Visualisation of PCA Analysis

Name: Nakul Sunil Shinde

The first task includes loading data from CSV file to R Dataframe.

```
#-----------------------------------------------------------------------------------------
# Read CSV file and Load the data
#-----------------------------------------------------------------------------------------
library(readr)

pendigits_dataframe<-read.csv(file = "E:\\Sem 2\\Data visualization\\pendigits.csv"
                              ,header=FALSE)

#head(pendigits_dataframe)

colnames(pendigits_dataframe)[17] <- "class_value"
head(pendigits_dataframe)
```

```
##   V1  V2  V3  V4 V5 V6  V7 V8  V9 V10 V11 V12 V13 V14 V15 V16 class_value
## 1 88  92   2  99 16 66  94 37  70   0   0  24  42  65 100 100           8
## 2 80 100  18  98 60 66 100 29  42   0   0  23  42  61  56  98           8
## 3  0  94   9  57 20 19   7  0  20  36  70  68 100 100  18  92           8
## 4 95  82  71 100 27 77  77 73 100  80  93  42  56  13   0   0           9
## 5 68 100   6  88 47 75  87 82  85  56 100  29  75   6   0   0           9
## 6 70 100 100  97 70 81  45 65  30  49  20  33   0  16   0   0           1
```

```
#-----------------------------------------------------------------------------------------
# Feature Preprocessing
#-----------------------------------------------------------------------------------------
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
pendigits_dataframe %>%
  mutate(class_value = case_when(
    class_value == "0" ~ "0",
```

```
    class_value == "1" ~ "1",
    class_value == "2" ~ "2",
    class_value == "3" ~ "3",
    class_value == "4" ~ "4",
    class_value == "5" ~ "5",
    class_value == "6" ~ "6",
    class_value == "7" ~ "7",
    class_value == "8" ~ "8",
    class_value == "9" ~ "9"
  )) %>%
  mutate(class_value = factor(class_value,
  levels =
    c("0", "1", "2", "3", "4", "5","6","7","8","9")))->pendigits_dataframe
```

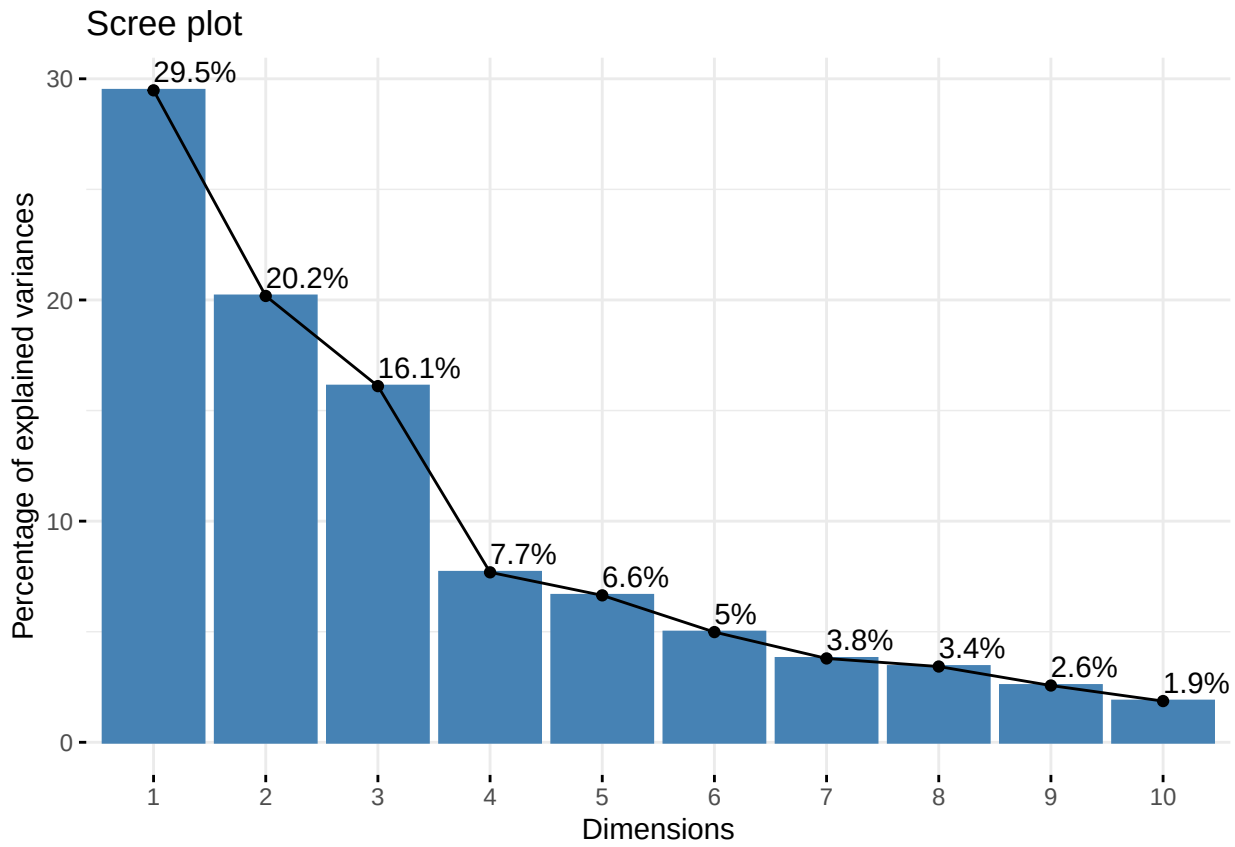
Generation of Scree Plot

```
#-------------------------------------------------------------------------------------------
# Scree Plot
#-------------------------------------------------------------------------------------------
library(FactoMineR)
library(dplyr)
dplyr::select(pendigits_dataframe, -class_value)%>%
  PCA(graph = FALSE) -> pca_analysis
library(factoextra)
```

```
## Loading required package: ggplot2
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

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



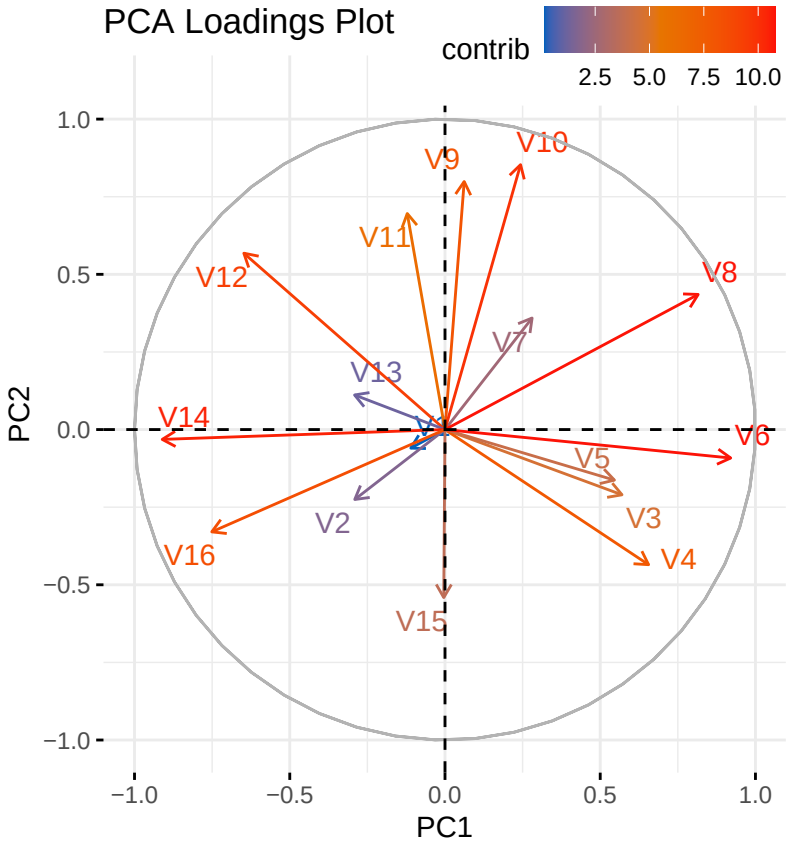
Analysis-

The first two principal components account for 49.7% of the total variation in the given pendigits dataset, according to the generated sreeplot. The first principal component accounts for 29.5 percent of overall variation, while the second component accounts for 20.2 percent. We can exhibit the data using these two components in the scatterplot because their combined variance is approximately equal to the total of all the remaining components in the scree plot.

Generation of Loadings Plot

```
#-------------------------------------------------------------------------------------------
# Loadings Plot
#-------------------------------------------------------------------------------------------
loadings_plot_pca <- fviz_pca_var(pca_analysis,
                                  col.var = "contrib",
                                  gradient.cols = c("#0060bc", "#e77500", "#fc1307"),
                                  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"),
  )

loadings_plot_pca
```



Analysis-

The strength of each characteristic's influence on the principal component is shown in a loadings plot. The correlations with the first principal component are plotted on the horizontal axis, but the correlations with the second principal component are plotted on the vertical axis, as can be seen.

After performing a thorough analysis, the following insights were found:

1) Feature V6 has a positive correlation with PC1, however V14 has a negative correlation with the same component.
2) Features V9 and V11 are positively correlated with PC2.Feature V15 has a negative relationship with PC2.
3) Feature V8 has a tendency to be negatively correlated with V1.Feature V4, which is in the opposite direction of V12, exhibits a similar pattern.
4) Both V5 and V3 have a tendency to be negatively correlated with V13.
5) The loadings plot's origin is further away for features V8, V12, and V16. As a result, these characteristics have a considerable impact on PC1 and PC2.

Creation of Custom Palette

```
#-------------------------------------------------------------------------------------
# Custom Palette
#-------------------------------------------------------------------------------------
# code for color palette and HCL_Picker
#install.packages('remotes')
#install.packages("colorspace", repos = "http://R-Forge.R-project.org")
#remotes::install_github("wilkelab/cowplot")
#remotes::install_github("clauswilke/colorblindr")

library(colorblindr)
```

```
## Loading required package: colorspace
```

```
## Warning: package 'colorspace' was built under R version 4.1.3
```

```
colr = c("#1E9E1F","#fce621","#39a5f9","#DEDDCB","#f96100","#e0108a",
                     "#F5BE66","#0c0000","#949494","#ac12f6")
#palette_plot(colr, label_size=3)

library(colorspace)
#hcl_color_picker()
colr_hcl_picker = c("#1E9E2F","#FCF621","#39A5E9","#DEDCCB","#F96300","#E0008A",
                     "#F5AE66","#0C0000","#949494","#AC42F6")

palette_plot(colr_hcl_picker, label_size=3)
```

```
colr_hcl_picker
```

```
##  [1] "#1E9E2F" "#FCF621" "#39A5E9" "#DEDCCB" "#F96300" "#E0008A" "#F5AE66"
##  [8] "#0C0000" "#949494" "#AC42F6"
```

Analysis-

It is necessary to select the right colors for personalized palette. Colorizer is a popular color color selection tool since it lets you choose up to ten colors, which is the minimum requirement for a palette.

After the color selection procedure is complete, another tool, color.adobe is used to see if the chosen colors are suitable for those with color vision deficiency. This tool includes a color blind simulator that helps to learn about the impact of selected colors on on a person with protanopia, deuteranopia, or tritanopia.

Finally, to perform slight changes in Hue, Chroma, and Luminance of colors, hcl_color_picker() was used. The final colors are organized sequantially in the vector colr_hcl_picker.

Colors for the palette were chosen with those with color vision impairments in mind.For this, the bright green and dark orange color hues were used. Because CVD patients can see multiple shades of black, the palette includes three distinct black colors with hexadecimal codes of #949494, #0c0000, and #DEDCCB. For a person with deuteranopia and protanopia, a dark magenta color shade with a hexadecimal value of #AC42F6 and a light pink color shade with a hexadecimal value of #F5AE66 are very easy to distinguish. The final palette is created by properly studying the role of each color in showing the data point and understanding the requirements of the CVD user.

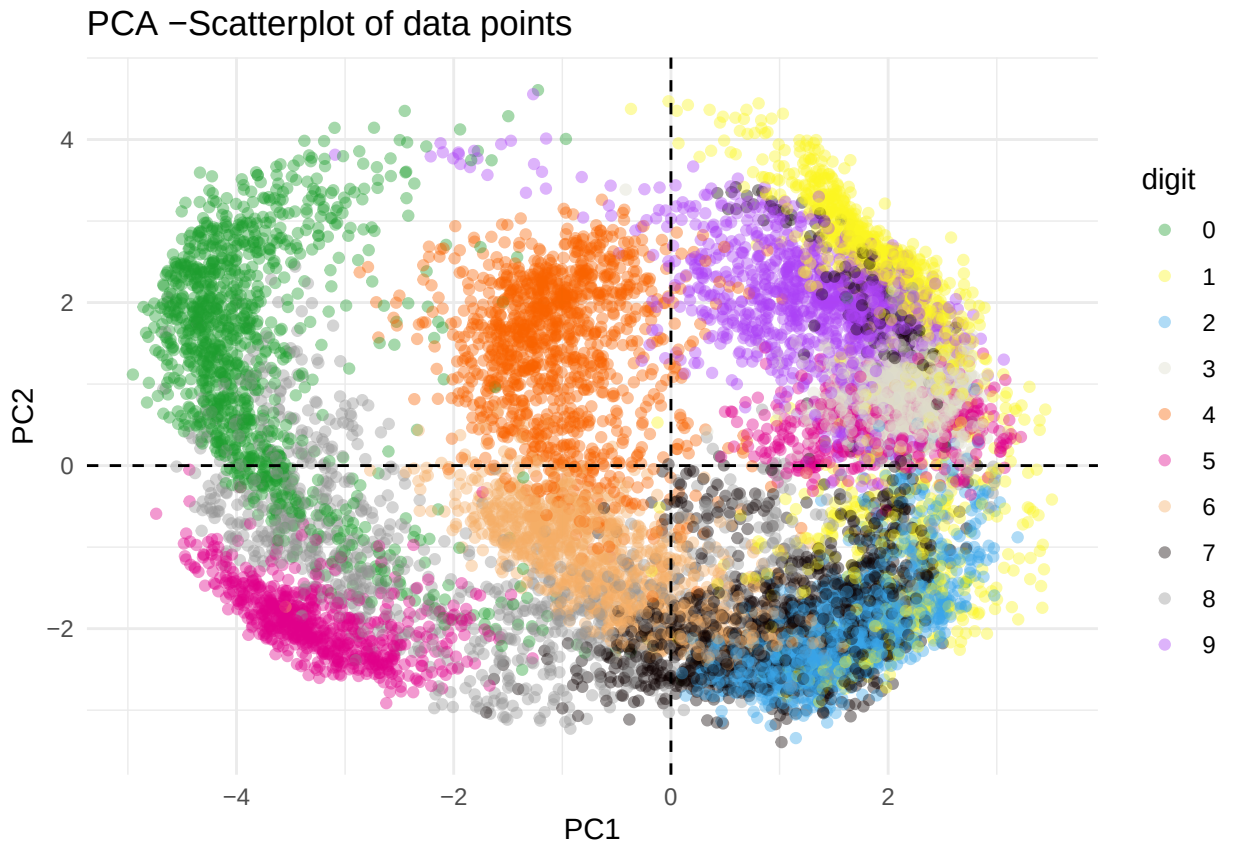
Principal Component Analysis-

```
#-------------------------------------------------------------------------------------------
# Principal Component Analysis
#-------------------------------------------------------------------------------------------
data_pca_individual <- get_pca_ind(pca_analysis)
#data_pca_individual

#head(data_pca_individual$coord)
data_pca_variable <- data_pca_individual$coord[,c(1,2)]
data_pca_variable <- as.data.frame(data_pca_variable)

names(data_pca_variable)[1] <- "PC1"
names(data_pca_variable)[2] <- "PC2"

data_pca_variable<- cbind(data_pca_variable, pendigits_dataframe$class_value)
names(data_pca_variable)[3] <- "digit"
head(data_pca_variable)
```

```
##          PC1         PC2 digit
## 1 -1.3400161 -1.78180963     8
## 2 -0.9418104 -2.08340573     8
## 3 -4.5516374 -0.01176024     8
## 4  1.6309516  2.36066213     9
## 5  0.9524490  2.14095480     9
## 6  2.2261574 -0.71280779     1
```

On the scatterplot, we only want to show the top two principal components out of the ten principal components. As a result, only these two columns have been chosen, and a new dataframe has been constructed for plotting using ggplot2. This dataframe has been expanded to include a new column called "class_value," whose values will be used to assign colors. The name of this column has been changed to "digit".

```
#-------------------------------------------------------------------------------------
# scatterplot
#-------------------------------------------------------------------------------------
library(ggplot2)
par(mar=c(1,3,1,1))
PCA_Plot<-ggplot(data_pca_variable, aes(x=PC1, y=PC2, color=digit)) +
  geom_point(size=1.5, alpha = 0.4) +
  ggtitle("PCA -Scatterplot of data points")+
  scale_color_manual(values = colr_hcl_picker) +
  geom_vline(xintercept = 0, linetype="dashed") +
  geom_hline(yintercept = 0, linetype="dashed") +
  theme_minimal()
PCA_Plot
```

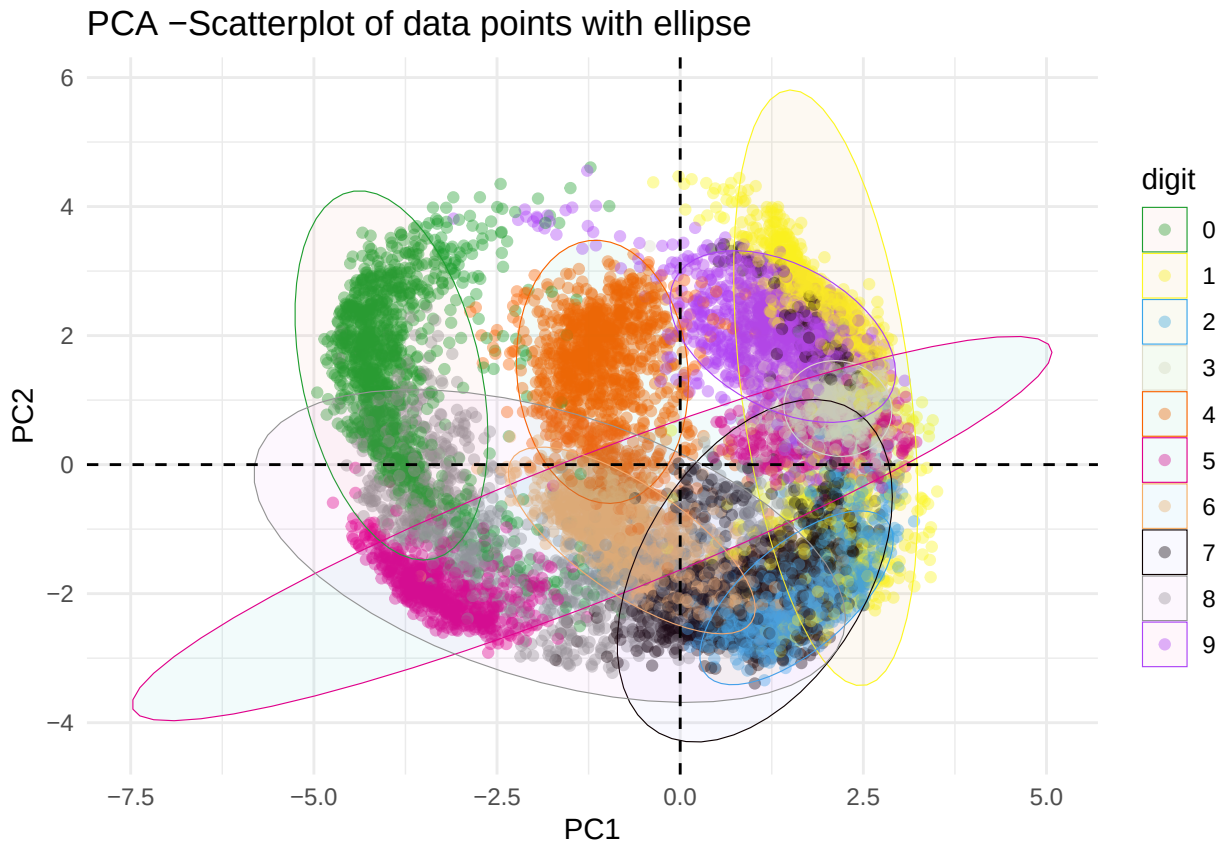


On the x-axis and y-axis, the first and second principal components are displayed. The color comes from the column digit.

The created color palette is implemented using the scale_color_manual() function, which allows a custom-built palette to be used. The stat_ellipse() function creates an ellipse for each group.

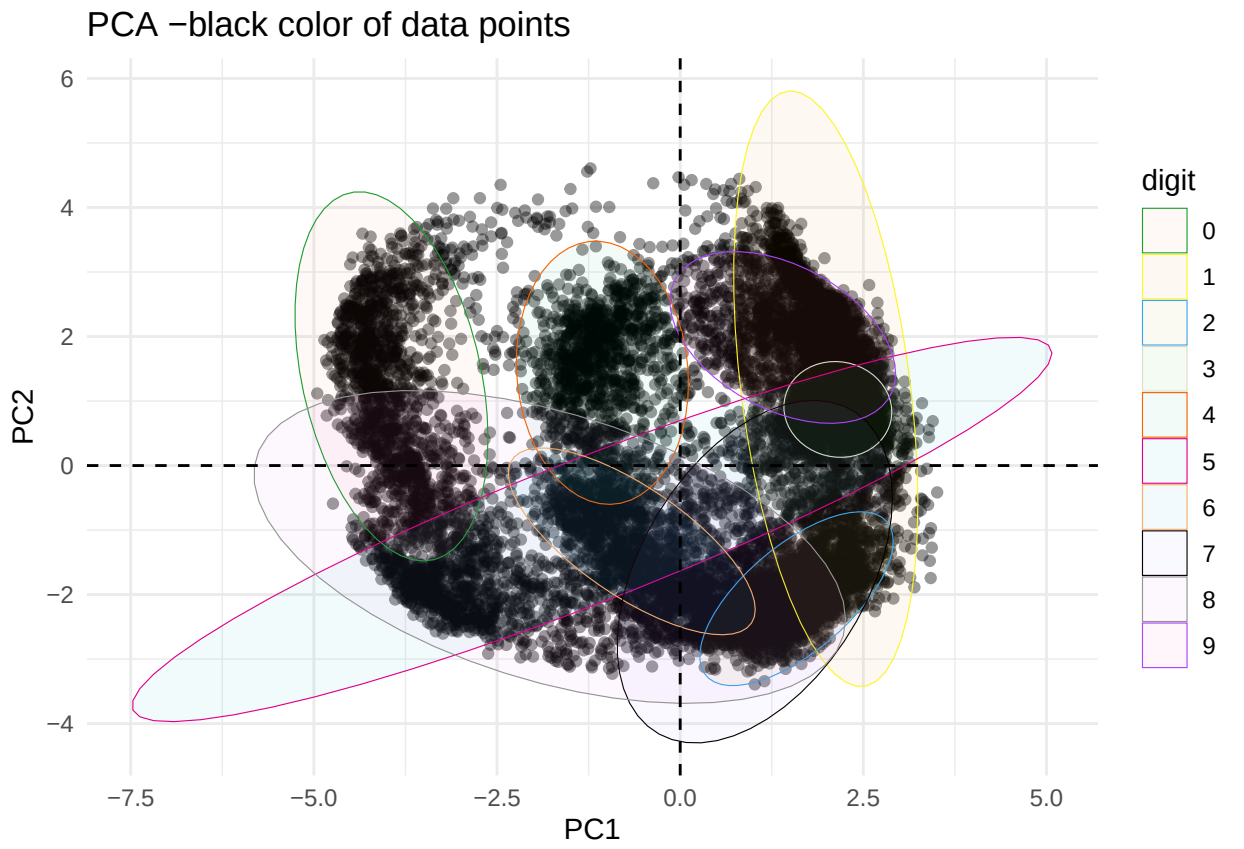
```
#-------------------------------------------------------------------------------------
# Scatterplot with ellipse
#-------------------------------------------------------------------------------------
ggplot(data_pca_variable,aes(x=PC1, y=PC2, color=digit)) +
  geom_point(size=1.5, alpha = 0.4) +
  scale_color_manual(values = colr_hcl_picker) +
  stat_ellipse(geom = "polygon",type = "t",size = 0.2,
```

```
              aes(fill = digit),
              alpha = 0.05) +
  ggtitle("PCA -Scatterplot of data points with ellipse")+
  geom_vline(xintercept = 0, linetype="dashed") +
  geom_hline(yintercept = 0, linetype="dashed") +
  theme_minimal()
```




```
#-------------------------------------------------------------------------------------------
# Scatterplot with black color data points
#-------------------------------------------------------------------------------------------
pca_datapoints_black <- ggplot(data_pca_variable,aes(x=PC1, y=PC2, color=digit)) +

geom_point(size=1.5, alpha = 0.4,color = "black") +

stat_ellipse(geom = "polygon",type = "t",size = 0.2,
aes(fill = digit),
alpha = 0.05) +
ggtitle("PCA -black color of data points") +
scale_color_manual(values = colr_hcl_picker) +

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


theme_minimal()
```

```
pca_datapoints_black
```



Analysis-

Descision to retain the colour of the data points as well as the coloured ellipses or not:

There are 10992 rows in the pendigits dataset. The resulting scatterplot depicts all of these data points with 10 original features because we are performing Principal Component Analysis on this set by examining the first two Principal Components. So, there's already a lot of information here, and retaining colored ellipses will make it more difficult to comprehend.However, The color of the data points must be retained as this scatterplot represents ten different features, and all of the data points that represent these features are scattered or overlapped with other points. Even some of the datapoints are separated from the clusters by a significant distance.

It will also be difficult to tell which data point belongs to which group if the color of the data points is removed as shown in "PCA -black color of data points" . In first scatterplot, We can see that datapoints with digit=7 and 8 are jumbled with class value 0 and 9 in the resulting graph. As a result, omitting color of data points will make comprehending the resulting scatterplot difficult.

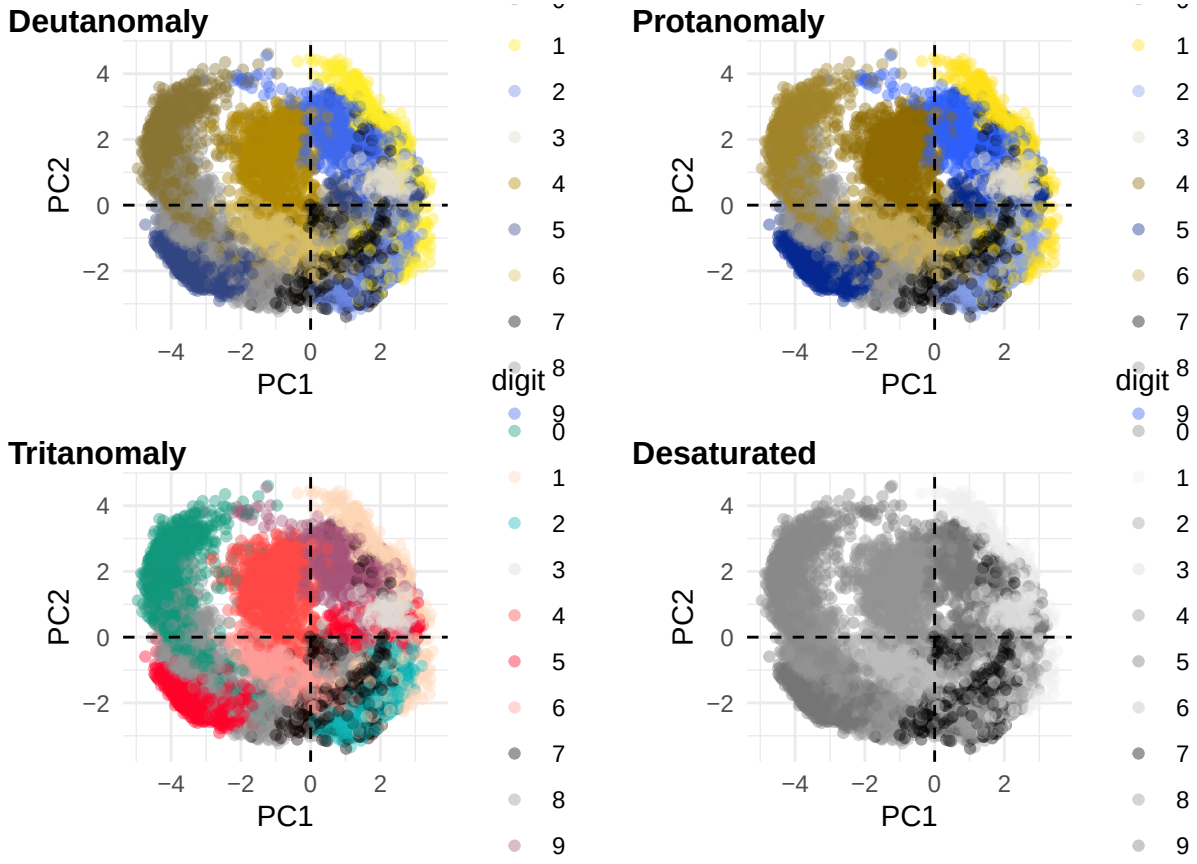
Explanation for some groups of data points representing certain class values overlap in the scatterplot:

The generated scatterplot represents principal component 1 on the x-axis with a variance of 29.5% and principal component 2 on the y-axis with a variance of 20.2%. The first two components account for 49.5 percent of the total variation. This percentage is insufficient to distinguish each feature's data points. Many of the data points with digit=7 are stretched across a large area of the graph, as can be seen in the scatterplot. digit=1 follows a similar pattern. As these points are not properly grouped together and more than half of

the variance is not represented by these 2 principal components, many data points are overlapped with each other.

CVD simulation of plot:

```
#-------------------------------------------------------------------------------------------
# CVD simulation Code
#-------------------------------------------------------------------------------------------
library(colorblindr)
PCA_Plot <- PCA_Plot + labs(title = NULL)
cvd_grid(PCA_Plot)
```



Analysis-

The inability to differentiate specific shades of color is known as color vision insufficiency. Deuteranopia is a kind of color blindness in which no green cones are present. Another kind is protanopia, which is characterized by the absence of red cones. Individuals with Tritanopia, on the other hand, lack blue cones. Because patients with CVD can't tell the difference between different colors like green and bluish green, creating a bespoke palette for them is tough.

To begin, a light green and draker orange combo was chosen. Three various colors of black were chosen since a person with CVD is able to detect different shades of black. The remaining colors are chosen based on their similarity. This created palette is ideal for persons who suffer from Tritanopia. The scatterplot's distinct color arrangements make it easier to evaluate the results for Deuteranopia and Protanopia patients. Colors with the hex values # ac42f6 and # 39a5e9, on the other hand, may be difficult to discern if you have severe color blindness, as these colors will appear dark blue and light blue to you. It is, nevertheless, perfectly suitable for someone with mild to moderate CVD.