

Evaluating the Impact of EPO Correction on Simulated Group Discrimination

Tecla Duran Fort

2025-05-03

Contents

1. Introduction	1
2. Load Data and Metadata	1
Visualizing Group vs Technical Effects	2
3. Define Simulation and Evaluation Functions	2
4. Run Simulation Across Clusters	4
5. Visualize AUC by Cluster	4
6. Robustness Check	20
Repeated Random Assignments	20
Global Validation Across Clusters and Perturbation Levels	21
7. Latent Structure Visualisation Before and After Correction	22
7.1 Latent Space Projection — Raw	24
7.2 Latent Space Projection — Corrected	25

1. Introduction

In GC-IMS data, external sources of variability such as time and batch can hinder the detection of biologically meaningful differences. Here, we simulate subtle group differences in randomly assigned “Control” and “Cancer” groups and evaluate the benefit of External Parameter Orthogonalization (EPO) using PLS-DA and AUC as performance metrics.

2. Load Data and Metadata

```
set.seed(1234)
df <- read.csv("../data/peak_table_var.csv")

meta <- df[, c("elapsed_time", "batch")]

# Get cluster names
clusters <- grep("^Cluster", names(df), value = TRUE)

# Assign random group labels
df$Group <- sample(rep(c("Control", "Cancer"), length.out = nrow(df)))
```

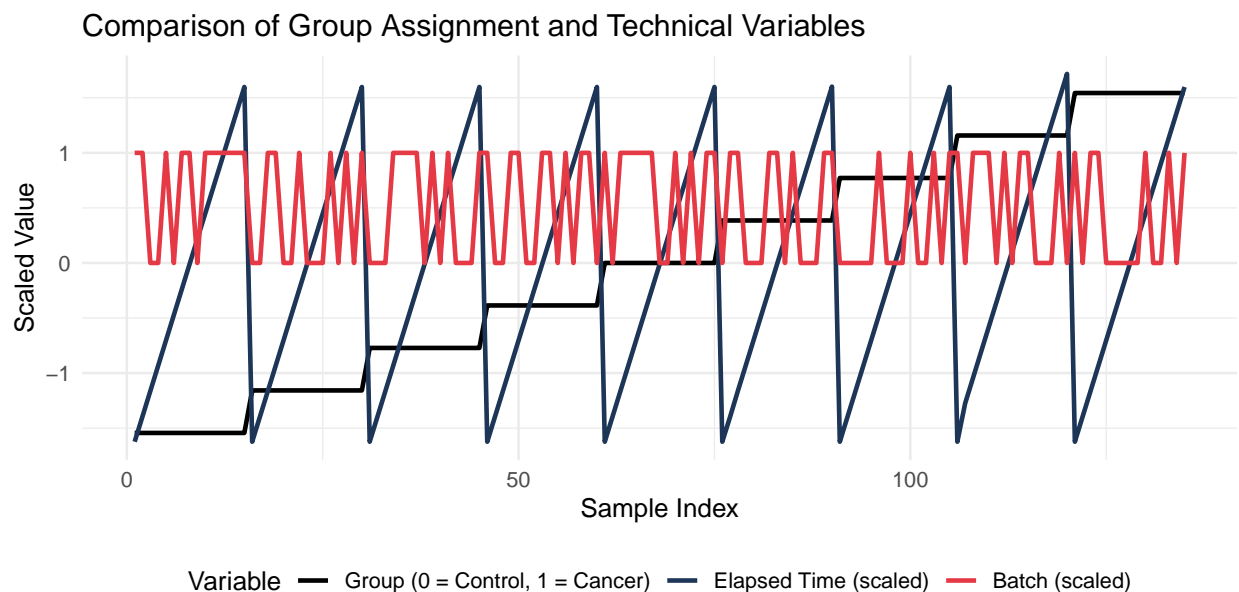
Visualizing Group vs Technical Effects

Before running the simulation, we visualize the alignment between the randomly assigned `Group` labels and the technical variables `elapsed_time` and `batch`. All values are scaled to the same range for direct comparison.

```
# Create a copy of df
df_viz <- df %>%
  dplyr::mutate(
    Sample = dplyr::row_number(),
    Group_numeric = ifelse(Group == "Control", 0, 1),
    elapsed_scaled = scale(elapsed_time)[, 1],
    batch_scaled = scale(as.numeric(batch))[, 1]
  )

df_long <- df_viz %>%
  dplyr::select(Sample, Group_numeric, elapsed_scaled, batch_scaled) %>%
  tidyr::pivot_longer(cols = -Sample, names_to = "Variable", values_to = "Value")

ggplot2::ggplot(df_long, ggplot2::aes(x = Sample, y = Value, color = Variable)) +
  ggplot2::geom_line(size = 1) +
  ggplot2::scale_color_manual(values = c("black", "#1D3557", "#E63946"),
                              labels = c("Group (0 = Control, 1 = Cancer)",
                                           "Elapsed Time (scaled)",
                                           "Batch (scaled)")) +
  ggplot2::theme_minimal(base_size = 12) +
  ggplot2::labs(title = "Comparison of Group Assignment and Technical Variables",
                x = "Sample Index", y = "Scaled Value", color = "Variable") +
  ggplot2::theme(legend.position = "bottom")
```



3. Define Simulation and Evaluation Functions

```
simulate_shift <- function(df, cluster, perc) {
  df_mod <- df
```

```

mean_val <- mean(df[[cluster]], na.rm = TRUE)
noise <- rnorm(nrow(df_mod), mean = perc * mean_val, sd = 0.05 * mean_val)
df_mod[[cluster]][df_mod$Group == "Cancer"] <- df_mod[[cluster]][df_mod$Group == "Cancer"] + noise[df_mod$Group == "Cancer"]
return(as.data.frame(df_mod))
}

compute_auc_plsda <- function(X, y, test_ratio = 0.5, ncomp = 2, seed=NULL) {
  if (!is.null(seed)) set.seed(seed)
  X <- as.matrix(X)
  y <- as.factor(y)
  classes <- levels(y)
  if (length(classes) != 2) return(NA)

  # Partició estratificada
  idx_class1 <- which(y == classes[1])
  idx_class2 <- which(y == classes[2])
  test_class1 <- sample(idx_class1, floor(length(idx_class1) * test_ratio))
  test_class2 <- sample(idx_class2, floor(length(idx_class2) * test_ratio))
  test_idx <- c(test_class1, test_class2)
  train_idx <- setdiff(seq_along(y), test_idx)

  X_train <- X[train_idx, , drop = FALSE]
  X_test <- X[test_idx, , drop = FALSE]
  y_train <- y[train_idx]
  y_test <- y[test_idx]

  # PLS-DA
  pls_model <- mixOmics::plsda(X_train, y_train, ncomp = ncomp)

  # Predicció
  pred <- predict(pls_model, X_test, dist = "max.dist")$predict[, , ncomp]
  prob <- pred[, classes[2]]

  # Càlcul de l'AUC
  roc_obj <- pROC::roc(response = y_test, predictor = prob, levels = classes)
  auc_val <- pROC::auc(roc_obj)

  return(as.numeric(auc_val))
}

evaluate_auc <- function(df_mod, meta, seed=NULL) {
  df_mod <- as.data.frame(df_mod)
  cluster_cols <- grep("^Cluster", names(df_mod), value = TRUE)
  X_raw <- as.matrix(df_mod[, cluster_cols])
  y <- as.factor(df_mod$Group)

  auc_raw <- compute_auc_plsda(X_raw, y, seed=seed)

  X_corr <- correction(correction(X_raw, meta$elapsed_time)$corrected,
                      meta$batch)$corrected

  auc_corr <- compute_auc_plsda(X_corr, y, seed=seed)
}

```

```

  return(c(Raw = auc_raw, Corrected = auc_corr))
}

```

4. Run Simulation Across Clusters

```

perturb_levels <- seq(0, 1, length.out = 20)
results_auc <- list()

for (cl in clusters) {
  cluster_auc <- purrr::map_dfr(perturb_levels, function(p) {
    df_mod <- simulate_shift(df, cl, p)
    aucs <- evaluate_auc(df_mod, meta, seed=42)
    tibble::tibble(Cluster = cl, Percent = p, Raw = aucs["Raw"], Corrected = aucs["Corrected"])
  })
  results_auc[[cl]] <- cluster_auc
}

auc_df <- dplyr::bind_rows(results_auc) %>%
  tidyr::pivot_longer(cols = c(Raw, Corrected), names_to = "Condition", values_to = "AUC")

```

5. Visualize AUC by Cluster

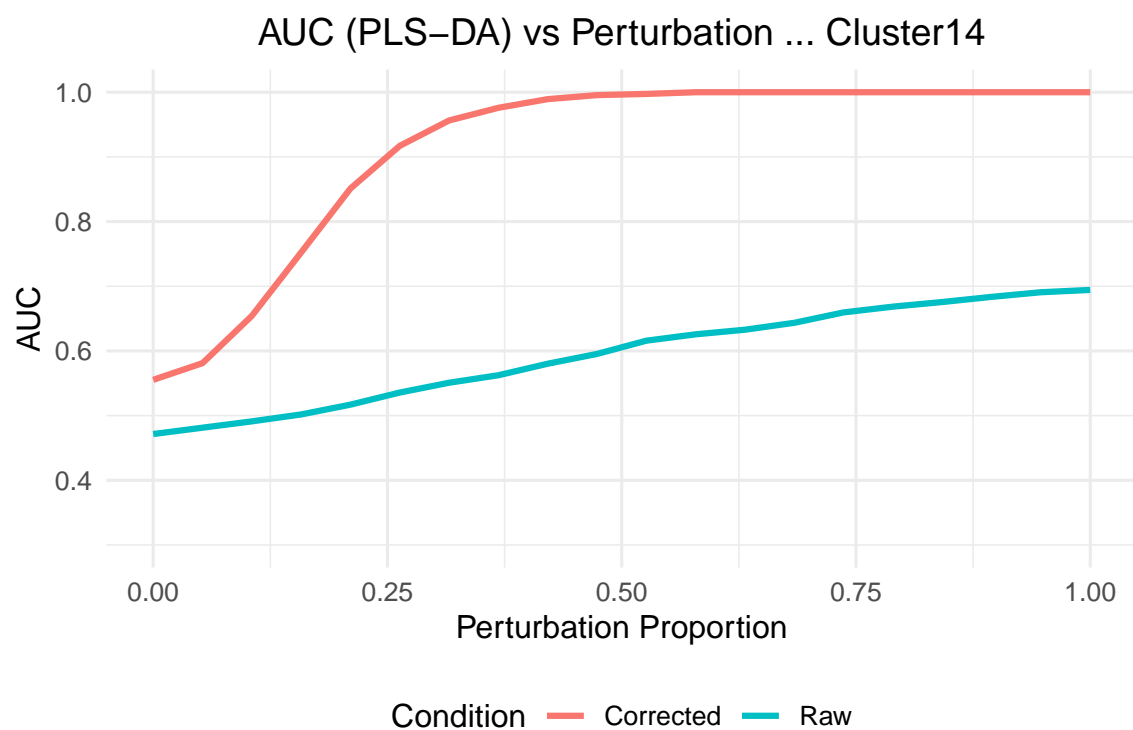
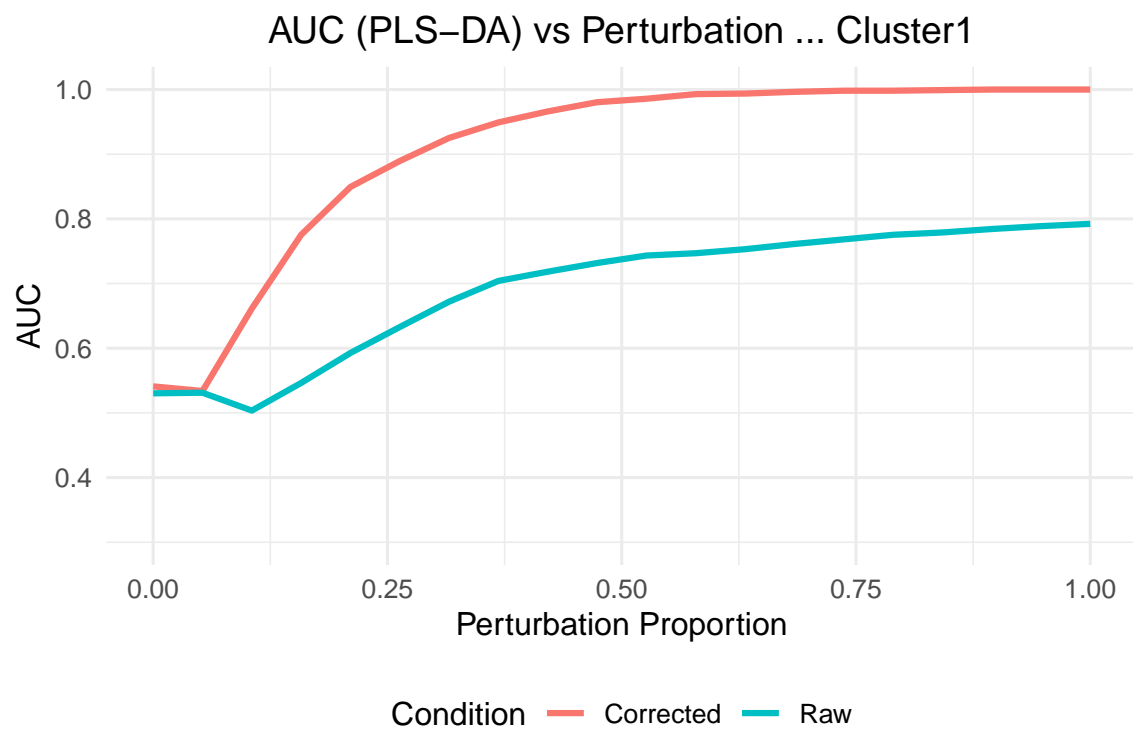
```

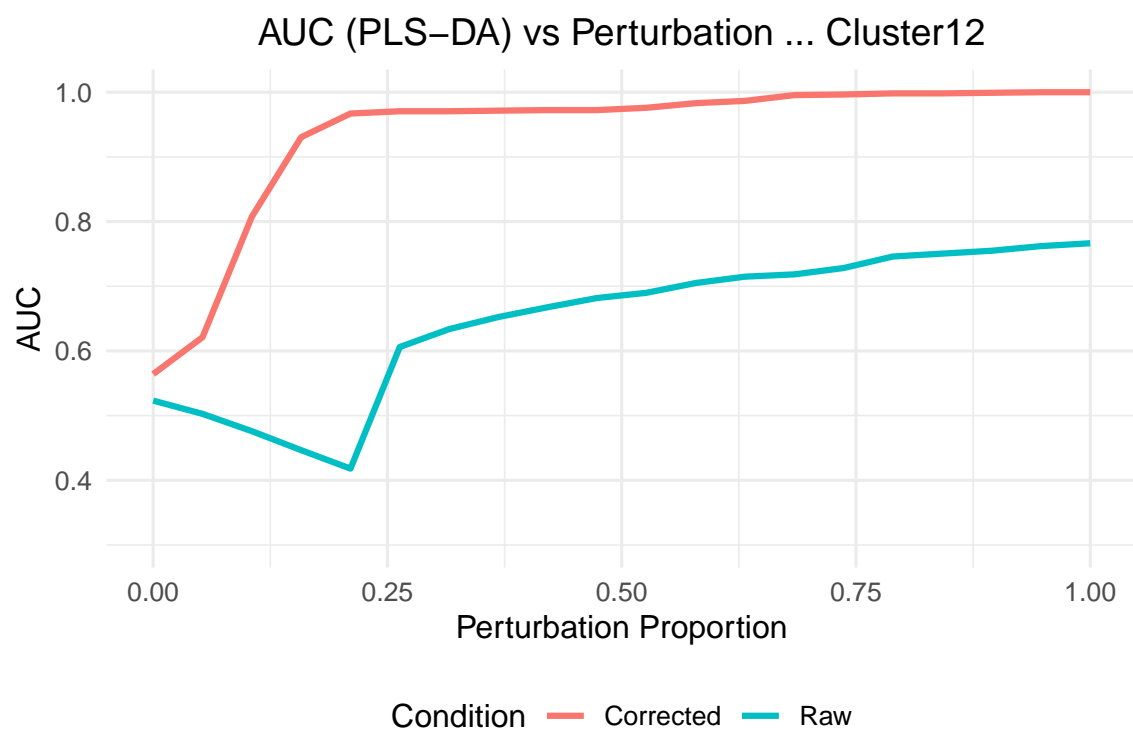
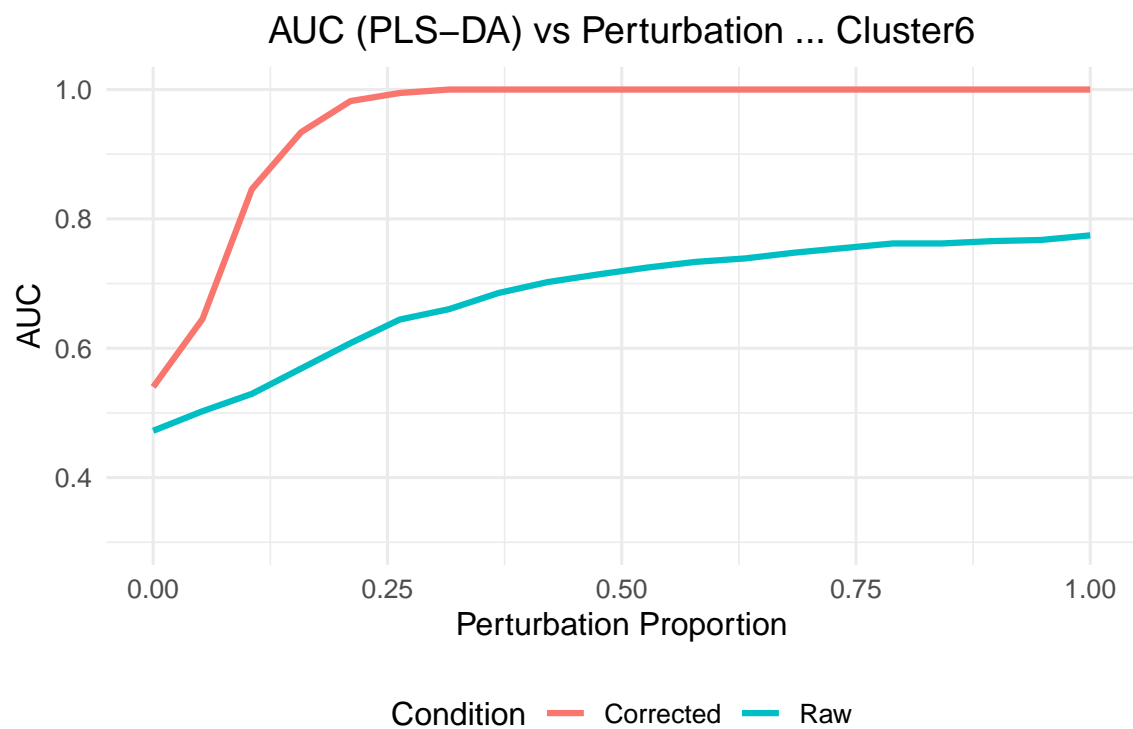
unique(auc_df$Cluster) %>% purrr::walk(function(cl) {
  df_plot <- dplyr::filter(auc_df, Cluster == cl)

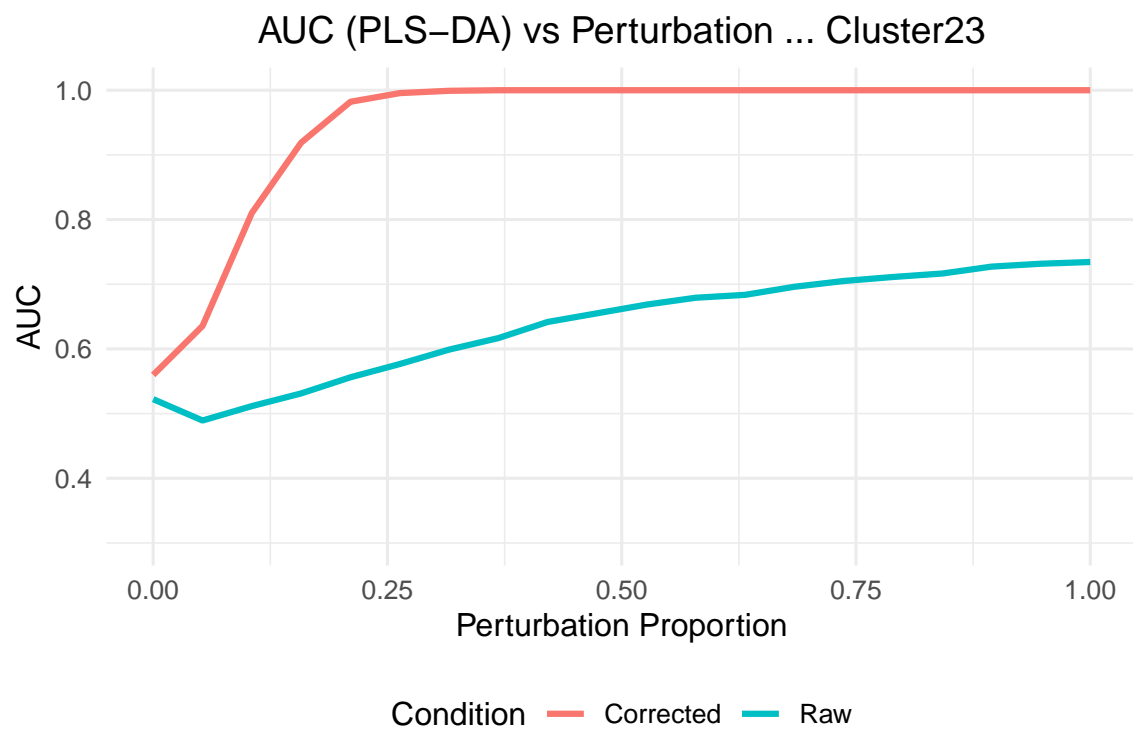
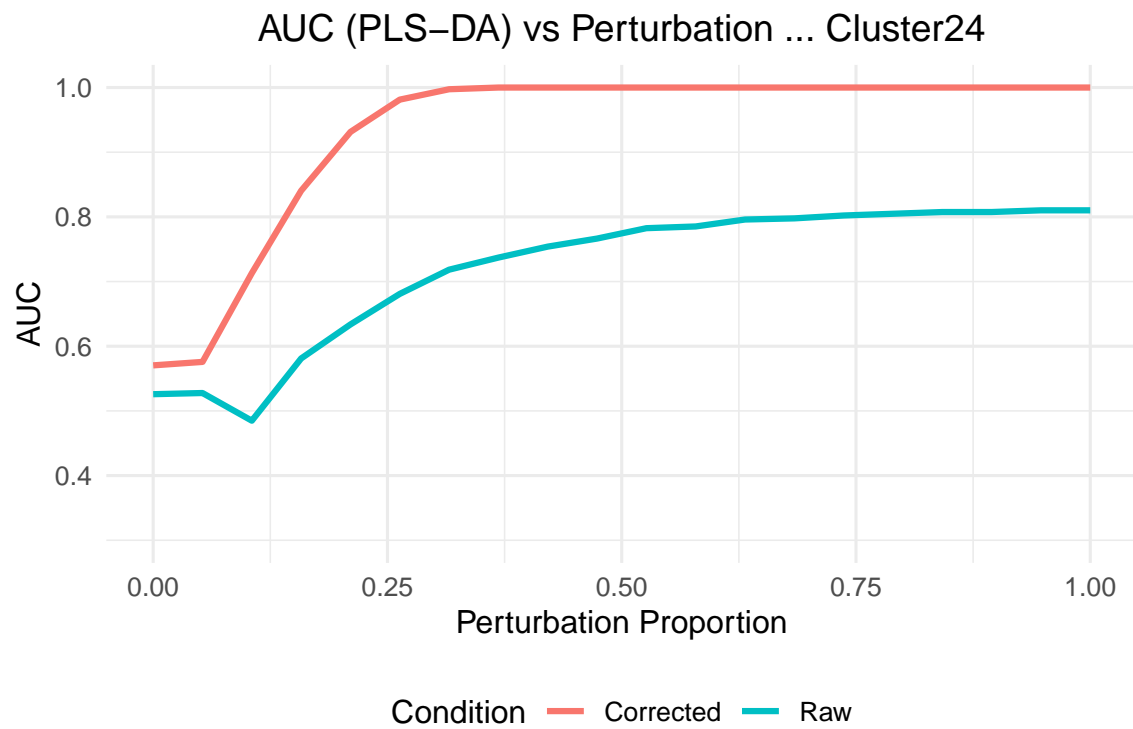
  p <- ggplot2::ggplot(df_plot, aes(x = Percent, y = AUC, color = Condition)) +
    ggplot2::geom_line(size = 1.1) +
    ggplot2::theme_minimal(base_size = 12) +
    ggplot2::labs(title = paste("AUC (PLS-DA) vs Perturbation -", cl),
                  x = "Perturbation Proportion", y = "AUC") +
    ggplot2::theme(legend.position = "bottom",
                  plot.title = element_text(hjust = 0.5))+
    ggplot2::ylim(0.3, 1)

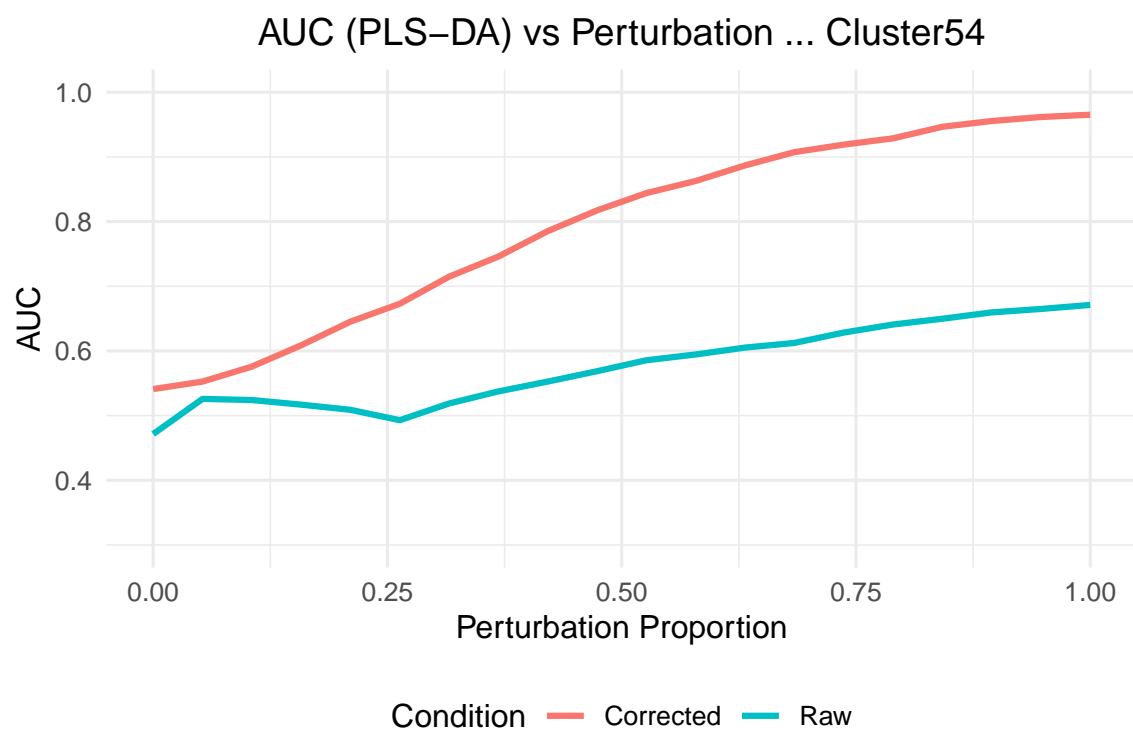
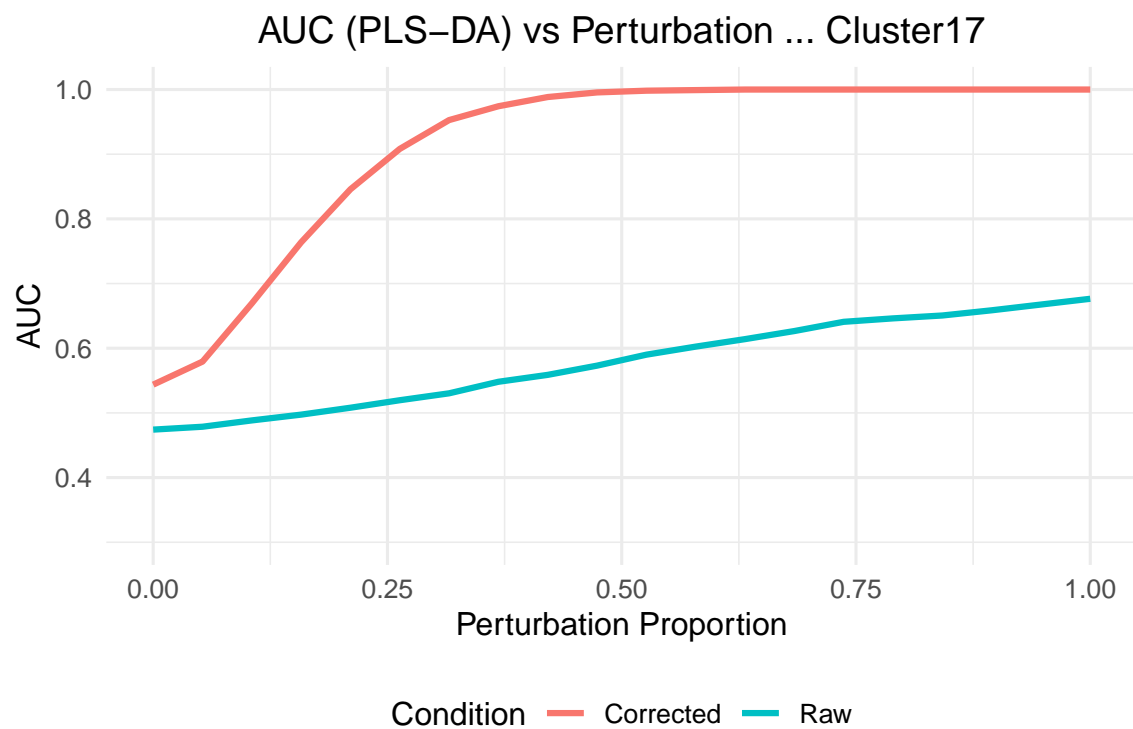
  print(p)
})

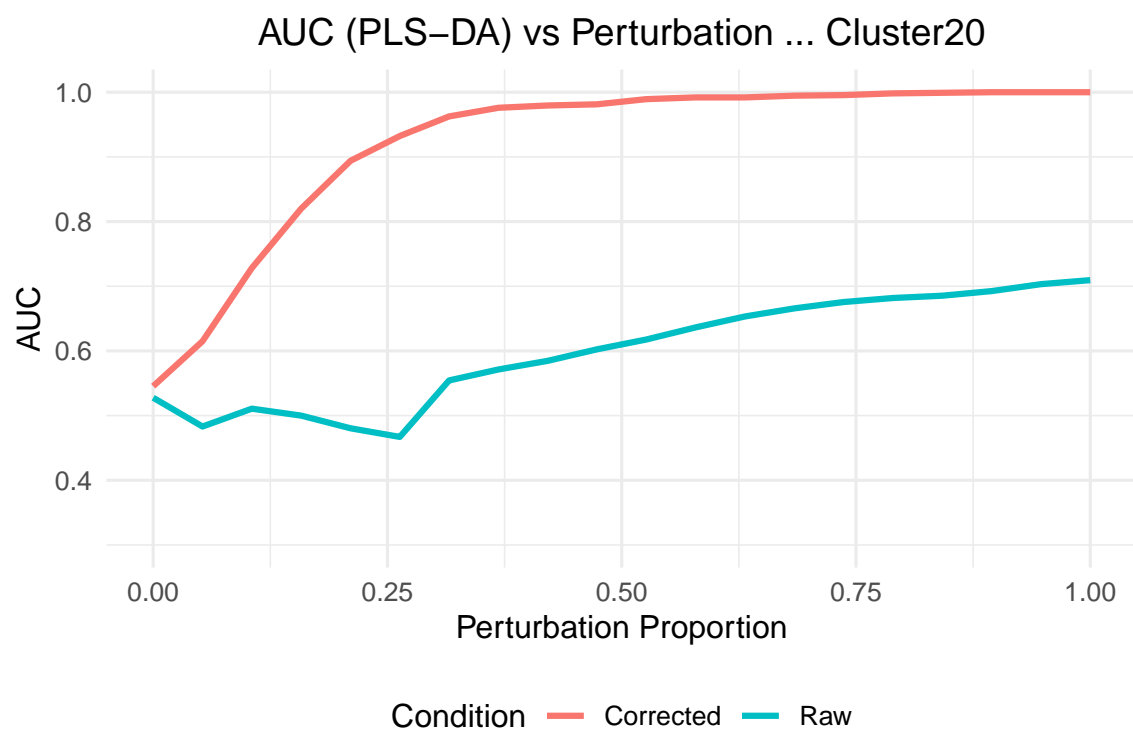
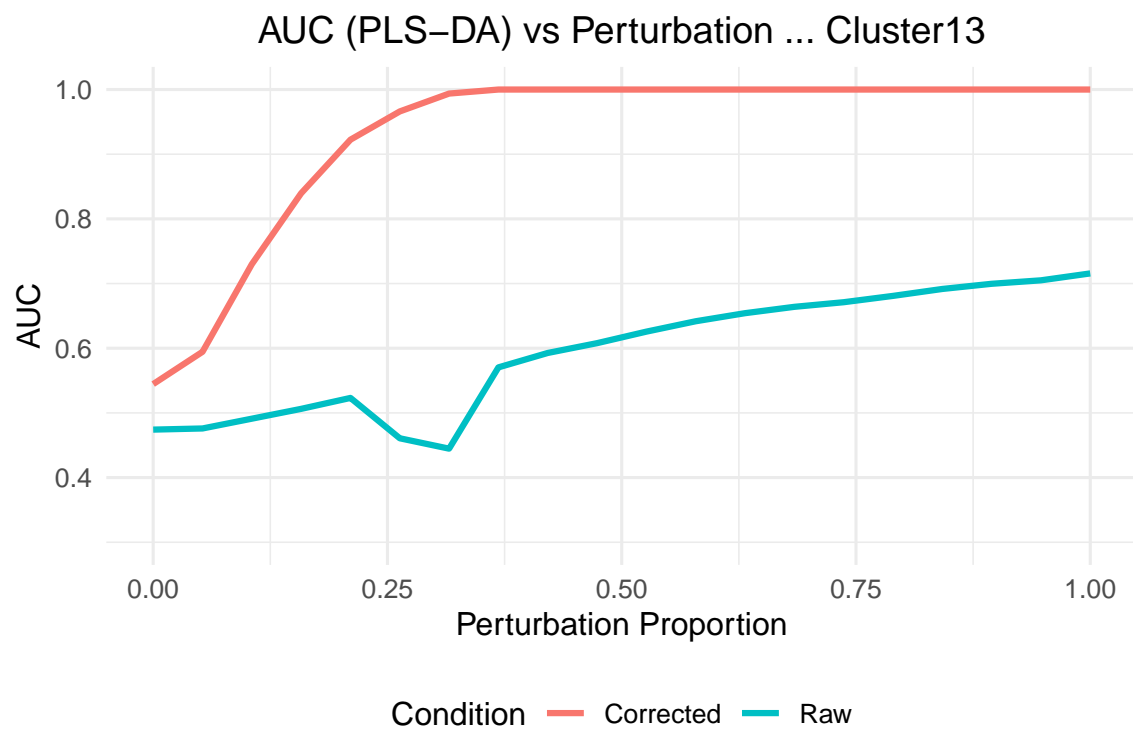
```

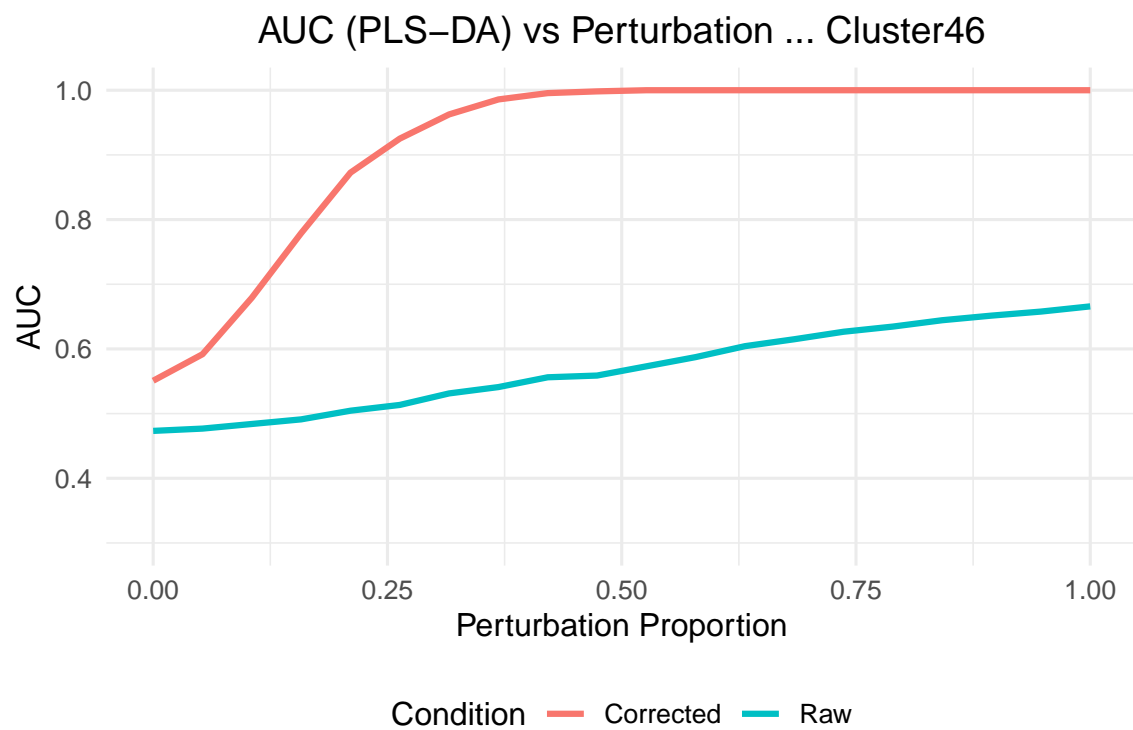
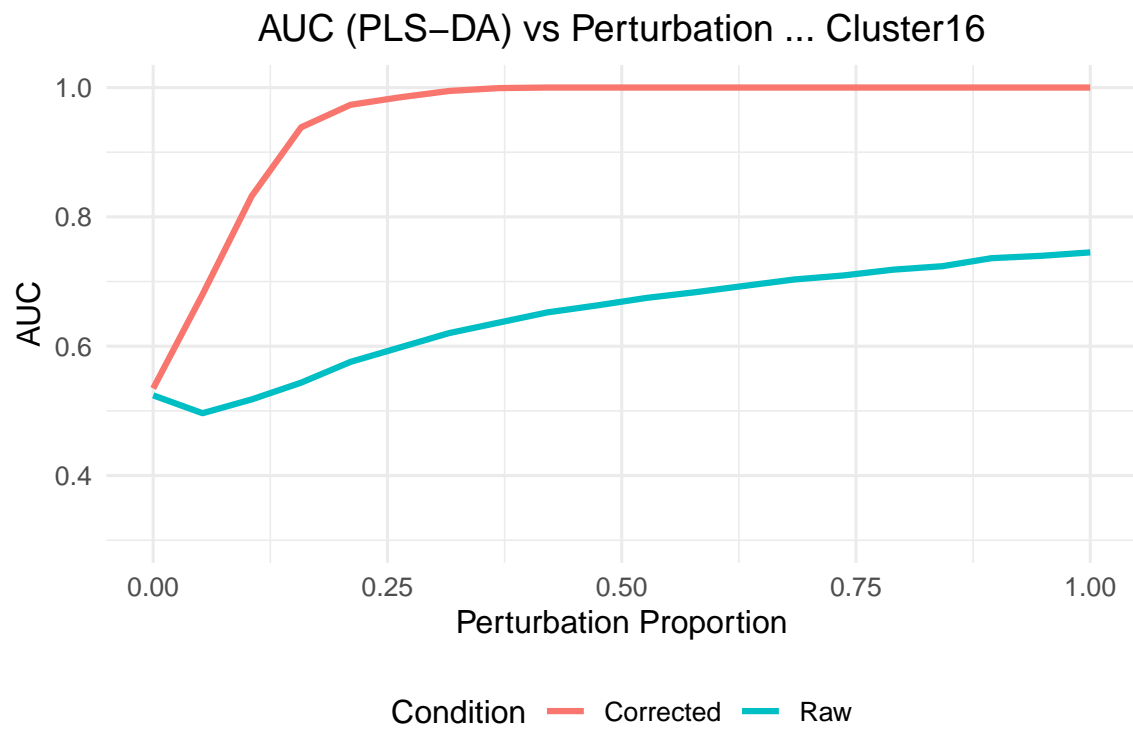


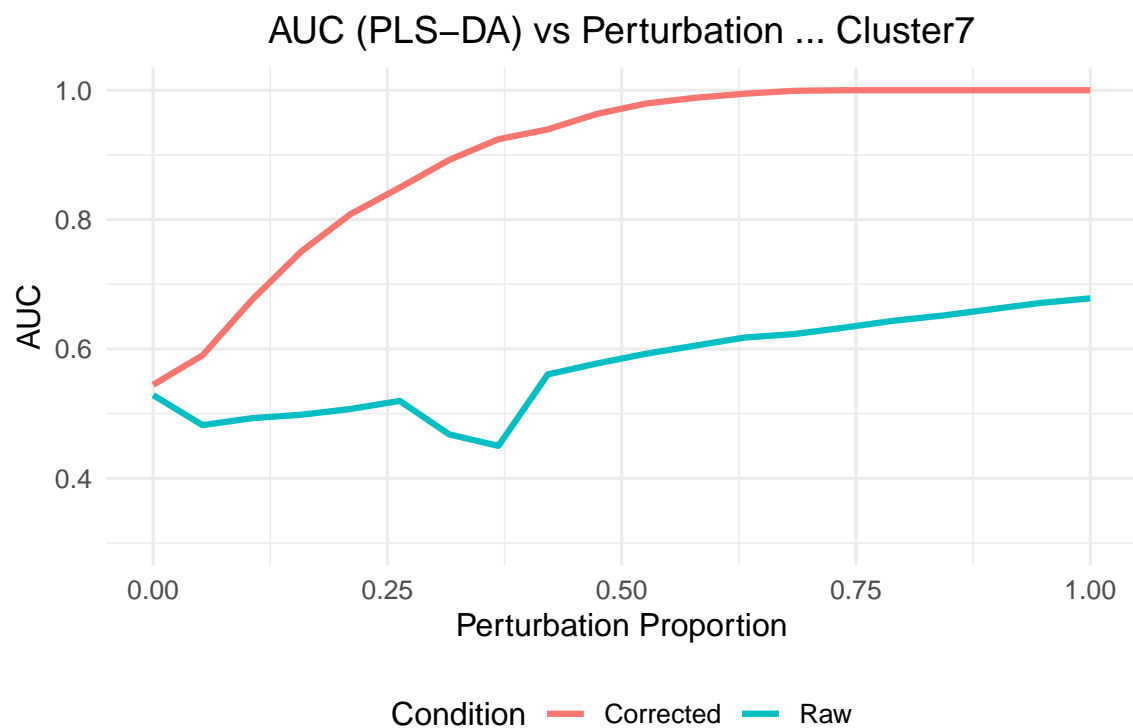
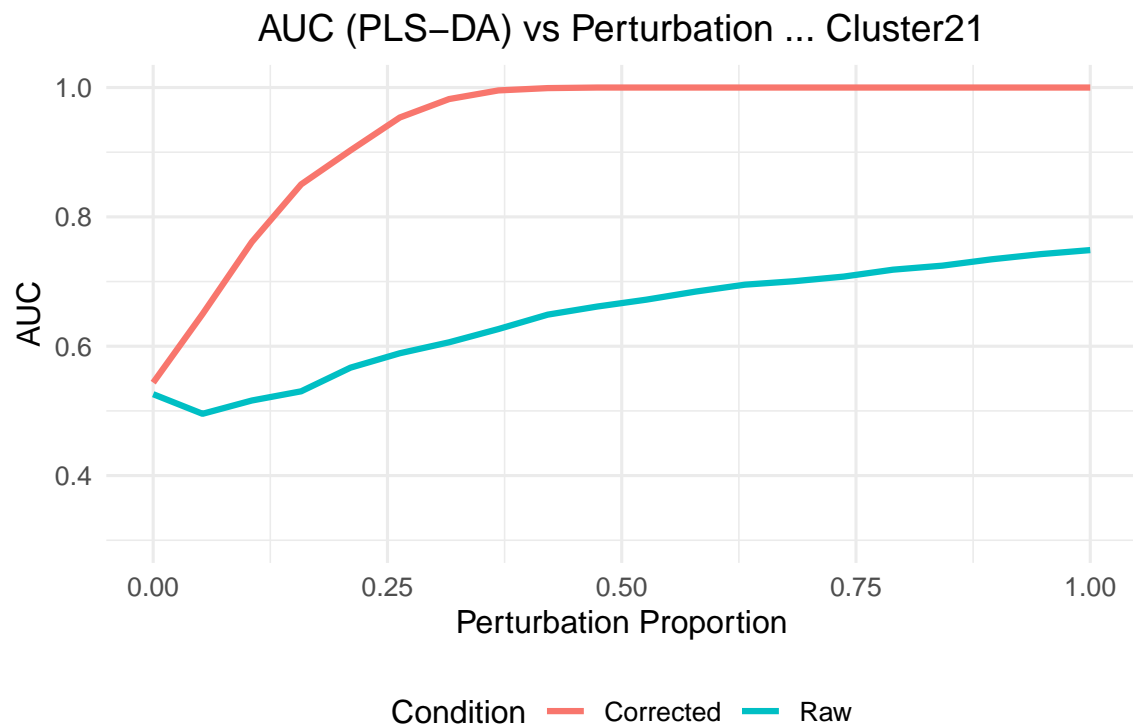


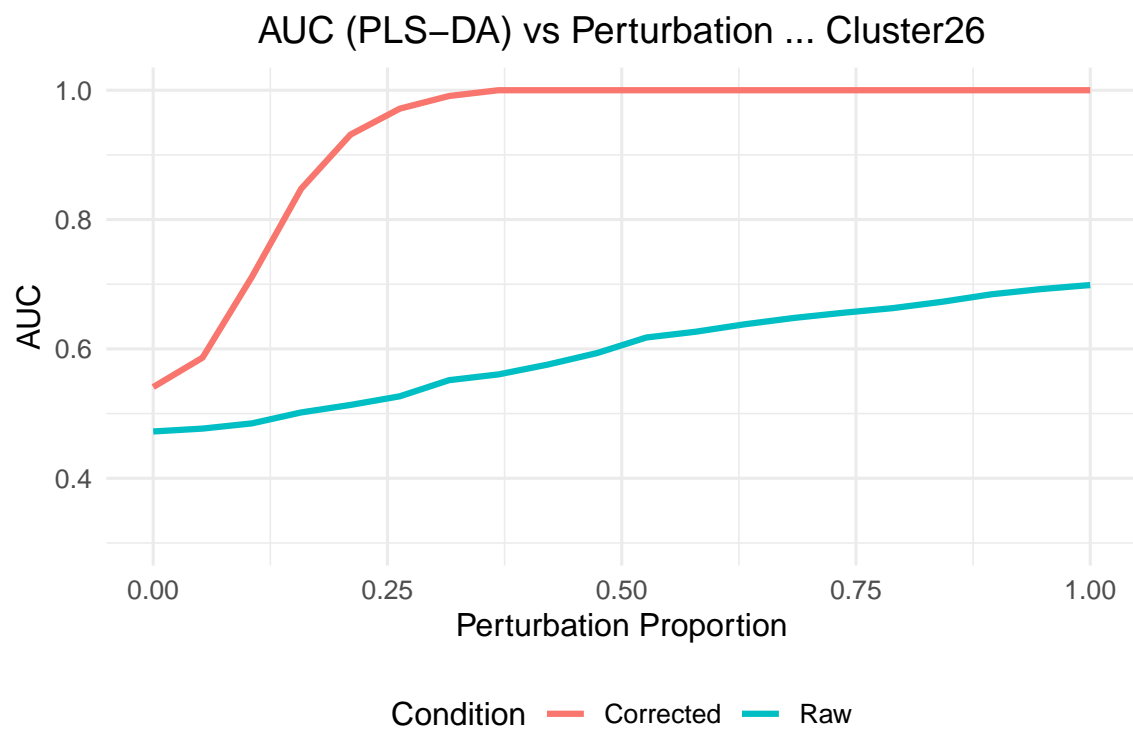
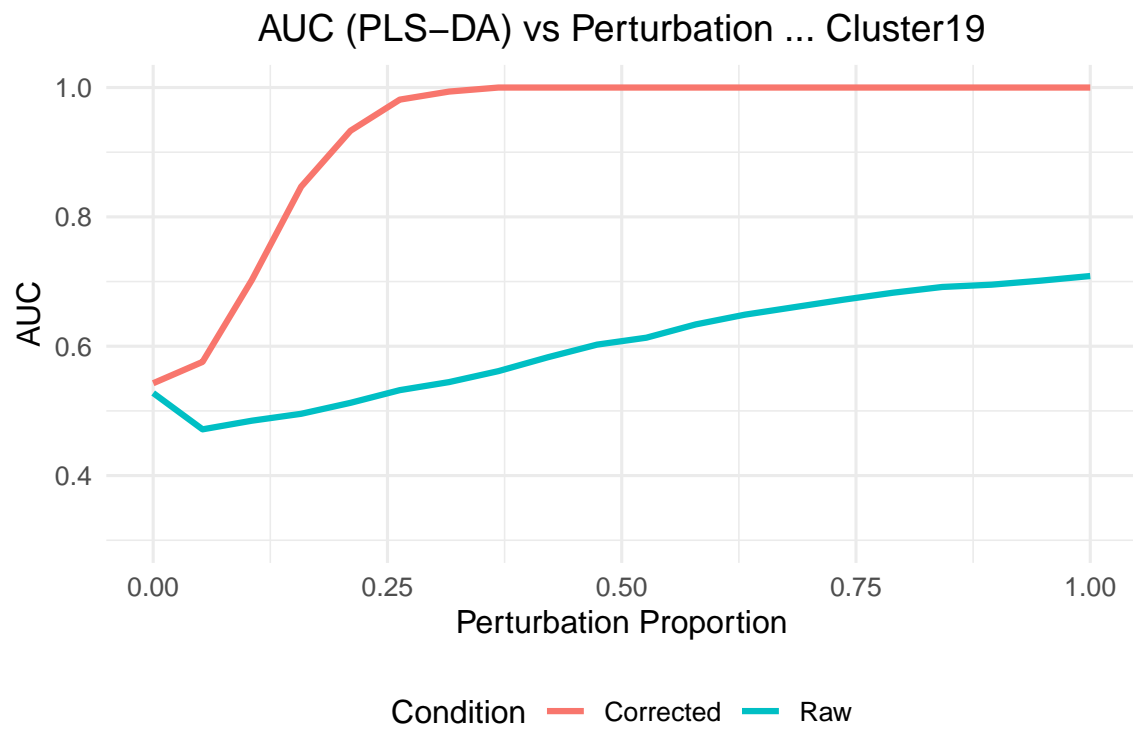


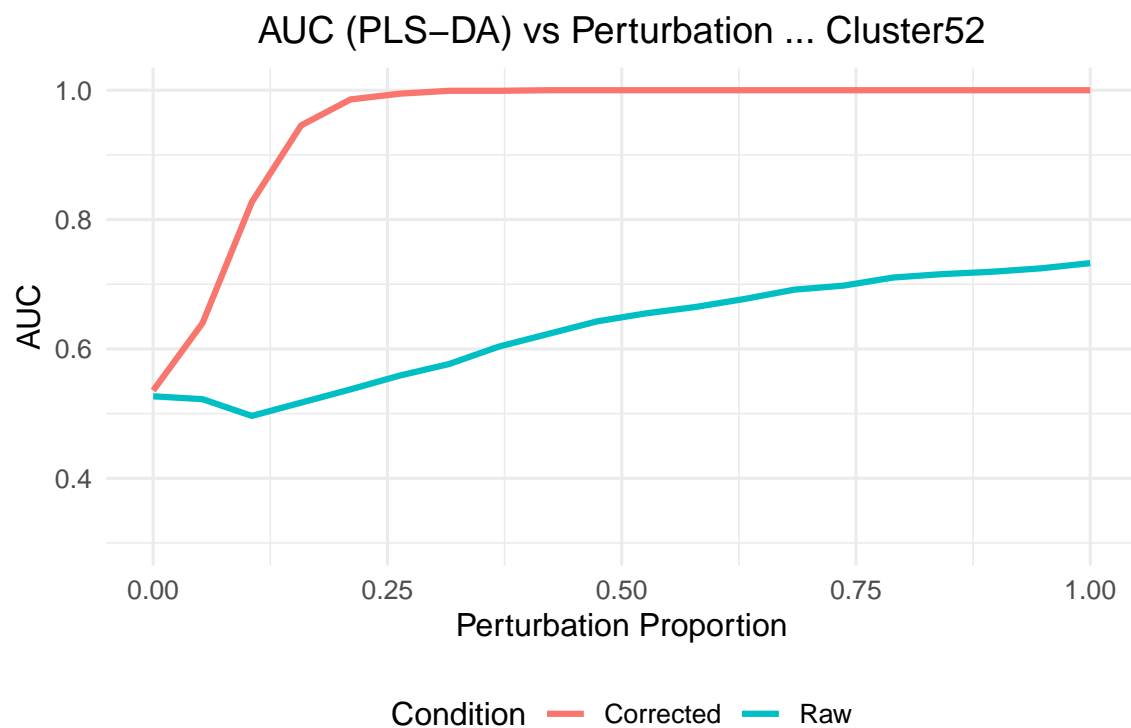
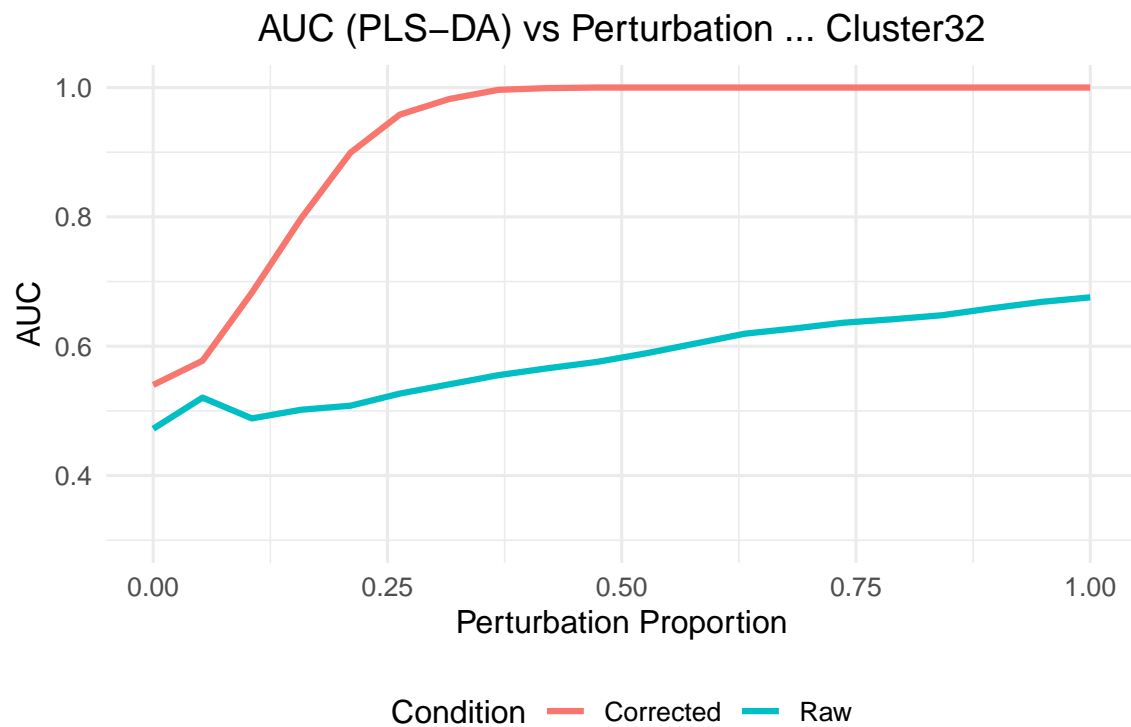


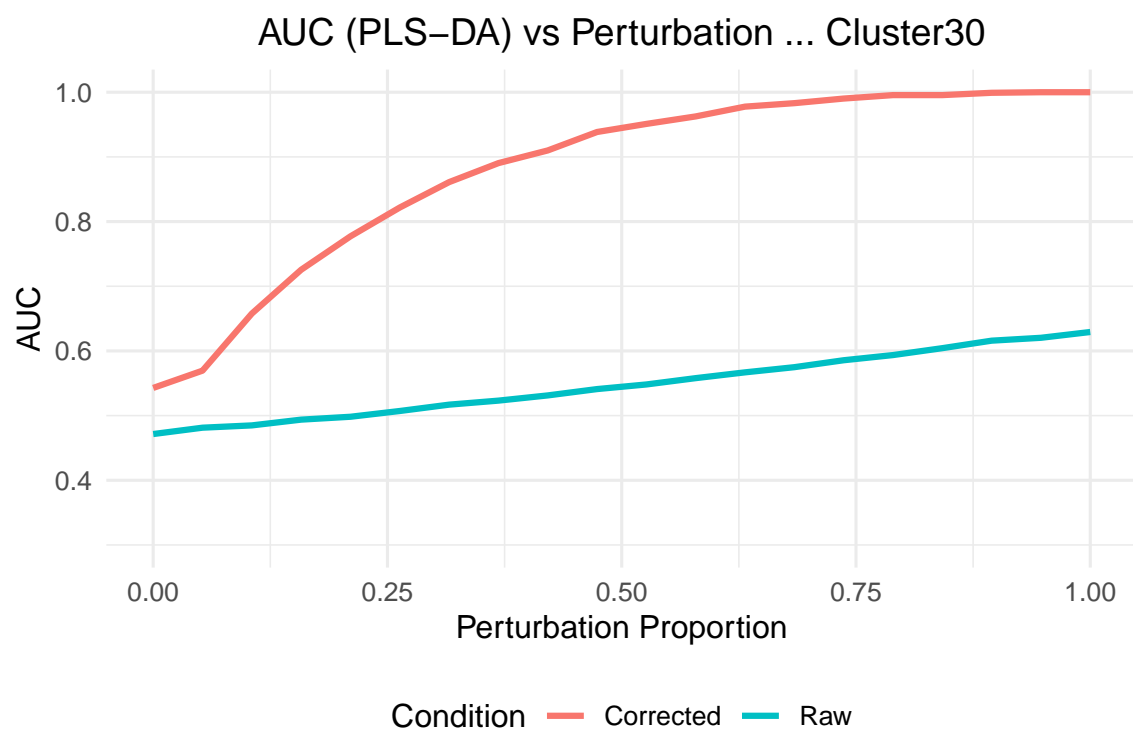
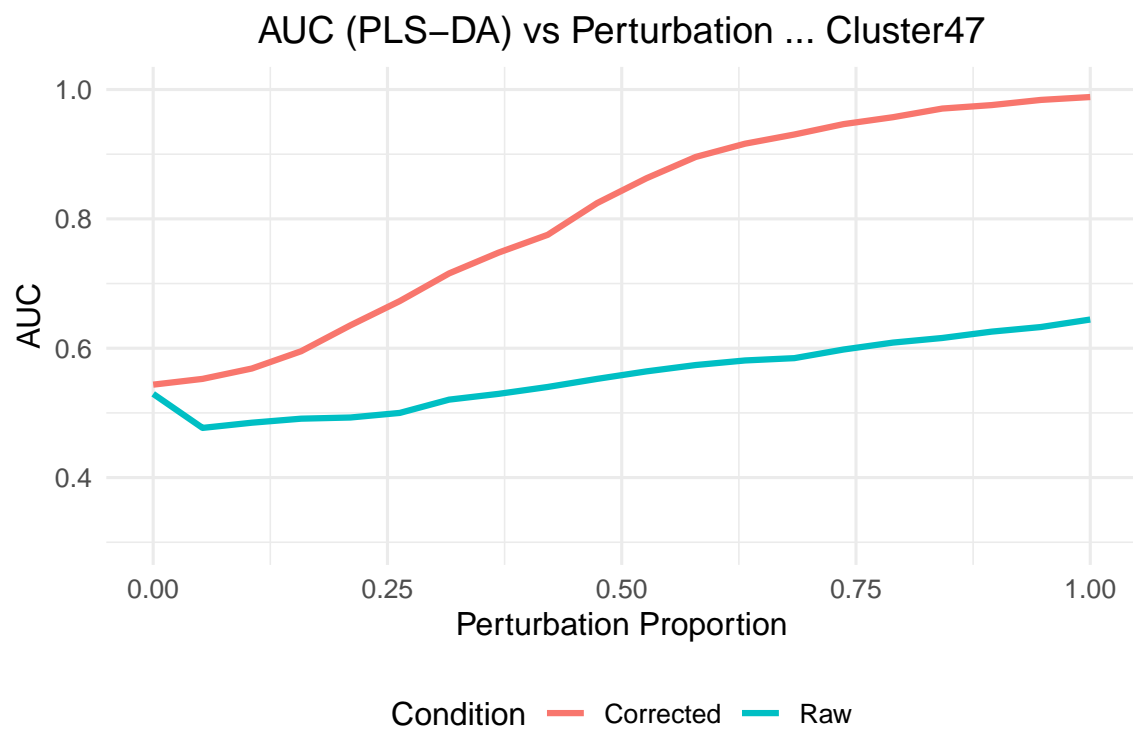


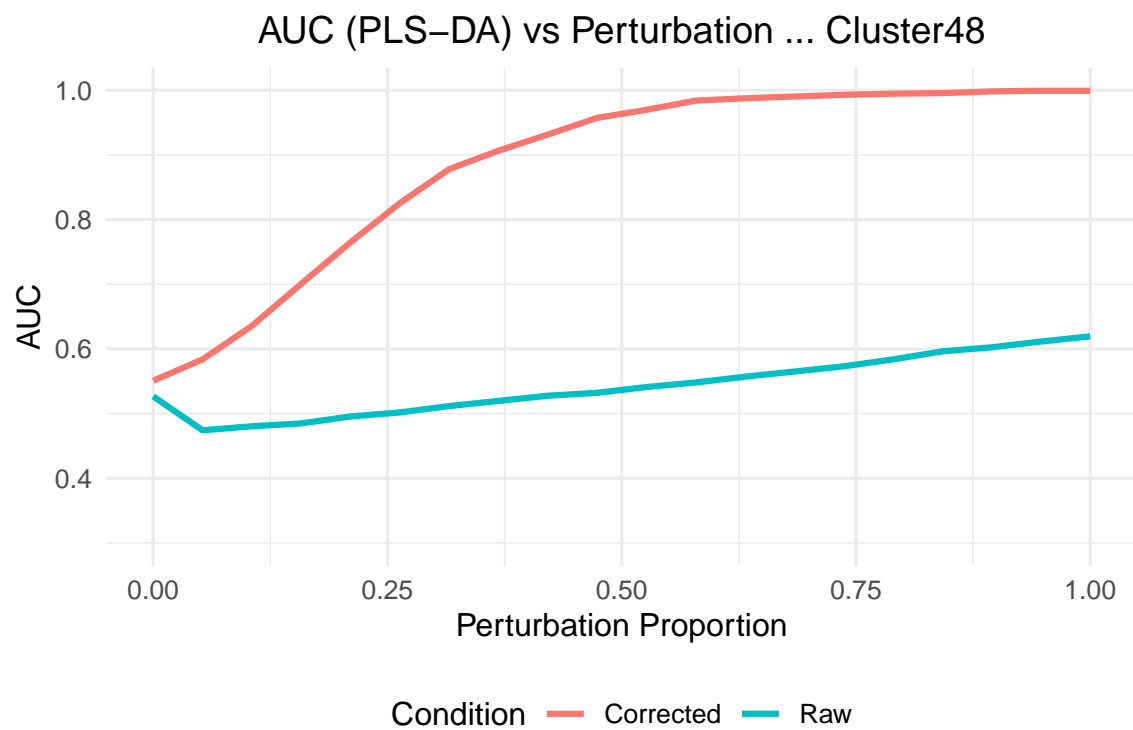
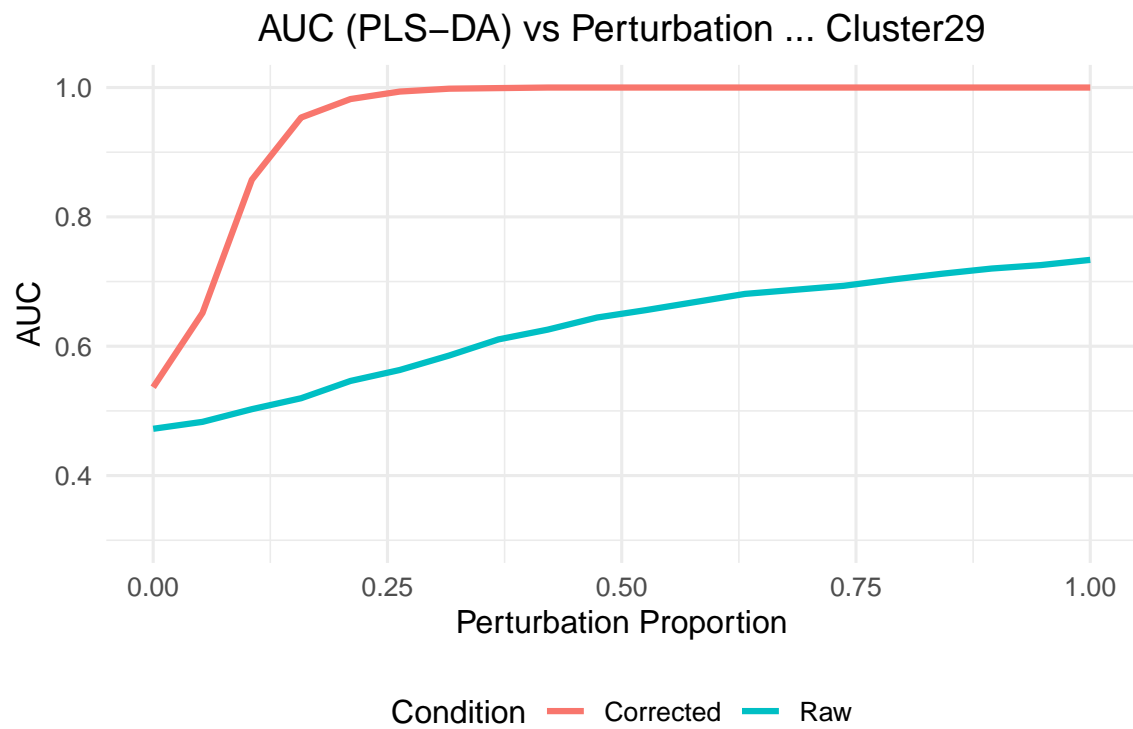


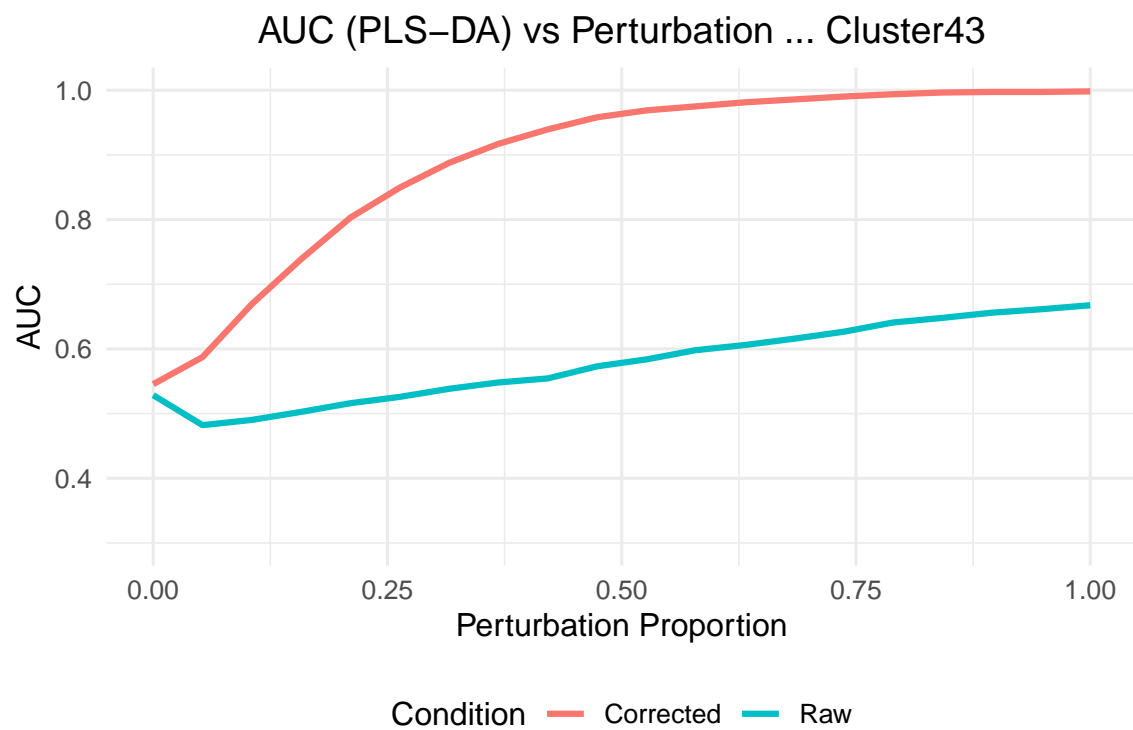
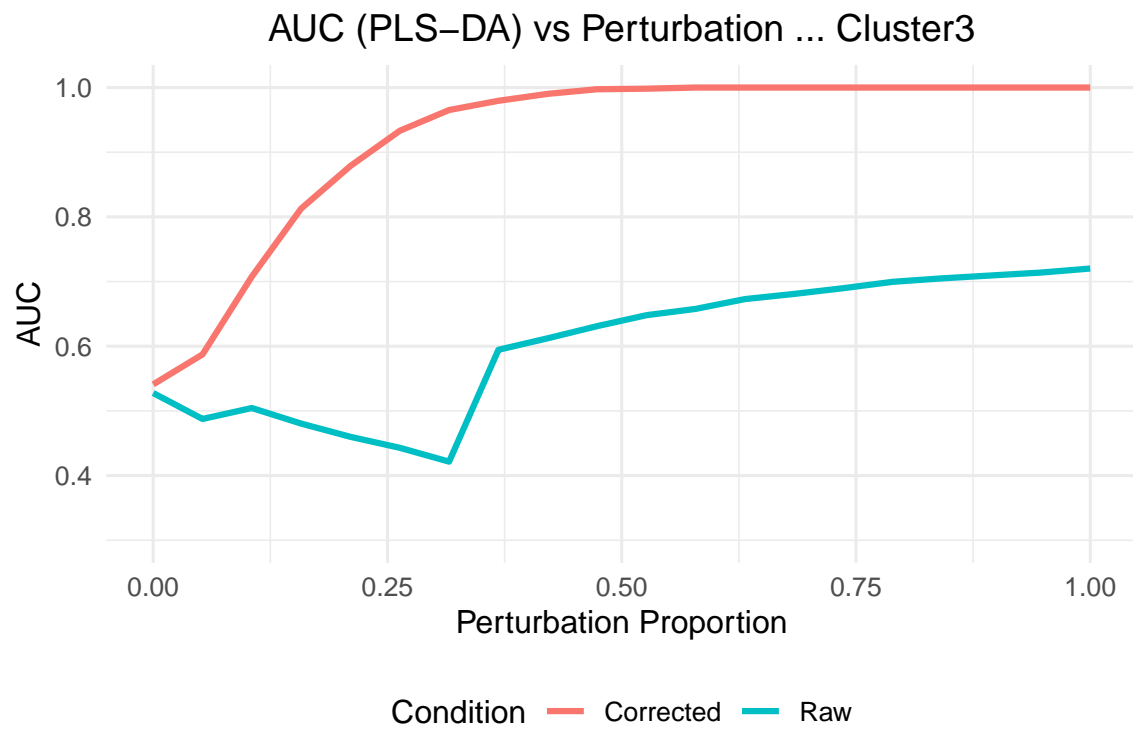


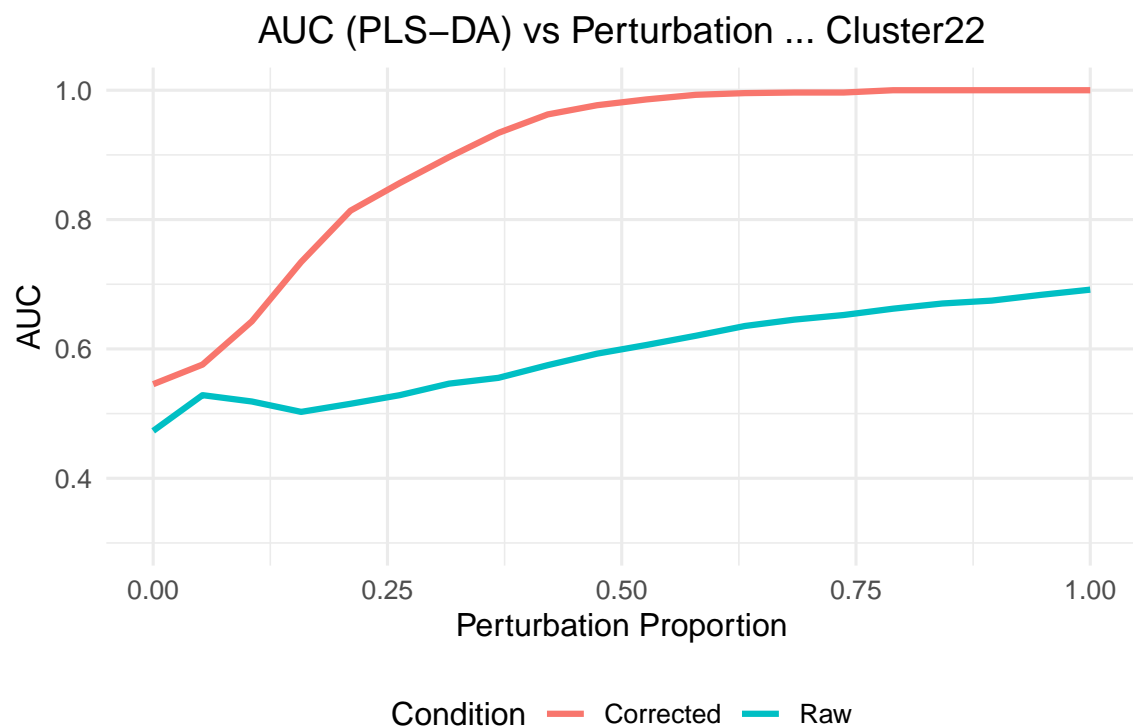
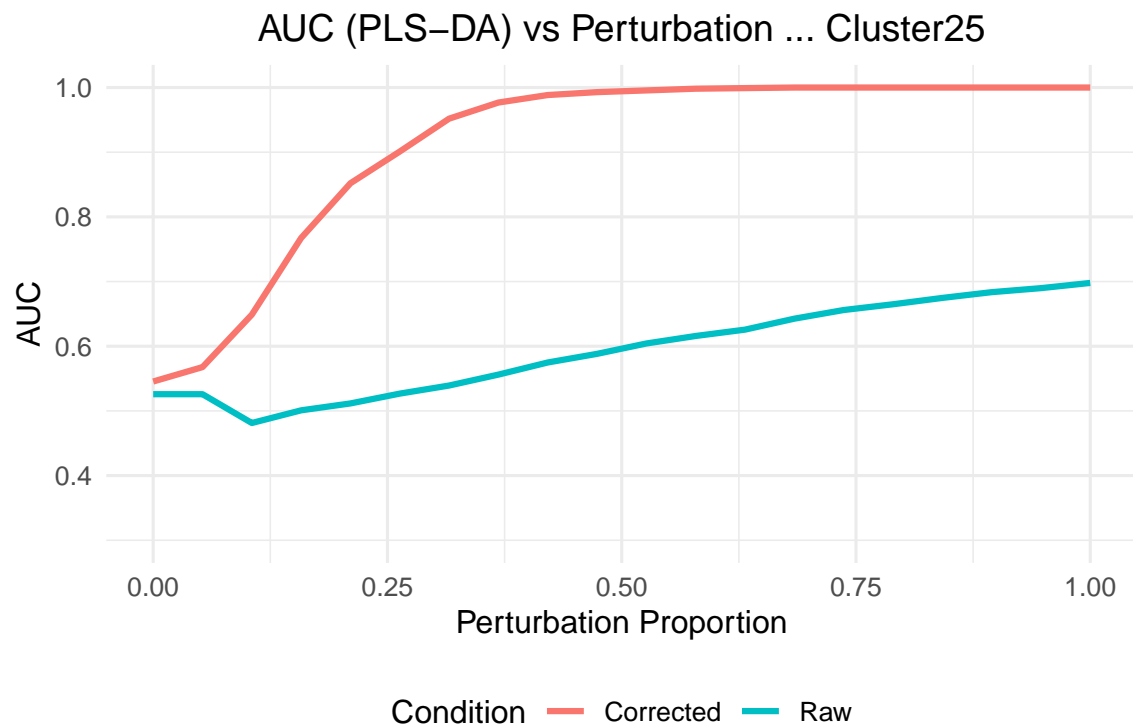


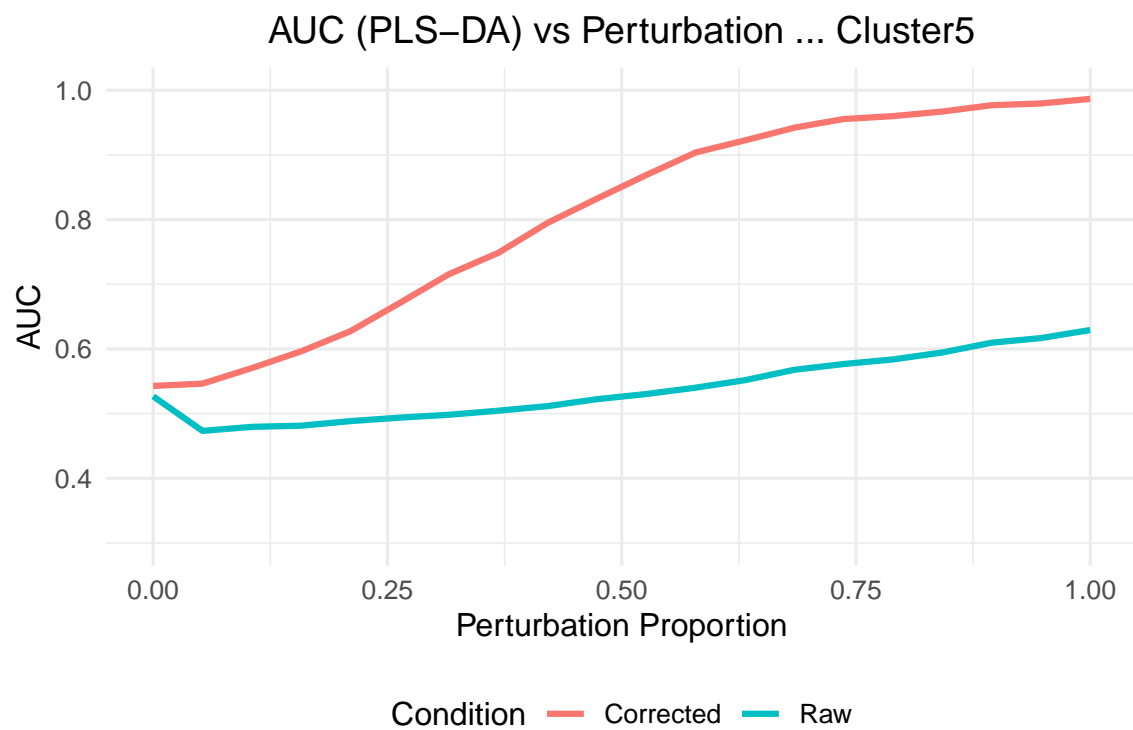
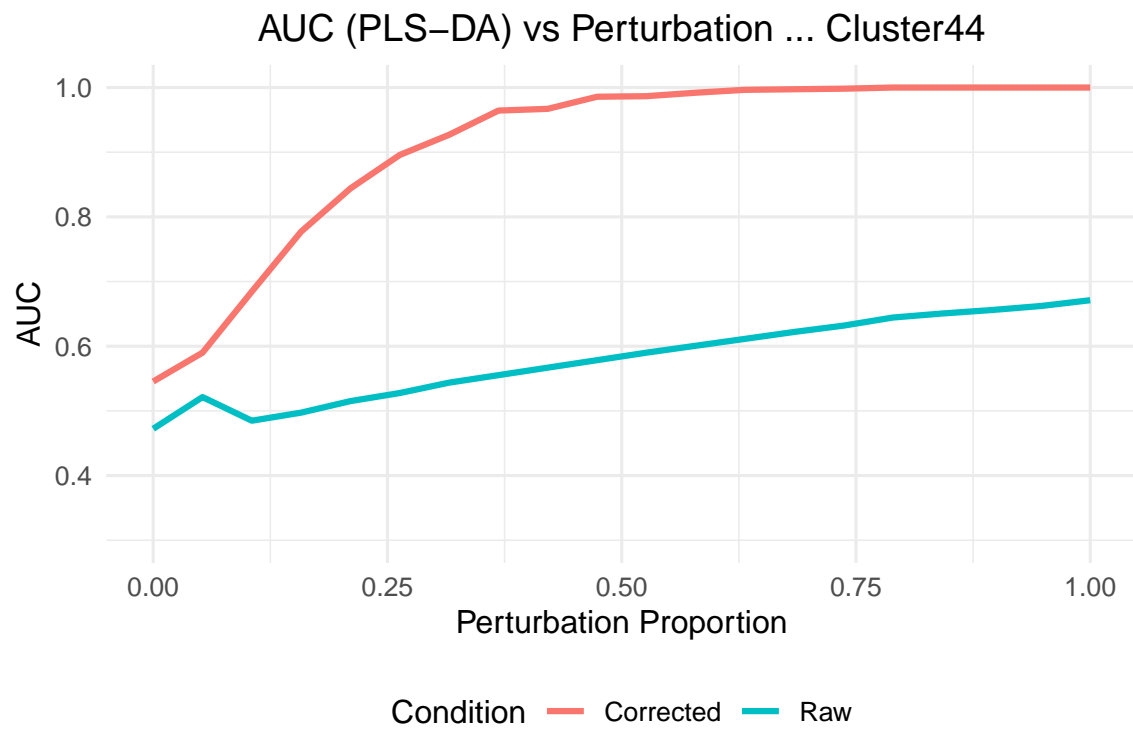


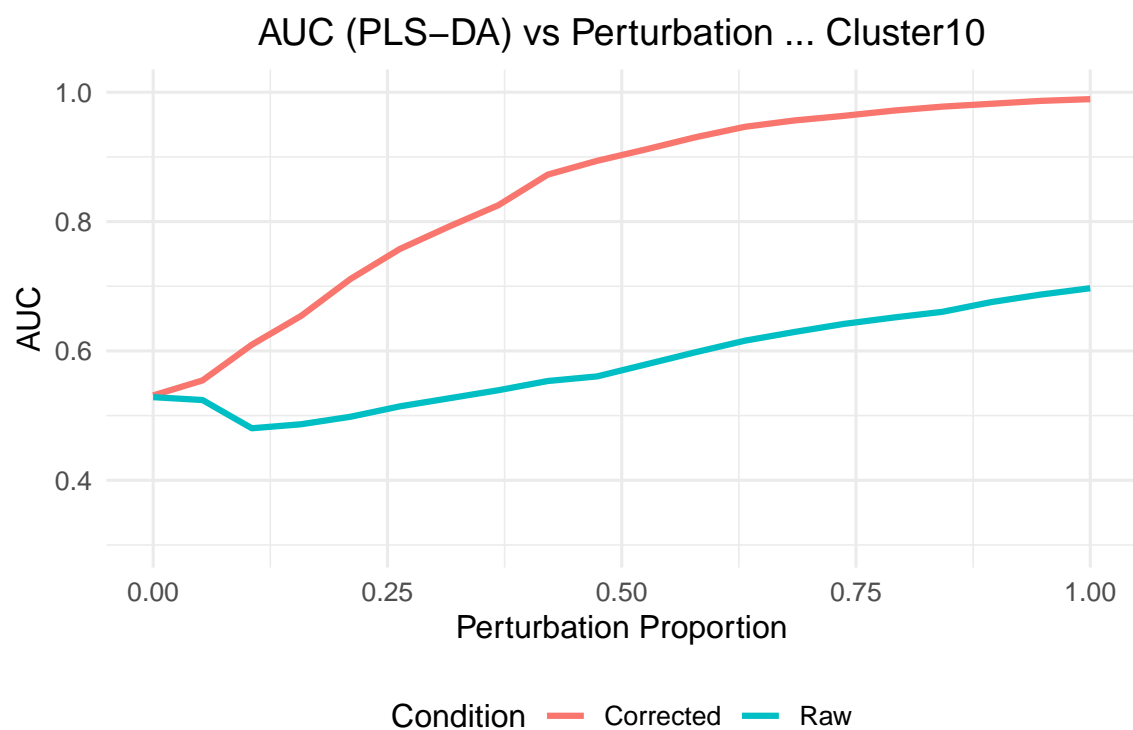
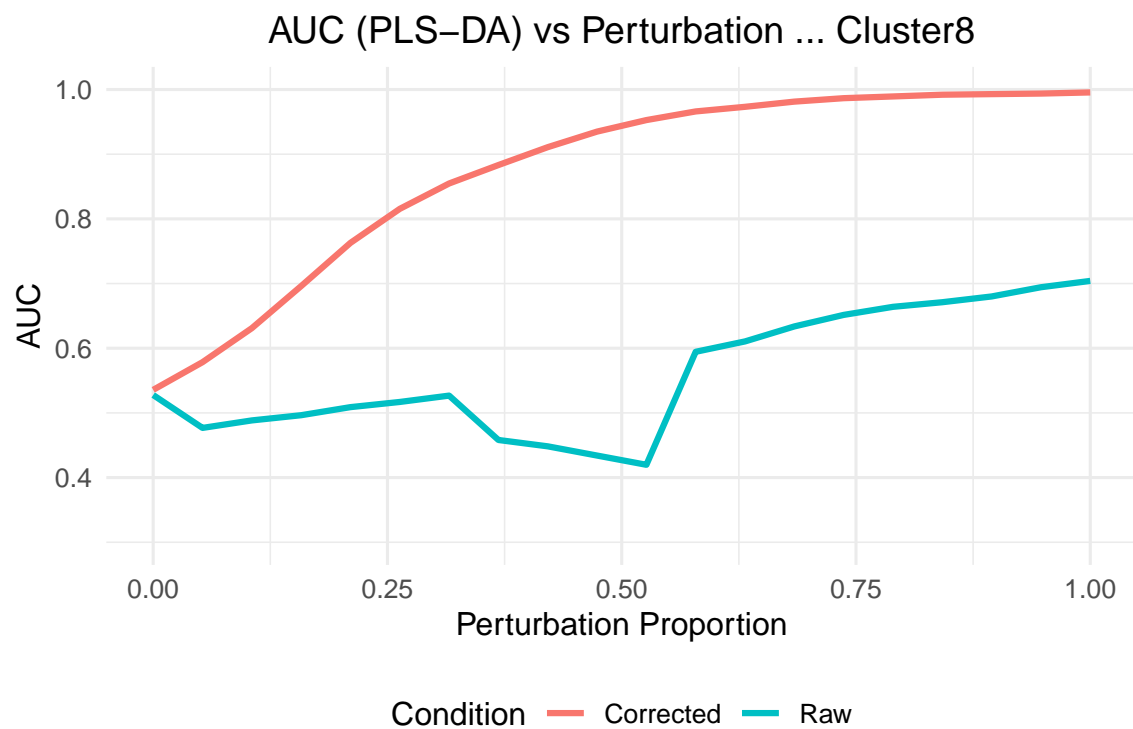


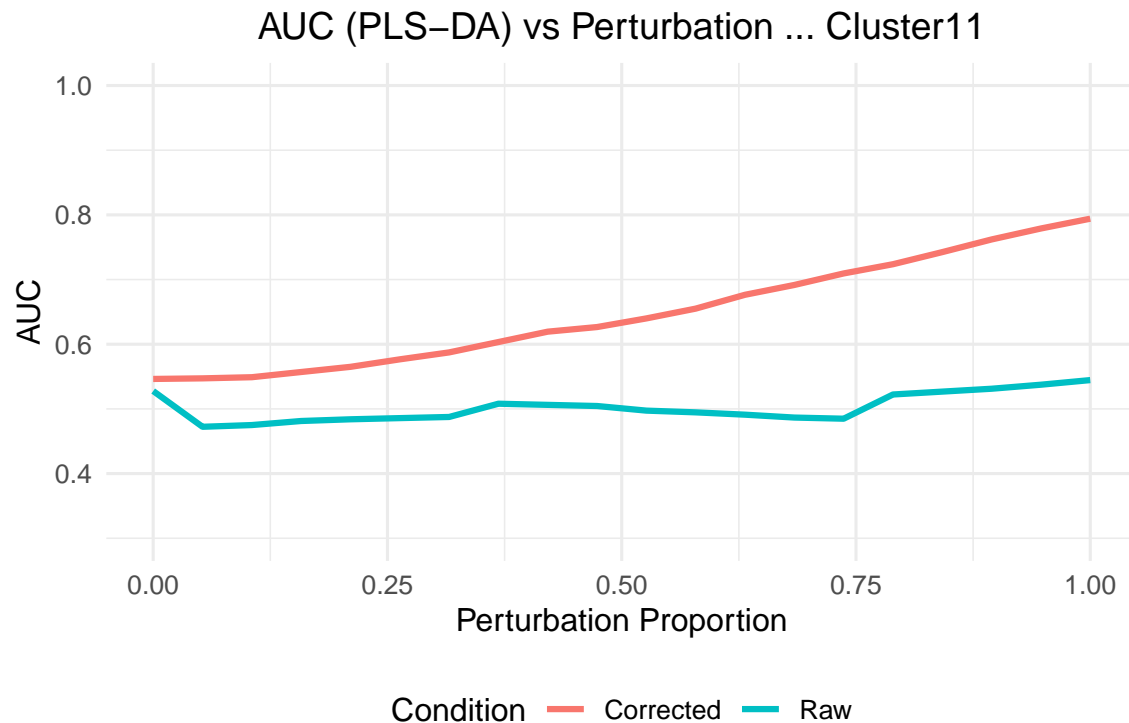












6. Robustness Check

Repeated Random Assignments

To evaluate the stability of the observed AUC improvements, we repeat the group assignment multiple times with different random seeds. This controls for potential biases introduced by a single, lucky alignment between group labels and technical variables.

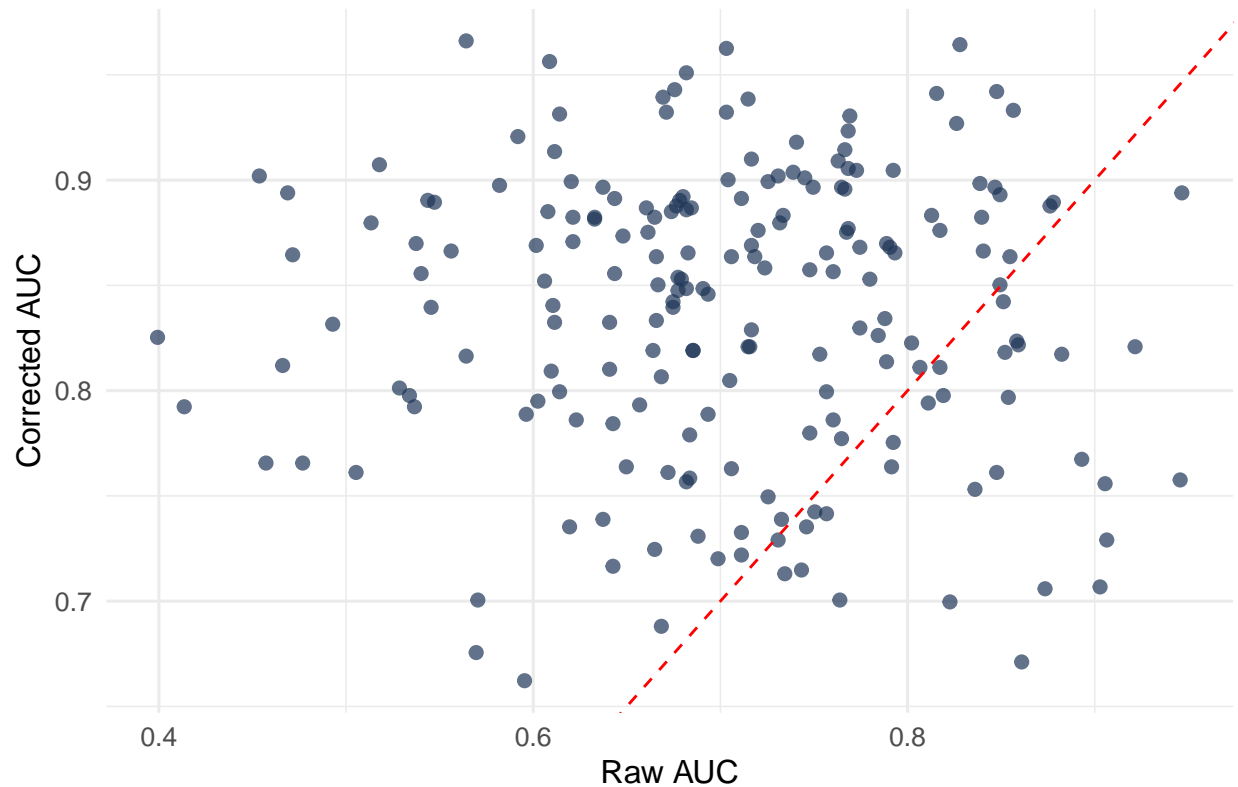
We compute the AUC with and without EPO correction across 200 repetitions for a single cluster (`Cluster14`) perturbation of 30% and visualize the distribution.

```
set.seed(NULL) # No seed fixation
n_repeats <- 200
target_cluster <- "Cluster14"

robust_results <- purrr::map_dfr(1:n_repeats, function(i) {
  df$Group <- sample(rep(c("Control", "Cancer"), length.out = nrow(df)))
  df_mod <- simulate_shift(df, cluster = target_cluster, perc = 0.25)
  aucs <- evaluate_auc(df_mod, meta)
  tibble::tibble(Iteration = i, Raw = aucs["Raw"], Corrected = aucs["Corrected"])
})

ggplot(robust_results, aes(x = Raw, y = Corrected)) +
  geom_point(color = "#1D3557", alpha = 0.7, size = 2) +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "red") +
  theme_minimal(base_size = 12) +
  labs(title = paste("Raw vs Corrected AUCs across 50 runs (", target_cluster, ")"),
       x = "Raw AUC", y = "Corrected AUC")
```

Raw vs Corrected AUCs across 50 runs (Cluster14)



Global Validation Across Clusters and Perturbation Levels

To confirm that the improvement provided by EPO correction is consistent across clusters and perturbation levels, we repeat the simulation for all clusters and multiple perturbation levels, with repeated random group assignments.

Run Simulations

```
perturb_seq <- seq(0, 1, by = 0.05)
n_repeats <- 10
results_global <- purrr::map_dfr(clusters, function(cl) {
  purrr::map_dfr(perturb_seq, function(p) {
    purrr::map_dfr(1:n_repeats, function(i) {
      df$Group <- sample(rep(c("Control", "Cancer"), length.out = nrow(df)))
      df_mod <- simulate_shift(df, cluster = cl, perc = p)
      aucs <- evaluate_auc(df_mod, meta)
      tibble::tibble(Cluster = cl, Perturbation = p, Rep = i,
                     Raw = aucs["Raw"], Corrected = aucs["Corrected"])
    })
  })
})
```

Summary Statistics by Perturbation Level

```
summary_global <- results_global %>%
  dplyr::group_by(Perturbation) %>%
  dplyr::summarise(
    p_value = wilcox.test(Corrected, Raw, paired = TRUE, alternative = "greater")$p.value,
    mean_gain = mean(Corrected - Raw),
    prop_improved = mean(Corrected > Raw)
  )

summary_global
```

```
## # A tibble: 21 x 4
##   Perturbation p_value mean_gain prop_improved
##         <dbl>   <dbl>     <dbl>         <dbl>
## 1           0 1.10e- 1 0.00501         0.561
## 2          0.05 7.96e- 5 0.0174         0.590
## 3          0.1 1.37e-15 0.0567         0.690
## 4          0.15 2.25e-27 0.0902         0.758
## 5          0.2 3.12e-30 0.104          0.765
## 6          0.25 7.58e-31 0.105          0.781
## 7          0.3 7.36e-33 0.112          0.8
## 8          0.35 8.20e-32 0.0996         0.794
## 9          0.4 7.19e-31 0.0926         0.790
## 10         0.45 6.97e-27 0.0834         0.719
## # i 11 more rows
```

7. Latent Structure Visualisation Before and After Correction

To visualize the effect of EPO correction on the latent structure of the data, we project training and test samples in PCA and PLS-DA space, before and after correction. We simulate a 25% shift in `Cluster24` for the “Cancer” group, using stratified splitting consistent with earlier AUC evaluations.

```
library(mixOmics)
library(pROC)

# Prepare data
df_mod <- simulate_shift(df, cluster = "Cluster24", perc = 0.25)
cluster_cols <- grep("^Cluster", names(df_mod), value = TRUE)
X_raw <- as.matrix(df_mod[, cluster_cols])
y <- as.factor(df_mod$Group)
classes <- levels(y)

# Stratified split
set.seed(42)
idx_class1 <- which(y == classes[1])
idx_class2 <- which(y == classes[2])
test_class1 <- sample(idx_class1, floor(length(idx_class1) * 0.5))
test_class2 <- sample(idx_class2, floor(length(idx_class2) * 0.5))
test_idx <- c(test_class1, test_class2)
train_idx <- setdiff(seq_along(y), test_idx)

X_train_raw <- X_raw[train_idx, , drop = FALSE]
X_test_raw <- X_raw[test_idx, , drop = FALSE]
```

```

y_train <- y[train_idx]
y_test <- y[test_idx]

# PLS-DA Raw
pls_raw <- plsda(X_train_raw, y_train, ncomp = 2)
pred_raw <- predict(pls_raw, X_test_raw, dist = "max.dist")$predict[, , 2]
prob_raw <- pred_raw[, classes[2]]
roc_raw <- roc(response = y_test, predictor = prob_raw, levels = classes)
auc_raw <- auc(roc_raw)

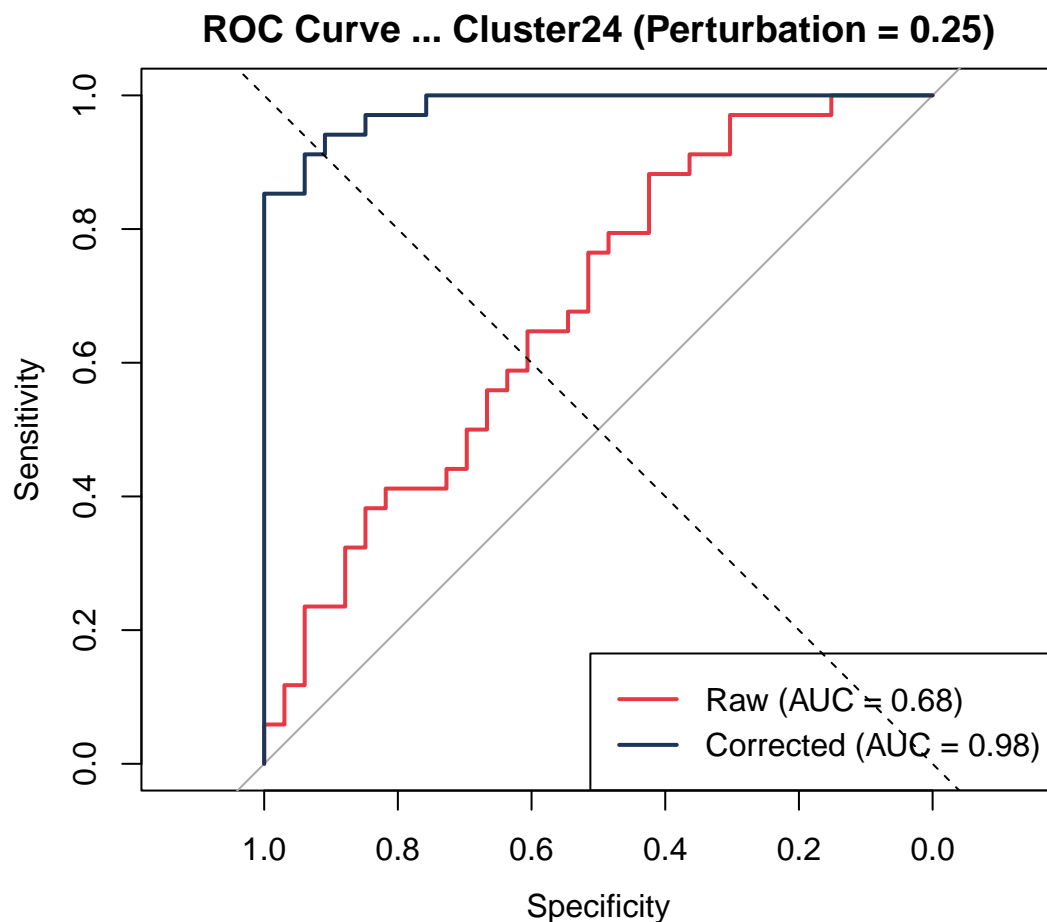
# EPO correction
X_corr <- correction(correction(X_raw, meta$elapsed_time)$corrected,
                    meta$batch)$corrected

X_train_corr <- X_corr[train_idx, , drop = FALSE]
X_test_corr <- X_corr[test_idx, , drop = FALSE]

# PLS-DA Corrected
pls_corr <- plsda(X_train_corr, y_train, ncomp = 2)
pred_corr <- predict(pls_corr, X_test_corr, dist = "max.dist")$predict[, , 2]
prob_corr <- pred_corr[, classes[2]]
roc_corr <- roc(response = y_test, predictor = prob_corr, levels = classes)
auc_corr <- auc(roc_corr)

# ROC plot
plot(roc_raw, col = "#E63946", lwd = 2, main = "ROC Curve - Cluster24 (Perturbation = 0.25)")
plot(roc_corr, col = "#1D3557", lwd = 2, add = TRUE)
abline(a = 0, b = 1, lty = 2)
legend("bottomright",
      legend = c(sprintf("Raw (AUC = %.2f)", auc_raw),
                  sprintf("Corrected (AUC = %.2f)", auc_corr)),
      col = c("#E63946", "#1D3557"), lwd = 2)

```



7.1 Latent Space Projection — Raw

```
# Train
latent_scores_train_raw <- as.data.frame(pls_raw$variates$X)
colnames(latent_scores_train_raw) <- c("LV1", "LV2")
latent_scores_train_raw$Group <- factor(as.character(y_train))
latent_scores_train_raw$Split <- "Train"

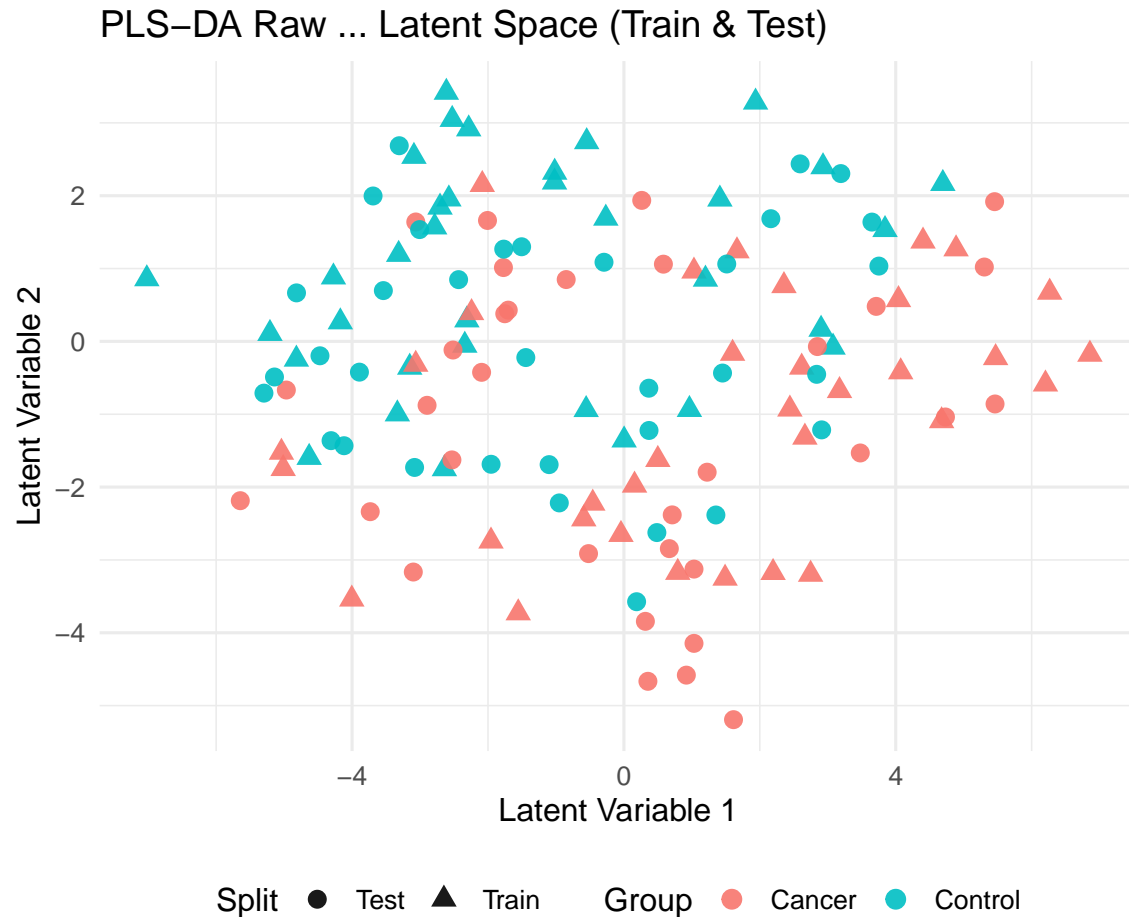
# Test
pred_raw <- predict(pls_raw, newdata = X_test_raw, dist = "max.dist")
latent_scores_test_raw <- as.data.frame(pred_raw$variates)
colnames(latent_scores_test_raw) <- c("LV1", "LV2")
latent_scores_test_raw$Group <- factor(as.character(y_test))
latent_scores_test_raw$Split <- "Test"

# Combine
latent_scores_raw <- rbind(latent_scores_train_raw, latent_scores_test_raw)

# Plot
ggplot(latent_scores_raw, aes(x = LV1, y = LV2, color = Group, shape = Split)) +
  geom_point(size = 3, alpha = 0.9) +
```



```
labs(title = "PLS-DA Raw - Latent Space (Train & Test)",
     x = "Latent Variable 1", y = "Latent Variable 2") +
theme_minimal(base_size = 12) +
theme(legend.position = "bottom")
```



7.2 Latent Space Projection — Corrected

```
# Train
latent_scores_train <- as.data.frame(pls_corr$variates$X)
colnames(latent_scores_train) <- c("LV1", "LV2")
latent_scores_train$Group <- factor(as.character(y_train))
latent_scores_train$Split <- "Train"

# Test
pred_corr <- predict(pls_corr, newdata = X_test_corr, dist = "max.dist")
latent_scores_test <- as.data.frame(pred_corr$variates)
colnames(latent_scores_test) <- c("LV1", "LV2")
latent_scores_test$Group <- factor(as.character(y_test))
latent_scores_test$Split <- "Test"

# Combine
```

```
latent_scores <- rbind(latent_scores_train, latent_scores_test)

# Plot
ggplot(latent_scores, aes(x = LV1, y = LV2, color = Group, shape = Split)) +
  geom_point(size = 3, alpha = 0.9) +
  labs(title = "PLS-DA Corrected - Latent Space (Train & Test)",
       x = "Latent Variable 1", y = "Latent Variable 2") +
  theme_minimal(base_size = 12) +
  theme(legend.position = "bottom")
```

