Load Libraries and set wd

Load Data

```
# Load & clean ISS expression matrix
ISS_gene_expression <- fread("ISS_gene_expression.txt", header = TRUE) # load file
rownames(ISS_gene_expression) <- ISS_gene_expression[[1]] # set rownames to the gene names stored in column 1
ISS_gene_expression[[1]] <- NULL # now remove column 1
colnames(ISS_gene_expression) <- as.character(colnames(ISS_gene_expression)) # ensure colnames are strings

# Load Invasive Cells data
load("/home/glennrdx/Documents/lab_projects/microenvironments/01_data/invasive_cells.RData")
```

Filter gene expression df to invasive tumours only (filter by invasive tumour cell ID)

```
inv_tumour_id <- invasive_cells$cell_id # get inv tumour ID's
inv_tumour_logical_vec <- colnames(ISS_gene_expression) %in% inv_tumour_id # create logical vec of tumour ID's
inv_tumour_gene_expression <- ISS_gene_expression[, ..inv_tumour_logical_vec] # finally, filter the gene expressio
gene_names <- rownames(ISS_gene_expression) # save gene names (save row names)
col_inv_id <- colnames(inv_tumour_gene_expression) # save cell id of invasive cells (save colnames)
inv_tumour_gene_expression <- data.frame(t(inv_tumour_gene_expression)) # transposing removes row/col names
colnames(inv_tumour_gene_expression) <- gene_names # reassign lost gene names
rownames(inv_tumour_gene_expression) <- col_inv_id # reassign lost row names
inv_tumour_gene_expression$cell_id <- as.integer(rownames(inv_tumour_gene_expression)) # create new column called
```

Attach surface classification data to inv tumour gene expression matrix

```
surface_class <- invasive_cells[, c('cell_id', 'surface_class')] # get surface class annotations
inv_tumour_gene_expression <- merge(surface_class, inv_tumour_gene_expression, by = 'cell_id') # merge two df's by
rownames(inv_tumour_gene_expression) <- inv_tumour_gene_expression$cell_id # assign cell id to rownames
inv_tumour_gene_expression$cell_id <- NULL
```

Split full dataset into training / test datasets (80/20 split)

```
# Take 80% for training dataset
inv_gene_exp_train <- inv_tumour_gene_expression %>%
  mutate(cell_id = rownames(.)) %>%
  group_by(surface_class) %>%
  slice_sample(prop = 0.8) %>%
  ungroup()

# Use remaining 20% for test dataset
inv_gene_exp_test <- inv_tumour_gene_expression %>%
  mutate(cell_id = rownames(.)) %>%
  filter(!cell_id %in% inv_gene_exp_train$cell_id)

# Restore the cell_id's to the rownames and remove the cell_id column
inv_gene_exp_train <- as.data.frame(inv_gene_exp_train) # convert to df to allow assignment of rownames
rownames(inv_gene_exp_train) <- inv_gene_exp_train$cell_id
inv_gene_exp_train$cell_id <- NULL

inv_gene_exp_test <- as.data.frame(inv_gene_exp_test) # convert to df to allow assignment of rownames
rownames(inv_gene_exp_test) <- inv_gene_exp_test$cell_id
inv_gene_exp_test$cell_id <- NULL
```

Feature selection

- Remove genes that have zero variance and thus are not discriminatory between groups.

```
gene_cols <- setdiff(colnames(inv_gene_exp_train), "surface_class")

# Calculate variance within each group for each gene
keep_genes <- sapply(gene_cols, function(gene) {
  var_by_group <- inv_gene_exp_train %>%
    group_by(surface_class) %>%
    summarise(var = var(.data[[gene]], na.rm = TRUE), .groups = 'drop')

  # Keep gene if it has non-zero variance in at least one group
  any(var_by_group$var > 0)
})

# Filter to keep only informative genes
genes_to_keep <- names(keep_genes)[keep_genes]
inv_gene_exp_filtered <- inv_gene_exp_train %>%
  dplyr::select(surface_class, all_of(genes_to_keep))
```

Fit LDA model

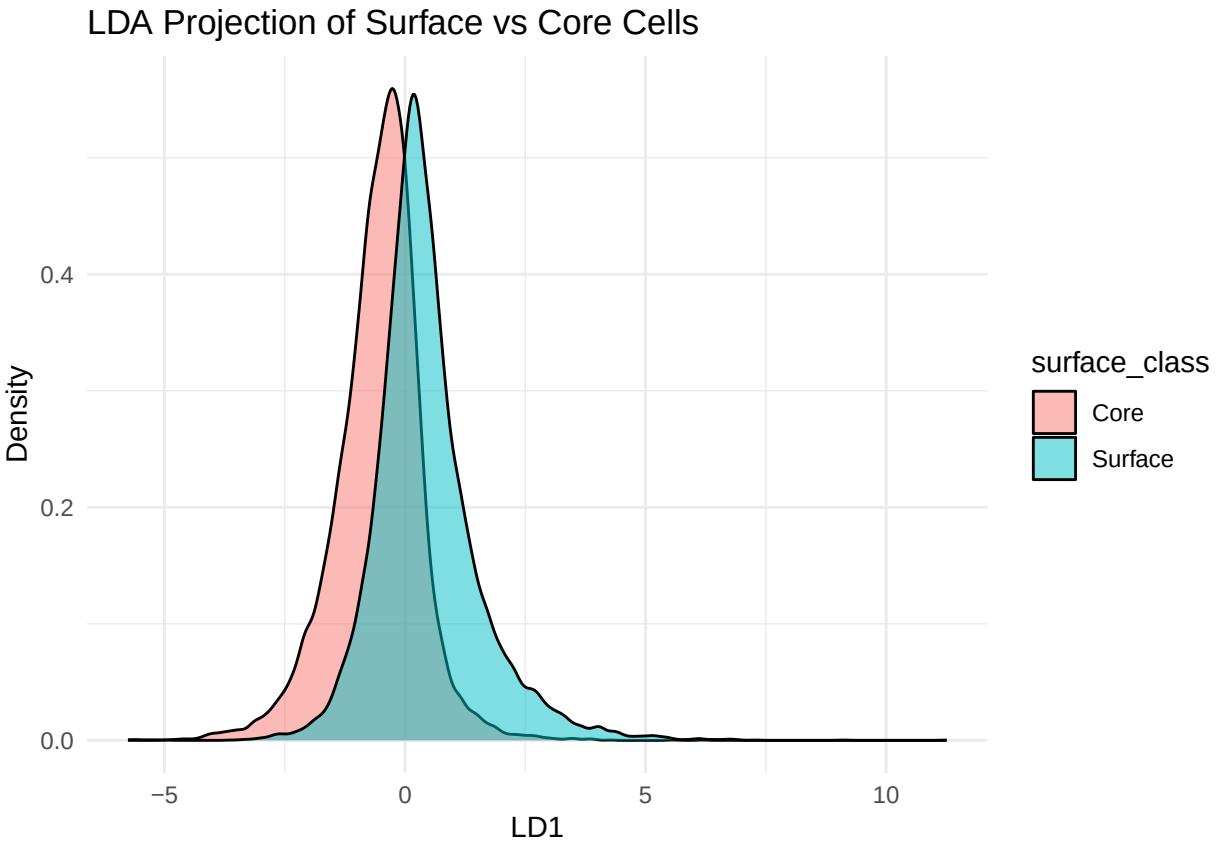
```
fit_lda <- lda(surface_class ~ ., inv_gene_exp_filtered)
```

Plot first two linear discriminants

```
library(ggplot2)

lda_values <- predict(fit_lda)$x
lda_df <- data.frame(LD1 = lda_values, surface_class = inv_gene_exp_filtered$surface_class)

ggplot(lda_df, aes(x = LD1, fill = surface_class)) +
  geom_density(alpha = 0.5) +
  theme_minimal() +
  labs(title = "LDA Projection of Surface vs Core Cells", x = "LD1", y = "Density")
```



Predict surface_class with LDA model

```
# Subset test data to only include genes used in the LDA model
inv_gene_exp_test_filtered <- inv_gene_exp_test %>%
  dplyr::select(all_of(genes_to_keep))

# Predict surface_class for test data
lda_predictions <- predict(fit_lda, newdata = inv_gene_exp_test_filtered)
```

```
predicted_classes <- lda_predictions$class
```

```
inv_gene_exp_test$predicted_surface_class <- predicted_classes
```

```
accuracy <- mean(predicted_classes == inv_gene_exp_test$surface_class)
print(paste("Test Accuracy:", round(accuracy, 3)))
```

```
## [1] "Test Accuracy: 0.707"
```

```
# Load caret for confusion matrix
library(caret)
```

```
## Loading required package: lattice
```

```
# Create confusion matrix
conf_matrix <- confusionMatrix(
  factor(predicted_classes),
  factor(inv_gene_exp_test$surface_class)
)

# Print confusion matrix and stats
print(conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Core Surface
##    Core    1831     817
##    Surface 1200    3028
```

```
##
##                Accuracy : 0.7067
##                  95% CI : (0.6957, 0.7174)
##     No Information Rate : 0.5592
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3969
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6041
##             Specificity : 0.7875
##          Pos Pred Value : 0.6915
##          Neg Pred Value : 0.7162
##              Prevalence : 0.4408
##          Detection Rate : 0.2663
##    Detection Prevalence : 0.3851
##       Balanced Accuracy : 0.6958
##
##        'Positive' Class : Core
##
```