## Set Up of Environment

# adjust it to all the other packages

if (requireNamespace("thematic"))

thematic::thematic\_rmd(font = "auto")

today <- Sys.Date()

output\_location <- paste(today,"\_Stroke\_Results", sep="")

setwd("/Users/ju5263ta/Github/Monocytes/Data/")

if (!dir.exists(output\_location)) {

dir.create(output\_location, recursive = TRUE)

}

getwd ()

rm(list=ls())

library(ggplot2)

library(tidyverse)

library(stats)

library(skimr)

library(sjPlot)

library(readxl)

library(thematic)

library(knitr)

library(lme4)

library(ggpubr)

library(dplyr)

library(reshape2)

library(pheatmap)

library(stringr)

library(colorRamp2)

library(rstatix)

library(scales)

#library(emmeans)

library(pwr)

library(lmtest)

library(writexl)

library(openxlsx)

#library(nondetects)

library(skimr)

library(visdat)

library(mice)

library(lmerTest)

library(beepr)

#library(Seurat)

thematic\_on(bg = "white", fg = "black", accent = "blue")

### \*Functions\* ---------------------------------------------------------------------

my\_colors <- c("darkgreen", "orange", "red", "magenta", "purple")

my\_grey\_scale <- c("grey", "black", "black", "black", "black")

## Create Folder

createFolder <- function(folderName) {

if (!dir.exists(folderName)) {

dir.create(folderName, recursive = TRUE)

}

}

## Split data

createSeparatedDataFiles <- function(data, categoryColumn, categoryFolder) {

uniqueCategories <- unique(data[[categoryColumn]])

for (category in uniqueCategories) {

categoryData <- data %>% filter(data[[categoryColumn]] == category)

categoryName <- as.character(category)

filename <- file.path(categoryFolder, paste0(categoryName, ".txt"))

write.table(categoryData, filename, sep = "\t", row.names = FALSE)

assign(categoryName, categoryData, envir = .GlobalEnv)

}

return(uniqueCategories)

}

## Function to get the failes samples

getMeTheFailedSamples<- function(data, gene, cutoff) {

# Extract the Sample IDs where the gene's value exceeds the cutoff

unique(data$Sample[which(data$Gene == gene & (data$CT\_Value > cutoff | is.na(data$CT\_Value)))])

}

## Function to perform correlation and linear regression by Timepoint

calculate\_linreg\_by\_timepoint <- function(data, cells\_colname) {

# Initialize an empty list to store results for each Timepoint

results\_list <- list()

# Loop through each unique Timepoint and perform the correlation and regression

for (timepoint in unique(data$Timepoint)) {

# Subset data for the current Timepoint

data\_subset <- subset(data, Timepoint == timepoint)

# Pearson correlation

cor\_test <- cor.test(data\_subset[[cells\_colname]], data\_subset$Age)

# Linear regression model

lm\_result <- lm(data\_subset[[cells\_colname]] ~ data\_subset$Age)

coef\_x <- summary(lm\_result)$coefficients[2, 1]

ci <- confint(lm\_result)[2, ]

intercept <- summary(lm\_result)$coefficients[1, 1]

# Calculate Fold Induction

Max <- max(data\_subset$Age, na.rm = TRUE)

Min <- min(data\_subset$Age, na.rm = TRUE)

FinalValue <- coef\_x \* Max + intercept

InitialValue <- coef\_x \* Min + intercept

Fold <- FinalValue / InitialValue

# Store the results in a data frame for this Timepoint

result <- data.frame(Cells = cells\_colname,

Timepoint = timepoint,

N = length(data\_subset[[cells\_colname]]),

p.value = cor\_test$p.value,

Coefficient = coef\_x,

Down95 = ci[1],

Up95 = ci[2],

Intercept = intercept,

Fold = Fold,

Max\_age = Max,

Min\_age = Min)

# Append to the results list

results\_list[[timepoint]] <- result

}

# Combine all results into a single data frame

final\_results <- do.call(rbind, results\_list)

return(final\_results)

}

# Function to plot linear regression for a specific group

plot\_linear\_regression1 <- function(data, cells\_colname, group\_name, output\_folder) {

# Fit the initial linear model to identify outliers

lm\_initial <- lm(data[[cells\_colname]] ~ data$Age)

residuals\_values <- residuals(lm\_initial)

outlier\_indices <- which(abs(residuals\_values) > (3 \* sd(residuals\_values)))

# Filter out the outliers

data\_filtered <- data[-outlier\_indices, ]

# Check if there are enough data points to fit the model

if (nrow(data\_filtered) < 2) {

warning(paste("Not enough data points after outlier removal for", group\_name, "(", cells\_colname, "). Skipping plot."))

return(NULL)

}

# Calculate linear regression results

rows\_group <- calculate\_linreg\_by\_timepoint(data\_filtered, cells\_colname)

if (is.null(rows\_group) || nrow(rows\_group) == 0) {

warning(paste("No results for", group\_name, "(", cells\_colname, "). Skipping plot."))

return(NULL)

}

rows\_group$Focus <- group\_name # Add the group label

Age\_correlation\_FACS <<- rbind(Age\_correlation\_FACS, rows\_group) # Update global variable

# Plot regression for the specific group with all timepoints

p\_group <- ggplot(data\_filtered, aes(x = Age, y = .data[[cells\_colname]], color = Timepoint)) +

geom\_point(size = 2) +

geom\_smooth(method = "lm", aes(color = Timepoint), se = FALSE, size = 1) +

ggtitle(paste("Linear Regression -", group\_name, "(", cells\_colname, ")", sep = " ")) +

xlab("Age") +

ylab(cells\_colname) +

theme(text = element\_text(size = 14)) +

theme(

panel.background = element\_rect(fill = "white", color = "black"),

plot.background = element\_rect(fill = "white", color = "black"),

text = element\_text(color = "black"),

panel.grid.major = element\_line(color = "gray", size = 0.2),

panel.grid.minor = element\_blank(),

panel.grid.major.x = element\_blank()

) +

scale\_color\_manual(values = my\_colors) +

scale\_fill\_manual(values = my\_colors)

# Add confidence interval only if p-value is <= 0.05

for (timepoint in unique(data\_filtered$Timepoint)) {

if (rows\_group$p.value[rows\_group$Timepoint == timepoint] <= 0.05) {

p\_group <- p\_group + geom\_smooth(data = subset(data\_filtered, Timepoint == timepoint),

method = "lm",

aes(fill = Timepoint),

se = TRUE,

alpha = 0.25,

color = NA)

}

# Save the plot with all timepoints

ggsave(filename = file.path(output\_folder, paste0("LinReg\_", group\_name, "\_", cells\_colname, ".png")), plot = p\_group)

p\_individual <- ggplot(data\_filtered %>% filter(Timepoint == timepoint),

aes(x = Age, y = .data[[cells\_colname]], color = Timepoint)) +

geom\_point(size = 2) +

geom\_smooth(method = "lm", aes(color = Timepoint), se = TRUE, size = 1) +

ggtitle(paste("Linear Regression -", group\_name, "(", cells\_colname, ")", " - Timepoint:", timepoint)) +

xlab("Age") +

ylab(cells\_colname) +

theme(text = element\_text(size = 14)) +

theme(

panel.background = element\_rect(fill = "white", color = "black"),

plot.background = element\_rect(fill = "white", color = "black"),

text = element\_text(color = "black"),

panel.grid.major = element\_line(color = "gray", size = 0.2),

panel.grid.minor = element\_blank(),

panel.grid.major.x = element\_blank()

) +

scale\_color\_manual(values = my\_colors[timepoint]) +

scale\_fill\_manual(values = my\_colors[timepoint])

# Save the individual plot

ggsave(filename = file.path(output\_folder, paste0("LinReg\_", group\_name, "\_", cells\_colname, "\_Timepoint\_", timepoint, ".png")), plot = p\_individual,

width = 7, height = 6)

}

# Create an additional plot with only significant timepoints if there are any

significant\_timepoints <- rows\_group$Timepoint[rows\_group$p.value <= 0.05]

if (length(significant\_timepoints) > 0) {

# Plot with all significant timepoints together

p\_significant <- ggplot(data\_filtered %>% filter(Timepoint %in% significant\_timepoints),

aes(x = Age, y = .data[[cells\_colname]], color = Timepoint)) +

geom\_point(size = 2) +

geom\_smooth(method = "lm", aes(color = Timepoint), se = TRUE, size = 1) +

ggtitle(paste("Linear Regression - Significant Only -", group\_name, "(", cells\_colname, ")", sep = " ")) +

xlab("Age") +

ylab(cells\_colname) +

theme(text = element\_text(size = 14)) +

theme(

panel.background = element\_rect(fill = "white", color = "black"),

plot.background = element\_rect(fill = "white", color = "black"),

text = element\_text(color = "black"),

panel.grid.major = element\_line(color = "gray", size = 0.2),

panel.grid.minor = element\_blank(),

panel.grid.major.x = element\_blank()

) +

scale\_color\_manual(values = my\_colors) +

scale\_fill\_manual(values = my\_colors)

# Save the plot with only significant timepoints together

ggsave(filename = file.path(output\_folder, paste0("LinReg\_SignificantOnly\_", group\_name, "\_", cells\_colname, ".png")), plot = p\_significant)

# Plot individual plots for each significant timepoint

for (sig\_timepoint in significant\_timepoints) {

p\_individual <- ggplot(data\_filtered %>% filter(Timepoint == sig\_timepoint),

aes(x = Age, y = .data[[cells\_colname]], color = Timepoint)) +

geom\_point(size = 2) +

geom\_smooth(method = "lm", aes(color = Timepoint), se = TRUE, size = 1) +

ggtitle(paste("Linear Regression -", group\_name, "(", cells\_colname, ")", " - Timepoint:", sig\_timepoint)) +

xlab("Age") +

ylab(cells\_colname) +

theme(text = element\_text(size = 14)) +

theme(

panel.background = element\_rect(fill = "white", color = "black"),

plot.background = element\_rect(fill = "white", color = "black"),

text = element\_text(color = "black"),

panel.grid.major = element\_line(color = "gray", size = 0.2),

panel.grid.minor = element\_blank(),

panel.grid.major.x = element\_blank()

) +

scale\_color\_manual(values = my\_colors[sig\_timepoint]) +

scale\_fill\_manual(values = my\_colors[sig\_timepoint])

# Save the individual plot

ggsave(filename = file.path(output\_folder, paste0("LinReg\_SignificantOnly\_", group\_name, "\_", cells\_colname, "\_Timepoint\_", sig\_timepoint, ".png")), plot = p\_individual)

}

}

}

# Function to plot linear regression for a specific group

plot\_linear\_regression <- function(data, cells\_colname, group\_name, output\_folder) {

# Fit the initial linear model to identify outliers

lm\_initial <- lm(data[[cells\_colname]] ~ data$Age)

residuals\_values <- residuals(lm\_initial)

outlier\_indices <- which(abs(residuals\_values) > (3 \* sd(residuals\_values)))

# Filter out the outliers

data\_filtered <- data[-outlier\_indices, ]

# Check if there are enough data points to fit the model

if (nrow(data\_filtered) < 2) {

warning(paste("Not enough data points after outlier removal for", group\_name, "(", cells\_colname, "). Skipping plot."))

return(NULL)

}

# Calculate linear regression results

rows\_group <- calculate\_linreg\_by\_timepoint(data\_filtered, cells\_colname)

if (is.null(rows\_group) || nrow(rows\_group) == 0) {

warning(paste("No results for", group\_name, "(", cells\_colname, "). Skipping plot."))

return(NULL)

}

rows\_group$Focus <- group\_name # Add the group label

Age\_correlation\_FACS <<- rbind(Age\_correlation\_FACS, rows\_group) # Update global variable

# Plot regression for the specific group with all timepoints

p\_group <- ggplot(data\_filtered, aes(x = Age, y = .data[[cells\_colname]], color = Timepoint)) +

geom\_point(size = 2) +

geom\_smooth(method = "lm", aes(color = Timepoint), se = FALSE, size = 1) +

ggtitle(paste("Linear Regression -", group\_name, "(", cells\_colname, ")", sep = " ")) +

xlab("Age") +

ylab(cells\_colname) +

theme(text = element\_text(size = 14)) +

theme(

panel.background = element\_rect(fill = "white", color = "black"),

plot.background = element\_rect(fill = "white", color = "black"),

text = element\_text(color = "black"),

panel.grid.major = element\_line(color = "gray", size = 0.2),

panel.grid.minor = element\_blank(),

panel.grid.major.x = element\_blank()

) +

scale\_color\_manual(values = my\_colors) +

scale\_fill\_manual(values = my\_colors)

# Add confidence interval only if p-value is <= 0.05

for (timepoint in unique(data\_filtered$Timepoint)) {

if (rows\_group$p.value[rows\_group$Timepoint == timepoint] <= 0.05) {

p\_group <- p\_group + geom\_smooth(data = subset(data\_filtered, Timepoint == timepoint),

method = "lm",

aes(fill = Timepoint),

se = TRUE,

alpha = 0.25,

color = NA)

}

# Save the plot with all timepoints

ggsave(filename = file.path(output\_folder, paste0("LinReg\_", group\_name, "\_", cells\_colname, ".png")), plot = p\_group)

p\_individual <- ggplot(data\_filtered %>% filter(Timepoint == timepoint),

aes(x = Age, y = .data[[cells\_colname]], color = Timepoint)) +

geom\_point(size = 2) +

geom\_smooth(method = "lm", aes(color = Timepoint), se = TRUE, size = 1) +

ggtitle(paste("Linear Regression -", group\_name, "(", cells\_colname, ")", " - Timepoint:", timepoint)) +

xlab("Age") +

ylab(cells\_colname) +

theme(text = element\_text(size = 14)) +

theme(

panel.background = element\_rect(fill = "white", color = "black"),

plot.background = element\_rect(fill = "white", color = "black"),

text = element\_text(color = "black"),

panel.grid.major = element\_line(color = "gray", size = 0.2),

panel.grid.minor = element\_blank(),

panel.grid.major.x = element\_blank()

) +

scale\_color\_manual(values = my\_colors[timepoint]) +

scale\_fill\_manual(values = my\_colors[timepoint])

# Save the individual plot

ggsave(filename = file.path(output\_folder, paste0("LinReg\_", group\_name, "\_", cells\_colname, "\_Timepoint\_", timepoint, ".png")), plot = p\_individual,

width = 8, height = 5)

}

# Create an additional plot with only significant timepoints if there are any

significant\_timepoints <- rows\_group$Timepoint[rows\_group$p.value <= 0.05]

if (length(significant\_timepoints) > 0) {

# Plot with all significant timepoints together

p\_significant <- ggplot(data\_filtered %>% filter(Timepoint %in% significant\_timepoints),

aes(x = Age, y = .data[[cells\_colname]], color = Timepoint)) +

geom\_point(size = 2) +

geom\_smooth(method = "lm", aes(color = Timepoint), se = TRUE, size = 1) +

ggtitle(paste("Linear Regression - Significant Only -", group\_name, "(", cells\_colname, ")", sep = " ")) +

xlab("Age") +

ylab(cells\_colname) +

theme(text = element\_text(size = 14)) +

theme(

panel.background = element\_rect(fill = "white", color = "black"),

plot.background = element\_rect(fill = "white", color = "black"),

text = element\_text(color = "black"),

panel.grid.major = element\_line(color = "gray", size = 0.2),

panel.grid.minor = element\_blank(),

panel.grid.major.x = element\_blank()

) +

scale\_color\_manual(values = my\_colors) +

scale\_fill\_manual(values = my\_colors)

# Save the plot with only significant timepoints together

ggsave(filename = file.path(output\_folder, paste0("LinReg\_SignificantOnly\_", group\_name, "\_", cells\_colname, ".png")), plot = p\_significant)

# Plot individual plots for each significant timepoint

for (sig\_timepoint in significant\_timepoints) {

p\_individual <- ggplot(data\_filtered %>% filter(Timepoint == sig\_timepoint),

aes(x = Age, y = .data[[cells\_colname]], color = Timepoint)) +

geom\_point(size = 2) +

geom\_smooth(method = "lm", aes(color = Timepoint), se = TRUE, size = 1) +

ggtitle(paste("Linear Regression -", group\_name, "(", cells\_colname, ")", " - Timepoint:", sig\_timepoint)) +

xlab("Age") +

ylab(cells\_colname) +

theme(text = element\_text(size = 14)) +

theme(

panel.background = element\_rect(fill = "white", color = "black"),

plot.background = element\_rect(fill = "white", color = "black"),

text = element\_text(color = "black"),

panel.grid.major = element\_line(color = "gray", size = 0.2),

panel.grid.minor = element\_blank(),

panel.grid.major.x = element\_blank()

) +

scale\_color\_manual(values = my\_colors[sig\_timepoint]) +

scale\_fill\_manual(values = my\_colors[sig\_timepoint])

# Save the individual plot

ggsave(filename = file.path(output\_folder, paste0("LinReg\_SignificantOnly\_", group\_name, "\_", cells\_colname, "\_Timepoint\_", sig\_timepoint, ".png")), plot = p\_individual)

}

}

}

plot\_linear\_regression\_NHISS <- function(data, cells\_colname, group\_name, output\_folder, result, x\_axis\_var) {

# Fit the initial linear model to identify outliers

lm\_initial <- lm(data[[cells\_colname]] ~ data[[x\_axis\_var]])

residuals\_values <- residuals(lm\_initial)

outlier\_indices <- which(abs(residuals\_values) > (3 \* sd(residuals\_values)))

# Filter out the outliers

data\_filtered <- data[-outlier\_indices, ]

# Check if there are enough data points to fit the model

if (nrow(data\_filtered) < 2) {

warning(paste("Not enough data points after outlier removal for", group\_name, "(", cells\_colname, "). Skipping plot."))

return(NULL)

}

# Calculate linear regression results on filtered data

rows\_group <- calculate\_linreg\_by\_timepoint(data\_filtered, cells\_colname)

if (is.null(rows\_group) || nrow(rows\_group) == 0) {

warning(paste("No results for", group\_name, "(", cells\_colname, "). Skipping plot."))

return(NULL)

}

rows\_group$Focus <- group\_name # Add the group label

result <- rbind(result, rows\_group) # Update global variable

# Plot regression for the specific group with all timepoints

p\_group <- ggplot(data\_filtered, aes\_string(x = x\_axis\_var, y = cells\_colname, color = "Timepoint")) +

geom\_point(size = 2) +

geom\_smooth(method = "lm", aes(color = Timepoint), se = FALSE, size = 1) +

ggtitle(paste("Linear Regression -", group\_name, "(", cells\_colname, ")", sep = " ")) +

xlab(x\_axis\_var) +

ylab(cells\_colname) +

theme(text = element\_text(size = 14)) +

theme(

panel.background = element\_rect(fill = "white", color = "black"),

plot.background = element\_rect(fill = "white", color = "black"),

text = element\_text(color = "black"),

panel.grid.major = element\_line(color = "gray", size = 0.2),

panel.grid.minor = element\_blank(),

panel.grid.major.x = element\_blank()

) +

scale\_color\_manual(values = my\_colors) +

scale\_fill\_manual(values = my\_colors)

# Add confidence interval only if p-value is <= 0.05

for (timepoint in unique(data\_filtered$Timepoint)) {

if (rows\_group$p.value[rows\_group$Timepoint == timepoint] <= 0.05) {

p\_group <- p\_group + geom\_smooth(data = subset(data\_filtered, Timepoint == timepoint),

method = "lm",

aes(fill = Timepoint),

se = TRUE,

alpha = 0.25,

color = NA)

}

}

# Save the plot with all timepoints

ggsave(filename = file.path(output\_folder, paste0("LinReg\_", x\_axis\_var, "\_", group\_name, "\_", cells\_colname, ".png")), plot = p\_group)

# Create an additional plot with only significant timepoints if there are any

significant\_timepoints <- rows\_group$Timepoint[rows\_group$p.value <= 0.05]

if (length(significant\_timepoints) > 0) {

# Plot with all significant timepoints together

p\_significant <- ggplot(data\_filtered %>% filter(Timepoint %in% significant\_timepoints),

aes\_string(x = x\_axis\_var, y = cells\_colname, color = "Timepoint")) +

geom\_point(size = 2) +

geom\_smooth(method = "lm", aes(color = Timepoint), se = TRUE, size = 1) +

ggtitle(paste("Linear Regression - Significant Only -", group\_name, "(", cells\_colname, ")", sep = " ")) +

xlab(x\_axis\_var) +

ylab(cells\_colname) +

theme(text = element\_text(size = 14)) +

theme(

panel.background = element\_rect(fill = "white", color = "black"),

plot.background = element\_rect(fill = "white", color = "black"),

text = element\_text(color = "black"),

panel.grid.major = element\_line(color = "gray", size = 0.2),

panel.grid.minor = element\_blank(),

panel.grid.major.x = element\_blank()

) +

scale\_color\_manual(values = my\_colors) +

scale\_fill\_manual(values = my\_colors)

# Save the plot with only significant timepoints together

ggsave(filename = file.path(output\_folder, paste0("LinReg\_SignificantOnly\_", x\_axis\_var, "\_", group\_name, "\_", cells\_colname, ".png")), plot = p\_significant)

}

return(result)

}

# summary data.frame for dotplot

calculate\_summary <- function(data, combo\_col, gene\_col) {

# Calculate mean\_capZ

mean\_summary <- data %>%

group\_by(!!sym(combo\_col), !!sym(gene\_col)) %>%

summarize(

mean\_capZ = mean(mean\_capZ, na.rm = TRUE),

count = n()

) %>%

filter(count > 1)

# Calculate sd\_mean\_capZ

sd\_summary <- data %>%

group\_by(!!sym(combo\_col), !!sym(gene\_col)) %>%

summarize(

sd\_mean\_capZ = sd(mean\_capZ, na.rm = TRUE) # Calculate standard deviation of mean\_capZ

)

# Combine both summaries

final\_summary <- mean\_summary %>%

left\_join(sd\_summary, by = c(combo\_col, gene\_col)) %>%

mutate(

Dot\_Size = scales::rescale(sd\_mean\_capZ, to = c(5, 1)) # Rescale SD to range 5 (low SD) to 1 (high SD)

)

return(final\_summary)

}

check\_sample\_counts <- function(data, timepoint\_col = "Timepoint", sampleid\_col = "SampleID") {

# List unique SampleID values for each timepoint

unique\_samples\_tp0 <- unique(data[[sampleid\_col]][data[[timepoint\_col]] == "TP0"])

unique\_samples\_tp1 <- unique(data[[sampleid\_col]][data[[timepoint\_col]] == "TP1"])

unique\_samples\_tp2 <- unique(data[[sampleid\_col]][data[[timepoint\_col]] == "TP2"])

unique\_samples\_tp3 <- unique(data[[sampleid\_col]][data[[timepoint\_col]] == "TP3"])

unique\_samples\_tp4 <- unique(data[[sampleid\_col]][data[[timepoint\_col]] == "TP4"])

# Store lengths of unique samples for each timepoint

lengths <- c(

TP0 = length(unique\_samples\_tp0),

TP1 = length(unique\_samples\_tp1),

TP2 = length(unique\_samples\_tp2),

TP3 = length(unique\_samples\_tp3),

TP4 = length(unique\_samples\_tp4)

)

# Check if all timepoints have the same number of unique SampleIDs

all\_equal\_lengths <- all(lengths == lengths[1])

# Print result of the comparison

if (all\_equal\_lengths) {

cat("All timepoints have the same number of unique SampleIDs.\n")

} else {

cat("Not all timepoints have the same number of unique SampleIDs.\n")

}

# Optionally return the lengths for further inspection

return(lengths)

}

# get significance stars

get\_significance <- function(p\_value) {

ifelse(is.na(p\_value), "NA", # Handle NA values

ifelse(p\_value < 0.001, "\*\*\*",

ifelse(p\_value < 0.01, "\*\*",

ifelse(p\_value < 0.05, "\*", "ns"))))

}

# extrTP p-values

get\_p\_value <- function(tp1, tp2, tukey\_comparisons, tukey\_result) {

comparison <- paste(tp1, tp2, sep = "-")

reverse\_comparison <- paste(tp2, tp1, sep = "-")

if (comparison %in% tukey\_comparisons) {

return(tukey\_result$Timepoint[comparison, "p adj"])

} else if (reverse\_comparison %in% tukey\_comparisons) {

return(tukey\_result$Timepoint[reverse\_comparison, "p adj"])

} else {

return(NA)

}

}

my\_colors <- c("darkgreen", "orange", "red", "magenta", "purple")

# FACS ANOVA

automate\_anova\_extraction <- function(results\_folder, plots\_folder, results\_name, plot\_save, dataset, loop\_vars, timepoint\_col = "Timepoint") {

# Create directories if they don't exist

if (!dir.exists(results\_folder)) {

dir.create(results\_folder)

}

if (!dir.exists(plots\_folder)) {

dir.create(plots\_folder)

}

# Initialize an empty dataframe to store results

results <- data.frame(Variable = character(), P\_Value = numeric(),

TP1\_P\_Value = numeric(), TP2\_P\_Value = numeric(),

TP3\_P\_Value = numeric(), TP4\_P\_Value = numeric())

# Loop through the specified variables (columns) in the dataset

for (var\_name in loop\_vars) {

# Subset the dataset for the variable and timepoints

df\_test <- dataset[, c(var\_name, timepoint\_col)]

# Filter out rows with missing values

df\_test <- df\_test %>% filter(!is.na(get(var\_name)), !is.na(get(timepoint\_col)))

# Ensure Timepoint is treated as a factor

df\_test[[timepoint\_col]] <- factor(df\_test[[timepoint\_col]], levels = c("TP0", "TP1", "TP2", "TP3", "TP4"))

# Check if we have sufficient data for all timepoints

if (length(unique(df\_test[[timepoint\_col]])) < 2) {

cat("Not enough timepoints with data for:", var\_name, "\n")

next

}

# Skip if not enough data

if (nrow(df\_test) < 3) {

cat("Not enough data to run Wilcoxon test for:", var\_name, "\n")

next

}

# Initialize timepoint p-values as NA

tp1\_p\_value <- tp2\_p\_value <- tp3\_p\_value <- tp4\_p\_value <- NA

# Perform Wilcoxon test comparing each timepoint to TP0

try({

if ("TP1" %in% df\_test[[timepoint\_col]]) {

tp1\_p\_value <- wilcox.test(df\_test[[var\_name]][df\_test[[timepoint\_col]] == "TP1"],

df\_test[[var\_name]][df\_test[[timepoint\_col]] == "TP0"], na.rm = TRUE)$p.value

}

if ("TP2" %in% df\_test[[timepoint\_col]]) {

tp2\_p\_value <- wilcox.test(df\_test[[var\_name]][df\_test[[timepoint\_col]] == "TP2"],

df\_test[[var\_name]][df\_test[[timepoint\_col]] == "TP0"], na.rm = TRUE)$p.value

}

if ("TP3" %in% df\_test[[timepoint\_col]]) {

tp3\_p\_value <- wilcox.test(df\_test[[var\_name]][df\_test[[timepoint\_col]] == "TP3"],

df\_test[[var\_name]][df\_test[[timepoint\_col]] == "TP0"], na.rm = TRUE)$p.value

}

if ("TP4" %in% df\_test[[timepoint\_col]]) {

tp4\_p\_value <- wilcox.test(df\_test[[var\_name]][df\_test[[timepoint\_col]] == "TP4"],

df\_test[[var\_name]][df\_test[[timepoint\_col]] == "TP0"], na.rm = TRUE)$p.value

}

}, silent = TRUE)

# Save the Wilcoxon p-values to the results dataframe

results <- rbind(results, data.frame(Variable = var\_name,

P\_Value = min(tp1\_p\_value, tp2\_p\_value, tp3\_p\_value, tp4\_p\_value, na.rm = TRUE),

TP1\_P\_Value = tp1\_p\_value, TP2\_P\_Value = tp2\_p\_value,

TP3\_P\_Value = tp3\_p\_value, TP4\_P\_Value = tp4\_p\_value))

# Generate and save the plots if required

if (plot\_save == "y") {

median\_TP0 <- median(df\_test[df\_test[[timepoint\_col]] == "TP0", var\_name], na.rm = TRUE)

my\_colors <- c("darkgrey", "black", "black", "black", "black")

# Boxplot generation

box\_plot <- ggboxplot(df\_test, x = timepoint\_col, y = var\_name, color = timepoint\_col,

add = "jitter", legend = "none",

ylab = paste(var\_name, "percentage"), width = 0.8,

add.params = list(size = 1, alpha = 0.5)) +

geom\_hline(yintercept = median\_TP0, linetype = 2) +

stat\_compare\_means(label = "p.signif",

method = "wilcox",

ref.group = "TP0",

hide.ns = TRUE,

label.y = max(df\_test[[var\_name]]),

size = 8) +

scale\_x\_discrete(labels = c(

"TP0" = "Control",

"TP1" = "24 hours",

"TP2" = "3-5 days",

"TP3" = "1 month",

"TP4" = "3 months"

)) +

scale\_color\_manual(values = my\_colors)

# Save boxplot

boxplot\_file\_name <- file.path(plots\_folder, paste(var\_name, "\_Boxplot\_Wilcox.png", sep = ""))

ggsave(filename = boxplot\_file\_name, plot = box\_plot)

# Barplot generation with SEM

mean\_values <- df\_test %>%

group\_by(get(timepoint\_col)) %>%

summarise(mean\_value = mean(get(var\_name), na.rm = TRUE),

sd\_value = sd(get(var\_name), na.rm = TRUE),

n = n()) %>%

mutate(SEM = sd\_value / sqrt(n)) %>%

rename(Timepoint = `get(timepoint\_col)`)

#my\_grey\_scale <- c("grey", "black", "black", "black", "black")

# Barplot with SEM and significance

bar\_plot <- ggbarplot(mean\_values, x = "Timepoint", y = "mean\_value", fill = "Timepoint",

ylab = paste(var\_name, "mean (+-SEM) percentage"),

add = "mean\_se", width = 0.8) +

scale\_fill\_manual(values = my\_colors) +

scale\_x\_discrete(labels = c(

"TP0" = "Control",

"TP1" = "24 hours",

"TP2" = "3-5 days",

"TP3" = "1 month",

"TP4" = "3 months"

)) +

geom\_errorbar(aes(ymin = mean\_value - SEM, ymax = mean\_value + SEM), width = 0.2) +

stat\_compare\_means(data = df\_test, aes(x = get(timepoint\_col), y = get(var\_name)),

method = "wilcox", ref.group = "TP0", hide.ns = TRUE,

label = "p.signif", label.y = max(mean\_values$mean\_value) + 0.09 \* max(mean\_values$mean\_value),

size = 8)

# Save barplot

barplot\_file\_name <- file.path(plots\_folder, paste(var\_name, "\_Barplot\_Wilcox\_SEM.png", sep = ""))

ggsave(filename = barplot\_file\_name, plot = bar\_plot)

}

}

# Write the results to a CSV file

results\_file <- file.path(paste(results\_name, "\_Results.csv", sep = ""))

write.csv(results, file = results\_file, row.names = FALSE)

# Return the results dataframe

return(results)

}

automate\_anova\_extraction\_Category <- function(results\_folder, plots\_folder, results\_name, plot\_save, dataset, loop\_vars, timepoint\_col = "Timepoint", color\_var) {

# Create directories if they don't exist

if (!dir.exists(results\_folder)) dir.create(results\_folder)

if (!dir.exists(plots\_folder)) dir.create(plots\_folder)

# Initialize an empty dataframe to store results

results <- data.frame(Category = character(), Variable = character(), P\_Value = numeric(),

TP1\_P\_Value = numeric(), TP2\_P\_Value = numeric(),

TP3\_P\_Value = numeric(), TP4\_P\_Value = numeric())

# Loop through the specified variables (columns) in the dataset

for (var\_name in loop\_vars) {

# Subset the dataset for the variable and timepoints

df\_test <- dataset[, c(var\_name, timepoint\_col, color\_var)]

df\_test <- df\_test %>% filter(!is.na(get(var\_name)), !is.na(get(timepoint\_col)), !is.na(get(color\_var)))

# Ensure Timepoint is treated as a factor

df\_test[[timepoint\_col]] <- factor(df\_test[[timepoint\_col]], levels = c("TP0", "TP1", "TP2", "TP3", "TP4"))

# Check if we have sufficient data

if (length(unique(df\_test[[timepoint\_col]])) < 2 || nrow(df\_test) < 3) next

for (color\_vari in unique(df\_test[[color\_var]])) {

df\_color <- df\_test %>% filter(get(color\_var) == color\_vari)

tp\_p\_values <- sapply(c("TP1", "TP2", "TP3", "TP4"), function(tp) {

if (tp %in% df\_color[[timepoint\_col]]) {

tryCatch(

wilcox.test(df\_color[[var\_name]][df\_color[[timepoint\_col]] == tp],

df\_color[[var\_name]][df\_color[[timepoint\_col]] == "TP0"])$p.value,

error = function(e) NA

)

} else {

NA

}

})

results <- rbind(results, data.frame(Category = color\_vari, Variable = var\_name,

P\_Value = min(tp\_p\_values, na.rm = TRUE),

TP1\_P\_Value = tp\_p\_values["TP1"],

TP2\_P\_Value = tp\_p\_values["TP2"],

TP3\_P\_Value = tp\_p\_values["TP3"],

TP4\_P\_Value = tp\_p\_values["TP4"]))

}

# Generate plots if required

if (plot\_save == "y") {

median\_TP0 <- median(df\_test[df\_test[[timepoint\_col]] == "TP0", var\_name], na.rm = TRUE)

my\_colors <- c("Female" = "#1D04C2","Male" = "#A67C00",

"MINOR" = "#A67C00","MODERATE" = "#700606",

"Bad" = "#700606","Good" = "#A67C00")

# Filter my\_colors to match the levels of color\_var in df\_test

used\_colors <- my\_colors[unique(df\_test[[color\_var]])]

# Boxplot with Correct Legend

box\_plot <- ggboxplot(df\_test, x = timepoint\_col, y = var\_name, color = color\_var,

add = "jitter", legend = "right", ylab = paste(var\_name, "percentage")) +

# geom\_hline(yintercept = median\_TP0, linetype = 2) +

scale\_x\_discrete(labels = c("TP0" = "Control", "TP1" = "24 hours",

"TP2" = "3-5 days", "TP3" = "1 month",

"TP4" = "3 months")) +

scale\_color\_manual(values = used\_colors,

name = "Group",

labels = names(used\_colors))

# Annotate significant comparisons for each category

for (color in unique(df\_test[[color\_var]])) {

df\_color <- results[results$Category == color & results$Variable == var\_name, ]

y\_base <- max(df\_test[[var\_name]], na.rm = TRUE) + 0.1

y\_step <- 0.15

for (tp in c("TP1", "TP2", "TP3", "TP4")) {

p\_value <- df\_color[[paste(tp, "P\_Value", sep = "\_")]]

if (!is.na(p\_value) && p\_value <= 0.05) {

asterisk <- ifelse(p\_value <= 0.001, "\*\*\*",

ifelse(p\_value <= 0.01, "\*\*", "\*"))

x\_position <- which(levels(df\_test[[timepoint\_col]]) == tp)

box\_plot <- box\_plot +

annotate("text", x = x\_position, y = y\_base,

label = asterisk, size = 12, color = my\_colors[color]) #+

#annotate("segment", x = x\_position - 0.15, xend = x\_position + 0.15,

# y = y\_base - 0.02, yend = y\_base - 0.02, color = my\_colors[color])

y\_base <- y\_base + y\_step

}

}

}

ggsave(filename = file.path(plots\_folder, paste(var\_name, "\_Boxplot\_Wilcox.png", sep = "")),

plot = box\_plot, width = 7.435, height = 5)

# # Barplot with SEM

# mean\_values <- df\_test %>%

# group\_by(get(timepoint\_col), get(color\_var)) %>%

# summarise(mean\_value = mean(get(var\_name), na.rm = TRUE),

# sd\_value = sd(get(var\_name), na.rm = TRUE),

# n = n(), .groups = "drop") %>%

# mutate(SEM = sd\_value / sqrt(n)) %>%

# rename(Timepoint = `get(timepoint\_col)`, Group = `get(color\_var)`)

#

# dodge\_position <- position\_dodge(width = 0.8)

# bar\_plot <- ggbarplot(mean\_values, x = "Timepoint", y = "mean\_value", fill = "Group",

# ylab = paste(var\_name, "mean (+-SEM) percentage"),

# add = "mean\_se", width = 0.6, position = dodge\_position) +

# scale\_fill\_manual(values = my\_colors) +

# scale\_x\_discrete(labels = c("TP0" = "Control", "TP1" = "24 hours",

# "TP2" = "3-5 days", "TP3" = "1 month",

# "TP4" = "3 months")) +

# geom\_errorbar(aes(ymin = mean\_value - SEM, ymax = mean\_value + SEM),

# width = 0.2, position = dodge\_position) +

# theme\_minimal()

#

# ggsave(filename = file.path(plots\_folder, paste(var\_name, "\_Barplot\_Wilcox\_SEM.png", sep = "")), plot = bar\_plot)

}

}

# Save results to a CSV file

write.csv(results, file.path(results\_folder, paste(results\_name, "\_results.csv", sep = "")), row.names = FALSE)

# Return the results dataframe

return(results)

}

demographics\_N\_Age\_Sex <- function(df) {

# Initialize an empty dataframe to store the results

Demographics\_FACS <- data.frame(Variable = character(), Controls = character(), Patients = character(), P\_value = numeric())

# Filter for controls and patients based on SampleID

df\_Controls <- df %>% filter(SampleID %in% c(unique(df$SampleID[grep("Ctr", df$SampleID)])))

df\_Patients <- df %>% filter(SampleID %in% c(unique(df$SampleID[grep("Patient", df$SampleID)])))

# Calculate N (sample count)

N\_control <- length(unique(df\_Controls$SampleID))

N\_patients <- length(unique(df\_Patients$SampleID))

row\_N <- data.frame(Variable = "N",

Controls = N\_control,

Patients = N\_patients,

P\_value = NA)

# Calculate Age, mean (SD) for controls and patients

mean\_age\_control <- round(mean(df\_Controls$Age, na.rm = TRUE), 2)

sd\_age\_control <- round(sd(df\_Controls$Age, na.rm = TRUE), 2)

mean\_age\_patients <- round(mean(df\_Patients$Age, na.rm = TRUE), 2)

sd\_age\_patients <- round(sd(df\_Patients$Age, na.rm = TRUE), 2)

# Perform t-test for Age comparison between controls and patients

t\_test\_age <- t.test(df\_Controls$Age, df\_Patients$Age)

p\_value\_age <- round(t\_test\_age$p.value, 3)

row\_Age <- data.frame(Variable = "Age",

Controls = paste(mean\_age\_control, " ± ", sd\_age\_control, sep = ""),

Patients = paste(mean\_age\_patients, " ± ", sd\_age\_patients, sep = ""),

P\_value = p\_value\_age)

# Calculate percentage of females in controls and patients

sex\_table\_controls <- table(df\_Controls$Sex)

sex\_table\_patients <- table(df\_Patients$Sex)

percent\_female\_controls <- round((sex\_table\_controls["F"] / sum(sex\_table\_controls)) \* 100, 2)

percent\_female\_patients <- round((sex\_table\_patients["F"] / sum(sex\_table\_patients)) \* 100, 2)

# Perform chi-square test for sex distribution comparison

sex\_table\_combined <- table(df\_Controls$Sex, df\_Patients$Sex)

chi\_test\_sex <- chisq.test(sex\_table\_combined)

p\_value\_sex <- round(chi\_test\_sex$p.value, 3)

row\_Sex <- data.frame(Variable = "Female %",

Controls = percent\_female\_controls,

Patients = percent\_female\_patients,

P\_value = p\_value\_sex)

# Combine rows into the Demographics table

Demographics\_FACS <- rbind(Demographics\_FACS, row\_N, row\_Age, row\_Sex)

return(Demographics\_FACS)

}

demographics\_Compare\_N\_Age\_Sex <- function(df, split\_column) {

# Initialize an empty dataframe to store the results

Demographics\_FACS <- data.frame(Variable = character(), Controls = character(), Patients = character(), P\_value = numeric())

# Split the data based on the specified column

groups <- unique(df[[split\_column]])

if (length(groups) != 2) {

stop("The specified column must contain exactly two groups (e.g., 'Control' and 'Patient').")

}

# Extract controls and patients based on the specified column

df\_Controls <- df %>% filter(!!sym(split\_column) == groups[1])

df\_Patients <- df %>% filter(!!sym(split\_column) == groups[2])

# Calculate N (sample count)

N\_control <- length(unique(df\_Controls$SampleID))

N\_patients <- length(unique(df\_Patients$SampleID))

row\_N <- data.frame(Variable = "N",

Controls = N\_control,

Patients = N\_patients,

P\_value = NA)

# Calculate Age, mean (SD) for controls and patients

mean\_age\_control <- round(mean(df\_Controls$Age, na.rm = TRUE), 2)

sd\_age\_control <- round(sd(df\_Controls$Age, na.rm = TRUE), 2)

mean\_age\_patients <- round(mean(df\_Patients$Age, na.rm = TRUE), 2)

sd\_age\_patients <- round(sd(df\_Patients$Age, na.rm = TRUE), 2)

# Perform t-test for Age comparison between controls and patients

t\_test\_age <- t.test(df\_Controls$Age, df\_Patients$Age)

p\_value\_age <- round(t\_test\_age$p.value, 3)

row\_Age <- data.frame(Variable = "Age",

Controls = paste(mean\_age\_control, " ± ", sd\_age\_control, sep = ""),

Patients = paste(mean\_age\_patients, " ± ", sd\_age\_patients, sep = ""),

P\_value = p\_value\_age)

# Calculate percentage of females in controls and patients

sex\_table\_controls <- table(df\_Controls$Sex)

sex\_table\_patients <- table(df\_Patients$Sex)

percent\_female\_controls <- round((sex\_table\_controls["F"] / sum(sex\_table\_controls)) \* 100, 2)

percent\_female\_patients <- round((sex\_table\_patients["F"] / sum(sex\_table\_patients)) \* 100, 2)

# Perform chi-square test for sex distribution comparison

sex\_table\_combined <- table(df\_Controls$Sex, df\_Patients$Sex)

chi\_test\_sex <- chisq.test(sex\_table\_combined)

p\_value\_sex <- round(chi\_test\_sex$p.value, 3)

row\_Sex <- data.frame(Variable = "Female %",

Controls = percent\_female\_controls,

Patients = percent\_female\_patients,

P\_value = p\_value\_sex)

# Combine rows into the Demographics table

Demographics\_FACS <- rbind(Demographics\_FACS, row\_N, row\_Age, row\_Sex)

return(Demographics\_FACS)

}

plot\_my\_pca = function(pca\_result, data.meta, what, PCA\_color=NULL, legend\_pos=NULL){

# Create a scatter plot of PC1 vs PC2

if (is.null(PCA\_color) ){

PCA\_color = rainbow( length(unique(data.meta[,what])))

}

plot(pca\_result$x[, 1], pca\_result$x[, 2],

xlab = "PC1", ylab = "PC2",

main = "PCA Scatter Plot (PC1 vs PC2)",

col = c(PCA\_color)[as.numeric( factor( data.meta[,what]))], pch = 16)

# Optional: Add legend for PCA\_Sample colors

if (!is.null(legend\_pos) ){

legend(legend\_pos, legend = unique(data.meta[,what]), col = unique(PCA\_color), pch = 16, title = "PCA Sample")

} }

# Helper function to remove outliers

remove\_outliers <- function(data, variable) {

lm\_initial <- lm(mean\_capZ ~ get(variable), data = data)

residuals\_values <- residuals(lm\_initial)

outlier\_indices <- which(abs(residuals\_values) > (3 \* sd(residuals\_values)))

if (length(outlier\_indices) > 0) {

data <- data[-outlier\_indices, ]

}

return(data)

}

# Helper function to append correlation and regression results

append\_correlation\_results <- function(data, variable, gene, subpop, tp, subgroup, value, results\_df, plot\_folder) {

tryCatch({

if (nrow(data) < 2) {

cat("Not enough data points for linear regression (less than 2) for", gene, subpop, "Monocytes at", tp, "for", subgroup, value, "- skipping this subset.\n")

return(results\_df)

}

# Pearson correlation

cor\_result <- cor.test(data$mean\_capZ, data[[variable]])

# Linear regression

lm\_result <- lm(mean\_capZ ~ get(variable), data = data)

coef\_x <- summary(lm\_result)$coefficients[2, 1]

ci <- confint(lm\_result)[2, ]

IC <- summary(lm\_result)$coefficients[1, 1]

# Append to results dataframe

row <- data.frame(

Gene = gene,

GeneID = data$GeneID[1],

Gene\_Group = data$Gene\_Group[1],

Subpopulation = subpop,

Timepoint = tp,

N = nrow(data),

p.value = cor\_result$p.value,

Estimate = cor\_result$estimate,

Est\_CI\_Lower = cor\_result$conf.int[1],

Est\_CI\_Upper = cor\_result$conf.int[2],

Coefficient = coef\_x,

Coef\_CI\_Lower = ci[1],

Coef\_CI\_Upper = ci[2],

Intercept = IC,

Subgroup = subgroup,

Subgroup\_Value = value

)

results\_df <- rbind(results\_df, row)

if (row$p.value <= 0.05) {

#titel <- paste(gene, " expression in ", subpop, " Monocytes at ", tp, " ( ", value, " Samples)", sep="")

titel <- paste(value, " Samples -", gene," expr. of ", subpop, " Monocytes at ", tp, sep="")

file\_name <- paste0(plot\_folder, "/",

value, "\_Samples\_",

gene, "\_",

subpop, "Monocytes\_", tp,

".png", sep="")

# Create the plot and save it

plot <- ggplot(data = data, aes\_string(x = variable, y = "mean\_capZ")) +

geom\_smooth(method = "glm", color = "black") +

geom\_point(aes(color = Recovery), size = 2) +

ggtitle(titel) +

xlab(variable) +

ylab(paste(gene, "expression (Z-scored from CT - CT Sample Median)")) +

scale\_color\_manual(values = c("MINOR" = "#A67C00", "MODERATE" = "#700606",

"Male" = "#A67C00", "Female" = "#1D04C2",

"Good" = "darkgreen", "Bad" = "#700606")) +

theme(text = element\_text(size = 14)) +

theme(

panel.background = element\_rect(fill = "white", color = "black"),

plot.background = element\_rect(fill = "white", color = "black"),

text = element\_text(color = "black"),

panel.grid.major = element\_line(color = "gray", size = 0.2),

panel.grid.minor = element\_blank(),

panel.grid.major.x = element\_blank()

)

# Save plot with specified dimensions

ggsave(filename = file\_name, plot = plot, width = 8, height = 6, dpi = 300)

cat("Plot saved successfully to:", file\_name, "\n")

} else {}

return(results\_df)

}, error = function(e) {

cat("Error for", gene, subpop, "Monocytes at", tp, "for", subgroup, value, "- skipping this subset.\n")

return(results\_df)

})

}

perform\_linear\_regression\_correlation <- function(data, variable, folder\_prefix) {

# Create the output folders if they do not exist

plot\_folder <- paste0("LinReg\_Plots\_", folder\_prefix, "\_capZ")

result\_folder <- paste0("LinReg\_Results\_capZ")

if (!dir.exists(plot\_folder)) dir.create(plot\_folder, recursive = TRUE)

if (!dir.exists(result\_folder)) dir.create(result\_folder, recursive = TRUE)

# Initialize dataframes for correlation results

correlation\_results <- data.frame(Subpopulation = character(),

Timepoint = character(),

Gene = character(),

GeneID = numeric(),

Gene\_Group = character(),

N = numeric(),

p.value = numeric(),

Estimate = numeric(),

Est\_CI\_Lower = numeric(),

Est\_CI\_Upper = numeric(),

Coefficient = numeric(),

Coef\_CI\_Lower = numeric(),

Coef\_CI\_Upper = numeric(),

Intercept = numeric(),

Subgroup = character(),

Subgroup\_Value = character())

# Loop over genes and combinations of subpopulations, timepoints, etc.

for (gene in unique(data$Gene)) {

df <- data %>% filter(Gene == gene)

df <- df %>% filter(!is.na(Timepoint), !is.na(mean\_capZ), !is.na(get(variable))) # Clean data

for (subpop in unique(df$Subpopulation)) {

df\_subpop <- df %>% filter(Subpopulation == subpop)

for (tp in unique(df\_subpop$Timepoint)) {

df\_tp <- df\_subpop %>% filter(Timepoint == tp)

if (nrow(df\_tp) <= 0) next

# Outlier removal for the overall group

df\_tp\_filtered <- remove\_outliers(df\_tp, variable)

if (nrow(df\_tp\_filtered) == 0) next

# Perform analysis for the whole group

correlation\_results <- append\_correlation\_results(df\_tp\_filtered, variable,

gene, subpop, tp, "All", "All", correlation\_results, plot\_folder)

# # Subgroup analysis for Sex, Category, and Recovery

# for (subgroup in c("Sex", "Category", "Recovery")) {

# for (value in unique(df\_tp[[subgroup]])) {

# df\_subgroup <- df\_tp %>% filter(.data[[subgroup]] == value)

# df\_subgroup\_filtered <- remove\_outliers(df\_subgroup, variable)

# if (nrow(df\_subgroup\_filtered) == 0) next

#

# correlation\_results <- append\_correlation\_results(df\_subgroup\_filtered, variable,

# gene, subpop, tp, subgroup, value, correlation\_results, plot\_folder)

# }

#}

}

}

}

# Write final results to CSV

write.csv(correlation\_results, file = paste0(result\_folder, "/LinReg\_", variable, "\_capZ.csv"), row.names = FALSE)

return(correlation\_results)

}

## \*Data\* ---------------------------------------------------------------------

#### Loading of the Data -----------------------------------------

setwd("/Users/ju5263ta/Github/Monocytes/rawData\_Stroke")

getwd ()

# make sure that before you check the files, that not an empty column has been added to the end of the file, remove and export as .csv otherwise.

files <- system( "ls \*B2M.csv", intern=T)

# if you just have one data set:

i=1

raw\_data <- read.csv(text=readLines(files[i])[-(1:11)], header = T, sep=',', dec = '.',

row.names = 1

)

# delete Call.1 column, and resent column names

if ( length(colnames(raw\_data)) > 13) {

raw\_data <- raw\_data[, !colnames(raw\_data) %in% "Call.1"]

colnames(raw\_data) <- c("Name","Type","rConc","Name.1","Type.1","Reference","Value",

"Quality","Call","Threshold","Value.1","Quality.1","Call.1")

}

raw\_data$Dataset<-files[i]

# for more the 1 data set:

i = 2

for (i in (2:length(files))) {

y = raw\_data

x <- read.csv(text=readLines(files[i])[-(1:11)], header = T, sep=',', dec = '.',row.names = 1)

if (length(colnames(x))>13) {

x <- x[, !colnames(x) %in% "Call.1"]

colnames(x) <- c("Name","Type","rConc","Name.1","Type.1","Reference","Value",

"Quality","Call","Threshold","Value.1","Quality.1","Call.1")

}

x$Dataset<-files[i]

y <- rbind(y, x)

i = i +1

raw\_data = y

}

#### trimming of data ------------------------------------------------------

data <- cbind.data.frame(Sample=raw\_data$Name,

Gene=raw\_data$Name.1,

Value=raw\_data$Value,

dCT\_Value=raw\_data$Value.1,

Call=raw\_data$Call.1)

#### evaluate PASS FAIL------------------------------------------------

data$Value<-ifelse(data$Call=='Flag', yes=35, no=data$Value) #999

data$Value<-ifelse(data$Call=='mFlag', yes=NA, no=data$Value) #999

data$dCT\_Value<-ifelse(data$Call=='Flag', yes=35, no=data$dCT\_Value) #999

data$dCT\_Value<-ifelse(data$Call=='mFlag', yes=NA, no=data$dCT\_Value) #999 instead of NA

# sort through quality and give NA to everything that Failed (=999) or is above CT 35

data$Value<-ifelse(data$Value>=35, yes=35, no=data$Value)

data$dCT\_Value<-ifelse(data$dCT\_Value>=35, yes=35, no=data$dCT\_Value)

#data$dCT\_Value<-ifelse(data$dCT\_Value>=35, yes=35, no=data$dCT\_Value)

# Trim your data

dataTRIM <-cbind.data.frame(Sample=data$Sample,

Gene1=data$Gene,

CT\_Value=data$Value,

dCT\_Value=data$dCT\_Value,

Comment = NA)

# Information on the data ---------------------------------------------------------------------

# adjust the indicators for the genes list here (for the heatmap) - line 138:

dataTRIM$TechnicalReplicate <- raw\_data$Name.1

info\_genes = t(data.frame(lapply( dataTRIM$TechnicalReplicate, function(x){

ret = unlist( stringr::str\_split( x, "[\_\\s\\.]"));

if ( length(ret) < 2){

ret = c(ret, rep(0, 2-length(ret) ))

}

ret } ) ))

colnames(info\_genes) = c("Gene", "TechnicalReplicate")

info\_genes <- as.data.frame(info\_genes)

dataTRIM$TechnicalReplicate <- NULL

dataTRIM = cbind(dataTRIM, info\_genes)

# List of Genes

gene\_names <- unique(dataTRIM$Gene)

# List of Genes

genes <- unique(dataTRIM$Gene)

genes

# Number of Genes

no\_genes <- length(genes)

no\_genes

# List of Samples

samples <- unique(dataTRIM$Sample)

# Split technical and (pseudo-)biological replicates....:

#without the B1/B2 segregation, we would have each read of the gene normalized against the mean of the two biological repeats of B2M. This is in theory correct but we lose the information on the mean/SD while doing so through the machine.

#without the T1/T2 separation, we have the plate as a whole normalized against all 4 reads... with it we can split the whole plate into two halves, each standing for its own technical read.

info\_rows <- t(data.frame(

sapply(dataTRIM$Sample, function(x) {

ret <- unlist(stringr::str\_split(x, "[\_\\s\\.]"))

if (length(ret) < 5) {

ret <- c(ret, rep(NA, 5 - length(ret)))

}

ret[1:5] # Select only the first five elements

})

))

colnames(info\_rows) = c("Subpopulation", "Cell\_Count", "SampleID","Timepoint","BiologicalReplicate")

# NRT: no Polymerase, technical control

# NTC: no Template, neg-Sample control

# 10xLC:Linarity control

info\_rows <- as.data.frame(info\_rows)

# Label Subpopulations:

# remove the C in NC to avoid problems:

info\_rows$Subpopulation <- gsub("NC","N",as.character(info\_rows$Subpopulation))

info\_rows$Subpopulation[grep("N",info\_rows$Subpopulation)] <- "nonclassical"

info\_rows$Subpopulation[grep("M",info\_rows$Subpopulation)] <- "all"

info\_rows$Subpopulation[grep("C",info\_rows$Subpopulation)] <- "classical"

info\_rows$Subpopulation[grep("I",info\_rows$Subpopulation)] <- "intermediate"

dataTRIM = cbind(dataTRIM, info\_rows)

## remove unnessessary info (for now)

#It's easier to make the imputation on less columnes/variables

# Remove Extra sorts

dataTRIM <- dataTRIM %>% filter(BiologicalReplicate != "Extra")

dataTRIM <- dataTRIM %>% filter(Subpopulation != "noRT")

dataTRIM <- dataTRIM %>% filter(Cell\_Count != "200")

# remove all Timepoint 5

dataTRIM <- dataTRIM %>% filter(Timepoint != "TP5")

dataTRIM <- dataTRIM %>% filter(SampleID != "Test01")

dataTRIM <- dataTRIM %>% filter(SampleID != "Test02")

dataTRIM <- dataTRIM %>% filter(SampleID != "Test03")

dataTRIM <- dataTRIM %>% filter(SampleID != "Test04")

dataTRIM$Cell\_Count <- NULL

#### Expand data --------------------------------------------------------

unique\_SampleID <- unique(dataTRIM$SampleID[!grepl("^Ctr", dataTRIM$SampleID)])

unique\_Gene <- unique(dataTRIM$Gene)

unique\_Subpopulation <- unique(dataTRIM$Subpopulation)

unique\_Timepoint <- unique(dataTRIM$Timepoint[!grepl("TP0", dataTRIM$Timepoint)])

unique\_TechnicalReplicate <- unique(dataTRIM$TechnicalReplicate)

unique\_BiologicalReplicate <- unique(dataTRIM$BiologicalReplicate)

expanded\_df <- expand.grid(SampleID = unique\_SampleID,

Gene = unique\_Gene,

Subpopulation = unique\_Subpopulation,

Timepoint = unique\_Timepoint,

TechnicalReplicate = unique\_TechnicalReplicate,

BiologicalReplicate = unique\_BiologicalReplicate)

dataEXPAND <- merge(expanded\_df, dataTRIM, by = c("SampleID", "Gene", "Subpopulation", "Timepoint", "TechnicalReplicate", "BiologicalReplicate"), all.x = TRUE)

dataEXPAND$SampleID <- as.character(dataEXPAND$SampleID)

dataEXPAND$Gene <- as.character(dataEXPAND$Gene)

dataEXPAND$Subpopulation <- as.character(dataEXPAND$Subpopulation)

dataEXPAND$Timepoint <- as.character(dataEXPAND$Timepoint)

dataEXPAND$TechnicalReplicate <- as.character(dataEXPAND$TechnicalReplicate)

dataEXPAND$BiologicalReplicate <- as.character(dataEXPAND$BiologicalReplicate)

dataCTR <- dataTRIM[grep("^Ctr", dataTRIM$SampleID), ]

dataEXPAND <- rbind(dataEXPAND, dataCTR)

dataEXPAND$Comment <- ifelse(is.na(dataEXPAND$Sample), yes = "No Fludigm run, imputed Values", no = dataEXPAND$Comment)

#### Metadata & FACSdata---------------------------------------------------------------

setwd("/Users/ju5263ta/Github/Monocytes/rawData\_Stroke")

metadataP <- read\_excel("Metadata\_PatientID.xlsx")

Metadata\_GeneID <- read\_excel("Metadata\_GeneID.xlsx")

Plate\_info <- read\_excel("241014\_PlateID.xlsx")

Metadata\_NeuroTest <- as.data.frame(read\_excel("Metadata\_NeuroTest.xlsx",

col\_types = c("text", "text", "numeric",

"numeric", "numeric", "numeric",

"numeric", "numeric")))

dataAll <- merge(dataEXPAND, metadataP, by = "SampleID", all.x = TRUE)

#### match the right controls

#Matched\_TP0\_Gene <- c("Ctr01", "Ctr02", "Ctr08", "Ctr09", "Ctr11", "Ctr14", "Ctr16", "Ctr17", "Ctr18", "Ctr25", "Ctr27", "Ctr29", "Ctr31","Ctr41", "Ctr43")

Matched\_TP0\_Gene <- c("Ctr20", "Ctr07", "Ctr10", "Ctr14", "Ctr25", "Ctr24", "Ctr18", "Ctr19",

"Ctr08", "Ctr26", "Ctr17", "Ctr09", "Ctr27","Ctr16", "Ctr11")

Unmatched\_TP0\_FACS <- c("Ctr16", "Ctr21", "Ctr23", "Ctr30", "Ctr39", "Ctr44", "Ctr33")

# FACS

FACSdata <- as.data.frame(read\_excel("241008\_StrokeControlMFI\_combined.xls", # 241008\_StrokeControlMFI\_combined.xls for updated one

col\_types = c("text", "numeric", "numeric",

"numeric", "numeric", "numeric",

"numeric", "numeric", "numeric",

"numeric", "numeric", "numeric",

"numeric", "numeric")))

# split Sample ID into all info

FACS\_info\_rows <- t(data.frame(sapply(FACSdata$...1, function(x) {

ret <- unlist(stringr::str\_split(x, "[\_\\s\\.]"))

if (length(ret) < 4) {

ret <- c(ret, rep(NA, 4 - length(ret)))

}

ret[1:4] # Select only the first five elements

})

))

colnames(FACS\_info\_rows) = c("Type", "Plate", "Sorting Map","Sorted Sample")

FACS\_info\_rows <- as.data.frame(FACS\_info\_rows)

FACS\_info\_rows$PlateID <- paste(FACS\_info\_rows$Plate,FACS\_info\_rows$`Sorted Sample` )

FACSdata <- cbind(FACSdata, FACS\_info\_rows)

# Match PlateID with the right sample with the right data

FACSdata <- merge(FACSdata, Plate\_info, by = "PlateID")

# remove the 5th timepoint

FACSdata <- FACSdata %>% filter(Timepoint != "TP5")

# add the metadata of Patients

FACSdata <- merge(FACSdata, metadataP, by = "SampleID", all.x = TRUE)

# add the metadata of NHISS score of Patients

FACSdata <- merge(FACSdata, Metadata\_NeuroTest, by = c("SampleID", "Timepoint"), all.x = TRUE)

# Combine the data frames based on common values in the "id" column

# --> here i loose all the test because they are not in the metadata

dataAll <- merge(dataEXPAND, metadataP, by = "SampleID", all.x = TRUE)

#relabel Male & Female

dataAll$Sex[grep("M",dataAll$Sex)] <- "Male"

dataAll$Sex[grep("F",dataAll$Sex)] <- "Female"

FACSdata$Sex[grep("M",FACSdata$Sex)] <- "Male"

FACSdata$Sex[grep("F",FACSdata$Sex)] <- "Female"

dataAll <- dataAll %>% filter(!is.na(Age), !is.na(SampleID), !is.na(Category), !is.na(Sex))

#### failed Genes --------------------------------------------------------

fails <- dataAll %>%

group\_by(Subpopulation, Gene, Timepoint) %>%

summarize(

N = n(),

Count\_Above\_35\_CT = sum(CT\_Value >= 35, na.rm = TRUE) + sum(is.na(CT\_Value)), # Include NAs in the count

)

fails$Per\_Above\_35\_CT = fails$Count\_Above\_35\_CT/fails$N \* 100

failed\_genes <- fails %>% filter(Per\_Above\_35\_CT >= 90) %>% select(Gene)

wrong\_threshold <- c("CLEC7A", "S100A8", "S100A9")

exclude\_genes <- unique(c("Xeno",failed\_genes$Gene, wrong\_threshold))

paste("The gene",unique(failed\_genes$Gene), "was excluded, as more than 90% of the reads failed.")

paste("The gene",wrong\_threshold, "was excluded, as the global threshold setting did not allow proper evaluation.")

# remove the failed genes:

dataWORKING <- dataAll %>% filter(!Gene %in% exclude\_genes)

#### failed Samples --------------------------------------------------------

fails\_Sample <- dataWORKING %>%

group\_by(Sample) %>%

summarize(

N = n(),

Count\_Above\_35\_CT <- sum(CT\_Value >= 35, na.rm = TRUE),

Per\_Above\_35\_CT = mean(CT\_Value >= 35, na.rm = TRUE) \* 100,

Count\_Above\_35\_dCT = sum(dCT\_Value >= 35, na.rm = TRUE),

Per\_Above\_35\_dCT = mean(dCT\_Value >= 35, na.rm = TRUE) \* 100

)

failed\_samples <- fails\_Sample %>% filter(Per\_Above\_35\_CT >= 30) %>% select(Sample)

failed\_samples <- failed\_samples$Sample

paste("The Sample ",failed\_samples, "was excluded, as more than 30% of the reads failed.")

# Set all Values from failed Samples (> 30%) NA:

dataWORKING$CT\_Value <- ifelse(dataWORKING$Sample %in% failed\_samples, yes = NA, no = dataWORKING$CT\_Value)

dataWORKING$dCT\_Value <- ifelse(dataWORKING$Sample %in% failed\_samples, yes = NA, no = dataWORKING$dCT\_Value)

dataWORKING$Comment <- ifelse(dataWORKING$Sample %in% failed\_samples, yes = "More 30% of CT values were 35 and above, imputed Values", no = dataWORKING$Comment)

unique(dataWORKING$Comment)

#### Evaluate Housekeeping genes -----------

FailedHK <-table(c(

getMeTheFailedSamples(dataWORKING, "ACTB", 30),

getMeTheFailedSamples(dataWORKING, "B2M", 30),

getMeTheFailedSamples(dataWORKING, "GAPDH", 30)))

hist(as.numeric(FailedHK),

main = "Histogram of Failed Samples",

xlab = "Failed Samples Count",

breaks = 10)

failedSAMPLES <- names(FailedHK) [ which(FailedHK > 2) ]

# Set CT & dCT NA for the Sampels with failed HK gene

# Set data$Value to NA for the samples that are in the failedSAMPLES vector

dataWORKING$CT\_Value <- ifelse(dataWORKING$Sample %in% failedSAMPLES, yes = NA, no = dataWORKING$CT\_Value)

dataWORKING$dCT\_Value <- ifelse(dataWORKING$Sample %in% failedSAMPLES, yes = NA, no = dataWORKING$dCT\_Value)

dataWORKING$Comment <- ifelse(dataWORKING$Sample %in% failedSAMPLES, yes = "More than 2 houskeeing gene reads failed", no = dataWORKING$Comment)

# Filter the data for the desired genes

filtered\_data <- dataWORKING %>%

filter(Gene %in% c("B2M", "ACTB", "GAPDH"))

# Calculate the CV for each group

variance\_table <- filtered\_data %>%

group\_by(Gene, Timepoint, Subpopulation) %>%

summarize(CV = sd(CT\_Value, na.rm = TRUE) / mean(CT\_Value, na.rm = TRUE)) %>%

ungroup()

# Display the variance table

print(variance\_table)

# Plotting the variances

ggplot(variance\_table, aes(x = Timepoint, y = CV, fill = Subpopulation)) +

geom\_bar(stat = "identity", position = "dodge") +

facet\_wrap(~ Gene, scales = "fixed") +

labs(title = "Coefficient of Variation Across Timepoints and Subpopulations",

x = "Timepoint", y = "Coefficient of Variation (CV)") +

theme\_minimal() +

scale\_fill\_manual(values = c("darkgreen", "orange", "red", "magenta"))

# \*Test: Pool TP1 & TP2 and TP3 & TP4\* --------------------------------------------------------------------------------------------

# add TP2 to TP1

# dataWORKING$Timepoint[grep("TP2",dataWORKING$Timepoint)] <- "TP1"

# add TP2 to TP1

# dataWORKING$Timepoint[grep("TP4",dataWORKING$Timepoint)] <- "TP3"

# \*Imputation\* --------------------------------------------------------------------------------------------

#### Prepare Matrix for imputation ---------------------------------------------------------------------

# visualize the missing values:

vis\_miss(dataWORKING)

#hist(dataWORKING$dCT\_Value, main = "Histogram of Original Data")

initial2<-mice(dataWORKING, maxit=0, print=F)

# Check

initial2$method

# exclude form the imputation

initial2$method["Sample"]<-"none"

initial2$method["Gene1"]<-"none"

initial2$method["TechnicalReplicate"]<-"none"

initial2$method["BiologicalReplicate"]<-"none"

initial2$method["Comment"]<-"none"

#Check again

initial2$method

# convert to factors

dataWORKING$Sex <- as.factor(dataWORKING$Sex)

dataWORKING$Subpopulation <- as.factor(dataWORKING$Subpopulation)

dataWORKING$Gene <- as.factor(dataWORKING$Gene)

dataWORKING$Category <- as.factor(dataWORKING$Category) # check the effect of ordered

dataWORKING$Recovery <- as.factor(dataWORKING$Recovery) # check the effect of ordered

dataWORKING$Timepoint <- factor(dataWORKING$Timepoint, ordered = TRUE)

# Matrix of predictors:

initial2$predictorMatrix

#### Impute Gene expression --------------------------------------------------------------------------------------------

data\_imputed<-mice(dataWORKING, m=20, maxit=10, seed=1234, meth=initial2$method, pred=initial2$predictorMatrix)

# extract the imputed data into a “normal” data frame, run the analyses on each imputation separately and do model diagnostics.

finaldata <- complete(data\_imputed, "long")

#View(finaldata)

table(finaldata$.imp)

data\_imputed\_mean <- finaldata %>%

group\_by(SampleID, Gene, Subpopulation,Timepoint, TechnicalReplicate, BiologicalReplicate, Sample, Gene1, Age, Sex, Category, Recovery) %>%

summarize(CT = mean(CT\_Value, na.rm = TRUE),

# CT\_sd = sd(CT\_Value, na.rm = TRUE),

# CT\_N = sum(!is.na(CT\_Value)),

# dCT\_B2M = mean(dCT\_Value, na.rm = TRUE),

# dCT\_B2M\_sd = sd(dCT\_Value, na.rm = TRUE),

.groups = "drop")

dataFINAL <- as.data.frame(data\_imputed\_mean)

# \*Normalize the data vs the Geometric mean of each Sample\* ----------------

# Calculate the CT median or mean

dataFINAL$CTmedian <- NA

#dataFINAL$CTmean <- NA

#dataFINAL$CTgmean <- NA # For geometric mean

dataFINAL <- dataFINAL %>%

group\_by(Sample) %>%

mutate(

CTmedian = median(CT[CT < 35], na.rm = TRUE),

#CTmean = mean(CT[CT < 35], na.rm = TRUE),

#CTgmean = exp(mean(log(CT[CT < 35]), na.rm = TRUE)) # Geometric mean

)

# normalize all CT value to the Media CT value of each Sample

dataFINAL$dCT\_median <- dataFINAL$CT - dataFINAL$CTmedian

#dataFINAL$dCT\_mean <- dataFINAL$CT - dataFINAL$CTmean

#dataFINAL$dCT\_gmean <- dataFINAL$CT - dataFINAL$CTgmean

# \*Zscore: Normalize the data vs Gene expression ----------------

# as such: apply( data\_summary , 1 , function(x) { (x["mean\_Z"]- x["MeanForGene"])/ x["sd\_for\_gene"] } )

dataFINAL <- dataFINAL %>%

group\_by(Gene) %>%

mutate(

mean\_dCT\_Gene = mean(dCT\_median, na.rm = TRUE), # Mean of mean\_capZ by Gene

sd\_dCT\_Gene = sd(dCT\_median, na.rm = TRUE), # Standard deviation of mean\_capZ by Gene

z\_score\_dCT = (dCT\_median - mean\_dCT\_Gene) / sd\_dCT\_Gene, # Z-score calculation

inverted\_z\_score = -z\_score\_dCT,

capped\_z\_score = pmin(pmax(inverted\_z\_score, -3), 3)

) %>%

ungroup()

#### Gene & NHISS Metadata ----------------

## Gene Metadata & other edits:

#Gene\_Info <- data.frame(Gene = Metadata\_GeneID$Gene, GeneID = Metadata\_GeneID$GeneID, Gene\_Group = Metadata\_GeneID$Gene\_Group)

#dataFINAL <- merge(dataFINAL, Gene\_Info, by = "Gene")

dataFINAL <- merge(dataFINAL, Metadata\_GeneID, by = "Gene")

# add the metadata of NHISS score of Patients

dataFINAL <- merge(dataFINAL, Metadata\_NeuroTest, by = c("SampleID", "Timepoint"), all.x = TRUE)

# List of Patients (2x 15))

patients <- unique(dataFINAL$SampleID)

no\_patients <- length(patients)

#### Days Post-Stroke data ----------------

# DaysPS <- read\_excel("Metadata\_DaysPS.xls")

dataFINAL$DaysPS <- as.numeric(NA)

dataFINAL$DaysPS[grep("TP0",dataFINAL$Timepoint)] <- 0

dataFINAL$DaysPS[grep("TP1",dataFINAL$Timepoint)] <- 1

dataFINAL$DaysPS[grep("TP2",dataFINAL$Timepoint)] <- 4

dataFINAL$DaysPS[grep("TP3",dataFINAL$Timepoint)] <- 30

dataFINAL$DaysPS[grep("TP4",dataFINAL$Timepoint)] <- 90

#dataFINAL$DaysPS[grep("TP5",dataFINAL$Timepoint)] <- 360

#### MEAN values for the ANOVA & so ----------------

dataFINALmean <- dataFINAL %>%

group\_by(SampleID, Sex, Age, Category, Timepoint, DaysPS, Subpopulation,

Gene, GeneID, Gene\_Group, Recovery, NHISS, mRS, Barthel, MoCA,

`HADS\_Anxiety`, `HADS\_Depression`) %>%

summarise(mean\_Z = mean(inverted\_z\_score), sd\_Z = sd(inverted\_z\_score),

mean\_capZ = mean(capped\_z\_score), sd\_capZ = sd(capped\_z\_score)

)%>%

ungroup()

#### File Location ----------------

today <- Sys.Date()

output\_location <- paste(today,"\_Stroke\_Results", sep="")

setwd(paste("/Users/ju5263ta/Github/Monocytes/Data/",output\_location, "/",sep=""))

getwd()

write\_xlsx(fails, "Fail\_rates.xlsx")

write\_xlsx(dataFINAL, "dataFINAL.xlsx")

write\_xlsx(dataFINALmean, "dataFINALmean.xlsx")

write\_xlsx(FACSdata, "FACSdata.xlsx")

# \*Dotplot of expressed Genes\* -----------------------------------------------------------------------------

createFolder("Dotplots\_Gene\_summary")

# Create a new column for Timepoint and Subpopulation combination

dataFINALmean <- dataFINALmean %>%

mutate(Sample\_Combo = paste(Timepoint, Subpopulation, sep = "\_"))

dataFINALmean <- dataFINALmean %>%

mutate(Sample\_Combo2 = paste(Subpopulation, Timepoint, sep = "\_"))

# Create a combined column for Gene and GeneID for better sorting

dataFINALmean <- dataFINALmean %>%

mutate(Gene\_Combined = paste(GeneID, Gene, sep = "\_")) # Combine GeneID and Gene

# Using the function for data\_summary and data\_summary2

data\_summary <- calculate\_summary(dataFINALmean, "Sample\_Combo", "Gene\_Combined")

data\_summary2 <- calculate\_summary(dataFINALmean, "Sample\_Combo2", "Gene\_Combined")

data\_summary <- data\_summary %>%

mutate(Gene\_Combined = factor(Gene\_Combined, levels = rev(unique(Gene\_Combined))))

data\_summary2 <- data\_summary2 %>%

mutate(Gene\_Combined = factor(Gene\_Combined, levels = rev(unique(Gene\_Combined))))

# Create a linear gradient function for the original plots

linear\_gradient <- function() {

# Define the colors for the gradient

colors <- c("blue", "white", "red")

# Set the breakpoints for the gradient (-1.2 to 1.2)

scale\_color\_gradientn(colors = colors,

limits = c(-1.2, 1.2), # Set the limits from -1.2 to 1.2

guide = "colorbar",

na.value = "grey50") # Color for NA values

}

ggplot(data\_summary, aes(x = Sample\_Combo, y = Gene\_Combined)) +

geom\_point(aes(size = Dot\_Size, color = mean\_capZ)) + # Use Dot\_Size for scaling

scale\_size(range = c(1, 5)) + # Size range reflects 1-5 scaling

linear\_gradient() + # Use the custom linear gradient

theme\_minimal() +

labs(

title = "Dot Plot of qPCR Data (high Expr. in red; low Expr. in blue)",

color = "Gene Expr. \n(Zscored)",

size = "high SD (small)\nlow SD (big)" # Use "\n" for line break

) +

theme(

axis.text.x = element\_text(angle = 90, hjust = 1),

plot.background = element\_rect(fill = "white"), # Set the plot background to white

panel.background = element\_rect(fill = "white"), # Set the panel background to white

panel.grid.major = element\_line(color = "grey", size = 0.5), # Optional: adjust grid lines

panel.grid.minor = element\_line(color = "lightgrey", size = 0.25) # Optional: adjust minor grid lines

)

ggsave(filename = "Dotplots\_Gene\_summary/Exp\_by\_TP.png")

ggplot(data\_summary2, aes(x = Sample\_Combo2, y = Gene\_Combined)) +

geom\_point(aes(size = Dot\_Size, color = mean\_capZ)) + # Use Dot\_Size for scaling

scale\_size(range = c(1, 5)) + # Size range reflects 1-5 scaling

linear\_gradient() + # Use the custom linear gradient

theme\_minimal() +

labs(

title = "Dot Plot of qPCR Data (high Expr. in red; low Expr. in blue)",

color = "Gene Expr. \n(Zscored)",

size = "high SD (small)\nlow SD (big)" # Use "\n" for line break

) +

theme(

axis.text.x = element\_text(angle = 90, hjust = 1),

plot.background = element\_rect(fill = "white"), # Set the plot background to white

panel.background = element\_rect(fill = "white"), # Set the panel background to white

panel.grid.major = element\_line(color = "grey", size = 0.5), # Optional: adjust grid lines

panel.grid.minor = element\_line(color = "lightgrey", size = 0.25) # Optional: adjust minor grid lines

)

ggsave(filename = "Dotplots\_Gene\_summary/Exp\_by\_Suptype.png")

# Create a linear gradient function for the original plots

linear\_gradient <- function() {

# Define the colors for the gradient

colors <- c("white", "grey", "black")

# Set the breakpoints for the gradient (-1.2 to 1.2)

scale\_color\_gradientn(colors = colors,

limits = c(-1.2, 1.2), # Set the limits from -1.2 to 1.2

guide = "colorbar",

na.value = "red") # Color for NA values

}

ggplot(data\_summary, aes(x = Sample\_Combo, y = Gene\_Combined)) +

geom\_point(aes(size = Dot\_Size, color = mean\_capZ)) + # Use Dot\_Size for scaling

scale\_size(range = c(1, 5)) + # Size range reflects 1-5 scaling

linear\_gradient() + # Use the custom linear gradient

theme\_minimal() +

labs(

title = "Dot Plot of qPCR Data (high Expr. in red; low Expr. in blue)",

color = "Gene Expr. \n(Zscored)",

size = "high SD (small)\nlow SD (big)" # Use "\n" for line break

) +

theme(

axis.text.x = element\_text(angle = 90, hjust = 1),

plot.background = element\_rect(fill = "white"), # Set the plot background to white

panel.background = element\_rect(fill = "white"), # Set the panel background to white

panel.grid.major = element\_line(color = "grey", size = 0.5), # Optional: adjust grid lines

panel.grid.minor = element\_line(color = "lightgrey", size = 0.25) # Optional: adjust minor grid lines

)

ggsave(filename = "Dotplots\_Gene\_summary/Exp\_by\_TP\_grey.png")

ggplot(data\_summary2, aes(x = Sample\_Combo2, y = Gene\_Combined)) +

geom\_point(aes(size = Dot\_Size, color = mean\_capZ)) + # Use Dot\_Size for scaling

scale\_size(range = c(1, 5)) + # Size range reflects 1-5 scaling

linear\_gradient() + # Use the custom linear gradient

theme\_minimal() +

labs(

title = "Dot Plot of qPCR Data (high Expr. in red; low Expr. in blue)",

color = "Gene Expr. \n(Zscored)",

size = "high SD (small)\nlow SD (big)" # Use "\n" for line break

) +

theme(

axis.text.x = element\_text(angle = 90, hjust = 1),

plot.background = element\_rect(fill = "white"), # Set the plot background to white

panel.background = element\_rect(fill = "white"), # Set the panel background to white

panel.grid.major = element\_line(color = "grey", size = 0.5), # Optional: adjust grid lines

panel.grid.minor = element\_line(color = "lightgrey", size = 0.25) # Optional: adjust minor grid lines

)

ggsave(filename = "Dotplots\_Gene\_summary/Exp\_by\_Suptype\_grey.png")

#### PCA -------

dataFINALmean <- dataFINALmean %>%

mutate(PCA\_Sample = paste(Timepoint, Subpopulation, SampleID, sep = "\_"))

# Reverse melt to create matrix for PCA

data\_wide <- dcast(dataFINALmean, PCA\_Sample ~ Gene, value.var = "mean\_capZ")

rownames(data\_wide) <- data\_wide$PCA\_Sample

data\_wide <- data\_wide[, -1] # Remove the Sample column to leave only the matrix

data.meta <- t( data.frame(stringr::str\_split( rownames( data\_wide), "\_")))

colnames(data.meta) <- c("Timepoint", "Subpopulation", "SampleID")

data.meta <- merge(data.meta , metadataP, by = "SampleID", all.x = TRUE)

#data.meta <- cbind( data.meta, dataFINALmean[rownames(data.meta), "Age"])

pca\_result <- prcomp(data\_wide, scale. = TRUE)

plot\_my\_pca(pca\_result, data.meta, "Timepoint", my\_colors, legend\_pos = "topleft")

subtype\_col <- c("grey", "red", "orange", "green")

plot\_my\_pca(pca\_result, data.meta, "Subpopulation", subtype\_col, "topleft")

Category\_col <- c("red", "orange", "green")

plot\_my\_pca(pca\_result, data.meta, "Category", Category\_col, "topleft")

plot\_my\_pca(pca\_result, data.meta, "Recovery", c("red", "darkgreen","grey"), "topleft")

plot\_my\_pca(pca\_result, data.meta, "Sex", c("#A67C00", "#1D04C2"), "topleft")

data.meta$Age <- factor(data.meta$Age, levels = sort(unique(data.meta$Age)), ordered = TRUE)

# Define Gradient\_colour based on the levels of Age

Gradient\_colour <- setNames(

colorRampPalette(c("blue", "red"))(length(levels(data.meta$Age))),

levels(data.meta$Age)

)

# Run the function to plot PCA, mapping Age levels to colors

plot\_my\_pca(pca\_result, data.meta, "Age", Gradient\_colour[data.meta$Age])

# \*FACS Statistics\* ----------------

FACSdata\_copy <- FACSdata

#### 1WAY ANOVA\* ----------------------------------------------------------------------------------------------------

ANOVA\_FACSdata <- FACSdata %>% filter(!(Timepoint == "TP0" & SampleID %in% Unmatched\_TP0\_FACS))

write\_xlsx(ANOVA\_FACSdata, "ANOVA\_FACSdata.xlsx")

# with wilcox

ANOVA\_FACSall <- automate\_anova\_extraction(output\_location, "Wilcox\_FACS\_Plots",

"Wilcox\_FACS", "y", ANOVA\_FACSdata,

colnames(ANOVA\_FACSdata)[9:12],

"Timepoint")

ANOVA\_FACSdata <- ANOVA\_FACSdata[order(ANOVA\_FACSdata$Sex == "Female", decreasing = TRUE), ]

ANOVA\_FACSsex <- automate\_anova\_extraction\_Category(output\_location, "Wilcox\_FACS\_Plots\_Sex",

"Wilcox\_FACS\_Sex", "y", ANOVA\_FACSdata,

colnames(ANOVA\_FACSdata)[9:12],

c(colnames(ANOVA\_FACSdata)[2]),c(colnames(ANOVA\_FACSdata)[22]))

ANOVA\_FACSdata <- ANOVA\_FACSdata[order(ANOVA\_FACSdata$Category == "MODERATE", decreasing = FALSE), ]

ANOVA\_FACScategory <- automate\_anova\_extraction\_Category(output\_location, "Wilcox\_FACS\_Plots\_Category",

"Wilcox\_FACS\_Category", "y", ANOVA\_FACSdata,

colnames(ANOVA\_FACSdata)[9:12],

c(colnames(ANOVA\_FACSdata)[2]),c(colnames(ANOVA\_FACSdata)[24]))

ANOVA\_FACSdata <- ANOVA\_FACSdata[order(ANOVA\_FACSdata$Recovery == "Bad", decreasing = TRUE), ]

ANOVA\_FACSrecovery <- automate\_anova\_extraction\_Category(output\_location, "Wilcox\_FACS\_Plots\_Recovery",

"Wilcox\_FACS\_Recovery", "y", ANOVA\_FACSdata,

colnames(ANOVA\_FACSdata)[9:12],

c(colnames(ANOVA\_FACSdata)[2]),c(colnames(ANOVA\_FACSdata)[25]))

write\_xlsx(ANOVA\_FACSsex, "ANOVA\_FACSsex.xlsx")

write\_xlsx(ANOVA\_FACScategory, "ANOVA\_FACScategory.xlsx")

write\_xlsx(ANOVA\_FACSrecovery, "ANOVA\_FACSrecovery.xlsx")

ANOVA\_FACSall$Category <- "All"

ANOVA\_FACScombined <- rbind (ANOVA\_FACSall,ANOVA\_FACSsex)

ANOVA\_FACScombined <- rbind (ANOVA\_FACScombined,ANOVA\_FACSrecovery)

ANOVA\_FACScombined <- rbind (ANOVA\_FACScombined,ANOVA\_FACScategory)

write\_xlsx(ANOVA\_FACScombined, "ANOVA\_FACScombined.xlsx")

# paired t.test

# FACSdata\_matched <- FACSdata %>%

# filter(!SampleID %in% Unmatched\_TP0\_FACS)

#

# paired\_Ttest\_FACS <- automate\_ttest\_extraction(output\_location, "paired\_Ttest\_FACS\_Plots",

# "paired\_Ttest\_FACS", "y", FACSdata\_matched, colnames(FACSdata\_matched)[4:16],

# "Timepoint")

#### Mean & SEM FACS----------------------------------------------------------------------------------------------------

folder <- "MEAN\_SEM\_FACS\_Sex"

createFolder(folder)

plot\_save <- "y"

# Initialize an empty dataframe to store Wilcoxon test results

results\_FACS\_Wilcox <- data.frame(Cells = character(), Focus = character(),

TP0\_P\_Value = numeric(), TP1\_P\_Value = numeric(),

TP2\_P\_Value = numeric(), TP3\_P\_Value = numeric(),

TP4\_P\_Value = numeric(), stringsAsFactors = FALSE)

for (i in 9:12) {

cells\_colname <- colnames(FACSdata)[i]

# Filter, clean, and rename the data column for the current column

df\_subpop\_clean <- FACSdata %>%

filter(!is.na(Sex), !is.na(Timepoint)) %>%

select(Timepoint, Sex, !!sym(cells\_colname)) %>%

rename(data\_column = !!sym(cells\_colname))

p\_values <- c(TP0\_P\_Value = NA, TP1\_P\_Value = NA, TP2\_P\_Value = NA,

TP3\_P\_Value = NA, TP4\_P\_Value = NA)

# Calculate Wilcoxon test p-values for each timepoint

for (tp in unique(df\_subpop\_clean$Timepoint)) {

df\_timepoint <- subset(df\_subpop\_clean, Timepoint == tp)

if (length(unique(df\_timepoint$Sex)) == 2) { # Ensure both sexes are present

wilcox\_test <- wilcox.test(data\_column ~ Sex, data = df\_timepoint)

p\_values[paste0(tp, "\_P\_Value")] <- wilcox\_test$p.value

}

}

# Append p-values to results dataframe

results\_FACS\_Wilcox <- rbind(results\_FACS\_Wilcox,

data.frame(Cells = cells\_colname, Focus = "Male vs. Female",

t(p\_values), stringsAsFactors = FALSE))

# Generate line plot with mean ± SEM for each sex across Timepoints

p <- ggline(df\_subpop\_clean,

x = "Timepoint",

y = "data\_column",

color = "Sex",

add = "mean\_se", # Add mean and standard error

size = 1.2) +

labs(y = paste(cells\_colname, "percentage"),

title = paste("Mean ± SEM of", cells\_colname, "by Sex across Timepoints")) +

scale\_color\_manual(values = c("Male" = "#A67C00", "Female" = "#1D04C2")) +

theme\_bw() +

theme(

panel.grid.major = element\_line(size = 0.2, linetype = 'solid', color = "gray80"),

panel.grid.minor = element\_blank(),

panel.border = element\_blank(),

axis.line = element\_line(color = "black")

) +

scale\_x\_discrete(labels = c(

"TP0" = "Control",

"TP1" = "24 hours",

"TP2" = "3-5 days",

"TP3" = "1 month",

"TP4" = "3 months"

)) +

stat\_compare\_means(method = "wilcox.test", # or "t.test"

aes(group = Sex),

label = "p.signif",

size = 5)

# Save plot if specified

if (plot\_save == "y") {

file\_name\_facet <- paste0(folder, "/", cells\_colname, "\_Mean\_SEM\_SexComparison.png")

ggsave(filename = file\_name\_facet, plot = p)

}

}

folder <- "MEAN\_SEM\_FACS\_Category"

createFolder(folder)

plot\_save <- "y"

for (i in 9:12) {

cells\_colname <- colnames(FACSdata)[i]

# Filter, clean, and rename the data column for the current column

df\_subpop\_clean <- FACSdata %>%

filter(!is.na(Category), !is.na(Timepoint)) %>%

select(Timepoint, Category, !!sym(cells\_colname)) %>%

rename(data\_column = !!sym(cells\_colname))

p\_values <- c(TP0\_P\_Value = NA, TP1\_P\_Value = NA, TP2\_P\_Value = NA,

TP3\_P\_Value = NA, TP4\_P\_Value = NA)

# Calculate Wilcoxon test p-values for each timepoint

for (tp in unique(df\_subpop\_clean$Timepoint)) {

df\_timepoint <- subset(df\_subpop\_clean, Timepoint == tp)

if (length(unique(df\_timepoint$Category)) == 2) { # Ensure both sexes are present

wilcox\_test <- wilcox.test(data\_column ~ Category, data = df\_timepoint)

p\_values[paste0(tp, "\_P\_Value")] <- wilcox\_test$p.value

}

}

# Append p-values to results dataframe

results\_FACS\_Wilcox <- rbind(results\_FACS\_Wilcox,

data.frame(Cells = cells\_colname, Focus = "Minor vs. Moderate",

t(p\_values), stringsAsFactors = FALSE))

# Generate line plot with mean ± SEM for each sex across Timepoints

p <- ggline(df\_subpop\_clean,

x = "Timepoint",

y = "data\_column",

color = "Category",

add = "mean\_se", # Add mean and standard error

size = 1.2) +

labs(y = paste(cells\_colname, "percentage"),

title = paste("Mean ± SEM of", cells\_colname, "by Category across Timepoints")) +

scale\_color\_manual(values = c("MINOR" = "#A67C00", "MODERATE" = "#700606")) +

scale\_x\_discrete(labels = c(

"TP0" = "Control",

"TP1" = "24 hours",

"TP2" = "3-5 days",

"TP3" = "1 month",

"TP4" = "3 months"

)) +

theme\_bw() +

theme(

panel.grid.major = element\_line(size = 0.2, linetype = 'solid', color = "gray80"),

panel.grid.minor = element\_blank(),

panel.border = element\_blank(),

axis.line = element\_line(color = "black")

) +

stat\_compare\_means(method = "wilcox.test", # or "t.test"

aes(group = Category),

label = "p.signif",

size = 5)

# Save plot if specified

if (plot\_save == "y") {

file\_name\_facet <- paste0(folder, "/", cells\_colname, "\_Mean\_SEM\_CategoryComparison.png")

ggsave(filename = file\_name\_facet, plot = p)

}

}

folder <- "MEAN\_SEM\_FACS\_Recovery"

createFolder(folder)

plot\_save <- "y"

for (i in 9:12) {

cells\_colname <- colnames(FACSdata)[i]

# Filter, clean, and rename the data column for the current column

df\_subpop\_clean <- FACSdata %>%

filter(!is.na(Recovery), !is.na(Timepoint)) %>%

select(Timepoint, Recovery, !!sym(cells\_colname)) %>%

rename(data\_column = !!sym(cells\_colname))

p\_values <- c(TP0\_P\_Value = NA, TP1\_P\_Value = NA, TP2\_P\_Value = NA,

TP3\_P\_Value = NA, TP4\_P\_Value = NA)

for (tp in unique(df\_subpop\_clean$Timepoint)) {

df\_timepoint <- subset(df\_subpop\_clean, Timepoint == tp)

if (length(unique(df\_timepoint$Recovery)) == 2) { # Ensure both sexes are present

wilcox\_test <- wilcox.test(data\_column ~ Recovery, data = df\_timepoint)

p\_values[paste0(tp, "\_P\_Value")] <- wilcox\_test$p.value

}

}

results\_FACS\_Wilcox <- rbind(results\_FACS\_Wilcox,

data.frame(Cells = cells\_colname, Focus = "Good vs. Bad Recovery",

t(p\_values), stringsAsFactors = FALSE))

max\_y\_value <- mean(df\_subpop\_clean$data\_column, na.rm = TRUE)+5

p <- ggline(df\_subpop\_clean,

x = "Timepoint",

y = "data\_column",

color = "Recovery",

add = "mean\_se", # Add mean and standard error

size = 1.2) +

labs(y = paste(cells\_colname, "percentage"),

title = paste("Mean ± SEM of", cells\_colname, "by Recovery across Timepoints")) +

scale\_color\_manual(values = c("Good" = "darkgreen", "Bad" = "#1D04C2")) +

theme\_bw() +

#scale\_y\_continuous(limits = c(0, max\_y\_value)) + # Dynamically set y-axis limits

theme(

panel.grid.major = element\_line(size = 0.2, linetype = 'solid', color = "gray80"),

panel.grid.minor = element\_blank(),

panel.border = element\_blank(),

axis.line = element\_line(color = "black")

) +

stat\_compare\_means(method = "wilcox.test", # or "t.test"

aes(group = Recovery),

label = "p.signif",

size = 5)

# Save plot if specified

if (plot\_save == "y") {

file\_name\_facet <- paste0(folder, "/", cells\_colname, "\_Mean\_SEM\_RecoveryComparison.png")

ggsave(filename = file\_name\_facet, plot = p)

}

}

ggline(df\_subpop\_clean,

x = "Timepoint",

y = "data\_column",

color = "Recovery",

add = "mean\_se", # Add mean and standard error

size = 1.2,

ylim = c(5, 25)) +

labs(y = paste(cells\_colname, "percentage"),

title = paste("Mean ± SEM of", cells\_colname, "by Recovery across Timepoints")) +

scale\_color\_manual(values = c("Good" = "darkgreen", "Bad" = "#1D04C2")) +

theme\_bw() +

theme(

panel.grid.major = element\_line(size = 0.2, linetype = 'solid', color = "gray80"),

panel.grid.minor = element\_blank(),

panel.border = element\_blank(),

axis.line = element\_line(color = "black")

) +

scale\_x\_discrete(labels = c(

"TP0" = "Control",

"TP1" = "24 hours",

"TP2" = "3-5 days",

"TP3" = "1 month",

"TP4" = "3 months"

)) +

stat\_compare\_means(method = "wilcox.test", # or "t.test"

aes(group = Recovery),

label = "p.signif",

size = 5,

label.y = 25)

ggsave(filename = "NonClassical Monocytes\_Mean\_SEM\_RecoveryComparison.png")

results\_FACS\_Wilcox$TP0\_Significance <- sapply(results\_FACS\_Wilcox$TP0\_P\_Value, get\_significance)

results\_FACS\_Wilcox$TP1\_Significance <- sapply(results\_FACS\_Wilcox$TP1\_P\_Value, get\_significance)

results\_FACS\_Wilcox$TP2\_Significance <- sapply(results\_FACS\_Wilcox$TP2\_P\_Value, get\_significance)

results\_FACS\_Wilcox$TP3\_Significance <- sapply(results\_FACS\_Wilcox$TP3\_P\_Value, get\_significance)

results\_FACS\_Wilcox$TP4\_Significance <- sapply(results\_FACS\_Wilcox$TP4\_P\_Value, get\_significance)

write.csv(results\_FACS\_Wilcox, file = "results\_FACS\_Wilcox.csv", row.names = FALSE)

#### Linear regression -------------------------------------

output\_folder <- "LinReg\_FACS\_Plots\_Age"

if (!dir.exists(output\_folder)) {

dir.create(output\_folder, recursive = TRUE)

}

# Define custom colors for the Timepoints

my\_colors <- c("TP0" = "darkgreen", "TP1" = "orange", "TP2" = "red", "TP3" = "magenta", "TP4" = "purple")

# Initialize data frames to store results

Age\_correlation\_FACS <- data.frame(Cells = character(), Timepoint = character(), Sex = character(), N = numeric(),

p.value = numeric(), Coefficient = numeric(), Down95 = numeric(),

Up95 = numeric(), Intercept = numeric(), Fold = numeric(),

Max\_age = numeric(), Min\_age = numeric())

# remove Patient40 as I do not have age and Sex from them

#FACSdata <- FACSdata[!is.na(FACSdata$Age), ]

# Main loop for each column

for (i in 9:12) {

cells\_colname <- colnames(FACSdata)[i]

# All samples

plot\_linear\_regression1(FACSdata, cells\_colname, "All", output\_folder)

# Male samples

FACSdata\_male <- FACSdata[FACSdata$Sex == "Male", ]

if (nrow(FACSdata\_male) > 0) {

plot\_linear\_regression(FACSdata\_male, cells\_colname, "Male", output\_folder)

}

# Female samples

FACSdata\_female <- FACSdata[FACSdata$Sex == "Female", ]

if (nrow(FACSdata\_female) > 0) {

plot\_linear\_regression(FACSdata\_female, cells\_colname, "Female", output\_folder)

}

# Minor category

FACSdata\_Minor <- FACSdata[FACSdata$Category == "MINOR", ]

if (nrow(FACSdata\_Minor) > 0) {

plot\_linear\_regression(FACSdata\_Minor, cells\_colname, "Minor", output\_folder)

}

# Moderate category

FACSdata\_Moderate <- FACSdata[FACSdata$Category == "MODERATE", ]

if (nrow(FACSdata\_Moderate) > 0) {

plot\_linear\_regression(FACSdata\_Moderate, cells\_colname, "Moderate", output\_folder)

}

# Good recovery

FACSdata\_Good <- FACSdata[FACSdata$Recovery == "Good", ]

FACSdata\_Good <- FACSdata\_Good[!is.na(FACSdata\_Good$Age), ]

if (nrow(FACSdata\_Good) > 0) {

plot\_linear\_regression(FACSdata\_Good, cells\_colname, "Good", output\_folder)

}

# Bad recovery

FACSdata\_Bad <- FACSdata[FACSdata$Recovery == "Bad", ]

FACSdata\_Bad <- FACSdata\_Bad[!is.na(FACSdata\_Bad$Age), ]

if (nrow(FACSdata\_Bad) > 0) {

plot\_linear\_regression(FACSdata\_Bad, cells\_colname, "Bad", output\_folder)

}

}

# Save results to CSV

write.csv(Age\_correlation\_FACS, "Age\_correlation\_FACS.csv", row.names = FALSE)

sigAge\_correlation\_FACS <- Age\_correlation\_FACS %>% filter(Age\_correlation\_FACS$p.value <0.05)

write.csv(sigAge\_correlation\_FACS, "sigAge\_correlation\_FACS.csv", row.names = FALSE)

#### NHISS linear regression -----------------

output\_folder <- "LinReg\_FACS\_Plots\_NHISS"

if (!dir.exists(output\_folder)) {

dir.create(output\_folder, recursive = TRUE)

}

NHISS\_correlation\_FACS <- data.frame(Cells = character(), Timepoint = character(), Sex = character(), N = numeric(),

p.value = numeric(), Coefficient = numeric(), Down95 = numeric(),

Up95 = numeric(), Intercept = numeric(), Fold = numeric(),

Max\_NHISS = numeric(), Min\_NHISS = numeric())

# remove Patient40 as I do not have NHISS and Sex from them

#FACSdata <- FACSdata[!is.na(FACSdata$NHISS), ]

# Main loop for each column

for (i in 9:12) {

cells\_colname <- colnames(FACSdata)[i]

# All samples

NHISS\_correlation\_FACS <- plot\_linear\_regression\_NHISS(FACSdata, cells\_colname, "All", output\_folder, NHISS\_correlation\_FACS, "NHISS")

# # Male samples

# FACSdata\_male <- FACSdata[FACSdata$Sex == "Male", ]

# if (nrow(FACSdata\_male) > 0) {

# plot\_linear\_regression\_NHISS(FACSdata\_male, cells\_colname, "Male", output\_folder, NHISS\_correlation\_FACS)

# }

#

# # Female samples

# FACSdata\_female <- FACSdata[FACSdata$Sex == "Female", ]

# if (nrow(FACSdata\_female) > 0) {

# plot\_linear\_regression\_NHISS(FACSdata\_female, cells\_colname, "Female", output\_folder, NHISS\_correlation\_FACS)

# }

#

# # Minor category

# FACSdata\_Minor <- FACSdata[FACSdata$Category == "MINOR", ]

# if (nrow(FACSdata\_Minor) > 0) {

# plot\_linear\_regression\_NHISS(FACSdata\_Minor, cells\_colname, "Minor", output\_folder, NHISS\_correlation\_FACS)

# }

#

# # Moderate category

# FACSdata\_Moderate <- FACSdata[FACSdata$Category == "MODERATE", ]

# if (nrow(FACSdata\_Moderate) > 0) {

# plot\_linear\_regression\_NHISS(FACSdata\_Moderate, cells\_colname, "Moderate", output\_folder, NHISS\_correlation\_FACS)

# }

#

# # Good recovery

# FACSdata\_Good <- FACSdata[FACSdata$Recovery == "Good", ]

# FACSdata\_Good <- FACSdata\_Good[!is.na(FACSdata\_Good$Age), ]

# if (nrow(FACSdata\_Good) > 0) {

# plot\_linear\_regression\_NHISS(FACSdata\_Good, cells\_colname, "Good", output\_folder, NHISS\_correlation\_FACS)

# }

#

# # Bad recovery

# FACSdata\_Bad <- FACSdata[FACSdata$Recovery == "Bad", ]

# FACSdata\_Bad <- FACSdata\_Bad[!is.na(FACSdata\_Bad$Age), ]

# if (nrow(FACSdata\_Bad) > 0) {

# plot\_linear\_regression\_NHISS(FACSdata\_Bad, cells\_colname, "Bad", output\_folder, NHISS\_correlation\_FACS)

#}

}

# Save results to CSV

write.csv(NHISS\_correlation\_FACS, "NHISS\_correlation\_FACS.csv", row.names = FALSE)

sigNHISS\_correlation\_FACS <- NHISS\_correlation\_FACS %>% filter(NHISS\_correlation\_FACS$p.value <0.05)

write.csv(sigNHISS\_correlation\_FACS, "sigNHISS\_correlation\_FACS.csv", row.names = FALSE)

#### NHISS\_End linear regression -----------------

output\_folder <- "LinReg\_FACS\_Plots\_NHISS\_End"

if (!dir.exists(output\_folder)) {

dir.create(output\_folder, recursive = TRUE)

}

# Correlation with "final" NHISS

# Initialize NHISS\_End with NA

FACSdata$NHISS\_End <- NA

# Assign NHISS\_End values for all timepoints based on TP4

TP4\_indices <- which(FACSdata$Timepoint == "TP4")

TP4\_values <- FACSdata$NHISS[TP4\_indices]

# Define timepoints to adjust

timepoints\_to\_adjust <- c("TP3", "TP2", "TP1")

# Loop through each timepoint and assign NHISS\_End

for (tp in timepoints\_to\_adjust) {

current\_indices <- which(FACSdata$Timepoint == tp)

min\_length <- min(length(current\_indices), length(TP4\_indices))

# Assign TP4 values to the current timepoint

FACSdata$NHISS\_End[current\_indices[1:min\_length]] <- TP4\_values[1:min\_length]

# Fill remaining rows with NA if TP4 has fewer values

if (length(current\_indices) > min\_length) {

FACSdata$NHISS\_End[current\_indices[(min\_length + 1):length(current\_indices)]] <- NA

}

}

# Ensure TP4 values are copied correctly

FACSdata$NHISS\_End[TP4\_indices] <- TP4\_values

FACSdata\_copy <- FACSdata

NHISS\_End\_correlation\_FACS <- data.frame(Cells = character(), Timepoint = character(), Sex = character(), N = numeric(),

p.value = numeric(), Coefficient = numeric(), Down95 = numeric(),

Up95 = numeric(), Intercept = numeric(), Fold = numeric(),

Max\_NHISS\_End = numeric(), Min\_NHISS\_End = numeric())

# Main loop for each column

for (i in 9:12) {

cells\_colname <- colnames(FACSdata)[i]

# Remove rows with NA in NHISS\_End

FACSdata <- FACSdata[!is.na(FACSdata$NHISS\_End), ]

# All samples

NHISS\_End\_correlation\_FACS <- plot\_linear\_regression\_NHISS(

FACSdata,

cells\_colname,

"All",

output\_folder,

NHISS\_End\_correlation\_FACS,

"NHISS\_End" # Pass NHISS\_End as a string

)

}

# Save results to CSV

write.csv(NHISS\_End\_correlation\_FACS, "NHISS\_End\_correlation\_FACS.csv", row.names = FALSE)

sigNHISS\_End\_correlation\_FACS <- NHISS\_End\_correlation\_FACS %>% filter(NHISS\_End\_correlation\_FACS$p.value <0.05)

write.csv(sigNHISS\_End\_correlation\_FACS, "sigNHISS\_End\_correlation\_FACS.csv", row.names = FALSE)

#### NHISS\_Diff linear regression -----------------

output\_folder <- "LinReg\_FACS\_Plots\_NHISS\_Diff"

if (!dir.exists(output\_folder)) {

dir.create(output\_folder, recursive = TRUE)

}

FACSdata <- FACSdata\_copy

# Initialize NHISS\_Diff with NA

FACSdata$NHISS\_Diff <- NA

# Calculate the difference for each SampleID

FACSdata <- FACSdata %>%

group\_by(SampleID) %>%

mutate(

NHISS\_Diff = if (all(c("TP1", "TP4") %in% Timepoint)) {

NHISS[Timepoint == "TP1"] - NHISS[Timepoint == "TP4"]

} else {

NA

}

) %>%

ungroup()

FACSdata\_copy <- FACSdata

NHISS\_Diff\_correlation\_FACS <- data.frame(Cells = character(), Timepoint = character(), Sex = character(), N = numeric(),

p.value = numeric(), Coefficient = numeric(), Down95 = numeric(),

Up95 = numeric(), Intercept = numeric(), Fold = numeric(),

Max\_NHISS\_Diff = numeric(), Min\_NHISS\_Diff = numeric())

# Main loop for each column

for (i in 9:12) {

cells\_colname <- colnames(FACSdata)[i]

# Remove rows with NA in NHISS\_End

FACSdata <- FACSdata[!is.na(FACSdata$NHISS\_Diff), ]

# All samples

NHISS\_Diff\_correlation\_FACS <- plot\_linear\_regression\_NHISS(

FACSdata,

cells\_colname,

"All",

output\_folder,

NHISS\_End\_correlation\_FACS,

"NHISS\_Diff" # Pass NHISS\_End as a string

)

}

# Save results to CSV

write.csv(NHISS\_Diff\_correlation\_FACS, "NHISS\_Diff\_correlation\_FACS.csv", row.names = FALSE)

sigNHISS\_Diff\_correlation\_FACS <- NHISS\_Diff\_correlation\_FACS %>% filter(NHISS\_Diff\_correlation\_FACS$p.value <0.05)

write.csv(sigNHISS\_Diff\_correlation\_FACS, "sigNHISS\_Diff\_correlation\_FACS.csv", row.names = FALSE)

#### NHISS\_Ratio linear regression -----------------

output\_folder <- "LinReg\_FACS\_Plots\_NHISS\_Ratio"

if (!dir.exists(output\_folder)) {

dir.create(output\_folder, recursive = TRUE)

}

FACSdata <- FACSdata\_copy

# Initialize NHISS\_Ratio with NA

FACSdata$NHISS\_Ratio <- NA

# Calculate the difference for each SampleID

FACSdata <- FACSdata %>%

group\_by(SampleID) %>%

mutate(

NHISS\_Ratio = if (all(c("TP1", "TP4") %in% Timepoint)) {

(NHISS[Timepoint == "TP1"] - NHISS[Timepoint == "TP4"])/NHISS[Timepoint == "TP1"]

} else {

NA

}

) %>%

ungroup()

FACSdata\_copy <- FACSdata

NHISS\_Ratio\_correlation\_FACS <- data.frame(Cells = character(), Timepoint = character(), Sex = character(), N = numeric(),

p.value = numeric(), Coefficient = numeric(), Down95 = numeric(),

Up95 = numeric(), Intercept = numeric(), Fold = numeric(),

Max\_NHISS\_Ratio = numeric(), Min\_NHISS\_Ratio = numeric())

# Main loop for each column

for (i in 9:12) {

cells\_colname <- colnames(FACSdata)[i]

# Remove rows with NA in NHISS\_End

FACSdata <- FACSdata[!is.na(FACSdata$NHISS\_Ratio), ]

# All samples

NHISS\_Ratio\_correlation\_FACS <- plot\_linear\_regression\_NHISS(

FACSdata,

cells\_colname,

"All",

output\_folder,

NHISS\_End\_correlation\_FACS,

"NHISS\_Ratio" # Pass NHISS\_End as a string

)

}

# Save results to CSV

write.csv(NHISS\_Ratio\_correlation\_FACS, "NHISS\_Ratio\_correlation\_FACS.csv", row.names = FALSE)

sigNHISS\_Ratio\_correlation\_FACS <- NHISS\_Ratio\_correlation\_FACS %>% filter(NHISS\_Ratio\_correlation\_FACS$p.value <0.05)

write.csv(sigNHISS\_Ratio\_correlation\_FACS, "sigNHISS\_Ratio\_correlation\_FACS.csv", row.names = FALSE)

# restore original dataframe, as its randomly deleting things....

FACSdata <- FACSdata\_copy

# \*Gene Expr. Statistics\* ----------------------------------------------------------------------------------------------------

# As they are non-normal disributed, dependent with similar variance

# Adjust to yes if you want plots to be saved or n if not

plot\_save <- "y"

# remove unmatched CTR for ANOVA too

data\_mean\_matched <- dataFINALmean %>% filter(!(Timepoint == "TP0" & !SampleID %in% Matched\_TP0\_Gene))

# Check if all groups have the same size:

check\_sample\_counts(data\_mean\_matched)

TP\_colors <- c("TP0" = "darkgreen", "TP1" = "orange", "TP2" = "red", "TP3" = "magenta", "TP4" = "purple")

#### Wilcox test for Z of Genes --------------------------------------------------

folder <- "Wilcox\_capZ\_Plots\_unpaired"

results <- data.frame(Gene = character(), Subpopulation = character(), P\_Value = numeric(),

TP1\_P\_Value = numeric(), TP2\_P\_Value = numeric(), TP3\_P\_Value = numeric(), TP4\_P\_Value = numeric())

check\_sample\_counts(data\_mean\_matched)

j = 1

for (j in 1:length(unique(data\_mean\_matched$Gene))) {

gene <- data\_mean\_matched$Gene[j]

df\_gene <- filter(data\_mean\_matched, Gene == gene)

df\_gene <- df\_gene %>% filter(!is.na(Category), !is.na(mean\_capZ), !is.na(Subpopulation), !is.na(Timepoint))

i = 1

for (i in 1:length(unique(df\_gene$Subpopulation))) {

sup <- df\_gene$Subpopulation[i]

df <- filter(df\_gene, Subpopulation == sup)

if (nrow(df) <= 0 || gene %in% exclude\_genes) {

cat("Skipping:", gene, "for subpopulation:", sup, "due to insufficient data or in excluded genes list\n")

next # Skip to the next iteration of the loop

}

# you can only exclud outliers if you are not using paired

df <- df[!df$mean\_capZ %in% boxplot.stats(df$mean\_capZ)$out, ]

df$Timepoint <- factor(as.vector(df$Timepoint))

baseline <- filter(df, Timepoint == "TP0")

median\_TP0 <- median(baseline$mean\_capZ)

# Perform Wilcoxon test for each comparison with TP0

timepoints <- c("TP1", "TP2", "TP3", "TP4")

tp\_p\_values <- c(NA, NA, NA, NA) # Initialize p-values for TP1 to TP4

# Loop through each timepoint and compare to TP0

for (tp in seq\_along(timepoints)) {

comparison\_tp <- timepoints[tp]

# Check if there are enough data for both TP0 and the current timepoint

if (nrow(filter(df, Timepoint == comparison\_tp)) > 1 && nrow(baseline) > 1) {

# Perform Wilcoxon test between TP0 and the current timepoint

test\_result <- wilcox.test(df$mean\_capZ[df$Timepoint == "TP0"],

df$mean\_capZ[df$Timepoint == comparison\_tp],

paired = FALSE, exact = TRUE)

tp\_p\_values[tp] <- test\_result$p.value # Store the p-value

}

}

# Save results to data frame

p\_value\_min <- min(tp\_p\_values, na.rm = TRUE)

results <- rbind(results, data.frame(Gene = gene, Subpopulation = sup,

P\_Value = min(tp\_p\_values, na.rm = TRUE), # Store the smallest p-value as general

TP1\_P\_Value = tp\_p\_values[1],

TP2\_P\_Value = tp\_p\_values[2],

TP3\_P\_Value = tp\_p\_values[3],

TP4\_P\_Value = tp\_p\_values[4]))

# Plot only if there is a significant p-value (<= 0.05)

# plot\_save <- "n"

# if (p\_value\_min <= 0.05) {

# plot\_save <- "y"

# }

#

if (plot\_save == "y") {

#my\_colors <- c("darkgreen", "orange", "red", "magenta", "purple")

plot\_name <- paste(gene, " expression of ", sup, " Monocytes (Control vs post-stroke", sep = "")

# Create a boxplot with significance levels

p <- ggboxplot(df,

x = "Timepoint",

y = "mean\_capZ",

color = "Timepoint",

add = "jitter",

legend = "none",

ylab = paste(gene, "expression (Z-scored from CT - CT Sample Median)"),

title = plot\_name,

width = 0.8,

add.params = list(size = 1, alpha = 0.5)) +

geom\_hline(yintercept = median\_TP0, linetype = 2) +

scale\_color\_manual(values = TP\_colors) +

scale\_x\_discrete(labels = c(

"TP0" = "Control",

"TP1" = "24 hours",

"TP2" = "3-5 days",

"TP3" = "1 month",

"TP4" = "3 months"))

# Add significance levels based on Wilcoxon tests

for (tp in seq\_along(timepoints)) {

comparison\_tp <- timepoints[tp]

p\_value <- tp\_p\_values[tp]

if (!is.na(p\_value)) {

# Use 'p' for raw p-value or 'p.signif' for significance symbols

p <- p + stat\_compare\_means(method = "wilcox",

ref.group = "TP0",

hide.ns = TRUE,

label = "p.signif", # You can use 'p.signif' if you prefer symbols

label.y = max(df$mean\_capZ)) # Adjust label.y as needed

}

}

# Save the plot

file\_name <- file.path(folder, paste(gene, "\_", sup, "\_capZ\_Wilcox.png", sep = ""))

ggsave(filename = file\_name, plot = p)

}

}

}

results$Significance <- sapply(results$P\_Value, get\_significance)

results$TP1\_Significance <- sapply(results$TP1\_P\_Value, get\_significance)

results$TP2\_Significance <- sapply(results$TP2\_P\_Value, get\_significance)

results$TP3\_Significance <- sapply(results$TP3\_P\_Value, get\_significance)

results$TP4\_Significance <- sapply(results$TP4\_P\_Value, get\_significance)

Wilcox\_capZ <- results

write.csv(Wilcox\_capZ, file = "Wilcox\_Results\_capZ\_unpaired.csv", row.names = FALSE)

#### Wilcox test for Z of Genes -SubtypesCombined --------------------------------------------------

folder <- "Wilcox\_capZ\_Plots\_SubtypesCombined\_unpaired"

results <- data.frame(Gene = character(), Subpopulation = character(), P\_Value = numeric(),

TP1\_P\_Value = numeric(), TP2\_P\_Value = numeric(), TP3\_P\_Value = numeric(), TP4\_P\_Value = numeric())

check\_sample\_counts(data\_mean\_matched)

SP\_colors <- c("all" = "black", "classical" = "red", "intermediate" = "purple", "nonclassical" = "blue")

# Define new x-axis labels

timepoint\_labels <- c(

"TP0" = "Healthy",

"TP1" = "24 hours post-stroke",

"TP2" = "3-5 days post-stroke",

"TP3" = "1 month post-stroke",

"TP4" = "3 months post-stroke"

)

for (j in 1:length(unique(data\_mean\_matched$Gene))) {

gene <- data\_mean\_matched$Gene[j]

df\_gene <- filter(data\_mean\_matched, Gene == gene)

df\_gene <- df\_gene %>% filter(!is.na(Category), !is.na(mean\_capZ), !is.na(Subpopulation), !is.na(Timepoint))

if (nrow(df\_gene) <= 0 || gene %in% exclude\_genes) {

cat("Skipping:", gene, "due to insufficient data or in excluded genes list\n")

next

}

# Exclude outliers

df\_gene <- df\_gene[!df\_gene$mean\_capZ %in% boxplot.stats(df\_gene$mean\_capZ)$out, ]

df\_gene$Timepoint <- factor(as.vector(df\_gene$Timepoint))

df\_gene$Subpopulation <- factor(df\_gene$Subpopulation, levels = names(SP\_colors))

baseline <- filter(df\_gene, Timepoint == "TP0")

median\_TP0 <- median(baseline$mean\_capZ)

# Perform Wilcoxon test for each comparison with TP0, for each Subpopulation

timepoints <- c("TP1", "TP2", "TP3", "TP4")

tp\_p\_values <- data.frame(Subpopulation = unique(df\_gene$Subpopulation),

TP1 = NA, TP2 = NA, TP3 = NA, TP4 = NA)

for (subpop in unique(df\_gene$Subpopulation)) {

for (tp in seq\_along(timepoints)) {

comparison\_tp <- timepoints[tp]

df\_subpop <- filter(df\_gene, Subpopulation == subpop)

baseline\_subpop <- filter(df\_subpop, Timepoint == "TP0")

if (nrow(filter(df\_subpop, Timepoint == comparison\_tp)) > 1 && nrow(baseline\_subpop) > 1) {

test\_result <- wilcox.test(df\_subpop$mean\_capZ[df\_subpop$Timepoint == "TP0"],

df\_subpop$mean\_capZ[df\_subpop$Timepoint == comparison\_tp],

paired = FALSE, exact = TRUE)

tp\_p\_values[tp\_p\_values$Subpopulation == subpop, timepoints[tp]] <- test\_result$p.value

}

}

}

# Save results to data frame

results <- rbind(results, data.frame(Gene = gene, tp\_p\_values))

# Calculate the y-axis limit dynamically

y\_max <- max(df\_gene$mean\_capZ, na.rm = TRUE)

y\_extend <- 0.5 + 0.2 \* length(unique(df\_gene$Subpopulation)) # Additional space for annotations

y\_axis\_limit <- y\_max + y\_extend

# Create a boxplot for all subpopulations

p <- ggboxplot(df\_gene,

x = "Timepoint",

y = "mean\_capZ",

color = "Subpopulation",

add = "jitter",

legend.title = "Subpopulation",

ylab = paste(gene, "expression (Z-scored from CT - CT Sample Median)"),

width = 0.5,

add.params = list(size = 1, alpha = 0.5)) +

scale\_color\_manual(values = SP\_colors) +

scale\_x\_discrete(labels = timepoint\_labels) +

ylim(-3, 3) + # Set the y-axis limit dynamically

labs(title = paste("Gene:", gene),

subtitle = "Expression across timepoints and subpopulations")

# Add asterisks for significant comparisons

for (tp in seq\_along(timepoints)) {

comparison\_tp <- timepoints[tp]

y\_base <- 2.6

y\_step <- 0.1 # Vertical step between asterisks

for (subpop\_idx in seq\_along(unique(df\_gene$Subpopulation))) {

subpop <- unique(df\_gene$Subpopulation)[subpop\_idx]

p\_value <- tp\_p\_values[tp\_p\_values$Subpopulation == subpop, comparison\_tp]

if (!is.na(p\_value) && p\_value <= 0.05) {

asterisk <- ifelse(p\_value <= 0.001, "\*\*\*",

ifelse(p\_value <= 0.01, "\*\*",

ifelse(p\_value <= 0.05, "\*", "")))

if (asterisk != "") {

p <- p + annotate("text",

x = which(levels(df\_gene$Timepoint) == comparison\_tp),

y = y\_base + y\_step \* (subpop\_idx - 0.1), # Increment y for each subpopulation

label = asterisk,

color = SP\_colors[subpop],

size = 5) # Adjust size as needed

}

}

}

}

# Add geom\_hline for each subpopulation

# Add geom\_hline for each subpopulation using median TP0 values

for (subpop in unique(df\_gene$Subpopulation)) {

subpop\_data <- filter(df\_gene, Subpopulation == subpop & Timepoint == "TP0")

median\_TP0 <- median(subpop\_data$mean\_capZ, na.rm = TRUE) # Calculate the median TP0 value for the subpopulation

if (!is.na(median\_TP0)) {

p <- p + geom\_hline(yintercept = median\_TP0,

linetype = 2,

color = SP\_colors[subpop],

size = 0.5) # Adjust line thickness as needed

}

}

# Save the plot

file\_name <- file.path(folder, paste(gene, "\_capZ\_Wilcoxon\_Subpopulations.png", sep = ""))

ggsave(filename = file\_name, plot = p)

}

# results$Significance <- sapply(results$P\_Value, get\_significance)

# results$TP1\_Significance <- sapply(results$TP1\_P\_Value, get\_significance)

# results$TP2\_Significance <- sapply(results$TP2\_P\_Value, get\_significance)

# results$TP3\_Significance <- sapply(results$TP3\_P\_Value, get\_significance)

# results$TP4\_Significance <- sapply(results$TP4\_P\_Value, get\_significance)

#

# Wilcox\_capZ\_SP <- results

# write.csv(Wilcox\_capZ, file = "Wilcox\_Results\_capZ\_unpaired.csv", row.names = FALSE)

#\*Wilcox for Comparison at individual TPs -------------------------------------

plot\_save <- "y"

#### Male vs Female Analysis (Mean ± SEM) -------------------------------------

folder <- "Mean\_SEM\_capZ\_Sex\_Plots"

createFolder(folder) # Ensure the folder exists

results <- data.frame(Sex = character(), Gene = character(), Subpopulation = character(),

P\_Value = numeric(), TP1\_P\_Value = numeric(), TP2\_P\_Value = numeric(),

TP3\_P\_Value = numeric(), TP4\_P\_Value = numeric())

for (i in 1:length(unique(data\_mean\_matched$Gene))) {

gene <- unique(data\_mean\_matched$Gene)[i]

df\_gene <- filter(data\_mean\_matched, Gene == gene)

df\_gene <- df\_gene %>% filter(!is.na(Timepoint), !is.na(mean\_capZ), !is.na(Subpopulation), !is.na(Sex))

for (j in 1:length(unique(df\_gene$Subpopulation))) {

sup <- unique(df\_gene$Subpopulation)[j]

df\_subpop <- filter(df\_gene, Subpopulation == sup)

# Remove outliers

df\_subpop\_clean <- df\_subpop[!df\_subpop$mean\_capZ %in% boxplot.stats(df\_subpop$mean\_capZ)$out, ]

if (nrow(df\_subpop\_clean) <= 0) next # Skip if no rows remain

tryCatch({

title\_facet <- paste(gene, "expression in", sup, "monocytes (Zscored dCT)", sep=" ")

# Line plot comparing Male vs Female with mean ± SEM across Timepoints

ggline(subset(df\_subpop\_clean, !is.na(Sex)),

x = "Timepoint",

y = "mean\_capZ",

color = "Sex",

add = "mean\_se", # Add mean and standard error

size = 1.2) +

labs(y = paste(gene, "expression (Z-scored from CT - CT Sample Median)"),

title = title\_facet) +

scale\_color\_manual(values = c("Male" = "#A67C00", "Female" = "#1D04C2")) +

theme\_bw() + # Apply white background

theme(panel.grid.major = element\_line(size = 0.2, linetype = 'solid', color = "gray80"),

panel.grid.minor = element\_blank(),

panel.border = element\_blank(),

axis.line = element\_line(color = "black")) +

stat\_compare\_means(method = "wilcox.test", # or "t.test"

aes(group = Sex),

label = "p.signif",

size = 5)

if (plot\_save == "y") {

file\_name\_facet <- paste0(folder, "/", title\_facet, "\_Mean\_SEM.png")

ggsave(filename = file\_name\_facet)

}

# Initialize a list to store p-values for timepoints

p\_values\_list <- list(TP0 = NA, TP1 = NA, TP2 = NA, TP3 = NA, TP4 = NA)

# Extract p-values for each timepoint comparison (pairwise Wilcoxon)

timepoints <- c("TP0", "TP1", "TP2", "TP3", "TP4")

for (tp in timepoints) {

male\_values <- df\_subpop\_clean$mean\_capZ[df\_subpop\_clean$Timepoint == tp & df\_subpop\_clean$Sex == "Male"]

female\_values <- df\_subpop\_clean$mean\_capZ[df\_subpop\_clean$Timepoint == tp & df\_subpop\_clean$Sex == "Female"]

# Perform Wilcoxon test for the comparison of Male vs Female at each timepoint

if (length(male\_values) > 0 && length(female\_values) > 0) {

wilcox\_result <- wilcox.test(male\_values, female\_values)

p\_values\_list[[tp]] <- wilcox\_result$p.value

}

}

# Add results for Male vs Female for all timepoints as a single row

results <- rbind(results, data.frame(WilcoxComparison = "Male vs Female", Gene = gene, Subpopulation = sup,

TP0\_P\_Value = p\_values\_list[["TP0"]],

TP1\_P\_Value = p\_values\_list[["TP1"]],

TP2\_P\_Value = p\_values\_list[["TP2"]],

TP3\_P\_Value = p\_values\_list[["TP3"]],

TP4\_P\_Value = p\_values\_list[["TP4"]]))

}, error = function(e) {

cat("Error processing", gene, ":", e$message, "\n")

})

}

}

Timecourse\_Wilcox\_Sex\_Z <- results

Timecourse\_Wilcox\_Sex\_Z$TP0\_Significance <- sapply(Timecourse\_Wilcox\_Sex\_Z$TP0\_P\_Value, get\_significance)

Timecourse\_Wilcox\_Sex\_Z$TP1\_Significance <- sapply(Timecourse\_Wilcox\_Sex\_Z$TP1\_P\_Value, get\_significance)

Timecourse\_Wilcox\_Sex\_Z$TP2\_Significance <- sapply(Timecourse\_Wilcox\_Sex\_Z$TP2\_P\_Value, get\_significance)

Timecourse\_Wilcox\_Sex\_Z$TP3\_Significance <- sapply(Timecourse\_Wilcox\_Sex\_Z$TP3\_P\_Value, get\_significance)

Timecourse\_Wilcox\_Sex\_Z$TP4\_Significance <- sapply(Timecourse\_Wilcox\_Sex\_Z$TP4\_P\_Value, get\_significance)

write.csv(Timecourse\_Wilcox\_Sex\_Z , file = "Timecourse\_Wilcox\_Sex\_Z\_Results.csv", row.names = FALSE)

#### Healthy vs MILD vs MODERATE (Mean ± SEM)\* -------------------------------------

folder <- "Mean\_SEM\_capZ\_Category\_Plots"

createFolder(folder) # Ensure the folder exists

results <- data.frame(Category = character(), Gene = character(), Subpopulation = character(),

P\_Value = numeric(), TP1\_P\_Value = numeric(), TP2\_P\_Value = numeric(),

TP3\_P\_Value = numeric(), TP4\_P\_Value = numeric())

for (i in 1:length(unique(data\_mean\_matched$Gene))) {

gene <- unique(data\_mean\_matched$Gene)[i]

df\_gene <- filter(data\_mean\_matched, Gene == gene)

df\_gene <- df\_gene %>% filter(!is.na(Timepoint), !is.na(mean\_capZ), !is.na(Subpopulation), !is.na(Category))

for (j in 1:length(unique(df\_gene$Subpopulation))) {

sup <- unique(df\_gene$Subpopulation)[j]

df\_subpop <- filter(df\_gene, Subpopulation == sup)

# Remove outliers

df\_subpop\_clean <- df\_subpop[!df\_subpop$mean\_capZ %in% boxplot.stats(df\_subpop$mean\_capZ)$out, ]

if (nrow(df\_subpop\_clean) <= 0) next # Skip if no rows remain

tryCatch({

title\_facet <- paste(gene, "expression in", sup, "monocytes (Zscored dCT)", sep=" ")

# Line plot comparing Male vs Female with mean ± SEM across Timepoints

ggline(subset(df\_subpop\_clean, !is.na(Category)),

x = "Timepoint",

y = "mean\_capZ",

color = "Category",

add = "mean\_se", # Add mean and standard error

size = 1.2) +

labs(y = paste(gene, "expression (Z-scored from CT - CT Sample Median)"),

title = title\_facet) +

scale\_color\_manual(values = c("MINOR" = "#A67C00", "MODERATE" = "#700606")) +

theme\_bw() + # Apply white background

theme(panel.grid.major = element\_line(size = 0.2, linetype = 'solid', color = "gray80"),

panel.grid.minor = element\_blank(),

panel.border = element\_blank(),

axis.line = element\_line(color = "black")) +

stat\_compare\_means(method = "wilcox.test", # or "t.test"

aes(group = Category),

label = "p.signif",

size = 5)

if (plot\_save == "y") {

file\_name\_facet <- paste0(folder, "/", title\_facet, "\_Mean\_SEM.png")

ggsave(filename = file\_name\_facet)

}

# Initialize a list to store p-values for timepoints

p\_values\_list <- list(TP0 = NA, TP1 = NA, TP2 = NA, TP3 = NA, TP4 = NA)

# Extract p-values for each timepoint comparison (pairwise Wilcoxon)

timepoints <- c("TP0", "TP1", "TP2", "TP3", "TP4")

for (tp in timepoints) {

minor\_values <- df\_subpop\_clean$mean\_capZ[df\_subpop\_clean$Timepoint == tp & df\_subpop\_clean$Category == "MINOR"]

moderate\_values <- df\_subpop\_clean$mean\_capZ[df\_subpop\_clean$Timepoint == tp & df\_subpop\_clean$Category == "MODERATE"]

# Perform Wilcoxon test for the comparison of Minor vs Moderate at each timepoint

if (length(minor\_values) > 0 && length(moderate\_values) > 0) {

wilcox\_result <- wilcox.test(minor\_values, moderate\_values)

p\_values\_list[[tp]] <- wilcox\_result$p.value

}

}

# Add results for Male vs Female for all timepoints as a single row

results <- rbind(results, data.frame(WilcoxComparison = "Minor vs Moderate", Gene = gene, Subpopulation = sup,

TP0\_P\_Value = p\_values\_list[["TP0"]],

TP1\_P\_Value = p\_values\_list[["TP1"]],

TP2\_P\_Value = p\_values\_list[["TP2"]],

TP3\_P\_Value = p\_values\_list[["TP3"]],

TP4\_P\_Value = p\_values\_list[["TP4"]]))

}, error = function(e) {

cat("Error processing", gene, ":", e$message, "\n")

})

}

}

Timecourse\_Wilcox\_Category\_Z <- results

Timecourse\_Wilcox\_Category\_Z$TP0\_Significance <- sapply(Timecourse\_Wilcox\_Category\_Z$TP0\_P\_Value, get\_significance)

Timecourse\_Wilcox\_Category\_Z$TP1\_Significance <- sapply(Timecourse\_Wilcox\_Category\_Z$TP1\_P\_Value, get\_significance)

Timecourse\_Wilcox\_Category\_Z$TP2\_Significance <- sapply(Timecourse\_Wilcox\_Category\_Z$TP2\_P\_Value, get\_significance)

Timecourse\_Wilcox\_Category\_Z$TP3\_Significance <- sapply(Timecourse\_Wilcox\_Category\_Z$TP3\_P\_Value, get\_significance)

Timecourse\_Wilcox\_Category\_Z$TP4\_Significance <- sapply(Timecourse\_Wilcox\_Category\_Z$TP4\_P\_Value, get\_significance)

write.csv(Timecourse\_Wilcox\_Category\_Z , file = "Timecourse\_Wilcox\_Category\_Z\_Results.csv", row.names = FALSE)

#### Good vs Bad Recovery (Mean ± SEM)\* -------------------------------------

folder <- "Mean\_SEM\_capZ\_Recovery\_Plots"

plot\_save <- "y"

createFolder(folder) # Ensure the folder exists

results <- data.frame(Recovery = character(), Gene = character(), Subpopulation = character(),

P\_Value = numeric(), TP1\_P\_Value = numeric(), TP2\_P\_Value = numeric(),

TP3\_P\_Value = numeric(), TP4\_P\_Value = numeric())

for (i in 1:length(unique(data\_mean\_matched$Gene))) {

gene <- unique(data\_mean\_matched$Gene)[i]

df\_gene <- filter(data\_mean\_matched, Gene == gene)

df\_gene <- df\_gene %>% filter(!is.na(Timepoint), !is.na(mean\_capZ), !is.na(Subpopulation), !is.na(Recovery))

for (j in 1:length(unique(df\_gene$Subpopulation))) {

sup <- unique(df\_gene$Subpopulation)[j]

df\_subpop <- filter(df\_gene, Subpopulation == sup)

# Remove outliers

df\_subpop\_clean <- df\_subpop[!df\_subpop$mean\_capZ %in% boxplot.stats(df\_subpop$mean\_capZ)$out, ]

if (nrow(df\_subpop\_clean) <= 0) next # Skip if no rows remain

tryCatch({

title\_facet <- paste(gene, "expression in", sup, "monocytes (Zscored dCT)", sep=" ")

# Line plot comparing Male vs Female with mean ± SEM across Timepoints

ggline(subset(df\_subpop\_clean, !is.na(Recovery)),

x = "Timepoint",

y = "mean\_capZ",

color = "Recovery",

add = "mean\_se", # Add mean and standard error

size = 1.2) +

labs(y = paste(gene, "expression (Z-scored from CT - CT Sample Median)"),

title = title\_facet) +

scale\_color\_manual(values = c("Good" = "darkgreen", "Bad" = "#700606")) +

theme\_bw() + # Apply white background

theme(panel.grid.major = element\_line(size = 0.2, linetype = 'solid', color = "gray80"),

panel.grid.minor = element\_blank(),

panel.border = element\_blank(),

axis.line = element\_line(color = "black")) +

stat\_compare\_means(method = "wilcox.test", # or "t.test"

aes(group = Recovery),

label = "p.signif",

size = 5)

if (plot\_save == "y") {

file\_name\_facet <- paste0(folder, "/", title\_facet, "\_Mean\_SEM.png")

ggsave(filename = file\_name\_facet, width = 9, height = 5)

}

# Initialize a list to store p-values for timepoints

p\_values\_list <- list(TP0 = NA, TP1 = NA, TP2 = NA, TP3 = NA, TP4 = NA)

# Extract p-values for each timepoint comparison (pairwise Wilcoxon)

timepoints <- c("TP0", "TP1", "TP2", "TP3", "TP4")

for (tp in timepoints) {

good\_values <- df\_subpop\_clean$mean\_capZ[df\_subpop\_clean$Timepoint == tp & df\_subpop\_clean$Recovery == "Good"]

bad\_values <- df\_subpop\_clean$mean\_capZ[df\_subpop\_clean$Timepoint == tp & df\_subpop\_clean$Recovery == "Bad"]

# Perform Wilcoxon test for the comparison of Good vs Bad Recovery at each timepoint

if (length(good\_values) > 0 && length(bad\_values) > 0) {

wilcox\_result <- wilcox.test(good\_values, bad\_values)

p\_values\_list[[tp]] <- wilcox\_result$p.value

}

}

# Add results for Male vs Female for all timepoints as a single row

results <- rbind(results, data.frame(WilcoxComparison = "Good vs Bad Recovery", Gene = gene, Subpopulation = sup,

TP0\_P\_Value = p\_values\_list[["TP0"]],

TP1\_P\_Value = p\_values\_list[["TP1"]],

TP2\_P\_Value = p\_values\_list[["TP2"]],

TP3\_P\_Value = p\_values\_list[["TP3"]],

TP4\_P\_Value = p\_values\_list[["TP4"]]))

}, error = function(e) {

cat("Error processing", gene, ":", e$message, "\n")

})

}

}

Timecourse\_Wilcox\_Recovery\_Z <- results

Timecourse\_Wilcox\_Recovery\_Z$TP0\_Significance <- sapply(Timecourse\_Wilcox\_Recovery\_Z$TP0\_P\_Value, get\_significance)

Timecourse\_Wilcox\_Recovery\_Z$TP1\_Significance <- sapply(Timecourse\_Wilcox\_Recovery\_Z$TP1\_P\_Value, get\_significance)

Timecourse\_Wilcox\_Recovery\_Z$TP2\_Significance <- sapply(Timecourse\_Wilcox\_Recovery\_Z$TP2\_P\_Value, get\_significance)

Timecourse\_Wilcox\_Recovery\_Z$TP3\_Significance <- sapply(Timecourse\_Wilcox\_Recovery\_Z$TP3\_P\_Value, get\_significance)

Timecourse\_Wilcox\_Recovery\_Z$TP4\_Significance <- sapply(Timecourse\_Wilcox\_Recovery\_Z$TP4\_P\_Value, get\_significance)

write.csv(Timecourse\_Wilcox\_Recovery\_Z , file = "Timecourse\_Wilcox\_Recovery\_Z\_Results.csv", row.names = FALSE)

#### All results -----------------

TP\_Wilcox\_All\_Z <- rbind(Timecourse\_Wilcox\_Recovery\_Z, Timecourse\_Wilcox\_Category\_Z,Timecourse\_Wilcox\_Sex\_Z)

write.csv(TP\_Wilcox\_All\_Z , file = "TP\_comparisons\_Wilcox\_All\_Z\_Results.csv", row.names = FALSE)

TP\_Wilcox\_Allsig\_Z <- TP\_Wilcox\_All\_Z %>% filter(if\_any(4:8, ~ . <= 0.05))

write.csv(TP\_Wilcox\_Allsig\_Z , file = "TP\_comparisons\_Wilcox\_Allsig\_Z\_Results.csv", row.names = FALSE)

# \*Age Regression\* --------------------------------------------------------

#### Pearson Gene vs Age --------------------------------------------------------

createFolder("LinReg\_Results\_capZ")

createFolder("LinReg\_Plots\_Age\_capZ")

# Pearson's Correlation uses linear relationship to correlate the data

# Initialize dataframes to store correlation results

Age\_correlation\_capZ <- data.frame(Subpopulation = character(), Timepoint = character(), Gene = character(), GeneID = numeric(), Gene\_Group = character(), N = numeric(), p.value = numeric(), Estimate = numeric(), Est\_CI\_Lower = numeric(), Est\_CI\_Upper = numeric(), Coefficient = numeric(), Coef\_CI\_Lower = numeric(), Coef\_CI\_Upper = numeric())

Age\_correlation\_Sex\_capZ <- data.frame(Subpopulation = character(), Sex = character(), Timepoint = character(), Gene = character(), GeneID = numeric(), Gene\_Group = character(), N = numeric(), p.value = numeric(), Estimate = numeric(), Est\_CI\_Lower = numeric(), Est\_CI\_Upper = numeric(), Coefficient = numeric(), Coef\_CI\_Lower = numeric(), Coef\_CI\_Upper = numeric())

Age\_correlation\_Category\_capZ <- data.frame(Subpopulation = character(), Category = character(), Timepoint = character(), Gene = character(), GeneID = numeric(), Gene\_Group = character(), N = numeric(), p.value = numeric(), Estimate = numeric(), Est\_CI\_Lower = numeric(), Est\_CI\_Upper = numeric(), Coefficient = numeric(), Coef\_CI\_Lower = numeric(), Coef\_CI\_Upper = numeric())

for (gene in unique(dataFINALmean$Gene)) {

df <- dataFINALmean %>% filter(Gene == gene)

df <- df %>% filter(!is.na(Timepoint), !is.na(mean\_capZ), !is.na(Subpopulation), !is.na(Sex), !is.na(Category))

for (subpop in unique(df$Subpopulation)) {

df\_subpop <- df %>% filter(Subpopulation == subpop)

for (tp in unique(df\_subpop$Timepoint)) {

df\_tp <- df\_subpop %>% filter(Timepoint == tp)

if (nrow(df\_tp) <= 0) next

# Outlier removal

lm\_initial <- lm(mean\_capZ ~ Age, data = df\_tp)

residuals\_values <- residuals(lm\_initial)

outlier\_indices <- which(abs(residuals\_values) > (3 \* sd(residuals\_values)))

# Remove outliers only if any are detected

if (length(outlier\_indices) > 0) {

df\_tp\_filtered <- df\_tp[-outlier\_indices, ]

} else {

df\_tp\_filtered <- df\_tp # No outliers, keep the original dataframe

}

if (nrow(df\_tp\_filtered) < 2) next # Ensure there are enough data points after outlier removal

tryCatch({

# Pearson's correlation

x <- cor.test(df\_tp\_filtered$mean\_capZ, df\_tp\_filtered$Age)

row <- data.frame(

Subpopulation = subpop,

Timepoint = tp,

Gene = gene,

GeneID = df\_tp\_filtered$GeneID[1],

Gene\_Group = df\_tp\_filtered$Gene\_Group[1],

N = nrow(df\_tp\_filtered),

p.value = x$p.value,

Estimate = x$estimate,

Est\_CI\_Lower = x$conf.int[1],

Est\_CI\_Upper = x$conf.int[2],

Coefficient = NA,

Coef\_CI\_Lower = NA,

Coef\_CI\_Upper = NA

)

# Linear regression for coefficient and CI

lm\_result <- lm(mean\_capZ ~ Age, data = df\_tp\_filtered)

coef\_x <- summary(lm\_result)$coefficients[2, 1]

ci <- confint(lm\_result)[2, ]

row$Coefficient <- coef\_x

row$Coef\_CI\_Lower <- ci[1]

row$Coef\_CI\_Upper <- ci[2]

if (row$p.value<=0.05){

titel <- paste(gene, " expression in ", subpop, " monocytes at ", tp," (All Samples)", sep="")

file\_name <- paste("LinReg\_Plots\_Age\_capZ/",titel, ".png", sep = "")

ggplot(data = df\_tp\_filtered, aes(x = Age, y = mean\_capZ)) +

geom\_smooth(method = "glm", color = "black") +

geom\_point(aes(color = Timepoint), size = 2) +

ggtitle(titel) +

xlab("Age") +

ylab(paste(gene, "expression (Z-scored from CT - CT Sample Median)")) +

scale\_color\_manual(values = my\_colors) +

theme(text = element\_text(size=14)) +

theme(

panel.background = element\_rect(fill = "white", color = "black"),

plot.background = element\_rect(fill = "white", color = "black"),

text = element\_text(color = "black"),

panel.grid.major = element\_line(color = "gray", size = 0.2),

panel.grid.minor = element\_blank(),

panel.grid.major.x = element\_blank()

)

ggsave(filename=file\_name)

}

Age\_correlation\_capZ <- rbind(Age\_correlation\_capZ, row)

}, error = function(e) {

cat("Error in Pearson's Correlation for gene", gene, "& Subpopulation with Age", subpop, "- skipping this comparison.\n")

})

for (sex in unique(df\_tp$Sex)) {

df\_sex <- df\_tp %>% filter(Sex == sex)

# Outlier removal

lm\_initial <- lm(mean\_capZ ~ Age, data = df\_sex)

residuals\_values <- residuals(lm\_initial)

outlier\_indices <- which(abs(residuals\_values) > (3 \* sd(residuals\_values)))

# Remove outliers only if any are detected

if (length(outlier\_indices) > 0) {

df\_sex\_filtered <- df\_sex[-outlier\_indices, ]

} else {

df\_sex\_filtered <- df\_sex # No outliers, keep the original dataframe

}

if (nrow(df\_sex\_filtered) < 2) next # Ensure there are enough data points after outlier removal

tryCatch({

# Pearson's correlation for Sex

x <- cor.test(df\_sex\_filtered$mean\_capZ, df\_sex\_filtered$Age)

row <- data.frame(

Subpopulation = subpop,

Sex = sex,

Timepoint = tp,

Gene = gene,

GeneID = df\_sex\_filtered$GeneID[1],

Gene\_Group = df\_sex\_filtered$Gene\_Group[1],

N = nrow(df\_sex\_filtered),

p.value = x$p.value,

Estimate = x$estimate,

Est\_CI\_Lower = x$conf.int[1],

Est\_CI\_Upper = x$conf.int[2],

Coefficient = NA,

Coef\_CI\_Lower = NA,

Coef\_CI\_Upper = NA

)

# Linear regression for coefficient and CI

lm\_result\_sex <- lm(mean\_capZ ~ Age, data = df\_sex\_filtered)

coef\_x <- summary(lm\_result\_sex)$coefficients[2, 1]

ci <- confint(lm\_result\_sex)[2, ]

row$Coefficient <- coef\_x

row$Coef\_CI\_Lower <- ci[1]

row$Coef\_CI\_Upper <- ci[2]

if (row$p.value<=0.05){

titel <- paste(gene, " expression in ", subpop, " monocytes at ", tp," ( ", sex, " Samples)", sep="")

file\_name <- paste("LinReg\_Plots\_Age\_capZ/",titel, ".png", sep = "")

ggplot(data = df\_sex\_filtered, aes(x = Age, y = mean\_capZ)) +

geom\_smooth(method = "glm", color = "black") +

geom\_point(aes(color = Sex), size = 2) +

ggtitle(titel) +

xlab("Age") +

ylab(paste(gene, "expression (Z-scored from CT - CT Sample Median)")) +

scale\_color\_manual(values = c("Male" = "#A67C00", "Female" = "#1D04C2")) +

theme(text = element\_text(size=14)) +

theme(

panel.background = element\_rect(fill = "white", color = "black"),

plot.background = element\_rect(fill = "white", color = "black"),

text = element\_text(color = "black"),

panel.grid.major = element\_line(color = "gray", size = 0.2),

panel.grid.minor = element\_blank(),

panel.grid.major.x = element\_blank()

)

ggsave(filename=file\_name)

}

Age\_correlation\_Sex\_capZ <- rbind(Age\_correlation\_Sex\_capZ, row)

}, error = function(e) {

cat("Error for", gene, tp, sex, "&", subpop, "- skipping this comparison.\n")

})

}

for (cat in unique(df\_tp$Category)) {

df\_cat <- df\_tp %>% filter(Category == cat)

# Outlier removal

lm\_initial <- lm(mean\_capZ ~ Age, data = df\_cat)

residuals\_values <- residuals(lm\_initial)

outlier\_indices <- which(abs(residuals\_values) > (3 \* sd(residuals\_values)))

# Remove outliers only if any are detected

if (length(outlier\_indices) > 0) {

df\_cat\_filtered <- df\_cat[-outlier\_indices, ]

} else {

df\_cat\_filtered <- df\_cat # No outliers, keep the original dataframe

}

if (nrow(df\_cat\_filtered) < 2) next # Ensure there are enough data points after outlier removal

tryCatch({

# Pearson's correlation for Category

x <- cor.test(df\_cat\_filtered$mean\_capZ, df\_cat\_filtered$Age)

row <- data.frame(

Subpopulation = subpop,

Category = cat,

Timepoint = tp,

Gene = gene,

GeneID = df\_cat\_filtered$GeneID[1],

Gene\_Group = df\_cat\_filtered$Gene\_Group[1],

N = nrow(df\_cat\_filtered),

p.value = x$p.value,

Estimate = x$estimate,

Est\_CI\_Lower = x$conf.int[1],

Est\_CI\_Upper = x$conf.int[2],

Coefficient = NA,

Coef\_CI\_Lower = NA,

Coef\_CI\_Upper = NA

)

# Linear regression for coefficient and CI

lm\_result\_cat <- lm(mean\_capZ ~ Age, data = df\_cat\_filtered)

coef\_x <- summary(lm\_result\_cat)$coefficients[2, 1]

ci <- confint(lm\_result\_cat)[2, ]

row$Coefficient <- coef\_x

row$Coef\_CI\_Lower <- ci[1]

row$Coef\_CI\_Upper <- ci[2]

if (row$p.value<=0.05){

titel <- paste(gene, " expression in ", subpop, " monocytes at ", tp," ( ", cat, " Samples)", sep="")

file\_name <- paste("LinReg\_Plots\_Age\_capZ/",titel, ".png", sep = "")

ggplot(data = df\_cat\_filtered, aes(x = Age, y = mean\_capZ)) +

geom\_smooth(method = "glm", color = "black") +

geom\_point(aes(color = Category), size = 2) +

ggtitle(titel) +

xlab("Age") +

ylab(paste(gene, "expression (Z-scored from CT - CT Sample Median)")) +

scale\_color\_manual(values = c("MINOR" = "#A67C00", "MODERATE" = "#700606")) +

theme(text = element\_text(size=14)) +

theme(

panel.background = element\_rect(fill = "white", color = "black"),

plot.background = element\_rect(fill = "white", color = "black"),

text = element\_text(color = "black"),

panel.grid.major = element\_line(color = "gray", size = 0.2),

panel.grid.minor = element\_blank(),

panel.grid.major.x = element\_blank()

)

ggsave(filename=file\_name)

}

Age\_correlation\_Category\_capZ <- rbind(Age\_correlation\_Category\_capZ, row)

}, error = function(e) {

cat("Error for", gene, tp, cat, "&", subpop, "- skipping this comparison.\n")

})

}

}

}

}

Age\_correlation\_capZ$Sex <- "All"

Age\_correlation\_capZ$Category <- "All"

Age\_correlation\_Sex\_capZ$Category <- "All"

Age\_correlation\_Category\_capZ$Sex<- "All"

LinReg\_Age\_capZ <- rbind(Age\_correlation\_capZ, Age\_correlation\_Sex\_capZ, Age\_correlation\_Category\_capZ)

LinReg\_Age\_sigGenes <- LinReg\_Age\_capZ %>% filter(LinReg\_Age\_capZ$p.value <0.05)

write.csv(LinReg\_Age\_sigGenes, "LinReg\_Results\_capZ/LinReg\_Age\_sigGenes.csv", row.names = FALSE)

# \*Neurological Tests Regressions\* --------------------------------------------------------

#### NHISS Regression --------------------------------------------------------

LinReg\_NHISS\_capZ <- perform\_linear\_regression\_correlation(dataFINALmean, "NHISS", "NHISS")

LinReg\_NHISS\_sigGenes <- LinReg\_NHISS\_capZ %>% filter(LinReg\_NHISS\_capZ$p.value <0.05)

write.csv(LinReg\_NHISS\_sigGenes, "LinReg\_Results\_capZ/LinReg\_NHISS\_sigGenes.csv", row.names = FALSE)

#### NHISS\_End Regression\* --------------------------------------------------------

# Correlation with "final" NHISS

dataFINALmean$NHISS\_End <- NA

# Assign values from NHISS where Timepoint is TP4

dataFINALmean$NHISS\_End[dataFINALmean$Timepoint == "TP4"] <- dataFINALmean$NHISS[dataFINALmean$Timepoint == "TP4"]

dataFINALmean$NHISS\_End[dataFINALmean$Timepoint == "TP3"] <- dataFINALmean$NHISS[dataFINALmean$Timepoint == "TP4"]

dataFINALmean$NHISS\_End[dataFINALmean$Timepoint == "TP2"] <- dataFINALmean$NHISS[dataFINALmean$Timepoint == "TP4"]

dataFINALmean$NHISS\_End[dataFINALmean$Timepoint == "TP1"] <- dataFINALmean$NHISS[dataFINALmean$Timepoint == "TP4"]

LinReg\_NHISS\_End\_capZ <- perform\_linear\_regression\_correlation(dataFINALmean, "NHISS\_End", "NHISS\_End")

#Focus only on TP1 & TP2

LinReg\_NHISS\_End\_capZ <-LinReg\_NHISS\_End\_capZ %>% filter(LinReg\_NHISS\_End\_capZ$Timepoint == c("TP1", "TP2"))

LinReg\_NHISS\_End\_sigGenes <- LinReg\_NHISS\_End\_capZ %>% filter(LinReg\_NHISS\_End\_capZ$p.value <0.05)

write.csv(LinReg\_NHISS\_End\_sigGenes, "LinReg\_Results\_capZ/LinReg\_NHISS\_End\_sigGenes.csv", row.names = FALSE)

#### NHISS\_Diff Regression\* --------------------------------------------------------

# Create a new column NHISS\_Diff initialized to NA

dataFINALmean$NHISS\_Diff <- NA

# Calculate the difference for each SampleID

dataFINALmean <- dataFINALmean %>%

group\_by(SampleID, Subpopulation, Gene) %>%

mutate(

NHISS\_Diff = if (all(c("TP1", "TP4") %in% Timepoint)) {

NHISS[Timepoint == "TP1"] - NHISS[Timepoint == "TP4"]

} else {

NA

}

) %>%

ungroup()

LinReg\_NHISS\_Diff\_capZ <- perform\_linear\_regression\_correlation(dataFINALmean, "NHISS\_Diff", "NHISS\_Diff")

#Focus only on TP1 & TP2

LinReg\_NHISS\_Diff\_capZ <-LinReg\_NHISS\_Diff\_capZ %>% filter(LinReg\_NHISS\_Diff\_capZ$Timepoint == c("TP1", "TP2"))

LinReg\_NHISS\_Diff\_sigGenes <- LinReg\_NHISS\_Diff\_capZ %>% filter(LinReg\_NHISS\_Diff\_capZ$p.value <0.05)

write.csv(LinReg\_NHISS\_Diff\_sigGenes, "LinReg\_Results\_capZ/LinReg\_NHISS\_Diff\_sigGenes.csv", row.names = FALSE)

#### NHISS\_Ratio Regression\* --------------------------------------------------------

# Create a new column NHISS\_Diff initialized to NA

dataFINALmean$NHISS\_Ratio <- NA

# Calculate the ratio for each SampleID

dataFINALmean <- dataFINALmean %>%

group\_by(SampleID, Subpopulation, Gene) %>%

mutate(

NHISS\_Ratio = if (all(c("TP1", "TP4") %in% Timepoint)) {

(NHISS[Timepoint == "TP1"] - NHISS[Timepoint == "TP4"])/NHISS[Timepoint == "TP1"]

} else {

NA

}

) %>%

ungroup()

LinReg\_NHISS\_Ratio\_capZ <- perform\_linear\_regression\_correlation(dataFINALmean, "NHISS\_Ratio", "NHISS\_Ratio")

#Focus only on TP1 & TP2

LinReg\_NHISS\_Ratio\_capZ <-LinReg\_NHISS\_Ratio\_capZ %>% filter(LinReg\_NHISS\_Ratio\_capZ$Timepoint == c("TP1", "TP2"))

#write.csv(LinReg\_NHISS\_Ratio\_capZ, "LinReg\_Results\_capZ/LinReg\_NHISS\_Ratio\_capZ.csv", row.names = FALSE)

LinReg\_NHISS\_Ratio\_sigGenes <- LinReg\_NHISS\_Ratio\_capZ %>% filter(LinReg\_NHISS\_Ratio\_capZ$p.value <0.05)

write.csv(LinReg\_NHISS\_Ratio\_sigGenes, "LinReg\_Results\_capZ/LinReg\_NHISS\_Ratio\_sigGenes.csv", row.names = FALSE)

# #### mRS Regression --------------------------------------------------------

# LinReg\_mRS\_capZ <- perform\_linear\_regression\_correlation(dataFINALmean, "mRS", "mRS")

#

# LinReg\_mRS\_sigGenes <- LinReg\_mRS\_capZ %>% filter(LinReg\_mRS\_capZ$p.value <0.05)

# write.csv(LinReg\_mRS\_sigGenes, "LinReg\_Results\_capZ/LinReg\_mRS\_sigGenes.csv", row.names = FALSE)

#

# #### Barthel Regression --------------------------------------------------------

# LinReg\_Barthel\_capZ <- perform\_linear\_regression\_correlation(dataFINALmean, "Barthel", "Barthel")

#

# LinReg\_Barthel\_sigGenes <- LinReg\_Barthel\_capZ %>% filter(LinReg\_Barthel\_capZ$p.value <0.05)

# write.csv(LinReg\_Barthel\_sigGenes, "LinReg\_Results\_capZ/LinReg\_Barthel\_sigGenes.csv", row.names = FALSE)

#

# #### MoCA Regression --------------------------------------------------------

# LinReg\_MoCA\_capZ <- perform\_linear\_regression\_correlation(dataFINALmean, "MoCA", "MoCA")

#

# LinReg\_MoCA\_sigGenes <- LinReg\_MoCA\_capZ %>% filter(LinReg\_MoCA\_capZ$p.value <0.05)

# write.csv(LinReg\_MoCA\_sigGenes, "LinReg\_Results\_capZ/LinReg\_MoCA\_sigGenes.csv", row.names = FALSE)

#

# #### HADS\_Anxiety Regression --------------------------------------------------------

# LinReg\_HADS\_Anxiety\_capZ <- perform\_linear\_regression\_correlation(dataFINALmean, "HADS\_Anxiety", "HADS\_Anxiety")

#

# LinReg\_HADS\_Anxiety\_sigGenes <- LinReg\_HADS\_Anxiety\_capZ %>% filter(LinReg\_HADS\_Anxiety\_capZ$p.value <0.05)

# write.csv(LinReg\_HADS\_Anxiety\_sigGenes, "LinReg\_Results\_capZ/LinReg\_HADS\_Anxiety\_sigGenes.csv", row.names = FALSE)

#

# #### HADS\_Depression Regression --------------------------------------------------------

# LinReg\_HADS\_Depression\_capZ <- perform\_linear\_regression\_correlation(dataFINALmean, "HADS\_Depression", "HADS\_Depression")

#

# LinReg\_HADS\_Depression\_sigGenes <- LinReg\_HADS\_Depression\_capZ %>% filter(LinReg\_HADS\_Depression\_capZ$p.value <0.05)

# write.csv(LinReg\_HADS\_Depression\_sigGenes, "LinReg\_Results\_capZ/LinReg\_HADS\_Depression\_sigGenes.csv", row.names = FALSE)

#

# #### SPAN Regression --------------------------

#

# # Create the SPAN variable as per Almekhlafi et. al 2014:

# dataFINALmean <- dataFINALmean %>%

# mutate(SPAN = Age + NHISS)

#

# LinReg\_SPAN\_capZ <- perform\_linear\_regression\_correlation(dataFINALmean, "SPAN", "SPAN")

#

# LinReg\_SPAN\_sigGenes <- LinReg\_SPAN\_capZ %>% filter(LinReg\_SPAN\_capZ$p.value <0.05)

# write.csv(LinReg\_SPAN\_sigGenes, "LinReg\_Results\_capZ/LinReg\_SPAN\_sigGenes.csv", row.names = FALSE)

# \*Dotplot of predictor Genes\* -----------------------------------------------------------------------------

folder <- "Dotplots\_Gene\_Predictiors"

createFolder(folder)

# lets just focus on all!

Predictors <- LinReg\_NHISS\_End\_sigGenes %>% filter(Subgroup == "All")

Predictors <- Predictors %>% filter(Timepoint != "TP0")

Predictors <- Predictors %>% filter(Timepoint != "TP4")

Predictors <- Predictors %>% filter(Timepoint != "TP3")

# Relabel the timepoints: PS = post-stroke

Predictors$Timepoint[grep("TP1",Predictors$Timepoint)] <- "24 hours PS"

Predictors$Timepoint[grep("TP2",Predictors$Timepoint)] <- "3-5 days PS"

# Create a new column for Timepoint and Subpopulation combination

Predictors <- Predictors %>%

mutate(Sample\_Combo2 = paste(Timepoint, Subpopulation, sep = "\_"))

Predictors <- Predictors %>%

mutate(Sample\_Combo = paste(Subpopulation, Timepoint, sep = "\_"))

# Create a combined column for Gene and GeneID for better sorting

Predictors <- Predictors %>%

mutate(Gene\_Combined = paste(GeneID, Gene, sep = "\_")) # Combine GeneID and Gene

# Create a linear gradient function for the original plots

linear\_gradient <- function() {

# Define the colors for the gradient

colors <- c("blue", "white", "red")

# Set the breakpoints for the gradient (-1.2 to 1.2)

scale\_color\_gradientn(colors = colors,

limits = c(-0.79, 0.79), # Set the limits from -1.2 to 1.2

guide = "colorbar",

na.value = "grey50") # Color for NA values

}

# Ensure Sample\_Combo is ordered alphabetically

Predictors <- Predictors %>%

mutate(Sample\_Combo = factor(Sample\_Combo, levels = sort(unique(Sample\_Combo))))

ggplot(Predictors, aes(x = Gene\_Combined, y = Sample\_Combo)) +

geom\_point(aes(size = -log10(p.value), color = Estimate)) + # Size uses -log10(p.value) for better scaling

scale\_size(range = c(1, 7), name = "p-value\n(-log10)") + # Customize dot size range

linear\_gradient() + # Apply the custom color gradient

theme\_minimal() +

labs(

title = "Dot Plot of Predictors Data",

color = "Estimate",

size = "Significance"

) +

theme(

axis.text.x = element\_text(angle = 90, hjust = 1),

axis.text.y = element\_text(angle = 0, vjust = 1),

plot.background = element\_rect(fill = "white"),

panel.background = element\_rect(fill = "white"),

panel.border = element\_blank(),

panel.grid.major = element\_line(color = "lightgrey", size = 0.25),

panel.grid.minor = element\_line(color = "white", size = 0.25)

) +

scale\_y\_discrete(limits = rev(levels(Predictors$Sample\_Combo))) # Reverse y-axis order

# Save the plot

ggsave(filename = "Dotplots\_Gene\_Predictiors/Dotplot\_Predictors.png", width = 10, height = 8)

# \*Heatmaps\* ----------------------------------

new\_folder <- "Pearsson\_Heatmaps"

createFolder(new\_folder)

# Create a linear gradient function for the original plots

linear\_gradient <- function() {

# Define the colors for the gradient

colors <- c("blue", "white", "red")

# Set the breakpoints for the gradient (-1.2 to 1.2)

scale\_color\_gradientn(colors = colors,

limits = c(-0.79, 0.79), # Set the limits from -1.2 to 1.2

guide = "colorbar",

na.value = "grey50") # Color for NA values

}

Predictors\_Ratio <- LinReg\_NHISS\_Ratio\_capZ

# Relabel the timepoints:

Predictors\_Ratio$Timepoint[grep("TP1",Predictors\_Ratio$Timepoint)] <- "24 hours"

Predictors\_Ratio$Timepoint[grep("TP2",Predictors\_Ratio$Timepoint)] <- "3-5 days"

#Create a new column for Timepoint and Subpopulation combination

Predictors\_Ratio <- Predictors\_Ratio %>%

mutate(Sample\_Combo2 = paste(Timepoint, Subpopulation, sep = "\_")) %>%

mutate(Sample\_Combo = paste(Subpopulation, Timepoint, sep = "\_")) %>%

mutate(Gene\_Combined = paste(GeneID, Gene, sep = "\_"))

# Ensure Sample\_Combo is ordered alphabetically

Predictors\_Ratio <- Predictors\_Ratio %>%

mutate(Sample\_Combo = factor(Sample\_Combo, levels = sort(unique(Sample\_Combo)))) %>%

mutate(Sample\_Combo2 = factor(Sample\_Combo2, levels = sort(unique(Sample\_Combo2)))) %>%

mutate(Gene\_Combined = paste(GeneID, Gene, sep = "\_"))

Predictors\_Ratio$Significance <- ifelse(Predictors\_Ratio $p.value < 0.05, "\*", "")

#### Heatmap - all genes

name\_plot <- "Correlation of acute (24 hours & 3-5 days PS) gene expr. with relative change in NHISS (24h vs 3 months PS)"

ggplot(Predictors\_Ratio , aes(Gene\_Combined, Sample\_Combo2, fill= Estimate)) +

geom\_tile(color = "white") + # Add white grid lines

scale\_fill\_gradient2(low = "blue", mid = "white", high = "red",

midpoint = 0, limits = c(-0.7, 0.8)) +

geom\_text(aes(label = Significance), color = "white", size = 10,

hjust = 0.5, vjust = 0.5) + # Center the text in the squares

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 90, hjust = 1)) +

labs(title = name\_plot, y = "Timepoint & Suptype", x = "Gene", fill = "Estimate")+

scale\_y\_discrete(limits = rev(levels(Predictors\_Ratio $Sample\_Combo2)))

ggsave(filename = paste(new\_folder, "/", name\_plot, ".png", sep = ""),

plot = last\_plot(), width = 15, height = 4.5, dpi = 300)

#### NHISS\_Ratio - sig. Genes: -----------------

Predictors\_Ratio <- merge(Predictors\_Ratio , Metadata\_GeneID, by = "Gene")

Predictors\_Ratio <- Predictors\_Ratio %>%

mutate(Gene\_Combined2 = paste(GeneID\_Ratio, Gene, sep = "\_")) # Combine GeneID and Gene

# Filter rows where p-value is <= 0.5

significant\_genes <- Predictors\_Ratio %>% filter(Predictors\_Ratio$p.value <0.05)

significant\_genes <- unique(significant\_genes$Gene)

# adjust data frame accordingly

Predictors\_Ratio <- Predictors\_Ratio %>%

filter(Gene %in% significant\_genes)

#### Heatmap - sig. Genes

name\_plot <- "Correlation of acute (24 hours & 3-5 days PS) gene expr. with relative change in NHISS (24h vs 3 months PS) - only sig. Genes"

ggplot(Predictors\_Ratio , aes(Gene\_Combined2, Sample\_Combo2, fill= Estimate)) +

geom\_tile(color = "white") + # Add white grid lines

scale\_fill\_gradient2(low = "blue", mid = "white", high = "red",

midpoint = 0, limits = c(-0.7, 0.8)) +

geom\_text(aes(label = Significance), color = "white", size = 12,

hjust = 0.5, vjust = 0.5) + # Center the text in the squares

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 90, hjust = 1)) +

labs(title = name\_plot, y = "Timepoint & Suptype", x = "Gene", fill = "Estimate")+

scale\_y\_discrete(limits = rev(levels(Predictors\_Ratio $Sample\_Combo2)))

ggsave(filename = paste(new\_folder, "/", name\_plot, ".png", sep = ""),

plot = last\_plot(), width = 9, height = 5, dpi = 300)

#### NHISS\_Ratio - TP1/TP2 sig. Genes: -----------------

Predictors\_Ratio\_TP1 <- Predictors\_Ratio %>% filter(Timepoint == "24 hours")

Predictors\_Ratio\_TP2 <- Predictors\_Ratio %>% filter(Timepoint == "3-5 days")

# Filter rows where p-value is <= 0.5

significant\_genes\_TP1 <- Predictors\_Ratio\_TP1 %>% filter(Predictors\_Ratio\_TP1$p.value <0.05)

significant\_genes\_TP1 <- unique(significant\_genes\_TP1$Gene)

significant\_genes\_TP2 <- Predictors\_Ratio\_TP2 %>% filter(Predictors\_Ratio\_TP2$p.value <0.05)

significant\_genes\_TP2 <- unique(significant\_genes\_TP2$Gene)

# adjust data frame accordingly

Predictors\_Ratio\_TP1 <- Predictors\_Ratio\_TP1 %>% filter(Gene %in% significant\_genes\_TP1)

Predictors\_Ratio\_TP2 <- Predictors\_Ratio\_TP2 %>% filter(Gene %in% significant\_genes\_TP2)

#### Heatmap - TP1 sig. Genes

name\_plot <- "Correlation of acute (24 hours) gene expr. with relative change in NHISS (24h vs 3 months PS) - only sig. Genes"

ggplot(Predictors\_Ratio\_TP1, aes(Gene\_Combined2, Sample\_Combo2, fill= Estimate)) +

geom\_tile(color = "white") + # Add white grid lines

scale\_fill\_gradient2(low = "blue", mid = "white", high = "red",

midpoint = 0, limits = c(-0.65, 0.71)) +

geom\_text(aes(label = Significance), color = "white", size = 12,

hjust = 0.5, vjust = 0.8) + # Center the text in the squares

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 90, hjust = 1)) +

labs(title = name\_plot, y = "Suptype", x = "Gene", fill = "Estimate")+

scale\_y\_discrete(limits = rev(levels(Predictors\_Ratio\_TP1$Sample\_Combo2)))

ggsave(filename = paste(new\_folder, "/", name\_plot, ".png", sep = ""),

plot = last\_plot(), width = 6, height = 5, dpi = 300)

#### Heatmap - TP2 sig. Genes

name\_plot <- "Correlation of sub-acute (3-5 days) gene expr. with relative change in NHISS (24h vs 3 months PS) - only sig. Genes"

ggplot(Predictors\_Ratio\_TP2, aes(Gene\_Combined2, Sample\_Combo2, fill= Estimate)) +

geom\_tile(color = "white") + # Add white grid lines

scale\_fill\_gradient2(low = "blue", mid = "white", high = "red",

midpoint = 0, limits = c(-0.68, 0.8)) +

geom\_text(aes(label = Significance), color = "white", size = 12,

hjust = 0.5, vjust = 0.8) + # Center the text in the squares

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 90, hjust = 1)) +

labs(title = name\_plot, y = "Suptype", x = "Gene", fill = "Estimate")+

scale\_y\_discrete(limits = rev(levels(Predictors\_Ratio\_TP2$Sample\_Combo2)))

ggsave(filename = paste(new\_folder, "/", name\_plot, ".png", sep = ""),

plot = last\_plot(), width = 6, height = 5, dpi = 300)

#### NHISS (Estimate) ---------------

LinReg\_NHISS\_capZ <- LinReg\_NHISS\_capZ %>%

mutate(Timepoint = factor(Timepoint, levels = sort(unique(Timepoint))))

LinReg\_NHISS\_capZ <- LinReg\_NHISS\_capZ %>%

mutate(Gene\_Ordered = paste(GeneID, Gene, sep = "\_"))

LinReg\_NHISS\_capZ <- LinReg\_NHISS\_capZ %>%

mutate(Groups = paste(Subpopulation, Timepoint, sep = "\_"))

# Estimate: - 0.99 to 0.92

name\_plot <- paste("Heatmap of NHISS Correlation Estimates")

# Add significance levels

LinReg\_NHISS\_capZ$Significance <- ifelse(LinReg\_NHISS\_capZ$p.value < 0.05, "\*", "")

# Create the heatmap with significance levels

ggplot(LinReg\_NHISS\_capZ, aes(x = Gene\_Ordered, y = Groups, fill = Estimate)) +

geom\_tile() +

scale\_fill\_gradient2(low = "blue", mid = "white", high = "purple",

midpoint = 0, limits = c(-0.72, 0.78)) +

geom\_text(aes(label = Significance), color = "black", size = 3) + # Add significance levels with white text

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 90, hjust = 1)) +

labs(title = name\_plot, x = "Gene", y = "Timepoint & Subpopulation", fill = "Estimate")

ggsave(filename = paste(new\_folder, "/", name\_plot, ".png", sep = ""),

plot = last\_plot(), width = 10, height = 5, dpi = 300)

name\_plot <- paste(name\_plot," (transposed)")

ggplot(LinReg\_NHISS\_capZ, aes(Groups, Gene\_Ordered, fill= Estimate)) +

geom\_tile() +

scale\_fill\_gradient2(low = "yellow", mid = "white", high = "purple",

midpoint = 0, limits = c(-0.72, 0.78)) +

geom\_text(aes(label = Significance), color = "black", size = 3) +

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 90, hjust = 1)) +

labs(title = name\_plot, x = "Timepoint & Subpopulation", y = "Timepoint", fill = "Estimate")

ggsave(filename = paste(new\_folder, "/", name\_plot, ".png", sep = ""),

plot = last\_plot(), width = 5, height = 10, dpi = 300)

#### NHISS Split by Subpopulation (Estimate) ---------------

for (sup in unique(LinReg\_NHISS\_capZ$Subpopulation)) {

df <- filter(LinReg\_NHISS\_capZ, Subpopulation == sup)

sigGenes <- filter(LinReg\_NHISS\_capZ, p.value <= 0.05)

sigGenes <- unique(sigGenes$Gene)

df <- df %>% filter(Gene %in% sigGenes)

df <- df %>% filter(!Gene %in% c("ACTB", "B2M", "GAPDH"))

name\_plot <- paste("Heatmap of NHISS Correlation Estimates - ", sup, " Monocytes")

ggplot(df, aes(Gene\_Ordered, Timepoint, fill= Estimate)) +

geom\_tile() +

scale\_fill\_gradient2(low = "yellow", mid = "white", high = "purple",

midpoint = 0, limits = c(-0.72, 0.78)) +

geom\_text(aes(label = Significance), color = "black", size = 6) +

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 90, hjust = 1)) +

labs(title = name\_plot, x = "Gene", y = "Timepoint", fill = "Coefficient")+

scale\_y\_discrete(labels = timepoint\_labels)+# Apply custom y-axis labels

scale\_x\_discrete(labels = function(x) str\_sub(x, start = 4)) + # Remove first three characters

theme(

axis.text.x = element\_text(angle = 90, hjust = 1, size = 14), # Adjust font size

axis.text.y = element\_text(size = 14), # Adjust font size

axis.title.x = element\_text(size = 16),

axis.title.y = element\_text(size = 16),

plot.title = element\_text(size = 18, face = "bold")

)

ggsave(filename = paste(new\_folder, "/", name\_plot, ".png", sep = ""),

plot = last\_plot(), width = 10, height = 5, dpi = 300)

}

for (sup in unique(LinReg\_NHISS\_capZ$Subpopulation)) {

df <- filter(LinReg\_NHISS\_capZ, Subpopulation == sup)

# sigGenes <- filter(LinReg\_NHISS\_capZ, p.value <= 0.05) if you want to compare all of them

sigGenes <- filter(df, p.value <= 0.05)

sigGenes <- unique(sigGenes$Gene)

df <- df %>% filter(Gene %in% sigGenes)

df <- df %>% filter(!Gene %in% c("ACTB", "B2M", "GAPDH"))

name\_plot <- paste("Heatmap of NHISS Correlation Estimates - ", sup, " Monocytes (only sigGenes)")

ggplot(df, aes(Gene\_Ordered, Timepoint, fill= Estimate)) +

geom\_tile() +

scale\_fill\_gradient2(low = "yellow", mid = "white", high = "purple",

midpoint = 0, limits = c(-0.72, 0.78)) +

geom\_text(aes(label = Significance), color = "black", size = 6) +

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 90, hjust = 1)) +

labs(title = name\_plot, x = "Gene", y = "Timepoint", fill = "Coefficient")+

scale\_y\_discrete(labels = timepoint\_labels)+# Apply custom y-axis labels

scale\_x\_discrete(labels = function(x) str\_sub(x, start = 4)) + # Remove first three characters

theme(

axis.text.x = element\_text(angle = 90, hjust = 1, size = 14), # Adjust font size

axis.text.y = element\_text(size = 14), # Adjust font size

axis.title.x = element\_text(size = 16),

axis.title.y = element\_text(size = 16),

plot.title = element\_text(size = 18, face = "bold")

)

ggsave(filename = paste(new\_folder, "/", name\_plot, ".png", sep = ""),

plot = last\_plot(), width = 10, height = 5, dpi = 300)

}

#### NHISS Split by Timepoint (Estimate) ---------------

for (tp in unique(LinReg\_NHISS\_capZ$Timepoint)) {

df <- filter(LinReg\_NHISS\_capZ, Timepoint == tp)

sigGenes <- filter(LinReg\_NHISS\_capZ, p.value <= 0.05)

sigGenes <- unique(sigGenes$Gene)

df <- df %>% filter(Gene %in% sigGenes)

df <- df %>% filter(!Gene %in% c("ACTB", "B2M", "GAPDH"))

name\_plot <- paste("Heatmap of NHISS Correlation Estimates - ", tp)

ggplot(df, aes(Gene\_Ordered, Subpopulation, fill= Estimate)) +

geom\_tile() +

scale\_fill\_gradient2(low = "yellow", mid = "white", high = "purple",

midpoint = 0, limits = c(-0.72, 0.78)) +

geom\_text(aes(label = Significance), color = "black", size = 6) +

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 90, hjust = 1)) +

labs(title = name\_plot, x = "Gene", y = "Monocyte Suptype", fill = "Coefficient")+

scale\_x\_discrete(labels = function(x) str\_sub(x, start = 4)) + # Remove first three characters

theme(

axis.text.x = element\_text(angle = 90, hjust = 1, size = 14), # Adjust font size

axis.text.y = element\_text(size = 14), # Adjust font size

axis.title.x = element\_text(size = 16),

axis.title.y = element\_text(size = 16),

plot.title = element\_text(size = 18, face = "bold")

)

ggsave(filename = paste(new\_folder, "/", name\_plot, ".png", sep = ""),

plot = last\_plot(), width = 10, height = 5, dpi = 300)

}

for (tp in unique(LinReg\_NHISS\_capZ$Timepoint)) {

df <- filter(LinReg\_NHISS\_capZ, Timepoint == tp)

# sigGenes <- filter(LinReg\_NHISS\_capZ, p.value <= 0.05) if you want to compare all of them

sigGenes <- filter(df, p.value <= 0.05)

sigGenes <- unique(sigGenes$Gene)

df <- df %>% filter(Gene %in% sigGenes)

df <- df %>% filter(!Gene %in% c("ACTB", "B2M", "GAPDH"))

name\_plot <- paste("Heatmap of NHISS Correlation Estimates - ", tp, " (only sigGenes)")

ggplot(df, aes(Gene\_Ordered, Subpopulation, fill= Estimate)) +

geom\_tile() +

scale\_fill\_gradient2(low = "yellow", mid = "white", high = "purple",

midpoint = 0, limits = c(-0.72, 0.78)) +

geom\_text(aes(label = Significance), color = "black", size = 6) +

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 90, hjust = 1)) +

labs(title = name\_plot, x = "Gene", y = "Monocyte Suptype", fill = "Coefficient")+

scale\_x\_discrete(labels = function(x) str\_sub(x, start = 4)) + # Remove first three characters

theme(

axis.text.x = element\_text(angle = 90, hjust = 1, size = 14), # Adjust font size

axis.text.y = element\_text(size = 14), # Adjust font size

axis.title.x = element\_text(size = 16),

axis.title.y = element\_text(size = 16),

plot.title = element\_text(size = 18, face = "bold")

)

ggsave(filename = paste(new\_folder, "/", name\_plot, ".png", sep = ""),

plot = last\_plot(), width = 10, height = 5, dpi = 300)

}

sigGenes <- filter(LinReg\_NHISS\_capZ, p.value <= 0.05)

unique(sigGenes$Gene)

#### Aging (Estimate) ---------------

Age\_correlation\_capZ <- Age\_correlation\_capZ %>%

mutate(Gene\_Ordered = paste(GeneID, Gene, sep = "\_"))

Age\_correlation\_capZ <- Age\_correlation\_capZ %>%

mutate(Groups = paste(Subpopulation, Timepoint, sep = "\_"))

Age\_correlation\_capZ$Significance <- ifelse(Age\_correlation\_capZ$p.value < 0.001, "\*\*\*",

ifelse(Age\_correlation\_capZ$p.value < 0.01, "\*\*",

ifelse(Age\_correlation\_capZ$p.value < 0.05, "\*", "")))

# Estimate: - 0.7 to 0.79

name\_plot <- "Heatmap of Age Correlation Estimates"

ggplot(Age\_correlation\_capZ, aes(Gene\_Ordered, Groups, fill= Estimate)) +

geom\_tile() +

scale\_fill\_gradient2(low = "blue", mid = "white", high = "red",

midpoint = 0, limits = c(-0.7, 0.804)) +

geom\_text(aes(label = Significance), color = "white", size = 3) +

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 90, hjust = 1)) +

labs(title = name\_plot, x = "Gene", y = "Timepoint & Suptype", fill = "Estimate")

ggsave(filename = paste(new\_folder, "/", name\_plot, ".png", sep = ""),

plot = last\_plot(), width = 10, height = 5, dpi = 300)

name\_plot <- paste(name\_plot," (transposed)")

ggplot(Age\_correlation\_capZ, aes(Groups, Gene\_Ordered, fill= Estimate)) +

geom\_tile() +

scale\_fill\_gradient2(low = "blue", mid = "white", high = "red",

midpoint = 0, limits = c(-0.7, 0.804)) +

geom\_text(aes(label = Significance), color = "white", size = 3) +

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 90, hjust = 1)) +

labs(title = name\_plot, x = "Timepoint & Suptype", y = "Gene", fill = "Estimate")

ggsave(filename = paste(new\_folder, "/", name\_plot, ".png", sep = ""),

plot = last\_plot(), width = 5, height = 10, dpi = 300)

#### Aging Split by Subpopulation (Estimate) ---------------

for (sup in unique(Age\_correlation\_capZ$Subpopulation)) {

df <- filter(Age\_correlation\_capZ, Subpopulation == sup)

sigGenes <- filter(Age\_correlation\_capZ, p.value <= 0.05)

sigGenes <- unique(sigGenes$Gene)

df <- df %>% filter(Gene %in% sigGenes)

df <- df %>% filter(!Gene %in% c("ACTB", "B2M", "GAPDH"))

name\_plot <- paste("Heatmap of Aging Correlation Estimates - ", sup, " Monocytes")

ggplot(df, aes(Gene\_Ordered, Timepoint, fill= Estimate)) +

geom\_tile() +

scale\_fill\_gradient2(low = "blue", mid = "white", high = "red",

midpoint = 0) +

geom\_text(aes(label = Significance), color = "black", size = 6) +

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 90, hjust = 1)) +

labs(title = name\_plot, x = "Gene", y = "Timepoint", fill = "Coefficient")+

scale\_y\_discrete(labels = timepoint\_labels)+ # Apply custom y-axis labels

scale\_x\_discrete(labels = function(x) str\_sub(x, start = 4)) + # Remove first three characters

theme(

axis.text.x = element\_text(angle = 90, hjust = 1, size = 14), # Adjust font size

axis.text.y = element\_text(size = 14), # Adjust font size

axis.title.x = element\_text(size = 16),

axis.title.y = element\_text(size = 16),

plot.title = element\_text(size = 18, face = "bold")

)

ggsave(filename = paste(new\_folder, "/", name\_plot, ".png", sep = ""),

plot = last\_plot(), width = 10, height = 5, dpi = 300)

}

for (sup in unique(Age\_correlation\_capZ$Subpopulation)) {

df <- filter(Age\_correlation\_capZ, Subpopulation == sup)

# sigGenes <- filter(Age\_correlation\_capZ, p.value <= 0.05) #if you want to compare all of them

sigGenes <- filter(df, p.value <= 0.05)

sigGenes <- unique(sigGenes$Gene)

df <- df %>% filter(Gene %in% sigGenes)

df <- df %>% filter(!Gene %in% c("ACTB", "B2M", "GAPDH"))

name\_plot <- paste("Heatmap of Aging Correlation Estimates - ", sup, " Monocytes (only sigGenes)")

ggplot(df, aes(Gene\_Ordered, Timepoint, fill= Estimate)) +

geom\_tile() +

scale\_fill\_gradient2(low = "blue", mid = "white", high = "red",

midpoint = 0) +

geom\_text(aes(label = Significance), color = "black", size = 6) +

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 90, hjust = 1)) +

labs(title = name\_plot, x = "Gene", y = "Timepoint", fill = "Coefficient")+

scale\_y\_discrete(labels = timepoint\_labels)+# Apply custom y-axis labels

scale\_x\_discrete(labels = function(x) str\_sub(x, start = 4)) + # Remove first three characters

theme(

axis.text.x = element\_text(angle = 90, hjust = 1, size = 14), # Adjust font size

axis.text.y = element\_text(size = 14), # Adjust font size

axis.title.x = element\_text(size = 16),

axis.title.y = element\_text(size = 16),

plot.title = element\_text(size = 18, face = "bold")

)

ggsave(filename = paste(new\_folder, "/", name\_plot, ".png", sep = ""),

plot = last\_plot(), width = 10, height = 5, dpi = 300)

}

#### Aging Split by Timepoint (Estimate) ---------------

for (tp in unique(Age\_correlation\_capZ$Timepoint)) {

df <- filter(Age\_correlation\_capZ, Timepoint == tp)

sigGenes <- filter(Age\_correlation\_capZ, p.value <= 0.05)

sigGenes <- unique(sigGenes$Gene)

df <- df %>% filter(Gene %in% sigGenes)

df <- df %>% filter(!Gene %in% c("ACTB", "B2M", "GAPDH"))

name\_plot <- paste("Heatmap of Aging Correlation Estimates - ", tp)

ggplot(df, aes(Gene\_Ordered, Subpopulation, fill= Estimate)) +

geom\_tile() +

scale\_fill\_gradient2(low = "blue", mid = "white", high = "red",

midpoint = 0) +

geom\_text(aes(label = Significance), color = "black", size = 6) +

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 90, hjust = 1)) +

labs(title = name\_plot, x = "Gene", y = "Monocyte Suptype", fill = "Coefficient")+

scale\_x\_discrete(labels = function(x) str\_sub(x, start = 4)) + # Remove first three characters

theme(

axis.text.x = element\_text(angle = 90, hjust = 1, size = 14), # Adjust font size

axis.text.y = element\_text(size = 14), # Adjust font size

axis.title.x = element\_text(size = 16),

axis.title.y = element\_text(size = 16),

plot.title = element\_text(size = 18, face = "bold")

)

ggsave(filename = paste(new\_folder, "/", name\_plot, ".png", sep = ""),

plot = last\_plot(), width = 10, height = 5, dpi = 300)

}

for (tp in unique(Age\_correlation\_capZ$Timepoint)) {

df <- filter(Age\_correlation\_capZ, Timepoint == tp)

# sigGenes <- filter(Age\_correlation\_capZ, p.value <= 0.05) #if you want to compare all of them

sigGenes <- filter(df, p.value <= 0.05)

sigGenes <- unique(sigGenes$Gene)

df <- df %>% filter(Gene %in% sigGenes)

df <- df %>% filter(!Gene %in% c("ACTB", "B2M", "GAPDH"))

name\_plot <- paste("Heatmap of Aging Correlation Estimates - ", tp, " (only sigGenes)")

ggplot(df, aes(Gene\_Ordered, Subpopulation, fill= Estimate)) +

geom\_tile() +

scale\_fill\_gradient2(low = "blue", mid = "white", high = "red",

midpoint = 0) +

geom\_text(aes(label = Significance), color = "black", size = 6) +

theme\_minimal() +

theme(axis.text.x = element\_text(angle = 90, hjust = 1)) +

labs(title = name\_plot, x = "Gene", y = "Monocyte Suptype", fill = "Coefficient")+

scale\_x\_discrete(labels = function(x) str\_sub(x, start = 4)) + # Remove first three characters

theme(

axis.text.x = element\_text(angle = 90, hjust = 1, size = 14), # Adjust font size

axis.text.y = element\_text(size = 14), # Adjust font size

axis.title.x = element\_text(size = 16),

axis.title.y = element\_text(size = 16),

plot.title = element\_text(size = 18, face = "bold")

)

ggsave(filename = paste(new\_folder, "/", name\_plot, ".png", sep = ""),

plot = last\_plot(), width = 10, height = 5, dpi = 300)

}

sigGenes <- filter(Age\_correlation\_capZ, p.value <= 0.05)

unique(sigGenes$Gene)

# \*Sig Regression - Gene Overview\* --------------------------

# with a mean for all Timepoints

Gene\_Overview\_NHISS <- LinReg\_NHISS\_capZ %>%

filter(p.value <= 0.05) %>%

filter(Subgroup == "All") %>%

group\_by(Subpopulation, Gene)%>%

summarise(NHISS\_Correlation = mean(Estimate, na.rm = TRUE))

Gene\_Overview\_NHISS\_End <- LinReg\_NHISS\_End\_capZ %>%

filter(p.value <= 0.05) %>%

filter(Subgroup == "All") %>%

group\_by(Subpopulation, Gene)%>%

summarise(NHISS\_End\_Correlation = mean(Estimate, na.rm = TRUE))

Gene\_Overview\_NHISS\_Diff <- LinReg\_NHISS\_Diff\_capZ %>%

filter(p.value <= 0.05) %>%

filter(Subgroup == "All") %>%

group\_by(Subpopulation, Gene)%>%

summarise(NHISS\_Diff\_Correlation = mean(Estimate, na.rm = TRUE))

Gene\_Overview\_NHISS\_Ratio <- LinReg\_NHISS\_Ratio\_capZ %>%

filter(p.value <= 0.05) %>%

filter(Subgroup == "All") %>%

group\_by(Subpopulation, Gene)%>%

summarise(NHISS\_Ratio\_Correlation = mean(Estimate, na.rm = TRUE))

# Gene\_Overview\_Age <- Age\_correlation\_capZ %>%

# filter(p.value <= 0.05) %>%

# group\_by(Subpopulation, Gene)%>%

# summarise(Age\_Correlation = mean(Estimate, na.rm = TRUE))

# Gene\_Overview\_SPAN <- LinReg\_SPAN\_capZ %>%

# filter(p.value <= 0.05) %>%

# filter(Subgroup == "All") %>%

# group\_by(Subpopulation, Gene)%>%

# summarise(SPAN\_Correlation = mean(Estimate, na.rm = TRUE))

#

# Gene\_Overview\_mRS <- LinReg\_mRS\_capZ %>%

# filter(p.value <= 0.05) %>%

# filter(Subgroup == "All") %>%

# group\_by(Subpopulation, Gene)%>%

# summarise(mRS\_Correlation = mean(Estimate, na.rm = TRUE))

#

# Gene\_Overview\_Barthel <- LinReg\_Barthel\_capZ %>%

# filter(p.value <= 0.05) %>%

# filter(Subgroup == "All") %>%

# group\_by(Subpopulation, Gene)%>%

# summarise(Barthel\_Correlation = mean(Estimate, na.rm = TRUE))

#

# Gene\_Overview\_MoCA <- LinReg\_MoCA\_capZ %>%

# filter(p.value <= 0.05) %>%

# filter(Subgroup == "All") %>%

# group\_by(Subpopulation, Gene)%>%

# summarise(MoCA\_Correlation = mean(Estimate, na.rm = TRUE))

#

# Gene\_Overview\_HADS\_Anxiety <- LinReg\_HADS\_Anxiety\_capZ %>%

# filter(p.value <= 0.05) %>%

# filter(Subgroup == "All") %>%

# group\_by(Subpopulation, Gene)%>%

# summarise(HADS\_Anxiety\_Correlation = mean(Estimate, na.rm = TRUE))

#

# Gene\_Overview\_HADS\_Depression <- LinReg\_HADS\_Depression\_capZ %>%

# filter(p.value <= 0.05) %>%

# filter(Subgroup == "All") %>%

# group\_by(Subpopulation, Gene)%>%

# summarise(HADS\_Depression\_Correlation = mean(Estimate, na.rm = TRUE))

Gene\_Overview\_woTP <- merge(Gene\_Overview\_NHISS,

Gene\_Overview\_NHISS\_End,

by = c("Gene", "Subpopulation"),

all = TRUE)

Gene\_Overview\_woTP <- merge(Gene\_Overview\_woTP,

Gene\_Overview\_NHISS\_Diff,

by = c("Gene", "Subpopulation"),

all = TRUE)

Gene\_Overview\_woTP <- merge(Gene\_Overview\_woTP,

Gene\_Overview\_NHISS\_Ratio,

by = c("Gene", "Subpopulation"),

all = TRUE)

# Gene\_Overview\_woTP <- merge(Gene\_Overview\_woTP,

# Gene\_Overview\_mRS,

# by = c("Gene", "Subpopulation"),

# all = TRUE)

# Gene\_Overview\_woTP <- merge(Gene\_Overview\_woTP,

# Gene\_Overview\_Barthel,

# by = c("Gene", "Subpopulation"),

# all = TRUE)

# Gene\_Overview\_woTP <- merge(Gene\_Overview\_woTP,

# Gene\_Overview\_MoCA,

# by = c("Gene", "Subpopulation"),

# all = TRUE)

# Gene\_Overview\_woTP <- merge(Gene\_Overview\_woTP,

# Gene\_Overview\_HADS\_Anxiety,

# by = c("Gene", "Subpopulation"),

# all = TRUE)

# Gene\_Overview\_woTP <- merge(Gene\_Overview\_woTP,

# Gene\_Overview\_HADS\_Depression,

# by = c("Gene", "Subpopulation"),

# all = TRUE)

# # Aging at the end!

# Gene\_Overview\_woTP <- merge(Gene\_Overview\_woTP,

# Gene\_Overview\_Age,

# by = c("Gene", "Subpopulation"),

# all = TRUE)

# Gene\_Overview\_woTP <- merge(Gene\_Overview\_woTP,

# Gene\_Overview\_SPAN,

# by = c("Gene", "Subpopulation"),

# all = TRUE)

# table including the details for the timepoints

Gene\_Overview\_NHISS <- LinReg\_NHISS\_capZ %>%

filter(p.value <= 0.05) %>%

filter(Subgroup == "All") %>%

group\_by(Subpopulation, Gene, Timepoint) %>%

summarise(NHISS\_Correlation = Estimate)

Gene\_Overview\_NHISS\_End <- LinReg\_NHISS\_End\_capZ %>%

filter(p.value <= 0.05) %>%

filter(Subgroup == "All") %>%

group\_by(Subpopulation, Gene, Timepoint) %>%

summarise(NHISS\_End\_Correlation = Estimate)

Gene\_Overview\_NHISS\_Diff <- LinReg\_NHISS\_Diff\_capZ %>%

filter(p.value <= 0.05) %>%

filter(Subgroup == "All") %>%

group\_by(Subpopulation, Gene, Timepoint) %>%

summarise(NHISS\_Diff\_Correlation = Estimate)

Gene\_Overview\_NHISS\_Ratio <- LinReg\_NHISS\_Ratio\_capZ %>%

filter(p.value <= 0.05) %>%

filter(Subgroup == "All") %>%

group\_by(Subpopulation, Gene, Timepoint) %>%

summarise(NHISS\_Ratio\_Correlation = Estimate)

# Gene\_Overview\_Age <- Age\_correlation\_capZ %>%

# filter(p.value <= 0.05) %>%

# group\_by(Subpopulation, Gene, Timepoint) %>%

# summarise(Age\_Correlation = Estimate)

# Gene\_Overview\_SPAN <- LinReg\_SPAN\_capZ %>%

# filter(p.value <= 0.05) %>%

# filter(Subgroup == "All") %>%

# group\_by(Subpopulation, Gene, Timepoint) %>%

# summarise(SPAN\_Correlation = Estimate)

#

# Gene\_Overview\_mRS <- LinReg\_mRS\_capZ %>%

# filter(p.value <= 0.05) %>%

# filter(Subgroup == "All") %>%

# group\_by(Subpopulation, Gene, Timepoint) %>%

# summarise(mRS\_Correlation = Estimate)

# Gene\_Overview\_Barthel <- LinReg\_Barthel\_capZ %>%

# filter(p.value <= 0.05) %>%

# filter(Subgroup == "All") %>%

# group\_by(Subpopulation, Gene, Timepoint) %>%

# summarise(Barthel\_Correlation = Estimate)

# Gene\_Overview\_MoCA <- LinReg\_MoCA\_capZ %>%

# filter(p.value <= 0.05) %>%

# filter(Subgroup == "All") %>%

# group\_by(Subpopulation, Gene, Timepoint) %>%

# summarise(MoCA\_Correlation = Estimate)

# Gene\_Overview\_HADS\_Anxiety <- LinReg\_HADS\_Anxiety\_capZ %>%

# filter(p.value <= 0.05) %>%

# filter(Subgroup == "All") %>%

# group\_by(Subpopulation, Gene, Timepoint) %>%

# summarise(HADS\_Anxiety\_Correlation = Estimate)

# Gene\_Overview\_HADS\_Depression <- LinReg\_HADS\_Depression\_capZ %>%

# filter(p.value <= 0.05) %>%

# filter(Subgroup == "All") %>%

# group\_by(Subpopulation, Gene, Timepoint) %>%

# summarise(HADS\_Depression\_Correlation = Estimate)

Gene\_Overview <- merge(Gene\_Overview\_NHISS,

Gene\_Overview\_NHISS\_End,

by = c("Gene", "Subpopulation", "Timepoint"),

all = TRUE)

Gene\_Overview <- merge(Gene\_Overview,

Gene\_Overview\_NHISS\_Diff,

by = c("Gene", "Subpopulation", "Timepoint"),

all = TRUE)

Gene\_Overview <- merge(Gene\_Overview,

Gene\_Overview\_NHISS\_Ratio,

by = c("Gene", "Subpopulation", "Timepoint"),

all = TRUE)

# Gene\_Overview <- merge(Gene\_Overview,

# Gene\_Overview\_mRS,

# by = c("Gene", "Subpopulation", "Timepoint"),

# all = TRUE)

# Gene\_Overview <- merge(Gene\_Overview,

# Gene\_Overview\_Barthel,

# by = c("Gene", "Subpopulation", "Timepoint"),

# all = TRUE)

# Gene\_Overview <- merge(Gene\_Overview,

# Gene\_Overview\_MoCA,

# by = c("Gene", "Subpopulation", "Timepoint"),

# all = TRUE)

# Gene\_Overview <- merge(Gene\_Overview,

# Gene\_Overview\_HADS\_Anxiety,

# by = c("Gene", "Subpopulation", "Timepoint"),

# all = TRUE)

# Gene\_Overview <- merge(Gene\_Overview,

# Gene\_Overview\_HADS\_Depression,

# by = c("Gene", "Subpopulation", "Timepoint"),

# all = TRUE)

# # put aging at the end

# Gene\_Overview <- merge(Gene\_Overview,

# Gene\_Overview\_Age,

# by = c("Gene", "Subpopulation", "Timepoint"),

# all = TRUE)

# Gene\_Overview <- merge(Gene\_Overview,

# Gene\_Overview\_SPAN,

# by = c("Gene", "Subpopulation", "Timepoint"),

# all = TRUE)

# Replace the Estimated with the arrows!

Gene\_Overview\_woTP\_arrows <- Gene\_Overview\_woTP %>%

mutate(across(c(NHISS\_Correlation, NHISS\_End\_Correlation, NHISS\_Diff\_Correlation, NHISS\_Ratio\_Correlation,

#Age\_Correlation, SPAN\_Correlation, mRS\_Correlation, Barthel\_Correlation, MoCA\_Correlation, HADS\_Anxiety\_Correlation, HADS\_Depression\_Correlation,

),

~ ifelse(is.na(.), "-", ifelse(. > 0, "↑", "↓"))))

Gene\_Overview\_arrows <- Gene\_Overview %>%

mutate(across(c(NHISS\_Correlation, NHISS\_End\_Correlation, NHISS\_Diff\_Correlation, NHISS\_Ratio\_Correlation,

#Age\_Correlation, SPAN\_Correlation, mRS\_Correlation, Barthel\_Correlation, MoCA\_Correlation, HADS\_Anxiety\_Correlation, HADS\_Depression\_Correlation,

),

~ ifelse(is.na(.), "-", ifelse(. > 0, "↑", "↓"))))

# Save the files:

file\_name\_s <- "LinReg\_Results\_capZ/Gene\_Overview\_woTP.csv"

write.csv(Gene\_Overview\_woTP, file\_name\_s, row.names = FALSE)

file\_name\_s <- "LinReg\_Results\_capZ/Gene\_Overview.csv"

write.csv(Gene\_Overview, file\_name\_s, row.names = FALSE)

file\_name\_s\_arrows <- "LinReg\_Results\_capZ/Gene\_Overview\_woTP\_arrows.csv"

write.csv(Gene\_Overview\_woTP\_arrows, file\_name\_s\_arrows, row.names = FALSE)

file\_name\_s\_arrows <- "LinReg\_Results\_capZ/Gene\_Overview\_arrows.csv"

write.csv(Gene\_Overview\_arrows, file\_name\_s\_arrows, row.names = FALSE)

# \*Demographics of Ctr & Patients\* -----------------------------------------------------------------------------

# for FACS

metadataP\_FACS <- metadataP %>% filter(!SampleID %in% Unmatched\_TP0\_FACS)

Demographics\_FACS <- demographics\_N\_Age\_Sex(metadataP\_FACS)

write.csv(Demographics\_FACS, file = "Demographics\_FACS.csv", row.names = FALSE)

# for Gene expr.

metadataP\_Gene <- metadataP %>% filter(SampleID %in% unique(data\_mean\_matched$SampleID))

Demographics\_Gene <- demographics\_N\_Age\_Sex(metadataP\_Gene)

write.csv(Demographics\_Gene, file = "Demographics\_Gene.csv", row.names = FALSE)

#### NHISS Recovery----------------------------------------------------------------------------------------------------

NHISS\_Recovery <- merge(Metadata\_NeuroTest, metadataP, by = "SampleID", all.x = TRUE)

# Remove Patient 15

NHISS\_Recovery <- NHISS\_Recovery %>% filter(!(SampleID == "Patient15"))

NHISS\_Recovery$Timepoint <- factor(NHISS\_Recovery$Timepoint, levels = c("TP0", "TP1", "TP2", "TP3", "TP4", "TP5"))

ggline(NHISS\_Recovery,

x = "Timepoint",

y = "NHISS",

color = "Recovery",

add = "mean\_se", # Add mean and standard error

size = 1.2) +

labs(title = paste("Mean ± SEM of NHISS by Recovery across Timepoints")) +

scale\_color\_manual(values = c("Good" = "darkgreen", "Bad" = "#700606")) +

theme\_bw() +

scale\_x\_discrete(labels = c(

"TP0" = "Control",

"TP1" = "24 hours",

"TP2" = "3-5 days",

"TP3" = "1 month",

"TP4" = "3 months",

"TP5" = ">1 year"

)) +

theme(

panel.grid.major = element\_line(size = 0.2, linetype = 'solid', color = "gray80"),

panel.grid.minor = element\_blank(),

panel.border = element\_blank(),

axis.line = element\_line(color = "black")

) +

stat\_compare\_means(method = "wilcox.test", # or "t.test"

aes(group = Recovery),

label = "p.signif",

size = 5)

ggsave(filename = "Patient\_NHISS\_Recovery.png")

# \*Age & NHISS correlation\* -----------------------------------------------------------------------------

folder <- "LinReg\_AgeVsNHISS"

createFolder(folder)

Metadata\_NeuroTest <- merge(Metadata\_NeuroTest, metadataP, by = "SampleID")

Metadata\_NeuroTest <- Metadata\_NeuroTest %>% filter(!is.na(NHISS), !is.na(Age), !is.na(Timepoint))

AgeVsNHISS\_correlation <- data.frame(Group = character(), Timepoint = character(), N = numeric(), p.value = numeric(), Estimate = numeric(), Est\_CI\_Lower = numeric(), Est\_CI\_Upper = numeric(), Coefficient = numeric(), Coef\_CI\_Lower = numeric(), Coef\_CI\_Upper = numeric())

for (tp in unique(Metadata\_NeuroTest$Timepoint)) {

Metadata\_NeuroTest\_tp <- Metadata\_NeuroTest %>% filter(Timepoint == tp)

if (nrow(Metadata\_NeuroTest\_tp) <= 0) next

# Outlier removal

lm\_initial <- lm(NHISS ~ Age, data = Metadata\_NeuroTest\_tp)

residuals\_values <- residuals(lm\_initial)

outlier\_indices <- which(abs(residuals\_values) > (3 \* sd(residuals\_values)))

# Remove outliers only if any are detected

if (length(outlier\_indices) > 0) {

Metadata\_NHISS\_tp\_filtered <- Metadata\_NeuroTest\_tp[-outlier\_indices, ]

} else {

Metadata\_NHISS\_tp\_filtered <- Metadata\_NeuroTest\_tp # No outliers, keep the original dataframe

}

if (nrow(Metadata\_NHISS\_tp\_filtered) < 2) next # Ensure there are enough data points after outlier removal

tryCatch({

# Pearson's correlation

x <- cor.test(Metadata\_NHISS\_tp\_filtered$NHISS, Metadata\_NHISS\_tp\_filtered$Age)

row <- data.frame(

Group = "All",

Timepoint = tp,

N = nrow(Metadata\_NHISS\_tp\_filtered),

p.value = x$p.value,

Estimate = x$estimate,

Est\_CI\_Lower = x$conf.int[1],

Est\_CI\_Upper = x$conf.int[2],

Coefficient = NA,

Coef\_CI\_Lower = NA,

Coef\_CI\_Upper = NA

)

# Linear regression for coefficient and CI

lm\_result <- lm(NHISS ~ Age, data = Metadata\_NHISS\_tp\_filtered)

coef\_x <- summary(lm\_result)$coefficients[2, 1]

ci <- confint(lm\_result)[2, ]

row$Coefficient <- coef\_x

row$Coef\_CI\_Lower <- ci[1]

row$Coef\_CI\_Upper <- ci[2]

titel <- paste("Age vs NHISS at ", tp," (All Samples)", sep="")

file\_name <- paste("LinReg\_AgeVsNHISS/",titel, ".png", sep = "")

ggplot(data = Metadata\_NHISS\_tp\_filtered, aes(x = Age, y = NHISS)) +

geom\_smooth(method = "glm", color = "black") +

geom\_point(aes(color = Timepoint), size = 2) +

ggtitle(titel) +

xlab("Age") +

ylab("NHISS") +

scale\_color\_manual(values = my\_colors) +

theme(text = element\_text(size=14)) +

theme(

panel.background = element\_rect(fill = "white", color = "black"),

plot.background = element\_rect(fill = "white", color = "black"),

text = element\_text(color = "black"),

panel.grid.major = element\_line(color = "gray", size = 0.2),

panel.grid.minor = element\_blank(),

panel.grid.major.x = element\_blank()

)

ggsave(filename=file\_name)

AgeVsNHISS\_correlation <- rbind(AgeVsNHISS\_correlation, row)

}, error = function(e) {

cat("Error in Pearson's Correlation for timepoint", tp, "- skipping this comparison.\n")

})

for (sex in unique(Metadata\_NeuroTest\_tp$Sex)) {

Metadata\_NHISS\_sex <- Metadata\_NeuroTest\_tp %>% filter(Sex == sex)

# Outlier removal

lm\_initial <- lm(NHISS ~ Age, data = Metadata\_NHISS\_sex)

residuals\_values <- residuals(lm\_initial)

outlier\_indices <- which(abs(residuals\_values) > (3 \* sd(residuals\_values)))

# Remove outliers only if any are detected

if (length(outlier\_indices) > 0) {

Metadata\_NHISS\_sex\_filtered <- Metadata\_NHISS\_sex[-outlier\_indices, ]

} else {

Metadata\_NHISS\_sex\_filtered <- Metadata\_NHISS\_sex # No outliers, keep the original dataframe

}

if (nrow(Metadata\_NHISS\_sex\_filtered) < 2) next # Ensure there are enough data points after outlier removal

tryCatch({

# Pearson's correlation for Sex

x <- cor.test(Metadata\_NHISS\_sex\_filtered$NHISS, Metadata\_NHISS\_sex\_filtered$Age)

row <- data.frame(

Group = sex,

Timepoint = tp,

N = nrow(Metadata\_NHISS\_sex\_filtered),

p.value = x$p.value,

Estimate = x$estimate,

Est\_CI\_Lower = x$conf.int[1],

Est\_CI\_Upper = x$conf.int[2],

Coefficient = NA,

Coef\_CI\_Lower = NA,

Coef\_CI\_Upper = NA

)

# Linear regression for coefficient and CI

lm\_result\_sex <- lm(NHISS ~ Age, data = Metadata\_NHISS\_sex\_filtered)

coef\_x <- summary(lm\_result\_sex)$coefficients[2, 1]

ci <- confint(lm\_result\_sex)[2, ]

row$Coefficient <- coef\_x

row$Coef\_CI\_Lower <- ci[1]

row$Coef\_CI\_Upper <- ci[2]

titel <- paste("Age vs NHISS at ", tp," (",sex, " Samples)", sep="")

file\_name <- paste("LinReg\_AgeVsNHISS/",titel, ".png", sep = "")

ggplot(data = Metadata\_NHISS\_sex\_filtered, aes(x = Age, y = NHISS)) +

geom\_smooth(method = "glm", color = "black") +

geom\_point(aes(color = Sex), size = 2) +

ggtitle(titel) +

xlab("Age") +

ylab("NHISS") +

scale\_color\_manual(values = c("M" = "#A67C00", "F" = "#1D04C2")) +

theme(text = element\_text(size=14)) +

theme(

panel.background = element\_rect(fill = "white", color = "black"),

plot.background = element\_rect(fill = "white", color = "black"),

text = element\_text(color = "black"),

panel.grid.major = element\_line(color = "gray", size = 0.2),

panel.grid.minor = element\_blank(),

panel.grid.major.x = element\_blank()

)

ggsave(filename=file\_name)

AgeVsNHISS\_correlation <- rbind(AgeVsNHISS\_correlation, row)

}, error = function(e) {

cat("Error for", tp, "&", sex, "- skipping this comparison.\n")

})

}

for (cat in unique(Metadata\_NeuroTest\_tp$Category)) {

Metadata\_NHISS\_cat <- Metadata\_NeuroTest\_tp %>% filter(Category == cat)

# Outlier removal

lm\_initial <- lm(NHISS ~ Age, data = Metadata\_NHISS\_cat)

residuals\_values <- residuals(lm\_initial)

outlier\_indices <- which(abs(residuals\_values) > (3 \* sd(residuals\_values)))

# Remove outliers only if any are detected

if (length(outlier\_indices) > 0) {

Metadata\_NHISS\_cat\_filtered <- Metadata\_NHISS\_cat[-outlier\_indices, ]

} else {

Metadata\_NHISS\_cat\_filtered <- Metadata\_NHISS\_cat # No outliers, keep the original dataframe

}

if (nrow(Metadata\_NHISS\_cat\_filtered) < 2) next # Ensure there are enough data points after outlier removal

tryCatch({

# Pearson's correlation for Category

x <- cor.test(Metadata\_NHISS\_cat\_filtered$NHISS, Metadata\_NHISS\_cat\_filtered$Age)

row <- data.frame(

Group = cat,

Timepoint = tp,

N = nrow(Metadata\_NHISS\_cat\_filtered),

p.value = x$p.value,

Estimate = x$estimate,

Est\_CI\_Lower = x$conf.int[1],

Est\_CI\_Upper = x$conf.int[2],

Coefficient = NA,

Coef\_CI\_Lower = NA,

Coef\_CI\_Upper = NA

)

# Linear regression for coefficient and CI

lm\_result\_cat <- lm(NHISS ~ Age, data = Metadata\_NHISS\_cat\_filtered)

coef\_x <- summary(lm\_result\_cat)$coefficients[2, 1]

ci <- confint(lm\_result\_cat)[2, ]

row$Coefficient <- coef\_x

row$Coef\_CI\_Lower <- ci[1]

row$Coef\_CI\_Upper <- ci[2]

titel <- paste("Age vs NHISS at ", tp, " (", cat, " Samples)", sep="")

file\_name <- paste("LinReg\_AgeVsNHISS/", titel, ".png", sep="")

ggplot(data = Metadata\_NHISS\_cat\_filtered, aes(x = Age, y = NHISS)) +

geom\_smooth(method = "glm", color = "black") +

geom\_point(aes(color = Category), size = 2) +

ggtitle(titel) +

xlab("Age") +

ylab("NHISS") +

scale\_color\_manual(values = c("MINOR" = "#A67C00", "MODERATE" = "#700606")) +

theme(text = element\_text(size=14)) +

theme(

panel.background = element\_rect(fill = "white", color = "black"),

plot.background = element\_rect(fill = "white", color = "black"),

text = element\_text(color = "black"),

panel.grid.major = element\_line(color = "gray", size = 0.2),

panel.grid.minor = element\_blank(),

panel.grid.major.x = element\_blank()

)

ggsave(filename = file\_name)

AgeVsNHISS\_correlation <- rbind(AgeVsNHISS\_correlation, row)

}, error = function(e) {

cat("Error for", tp, "&", cat, "- skipping this comparison.\n")

})

}

}

file\_name\_s<- "LinReg\_AgeVsNHISS/Pearson's Correlation\_AgeVsNHISS.csv"

write.csv(AgeVsNHISS\_correlation, file\_name\_s, row.names = FALSE)