```r
# Confirm working directory

getwd()


# Correctly read the CSV file into a data frame

data <- read.csv("20240725_ukb2.csv", header = TRUE, na.strings =
".")


# Dropping s2_groups_more from "data"

data <- select(data, -s2_groups_more)


# Moving s2_groups to third column

data <- data %>%

  select(

    1:2,

    s2_groups,

    everything()

  )


# Perform mean imputation

mean_data <- data %>%

  mutate(across(-c(1, 2, 3), ~ replace_na(., mean(., na.rm =
TRUE))))


# Encoding missing groups in s2_groups

data$s2_groups <- factor(data$s2_groups, levels = c("1", "2", "3",
NA))


# Perform one-hot encoding

data_encoded <- dummy_cols(data, select_columns = "s2_groups",
remove_first_dummy = FALSE)


# Remove the original 's2_groups' column

data_final <- data_encoded[, !names(data_encoded) %in% "s2_groups"]
```

# Identify continuous numeric columns for scaling (excluding one-hot encoded columns)

```
numeric_columns <- names(data_final)[sapply(data_final, is.numeric)
& !grepl("s2_groups_", names(data_final))]
```

# Scale and center numeric data for PCA

```
numeric_data <- data_final %>%
  select(all_of(numeric_columns)) %>%
  scale()
```

# Perform PCA

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

# Prepare PCA data

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

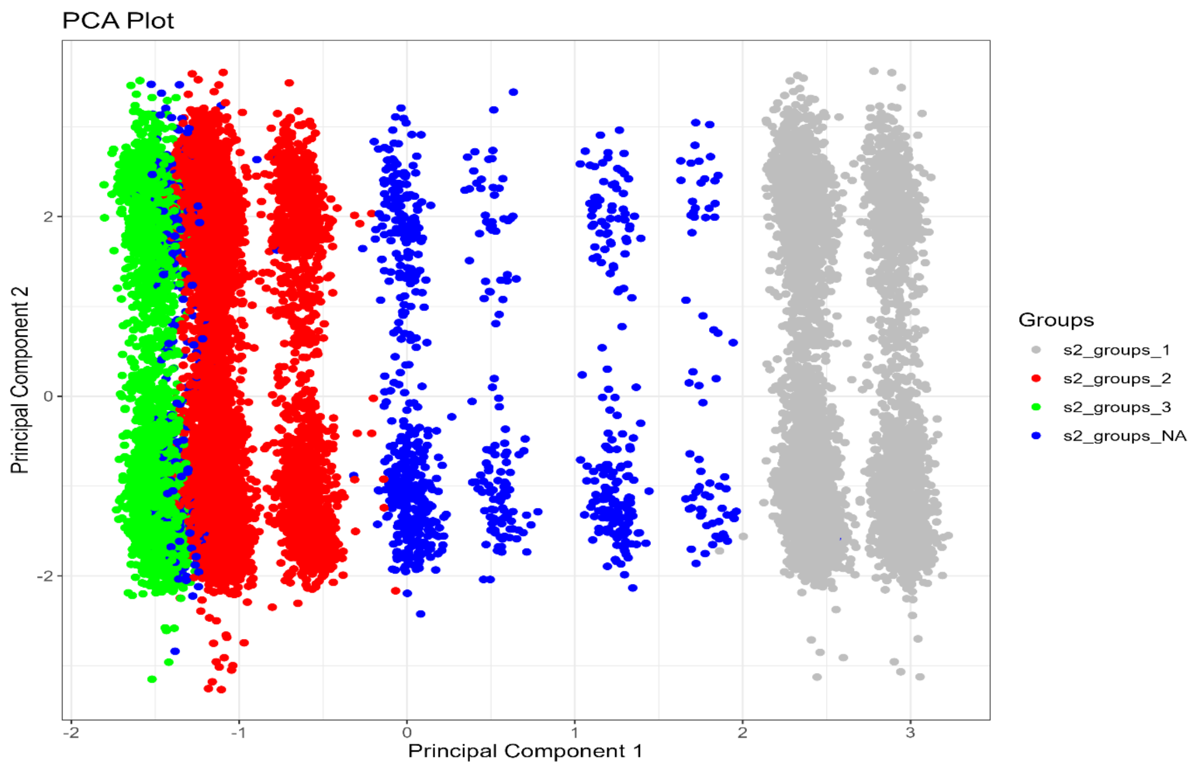
# Extract the group columns

```
group_columns <- c("s2_groups_1", "s2_groups_2", "s2_groups_3",
"s2_groups_NA")
pca_data$Groups <- apply(data_final[, group_columns], 1, function(x)
{
  if (!is.na(x["s2_groups_1"]) && x["s2_groups_1"] == 1)
return("s2_groups_1")
  if (!is.na(x["s2_groups_2"]) && x["s2_groups_2"] == 1)
return("s2_groups_2")
  if (!is.na(x["s2_groups_3"]) && x["s2_groups_3"] == 1)
return("s2_groups_3")
  if (!is.na(x["s2_groups_NA"]) && x["s2_groups_NA"] == 1)
return("s2_groups_NA")
  return(NA)  # Return NA for unknown or missing values
})
```

# Convert Groups to a factor and handle NA values correctly

```
pca_data$Groups <- factor(pca_data$Groups, levels = c("s2_groups_1",
"s2_groups_2", "s2_groups_3", "s2_groups_NA"))
```

```
# Remove rows with NA in Groups for plotting

pca_data <- pca_data[!is.na(pca_data$Groups), ]


# Plot PCA results

pca_plot <- ggplot(pca_data, aes(x = PC1, y = PC2, color = Groups))
+

  geom_point(size = 2) +

  scale_color_manual(values = c("s2_groups_1" = "gray",
"s2_groups_2" = "red", "s2_groups_3" = "green", "s2_groups_NA" =
"blue")) +

  labs(title = "PCA Plot", x = "Principal Component 1", y =
"Principal Component 2", color = "Groups") +

  theme_bw() +

  theme(text = element_text(size = 14), legend.key.size = unit(1.5,
"lines"))


# Save the PCA plot to a file

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



# Performing tSNE

# Set seed for reproducibility

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

# Define parameters

```
max_iter <- 1500  # Updated maximum iterations

theta <- 0.1

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

pcaDims <- c(2, 5, 10)

mycolors <- c("s2_groups_1" = "gray", "s2_groups_2" = "red",
"s2_groups_3" = "green", "s2_groups_NA" = "blue")

figWidth <- 2000

pointSize <- 0.5

legendSize <- 5

textSize <- 5

num_threads <- 0
```

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

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

  tsne <- Rtsne(data_final[, !(names(data_final) %in% c("FID_71392",
"FID_71392.1", "s2_groups_1", "s2_groups_2", "s2_groups_3",
"s2_groups_NA"))],

                initial_dims = pcaDim,

                dims = 2,

                perplexity = perplexity,

                verbose = TRUE,

                max_iter = max_iter,

                theta = theta,

                num_threads = num_threads)
```

# Recreate the Groups column for plotting

```
  tsne_plot <- data.frame(x = tsne$Y[, 1], y = tsne$Y[, 2], Groups =
factor(
    apply(data_final[, c("s2_groups_1", "s2_groups_2",
"s2_groups_3", "s2_groups_NA")], 1, function(x) {
      if (!is.na(x["s2_groups_1"]) && x["s2_groups_1"] == 1)
return("s2_groups_1")
      if (!is.na(x["s2_groups_2"]) && x["s2_groups_2"] == 1)
return("s2_groups_2")
      if (!is.na(x["s2_groups_3"]) && x["s2_groups_3"] == 1)
return("s2_groups_3")
      if (!is.na(x["s2_groups_NA"]) && x["s2_groups_NA"] == 1)
return("s2_groups_NA")
      return("Unknown")  # Return "Unknown" if no group matches
    })
  ))

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

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

# Iterate over datasets

```
for (datname in names(datasets)) {
  dat <- datasets[[datname]]
```

 # Check for required grouping columns

```
 if (all(c("s2_groups_1", "s2_groups_2", "s2_groups_3",
"s2_groups_NA") %in% names(dat))) {
```

  # Initialize a list to store plots

```
    pls <- list()
```

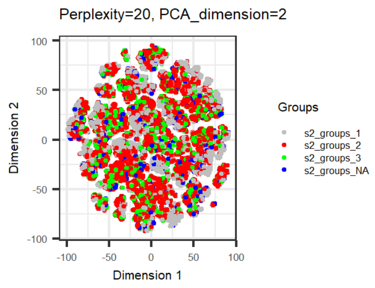
  # Perform t-SNE and plot for each PCA dimension and perplexity combination

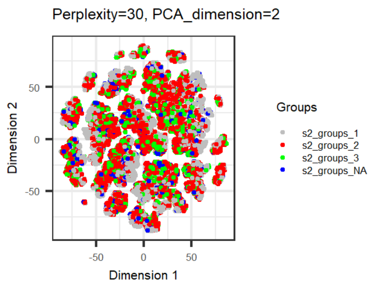
```
 for (pcaDim in pcaDims) {
      plots <- lapply(perplexities, function(perplexity)
doRtsne(perplexity, pcaDim))
      pls <- c(pls, plots)
    }
```

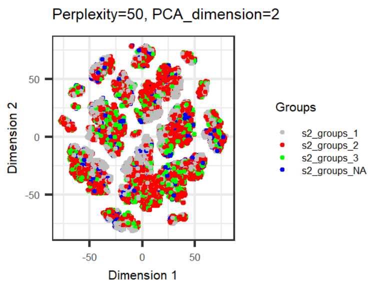
  # Arrange plots in a grid

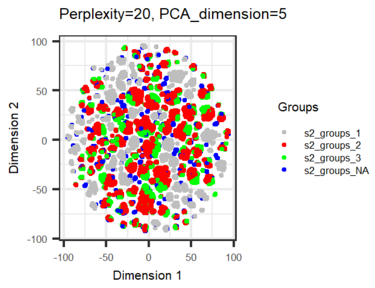
```
    grid_plot_filename <- paste0("tsne_2d_grid_", datname, ".png")
    png(grid_plot_filename, width = figWidth, height = figWidth *
0.75, units = "px", res = 300)
    grid.arrange(grobs = pls, nrow = length(pcaDims), ncol =
length(perplexities))
    dev.off()

  } else {
    warning(paste("Some one-hot encoded columns are missing in",
datname))
  }
}
```

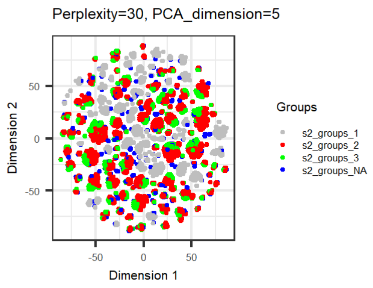
tSNE and grouping with s2_groups

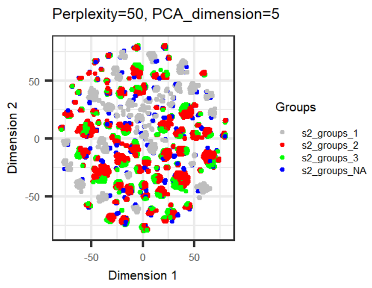


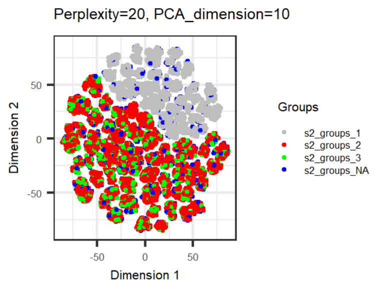


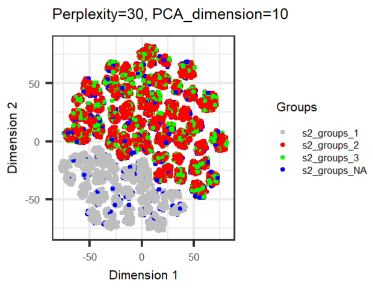


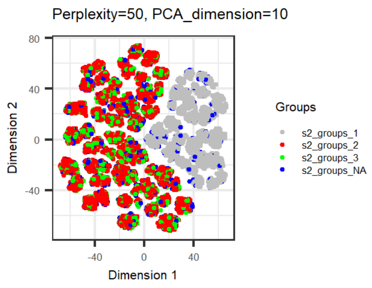












#Data cleaning to perform tSNE and group with s2_groups_more

# Correctly read the CSV file into a data frame

```
data <- read.csv("20240725_ukb2.csv", header = TRUE, na.strings = 
".")
```

#Creating a replicate of the data set since I want to compare two grouping (S2_groups with S2_groups_more)

```
datamore <- read.csv("20240725_ukb2.csv", header = TRUE, na.strings 
= ".")
```

# Drop s2_groups from "datamore" using dplyr

```
datamore <- select(datamore, -s2_groups)
```

# Move s2_groups_more to the third column for easy data manipulation

```
datamore <- datamore %>%

  select(

    1:2,

    s2_groups_more,

    everything()

  )
```

# Perform mean imputation

```
mean_data <- datamore %>%

  mutate(across(-c(1, 2, 3), ~ replace_na(., mean(., na.rm = 
TRUE))))
```

# Encoding missing groups in s2_groups_more

```
datamore$s2_groups_more <- factor(datamore$s2_groups_more, levels = 
c("1", "2", "3", NA))
```

# Perform one-hot encoding

```r
datamore_encoded <- dummy_cols(datamore, select_columns = 
"s2_groups_more", remove_first_dummy = FALSE)
```

# Remove the original 's2_groups_more' column

```r
datamore_final <- datamore_encoded[, !names(datamore_encoded) %in% 
"s2_groups_more"]
```

# Identify continuous numeric columns for scaling (excluding one-hot encoded columns)

```r
numeric_columns <- names(datamore_final)[sapply(datamore_final, 
is.numeric) & !grepl("s2_groups_more_", names(datamore_final))]
```

# Scale and center numeric data for PCA

```r
numeric_data <- datamore_final %>%

  select(all_of(numeric_columns)) %>%

  scale()
```

# Perform PCA

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

# Prepare PCA data

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

# Extract the group columns

```r
group_columns <- c("s2_groups_more_1", "s2_groups_more_2", 
"s2_groups_more_3", "s2_groups_more_NA")

pca_data$Groups <- apply(datamore_final[, group_columns], 1, 
function(x) {

  if (!is.na(x["s2_groups_more_1"]) && x["s2_groups_more_1"] == 1) 
return("s2_groups_more_1")

  if (!is.na(x["s2_groups_more_2"]) && x["s2_groups_more_2"] == 1) 
return("s2_groups_more_2")

  if (!is.na(x["s2_groups_more_3"]) && x["s2_groups_more_3"] == 1) 
return("s2_groups_more_3")

  if (!is.na(x["s2_groups_more_NA"]) && x["s2_groups_more_NA"] == 1) 
return("s2_groups_more_NA")

  return(NA)  # Return NA for unknown or missing values
})
```

# Convert Groups to a factor and handle NA values correctly

```
pca_data$Groups <- factor(pca_data$Groups, levels =
c("s2_groups_more_1", "s2_groups_more_2", "s2_groups_more_3",
"s2_groups_more_NA"))
```

# Remove rows with NA in Groups for plotting

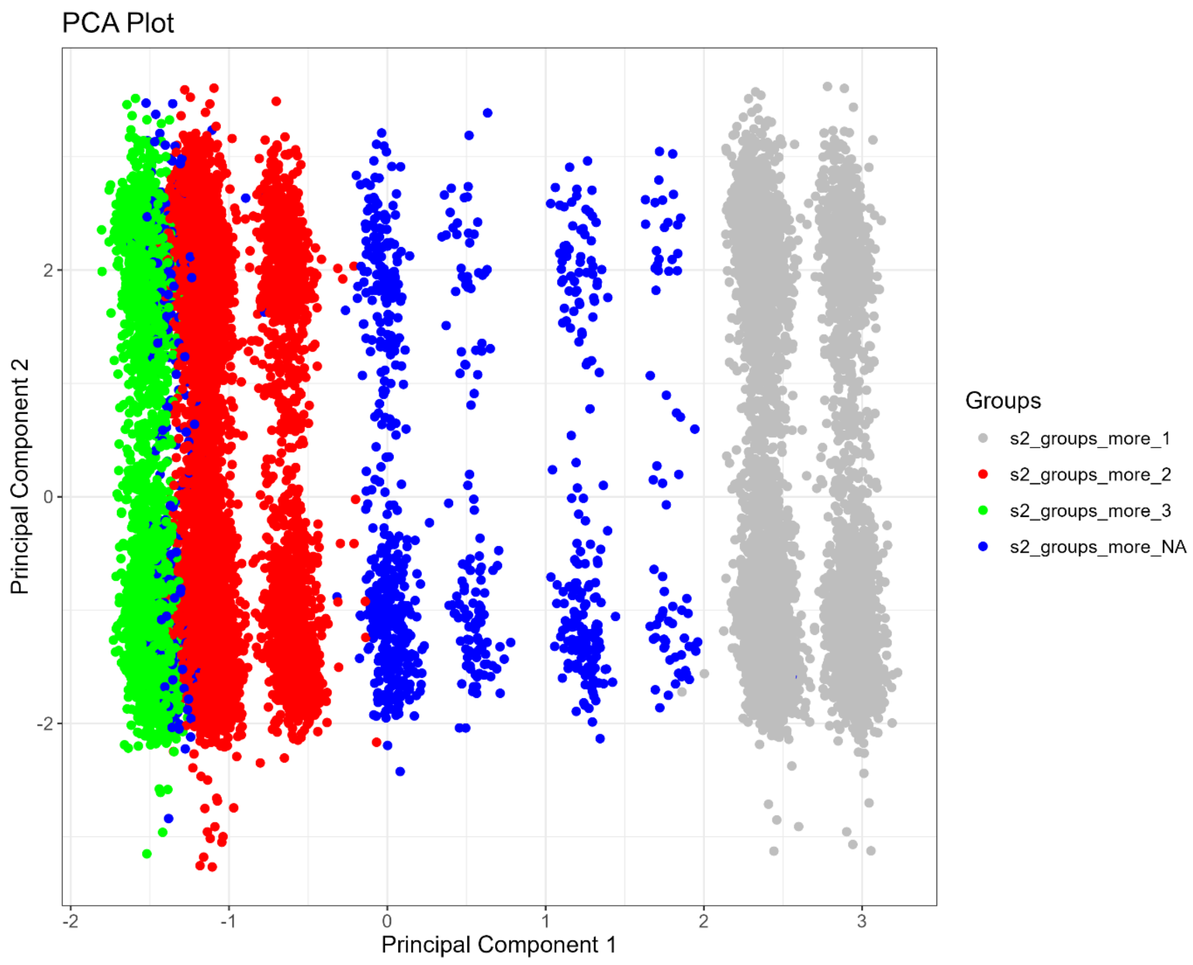
```
pca_data <- pca_data[!is.na(pca_data$Groups), ]
```

# Plot PCA results

```
pca_plot <- ggplot(pca_data, aes(x = PC1, y = PC2, color = Groups))
+

  geom_point(size = 2) +

  scale_color_manual(values = c("s2_groups_more_1" = "gray",
"s2_groups_more_2" = "red", "s2_groups_more_3" = "green",
"s2_groups_more_NA" = "blue")) +

  labs(title = "PCA Plot", x = "Principal Component 1", y =
"Principal Component 2", color = "Groups") +

  theme_bw() +

  theme(text = element_text(size = 14), legend.key.size = unit(1.5,
"lines"))
```

# Save the PCA plot to a file

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



#Performing tSNE

# Set seed for reproducibility

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

# Define parameters

```
max_iter <- 1000  # Maximum iterations for t-SNE

theta <- 0.1

perplexities <- c(10, 25, 50)  # Perplexity values

pcaDims <- c(2, 5, 10)

mycolors <- c("s2_groups_more_1" = "gray", "s2_groups_more_2" =
"red", "s2_groups_more_3" = "green", "s2_groups_more_NA" = "blue")

figWidth <- 2000

pointSize <- 0.5

legendSize <- 5
```

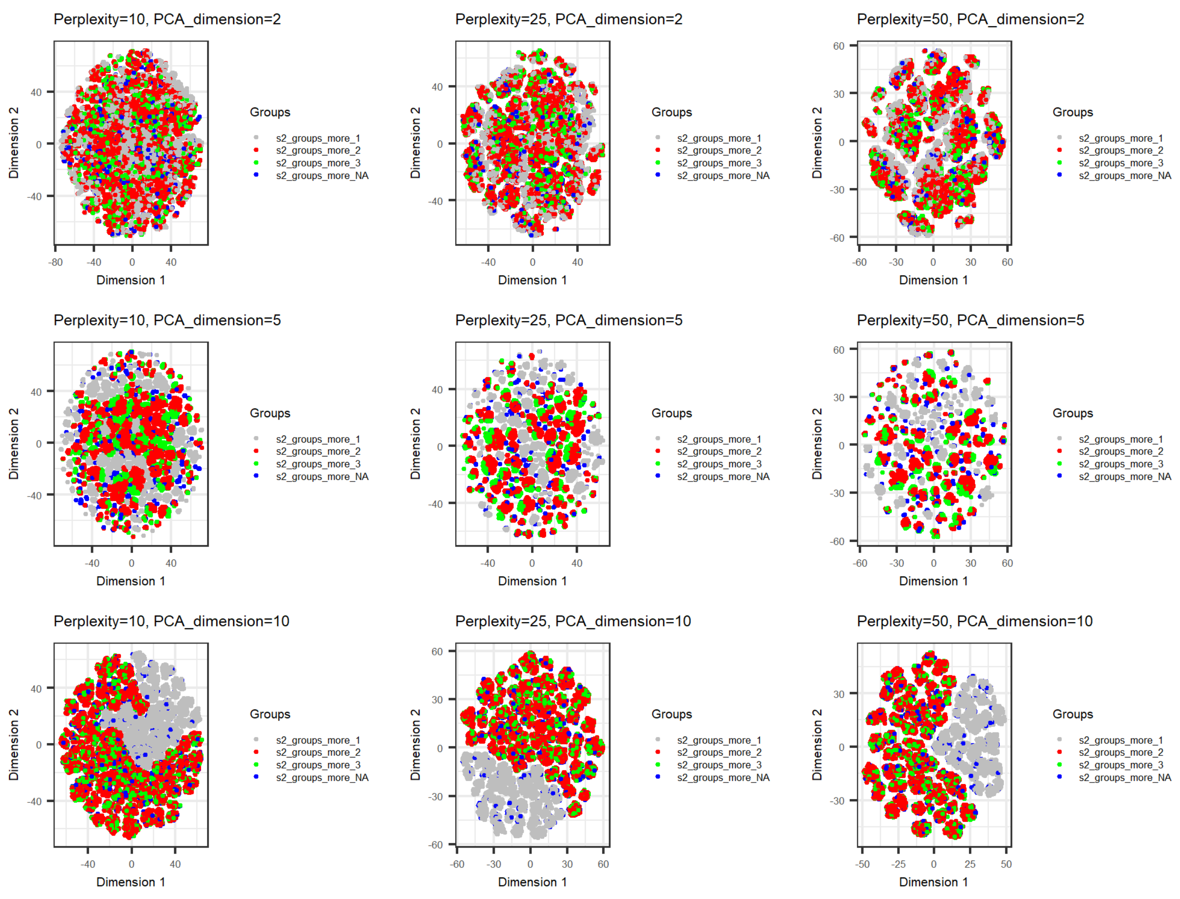
```
textSize <- 5

num_threads <- 0
```

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

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

  tsne <- Rtsne(data_final[, !(names(data_final) %in% c("FID_71392",
"FID_71392.1", "s2_groups_more_1", "s2_groups_more_2",
"s2_groups_more_3", "s2_groups_more_NA"))],

                initial_dims = pcaDim,

                dims = 2,

                perplexity = perplexity,

                verbose = TRUE,

                max_iter = max_iter,

                theta = theta,

                num_threads = num_threads)
```

 # Recreate the Groups column for plotting

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

    apply(data_final[, c("s2_groups_more_1", "s2_groups_more_2",
"s2_groups_more_3", "s2_groups_more_NA")], 1, function(x) {

      if (!is.na(x["s2_groups_more_1"]) && x["s2_groups_more_1"] ==
1) return("s2_groups_more_1")

      if (!is.na(x["s2_groups_more_2"]) && x["s2_groups_more_2"] ==
1) return("s2_groups_more_2")

      if (!is.na(x["s2_groups_more_3"]) && x["s2_groups_more_3"] ==
1) return("s2_groups_more_3")

      if (!is.na(x["s2_groups_more_NA"]) && x["s2_groups_more_NA"]
== 1) return("s2_groups_more_NA")

      return("Unknown")  # Return "Unknown" if no group matches

    })

  ))


  plot <- ggplot(tsne_plot) +

    geom_point(aes(x = x, y = y, color = Groups), 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)
}

# Define your datasets
datasets <- list(
  data_final = data_final
)

# Iterate over datasets
for (datname in names(datasets)) {
  dat <- datasets[[datname]]

  # Check for required grouping columns
  if (all(c("s2_groups_more_1", "s2_groups_more_2", 
"s2_groups_more_3", "s2_groups_more_NA") %in% names(dat))) {

    # 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(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()

  } else {
    warning(paste("Some one-hot encoded columns are missing in", datname))
  }
}
```


Perplexity=10, PCA_dimension=2
Perplexity=25, PCA_dimension=2
Perplexity=50, PCA_dimension=2
Perplexity=10, PCA_dimension=5
Perplexity=25, PCA_dimension=5
Perplexity=50, PCA_dimension=5
Perplexity=10, PCA_dimension=10
Perplexity=25, PCA_dimension=10
Perplexity=50, PCA_dimension=10
Dimension 1
Dimension 2
Groups
s2_groups_more_1
s2_groups_more_2
s2_groups_more_3
s2_groups_more_NA