Assignment Multivariate

2024-03-23

1.

Load the data set Temperature_Data.csv Use the set.seed function in R to set the seed to your student number Select a random subset of 1000 observations from the dataset.

```
data = read.csv("Temperature_Data.csv")
set.seed(19334466)

# Sample without replacement, random rows, all columns
samp_data = data[sample(1:nrow(data), 1000), ]
```

ignore this

https://physionet.org/content/face-oral-temp-data/1.0.0/_Table1.pdf

https://physionet.org/content/face-oral-temp-data/1.0.0/_Table2.pdf

(1st,2nd,3rd,4th round of images) Data we reviewing only contains 1st round images

Max1R13_1: A circle with diameter of 13 pixels from the right canthus point to the face centerline Max temp

T_RC1: A square of 24x24 pixels around the right canthus, with 2/3 toward the face center (dry area, 16x24 pixels) and 1/3 away from the face center (wet area, 8x24 pixels). Avg temp

T_LC1: A square of 24x24 pixels around the left canthus, with 2/3 toward the face center (dry area, 16x24 pixels) and 1/3 away from the face center (wet area, 8x24 pixels) Avg temp

RCC1: A square of 3x3 pixels centered at the right canthus point Avg temp

canthiMax1: Extended canthi area, see definition in our papers [1,2] Max temp

*T_FHCC1: Center point of forehead, a square of 3x3 pixels. Avg temp

T_FHLC1: Left point of the forehead, , a square of 3x3 pixels. Avg temp

T_FHTC1: Top point of the forehead, , a square of 3x3 pixels. Avg temp

T_OR1: Oral/mouth Region, see definition in our papers [1,2]. Avg temp

aveAllR13_1: A circle with diameter of 13 pixels from the right canthus point to the face centerline Avg temp

aveOralM: Average oral temperature measured twice with the oral thermometer under monitor mode.

The data set is from a study looking into elevated body temperature. 1 oral physical and 10 Infrared imaging temperature measurements. ### ignore thisˆ

2.

Remove from the data set any record/observation which has a missing/NA value for Oral_Temp.

There are no missing/NA values in oral_temp

```r
oral_temp = data$aveOralM
# No NA in oral_temp
which(is.na(oral_temp))
```

```
## integer(0)
```

```r
# Returns error if we have NA
# na.fail(oral_temp)
```

There is NA values for oral/mouth region measured by IRT (image)

```r
#na.fail(data$T_OR1)

# Indices of NA observations
which(is.na(data$T_OR1))
```

```
## [1]  16 161
```

NAs in entire data set I tried to keep the NA observations, however I decided its easiest to remove them as they got in the way. For example I want to calculate the correlation between 2 variables and I couldn't due to NAs. We remove 5 observations.

```r
# Lets check the entire data set, Use array index
na_elements = which(is.na(data), arr.ind=TRUE)
na_indices = na_elements[ ,1]
# 5 observations with NA values
na_indices = unique(na_indices)

# Remove all NA from entire data set
data = data[-na_indices, ]
remove(na_indices, na_elements)


oral_temp = data$aveOralM
```

Error I had earlier: NAs will propagate conceptually, i.e., a resulting value will be NA whenever one of its contributing observations is NA

```r
cor(oral_temp, data$T_OR1)
```

```
## [1] 0.7482178
```

Check again for NAs. No NAs present in data set

```r
which(is.na(data), arr.ind = TRUE)
```

```
##      row col
```

```r
# na.fail(data)
str(data)
```

```
## 'data.frame':    1232 obs. of  15 variables:
##  $ Max1R13_1  : num  36.3 36.6 35.1 35.5 36 ...
##  $ T_RC1      : num  36.3 36.4 35.3 35.6 35.9 ...
##  $ T_LC1      : num  36.6 36.7 35.2 35.9 35.6 ...
##  $ RCC1       : num  36 36.3 34.9 34.9 35.7 ...
##  $ canthiMax1 : num  36.6 36.7 35.3 36 36 ...
##  $ T_FHCC1    : num  36.3 35.5 34.7 34.3 35.1 ...
##  $ T_FHLC1    : num  35.7 35.9 34.7 34.5 34.6 ...
```

```
##  $ T_FHTC1    : num  35.8 35.9 34.7 34 34.5 ...
##  $ T_OR1      : num  36 36.8 35.3 35.8 35.5 ...
##  $ aveAllR13_1: num  35.5 35.9 34.9 34.5 35.5 ...
##  $ aveOralM   : num  38 37.8 37.1 37.2 36.8 ...
##  $ Gender     : chr  "Male" "Female" "Female" "Female" ...
##  $ Age        : chr  "18-20" "21-25" "18-20" "18-20" ...
##  $ Ethnicity  : chr  "White" "Hispanic/Latino" "Asian" "White" ...
##  $ pyrexic    : int  1 0 0 0 0 1 0 1 0 1 ...
```

Lets resample (In case NA observations was included)

```
samp_data = data[sample(1:nrow(data), 1000), ]
oral_temp = samp_data$aveOralM
which(is.na(samp_data))
```

```
## integer(0)
```

Visualise the facial and oral temperature measurements using suitable plots

Renaming so box plots more clear. Where "Max" is not specified, assume avg measurement Avg_RCanthus_13IR should be RCanthus_13IR, I will leave it as is though.

```
colnames(samp_data)[1:11] = c("Max_RCanthus_13IR", "RCanthus_24x24IR", "LCanthus_24x24IR", "RCanthus_3x3
```

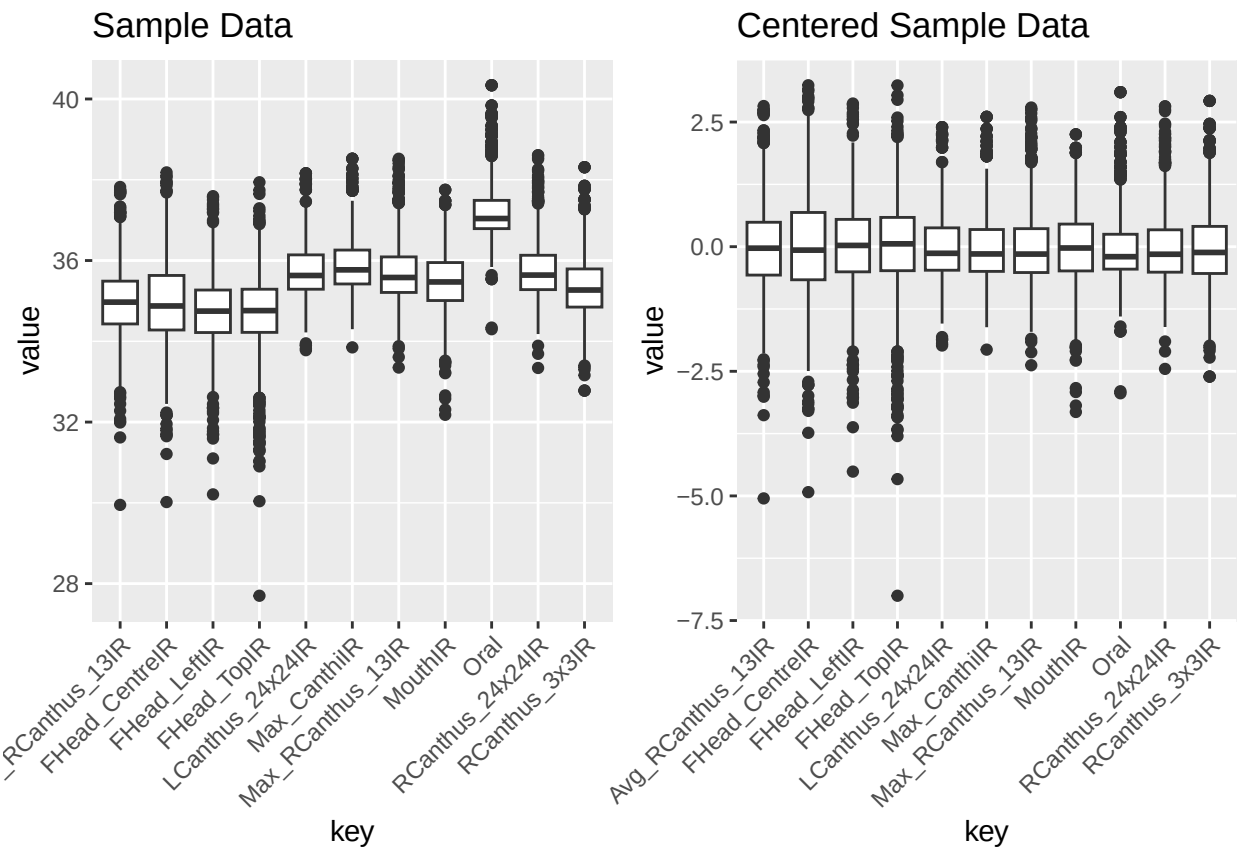
```
library("ggplot2")
```

```
## Warning: package 'ggplot2' was built under R version 4.3.2
```

```
library("tidyr")
library("gridExtra")
```

```
## Warning: package 'gridExtra' was built under R version 4.3.3
```

```
# Need long format for ggplot2
# Exclude the last few columns
samp_data_long = gather(samp_data[, -c(12:15)])


centered_samp_data = sweep(samp_data[, -c(12:15)], 2, colMeans(samp_data[, -c(12:15)]), "-")
centered_samp_data_long = gather(centered_samp_data)

plot1 = ggplot(samp_data_long, aes(x=key, y=value)) +
  geom_boxplot() + theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Sample Data")

plot2 = ggplot(centered_samp_data_long, aes(x=key, y=value)) +
  geom_boxplot() + theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Centered Sample Data")

grid.arrange(plot1, plot2, ncol = 2)
```




```
# Less clutter
remove(plot1, plot2, centered_samp_data_long, samp_data_long, centered_samp_data)
```

Comment on the plots

From the boxplot we can see that IR temperature measurements are consistently lower than the Oral temperature. Including the 'max' readings from the canthi & canthus areas. So initially when commented below I did not have the data centered.. On the right I have centered the mean. We are interested in the variance/spread of the data. Rather than the mean, as for example we could add to the mean a calibration value to solve the problem of the IR temperature readings being consistently lower.

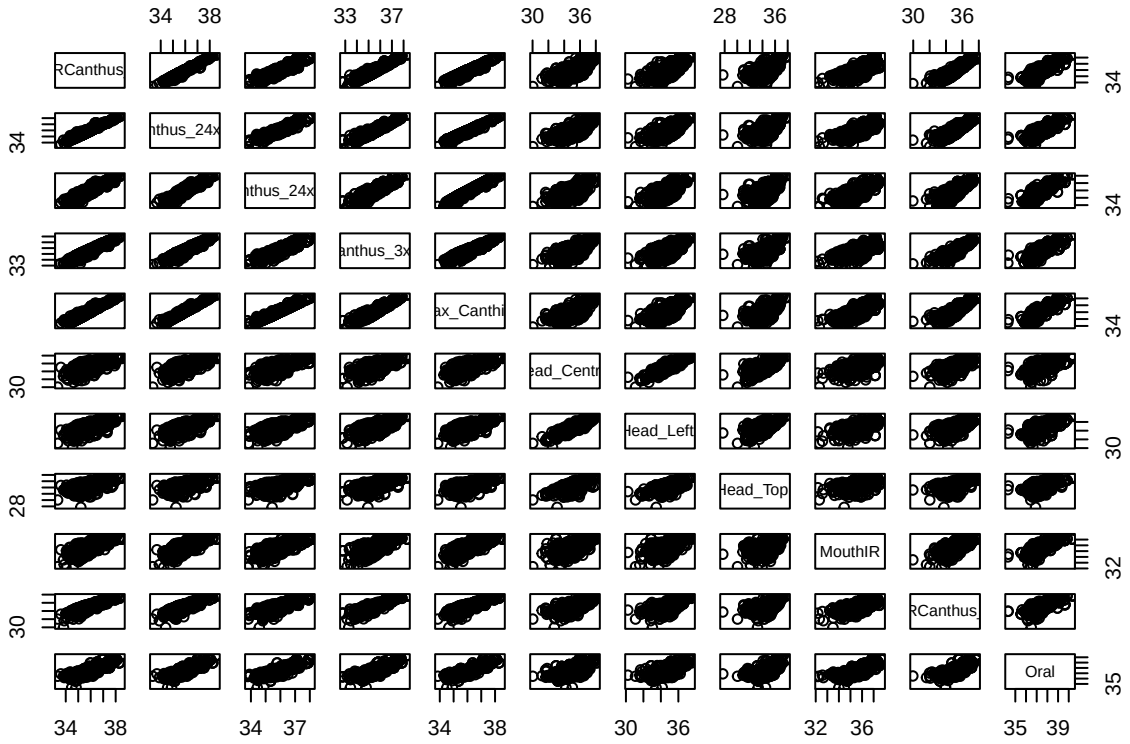
The range and interquartile range is tighter for the Oral temperatures.

A lot of very low readings (outliers) for the forehead measurements (boxplots 2 to 4)

Apart from measuring below the temperature of oral the two best ones from the looks of this boxplot is right canthus24x24 and max right canthus 13 diameter.

I think they capture most of the positive outliers (people with high temperature), look at the amount of dots above the whisker. They don't have a lot of negative outliers. And the spread is tight

```
pairs(samp_data[, -c(12:15)])
```



Comment on the plots From the pairs diagram we can see all the temperature measurements are positively correlated. Some more than others. They all move in the same general direction (as expected). The canthus/canthi areas seem to be tightly correlated with one another (example Max_RCanthus_13IR and RCanthus_24x24IR) in comparison to canthus/canthi and forehead/oral/mouth. In terms of comparing IR areas with the oral. Oral and Max Right Canthus13 or Right Canthus 24x24 area seems to be the most tightly packed correlation. Less variance between the 2. Look at bottom 2 left pictures. We will verify this with correlation values. Compared to last 5 columns on the last row which are forehead/mouthIR measurements vs oral. This is surprising as I would have thought the forehead region would be the more accurate measurement as this was what was done during the pandemic.

Remove any observations with Oral_Temp more than 4 standard deviations below the mean

With my seed there are no observations of Oral more than 4 standard deviations away.

```
oral_temp = samp_data$Oral
# Any reading below this is removed.
mean(oral_temp) - 4*sd(oral_temp)
```

```
## [1] 34.25912
```

```
oral_temp_outliers = which(oral_temp <= (mean(oral_temp) - 4*sd(oral_temp)))
oral_temp_outliers
```

```
## integer(0)
```

There is one very close though, so I will remove it regardless.

```r
outlier = min(oral_temp)
outlier
```

```
## [1] 34.3
```

```r
remove(oral_temp_outliers)
```

```r
# Remove from sample, tidy up
#samp_data = samp_data[-oral_temp_outliers, ]
#remove(oral_temp_outliers, data)
#oral_temp = samp_data$Oral

outlier_index = which(oral_temp == outlier) # Observation 435 is an outlier
samp_data = samp_data[-outlier_index, ]
oral_temp = samp_data$Oral
remove(outlier, outlier_index)
```

Correlations

```r
cor_list = c()
for (i in 1:10){
  cor_list[i] = cor(oral_temp, samp_data[, i])
}
remove(i)
max(cor_list)
```

```
## [1] 0.8629487
```

```r
sorted_cor_list = sort(cor_list, decreasing = TRUE)

which(cor_list == sorted_cor_list[c(1:2)])
```

```
## [1] 2 5
```

```r
colnames(samp_data)[c(2,5)] # highest cor
```

```
## [1] "RCanthus_24x24IR" "Max_CanthiIR"
```

So our correlation shows that Max_CanthiIR and RCanthus_24x24IR has the least covariance with Oral (highest correlation of 0.863 & 0.852). We want to find a temperature measuring method that will follow the Oral temperature as much as possible, so comparing correlations is a good start. I'm slightly off by what I said previously by just looking visually at the box plots, but I was in the rough ballpark.

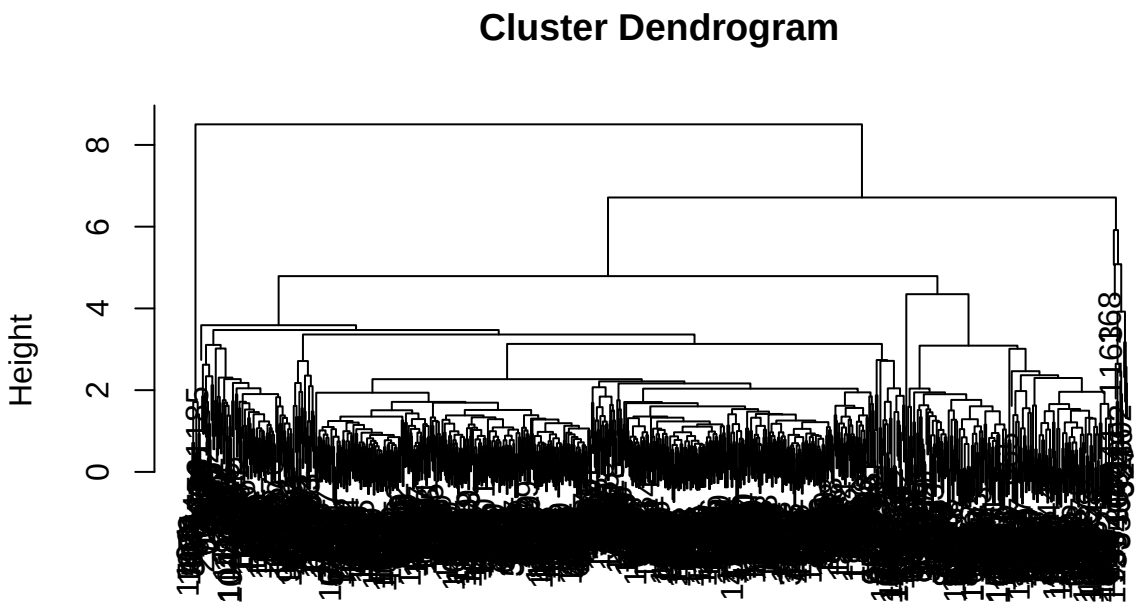
Side note: Don't need to scale data since same measurement units used

```r
remove(sorted_cor_list, cor_list)
```

3.

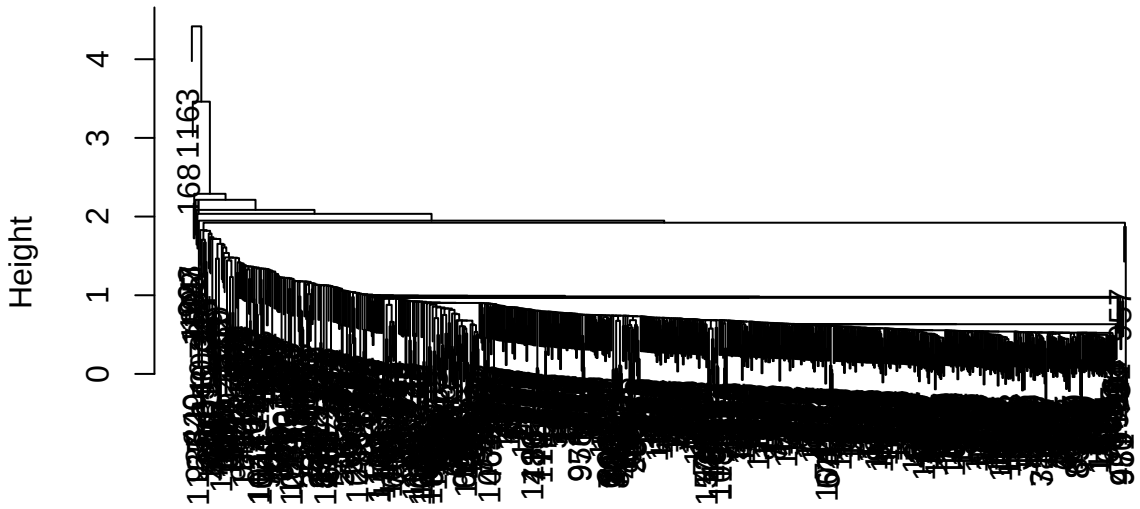
Use hierarchical clustering and k-means clustering on the facial temperature measurements

```r
cluster_avg = hclust(dist(samp_data[, -c(12:15)]), method = "average")
plot(cluster_avg, xlab="Average linkage", sub="")
```



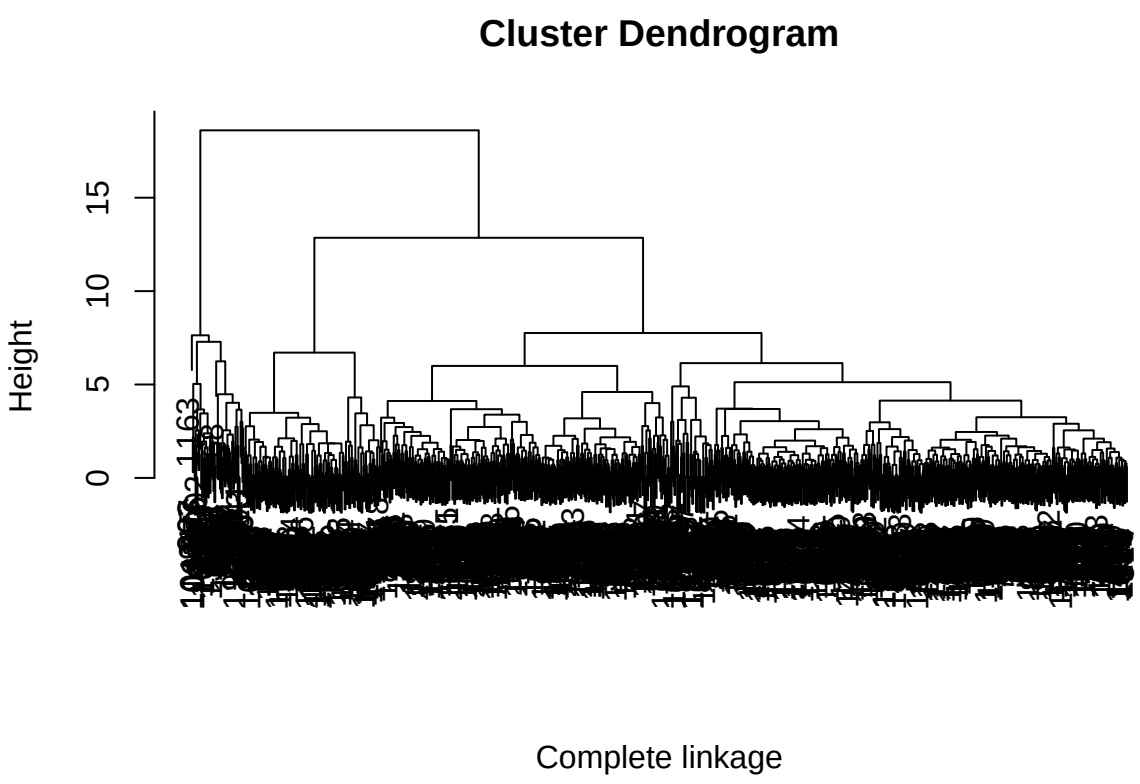
Average linkage

```
cluster_sing = hclust(dist(samp_data[, -c(12:15)]), method = "single")
plot(cluster_sing, xlab="Single linkage", sub="")
```



Single linkage

```
cluster_compl = hclust(dist(samp_data[, -c(12:15)]), method ="complete")
plot(cluster_compl, xlab="Complete linkage", sub="")
```



Complete linkage

```
remove(cluster_sing)
```

Motivate any decisions you make

From looking at the clusters, the single linkage has resulted in chaining, we will discard this method. Complete linkage has found 2 or 3 distinct groups. Average linkage has found 3 distinct groups. But 2 of these groups contain very little observations (maybe outliers?)

If we look at the pyrexic, we have a binary class group. This can be viewed as the true categories, we can use this for comparison later. 37.89 is the cut off temperature, oral reading equal or above this is pyrexic 1.

```
min(samp_data[samp_data$pyrexic == 1, "Oral"])
```

```
## [1] 37.89
```

I chose Euclidean distance as this is a standard distance method, I don't see why we shouldn't use this, as it measures the distances between observations in multi dimensional space and the measurement units are the same.

TRUE CLASSES We will use theses plot to compare our Hierarchical Clusters. I plot 2 of the variables that had the most correlation with Oral, i.e the 2 IR measuring methods that seemed to move in line with the Oral temperatures the most. (Below that is also of MouthIR and ForeheadIR against Oral.) There mean can be adjusted in a calibration later. I do find it strange how we have not a small number of observations whose temperatures were recorded below 36.4 orally. I would consider these outliers (maybe measurement was taken incorrect, or faulty device etc) or if they are accurate have another class, because having too low of a temperature is also indication of illness. It would be good to know the Oral temperatures device standard margin of error.

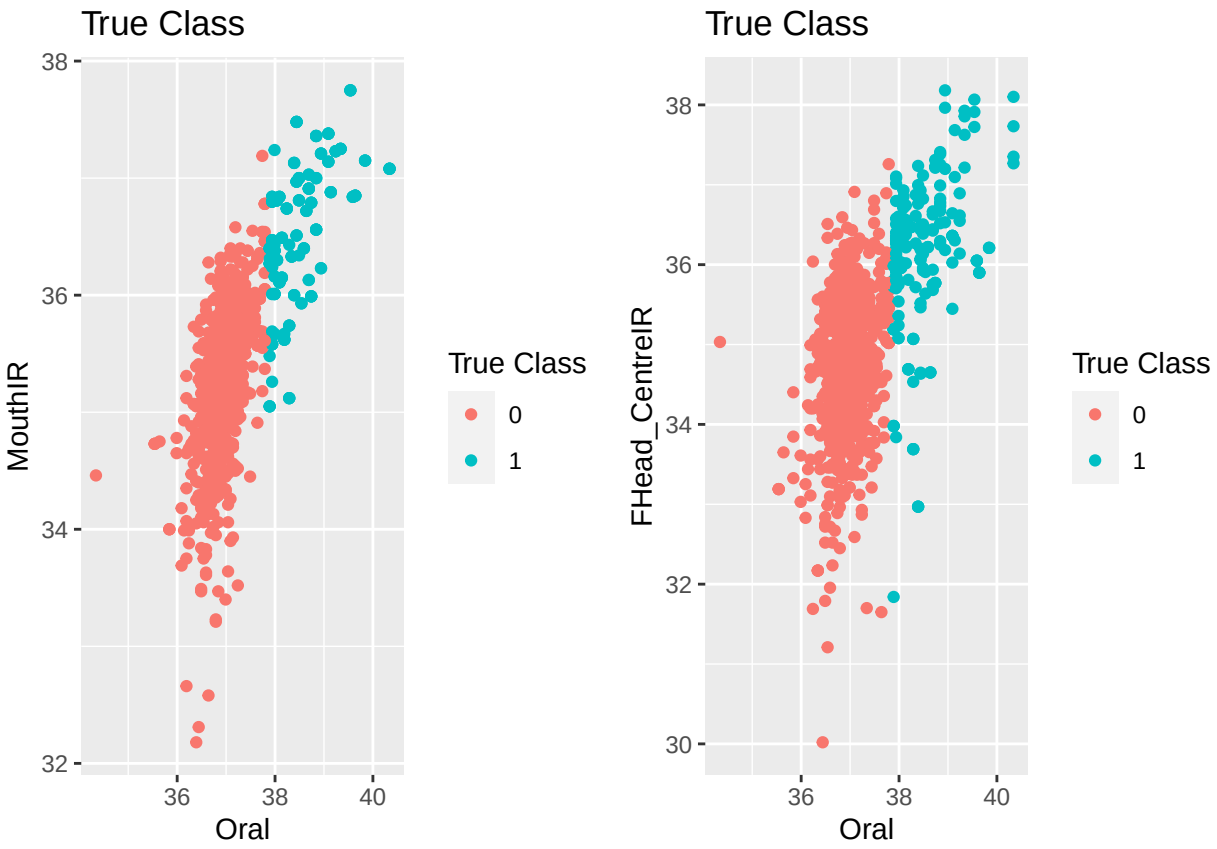
Max CanthiIR vs Oral, Right Canthus 24x24 vs Oral

```
# Need this for side by side
library(gridExtra)

plot1 = ggplot(samp_data, aes(x=Oral, y=Max_CanthiIR, colour=factor(pyrexic)))+geom_point()+labs(title =
  guides(colour = guide_legend(title = "True Class"))
plot2 = ggplot(samp_data, aes(x=Oral, y=RCanthus_24x24IR, colour=factor(pyrexic)))+geom_point()+labs(ti
  guides(colour = guide_legend(title = "True Class"))

grid.arrange(plot1, plot2, ncol = 2)
```


```
remove(plot1 ,plot2)
```

MouthIR area vs Oral, Forehead CenterIR vs Oral

```
plot3 = ggplot(samp_data, aes(x=Oral, y=MouthIR, colour=factor(pyrexic)))+geom_point()+labs(title = "Tru
  guides(colour = guide_legend(title = "True Class"))
plot4 = ggplot(samp_data, aes(x=Oral, y=FHead_CentreIR, colour=factor(pyrexic)))+geom_point()+labs(titl
  guides(colour = guide_legend(title = "True Class"))
grid.arrange(plot3, plot4, ncol = 2)
```




```
remove(plot3 ,plot4)
```

So from looking at these graphs I have also noticed the Oral temperature readings seem to move in steps. It's not continuous, if you look at the data set Oral, you can see the same exact temperature values appear multiple times.

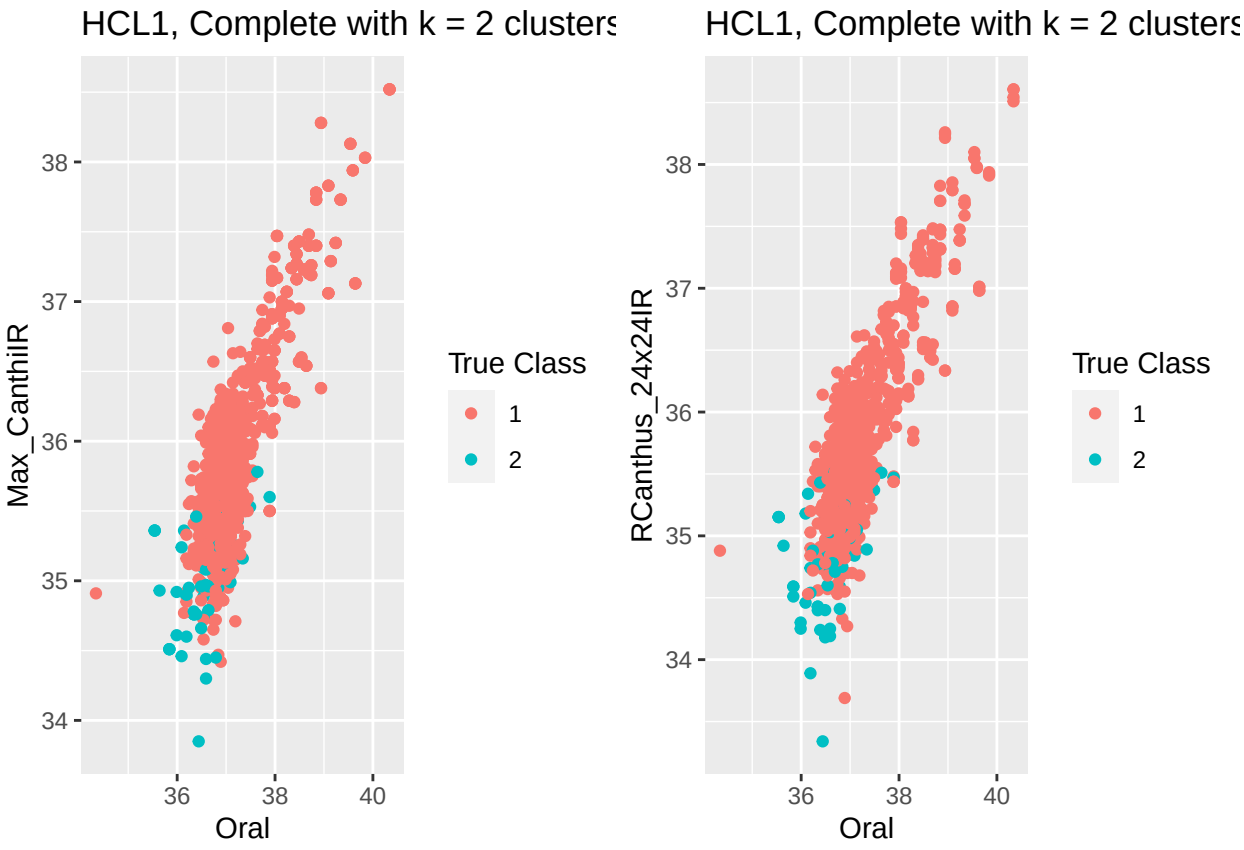
Complete HCL k = 2 clusters Not successful, discard this. However you could argue it detects extreme low temperature outliers

```
hcl1 = cutree(cluster_compl, k = 2)
table(hcl1)
```

```
## hcl1
##   1   2
## 942  57
```

```
plot5 = ggplot(samp_data,
               aes(x=Oral,
              y=Max_CanthiIR,
              colour=factor(hcl1),
              factor(pyrexic)))+
  geom_point()+
  labs(title = "HCL1, Complete with k = 2 clusters")+ scale_shape_manual(values = c(16, 17))+
  guides(colour = guide_legend(title = "True Class"))

plot6 = ggplot(samp_data, aes(x=Oral, y=RCanthus_24x24IR, colour=factor(hcl1), factor(pyrexic)))+
  geom_point()+
  labs(title = "HCL1, Complete with k = 2 clusters")+
```

```
  scale_shape_manual(values = c(16, 17))+
  guides(colour = guide_legend(title = "True Class"))


grid.arrange(plot5, plot6, ncol = 2)
```




```
remove(plot5, plot6, hcl1)
```

Complete HCL with k = 3 clusters

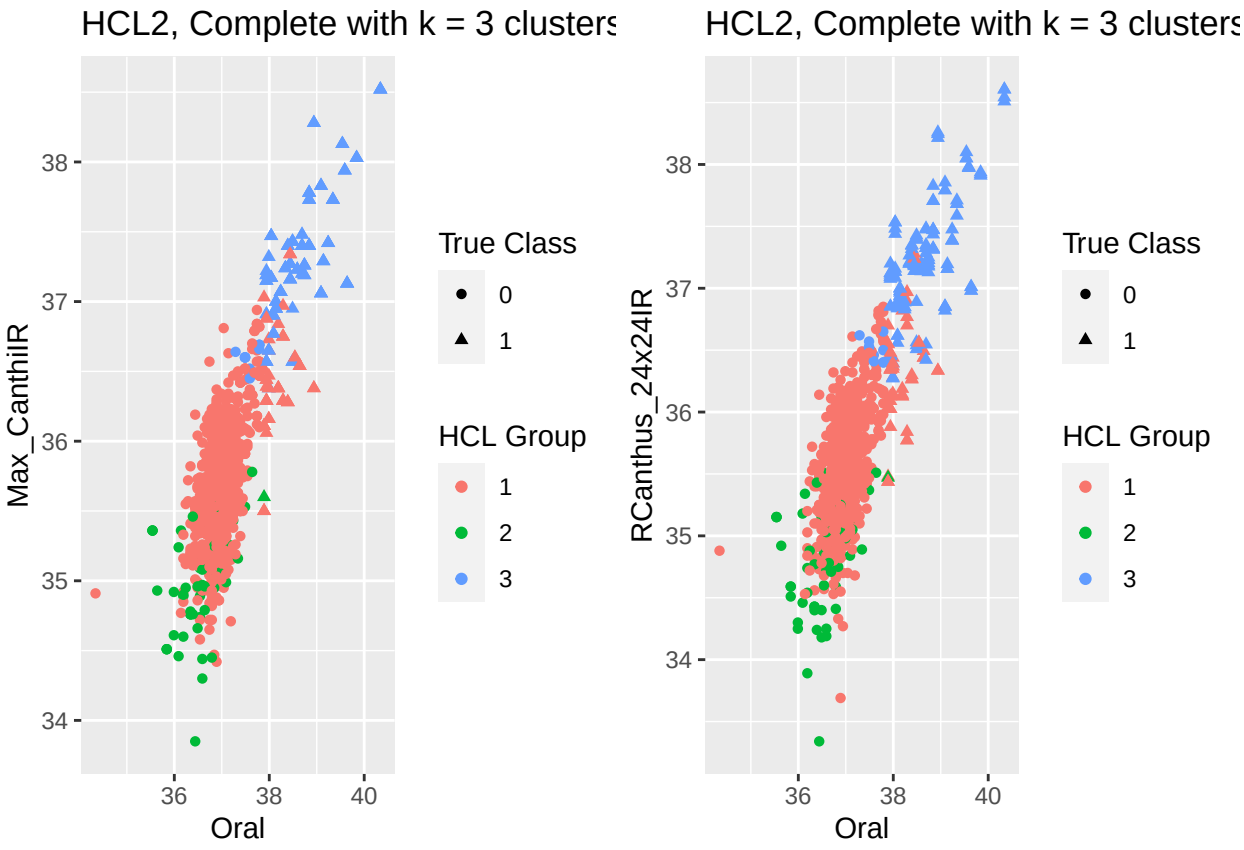
```
hcl2 = cutree(cluster_compl, k = 3)
table(hcl2)
```

```
## hcl2
##   1   2   3
## 797  57 145
```

I don't think this graph does a horrible job, it maybe sorts into 'low', 'normal' and 'high' temperatures. If you combine group1 and group2(red&green) it looks similar to the graph with 'pyrexic' as class.

```
plot7 = ggplot(samp_data, aes(x=Oral, y=Max_CanthiIR, colour=factor(hcl2), shape = factor(pyrexic)))+
  geom_point()+
  labs(title = "HCL2, Complete with k = 3 clusters")+
  scale_shape_manual(values = c(16, 17))+
  guides(shape = guide_legend(title = "True Class"), colour = guide_legend(title="HCL Group"))

plot8 = ggplot(samp_data, aes(x=Oral, y=RCanthus_24x24IR, colour=factor(hcl2), shape = factor(pyrexic)))
  geom_point()+
```

```
  labs(title = "HCL2, Complete with k = 3 clusters")+
  scale_shape_manual(values = c(16, 17))+
  guides(shape = guide_legend(title = "True Class"), colour = guide_legend(title="HCL Group"))

grid.arrange(plot7, plot8, ncol = 2)
```




```
remove(plot7, plot8)
```

Lets combine HCL group 1 & 2, since we are only focusing on categorizing those with high temperature into 'pyrexic'. Our TRUE class is 'y' (pyrexic), the classes assigned with HCL is 'yhat'

```
vector_hcl2 = as.vector(hcl2)

for (i in 1:length(vector_hcl2)){
  if (vector_hcl2[i] == 2 | vector_hcl2[i] == 1) {
    vector_hcl2[i] = 0
  }
}

for (i in 1:length(vector_hcl2)){
  if (vector_hcl2[i] == 3) {
    vector_hcl2[i] = 1
  }
}

row_names = c("yhat = 0", "yhat = 1")
col_names = c("y = 0", "y = 1")
```

```r
vector_hcl2 = as.factor(vector_hcl2)
tab1 = table(vector_hcl2, samp_data$pyrexic)
tab1 = as.matrix(tab1)
dimnames(tab1) = list(row_names, col_names)
tab1
```

```
##          y = 0 y = 1
## yhat = 0   798    56
## yhat = 1    11   134
```

Performance Metrics Function

```r
performance_metrics = function(m){

  acc = sum(diag(m)) / sum(m)  # P(y = yhat | y, yhat)
  tpr = m[2, 2] / sum(m[, 2])  # P(yhat = 1 | y = 1)
  tnr = m[1, 1] / sum(m[, 1])  # P(yhat = 0 | y = 0)

  precision = m[2, 2] / sum(m[2, ])  # P(y = 1 | yhat = 1)

  sum_sens_spec = tpr + tnr  # Maximize this to balance true positives & false positives

  f1 = 2*( (precision*tpr) / (precision+tpr) ) #balanced ability to both detect positive cases (recall)

  # If the classes are significantly imbalanced, F1 and sensitivity + specificity can lead to different
  metrics = matrix(NA, nrow = 1, ncol = 6, dimnames = list(NULL, c("acc", "tpr", "tnr", "precision", "S
  metrics[1, ] = c(acc, tpr, tnr, precision, sum_sens_spec, f1)
  return(metrics)
}
```

```r
performance_metrics(tab1)
```

```
##            acc       tpr      tnr precision Sens+Spec  f1
## [1,] 0.9329329 0.7052632 0.986403 0.9241379  1.691666 0.8
```

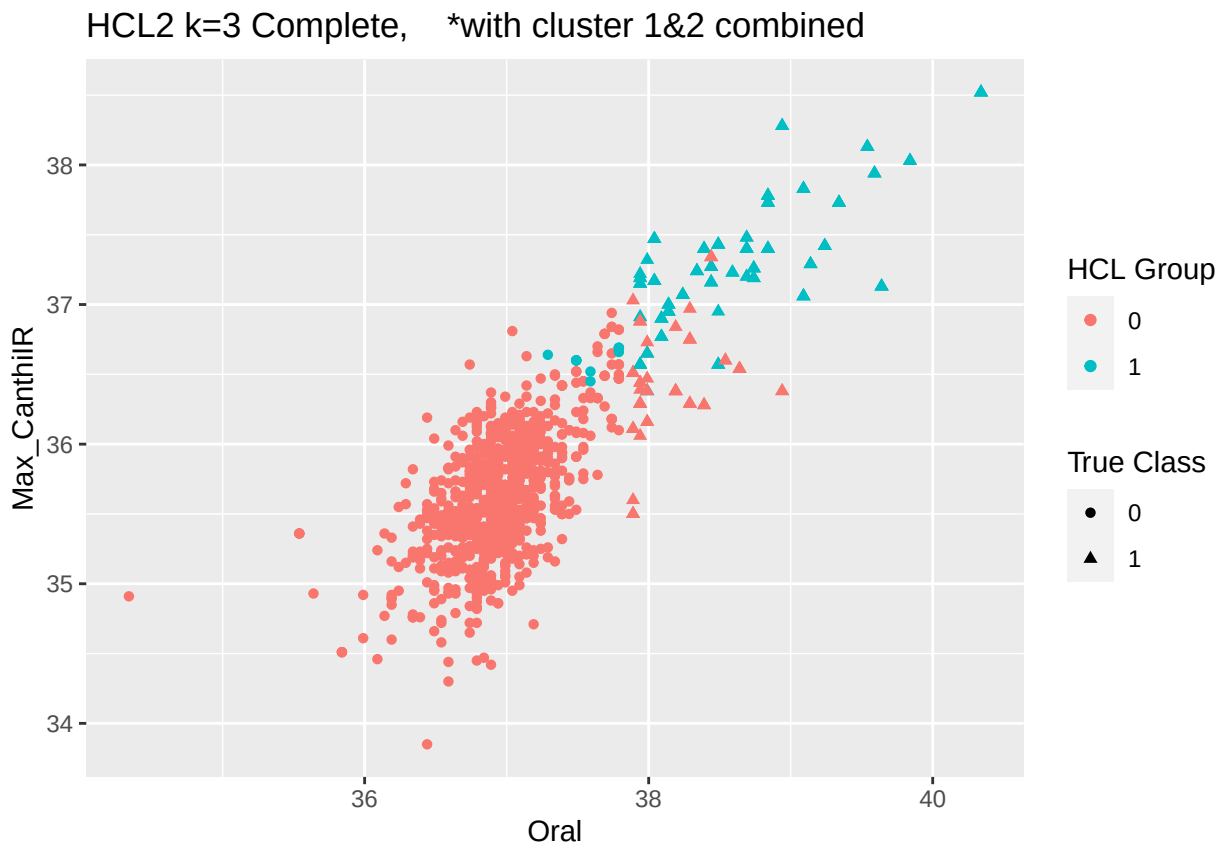
```r
remove(cluster_compl, col_names, hcl2, tab1, row_names, i)
```

Complete HCL with k=3 and we combine clusters 1&2 into 1 single group. So in actuality we have 2 clusters. This method does a pretty decent job, With an accuracy of 0.93. However this metric is skewed due to the low prevalence of pyrexic = 1, which is 0.19. Misclasification rate = 0.07.

A better metric to look at is the TPR, I think we would be more concerned about accurately classifying those that are truly pyrexic. This is P(y = 1 | yhat = 1) = 0.71.

To conclude this HCL method is a good starting point to get an idea of how many groups the data can be divided into based on dissimilarity, we know now that there are 2 groups. We combined the lower outliers with the 'normal' temperatures and separated them from the high outliers (high temperatures).

```r
ggplot(samp_data, aes(x=Oral, y=Max_CanthiIR, colour=vector_hcl2, shape = factor(pyrexic)))+
  geom_point()+
  labs(title = "HCL2 k=3 Complete,    *with cluster 1&2 combined")+
  scale_shape_manual(values = c(16, 17))+
  guides(shape = guide_legend(title = "True Class"), colour = guide_legend(title="HCL Group"))
```

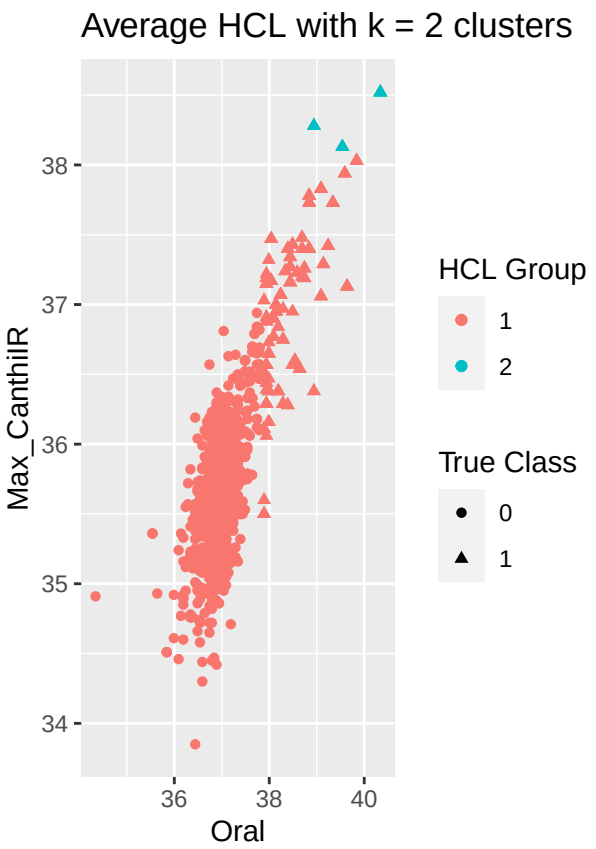


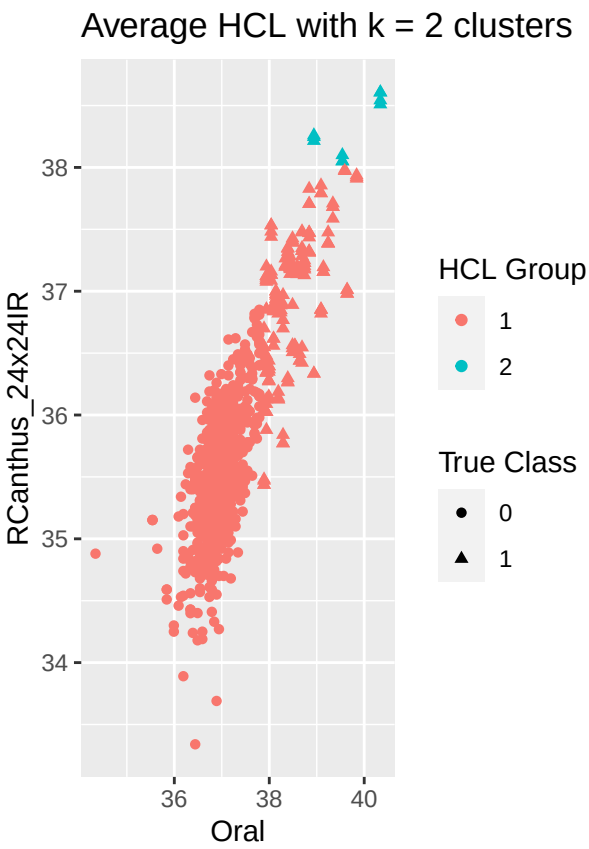
Average HCL with k = 2 clusters This is not useful, we can discard this

```
hcl3 = cutree(cluster_avg, k = 2)
table(hcl3)
```

```
## hcl3
##   1   2
## 989  10
```

```
plot9 = ggplot(samp_data, aes(x=Oral, y=Max_CanthiIR, colour=factor(hcl3), shape = factor(pyrexic)))+
  geom_point()+
  labs(title = "Average HCL with k = 2 clusters")+
  scale_shape_manual(values = c(16, 17))+
  guides(shape = guide_legend(title = "True Class"), colour = guide_legend(title="HCL Group"))

plot10 = ggplot(samp_data, aes(x=Oral, y=RCanthus_24x24IR, colour=factor(hcl3), shape = factor(pyrexic)
  geom_point()+
  labs(title = "Average HCL with k = 2 clusters")+
  scale_shape_manual(values = c(16, 17))+
  guides(shape = guide_legend(title = "True Class"), colour = guide_legend(title="HCL Group"))

grid.arrange(plot9, plot10, ncol = 2)
```





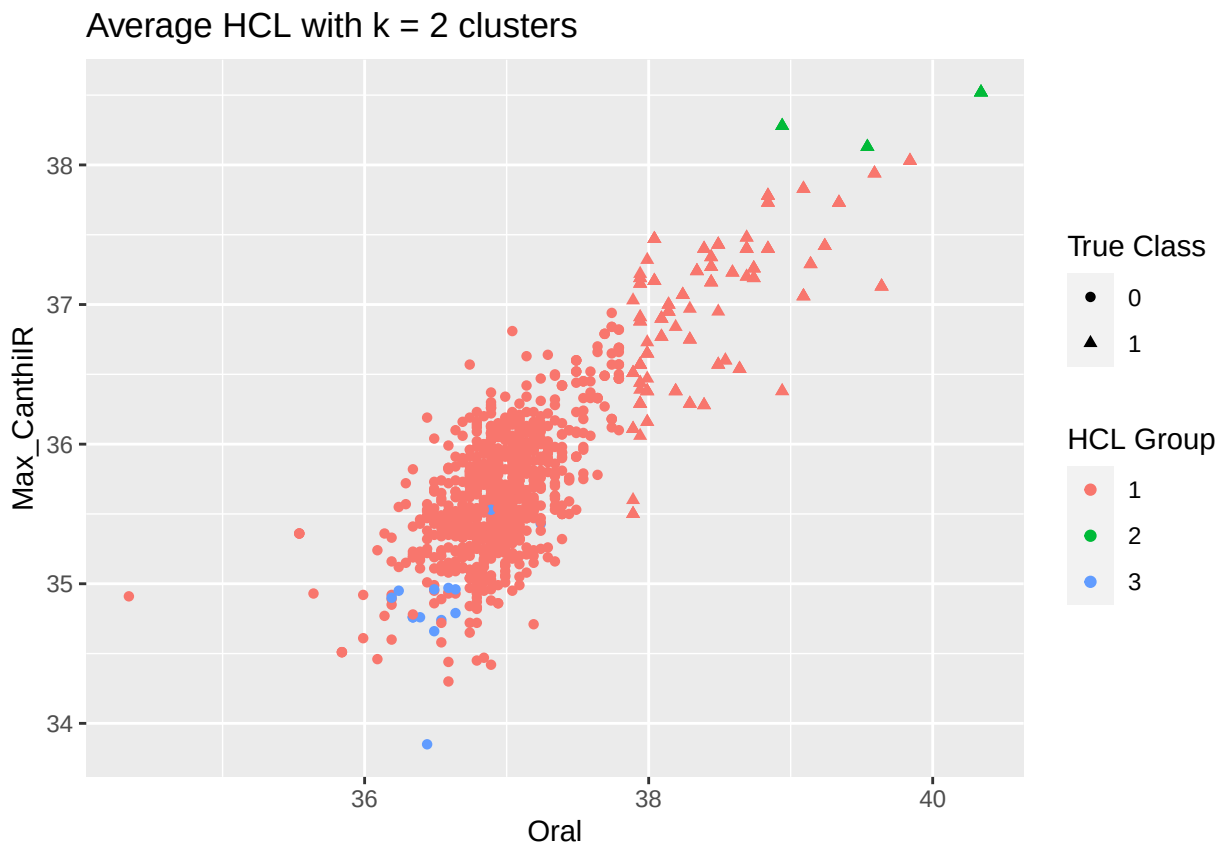

```
remove(hcl3, plot9, plot10)
```

Average HCL with k = 3 clusters I tested 3,4,5 (no good). This is not useful we can discard this.

```
hcl4 = cutree(cluster_avg, k = 3)
table(hcl4)
```

```
## hcl4
##   1   2   3
## 974  10  15
```

```
ggplot(samp_data, aes(x=Oral, y=Max_CanthiIR, colour=factor(hcl4), shape = factor(pyrexic)))+geom_point
  guides(shape = guide_legend(title = "True Class"), colour = guide_legend(title="HCL Group"))
```




```
remove(hcl4)
```

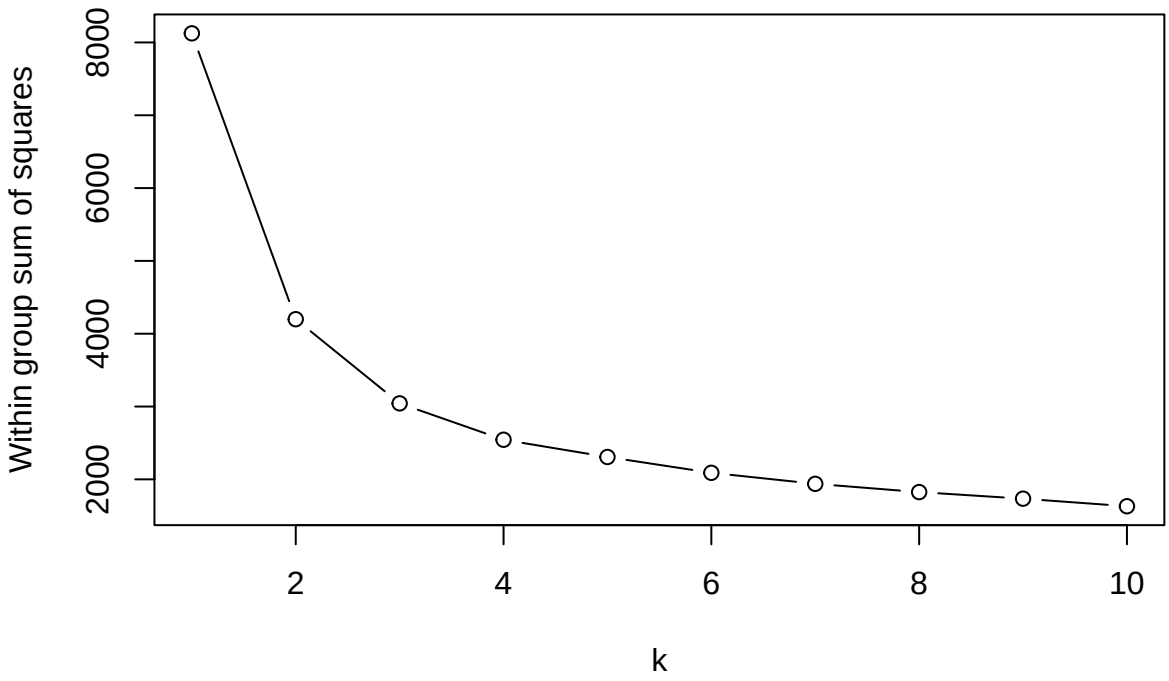
Clean Up

```
remove(cluster_avg, vector_hcl2)
```

k-means Clustering

Within group sum of squares

```
numeric_samp_data = samp_data[, 1:11]

wgss = rep(0, 10)
n = nrow(numeric_samp_data)
#WGSS for when k=1
wgss[1] = (n - 1)*sum(apply(numeric_samp_data, 2, var))

for (k in 2:10){
  wgss[k] = sum(kmeans(numeric_samp_data, centers=k, nstart=10)$withinss)
}

plot(1:10, wgss, type="b", xlab="k", ylab="Within group sum of squares")
```



The plot indicates 2 or 3 groups, we look for a bend in the graph. Again with 3 groups we could have 'low', 'normal' and 'high'.

k-means initial cluster centers chosen randomly, updates centroid location when observations added. Iterative. Selects the one that results in the lowest total within-cluster variation, also known as the "total withinss." nstarts is how many times this is repeated, to avoid local minimums.

K-Means with k = 2 Clusters

```
k = 2
pcl1 = kmeans(numeric_samp_data, center=k, nstart=10)

row_names = c("yhat = 0", "yhat = 1")
col_names = c("y = 0", "y = 1")

tab1 = table(pcl1$cluster, samp_data$pyrexic)
tab1 = as.matrix(tab1)
dimnames(tab1) = list(row_names, col_names)
#tab1
# NOTE! due to randomness of centroid positioning, the groups can be inverted
# ENSURE CORRECT DIAGONALS IN TABLE BELOW
# For my seed , need to run 3 times

row_sums = rowSums(tab1)
col_sums = colSums(tab1)

order_rows = order(row_sums)
order_cols = order(col_sums)
```

```
tab1_reordered = tab1[order_rows, order_cols]

row_names = c("yhat = 0", "yhat = 1")
col_names = c("y = 0", "y = 1")


dimnames(tab1_reordered) = list(row_names[order_rows], col_names[order_cols])
tab1_reordered
```

```
##          y = 1 y = 0
## yhat = 0   170    72
## yhat = 1    20   737
```

YHAT IS ARBITRARY RANDOM GROUP ASSIGNMENT NAME

Performance Metrics

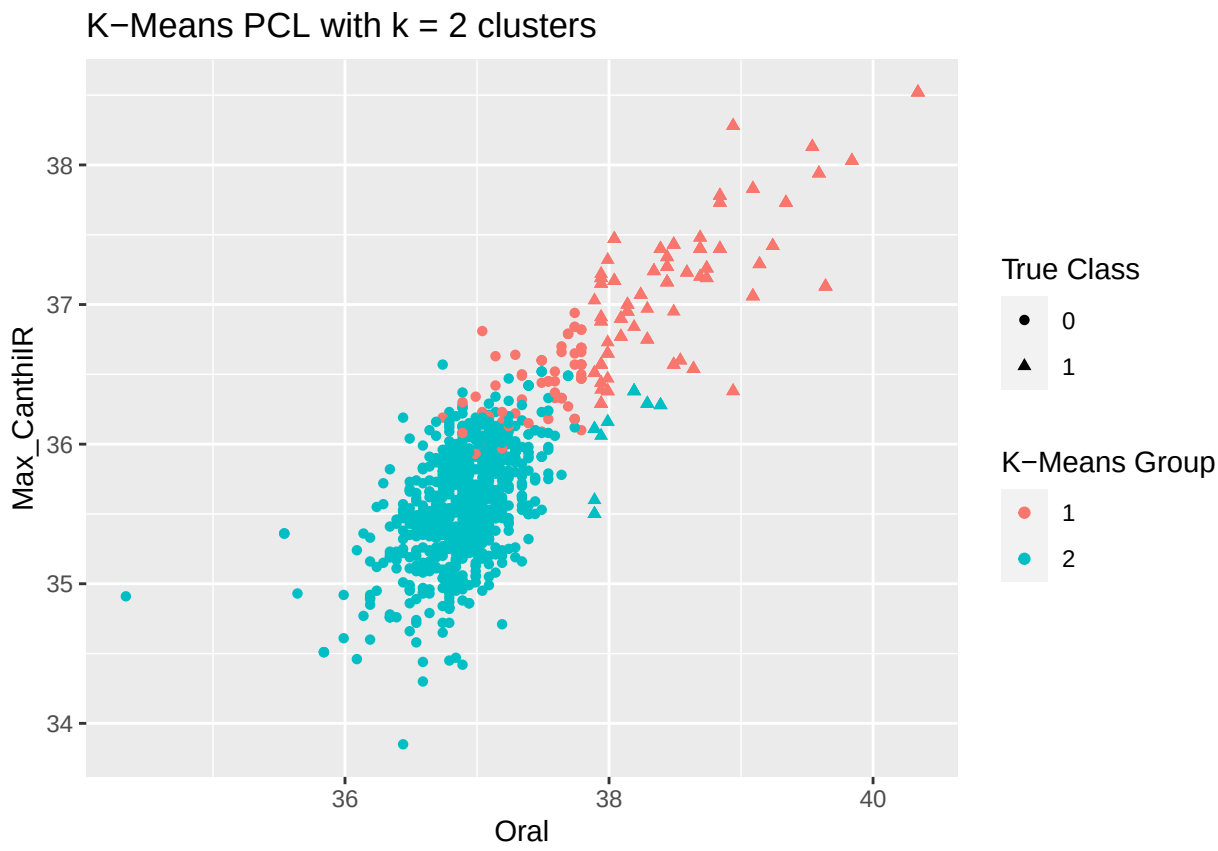
```
performance_metrics(tab1_reordered)
```

```
##            acc       tpr       tnr precision Sens+Spec        f1
## [1,] 0.9079079 0.9110012 0.8947368 0.9735799  1.805738 0.9412516
```

We can see straight away the K-Means method is much better at correctly classifying TRUE pyrexic = 1 groups. However the precision (given yhats what proportion of them are actually true ys) has decreased, the k-means has over classified 1's (positive cases).

During a pandemic for example I think it is better to over classify than to under classify.

```
ggplot(samp_data, aes(x=Oral, y=Max_CanthiIR, colour=factor(pcl1$cluster), shape = factor(pyrexic)))+
  geom_point()+
  labs(title = "K-Means PCL with k = 2 clusters")+
  scale_shape_manual(values = c(16, 17))+
  guides(shape = guide_legend(title = "True Class"), colour = guide_legend(title="K-Means Group"))
```

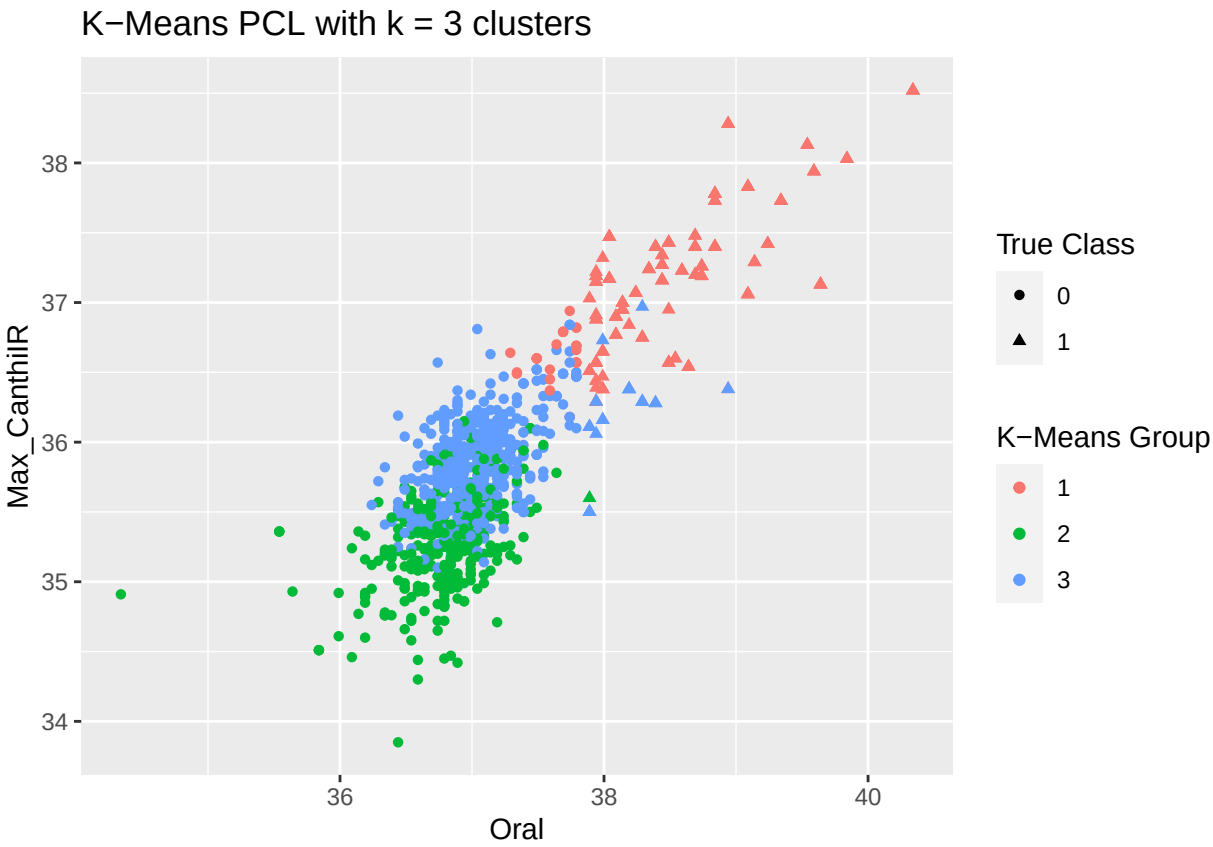


K-Means with k = 3 Clusters

```
k = 3
pcl1 = kmeans(numeric_samp_data, center=k, nstart=10)

row_names = c("yhat = 0", "yhat = 1")
col_names = c("y = 0", "y = 1")

tab1 = table(pcl1$cluster)
tab1
```

```
##
##   1   2   3
## 190 309 500
```

```
ggplot(samp_data, aes(x=Oral, y=Max_CanthiIR, colour=factor(pcl1$cluster), shape = factor(pyrexic)))+
  geom_point()+
  labs(title = "K-Means PCL with k = 3 clusters")+
  scale_shape_manual(values = c(16, 17))+
  guides(shape = guide_legend(title = "True Class"), colour = guide_legend(title="K-Means Group"))
```



With K-Means k = 3 clusters, our data appears to be divided up accordingly as temperature goes up into 3 groups.

Performance Metrics k = 3

```
# if we combine blue and red
x1 = 809
x2 = 190
table(samp_data$pyrexic)
```

```
##
##   0   1
## 809 190
```

Clean up

```
remove(k, n,row_names, col_names, tab1, wgss, x1, x2, pcl1)
```

4.

Perform a linear or quadratic discriminant analysis to classify subjects by gender

LDA has a vector of different means and a common covariance structure for the different variables.

To minimize misclassification we can maximize P(g|x) prob of belonging to group g given the observation by maximizing the LOG of the top part of Bayes formula. This is linear discriminant function. By maximizing the LDF we find the parameters that best fit that x observation. I.e we sub in the different mean vectors for different classes and the same common cov matrix and pick the group that returns the highest value.

The prior is the sample prevalence of the classes.

To get common covariance you get the weighted mean of all the covariances given the different classes. For example cov1 = covariance between the predictor variables for male data only cov2 = covariance between between the predictor variables for female data only Then take a weighted average of this.

```
library(MASS)
lda1 = lda(Gender ~ . ,data = samp_data[, -c(13:15)])
lda1
```

```
## Call:
## lda(Gender ~ ., data = samp_data[, -c(13:15)])
##
## Prior probabilities of groups:
##    Female      Male
## 0.5515516 0.4484484
##
## Group means:
##        Max_RCanthus_13IR RCanthus_24x24IR LCanthus_24x24IR RCanthus_3x3IR
## Female          35.56001         35.63388         35.61851       35.21448
## Male            35.93837         35.98856         35.94315       35.60036
##        Max_CanthiIR FHead_CentreIR FHead_LeftIR FHead_TopIR  MouthIR
## Female     35.76481       34.46414     34.50463    34.57935 35.36584
## Male       36.10420       35.53627     34.99273    34.85789 35.66047
##        Avg_RCanthus_13IR     Oral
## Female          34.80992 37.14463
## Male            35.23669 37.36545
##
## Coefficients of linear discriminants:
##                             LD1
## Max_RCanthus_13IR  0.598888579
## RCanthus_24x24IR  -1.065790012
## LCanthus_24x24IR  -0.371165942
## RCanthus_3x3IR    -0.078733528
## Max_CanthiIR       0.928456956
## FHead_CentreIR     2.833451765
## FHead_LeftIR      -1.424507872
## FHead_TopIR       -0.796120030
## MouthIR            0.004064859
## Avg_RCanthus_13IR -0.058564086
## Oral              -0.272614572
```

Assess how well your classifier performs using an appropriate method

Cross Validation Predicted Class (You can access posterior in the lda1_cv) For the table classes assigned due to LDA is on the left (rows) and true class is on top (columns)

```
lda1_cv = lda(Gender ~ . ,data = samp_data[, -c(13:15)], CV=TRUE)

#Predictions
tab1 = table(lda1_cv$class, samp_data$Gender)
tab1
```

```
##
##          Female Male
##   Female    521   91
```

```
##   Male       30  357
```

Performance Metrics

```
performance_metrics(tab1)
```

```
##            acc      tpr       tnr precision Sens+Spec        f1
## [1,] 0.8788789 0.796875 0.9455535 0.9224806  1.742429 0.8550898
```

Here TPR corresponds to predicted class being male given the observation is actually male and TNR the same for female. LDA does a good job in discriminating between the classes. This makes sense as the covariance should be the same for male/female, and it is common knowledge that the avg temperature of women is different than to men (or so I've heard). So it is intuitive that the data would be separated well according to genders using the temperatures from the various methods.

If the discriminant score for class 1 is greater than the score for class 2, the observation is classified as class 1; otherwise, it is classified as class 2.

To get the LDF values for each observation multiply each observation with the LD coefficients.

```
lda_coef = lda1$scaling[, 1]

ldf_values = as.matrix(samp_data[, -c(12:15)]) %*% lda_coef
```

LDF value for 2nd observation

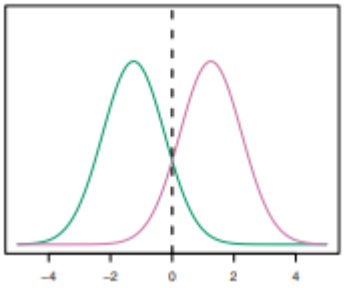
```
as.matrix(samp_data[4, -c( 12:15)]) %*% lda_coef
```

```
##         [,1]
## 656 12.81075
```

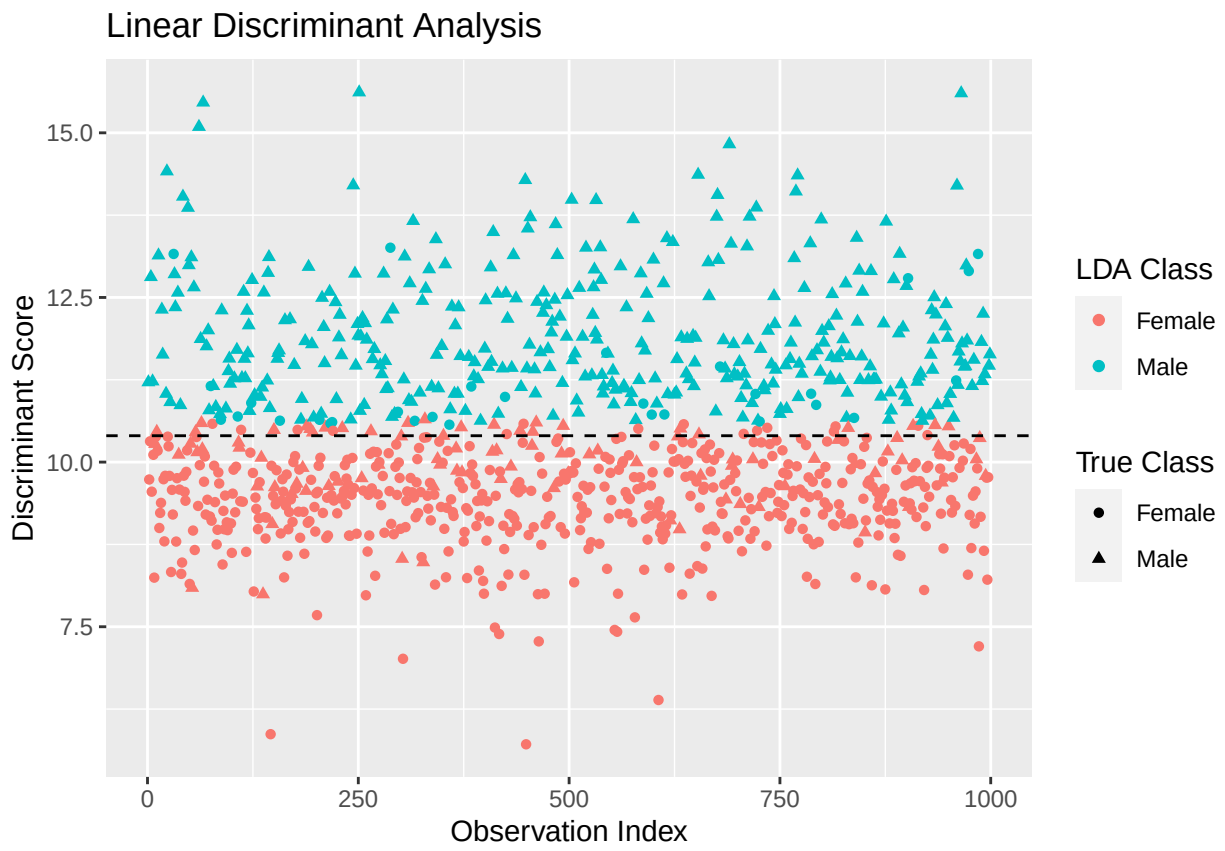
Produce a plot showing the linear decision boundary for your LDA

Another good way to visualize this is, if you have 2 normal distributions, with same variance. but separate means, the linear decision boundary is where these 2 normal distributions intersect

```
knitr::include_graphics("LDA boundary example.png")
```




```
ldf_mean = mean(ldf_values)

ggplot(data = NULL, aes(x = 1:length(ldf_values), y = ldf_values, color = factor(lda1_cv$class), shape =
  geom_point()+
  labs(title = "Linear Discriminant Analysis", x="Observation Index", y="Discriminant Score")+
  scale_shape_manual(values = c(16, 17))+
  guides(shape = guide_legend(title = "True Class"), color=guide_legend(title = "LDA Class"))+
  geom_hline(yintercept = ldf_mean, linetype = "dashed", color = "black")
```



Quadratic

```
qda1 = qda(Gender ~ . ,data = samp_data[, -c(13:15)])
qda1
```

```
## Call:
## qda(Gender ~ ., data = samp_data[, -c(13:15)])
##
## Prior probabilities of groups:
##    Female      Male
## 0.5515516 0.4484484
##
## Group means:
##        Max_RCanthus_13IR RCanthus_24x24IR LCanthus_24x24IR RCanthus_3x3IR
## Female          35.56001         35.63388         35.61851       35.21448
## Male            35.93837         35.98856         35.94315       35.60036
##        Max_CanthiIR FHead_CentreIR FHead_LeftIR FHead_TopIR  MouthIR
## Female     35.76481       34.46414     34.50463    34.57935 35.36584
## Male       36.10420       35.53627     34.99273    34.85789 35.66047
##        Avg_RCanthus_13IR     Oral
## Female          34.80992 37.14463
## Male            35.23669 37.36545
```

```
qda1_cv = qda(Gender ~ . ,data = samp_data[, -c(13:15)], CV=TRUE)
tab2 = table(qda1_cv$class, samp_data$Gender)
tab2
```

```
##
```

```
##          Female Male
##   Female    510  100
##   Male       41  348
```

Performance Metrics

```
performance_metrics(tab2)
```

```
##            acc       tpr       tnr precision Sens+Spec        f1
## [1,] 0.8588589 0.7767857 0.9255898 0.8946015  1.702376 0.8315412
```

Compare the performance of LDA and QDA in classifying the gender, and comment on your results.

The QDA performs slightly worse. This could be due to that QDA assumes each class has its own unique covariance, while LDA assumes a common covariance for both classes. I see no reason why the covariance between the 11 temperature measuring devices would be different for each gender.

clean up

```
remove(lda1, lda1_cv, ldf_values, qda1, qda1_cv, lda_coef, ldf_mean, tab1, tab2)
```

5.

PCA aims to describe the variation in a set of correlated variables, in-terms of a new set of uncorrelated variables.

The new uncorrelated variables are the principal components.

The goal is to find a set of orthogonal axes (uncorrelated principal components)

Think about it like that. You have, let's say, 1000 data points in 12 -dimensional space (i.e. your data matrix X is of 1000×12 size). PCA finds directions in this space that capture maximal variance. So for example PC1 direction is a certain axis in this 12 -dimensional space, i.e. a vector of length 12. PC2 direction is another axis, etc.

All your 1000 data points can be projected onto each of these directions/axes, yielding coordinates of 1000 data points along each PC direction; these projections are what is called PC scores.

Question: https://towardsdatascience.com/principal-component-analysis-pca-8133b02f11bd

If I want to plot for visualization, the first 2 variables of p variables, and the first 2 corresponding directions of length 2 (not of length p, i.e not the full eigenvector). Will I get the correct 2 equation of the lines (orthogonal) that show the direction of most and 2nd most variability. The answer is no, if you have pxp cov matrix and take a smaller subset of that matrix say qxq the eigen vector of length q will be completely different to the eigen vector of length p.

Apply principal components analysis to the facial temperature data

```
pca1 = prcomp(numeric_samp_data)
pca1
```

```
## Standard deviations (1, .., p=11):
##  [1] 2.50056782 0.93252638 0.53898270 0.49345162 0.42174142 0.32516971
##  [7] 0.31706578 0.23778986 0.17635797 0.09022222 0.06940081
##
## Rotation (n x k) = (11 x 11):
##                          PC1         PC2          PC3         PC4           PC5
```

```
## Max_RCanthus_13IR 0.3010326 -0.2080242  0.033828242 -0.19577780  0.0746846460
## RCanthus_24x24IR  0.2923008 -0.2018746  0.014436197 -0.08692633  0.1919862094
## LCanthus_24x24IR  0.2737115 -0.1744617 -0.007091126  0.01450070  0.2741289004
## RCanthus_3x3IR    0.3085943 -0.1964599  0.018422268 -0.24401409  0.0797352035
## Max_CanthiIR      0.2803535 -0.2009980  0.008284719 -0.02254744  0.2478784613
## FHead_CentreIR    0.3572309  0.4513118 -0.529835878 -0.04915519  0.0151079557
## FHead_LeftIR      0.3094800  0.4037237 -0.352065351  0.04236166  0.0295206141
## FHead_TopIR       0.3288188  0.5471835  0.753384407  0.11562502 -0.0004897355
## MouthIR           0.2590980 -0.2275070 -0.094304426  0.76319712 -0.4724299253
## Avg_RCanthus_13IR 0.3390906 -0.1860218  0.095717263 -0.43803246 -0.6773497722
## Oral              0.2481236 -0.2325737  0.088711292  0.31926661  0.3626605929
##                          PC6        PC7          PC8           PC9
## Max_RCanthus_13IR  0.04433131  0.1258087 -0.066077993  0.4900401759
## RCanthus_24x24IR   0.07374654  0.1672535  0.017075621  0.5006180223
## LCanthus_24x24IR   0.17049062  0.1664916  0.535339145 -0.5525257543
## RCanthus_3x3IR     0.10600645  0.1734762 -0.750869700 -0.4352915846
## Max_CanthiIR       0.10073915  0.1546716  0.272945078  0.0488067801
## FHead_CentreIR    -0.56500533  0.2544285  0.026945316 -0.0252319707
## FHead_LeftIR       0.63888645 -0.4478933 -0.046026824  0.0600247691
## FHead_TopIR       -0.02493166  0.1041315 -0.006033768 -0.0008699059
## MouthIR            0.08922661  0.2268089 -0.079875749  0.0141278679
## Avg_RCanthus_13IR -0.13495131 -0.3226762  0.225079362 -0.0832662356
## Oral              -0.43352236 -0.6653675 -0.102606142 -0.0268961431
##                           PC10         PC11
## Max_RCanthus_13IR -0.7226306456  0.191207735
## RCanthus_24x24IR   0.4383764478 -0.593874923
## LCanthus_24x24IR  -0.2769493298 -0.306394792
## RCanthus_3x3IR     0.0725212919  0.007636699
## Max_CanthiIR       0.4381008179  0.718398109
## FHead_CentreIR    -0.0003462220  0.002355204
## FHead_LeftIR      -0.0034902814  0.009766150
## FHead_TopIR        0.0006803609  0.002896474
## MouthIR           -0.0232753326  0.005521663
## Avg_RCanthus_13IR  0.1014659551 -0.018852053
## Oral              -0.0298093673 -0.015001497
```

Plot the cumulative proportion of the variance explained by the principal components

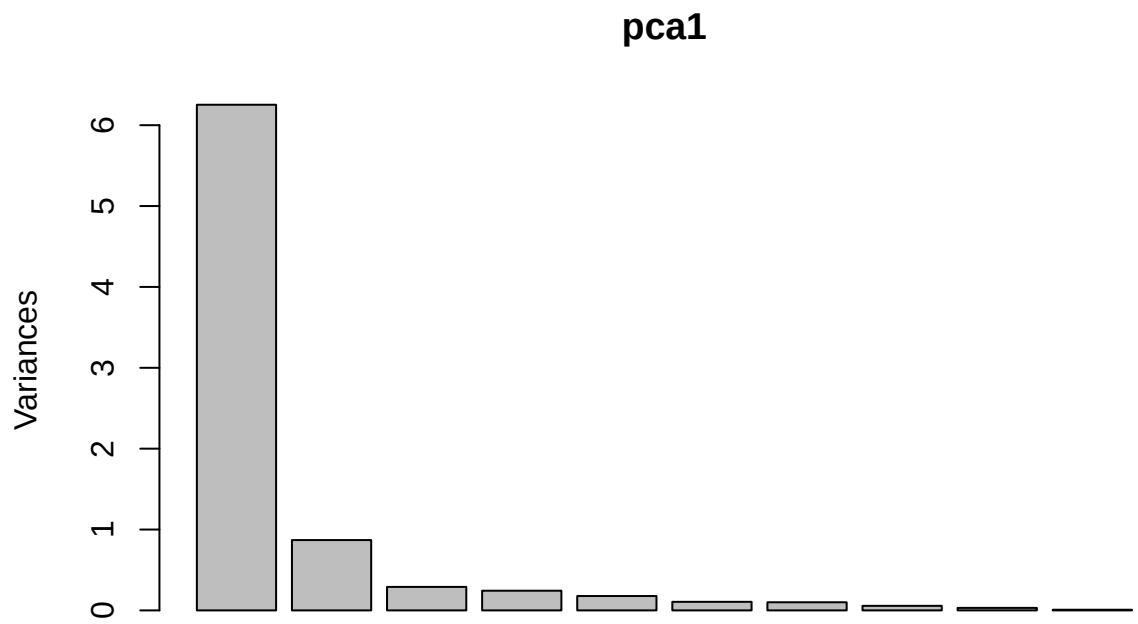
```
# Can't extract "Cumulative proportion" from this directly
tab1 = summary(pca1)
tab1
```

```
## Importance of components:
##                            PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.5006 0.9325 0.53898 0.49345 0.42174 0.32517 0.31707
## Proportion of Variance 0.7681 0.1068 0.03568 0.02991 0.02185 0.01299 0.01235
## Cumulative Proportion  0.7681 0.8749 0.91055 0.94046 0.96231 0.97529 0.98764
##                            PC8     PC9    PC10    PC11
## Standard deviation     0.23779 0.17636 0.09022 0.06940
## Proportion of Variance 0.00695 0.00382 0.00100 0.00059
## Cumulative Proportion  0.99459 0.99841 0.99941 1.00000
```

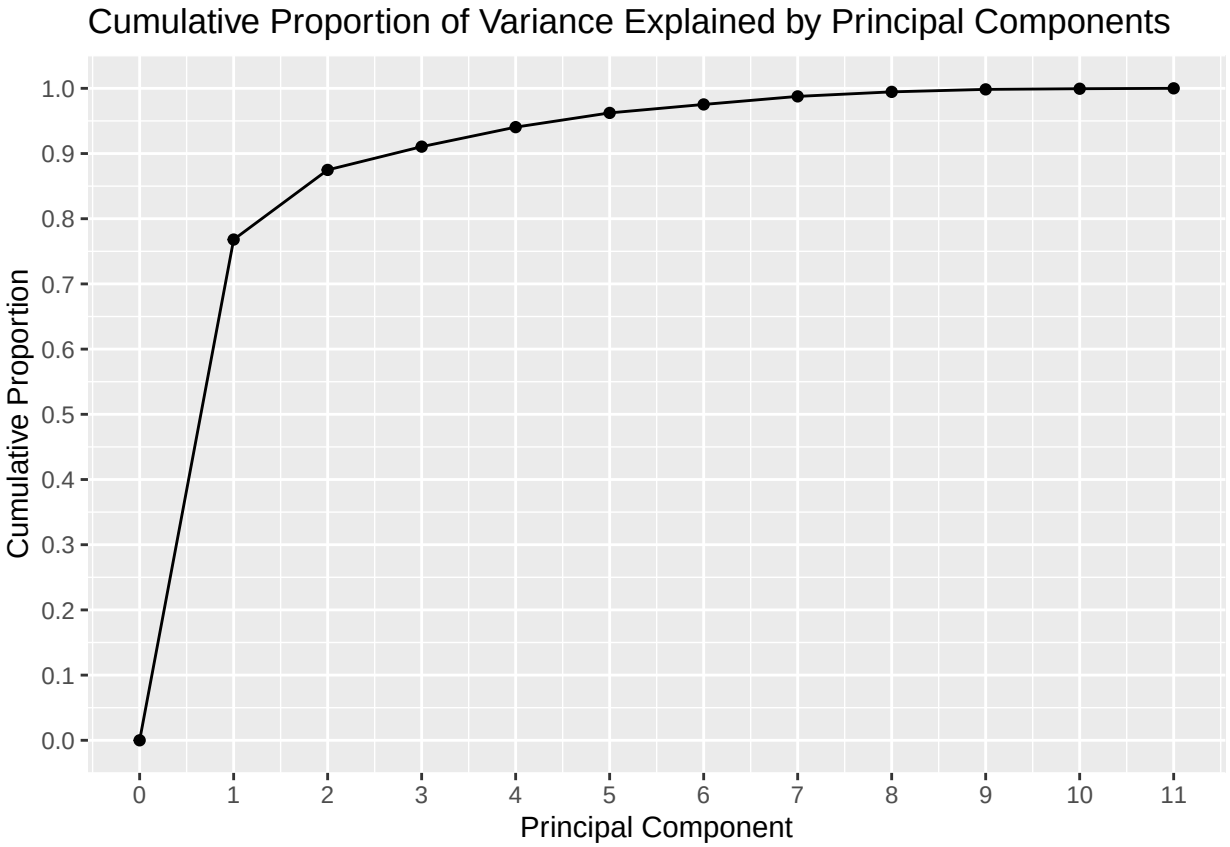
```r
# sqr root Eigenvalue = std, Eigenvalue = std^2
# jth Eigenvalue / sum of all Eigenvalues = proportion explained by j
eigenvalue = (pca1$sdev)^2
prop = (eigenvalue / sum(eigenvalue) )
cum_prop = cumsum(prop)
cum_prop
```

```
##  [1] 0.7680511 0.8748668 0.9105499 0.9404589 0.9623066 0.9752943 0.9876427
##  [8] 0.9945882 0.9984085 0.9994084 1.0000000
```

```r
# Without if, it will keep adding 0, when code run again
if (cum_prop[1] != 0){
  cum_prop = c(0, cum_prop)
}
plot(pca1)
```




```r
ggplot(data = NULL)+
  geom_line(aes(x = 0:(length(cum_prop)-1), y = cum_prop))+
  geom_point(aes(x = 0:(length(cum_prop)-1), y = cum_prop))+
  labs(x = "Principal Component", y = "Cumulative Proportion", title = "Cumulative Proportion of Varian
  scale_x_continuous(breaks = seq(0, length(cum_prop), by = 1))+
  scale_y_continuous(breaks = seq(0, length(cum_prop), by = 0.1))
```



How many principal components do you think are required to represent the data?

I think 2 is sufficient, 3 to be sure.

Principal Component Direction 1

```
pca1$rotation[, 1]
```

```
## Max_RCanthus_13IR  RCanthus_24x24IR  LCanthus_24x24IR    RCanthus_3x3IR
##         0.3010326         0.2923008         0.2737115         0.3085943
##      Max_CanthiIR    FHead_CentreIR      FHead_LeftIR       FHead_TopIR
##         0.2803535         0.3572309         0.3094800         0.3288188
##           MouthIR Avg_RCanthus_13IR              Oral
##         0.2590980         0.3390906         0.2481236
```

From above we can see that all measuring method move in the same direction. Interpretation of this: higher reading on these measurement devices = indicate a higher body temperature. This makes sense, if you can picture X and Y axis, where X can be 'Oral' and Y any of the other above, picture the most variability as a line from the bottom left of the graph to the top right. As the temperature of one measurement device increases so does temperature of the other devices generally.

Principal Component Direction 2

```
pca1$rotation[, 2]
```

```
## Max_RCanthus_13IR  RCanthus_24x24IR  LCanthus_24x24IR    RCanthus_3x3IR
##        -0.2080242        -0.2018746        -0.1744617        -0.1964599
##      Max_CanthiIR    FHead_CentreIR      FHead_LeftIR       FHead_TopIR
##        -0.2009980         0.4513118         0.4037237         0.5471835
```

```
##           MouthIR Avg_RCanthus_13IR              Oral
##        -0.2275070        -0.1860218        -0.2325737
```

Here it makes the distinction between the forehead measurement methods and the others. This means in some way the forehead measurements do not behave in the same way as the others. This is like more "fine tuning", to explain the 2nd most variation using all measurement methods, we need to take into account this behavior of the forehead measurements, as they act differently in comparison to the others.

I think these 2 components are sufficient, but I will highlight the 3rd one.

Principal Component Direction 3

```
pca1$rotation[, 3]
```

```
## Max_RCanthus_13IR  RCanthus_24x24IR  LCanthus_24x24IR    RCanthus_3x3IR
##       0.033828242       0.014436197      -0.007091126       0.018422268
##      Max_CanthiIR    FHead_CentreIR      FHead_LeftIR       FHead_TopIR
##       0.008284719      -0.529835878      -0.352065351       0.753384407
##           MouthIR Avg_RCanthus_13IR              Oral
##      -0.094304426       0.095717263       0.088711292
```

Again there is a distinction between forehead but also Mouth area and Left Canthus. Again this is further fine tuning or capturing/explaining more of the variance.

6.

Derive the principal component scores for each subject from first principles

```
# p x p covariance matrix of the predictor variables
cov_x = cov(numeric_samp_data)

# Yi = ai.Xi, Y is some linear transformation of X
# In the notes we maximise Var(aX) which is equal to aSa (I leave out transpose for easier to read)
# Where S is covariance matrix we found earlier

# To find the vector a from aSa
# We need to use Lagrange multiplier with the constraints

# Constraint 1: a transpose a = 1. Maintain unit vector scaling
# Constraint 2: Cov(a1X1, a2X2) = 0 i.e Cov(Y1, Y2) = 0
# The new principal component must be uncorrelated to ALL previous PCs.

# Find eigenvalues and eigenvectors
eigen_result = eigen(cov_x)


eigen_values = eigen_result$values

# These are the "loadings" (vectors: a1, a2, ... , ap)
# Or coefficients, we multiply this for each observation with the variable values
#
eigen_vectors = eigen_result$vectors
```

Calculate PC1 scores for the 99 observations. PC1 Loading = a1 = 1x11 vector = largest eigenvector

Need to center the data, the pca() functions do this automatically.

```r
means = colMeans(numeric_samp_data)

centered_numeric_samp_data = sweep(numeric_samp_data, 2, means, "-")

pc1_score = (eigen_vectors[, 1] %*% t(centered_numeric_samp_data))
pc1_score[1:10]
```

```
##  [1] -0.4098517  1.2100921 -1.0478964  2.6441206  1.5285796  2.9993609
##  [7] -1.0076839  4.2982223  1.2113791  0.8391702
```

The signs are arbitrary. As you can see we have a match of PC scores between the ones I calculated and the ones calculated by the predict() function.

```r
# For comparison
true_scores = predict(pca1)
true_score1 = true_scores[, 1]
true_score1[1:10]
```

```
##        804         80        793        656        182         82        245
##  0.4098517 -1.2100921  1.0478964 -2.6441206 -1.5285796 -2.9993609  1.0076839
##        263        460        524
## -4.2982223 -1.2113791 -0.8391702
```

1st column is scores for each observation for PC1, 2nd column PC2, ... note: Compare signs

```r
pc_scores = t(eigen_vectors) %*% t(centered_numeric_samp_data)
pc_scores = t(pc_scores)
pc_scores[1:10, ]
```

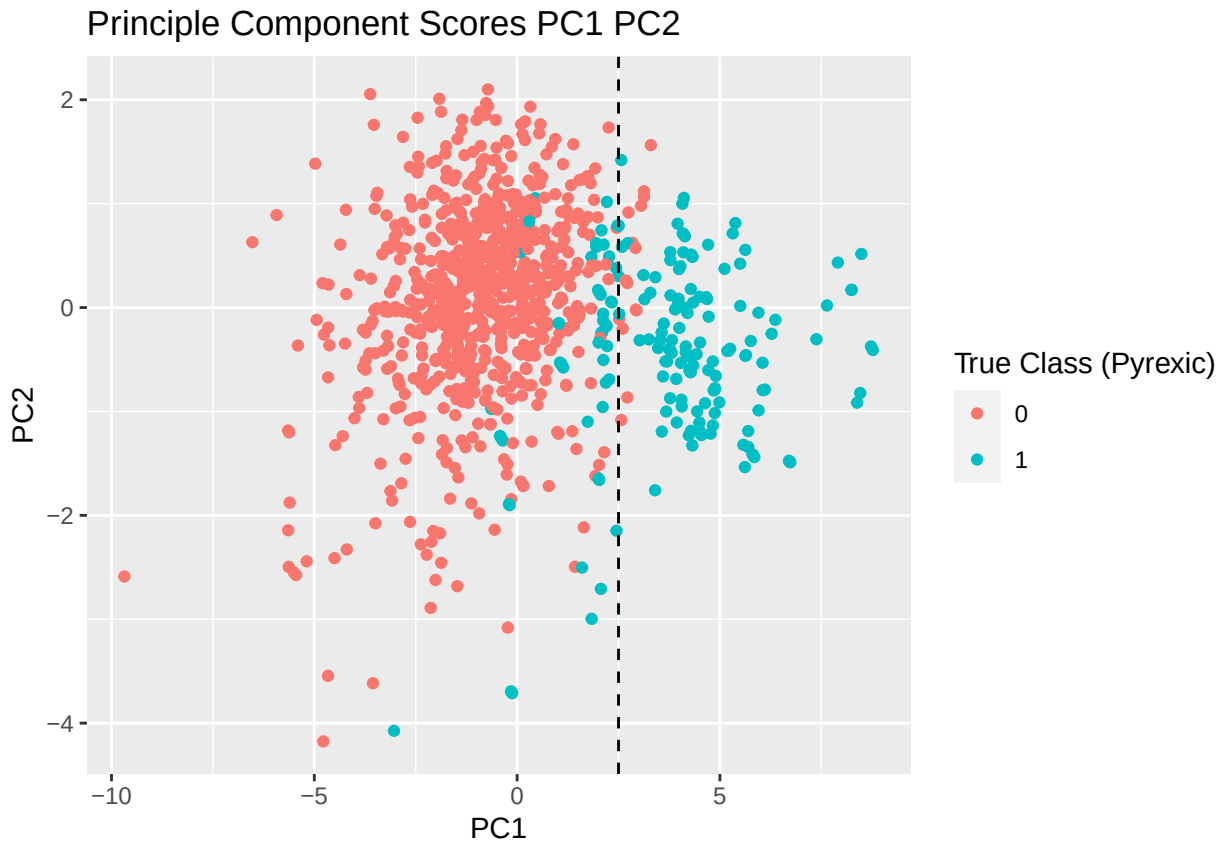
```
##           [,1]         [,2]        [,3]         [,4]         [,5]        [,6]
## 804 -0.4098517  0.726496211 -0.32070881 -0.088149462 -0.235698188 -0.04154259
## 80   1.2100921 -0.101101211  0.25521464 -0.053553928  0.007630297  0.03597967
## 793 -1.0478964 -0.249676903  0.28539164 -0.308427729  0.486478033 -0.14640945
## 656  2.6441206  1.354115379 -0.39551474 -0.246296697 -0.632764661  0.64042751
## 182  1.5285796  0.396920697  0.33403590 -0.195450139 -0.042882444  0.06721496
## 82   2.9993609  0.004331893 -0.40774072 -0.021730807 -0.008303980  0.12320995
## 245 -1.0076839  0.311186642  0.10121550  0.069546700  0.484501509 -0.22327618
## 263  4.2982223 -1.237682219  0.04123488 -0.969667930 -0.916267065 -0.23329212
## 460  1.2113791  0.582771369 -0.05037564 -0.008688355  0.497910854 -0.10808628
## 524  0.8391702  0.132979596  0.48269814  0.111739164  0.710754610  0.32011663
##           [,7]        [,8]        [,9]        [,10]        [,11]
## 804 -0.08939078  0.11562322  0.29455046 -0.045514652 -0.095876273
## 80   0.01368241 -0.08692180  0.17117482 -0.012004413  0.043397326
## 793  0.28338934 -0.35906986 -0.12110157 -0.084553814 -0.079018882
## 656  0.31692964 -0.36177380 -0.08638863 -0.001360386  0.004511752
## 182 -0.17843294  0.14631161  0.09320408 -0.037698814  0.061275570
## 82   0.21426162  0.32088127  0.11739616  0.061364262 -0.083348124
## 245  0.05475673  0.06025240 -0.01266915  0.078818117  0.041252078
## 263 -0.15438344  0.01278203  0.01949305 -0.128108008  0.109089138
## 460  0.05689045 -0.04016300  0.04015002 -0.003917530 -0.006826073
## 524  0.13096879 -0.06500995  0.17797950  0.028209503  0.055028534
```

TEST calculating eigen vectors for first 2 variables only. For earlier question.

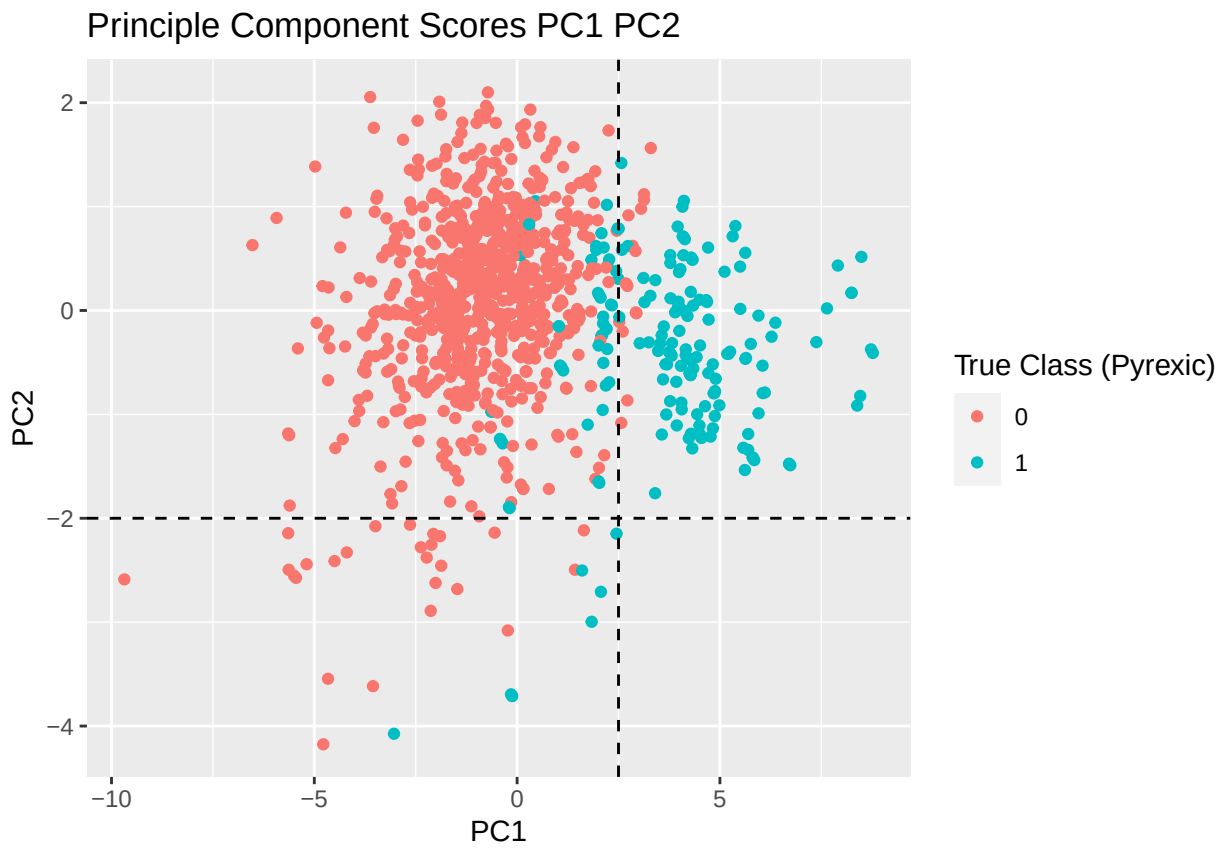
```r
test_cov = cov_x[1:2, 1:2]
test_eigen = eigen(test_cov)
test_eigenvectors = test_eigen$vectors
```

Plot the principal component scores for the subjects

```
ggplot(data = NULL, aes(x = pca1$x[, 1], y = pca1$x[, 2], color=factor(samp_data$pyrexic))) +
  geom_point()+
  labs(title = "Principle Component Scores PC1 PC2", x="PC1", y="PC2") +
  geom_vline(xintercept = 2.5, linetype = "dashed", color = "black")+
  guides(colour = guide_legend(title = "True Class (Pyrexic)"))
```



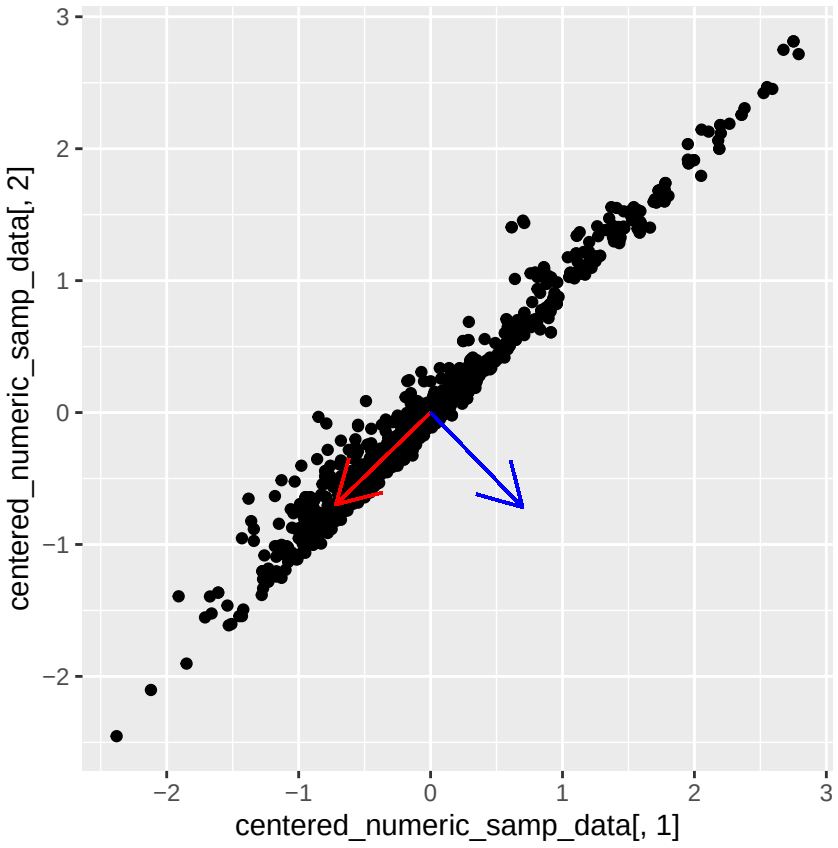

```
ggplot(data = NULL, aes(x = pca1$x[, 1], y = pca1$x[, 2], color=factor(samp_data$pyrexic))) +
  geom_point()+
  labs(title = "Principle Component Scores PC1 PC2", x="PC1", y="PC2") +
  geom_vline(xintercept = 2.5, linetype = "dashed", color = "black")+
  guides(colour = guide_legend(title = "True Class (Pyrexic)"))+
  geom_hline(yintercept = -2, linetype = "dashed", color = "black")
```



Comment on any structure you observe. We can clearly see here PC1 is all you really need to be able to identify those with high temperature or classed as pyrexic. Past the dotted line which is a PC1 score of 2.5 you can be almost guaranteed the observation is pyrexic. Again this makes sense as we discussed earlier all of the coefficients for the PC1 is positive and is a general measure of temperature from all the measuring methods. PC2 doesn't really add much in terms of classifying those who are pyrexic, as those with pyrexic = 1 have the same PC2 scores as pyrexic = 0 In the second graph I have added a horizontal line to demonstrate how if this was different data we could try to class those in the bottom left section as seperate from the other two. To conclude PC1 does a good job of predicting those as pyrexic.

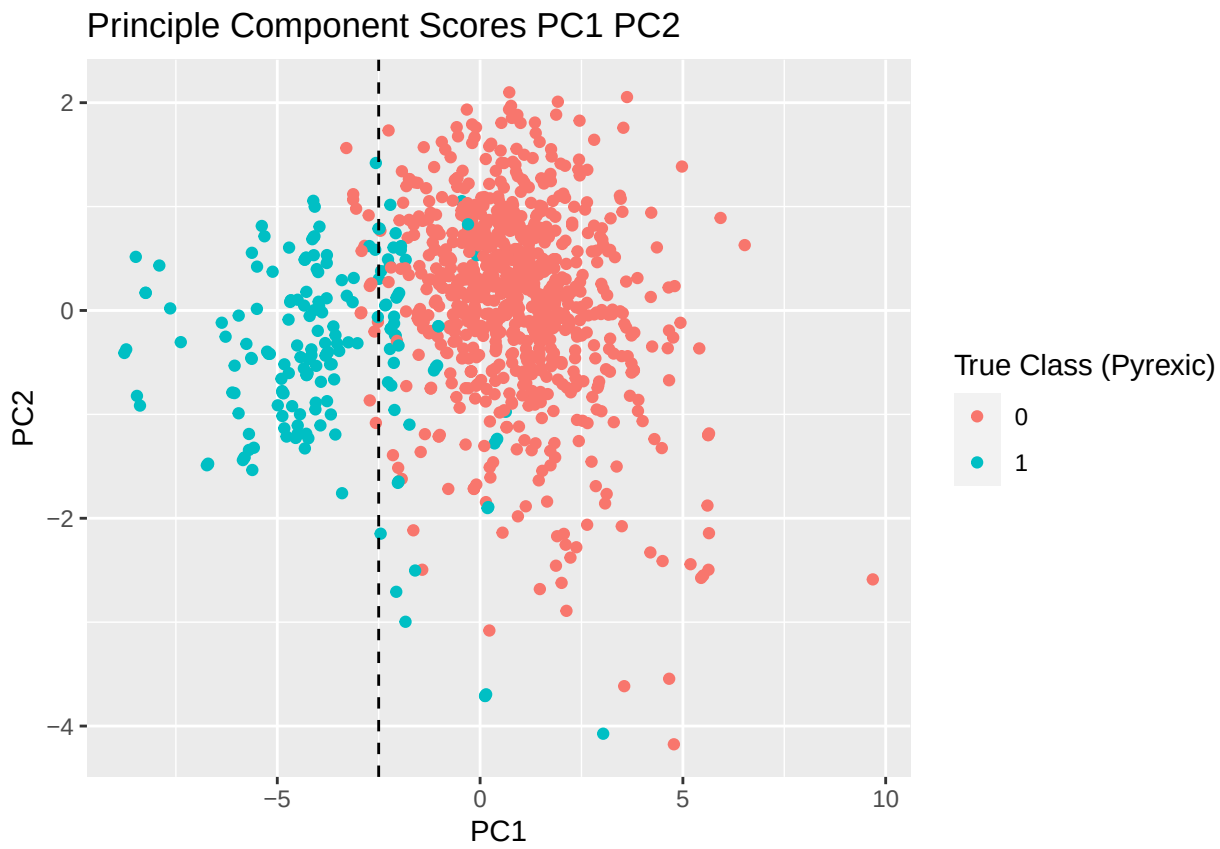
This graph below demonstrates how the first eigen vector (Principal component direction) captures most of the variation (from bottom left to upper right of the data, upward sloping). The 2nd eigen vector captures the variation perpendicular to that direction. Now in 2d space there are only 2 perpendicular eigen vectors, but as the dimensions increase so do the amount of orthogonal eigen vectors. TEST calculating eigen vectors for first 2 variables only. For earlier question.

```
test_cov = cov_x[1:2, 1:2]
test_eigen = eigen(test_cov)
test_eigenvectors = test_eigen$vectors

ggplot(data = samp_data, aes(x = centered_numeric_samp_data[, 1], y = centered_numeric_samp_data[, 2]))+
  geom_point()+
  geom_segment(x = 0, y = 0, xend = test_eigenvectors[1,1], yend = test_eigenvectors[2,1], color = "red
  geom_segment(x = 0, y = 0, xend = test_eigenvectors[1,2], yend = test_eigenvectors[2,2], color = "blu
  coord_fixed(ratio = 1) # Wrong scale was really putting me off
```



This is to highlight how signs in the PCs can be arbitrary. This graph is using PCs computed from first principles, while the previous scores are extracted from the prcomp() function.

```
ggplot(data = NULL, aes(x = pc_scores[, 1], y = pc_scores[, 2], color=factor(samp_data$pyrexic))) +
  geom_point()+
  labs(title = "Principle Component Scores PC1 PC2", x="PC1", y="PC2") +
  geom_vline(xintercept = -2.5, linetype = "dashed", color = "black")+
  guides(colour = guide_legend(title = "True Class (Pyrexic)"))
```



Clean up

```
remove(cov_x, eigen_result, eigen_vectors, pc_scores, pc1_score, pca1, tab1, test_cov, test_eigen, true
```

7.

Principal Components Regression (PCR)

In your own words, write a maximum 1 page synopsis of the PCR method.

Notes: " situations where p is large relative to n, selecting a value of M « p can significantly reduce the variance of the fitted coefficients"

If we fit regression using dimension reduced least squares we can get better results than least squares on the original variables. Due to high correlation, can result in overfitting? Predictions are better when model not over fitted.

PCR is unsupervised Consequently, PCR suffers from a drawback: there is no guarantee that the directions that best explain the predictors will also be the best directions to use for predicting the response

PLS approach attempts to find directions that help explain both the response and the predictors

The PLS direction does not fit the predictors as closely as does PCA, but it does a better job explaining the response

To identify the second PLS direction we first adjust each of the variables for Z1, by regressing each variable on Z1 and taking residuals. These residuals can be interpreted as the remaining information that has not been explained by the first PLS direction

The residuals are orthogonal to Z1 because the regression process removes the shared variance with Z1.

Then we find the linear combination of the orthogonalized predictor variables that maximizes the covariance with the response variable (Y). This ensures that Z2 captures additional unique information in the predictor variables that is not already explained by Z1.

As with PCR, the number M of partial least squares directions used in PLS is a tuning parameter that is typically chosen by cross-validation.

We generally standardize the predictors and response before performing PLS.

While the supervised dimension reduction of PLS can reduce bias, it also has the potential to increase variance

danger of overfitting when using linear OLs for high dimensions

Using least squares regression,logistic regression, linear discriminant analysis, and other classical statistical approaches. When p >= n strictly results in: The reason is simple: regardless of whether or not there truly is a relationship between the features and the response, least squares will yield a set of coefficient estimates that result in a perfect fit to the data, such that the residuals are zero.

though it is possible to perfectly fit the training data in the high-dimensional setting, the resulting linear model will perform extremely poorly on an independent test set, and therefore does not constitute a useful model.

Explain the method's purpose

We want to predict a response variable Y using the Principal Components, i.e you get the PC scores for each variable for the selected PCs, input it into your model and then hopefully you are able to get an accurate prediction value.

provide a general description of how the method works

You do a normal Principal Component Analysis on the data. I.e find a new set of uncorrelated predictor variables which is a linear transformation of the original data. (With the constraints) . You then select these new Principal Components and use them in least squares regression on a response variable.

detail any choices that need to be made when using the method

Choosing the number of PCs as predictors variables. In PCR we assume the directions in which our data shows the most variation are the directions that are associated with our Response variable Y in mind. This assumption is not guaranteed to be true, but it still gives reasonable predictions most of the time. We can use Partial Least Squares to take this into account and ensure the directions are identified in a supervised way, i.e the directions are good at predicting the response.

When the number of predictor variables, p >= n (number of observations) strictly results in perfect fit (overfitting). Look at example of 2 variables with 2 observations, it is mapped perfectly, overfitting, it will not perform good on the new, separate, test data.

outline the advantages and disadvantages of the method

Advantages: Can perform better than normal least squares using all predictor variables (all dimensions). Normal least squares can have high correlation between the variables. Using all predictors can result in overfitting. Predictions are better when model not over fitted. PCR reduces dimensions, less predictor variables needed to work with. PCR ensure the predictors are uncorrelated. PCR can make it easier to understand the relationships between the predictors and the response variable.

Disadvantage: In PCR we assume the directions in which our data shows the most variation are the directions that are associated with our Response variable Y in mind. This assumption is not guaranteed to be true. We can use PLS instead for this. Although contradictory, to my previous advantage stated, sometimes the PCs can be quite hard to interpret especially the ones past first few. PCR can still result in overfitting if too many PCs are selected, bad at predicting new test data.

8.

Divide your data into training and test sets

For training and test allocation, we should perform this multiple times to ensure randomness is captured. E.g Split data intp 70% 30% randomly, train/test this data then repeat the process multiple times. Can use k folds, or leave one out validation. In the code I have kept it simple.

```
# Training
n = nrow(samp_data)
training_indices = sample(1:n, round(n*0.7))
training_data = samp_data[training_indices, ]

# Test
test_data = samp_data[-training_indices,]
```

use the function pcr in the pls R package to perform PCR on the training data

We now apply PCR to the sample data, in order to predict Oral temperature.

validation="CV" causes pcr() to compute the ten-fold cross-validation error for each possible value of M, the number of principal components used.

The CV outputs is the root mean squared error, square this quantity to get MSE.

```
library(pls)
```

```
## Warning: package 'pls' was built under R version 4.3.3
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

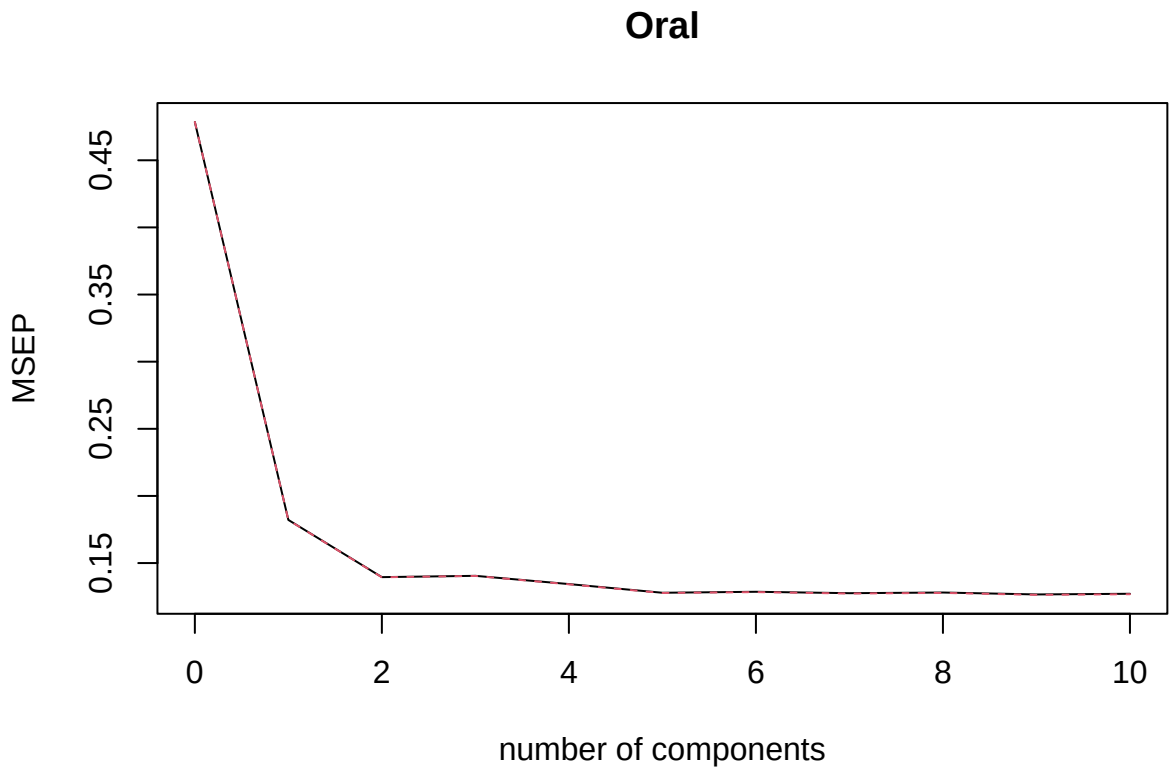
```
# it should be centered by default anyways
pcr1 = pcr(Oral ~ . , data = training_data[, -c(12:15)], validation= "CV")
summary(pcr1)
```

```
## Data:    X dimension: 699 10
##  Y dimension: 699 1
## Fit method: svdpc
## Number of components considered: 10
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV          0.6918   0.4268   0.3735   0.3748   0.3665   0.3576   0.3586
## adjCV       0.6918   0.4266   0.3734   0.3747   0.3663   0.3574   0.3583
##        7 comps  8 comps  9 comps  10 comps
## CV      0.3572   0.3579   0.3558    0.3565
## adjCV   0.3568   0.3575   0.3554    0.3561
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       76.68    88.14    91.82    94.99    97.20    98.59    99.39    99.82
## Oral    62.32    71.18    71.20    72.52    73.94    73.99    74.30    74.30
```

```
##       9 comps  10 comps
## X       99.93    100.00
## Oral    74.67     74.69
```

This plots the MSE for each number of PC components included in our model pcr1. The number of new predictor variables in the regression. This confirms our comments earlier that 2 PCs should be enough, even the 1st one already has a low. Past 2 we see marginal benefit in MSE, it appears that MSE even slightly increases for 3 PCs.

```
validationplot(pcr1, val.type = "MSEP")
```



As a general rule R^2 increases and training MSE decreases as the number of variables increase, while the test MSE increases with increasing number of variables due to overfitting.

Use your fitted model to predict the Oral Temperature in the test set

Our MSE is low for the test data, use previous values in training MSE for relativity.

```
pcr_pred = predict(pcr1, test_data[, -c(12:15)], ncomp = 2)

# mean squared differences between actual and predicted values
mse_test = mean((test_data$Oral - pcr_pred)^2)
mse_test
```

```
## [1] 0.1670022
```

```
ggplot(data = NULL, aes(x = pcr_pred, y = test_data$Oral))+
  geom_point() +
  geom_abline(intercept = 0, slope = 1, color = "red", linetype = "dashed") +
```

```
labs(x = "Predicted Values", y = "True Values", title = "True vs. Predicted Values of Oral")
```