,
##     data = trainset)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.74198  -0.00096   0.10297   0.26115   1.90094
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  2090.1227   465.2353   4.493 7.04e-06 ***
## RI          -1130.3595   255.7360  -4.420 9.87e-06 ***
## Na             -2.8340     0.7129  -3.975 7.02e-05 ***
## Al            -10.9589     2.3027  -4.759 1.94e-06 ***
## Si             -4.3872     1.0431  -4.206 2.60e-05 ***
## Fe             12.1439     4.2386   2.865  0.00417 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 171.917  on 149  degrees of freedom
## Residual deviance:  52.728  on 144  degrees of freedom
## AIC: 64.728
##
## Number of Fisher Scoring iterations: 7
```

Now all the variables in this improved model are significant, so we'll proceed with this model for our further questions.

2.iii) Give the resulting model equation.

Resulting model equation is:

$$\hat{P}(\text{Type} = \text{Window}) = \frac{e^{2090.1227-1130.3595\times\text{RI}-2.8340\times\text{Na}-10.9589\times\text{Al}-4.3872\times\text{Si}+12.1439\times\text{Fe}}}{1+e^{2090.1227-1130.3595\times\text{RI}-2.8340\times\text{Na}-10.9589\times\text{Al}-4.3872\times\text{Si}+12.1439\times\text{Fe}}}$$

where, $\hat{P}$ is the estimated probability

2.iv) Predict the type of the glass for the test set (in question 1) using the improved model.

'logistic_model2' is our improved model, so we'll use this and predict the type of the glass for the testset data that we obtained in question 1.

```
predict_testset <- predict(logistic_model2, newdata = testset, type="response")
testset_predicted <- ifelse(predict_testset >= 0.5, "Window", "Non-window")
```

2.v) Construct the misclassification table (confusion matrix).

Now we've already made the prediction in above question, we will construct a confusion matrix to assess the accuracy of these predictions by distinguishing between correct and incorrect predictions.

```
actual_values <- testset$Type
confusion_matrix <- table(testset_predicted, actual_values)
confusion_matrix
```

```
##                  actual_values
## testset_predicted Non-window Window
##        Non-window         12      4
##        Window              0     48
```

From above output we can see that 4 of the Non-window type glasses are predicted falsey as window type. For the window type it seems all are correctly predicted. In the next step, we will calculate the overall misclassification error rate to have more understanding.

2.vi) Calculate the misclassification error rate.

```
Misclassification_Rate<-(4+0)/(12+4+48)
Misclassification_Rate
```

```
## [1] 0.0625
```

Here we can see that the misclassification rate is 6.25 percent for the prediction that we've done in part 2.iv for our test dataset.

2.vii) Give the probability that the observation belongs to "Window" category where RI=1.5, Na=13, Mg=3, Al=3, Si=72, K=0.5, Fe=0.

To find the probability that the observation belongs to "Window" category for this given observations, we first create a new data frame with the given details and use that data frame to predict the probability:

```r
new_observation <- data.frame(RI = 1.5, Na = 13, Mg = 3, Al = 3, Si = 72, K = 0.5,
                              Fe = 0)
probability <- predict(logistic_model2, newdata = new_observation, type = "response")
probability
```

```
##         1
## 0.9998751
```

Thus, the probability that the observation belongs to "Window" category where RI=1.5, Na=13, Mg=3, Al=3, Si=72, K=0.5, Fe=0 is 0.99987.

Alternatively, we can also use the equation that we found above in ques.iii, to calculate this probability:

$$\hat{P}(\text{Type} = \text{Window}) = \frac{e^{2090.1227-1130.3595\times1.5-2.8340\times13-10.9589\times3-4.3872\times72+12.1439\times0}}{1+e^{2090.1227-1130.3595\times1.5-2.8340\times13-10.9589\times3-4.3872\times72+12.1439\times0}}$$

Therefore,

$$\hat{P}(\text{Type} = \text{Window}) = 0.99987$$

Task 3: Classification tree Model

3.i) Fit a classification tree model for the training dataset (in question 1). Give the summary output and the tree diagram.

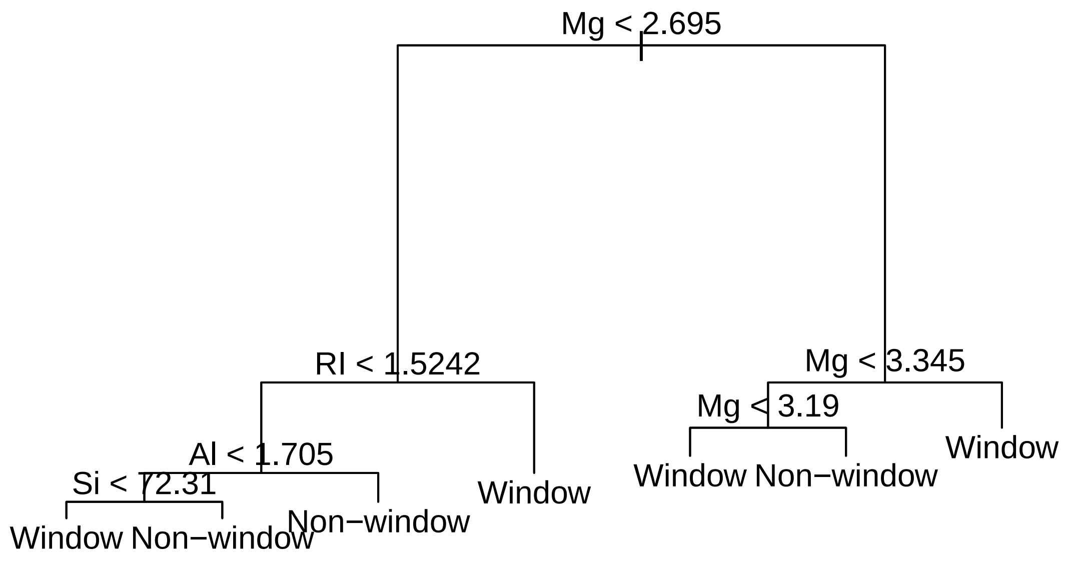
To fit a classification tree model for the training dataset that we obtained in question 1, we make use of tree() function under tree library:

```r
tree_glass_train <- tree(Type ~ ., trainset)
summary(tree_glass_train)
```

```
##
## Classification tree:
## tree(formula = Type ~ ., data = trainset)
## Variables actually used in tree construction:
## [1] "Mg" "RI" "Al" "Si"
## Number of terminal nodes:  7
## Residual mean deviance:  0.141 = 20.16 / 143
## Misclassification error rate: 0.03333 = 5 / 150
```

tree diagram:

```r
plot(tree_glass_train)
text(tree_glass_train, pretty=0)
```

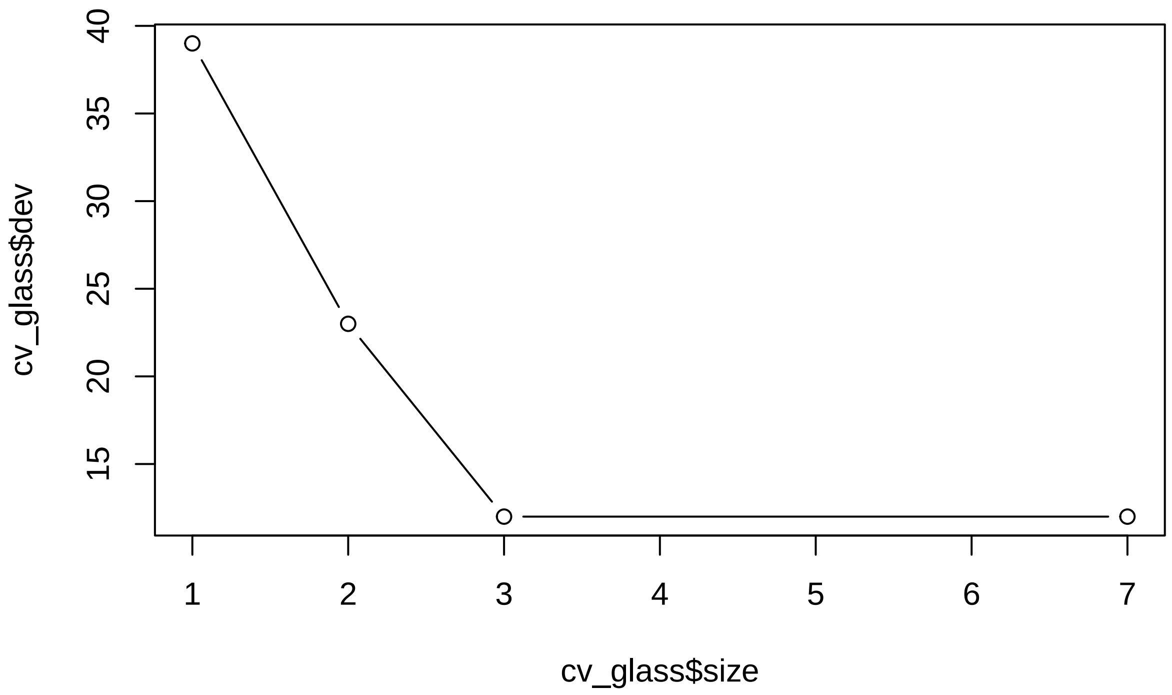


3.ii) Obtain the best classification tree by pruning.

In order to obtain the best classification tree by pruning,we first need to construct the cross validation plot to obtain best size of the tree model.

cross validation plot:

```
set.seed(1)
cv_glass<-cv.tree(tree_glass_train, FUN = prune.misclass)
plot(cv_glass$size, cv_glass$dev, type = "b")
```

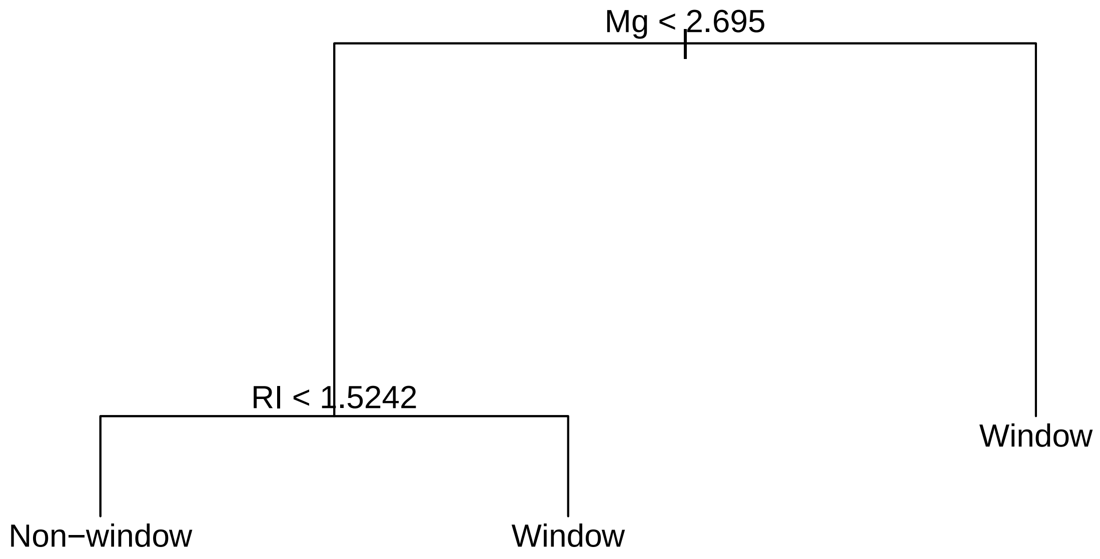


According to the plot above, we can see that the best size of the tree model is 3. So,

```
pruned_glass<-prune.misclass(tree_glass_train, best = 3)
pruned_glass
```

```
## node), split, n, deviance, yval, (yprob)
##       * denotes terminal node
##
## 1) root 150 171.90 Window ( 0.26000 0.74000 )
##   2) Mg < 2.695 47  51.15 Non-window ( 0.76596 0.23404 )
##     4) RI < 1.5242 40  26.01 Non-window ( 0.90000 0.10000 ) *
##     5) RI > 1.5242 7   0.00 Window ( 0.00000 1.00000 ) *
##   3) Mg > 2.695 103  27.13 Window ( 0.02913 0.97087 ) *
```

```
plot(pruned_glass)
text(pruned_glass, pretty=0)
```



So the plot above is the best classification tree obtained by pruning. And we'll describe this plot in next question.

3.iii) Describe terminal nodes of the best tree model (i.e., give the conditions of the two profiles "Window" and "Non-window").

a) If the conditions are met as follows:

- RI > 1.5245 (Refractive Index greater than 1.5245) AND
- Mg < 2.695 (Magnesium less than 2.695)

Then, the type of glass is classified as "Window."

b) Alternatively, if the conditions are met as follows:

- RI < 1.5245 (Refractive Index less than 1.5245) AND
- Mg < 2.695 (Magnesium less than 2.695)

Then, the type of glass is classified as "Non-Window."

c) In another scenario, if the condition is met as follows:

- Mg > 2.695 (Magnesium greater than 2.695)

Then, the type of glass is classified as "Window."

3.iv) Predict the type of the glass for the test set (in question 1) using the best tree model.

```
tree_predict<-predict(pruned_glass, testset, type="class")
tree_predict
```

```
##  [1] Window     Window     Window     Window     Window     Window    
##  [7] Window     Window     Window     Window     Window     Window    
## [13] Window     Window     Window     Window     Window     Window    
## [19] Window     Window     Window     Window     Window     Window    
## [25] Window     Window     Window     Window     Window     Window    
## [31] Window     Window     Window     Window     Non-window Window    
## [37] Window     Window     Window     Window     Non-window Window    
## [43] Window     Window     Window     Window     Window     Window    
## [49] Window     Window     Window     Window     Non-window Non-window
## [55] Non-window Non-window Non-window Non-window Non-window Non-window
## [61] Non-window Non-window Non-window Non-window
## Levels: Non-window Window
```

3.v) Construct the misclassification table (confusion matrix).

```
actual_values <- testset$Type # Actual values for the test set
confusion_matrix <- table(tree_predict, actual_values)
confusion_matrix
```

```
##             actual_values
## tree_predict Non-window Window
##   Non-window         12      2
##   Window              0     50
```

From above output we can see that 2 of the Non-window type glasses are predicted falsey as window type. For the window type it seems all are correctly predicted. In the next step, we will calculate the overall misclassification error rate to have more understanding.

3.vi) Calculate the misclassification error rate.

```
Misclassification_Rate<-(2+0)/(12+2+50)
Misclassification_Rate
```

```
## [1] 0.03125
```

```
#OR(generalised formula):
#misclassification_rate <- sum(confusion_matrix[1, 2] +
                                #confusion_matrix[2, 1])/sum(confusion_matrix)
```

Here we can see that the misclassification rate is 3.12 percent for the prediction that we've done in part 3.iv for our test dataset.

3.vii) Predict the type for an observation where RI=1.5, Na=13, Mg=3, Al=3, Si=72, K=0.5, Fe=0.

To predict the type for this given observations, we first create a new data frame with the given details and use that data frame to make the prediction:

```
new_observation <- data.frame(
  RI = 1.5,
  Na = 13,
  Mg = 3,
  Al = 3,
  Si = 72,
  K = 0.5,
  Fe = 0
)

prediction <- predict(pruned_glass, newdata = new_observation, type = "class") # Predict the type using
prediction
```

```
## [1] Window
## Levels: Non-window Window
```

So, the type for an observation where RI=1.5, Na=13, Mg=3, Al=3, Si=72, K=0.5, Fe=0 is “window”.

Task 4 : Comparision

Compare and contrast the Logistic regression model in question 2 and Classification tree model in question 3.

Metric	Logistic Regression (Question 2)	Classification Tree (Question 3)	Comment
Model Complexity	Complex	Simple	Logistic Regression is more complex as it estimates continuous probabilities, while the Classification Tree offers a simpler, more intuitive model.
Clarity	Moderate	High	The Classification Tree provides clear decision rules, making it highly interpretable, while Logistic Regression offers coefficients but may lack intuitive understanding.
Model Improvement	Hypothesis Testing and Variable Selection	Pruning (Reducing Tree Complexity)	Logistic Regression improved by selecting significant variables through hypothesis testing, while the Classification Tree improved via pruning to reduce overfitting.
Performance	6.25 percent	3.12 percent	The Classification Tree outperforms Logistic Regression on the test set, with a lower misclassification rate, indicating better predictive accuracy.

Metric	Logistic Regression (Question 2)	Classification Tree (Question 3)	Comment
Prediction Method (2.VII & 3.VII)	Probability Estimates (0.9998751)	Direct Classification (Window)	Logistic Regression predicts probabilities of class membership numerically (0.9998751 for "Window"), while the Classification Tree directly classifies into categories, providing a word label ("Window").

In summary, both models have their strengths and are chosen based on the trade-off between complexity and interpretability. Logistic Regression estimates probabilities numerically, while the Classification Tree offers a simpler model with better performance in terms of predictive accuracy and provides word labels for classification. Additionally, Logistic Regression improved by selecting significant variables, while the Classification Tree improved through pruning to enhance its generalization capability. The preference between these two models depends on the specific goals of the prediction task and the need for probability estimates versus clear decision rules.

Part II – Unsupervised Learning

Description of Data set

This section uses the dataset "PollutionData.csv". The data consists of the annual averages of 8 air pollution concentrations monitored at 16 different monitoring sites across NSW in the year 2022 (This data was obtained from the Department of Planning and Environment website). The variables in the data include:

- **CO** - Annual averages of Carbon monoxide (in ppm)
- **PM10** - Annual averages of Particulate Matter ($< 10\mu m$) (in $\mu g/m^3$)
- **NO** - Annual averages of Nitric Oxide (in pphm)
- **PM2.5** - Annual averages of Particulate Matter ($< 2.5\mu m$) (in $\mu g/m^3$)
- **NO2** - Annual averages of Nitrogen Dioxide (in pphm)
- **SO2** - Annual averages of Sulphur Dioxide (in pphm)
- **OZONE** - Annual averages of ozone (in pphm)
- **NEPH** - Visibility reduction ($\times 10^{-4} m^{-1}$)

Data Preprocessing

Use the following set of R codes to load and prepare the data.

```r
PollutionData = read.csv("C:/Users/rosmi/OneDrive/Desktop/Sem 2/Intro to DS/Pollution.csv")
row.names(PollutionData) = PollutionData$SiteName
PollutionData = PollutionData[,-1]
head(PollutionData)
```

```
##                     CO  NO NO2 OZONE PM10 PM2.5 SO2 NEPH
## ROZELLE            0.2 0.4 0.6   1.7 12.9   4.6 0.0 0.16
## LIVERPOOL         -0.1 1.1 0.8   1.4 14.6   5.5 0.0 0.18
## WYONG             -0.1 0.2 0.2   1.6 11.7   4.0 0.1 0.12
## NEWCASTLE          0.2 0.4 0.4   1.7 17.5   5.5 0.1 0.18
## WOLLONGONG         0.2 0.3 0.4   1.8 14.3   4.7 0.1 0.14
## PARRAMATTA_NORTH  0.2 0.9 0.7   1.5 14.1   5.2 0.0 0.16
```

Task 5 : Principal Component Analysis (PCA)

5.i) Examine the mean and variance of the air pollutants. Comment on your results.

To calculate the mean of the air pollutants we can use sapply() function:

```
sapply(PollutionData, mean)
```

```
##       CO       NO      NO2    OZONE     PM10    PM2.5      SO2     NEPH
##  0.07500  0.46875  0.53125  1.61250 12.38750  4.53750  0.04375  0.15125
```

The mean values of the air pollutants shows variations in their concentrations. Notably:

- Carbon Monoxide (CO) has a relatively low mean concentration of 0.075 ppm.
- Nitric Oxide (NO) exhibits a moderate mean concentration of 0.46875 pphm.
- Nitrogen Dioxide (NO2) also shows a moderate mean concentration at 0.53125 pphm.
- Ozone has a relatively higher mean concentration of 1.61250 pphm.
- Particulate Matter (PM10) averages at 12.38750 μg/m^3, while PM2.5 averages at 4.53750 μg/m^3.
- Sulphur Dioxide (SO2) maintains a low mean concentration of 0.04375 pphm.
- Visibility reduction (NEPH) is measured at 0.15125 $\times 10^{-4}$ m^{-1}.

These mean values provide insights into the relative levels of these pollutants in the air, with some pollutants showing higher concentrations than others.

To calculate the mean of the air pollutants we can use sapply() function:

```
sapply(PollutionData, var)
```

```
##          CO          NO         NO2       OZONE        PM10       PM2.5
## 0.015333333 0.111625000 0.076958333 0.063833333 5.550500000 0.739833333
##         SO2        NEPH
## 0.003958333 0.000505000
```

It can be seen that the given air pollutants have different variances. Variance of Particulate Matter (PM10) is comparatively larger than other air pollutants.These variance values provide insights into the degree of variability and fluctuations in pollutant concentrations. High variances may indicate greater uncertainty and potential challenges in predicting air quality, while low variances suggest more stable and predictable pollutant levels.

5.ii) Is it necessary to scale the variables? Justify your answer.

It's always a good practice to scale the variables before performing the Principal Component Analysis. Saying that, it also depends on the characteristics of our dataset and our specific objectives. However, in our given dataset, the considerable variation in the variances of our variables can be seen, so the scaling is necessary in our case. Scaling ensures that all variables in our dataset contributes more equally to the principal components and that the results focus on the underlying data structure rather than the magnitude of individual variables.

5.III) Perform principal component analysis for the given dataset.

To perform the principal component analysis for our given dataset we can make use of prcomp() function:

```r
pca = prcomp(PollutionData, scale. = TRUE)
pca
```

```
## Standard deviations (1, .., p=8):
## [1] 2.0301762 1.3731539 0.9708602 0.6772438 0.5573237 0.4281196 0.2592025
## [8] 0.1747053
##
## Rotation (n x k) = (8 x 8):
##              PC1        PC2         PC3          PC4          PC5          PC6
## CO    -0.1023467 -0.4825564 -0.70502052 -0.216502159  0.174720942 -0.240880127
## NO    -0.4015422  0.2614249  0.30733340 -0.314319978 -0.298276995 -0.389609203
## NO2   -0.3898915  0.2593672 -0.06317732 -0.640636504  0.375109675  0.057018211
## OZONE  0.3377903 -0.3658119  0.10022469 -0.572466391 -0.601164377  0.147120525
## PM10  -0.3219383 -0.4855028  0.23183861  0.160968984 -0.027695022 -0.480401282
## PM2.5 -0.4335912 -0.2732344  0.16774729  0.221045103 -0.155193133  0.113021162
## SO2    0.2841810 -0.3566177  0.55749778 -0.203524276  0.594574443  0.001529934
## NEPH  -0.4356224 -0.2452762  0.04199788 -0.001668561 -0.005182574  0.722304917
##               PC7        PC8
## CO     0.32369505 -0.1393132
## NO     0.42152139 -0.3984034
## NO2   -0.30972695  0.3555526
## OZONE -0.10935166  0.1389811
## PM10  -0.58667106 -0.0724380
## PM2.5  0.42348118  0.6664432
## SO2    0.28373537 -0.0762378
## NEPH  -0.07501775 -0.4700289
```

By default, the prcomp() function centers the variables to have mean zero. By using the option scale=TRUE, we scale the variables to have standard deviation one.

We can see the components associated with a Principal Component Analysis(PCA) by using names() function:

```r
names(pca)
```

```
## [1] "sdev"     "rotation" "center"   "scale"    "x"
```

Here,

1. **"sdev"**: gives the standard deviations of the principal components.
2. **"rotation"**: gives the loadings of the original variables on the principal components.
3. **"center"**: gives the mean values of the original variables.
4. **"scale"**: gives the standard deviations of the original variables.
5. **"x"**: represents the data after PCA transformation.

For instance, let's use pca$center and see what outputs do we get:

```
pca$center
```

```
##       CO       NO      NO2    OZONE     PM10    PM2.5      SO2     NEPH
##  0.07500  0.46875  0.53125  1.61250 12.38750  4.53750  0.04375  0.15125
```

Here we can see it's the mean values of the original air pollutants that we just calculated above.

We can also calculate the standard deviations of the original variables:

```
pca$scale
```

```
##         CO         NO        NO2      OZONE       PM10      PM2.5        SO2
## 0.12382784 0.33410328 0.27741365 0.25265259 2.35594991 0.86013565 0.06291529
##       NEPH
## 0.02247221
```

So the output above is the standard deviations of our original air pollutants.

5.IV) Write the first two Principal Components (PCs) in terms of Original Variables. (Note: provide the necessary R output)

Let's first get the principal component loading vectors.

```
pca$rotation
```

```
##              PC1        PC2         PC3          PC4          PC5          PC6
## CO    -0.1023467 -0.4825564 -0.70502052 -0.216502159  0.174720942 -0.240880127
## NO    -0.4015422  0.2614249  0.30733340 -0.314319978 -0.298276995 -0.389609203
## NO2   -0.3898915  0.2593672 -0.06317732 -0.640636504  0.375109675  0.057018211
## OZONE  0.3377903 -0.3658119  0.10022469 -0.572466391 -0.601164377  0.147120525
## PM10  -0.3219383 -0.4855028  0.23183861  0.160968984 -0.027695022 -0.480401282
## PM2.5 -0.4335912 -0.2732344  0.16774729  0.221045103 -0.155193133  0.113021162
## SO2    0.2841810 -0.3566177  0.55749778 -0.203524276  0.594574443  0.001529934
## NEPH  -0.4356224 -0.2452762  0.04199788 -0.001668561 -0.005182574  0.722304917
##               PC7        PC8
## CO     0.32369505 -0.1393132
## NO     0.42152139 -0.3984034
## NO2   -0.30972695  0.3555526
## OZONE -0.10935166  0.1389811
## PM10  -0.58667106 -0.0724380
## PM2.5  0.42348118  0.6664432
## SO2    0.28373537 -0.0762378
## NEPH  -0.07501775 -0.4700289
```

From above output, it can be seen that there are eight principal components. PC1 seems primarily influenced by OZONE and SO2. While, PC2 is mainly driven by NO and NO2, which contribute nearly equally to this component.

Now with this information, we can write the equatiion of the first two Principal Components (PCs) in terms of Original Variables:

$$\begin{aligned} \text{PC1} = &-0.1023467 \times \text{CO} - 0.4015422 \times \text{NO} - 0.3898915 \times \text{NO2} \\ &+ 0.3377903 \times \text{OZONE} - 0.3219383 \times \text{PM10} - 0.4335912 \times \text{PM2.5} \\ &+ 0.2841810 \times \text{SO2} - 0.4356224 \times \text{NEPH} \end{aligned}$$

$$\begin{aligned} \text{PC2} = &-0.4825564 \times \text{CO} + 0.2614249 \times \text{NO} + 0.2593672 \times \text{NO2} \\ &- 0.3658119 \times \text{OZONE} - 0.4855028 \times \text{PM10} - 0.2732344 \times \text{PM2.5} \\ &- 0.3566177 \times \text{SO2} - 0.2452762 \times \text{NEPH} \end{aligned}$$

These equations represents how PC1 & PC2 are calculated as a linear combination of the original variables (CO, NO, NO2, OZONE, PM10, PM2.5, SO2, and NEPH) with their respective coefficients.

5.V) Explain the proportion of variance explained by each PCs.

To explain the proportion of variance explained by each PCs we use the help of summary() function:

```r
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6    PC7
## Standard deviation     2.0302 1.3732 0.9709 0.67724 0.55732 0.42812 0.2592
## Proportion of Variance 0.5152 0.2357 0.1178 0.05733 0.03883 0.02291 0.0084
## Cumulative Proportion  0.5152 0.7509 0.8687 0.92605 0.96488 0.98779 0.9962
##                            PC8
## Standard deviation     0.17471
## Proportion of Variance 0.00382
## Cumulative Proportion  1.00000
```

The "Proportion of Variance" column in the above table represents the proportion of the total variance in the dataset that is explained by each Principal Component (PC).

1. **PC1:** PC1 explains approximately 51.52 percent of the total variance in the dataset. It captures the largest share of the variation in the data and is the most significant PC.
2. **PC2:** PC2 explains approximately 23.57 percent of the total variance. While it explains a substantial portion of the variance, it is less influential than PC1.
3. **PC3:** PC3 explains about 11.78 percent of the total variance. It contributes to capturing the variation but to a lesser extent than PC1 and PC2.
4. **PC4:** PC4 explains around 5.73 percent of the total variance. It represents a smaller portion of the data's variability compared to the earlier PCs.

5. **PC5:** PC5 explains roughly 3.88 percent of the total variance. It captures even less of the variation in the data.
6. **PC6:** PC6 explains about 2.29 percent of the total variance. Its contribution to explaining variance is relatively minor.
7. **PC7:** PC7 explains approximately 0.84 percent of the total variance. It represents a very small portion of the variation.
8. **PC8:** PC8 explains about 0.38 percent of the total variance. It captures the least amount of variation among all the PCs.

We can also see the cumulative proportion in our table above. Cumulative proportion provides insights into how much of the total variation in the dataset is captured when considering a specific number of PCs in sequence.

The first and second PCs together explain about 75.09 percent of the total variation in the dataset. The first, second, and third PCs together explain about 86.87 percent of the total variation, and so forth.

When all eight PCs (PC1 to PC8) are considered together, they collectively explain 100 percent of the total variation in the data.

We can also draw a graph to visualize the proportion of variances and cumulative proportion explained by each principal components.For to draw the graph, we first need to do following calculations:

- let's find the standard deviation of each principal component:

```
pca$sdev
```

```
## [1] 2.0301762 1.3731539 0.9708602 0.6772438 0.5573237 0.4281196 0.2592025
## [8] 0.1747053
```

- let's find the variance of each principal component:

```
pca_var = pca$sdev^2
pca_var
```

```
## [1] 4.12161555 1.88555166 0.94256962 0.45865921 0.31060968 0.18328642 0.06718592
## [8] 0.03052194
```

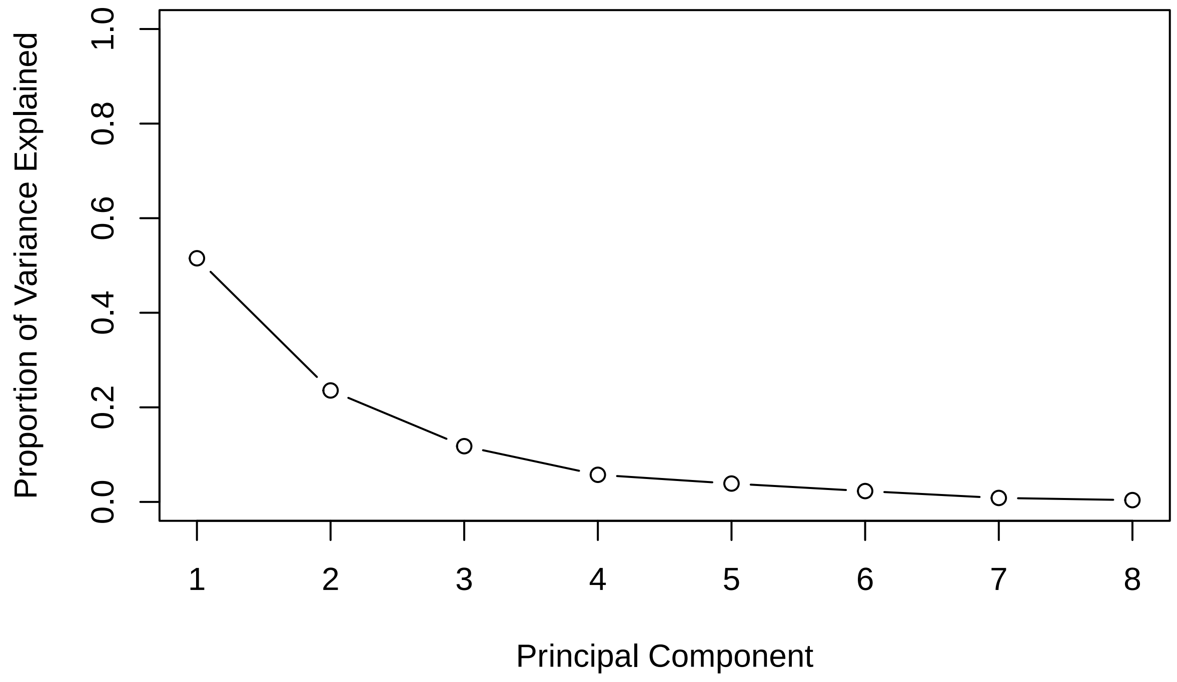
- let's find the proportion of variance explained by each principal component:

```
pve = pca_var/sum(pca_var)
pve
```

```
## [1] 0.515201944 0.235693958 0.117821202 0.057332401 0.038826210 0.022910803
## [7] 0.008398240 0.003815242
```

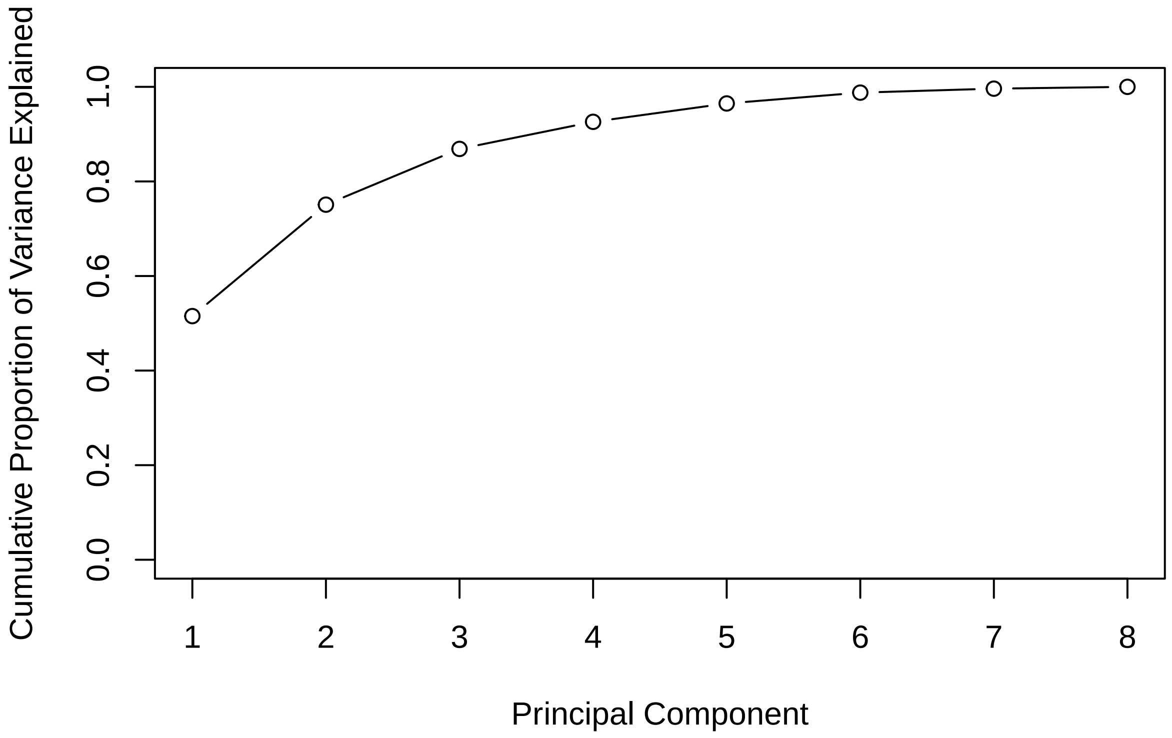
- Now, we can finally use our above outputs for proportion of variance explained by each principal component,stored in pve variable, to plot the graph of proportion of variance explained by each principal component:

```r
plot(pve,xlab="Principal Component",
     ylab="Proportion of Variance Explained",
ylim = c(0,1), type = 'b')
```



Let's visualize the cumulative proportion of variance explained by each principal component as well:

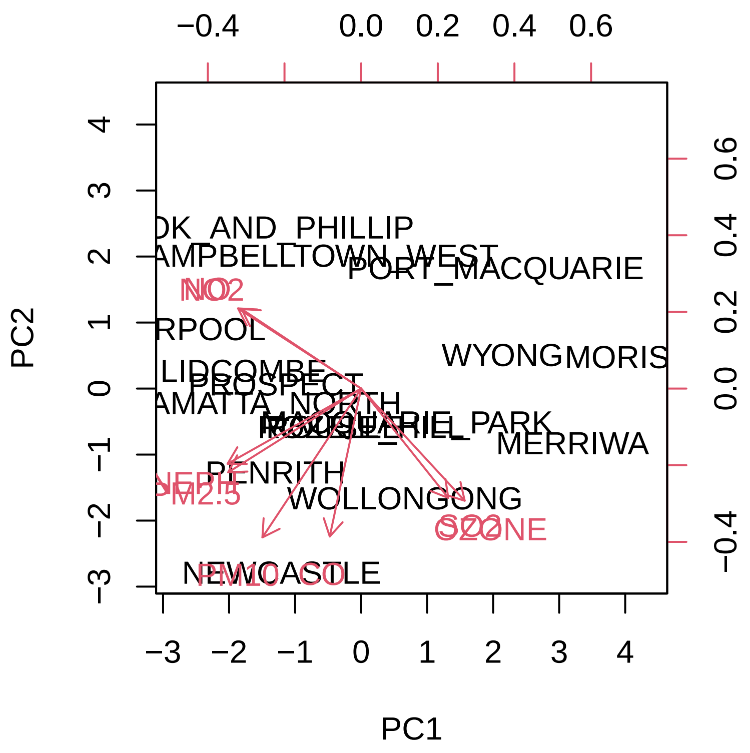
```r
plot(cumsum(pve),xlab="Principal Component",
     ylab="Cumulative Proportion of Variance Explained",
ylim = c(0,1), type = 'b')
```



5.VI) Draw the biplot.

Since we already have calculated principal component analysis for our given dataset, we can draw the biplot easily by the help of biplot() function:

```r
biplot(pca, scale = 0)
```



The `scale=0` *argument to* `biplot()` *ensures that the arrows are scaled to represent the loadings; other values for scale give slightly different biplots with different interpretations.*

5.VII) Use the biplot to answer the following questions:

a. Comment on the association between SO2 and OZONE: It can be seen in the above biplot that SO2 and OZONE are pointing in a similar direction and also are located so close to each other, which indicates a strong positive co-relation between them. In other words, we can say when SO2 levels increase, OZONE levels tend to increase as well and vice versa. Additionally, it seems OZONE contributes little bit more for the variation in PC1 than SO2.

b. Comment on the association between OZONE and NO2: In the biplot above, a clear negative correlation between NO2 and OZONE can be seen. This is apparent from the fact that these two variables have vectors pointing in opposite directions and are positioned at a considerable distance from each other. In simpler terms, when OZONE levels rise, NO2 levels tend to decrease, and conversely, when OZONE levels decrease, NO2 levels tend to increase.

c. Comment on the association between visibility reduction and OZONE: Based on the biplot, there seems to be an inverse relationship between OZONE and visibility reduction, where changes in OZONE levels are associated with changes in visibility, potentially indicating that higher OZONE levels are linked to improved visibility.

d. Consider Penrith and Port Macquarie, which site recorded the highest annual PM2.5 in 2022? The first loading vector (PC1) places more weight on PM2.5 and the Penrith has large positive score on the PC1 in compare to Port Macquarie, thus penrith has recorded the highest annual PM2.5 in 2022.

Task 6 : K-means Clustering & Hierarchical Clustering

K-means Clustering

6.I) Perform k-means clustering to group the sites into 2 clusters

To perform k-means clustering to group the given sites into two clusters, we first set the seed and then we use the kmeans() function. Here we use centers=2, since we need two clusters and nstart=20 means we are instructing the algorithm to run 20 separate times, each with a different random initialization of cluster centroids.

```
set.seed(5)
km2 = kmeans(PollutionData, centers = 2, nstart = 20)
km2$cluster #gives cluster assignments
```

```
##          ROZELLE         LIVERPOOL             WYONG         NEWCASTLE
##                1                 1                 2                 1
##       WOLLONGONG  PARRAMATTA_NORTH          PROSPECT           MERRIWA
##                1                 1                 1                 2
## CAMPBELLTOWN_WEST    MACQUARIE_PARK    PORT_MACQUARIE        ROUSE_HILL
##                2                 2                 2                 2
##  COOK_AND_PHILLIP          LIDCOMBE           PENRITH          MORISSET
##                2                 1                 1                 2
```

```
km2
```

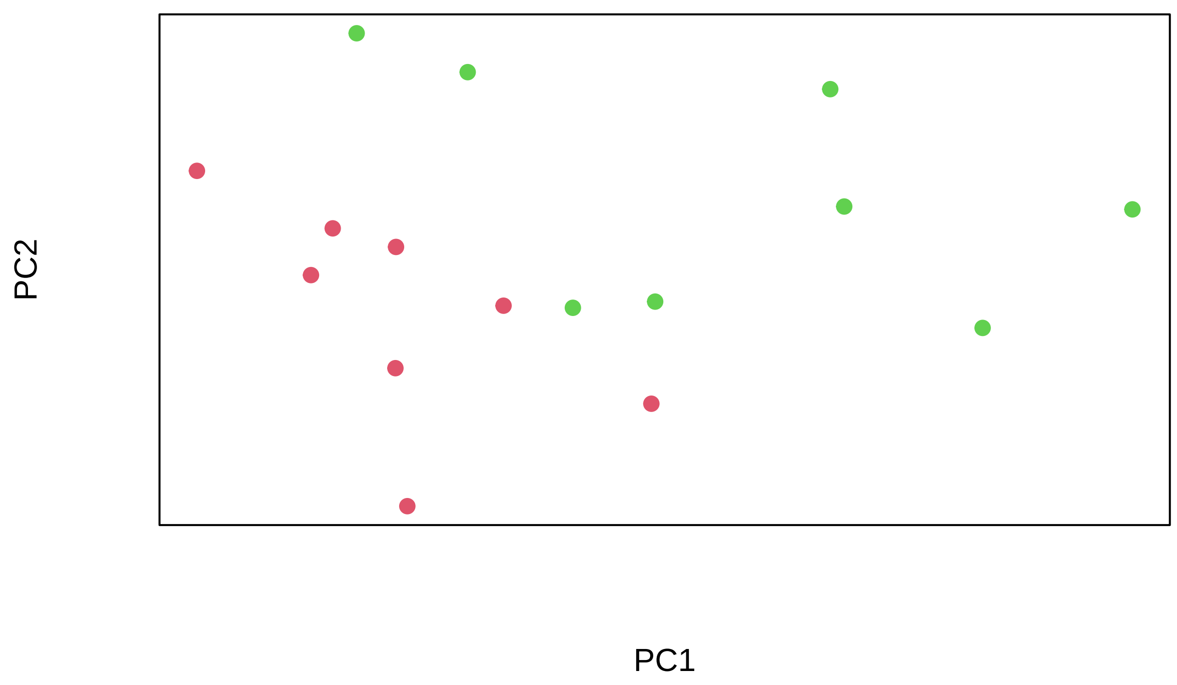
```
## K-means clustering with 2 clusters of sizes 8, 8
##
## Cluster means:
##      CO     NO    NO2 OZONE    PM10  PM2.5    SO2    NEPH
## 1 0.125 0.6000 0.6250 1.575 14.2375 5.1625 0.0375 0.16625
## 2 0.025 0.3375 0.4375 1.650 10.5375 3.9125 0.0500 0.13625
##
## Clustering vector:
##          ROZELLE         LIVERPOOL             WYONG         NEWCASTLE
##                1                 1                 2                 1
##       WOLLONGONG  PARRAMATTA_NORTH          PROSPECT           MERRIWA
##                1                 1                 1                 2
## CAMPBELLTOWN_WEST    MACQUARIE_PARK    PORT_MACQUARIE        ROUSE_HILL
##                2                 2                 2                 2
##  COOK_AND_PHILLIP          LIDCOMBE           PENRITH          MORISSET
##                2                 1                 1                 2
##
## Within cluster sum of squares by cluster:
## [1] 16.84264 20.10259
##  (between_SS / total_SS =  62.5 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

In the above output we can see the cluster means, within cluster sum of squares and other many details.

6.II) Visualise the clusters you obtained in part I using the first 2 PCs.

We can plot to visualize the clusters we obtained in part I using the first 2 PCs.

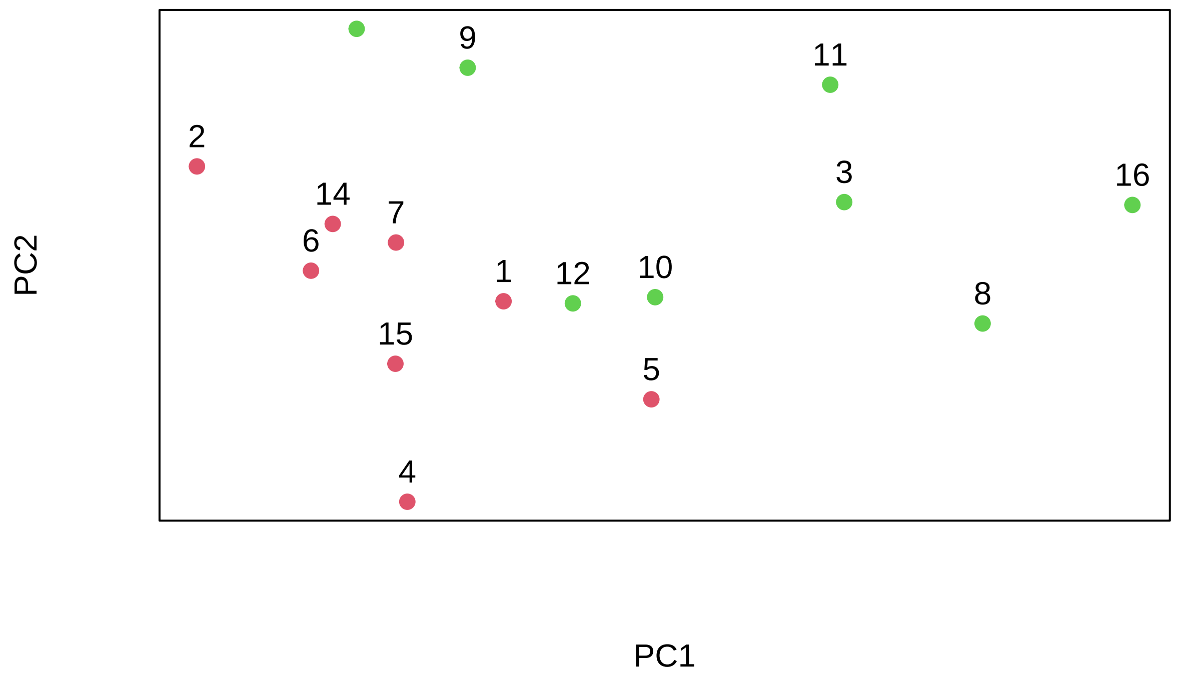
```r
pp = prcomp(PollutionData, scale = TRUE)
plot(pp$x[,1:2], col=fitted(km2, "classes")+1, pch=19, xaxt="n", yaxt="n")
```



In above plot we can see our sites grouped into two clusters.

OR, if we want to know which site get grouped into which clusters just by looking the plot we can do this as well:

```r
pp = prcomp(PollutionData, scale = TRUE)
plot(pp$x[,1:2], col=fitted(km2, "classes")+1,
     pch=19, xaxt="n", yaxt="n")
text(pp$x[,1], pp$x[,2], labels=1:nrow(PollutionData), pos=3)
```



In plot above red dots belong to cluster1 and green dots as cluster2.

6.III) Clearly list the sites in the two clusters that you obtained in part I.

```r
clustered_sites <- data.frame(Site = rownames(PollutionData),
                              Cluster = km2$cluster)
cluster1_sites <- character(0)
cluster2_sites <- character(0)

max_length <- max(length(cluster1_sites), length(cluster2_sites))

if (length(cluster1_sites) < max_length) {
  cluster1_sites <- c(cluster1_sites, rep("", max_length -
                                            length(cluster1_sites)))
}
if (length(cluster2_sites) < max_length) {
  cluster2_sites <- c(cluster2_sites, rep("", max_length -
                                            length(cluster2_sites)))
}

table_data <- data.frame(
  `Cluster 1 Sites` = paste(clustered_sites$Site[clustered_sites$Cluster == 1],
                            collapse = ", "),
  `Cluster 2 Sites` = paste(clustered_sites$Site[clustered_sites$Cluster == 2],
                            collapse = ", ")
)

kable(table_data, format = "markdown")
```

Cluster.1.Sites	Cluster.2.Sites
ROZELLE, LIVERPOOL, NEWCASTLE, WOLLONGONG, PARRAMATTA_NORTH, PROSPECT, LIDCOMBE, PENRITH	WYONG, MERRIWA, CAMPBELLTOWN_WEST, MACQUARIE_PARK, PORT_MACQUARIE, ROUSE_HILL, COOK_AND_PHILLIP, MORISSET

From the output above, we can see our sites clearly listed into their respective clusters.

Hierarchical Clustering

6.IV) Perform hierarchical clustering to group the sites into 2 clusters. Consider the distance between two points as the maximum distance when calculating the dissimilarity measure.

To perform hierarchical clustering to group the sites into 2 clusters we make use of the hclust() function. Before using, hclust() we need the dissimilarity matrix. As our question has asked us to calculate the distance between two points as the maximum distance when calculating the dissimilarity measure, we do it that way:

dissimilarity_matrix:

```r
dissimilarity_matrix = dist(PollutionData, method = "maximum")
```

Now we've our dissimilarity matrix, we can proceed to do the hierarchical clustering.

Since our question specifies the use of the maximum distance criterion, we exclusively use complete linkage in our hierarchical clustering. This is because complete linkage calculates dissimilarity between two clusters by considering the maximum distance, which involves the farthest pair of points within those clusters. By doing so, we adhere to the requirement for maximum distance-based clustering, rendering complete linkage the appropriate choice for this hierarchical clustering.

```r
hc.complete = hclust(dissimilarity_matrix, method = "complete")
hc.complete
```

```
##
## Call:
## hclust(d = dissimilarity_matrix, method = "complete")
##
## Cluster method   : complete
## Distance         : maximum
## Number of objects: 16
```

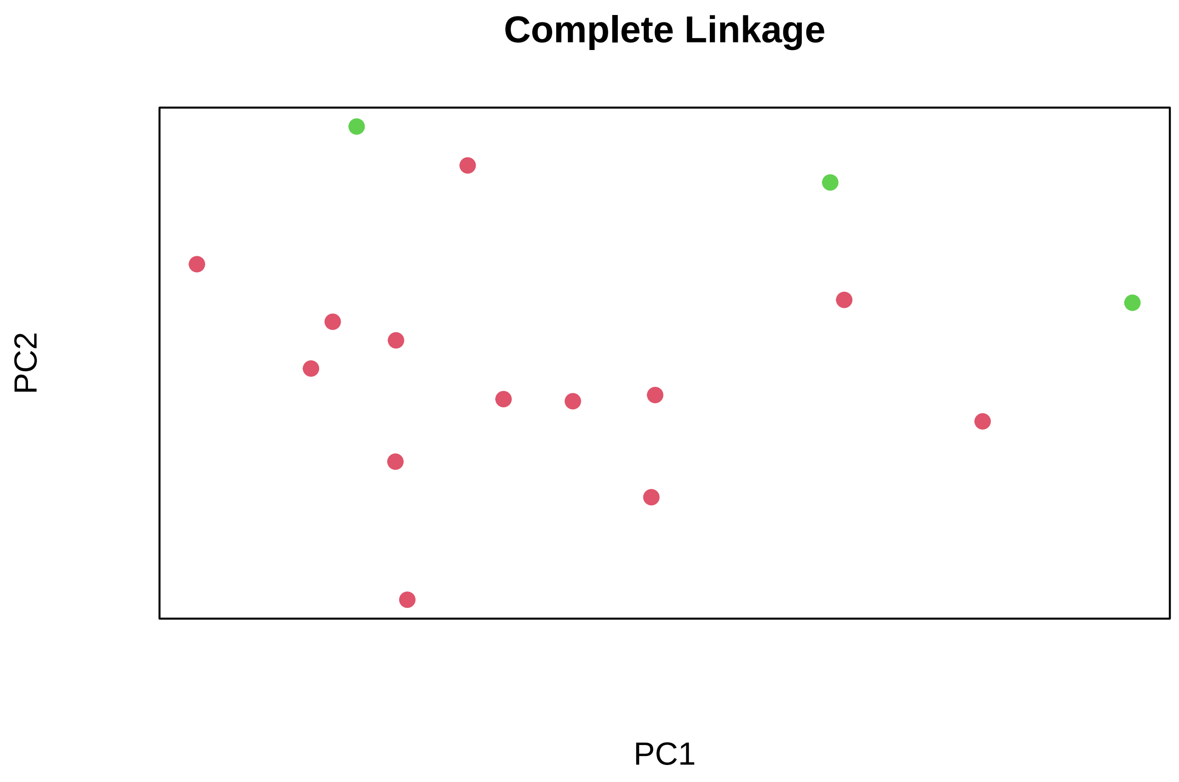
```r
clusters = cutree(hc.complete, k = 2)
clusters
```

```
##          ROZELLE        LIVERPOOL            WYONG        NEWCASTLE
##                1                1                1                1
##       WOLLONGONG  PARRAMATTA_NORTH         PROSPECT          MERRIWA
##                1                1                1                1
## CAMPBELLTOWN_WEST    MACQUARIE_PARK    PORT_MACQUARIE        ROUSE_HILL
##                 1                 1                 2                 1
##   COOK_AND_PHILLIP          LIDCOMBE           PENRITH          MORISSET
##                 2                 1                 1                 2
```

6.V) Visualise the clusters you obtained in part IV using the first 2 PCs.

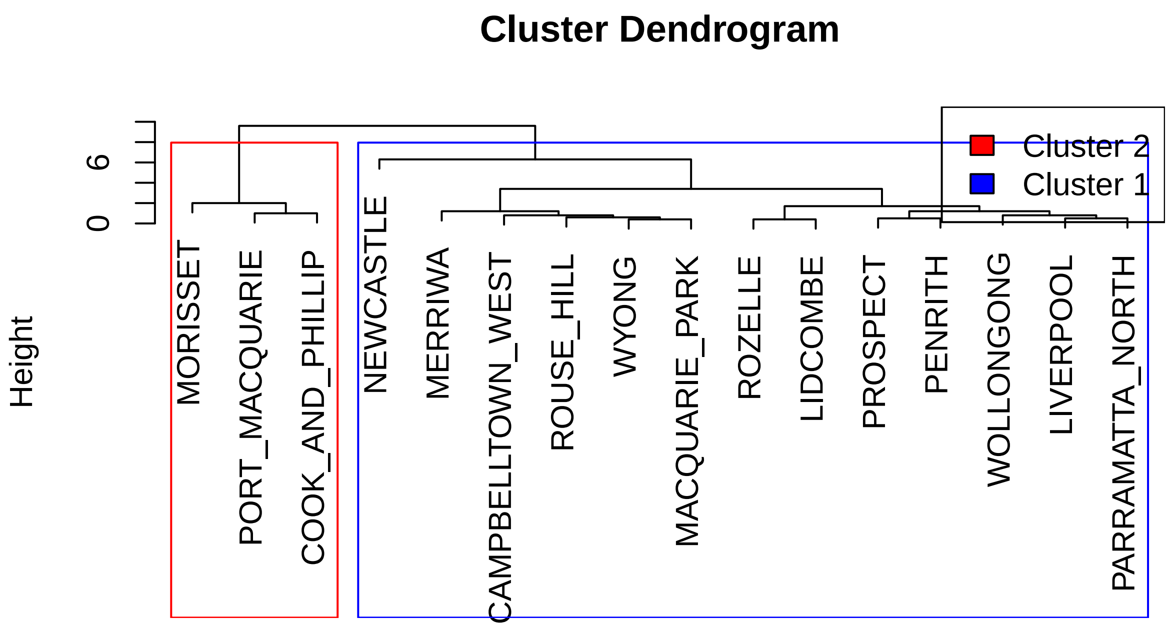
We use plot to visualize the clusters we obtained in part IV using the first 2 PCs.

```r
plot(pp$x[,1:2], col=cutree(hc.complete, k=2)+1,main="Complete Linkage",
     pch=19, xaxt="n", yaxt="n")
```



6.VI) Draw a dendrogram for the clustering method in part IV and highlight the two clusters.

```r
plot(hc.complete, xlab=" ", sub="Complete linkage cluster analysis")
cluster_colors <- c("red", "blue")
rect.hclust(hc.complete, k=2, border = cluster_colors)
legend("topright", legend = c("Cluster 2", "Cluster 1"), fill = cluster_colors)
```



Complete linkage cluster analysis

From the above complete linkage cluster dendrogram, we can see 3 sites belong in cluster2 and rest on cluster1.

6.VII) Clearly list the sites in the two clusters that you obtained in part IV.

To list the sites in the two clusters that we obtained in part IV we'll create a table so it'll be easier to visualie.

```r
clustered_sites <- data.frame(Site = rownames(PollutionData), Cluster = clusters)

cluster1_sites <- character(0)
cluster2_sites <- character(0)

max_length <- max(length(cluster1_sites), length(cluster2_sites))

if (length(cluster1_sites) < max_length) {
  cluster1_sites <- c(cluster1_sites, rep("", max_length - length(cluster1_sites)))
}
if (length(cluster2_sites) < max_length) {
  cluster2_sites <- c(cluster2_sites, rep("", max_length - length(cluster2_sites)))
}

table_data <- data.frame(
  `Cluster 1 Sites` = paste(clustered_sites$Site[clustered_sites$Cluster == 1],
                           collapse = ", "),
  `Cluster 2 Sites` = paste(clustered_sites$Site[clustered_sites$Cluster == 2],
                            collapse = ", ")
)

kable(table_data, format = "markdown")
```

Cluster.1.Sites	Cluster.2.Sites
ROZELLE, LIVERPOOL, WYONG, NEWCASTLE, WOLLONGONG, PARRAMATTA_NORTH, PROSPECT, MERRIWA, CAMPBELLTOWN_WEST, MACQUARIE_PARK, ROUSE_HILL, LIDCOMBE, PENRITH	PORT_MACQUARIE, COOK_AND_PHILLIP, MORISSET

Comparision

6.VIII) Compare and comment on your results in part III and VII.

Consistency in Cluster 1: One notable observation is the high degree of consistency in Cluster 1 across both clustering methods. Most of the sites that were grouped in Cluster 1 in Part III, including ROZELLE, LIVERPOOL, NEWCASTLE, and others, were also found in Cluster 1 of Part VII. This alignment suggests that there are strong and consistent patterns within this group of sites that both k-means and hierarchical clustering methods successfully capture.

Differences in Cluster 2: While Cluster 1 shows remarkable consistency, there are slight differences in Cluster 2 between the two methods. In Part III, Cluster 2 included sites like WYONG, MERRIWA, CAMPBELLTOWN_WEST, ROUSE_HILL, MACQUARIE_PARK, among others. However, in Part VII, these same sites were grouped together in Cluster 1 instead of Cluster 2, as in Part III.

Interpretation: The strong agreement in Cluster 1 across both methods reinforces our confidence in the validity of the patterns identified within this cluster. It indicates that these patterns are robust and not highly sensitive to the choice of clustering technique.

On the other hand, the differences observed in Cluster 2 emphasize the potential sensitivity of clustering methods to the specific data characteristics and linkage criteria used. These variations may provide valuable insights into borderline or ambiguous cases within the data.

In summary, while Cluster 1 remains consistent, the slight differences in Cluster 2 indicate sensitivity to method and data characteristics. Both clustering methods contribute to our understanding of the data, emphasizing stable patterns while also revealing nuanced variations.

Summary of Findings:

Principal Component Analysis (PCA): PCA was performed on the dataset of annual air pollutant averages from 16 sites. The first two principal components explained about 75.09% of the total variance, with PC1 primarily influenced by OZONE and SO2, and PC2 driven by NO and NO2.

Biplot Analysis: SO2 and OZONE showed a strong positive correlation, while OZONE and NO2 had a clear negative correlation. Penrith recorded the highest annual PM2.5.

K-means Clustering: Sites were grouped into two clusters using K-means clustering. Visualization with the first two principal components revealed distinct clustering patterns.

Hierarchical Clustering: Hierarchical clustering with complete linkage was used to group sites into two clusters. The dendrogram highlighted the cluster assignments.

Comparison of Clustering Methods: Both K-means and hierarchical clustering showed consistency in Cluster 1, but differences in Cluster 2 highlighted the sensitivity of clustering methods to data characteristics.