Data Visualisation Assignment 1

Bryan Mannix-

21129786

```
install.packages("tidyverse")
install.packages("ggpubr")
library(tidyverse)
install.packages ("dplyr")
library(dplyr)
library(ggplot2)
install.packages("FactoMineR")
install.packages ("factoextra")
library(readr)
library("FactoMineR")
library(factoextra)
install.packages('tinytex')
tinytex::install_tinytex()
install.packages("colorspace")
library("colorspace")
install.packages("shiny")
install.packages("shinyjs")
install.packages("colorspace")
library(cowplot)
library(colorspace)
library(colorblindr)
library(RColorBrewer)
library(scales)
```

```
install.packages("remotes")

remotes::install_github("wilkelab/cowplot")

remotes::install_github("clauswilke/colorblindr")

pen<-read_csv(file = "C:/Users/bryan/Documents/DataVisualisationAssignment1/p
endigitss.csv")

## New names:
## * `0` -> `0...10`
## * `0` -> `0...11`
## * `100` -> `100...15`
## * `100` -> `100...16`

## Rows: 10991 Columns: 17

## -- Column specification ----------------------------------------------------
------
## Delimiter: ","
## dbl (17): 88, 92, 2, 99, 16, 66, 94, 37, 70, 0...10, 0...11, 24, 42, 65, 1
00...

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this m
essage.

pen

## # A tibble: 10,991 x 17
##     `88`  `92`   `2`  `99`  `16`  `66`  `94`  `37`  `70` `0...10` `0...11`
`24`
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
<dbl>
##  1    80   100    18    98    60    66   100    29    42        0        0
23
##  2     0    94     9    57    20    19     7     0    20       36       70
68
##  3    95    82    71   100    27    77    77    73   100       80       93
42
##  4    68   100     6    88    47    75    87    82    85       56      100
29
##  5    70   100   100    97    70    81    45    65    30       49       20
33
##  6    40   100     0    81    15    58   100    57    47       87       50
88
##  7     3    71     0    95    45   100   100    99    79       78       48
53
##  8    79    87    98    81    71   100    72    73   100       66       91
21
##  9    92    95    30   100    34    68    87    89    84       78      100
```

```
35
## 10    58    64   100    96    27   100     0    63    79       65       91
72
## # ... with 10,981 more rows, and 5 more variables: `42` <dbl>, `65` <dbl>,
## #   `100...15` <dbl>, `100...16` <dbl>, `8` <dbl>
```

identifying what the class column name is which is 8, then setting that column as a factor.

```
summary(pen)
```

```
##        88               92               2                99        
##  Min.   :  0.00   Min.   :  0.00   Min.   :  0.00   Min.   :  0.00  
##  1st Qu.:  6.00   1st Qu.: 76.00   1st Qu.: 20.00   1st Qu.: 72.00  
##  Median : 32.00   Median : 89.00   Median : 40.00   Median : 91.00  
##  Mean   : 38.81   Mean   : 85.12   Mean   : 40.61   Mean   : 83.77  
##  3rd Qu.: 65.00   3rd Qu.:100.00   3rd Qu.: 58.00   3rd Qu.:100.00  
##  Max.   :100.00   Max.   :100.00   Max.   :100.00   Max.   :100.00  
##        16               66               94               37       
##  Min.   :  0.00   Min.   :  0.00   Min.   :  0.00   Min.   :  0.0  
##  1st Qu.: 18.00   1st Qu.: 49.00   1st Qu.: 28.00   1st Qu.: 23.0  
##  Median : 53.00   Median : 71.00   Median : 53.00   Median : 43.0  
##  Mean   : 49.77   Mean   : 65.57   Mean   : 51.22   Mean   : 44.5  
##  3rd Qu.: 78.00   3rd Qu.: 86.00   3rd Qu.: 74.00   3rd Qu.: 64.0  
##  Max.   :100.00   Max.   :100.00   Max.   :100.00   Max.   :100.0  
##        70             0...10          0...11             24        
##  Min.   :  0.00   Min.   :  0.0   Min.   :  0.00   Min.   :  0.00  
##  1st Qu.: 29.00   1st Qu.:  7.0   1st Qu.: 23.00   1st Qu.: 11.00  
##  Median : 60.00   Median : 33.0   Median : 73.00   Median : 30.00  
##  Mean   : 56.87   Mean   : 33.7   Mean   : 60.52   Mean   : 34.83  
##  3rd Qu.: 89.00   3rd Qu.: 54.0   3rd Qu.: 97.00   3rd Qu.: 55.00  
##  Max.   :100.00   Max.   :100.0   Max.   :100.00   Max.   :100.00  
##        42               65            100...15         100...16     
##  Min.   :  0.00   Min.   :  0.00   Min.   :  0.00   Min.   :  0.00  
##  1st Qu.: 42.00   1st Qu.:  5.00   1st Qu.:  0.00   1st Qu.:  0.00  
##  Median : 53.00   Median : 27.00   Median : 40.00   Median :  9.00  
##  Mean   : 55.02   Mean   : 34.93   Mean   : 47.28   Mean   : 28.84  
##  3rd Qu.: 68.00   3rd Qu.: 47.00   3rd Qu.:100.00   3rd Qu.: 51.00  
##  Max.   :100.00   Max.   :100.00   Max.   :100.00   Max.   :100.00  
##        8        
##  Min.   :0.000  
##  1st Qu.:2.000  
##  Median :4.000  
##  Mean   :4.431  
##  3rd Qu.:7.000  
##  Max.   :9.000
```

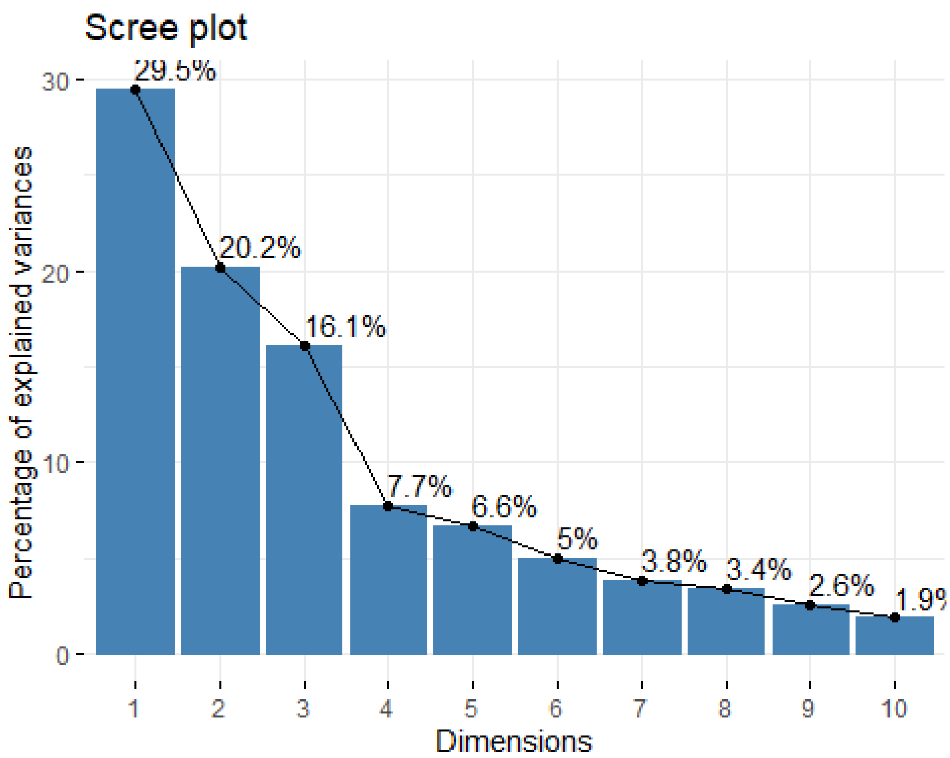
```
pen$'8' <- as.factor(pen$'8')
```

excluding class value as it will not help discriminating between the different signiture features.

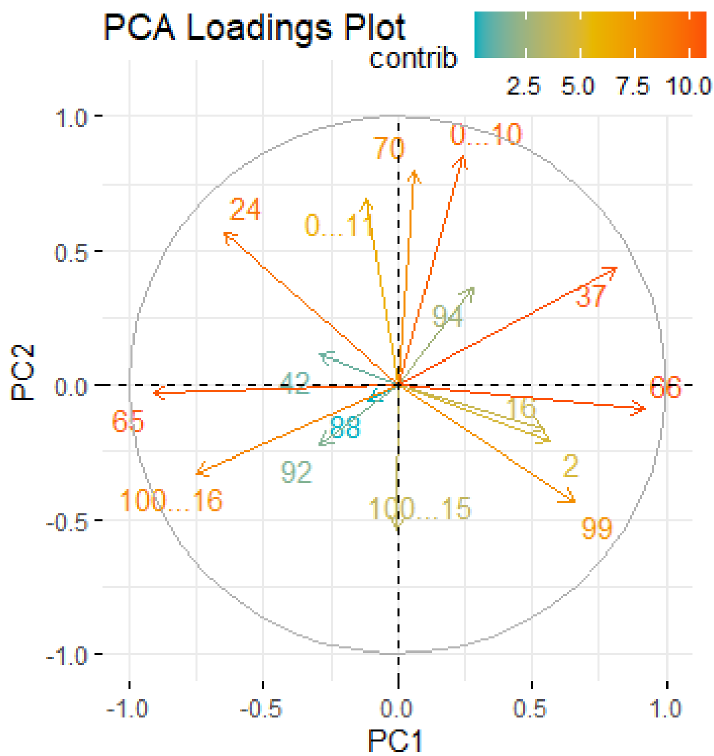
```
dplyr::select(pen,-'8') %>%
  PCA(graph = FALSE) -> ppa

ppa

## **Results for the Principal Component Analysis (PCA)**
## The analysis was performed on 10991 individuals, described by 16 variables
## *The results are available in the following objects:
##
##    name               description
## 1  "$eig"             "eigenvalues"
## 2  "$var"             "results for the variables"
## 3  "$var$coord"       "coord. for the variables"
## 4  "$var$cor"         "correlations variables - dimensions"
## 5  "$var$cos2"        "cos2 for the variables"
## 6  "$var$contrib"     "contributions of the variables"
## 7  "$ind"             "results for the individuals"
## 8  "$ind$coord"       "coord. for the individuals"
## 9  "$ind$cos2"        "cos2 for the individuals"
## 10 "$ind$contrib"     "contributions of the individuals"
## 11 "$call"            "summary statistics"
## 12 "$call$centre"     "mean of the variables"
## 13 "$call$ecart.type" "standard error of the variables"
## 14 "$call$row.w"      "weights for the individuals"
## 15 "$call$col.w"      "weights for the variables"
```

from the screeplot , we can see The first two dimensions have almost 50% of the of the variance in the data set is explained by the first two principal components This means that a 2D scatter plot of these two components should show good separation of the points. If the percentage of variance explained by the first two principal components is low, then the scatterplot produced will not illustrate groupings in the data.

```
fviz_screeplot(ppa, choice="variance", addlabels = TRUE,)
```




```
ppaloadings_plot <- fviz_pca_var(ppa,
                                 col.var = "contrib",
                                 gradient.cols = c("#00AFBB", "#E7B800", "#FC4E0
7"),
                                 repel = TRUE
) +
  xlab("PC1") +
  ylab("PC2") +


ylim(c(-1,1.1))+
  ggtitle("PCA Loadings Plot") +

  theme(
    legend.box.background = element_rect(fill = "white", color = "white"),
    legend.position = c(0.74, 1.01),
    legend.direction = "horizontal",
    legend.box.margin = margin(0.05, 0.05, 0.05, 0.05),
    legend.key = element_rect(fill = "white"),
  )
ppaloadings_plot
```



from observation, it looks as if both 37 and 0..10 components of the data set have a big influence on the principal comononts with positive correlation.

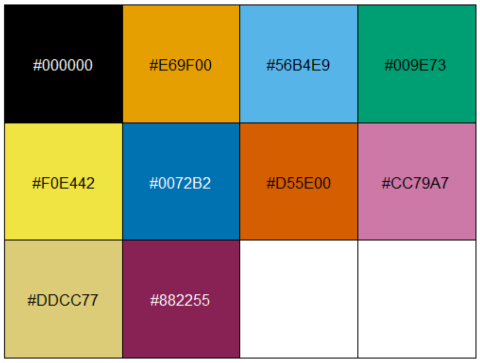
```
coloursorg1=c('#000000','#E69F00','#56B4E9','#009E73','#F0E442','#0072B2',
              '#D55E00','#CC79A7','#DDCC77','#882255')
```

show_col(coloursorg1)

imported colours from colourizer.org. used hcl colour picker r to adjust some of the colours. When choosing these colours, i tried not to use both red and green together because it is documented that red and green is one of the main combinations of colour that people with colour blindness have difficulty with.

I used a combination of light colours light blue, light yellow, pink, orange, light brown, and a combination of dark colours such as black, purple etc. also keeping in mind that colour blind people don't have an issue with telling the difference between light and dark colours. The main way of generating distinctiveness between colours is through their hues.

Additional variation can be obtained by adjusting lightness and saturation, but it’s a good idea not to make the differences too large. Too much difference might suggest that some colours are more important than others. so i chose some colour and dimmed the hue for example light blue and dark blue, light orange dark orange, dark purple, light purple, light yellow, dark yellow etc. it’s clear that the difference in hue is not significant therefore colours won’t seem more important than others.



Getting the PCA values of the data

#used pca_ind to get a projection of the individual data pointsin terms of the principal com ponents

```
data_pca_ind <- get_pca_ind(ppa)
```

I am interested in the first two principal components. # The first two columns of the mat rix. We select these here

```
head(data_pca_ind$coord)

data_pca <- data_pca_ind$coord[,c(1,2)]
```

For use in ggplot2, I need to convert the matrix to a dataframe

```
data_pca <- as.data.frame(data_pca)
```

rename the columns

```
names(data_pca)[1] <- "PC1"

names(data_pca)[2] <- "PC2"
```

Add the “8” labels to the data. # I will want to colour the data points by the “8”s label

```
data_pca<- cbind(data_pca, pen$'8')

#rennamed to Type

names(data_pca)[3] <- "Type"

showing first 6 rows of data to be plotted.

head(data_pca)

PC1         PC2 Type
```

```
1 -0.9415664 -2.08301854    8
2 -4.5519681 -0.01279134    8
3  1.6306798  2.36074349    9
4  0.9521783  2.14116957    9
5  2.2261453 -0.71289452    1
6 -0.6628515  2.12081660    4
```

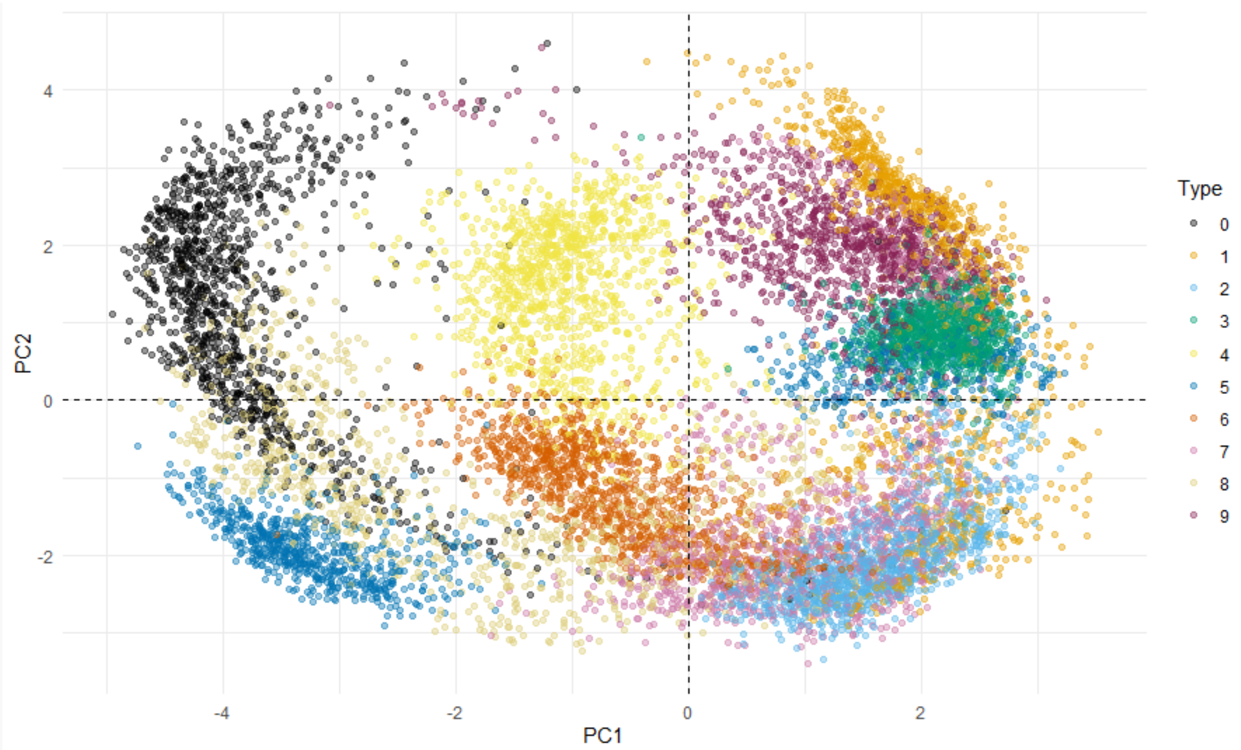
Plotting with ggplot2

I use geom_vline and geom_hline geom types to add vertical and horizontal dotted line thro ugh the origin (the 0 point). This will help us to compare this plot to the loadings plot so tha t we understand which features are discriminatory for each Type value. I use scale color ma nual and enter in the manually made colour palette hex values form earlier.

```
par(mar=c(1,3,1,1))
 ggplot(data_pca, aes(x=PC1, y=PC2, color=Type)) +
  geom_point(size=1.5, alpha = 0.4) +
  scale_color_manual(values= c('#000000','#E69F00','#56B4E9','#009E73','#F0E4
42','#0072B2'
                               '#D55E00','#CC79A7','#DDCC77','#882255')) +
  # I add vertical and horizontal axis lines
  geom_vline(xintercept = 0, linetype="dashed") +
  geom_hline(yintercept = 0, linetype="dashed") +
  theme_minimal()
```

#Here is the scatter plot with horizontal and vertical lines and with no

Ellipse.

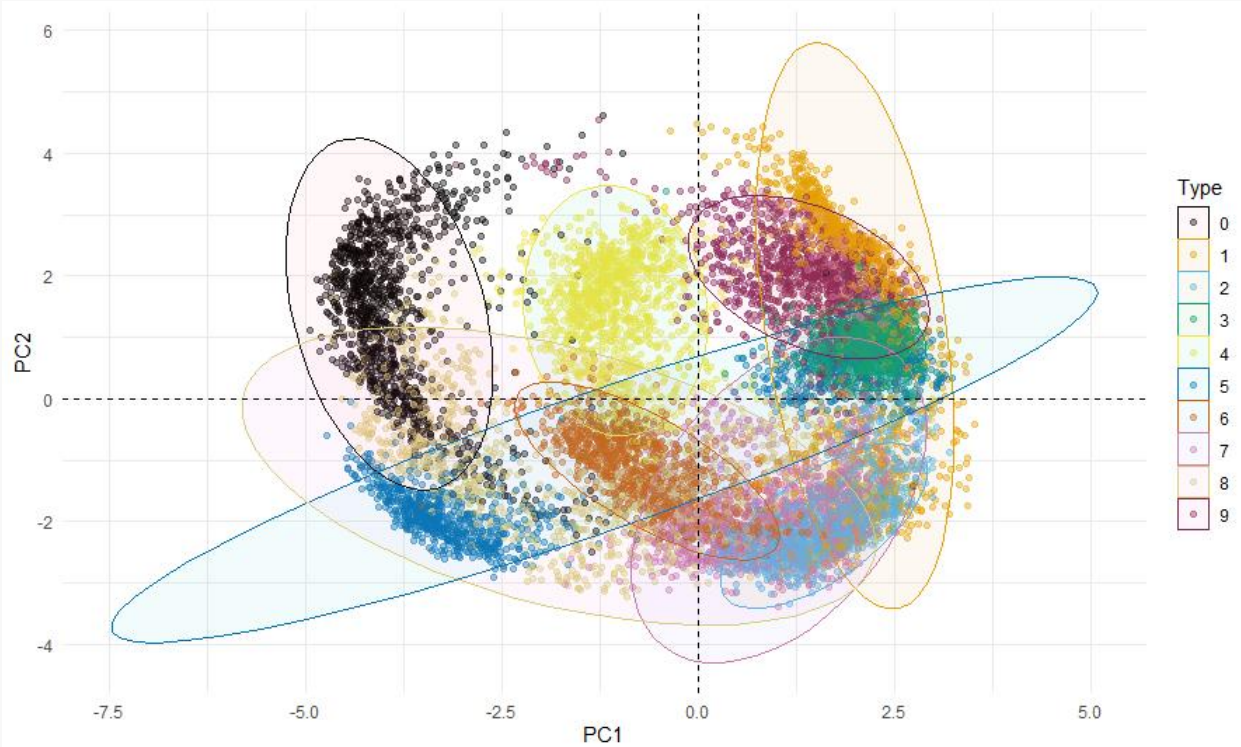


Fitting an Ellipse around groups of points

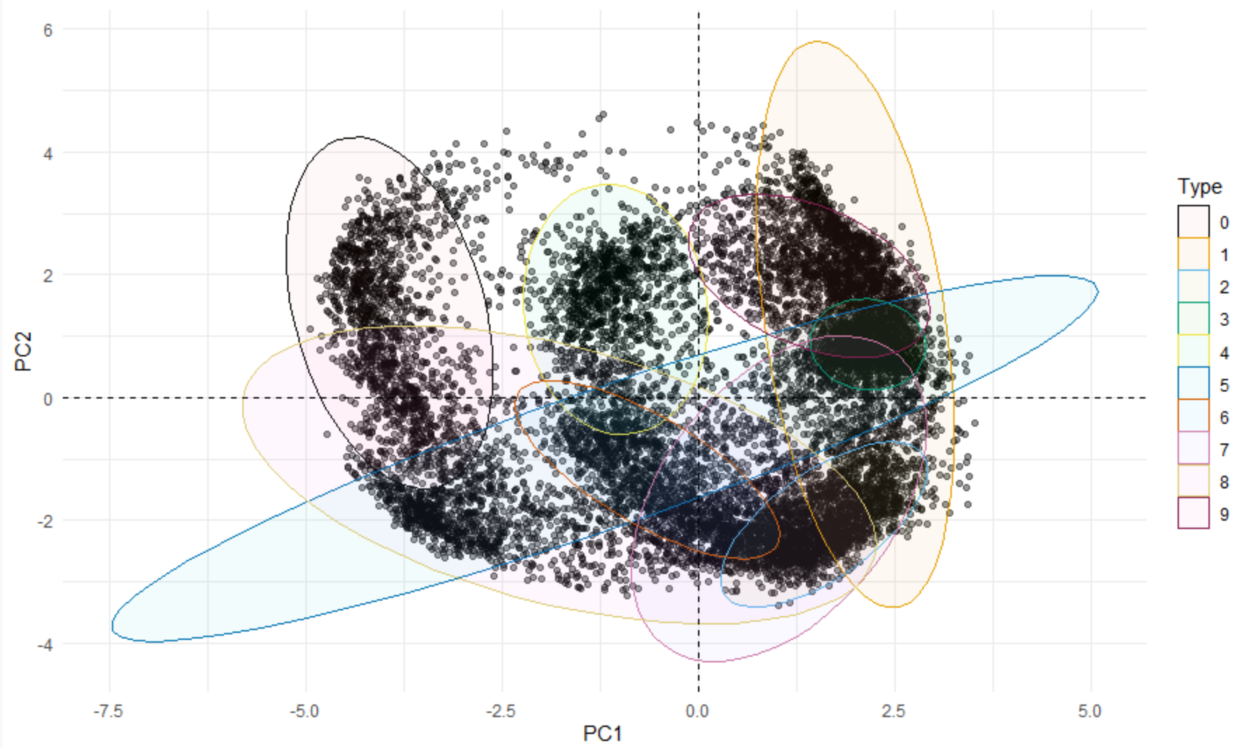
A useful way to visualise the groups is to fit an ellipse around each. Here is the code:

```
par(mar=c(1,3,1,1))

 ggplot(data_pca,aes(x=PC1, y=PC2, color=Type)) +

   geom_point(size=1.5, alpha = 0.4) +

   # override the default colour allocation with a manual colour palette

   scale_color_manual(values= c('#000000','#E69F00','#56B4E9','#009E73','#F0E442','#0072B2',

                                '#D55E00','#CC79A7','#DDCC77','#882255')) +

   # plot an ellipse that encompasses the points belong to each Type value

   stat_ellipse(geom = "polygon",type = "t",size = 0.2,

                aes(fill = Type),

                alpha = 0.05) +

   # the ellipse has a fill colour according to each Type value

   # I override the default fill allocation with a manual palette
```

```
  scale_color_manual(values= c('#000000','#E69F00','#56B4E9','#009E73','#F0E
442','#0072B2',

                                '#D55E00','#CC79A7','#DDCC77','#882255')) +

  # we add vertical and horizontal axis lines manually.

  geom_vline(xintercept = 0, linetype="dashed") +


  geom_hline(yintercept = 0, linetype="dashed") +

  theme_minimal()
```



For Visual purposes I will keep the ellipse for the remainder of the analysis because there is a lot of plots overlapping and some colours like light brown beige color are harder to spot without the ellipse surrounding it.



I will also keep on to the colour going forward as this makes it extremely difficult to identif y each class value by simply going off the elipse as many of the elipse are overlapping whic h makes it hard to spot some elipses., it is therefore better to have the colours along with t he elipses.

Using Labels instead of a Legend

Having to consult the legend makes it difficult to perceive the Type value associated with ea ch ellipse. This will require adding a label to each ellipse instead of having to use the legend Each group of same coloured points represent a Type value. For each Type value, I calculate the mean PC1 (x-axis) and mean PC2 (y-values) values of data points. This gives me a mean PC1 (x-axis) and PC2 (x-axis) value for each Type, which I can plot on the PCA scatter plot.

```
means <- data_pca %>%

  group_by(Type) %>%

  summarise(PC1 = mean(PC1),

        PC2 = mean(PC2))
```

I can plot these mean values by themselves and we use a new geom and aesthetic mapping. The aesthetic label, represents a textual label. It is mapped to the Type value, and will cause a textual label to be produced for each Type value.

```
#-----------

 ggplot(means, aes(x = PC1, y = PC2, label = Type)) +

   geom_point() +

   geom_label() +

   theme_minimal()
```

We can see that some of the labels overlap because the clusters occupy the same space. Th e library ggrepel has a label geom type, geom_label_repel that will position each geom_label _repel label so that it does not overlap with other geom_label_repel labels in the plot.

```
 library(ggrepel)

 ggplot(means, aes(x = PC1, y = PC2, label = Type)) +

   geom_point() +

   geom_label_repel() +

   theme_minimal()
```

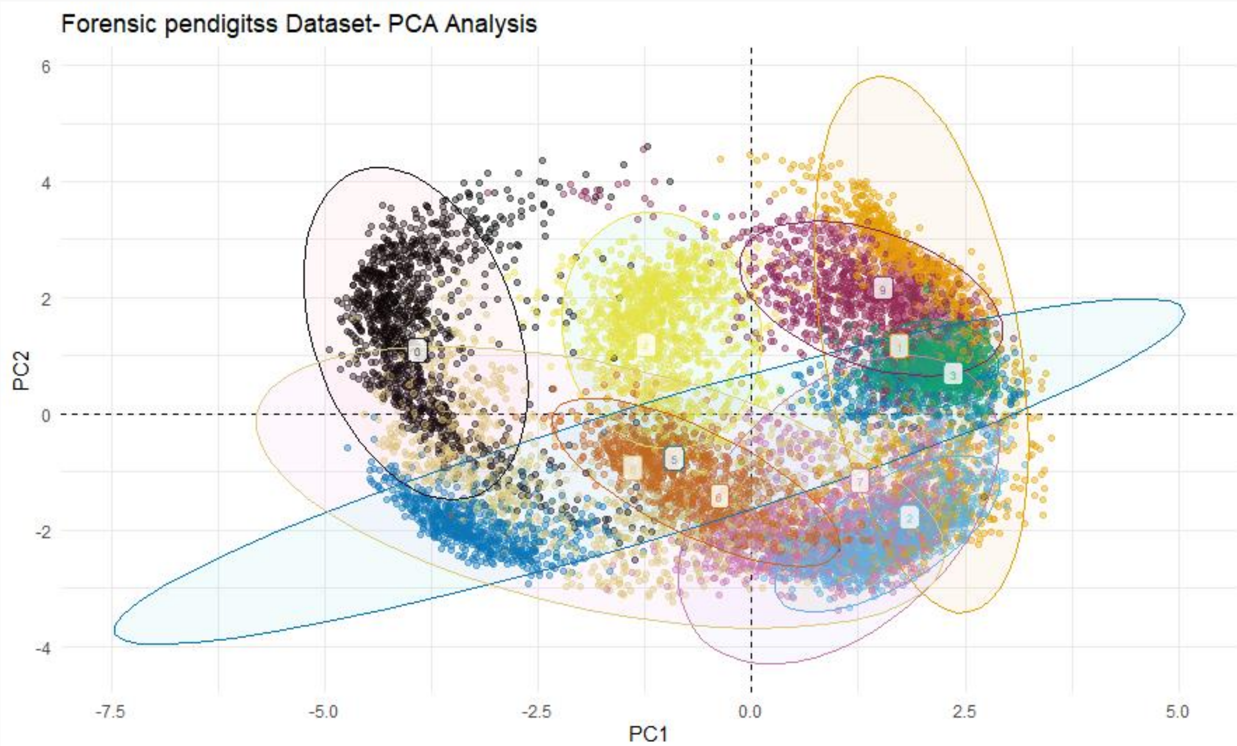
We now add these labels to the earlier plot. The geom_label_repel geom will refer to the me ans dataframe that we created. It will require its own aes mappings that override the aes m appings specified in the parent ggplot function. This is a typical example of how any geom c an override the data and aes declarations made in the parent ggplot function.

I will also remove the legend by setting the legend position to “none” : theme(legend.positi on = "none") and to add a title, here is the code:

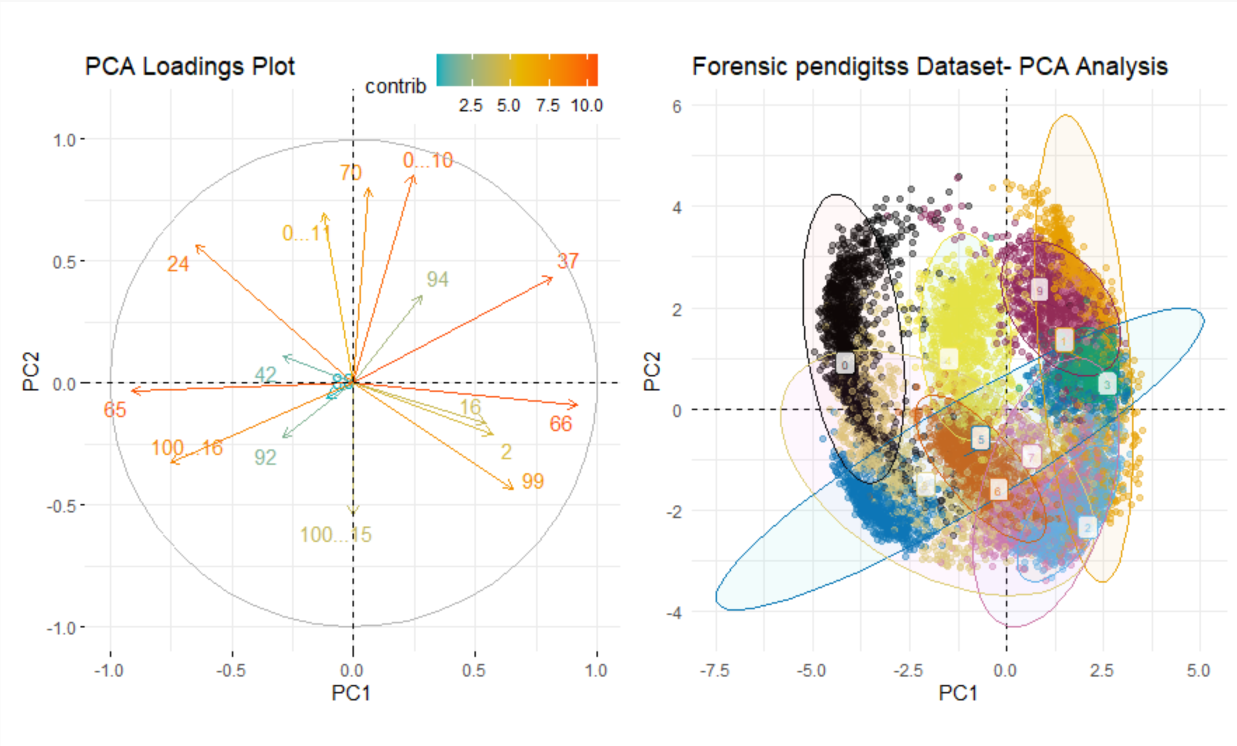
```
par(mar=c(1,3,1,1))

 pca_scatter<- ggplot(data_pca, aes(x=PC1, y=PC2, colour=Type))+

   # We add vertical and horizontal axis lines manually.

   # This helps us to compare this plot to the one earlier
```

```
  # so that we understand which features are discriminatory for each Type value.

  # For example, we can see that the headlamp glass is particularly

  # defined by Al, Ba and Na values.

  geom_vline(xintercept = 0, linetype="dashed", size = 0.2) +

  geom_hline(yintercept = 0, linetype="dashed", size = 0.2) +

  geom_point(size=1.5, alpha = 0.4) +

  # override the default colour allocation of the points with custom made palette using scal
e color manual

  scale_color_manual(values= c('#000000','#E69F00','#56B4E9','#009E73','#F0E442','#00
72B2',

                        '#D55E00','#CC79A7','#DDCC77','#882255')) +

  # plot an ellipse that encompasses the points belong to each Type value

  stat_ellipse(geom = "polygon",type = "t",size = 0.2,

             aes(color=Type, fill = Type),

             alpha = 0.05) +

  # the ellipse has a fill colour according to each Type value

  # override the default colour allocation of the points with custom made palette using scal
e color manual

  scale_color_manual(values= c('#000000','#E69F00','#56B4E9','#009E73','#F0E442','#00
72B2',

                        '#D55E00','#CC79A7','#DDCC77','#882255')) +

  # note how it has its own data and aes declaration,

  # this overrides the data and aes declaration made in the ggplot function

  geom_label_repel(data = means, size = 2,

               aes(x = PC1, y = PC2,

                 label = Type), alpha=0.8) +

  theme_minimal() +

  # removing the legend

  theme(legend.position = "none") +
```

```
  ggtitle("Forensic pendigitss Dataset- PCA Analysis")
plot(pca_scatter)
```



```
#here is the loading plot and scatter plot side by side for easier observation

library(patchwork)

 ppaloadings_plot + pca_scatter
```



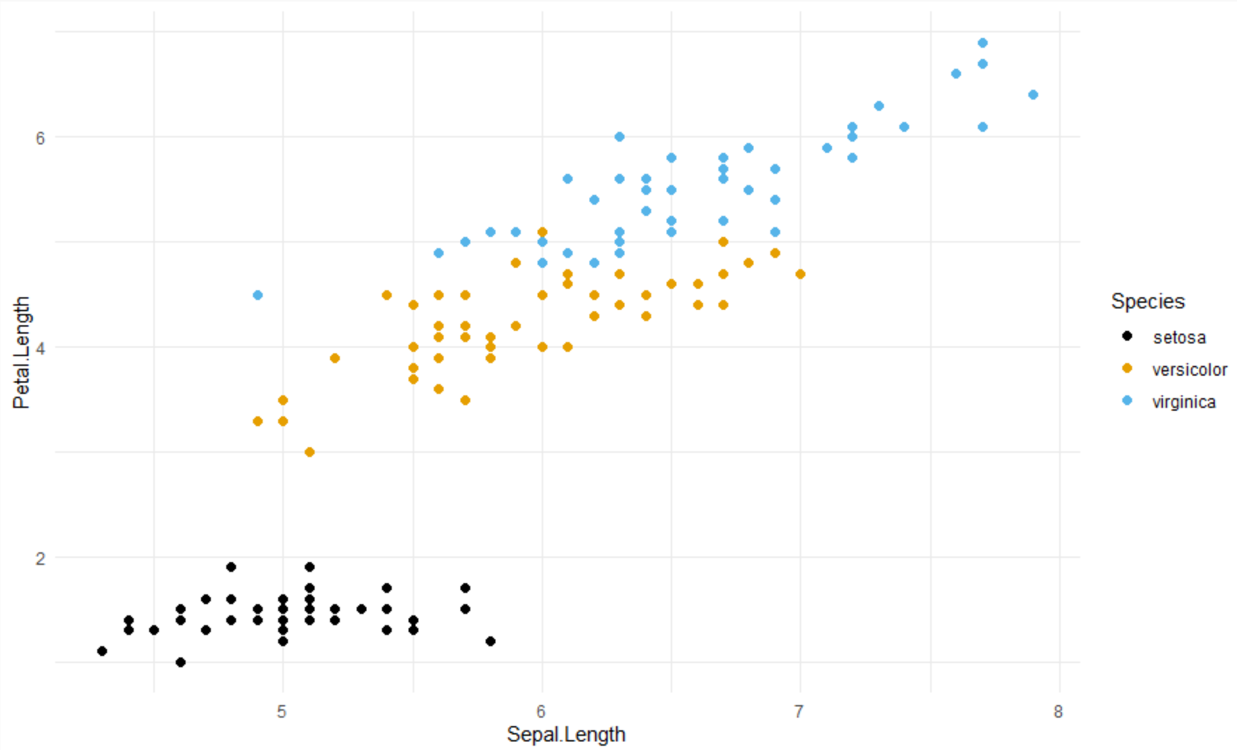
--

Color blindness test using R

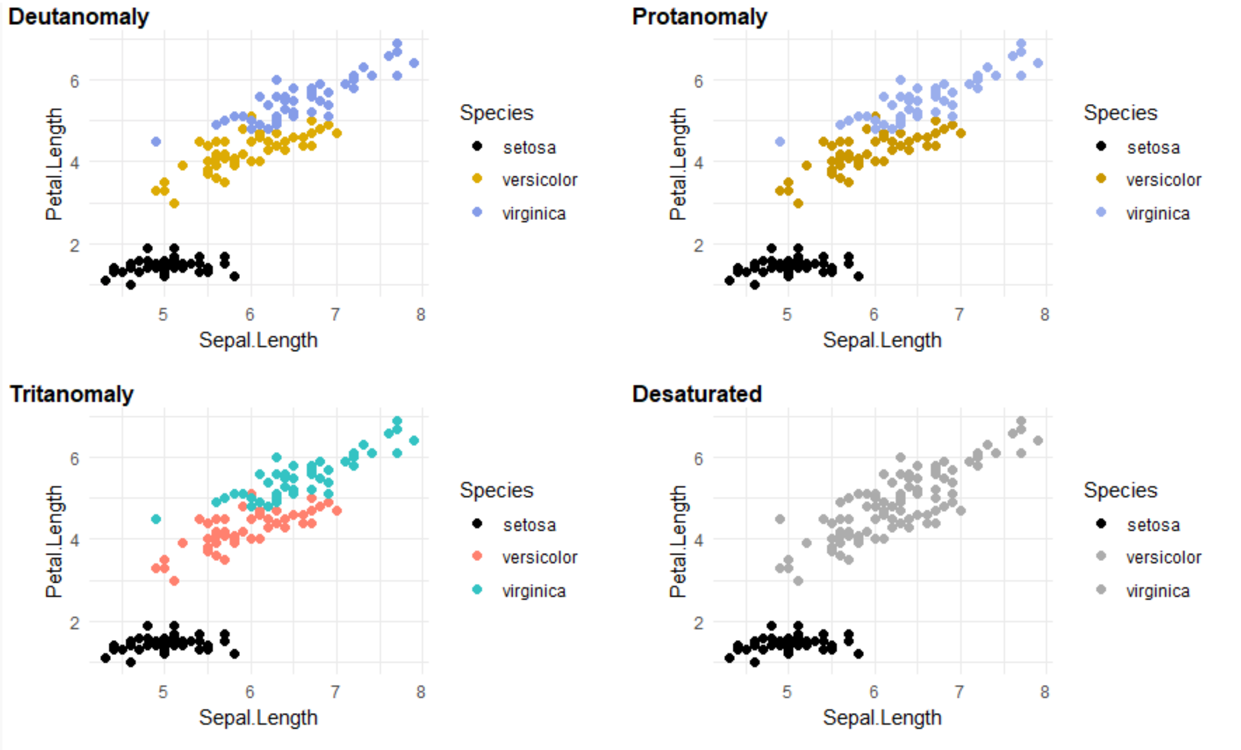
```
fig <- ggplot(iris, aes(x=Sepal.Length,y=Petal.Length, colour = Species)) +

  geom_point(size = 2) +

  scale_color_manual(values= c('#000000','#E69F00','#56B4E9','#009E73','#F0E442','#0072B2',

                       '#D55E00','#CC79A7','#DDCC77','#882255')) +

  theme_minimal()
```

```
fig
```

```
#using color blindr library to see  custom made palette from the perspective of a color blind person.  The different class colors plotted on the graph should come out as different colours in order to be readable for color blind people.
```




```
cvd_grid(fig)
```



As we can see by both graphs , based on the colours chosen, the color palette that I created makes it possible for a person with deuteranomaly , protanomaly and tritanomaly color bli ndless to be able to read the data set. Therefore I believe it is a colour blind friendly colour palette.

- Question: Explain why some groups of data points representing certain class values overlap in the scatterplot.?

Because there is so many data points over 10000 in the scatterplot with each class scattere d around the graph ina circle in clumps. Some of the classes such as class 1(orange) and cla ss 9 (purple) have very similar relationships with the PC1 and PC2 as seen with 0.10 and th e 37 arrow in the loadings plot being both positively correlated to pc2 going in the same dir ection, so naturally they will overlap on the scatter plot.