##Performing Analysis on UKB4

```
library(dplyr)
library(ggplot2)
library(Rtsne)
library(data.table)
library(targets)
library(DT)
library(tidyverse)
library(utils)
library(gridExtra)
```

# Confirm working directory

```
getwd()
```

# Correctly read the CSV file into a data frame

```
data <- read.csv("ukb4.csv", header = TRUE, na.strings = ".")
```

##Visual exploration of the improve dataset shows that the genotypes in ukb4 are coded as 11,12, and 22.

##Transforming genotypes to allele frequencies in the ukb4 dataset

```
ukb4[ukb4==11] = 0
ukb4[ukb4==12] = 1
ukb4[ukb4==22] = 2
```

##Data Preparation Steps.

#Confirm that the data is in numerical data type. Change to numerical if not.

str(ukb4)

#Output shows data is numerical.

#To handle missing data using mean imputation:

# Function to impute missing values with the mean of the column

```
impute_mean <- function(vec) {
  m <- mean(vec, na.rm = TRUE)
  vec[is.na(vec)] <- m
  return(vec)
}
```

```
meanukb4 <- ukb4 # To Copy the original ukb4 data to preserve it
meanukb4[,-c(1)] <- apply(ukb4[,-c(1)], 2, impute_mean)   # Applying imputation
```

```
# Print the updated data frame and check its structure
print(head(meanukb4)) # Print the first few rows of the imputed data
str(meanukb4) # Check the structure of the updated data frame
```

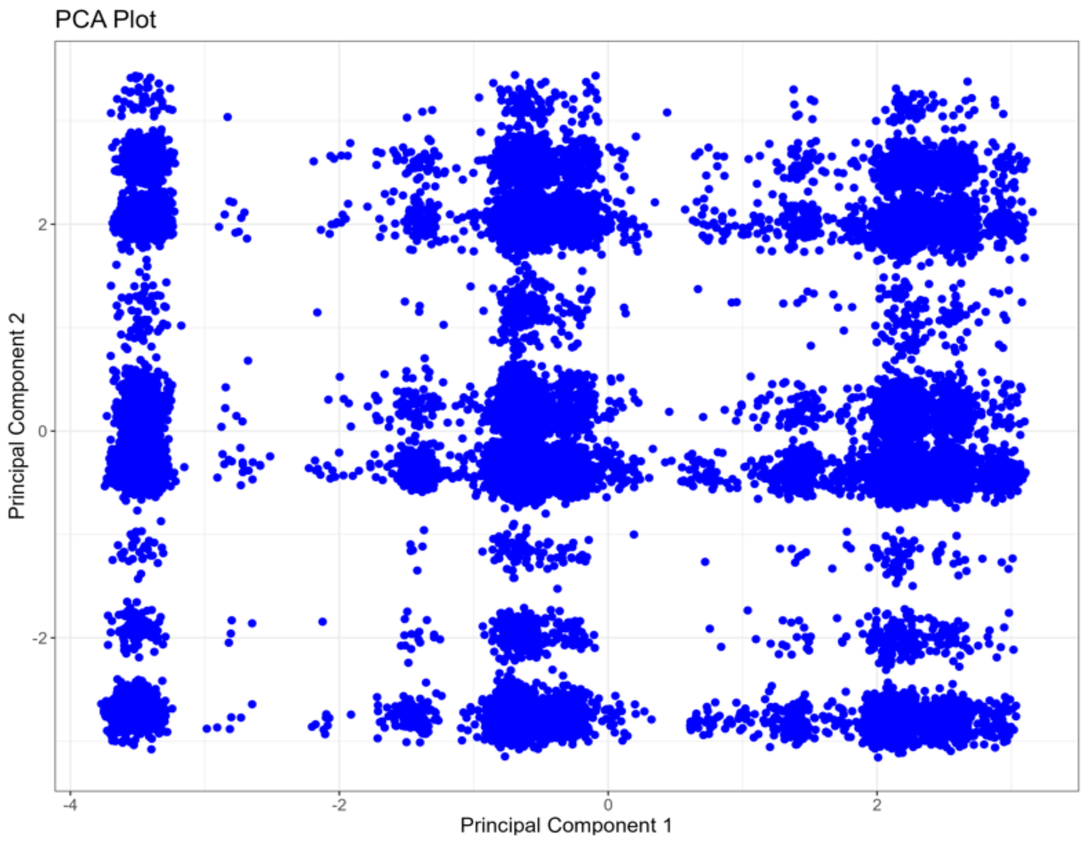
```
# Performing PCA and excluding column1 which is ID column
pca_result <- prcomp(meanukb4[, -1], center = TRUE, scale. = TRUE)
```

```
# Prepare PCA data
pca_data <- as.data.frame(pca_result$x)
```

```
# Add PCA results to the original data (for any potential future reference)
# pca_data$Original_Data_Column <- meanukb4$Your_Column  # Optional if you need to keep original columns
```

```
# Plot PCA results
# Using the first two principal components (PC1 and PC2) for the plot
pca_plot <- ggplot(pca_data, aes(x = PC1, y = PC2)) +
  geom_point(size = 2, color = "blue") +  # Color all points in blue (or any color of your choice)
  labs(title = "PCA Plot", x = "Principal Component 1", y = "Principal Component 2") +
  theme_bw() +
  theme(text = element_text(size = 14))
```

```
# Save the PCA plot to a file
ggsave("pca_plot.png", plot = pca_plot, width = 10, height = 8, units = "in", dpi = 300)
```



# Now moving on to perform t-SNE

# Set seed for reproducibility

```
set.seed(42)
```

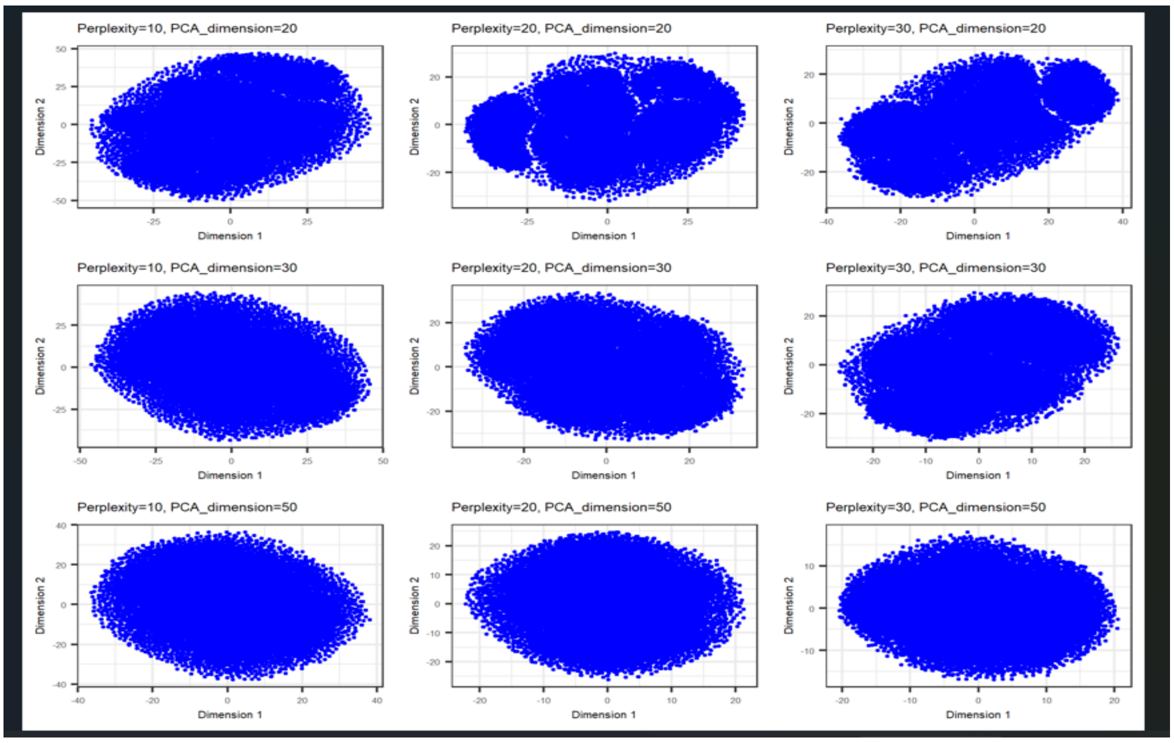
# Define parameters

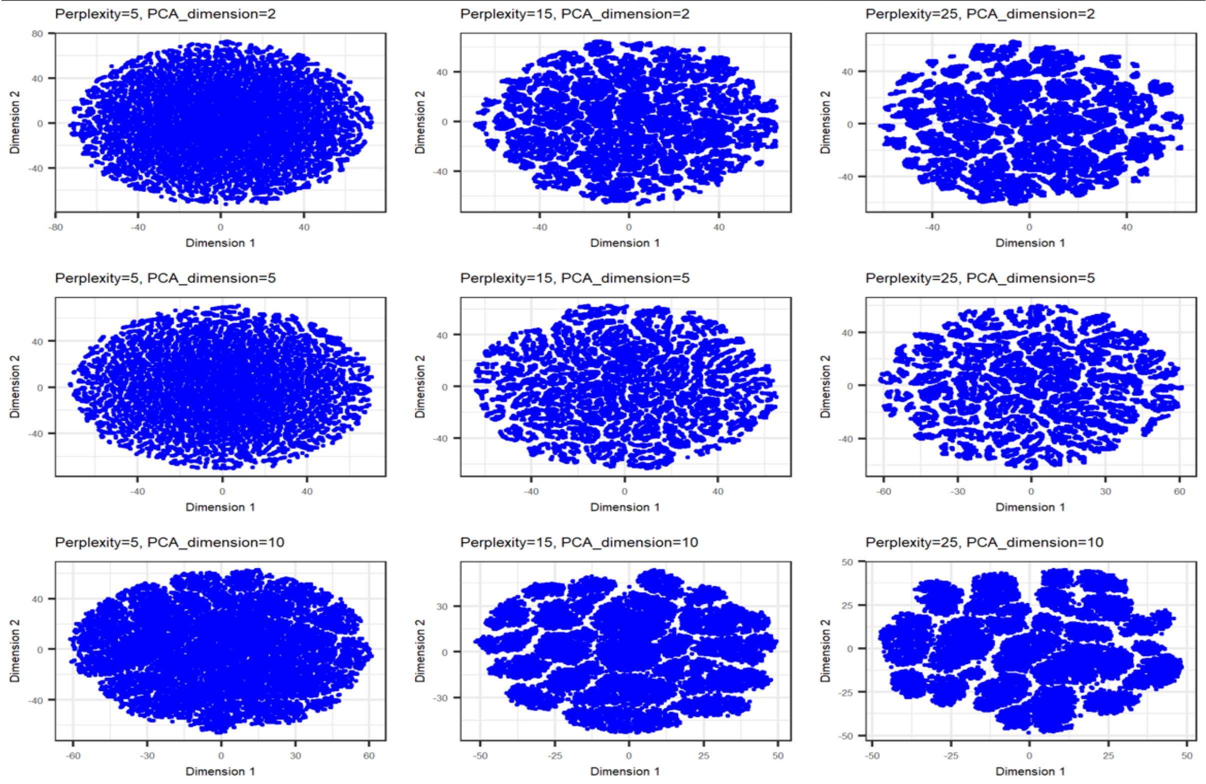
```
max_iter <- 1000 # Maximum iterations
theta <- 0.1
perplexities <- c(10, 20, 30) # Perplexity values
pcaDims <- c(20, 30, 50) # PCA dimensions
figWidth <- 2000
pointSize <- 0.5
textSize <- 5
num_threads <- 0
```

# Function to perform t-SNE with PCA initialisation and create plots

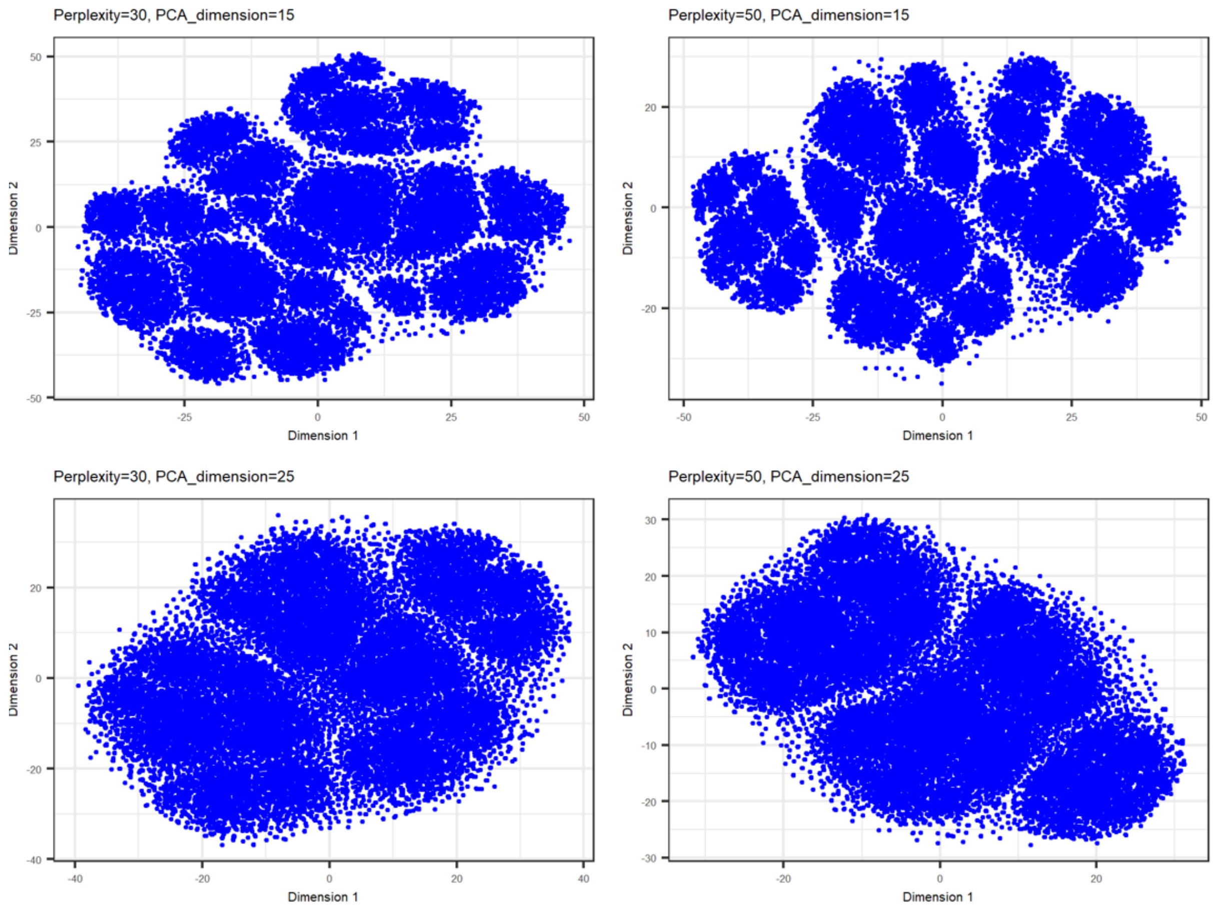
```
doRtsne <- function(data, perplexity, pcaDim) {
  tsne <- Rtsne(data,
                initial_dims = pcaDim,
                dims = 2,
                perplexity = perplexity,
                verbose = TRUE,
                max_iter = max_iter,
                theta = theta,
                num_threads = num_threads)

 # Create a data frame with t-SNE results
  tsne_plot <- data.frame(x = tsne$Y[, 1], y = tsne$Y[, 2])

 plot <- ggplot(tsne_plot, aes(x = x, y = y)) +
    geom_point(size = pointSize, color = "blue") +  # Default color for all
points
    ggtitle(paste0("Perplexity=", perplexity, ", PCA_dimension=", pcaDim))
+
    xlab("Dimension 1") +
    ylab("Dimension 2") +
    theme_bw() +
    theme(text = element_text(size = textSize))

  return(plot)
}

# Perform PCA to initialize t-SNE
pca_result <- prcomp(meanukb4[, -1], center = TRUE, scale. = TRUE)
pca_data <- as.data.frame(pca_result$x)

# Define your datasets (if you only have one dataset `meanukb4`, just use it directly)
datasets <- list(
  meanukb4 = pca_data
)

# Iterate over datasets
for (datname in names(datasets)) {
  dat <- datasets[[datname]]
 # Initialize a list to store plots
  pls <- list()


  # Perform t-SNE and plot for each PCA dimension and perplexity
combination
  for (pcaDim in pcaDims) {
    plots <- lapply(perplexities, function(perplexity)
      doRtsne(dat, perplexity, pcaDim))
    pls <- c(pls, plots)
  }


 # Arrange plots in a grid
 grid_plot_filename <- paste0("tsne_2d_grid_", datname, ".png")
  png(grid_plot_filename, width = figWidth, height = figWidth * 0.75, units
= "px", res = 300)
  grid.arrange(grobs = pls, nrow = length(pcaDims), ncol =
length(perplexities))
  dev.off()
}
```

Plot:





```
Max_iter: 1,500
```



Merging ukb4 with ukb2

## Now to combine ukb2 with ukb4 to see if there is any new insight

##Visual exploration of the improve dataset shows that the genotypes in ukb4 are coded as 11,12, and 22.

##Transforming genotypes to allele frequencies in the ukb4 dataset

```
sukb2[sukb2==11] = 0

sukb2[sukb2==12] = 1

sukb2[sukb2==22] = 2
```

##Data Preparation Steps.

#Confirm that the data is in numerical data type. Change to numerical if not.

```
str(sukb2)
```

#Output shows data is numerical.

#To handle missing data using mean imputation:

# Function to impute missing values with the mean of the column

```
impute_mean <- function(vec) {

  m <- mean(vec, na.rm = TRUE)  # Calculate the mean excluding NA values

  vec[is.na(vec)] <- m  # Replace NA values with the mean

  return(vec)  # Return the updated vector

}
```

# Make a copy of the original sukb2 data

```
msukb2 <- sukb2
```

# Apply mean imputation to the relevant columns in sukb2 (all apart first four)

```
msukb2[,-c(1,2,3,4)] <- apply(sukb2[,-c(1,2,3,4)], 2, impute_mean)
```

# Print the first few rows of the updated data frame

```
print(head(msukb2))
```

# Check the structure of the updated data frame

```
str(msukb2)
```

# Columns in meanukb4 but not in msukb2

```
cols_in_meanukb4_not_in_msukb2 <- setdiff(names(meanukb4), names(msukb2))
```

# Columns in msukb2 but not in meanukb4

```
cols_in_msukb2_not_in_meanukb4 <- setdiff(names(msukb2), names(meanukb4))
```

# Print the results

```
print(cols_in_meanukb4_not_in_msukb2)

print(cols_in_msukb2_not_in_meanukb4)
```

```
> print(cols_in_meanukb4_not_in_msukb2)
[1] "neid"       "rs12067567" "rs13081155" "rs11191438" "rs10786740"
[6] "rs11191609"
> print(cols_in_msukb2_not_in_meanukb4)
[1] "FID_71392...1"  "FID_71392...2"  "s2_groups"      "s2_groups_more"
[5] "rs12067700"     "rs1108842"      "rs6162"         "rs117814456"
```

# PCA on combined_data

# Perform PCA on the entire dataset

```
pca_result <- prcomp(combined_data, center = TRUE, scale. = TRUE)
```

# Prepare PCA data

```
pca_data <- as.data.frame(pca_result$x)
```

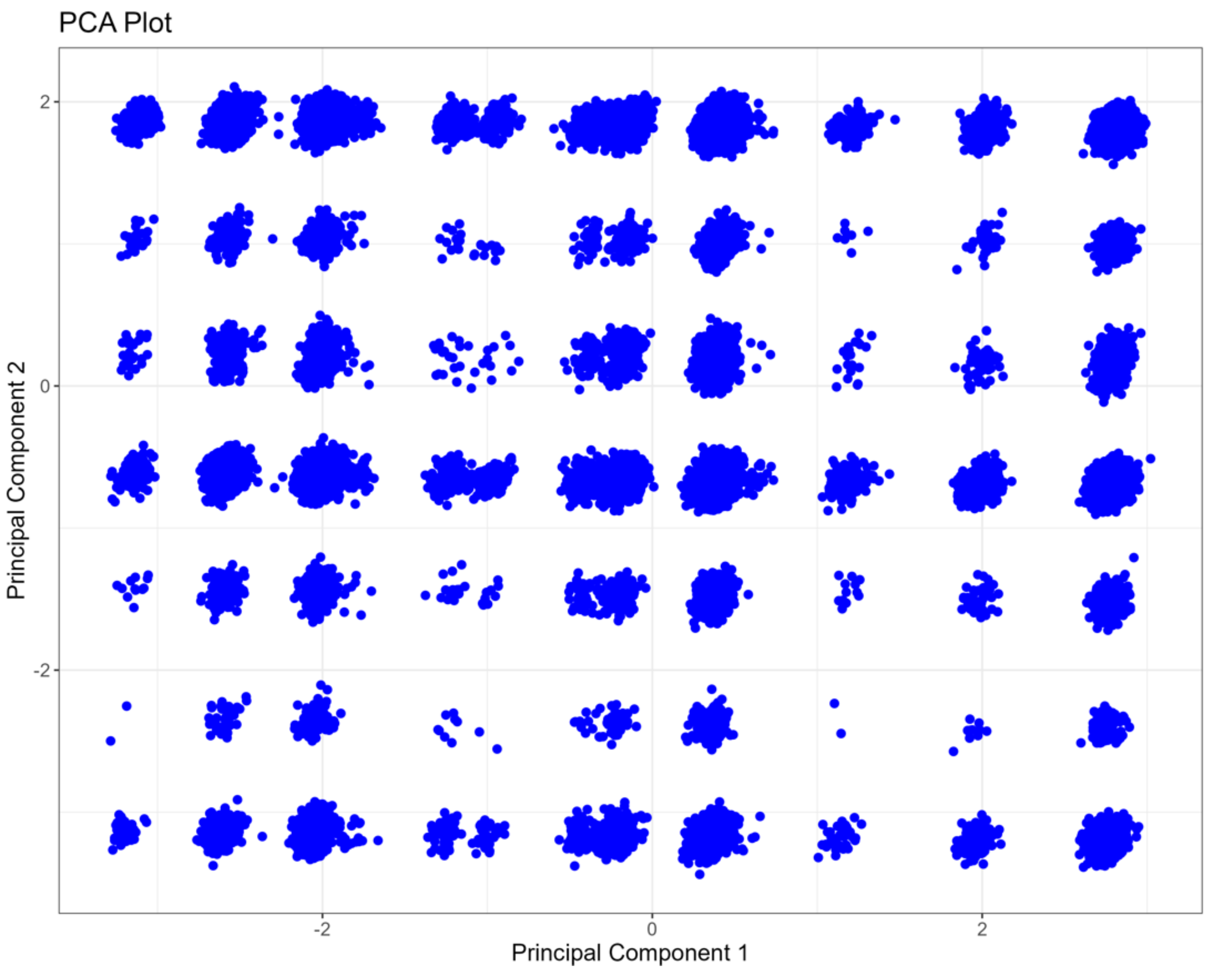
# Plot PCA results

# Using the first two principal components (PC1 and PC2) for the plot

```
pca_plot <- ggplot(pca_data, aes(x = PC1, y = PC2)) +
  geom_point(size = 2, color = "blue") +  # Color all points in blue (or
any color of your choice)
  labs(title = "PCA Plot", x = "Principal Component 1", y = "Principal
Component 2") +
  theme_bw() +
  theme(text = element_text(size = 14))
```

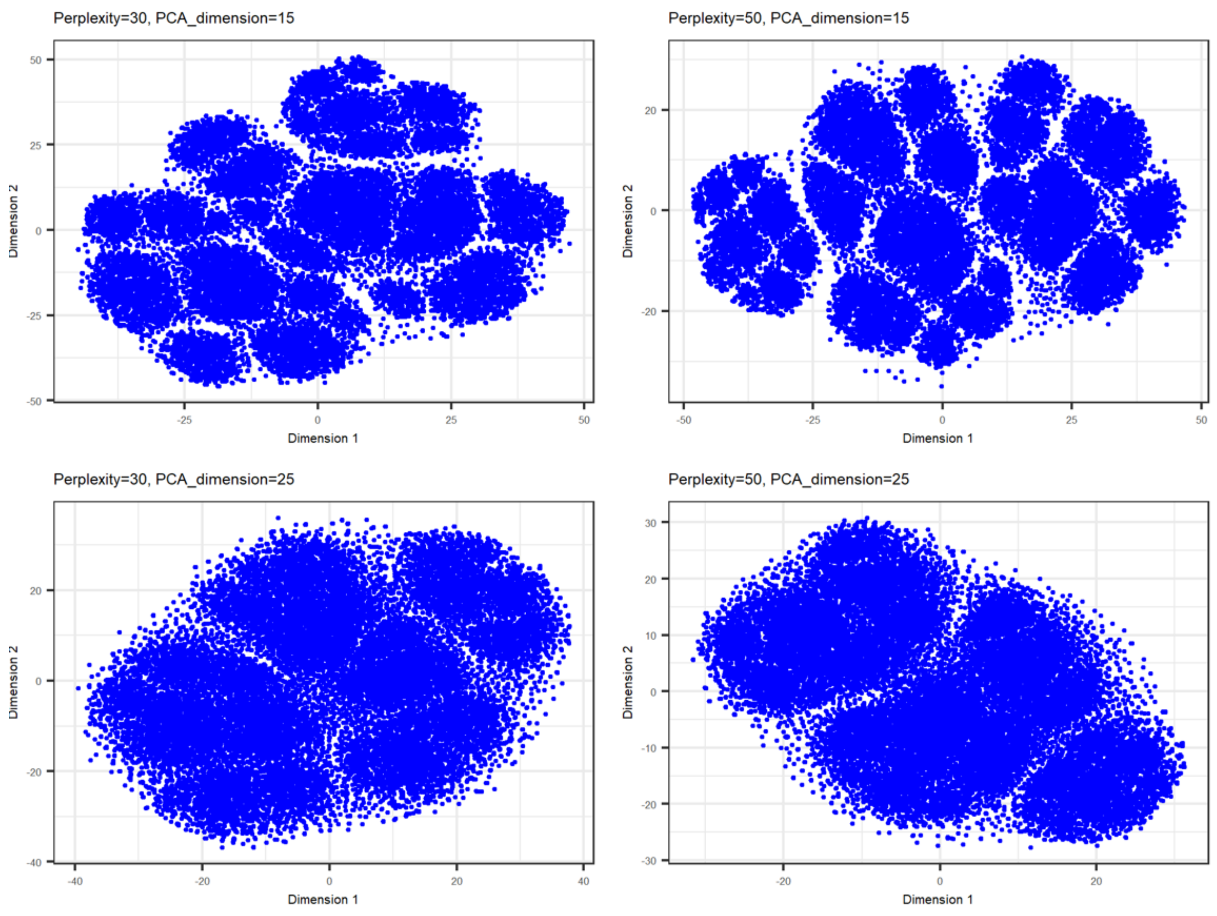
Save the PCA plot to a file

```
ggsave("pca_plot.png", plot = pca_plot, width = 10, height = 8, units =
"in", dpi = 300)
```



Next Steps:

Building on the previous result:



1. Running tSNE blind on UKB4 data blind and then assigning groups

```
library(dplyr)
library(ggplot2)
library(Rtsne)
library(data.table)
library(targets)
library(DT)
library(tidyverse)
library(utils)
library(gridExtra)]

getwd()
```

##Visual exploration of the improve dataset shows that the genotypes in ukb4 are coded as 11,12, and 22.

##Transforming genotypes to allele frequencies in the ukb4 dataset

```
ukb4[ukb4==11] = 0

ukb4[ukb4==12] = 1

ukb4[ukb4==22] = 2
```

##Data Preparation Steps.

#Confirm that the data is in numerical data type. Change to numerical if not.

```
str(ukb4)
```

#Output shows data is numerical.

#To handle missing data using mean imputation:

# Function to impute missing values with the mean of the column

```
impute_mean <- function(vec) {

  m <- mean(vec, na.rm = TRUE)

  vec[is.na(vec)] <- m

  return(vec)

}
```

`meanukb4 <- ukb4` # To Copy the original ukb4 data to preserve it

```
meanukb4[,-c(1)] <- apply(ukb4[,-c(1)], 2, impute_mean)  # Applying
imputation
```

# Print the updated data frame and check its structure

```
print(head(meanukb4)

str(meanukb4)
```

# Set seed for reproducibility

```
set.seed(42)
```

# Define parameters

```
max_iter <- 1500

theta <- 0.1

perplexities <- c(30, 50, 100)

pcaDims <- c(15, 25)
```

```r
figWidth <- 2000
pointSize <- 0.5
textSize <- 5
num_threads <- 0

# Function to perform t-SNE with PCA initialization, clustering, and create plots
perform_tsne_with_clustering <- function(data, perplexity, pcaDim) {
 # Step 1: Perform t-SNE
  tsne_result <- Rtsne(
    data,
    initial_dims = pcaDim,
    dims = 2,
    perplexity = perplexity,
    verbose = TRUE,
    max_iter = max_iter,
    theta = theta,
    num_threads = num_threads
  )



  # Step 2: Store t-SNE results in a data frame
  tsne_data <- data.frame(x = tsne_result$Y[, 1], y = tsne_result$Y[, 2])

  # Step 3: Perform k-means clustering on the t-SNE results
  num_clusters <- 3   # Adjust based on visual inspection or domain knowledge
  kmeans_result <- kmeans(tsne_data, centers = num_clusters)
  tsne_data$Cluster <- as.factor(kmeans_result$cluster)

  # Step 4: Plot the t-SNE results with cluster assignments
  plot <- ggplot(tsne_data, aes(x = x, y = y, color = Cluster)) +
    geom_point(size = pointSize) +
    ggtitle(paste0("Perplexity=", perplexity, ", PCA_dim=", pcaDim)) +
    xlab("Dimension 1") +
    ylab("Dimension 2") +
    theme_bw() +
    theme(text = element_text(size = textSize)) +
    scale_color_manual(values = c("1" = "red", "2" = "blue", "3" =
"green"))


  return(list(plot = plot, tsne_data = tsne_data, clusters =
kmeans_result$cluster))

}


# Step 5: Perform PCA on the data to prepare for t-SNE

# Assuming `meanukb4` is your dataset and it has an ID column that should be excluded

pca_result <- prcomp(meanukb4[, -1], center = TRUE, scale. = TRUE)

pca_data <- as.data.frame(pca_result$x)


# Step 6: Initialize a list to store the results (plots and clustered data)

results <- list()


# Step 7: Perform t-SNE with clustering for each combination of PCA dimension and perplexity

for (pcaDim in pcaDims) {

  for (perplexity in perplexities) {

    result <- perform_tsne_with_clustering(pca_data, perplexity, pcaDim)

    results[[paste0("PCA_", pcaDim, "_Perplexity_", perplexity)]] <- result

  }

}


# Step 8: Save the plots and clustered data

for (name in names(results)) {

 # Save the plot

  plot <- results[[name]]$plot

  plot_filename <- paste0("tsne_", name, ".png")

  ggsave(plot_filename, plot = plot, width = 10, height = 8, units = "in",
dpi = 300)


 # Save the t-SNE data with clusters

  tsne_data <- results[[name]]$tsne_data

  tsne_data_filename <- paste0("tsne_data_", name, ".csv")

  write.csv(tsne_data, tsne_data_filename, row.names = FALSE)

}
```

# Step 9: Save the meanukb4 data with cluster assignments

```
meanukb4$Cluster <- results[[1]]$clusters  # Assign clusters from the first
tsne result as an example

write.csv(meanukb4, "meanukb4_with_clusters.csv", row.names = FALSE)
```

# Step 10: Inspect the meanukb4 data with clusters

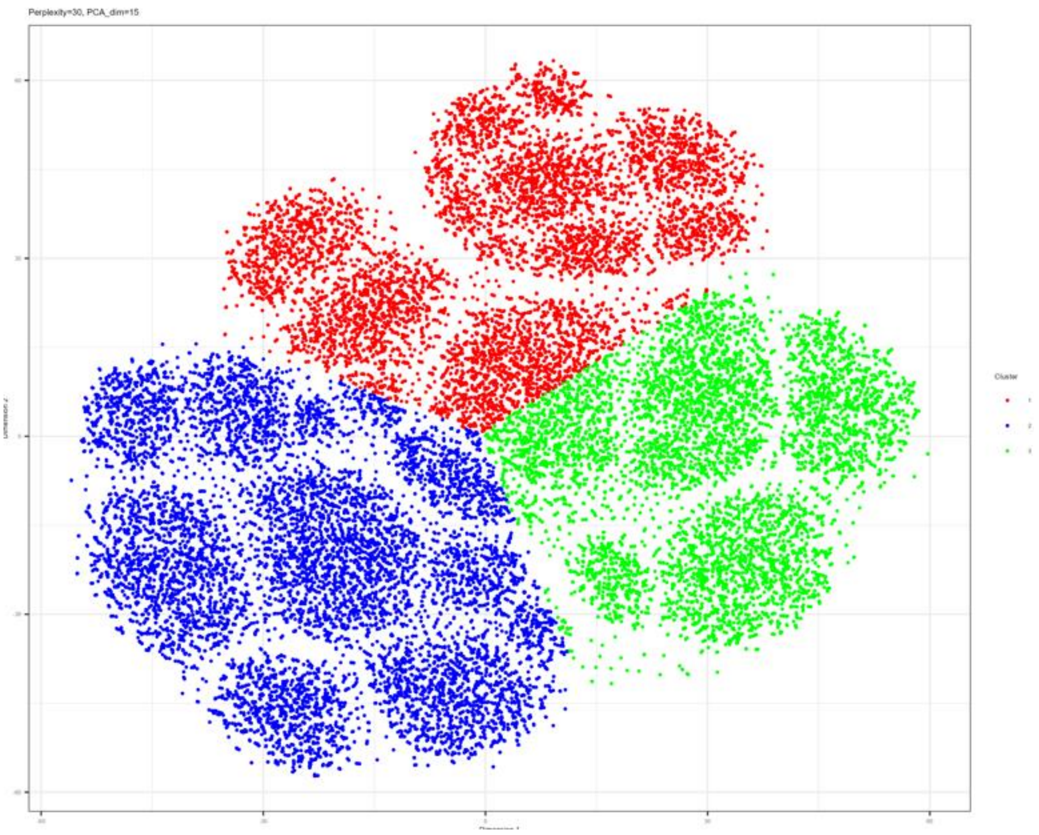
```
head(meanukb4)
```

Results:

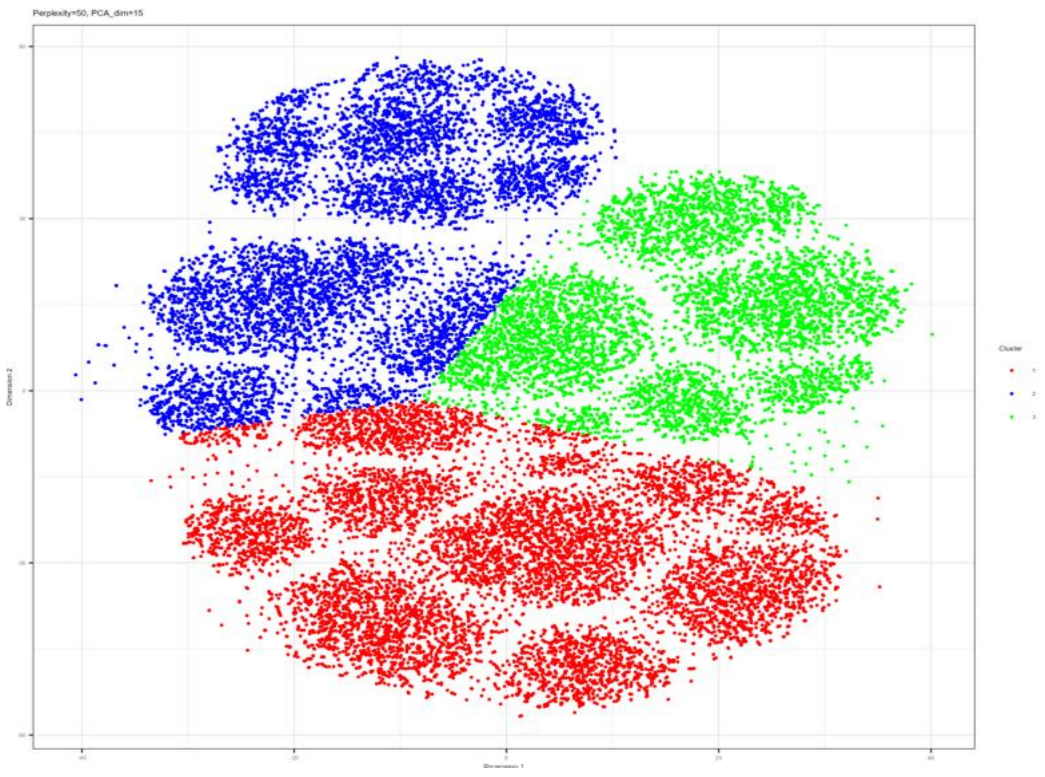
Run time: 56 minutes

Plots:

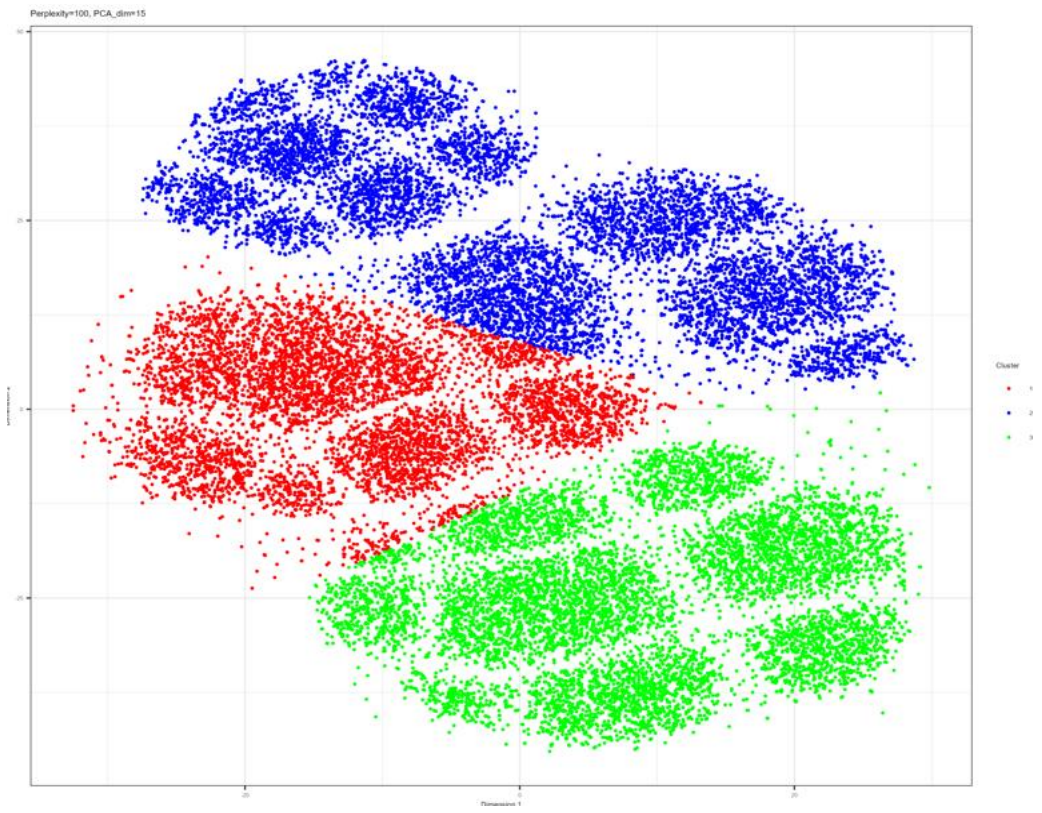
1. Perplexity 30; PCA_dim 15



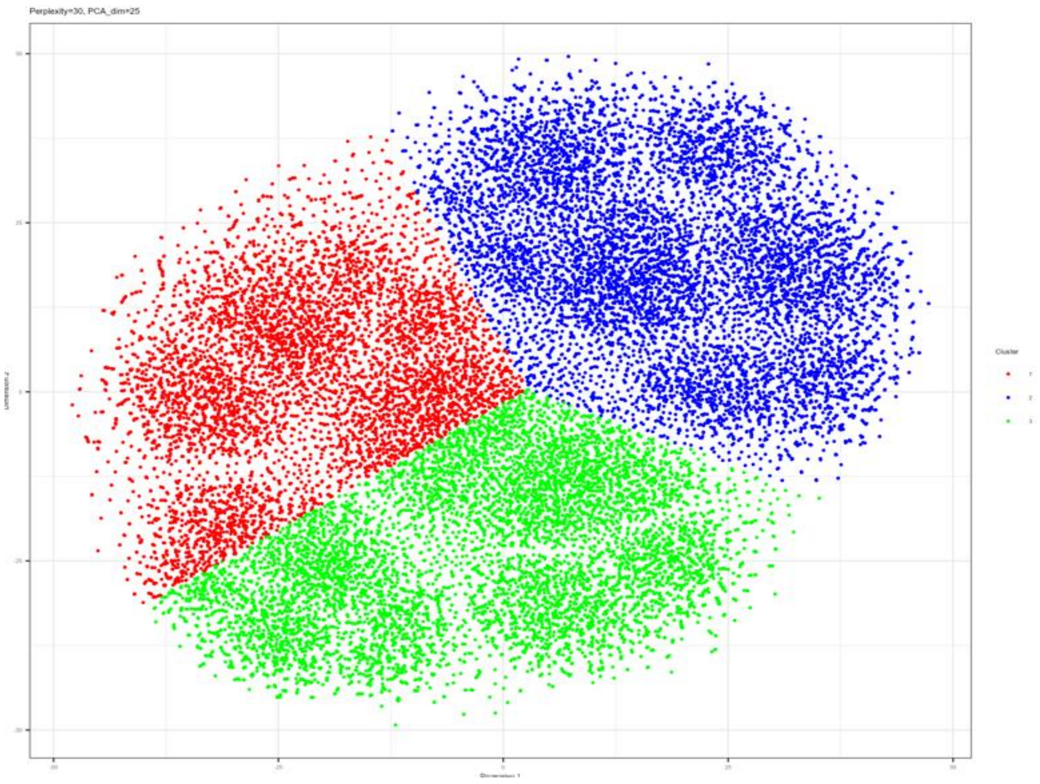
2. Perplexity 50; PCA_dim 15



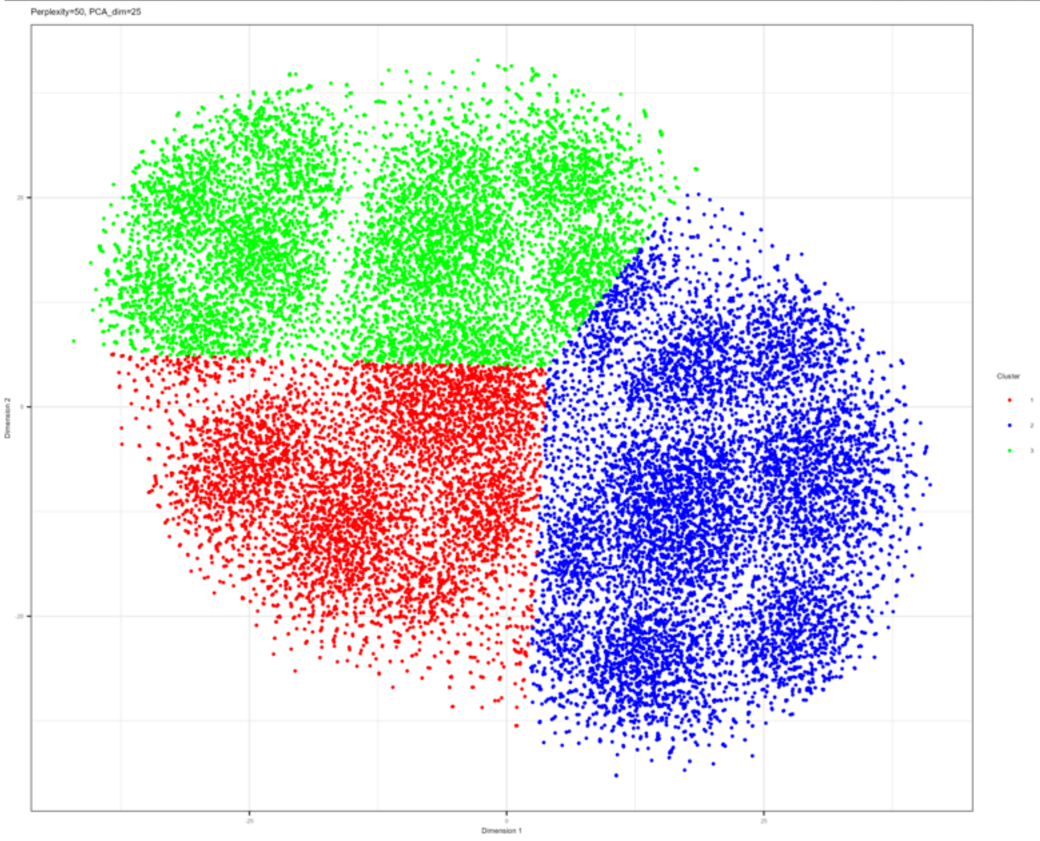
3. Perplexity 100; PCA_dim 15



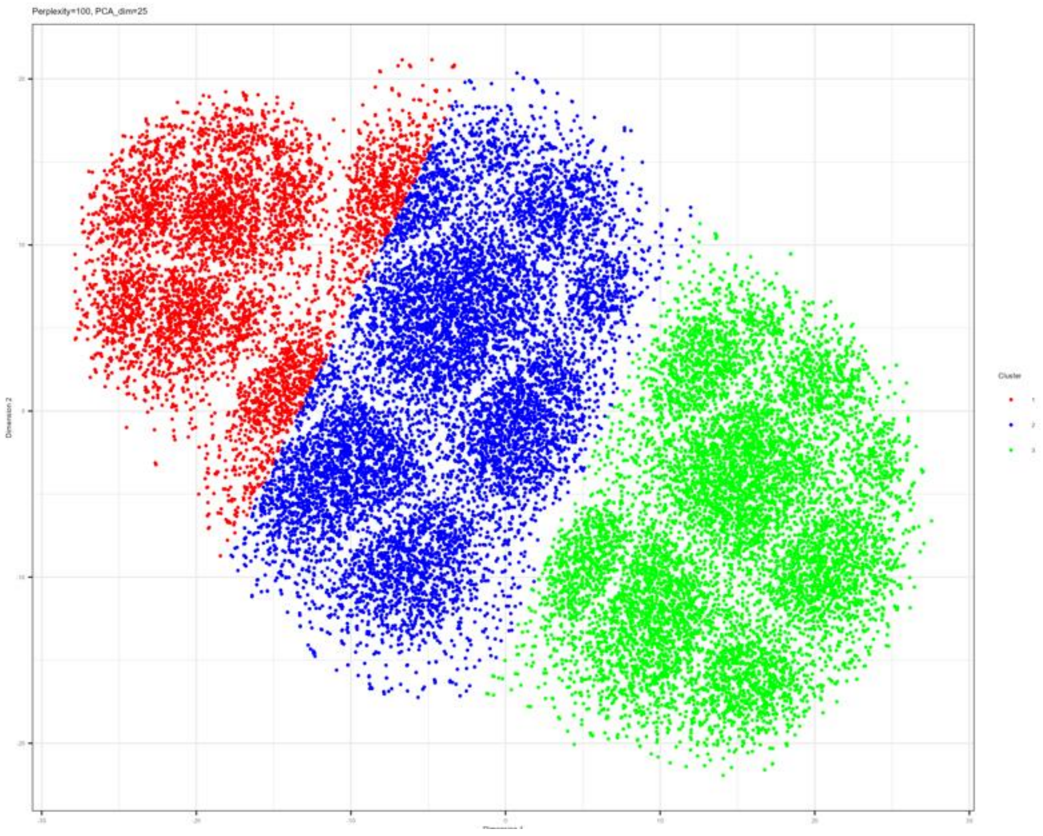
4. Perplexity 30; PCA_dim 25



5. Perplexity 50; PCA_dim 25



6. Perplexity 100; PCA_dim 25



2. Compare the grouping with MDS grouping

```
Result: Visual inspection shows that the groping does not match the one
from MDS
```

3. UKB4 coloured with MDS Grouping

##merging ukb4 with predetermined groupings

```
grouped_ukb4 <- merge(ukb4, ukb4_groups, by = "neid")


write.csv(grouped_ukb4, "grouped_ukb4.csv", row.names = FALSE)
```

##Moving grouping column to column 2 for easier analysis:

# Identify the name of the column to move

```
column_to_move <- "s4_groups"
```

# Get the current column order

```
current_columns <- names(grouped_ukb4)
```

# Create a new column order with 's4_groups' moved to the second position

```
new_column_order <- c(
  current_columns[1],                           # Keep the first column
  column_to_move,                               # Move 's4_groups' to the second
position
  current_columns[!(current_columns %in% c(current_columns[1],
column_to_move))]  # The rest of the columns
)
```

# Reorder the columns

```
grouped_ukb4 <- grouped_ukb4[, new_column_order]
```

str(grouped_ukb4_encoded)

##Running tSNE

# Load necessary libraries

```
library(Rtsne)
library(ggplot2)
library(gridExtra)
library(dplyr)
library(tidyr)
```

# Set seed for reproducibility

```
set.seed(42)
```

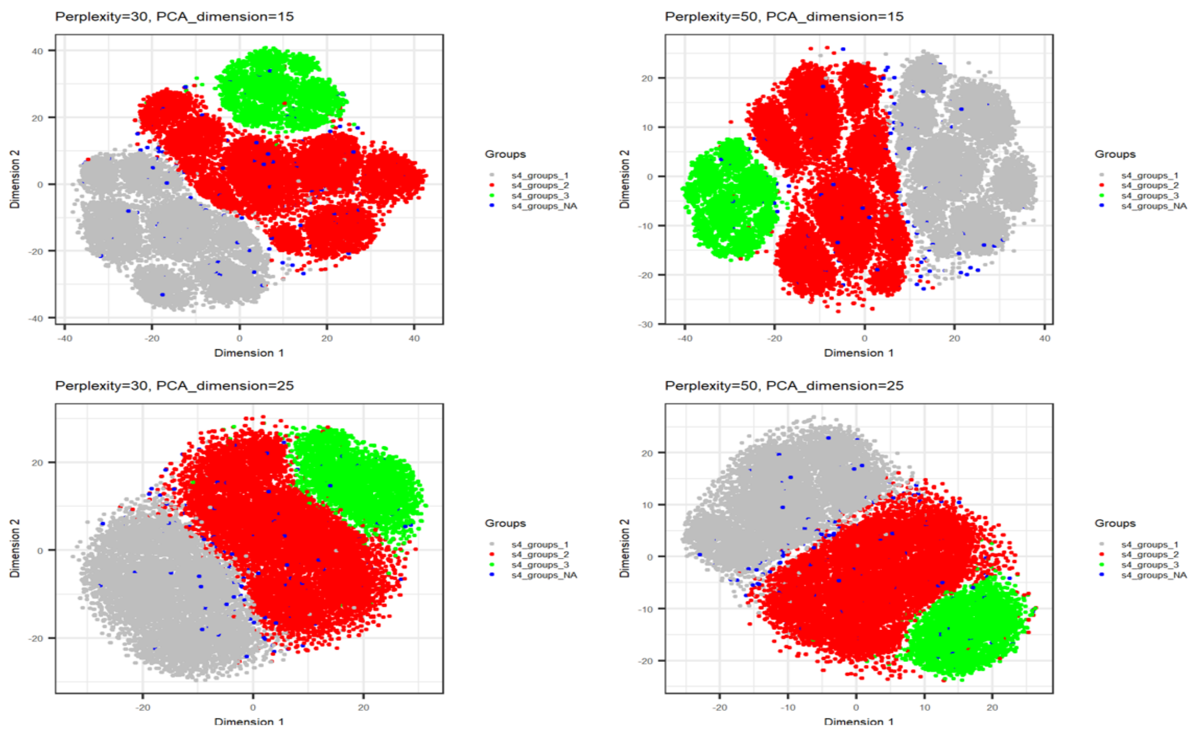
# Define parameters

```
max_iter <- 1500
theta <- 0.1
perplexities <- c(30, 50, 100)
pcaDims <- c(15, 25)
figWidth <- 2000
pointSize <- 0.5
legendSize <- 5
textSize <- 5
num_threads <- 0
```

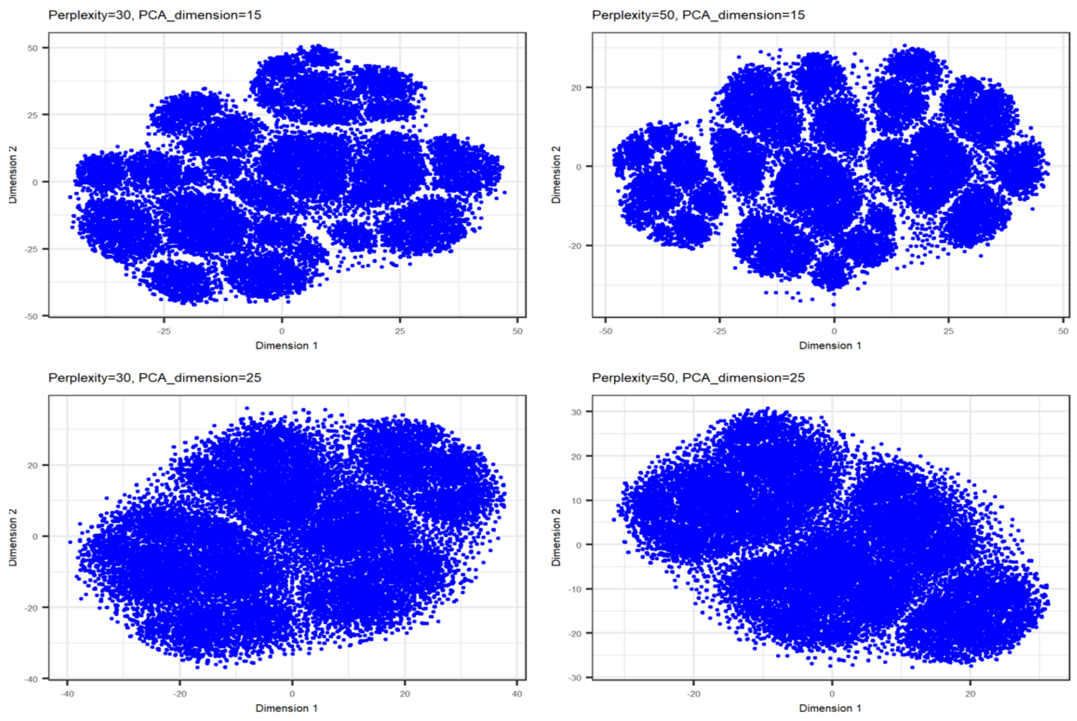
```
mycolors <- c("s4_groups_1" = "gray", "s4_groups_2" = "red",
              "s4_groups_3" = "green", "s4_groups_NA" = "blue")


# Function to perform t-SNE with PCA initialization and create plots
doRtsne <- function(data, perplexity, pcaDim) {
 # Exclude non-numeric columns
  numeric_data <- data %>%
    select_if(is.numeric) %>%
    select(-one_of(c("neid", "s4_groups_1", "s4_groups_2", "s4_groups_3",
"s4_groups_NA")))


 # Check for NA values in numeric data and remove rows with NA
  numeric_data <- na.omit(numeric_data)


 # Perform PCA
 pca_result <- prcomp(numeric_data, center = TRUE, scale. = TRUE)
  pca_data <- as.data.frame(pca_result$x)


 # Ensure pca_data has enough dimensions for t-SNE
  if (ncol(pca_data) < pcaDim) {
    stop("Not enough dimensions in PCA data.")
  }


 # Perform t-SNE
 tsne <- Rtsne(pca_data[, 1:pcaDim],
               dims = 2,
               perplexity = perplexity,
               verbose = TRUE,
               max_iter = max_iter,
               theta = theta,
               num_threads = num_threads)


 # Create t-SNE plot data
 tsne_plot <- data.frame(x = tsne$Y[, 1], y = tsne$Y[, 2])


 # Add the one-hot encoded group columns for plotting
```

```r
 tsne_plot <- cbind(tsne_plot, data[, grep("s4_groups_", names(data))])


 # Create Groups column from one-hot encoded columns

  tsne_plot$Groups <- apply(tsne_plot[, grep("s4_groups_",
names(tsne_plot))], 1, function(x) {

    if (x["s4_groups_1"] == 1) return("s4_groups_1")

    if (x["s4_groups_2"] == 1) return("s4_groups_2")

    if (x["s4_groups_3"] == 1) return("s4_groups_3")

    if (x["s4_groups_NA"] == 1) return("s4_groups_NA")

    return("Unknown")

  })


 # Plot

 plot <- ggplot(tsne_plot, aes(x = x, y = y, color = Groups)) +

    geom_point(size = pointSize) +

    scale_color_manual(values = mycolors) +

    ggtitle(paste0("Perplexity=", perplexity, ", PCA_dimension=", pcaDim))
+

    xlab("Dimension 1") +

    ylab("Dimension 2") +

    theme_bw() +

    theme(text = element_text(size = textSize), legend.key.size =
unit(legendSize, "point"))


  return(plot)

}


# Generate t-SNE plots for each combination of PCA dimensions and perplexities

plots <- list()

for (pcaDim in pcaDims) {

  for (perplexity in perplexities) {

    plot <- doRtsne(grouped_ukb4_encoded, perplexity, pcaDim)

    plots[[paste0("pcaDim_", pcaDim, "_perplexity_", perplexity)]] <- plot

  }

}


# Arrange plots in a grid
```

```
grid_plot_filename <- "tsne_2d_grid_grouped_ukb4_encoded.png"

png(grid_plot_filename, width = figWidth, height = figWidth * 0.75, units =
"px", res = 300)

grid.arrange(grobs = plots, nrow = length(pcaDims), ncol =
length(perplexities))

dev.off()
```



Comparison with Blind:



4. UKB4 + UKB2 (blind)

# Perform t-SNE with PCA initialisation

# Set seed for reproducibility

```
set.seed(42)
```

# Define parameters

```
max_iter <- 1500 # Maximum iterations

theta <- 0.1

perplexities <- c(20, 30, 50) # Perplexity values

pcaDims <- c(15, 25) # PCA dimensions

figWidth <- 2000

pointSize <- 0.5

textSize <- 5

num_threads <- 0
```

# Function to perform t-SNE with PCA initialisation and create plots

```
doRtsne <- function(data, perplexity, pcaDim) {

  # Ensure data used for t-SNE is numeric and use the first pcaDim columns

  pca_data_subset <- data[, 1:pcaDim, drop = FALSE]
```

 # Perform t-SNE

```
  tsne <- Rtsne(pca_data_subset,

                dims = 2,

                perplexity = perplexity,

                verbose = TRUE,

                max_iter = max_iter,

                theta = theta,

                num_threads = num_threads)
```

  # Create a data frame with t-SNE results

```
  tsne_plot <- data.frame(x = tsne$Y[, 1], y = tsne$Y[, 2])


  plot <- ggplot(tsne_plot, aes(x = x, y = y)) +

    geom_point(size = pointSize, color = "blue") +  # Default color for all
points
```

```
    ggtitle(paste0("Perplexity=", perplexity, ", PCA_dimension=", pcaDim))
+
    xlab("Dimension 1") +
    ylab("Dimension 2") +
    theme_bw() +
    theme(text = element_text(size = textSize))

  return(plot)
}
```

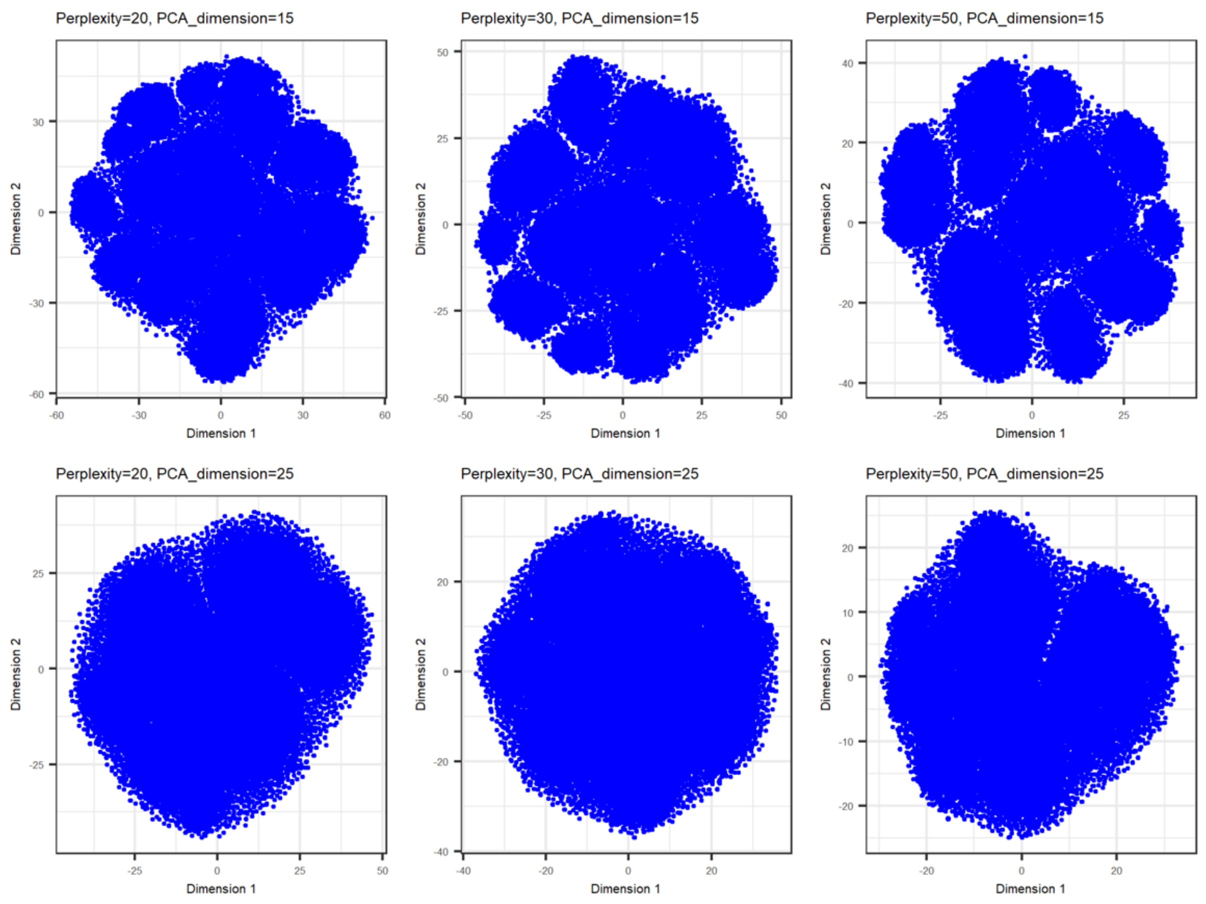
# Generate t-SNE plots for each combination of PCA dimensions and perplexities

```
plots <- list()
for (pcaDim in pcaDims) {
  for (perplexity in perplexities) {
    plot <- doRtsne(pca_data, perplexity, pcaDim)
    plots[[paste0("pcaDim_", pcaDim, "_perplexity_", perplexity)]] <- plot
  }
}
```

# Arrange plots in a grid

```
grid_plot_filename <- "tsne_2d_grid_combined_data.png"
png(grid_plot_filename, width = figWidth, height = figWidth * 0.75, units =
"px", res = 300)
grid.arrange(grobs = plots, nrow = length(pcaDims), ncol =
length(perplexities))
dev.off()
```

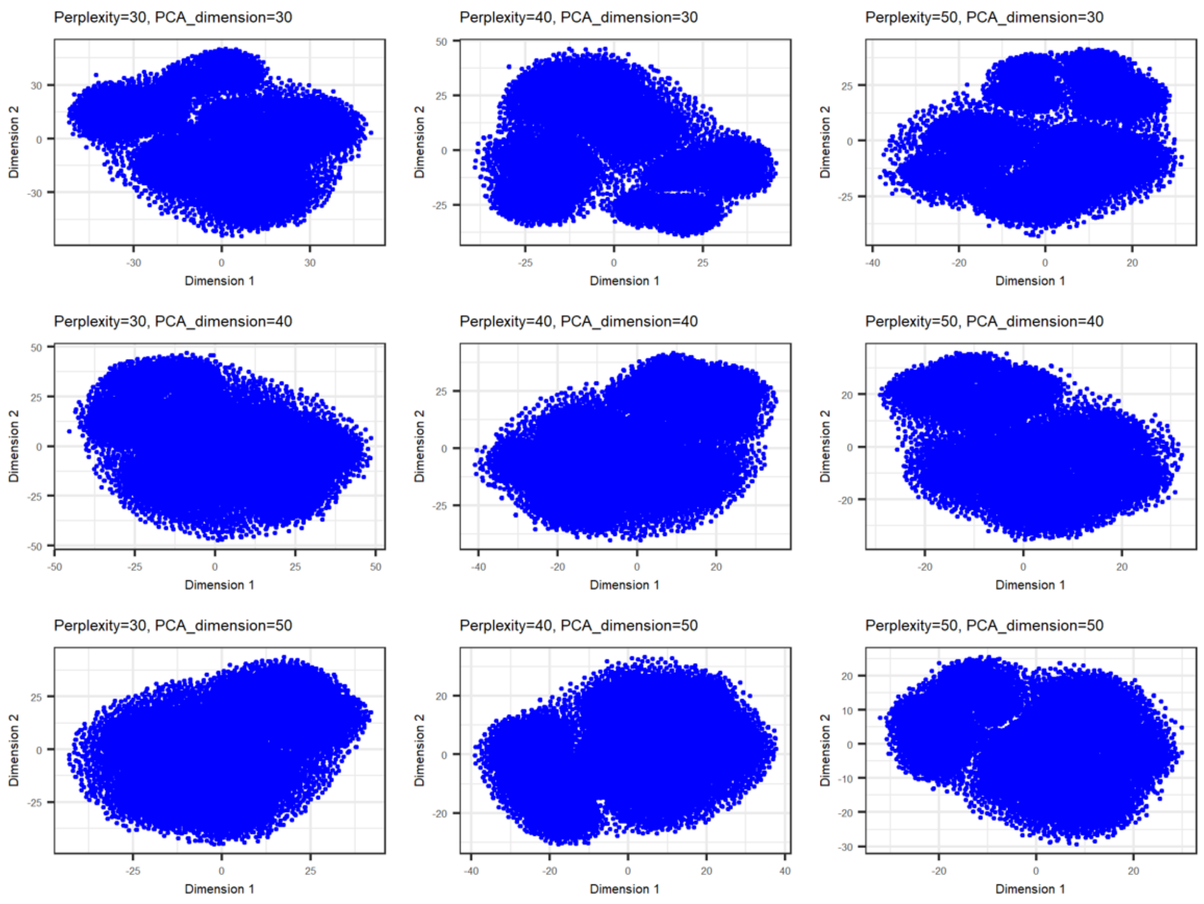
Run time: 2hrs (Max_iter: 1500)



Rerunning the analysis and experimenting with different perplexities and PCA_dims. Core changes: Max_iter 1,500 and

- `perplexities <- c(30, 40, 50)`
- `pcaDims <- c(30, 40, 50)`

```
Run time: 1hour 12mins

##Could the fact that I just put on the computer have made the computing
faster than expected?
```


Perplexity=30, PCA_dimension=30
Perplexity=40, PCA_dimension=30
Perplexity=50, PCA_dimension=30
Perplexity=30, PCA_dimension=40
Perplexity=40, PCA_dimension=40
Perplexity=50, PCA_dimension=40
Perplexity=30, PCA_dimension=50
Perplexity=40, PCA_dimension=50
Perplexity=50, PCA_dimension=50
Dimension 1
Dimension 2