Normalization Test

Jing Wu

2024-06-17

Instrution

The report evaluates the efficacy of various normalization methods in mitigating batch effects. The raw data, accessible from the Gene Expression Omnibus (GEO)https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi under accession number GSE141549, can be retrieved from NCBI's GEO database. For detailed information on data preprocessing, please refer to the comment sheet accompanying the raw data.

The data set encompasses measurements from two distinct genes. To illustrate the impact of the normalization techniques, principal component analysis (PCA) and Volcano plots are employed as the primary visualization strategies. Furthermore, the statistical significance of the findings is determined using the Wilcoxon test, a non-parametric alternative to the paired t-test.

Visualization for raw data distribution

```
# normalization method test
# get the sample id
sample_id <- read.xlsx("/Users/willowwu/Downloads/GSE141549_data for comparing normalizations.xlsx")
sample_id_list <- as.character(sample_id[2,2:dim(sample_id)[2]])

# remove na value in sample_id_list
sample_id_list <- sample_id_list[!is.na(sample_id_list)]

# match the data
WG6 <- read.xlsx("/Users/willowwu/Downloads/lab/GSE141549_non-normalizeWG-6_genesymbol.xlsx")
HT12 <- read.xlsx("/Users/willowwu/Downloads/lab/GSE141549_non-normalizeHT-12.xlsx")

WG6_sample <- colnames(WG6)
HT12_sample <- colnames(HT12)

# print the match result
match(sample_id_list, WG6_sample)
```

```
##  [1]   7   3  NA  NA  NA  23  28  NA  NA  NA  34  33  37  36  49  47  52  NA  59
## [20]  57  69  67  74  NA  77  76 106 104 119 118 127 126 131 130 137 135 141 138
## [39] 144 143 149 147 152 151 159 158 162 161 173 171 181 179 190 188 198  NA 199
## [58]  NA 201 200 206 205 211  NA 215  NA 246 245 264  NA 270  NA 276  NA 285 284
## [77] 296 295
```

```r
match(sample_id_list, HT12_sample)
```

```
##  [1] NA NA  4  3 11 NA NA 14 21 20 NA NA NA NA NA NA NA 28 NA NA NA NA NA 33 NA
## [26] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [51] NA NA NA NA NA 49 NA 50 NA NA NA NA NA 53 NA 54 NA NA NA 61 NA 62 NA 63 NA
## [76] NA NA NA
```

```r
# check the number of matching id
sum(!is.na(match(sample_id_list, WG6_sample)))
```

```
## [1] 63
```

```r
sum(!is.na(match(sample_id_list, HT12_sample)))
```

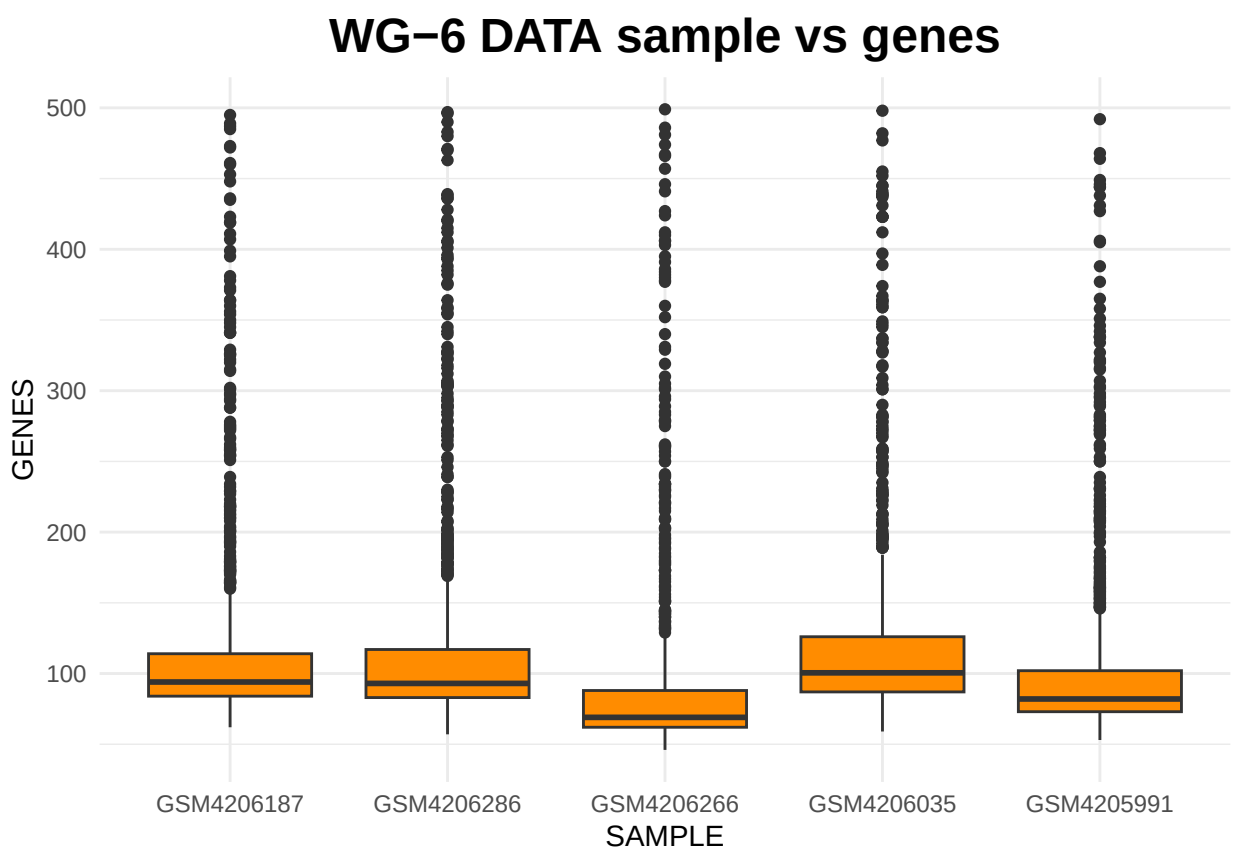
```
## [1] 15
```

```r
# generate summary and distribution for 2 raw data
num_HT12 <- HT12[8:dim(HT12)[1], 3:dim(HT12)[2]]
num_WG6 <- WG6[8:dim(WG6)[1], 3:dim(WG6)[2]]

# WG6
# convert the data into numeric
num_WG6 <- apply(num_WG6, 2, as.numeric)

# randomly select 5 samples and 1000 genes to see the distribution
set.seed(123)
selected_samples <- sample(dim(num_WG6)[2], 5)
selected_genes <- sample(dim(num_WG6)[1], 1000)

# filter the data
filtered_data <- num_WG6[c(selected_genes), c(selected_samples)]
filtered_data <- as.data.table(filtered_data)
data_long <- melt(filtered_data)
```

```
## Warning in melt.data.table(filtered_data): id.vars and measure.vars are
## internally guessed when both are 'NULL'. All non-numeric/integer/logical type
## columns are considered id.vars, which in this case are columns []. Consider
## providing at least one of 'id' or 'measure' vars in future.
```

```r
colnames(data_long) <- c("Sample", "Value")

data_long %>%
  filter(Value >= 0L & Value <= 500L | is.na(Value)) %>%
  ggplot() +
  aes(x = Sample, y = Value) +
  geom_boxplot(fill = "#FF8C00") +
  labs(
    x = "SAMPLE",
    y = "GENES",
    title = "WG-6 DATA sample vs genes"
```

```
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(size = 18L,
                              face = "bold",
                              hjust = 0.5)
  )
```

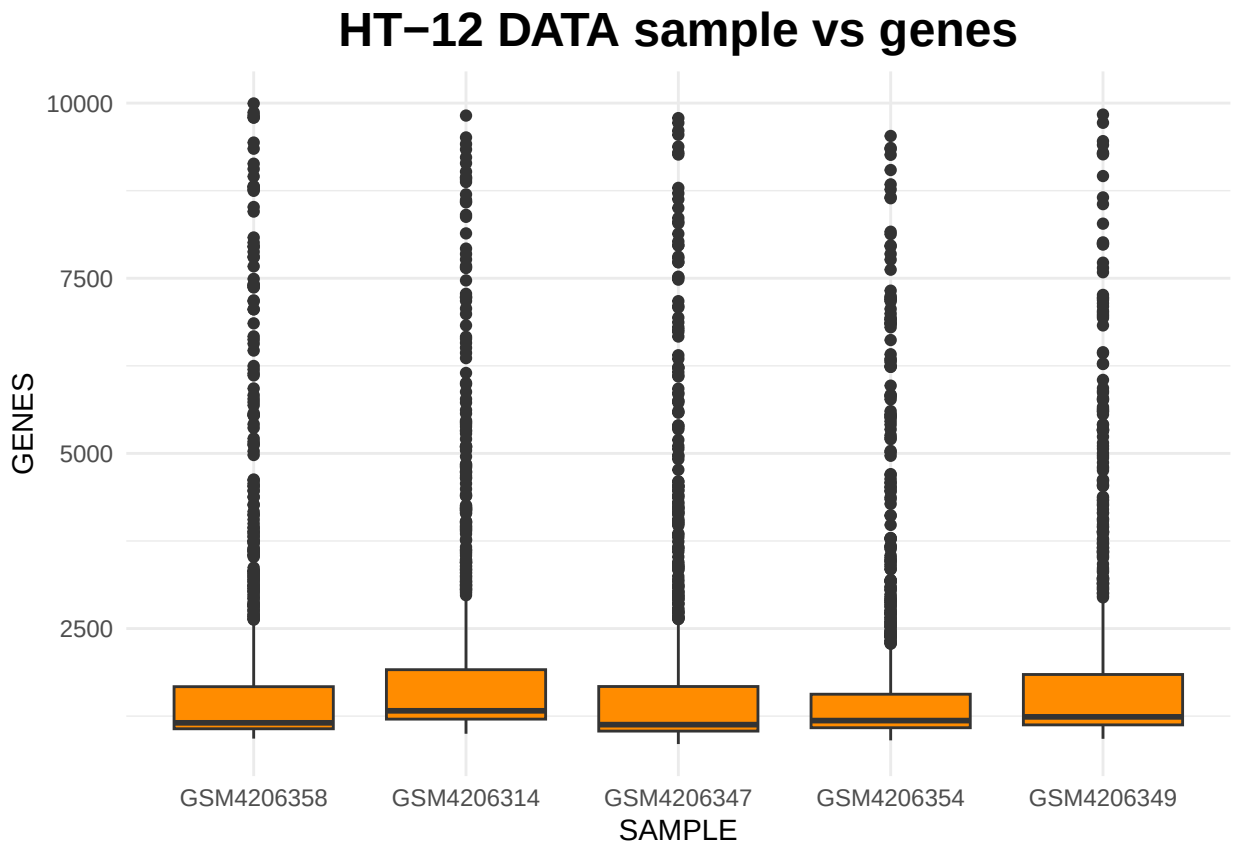



```
# HT12
# convert the data into numeric
num_HT12 <- apply(num_HT12, 2, as.numeric)

# randomly select 5 samples and 1000 genes to see the distribution
set.seed(123)
selected_samples <- sample(dim(num_HT12)[2], 5)
selected_genes <- sample(dim(num_HT12)[1], 1000)

# filter the data
filtered_data <- num_HT12[c(selected_genes), c(selected_samples)]
filtered_data <- as.data.table(filtered_data)
data_long <- melt(filtered_data)
```

```
## Warning in melt.data.table(filtered_data): id.vars and measure.vars are
## internally guessed when both are 'NULL'. All non-numeric/integer/logical type
## columns are considered id.vars, which in this case are columns []. Consider
```

```
## providing at least one of 'id' or 'measure' vars in future.
```

```r
colnames(data_long) <- c("Sample", "Value")

data_long %>%
  filter(Value >= 0L & Value <= 10000L | is.na(Value)) %>%
  ggplot() +
  aes(x = Sample, y = Value) +
  geom_boxplot(fill = "#FF8C00") +
  labs(
    x = "SAMPLE",
    y = "GENES",
    title = "HT-12 DATA sample vs genes"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(size = 18L,
                              face = "bold",
                              hjust = 0.5)
  )
```

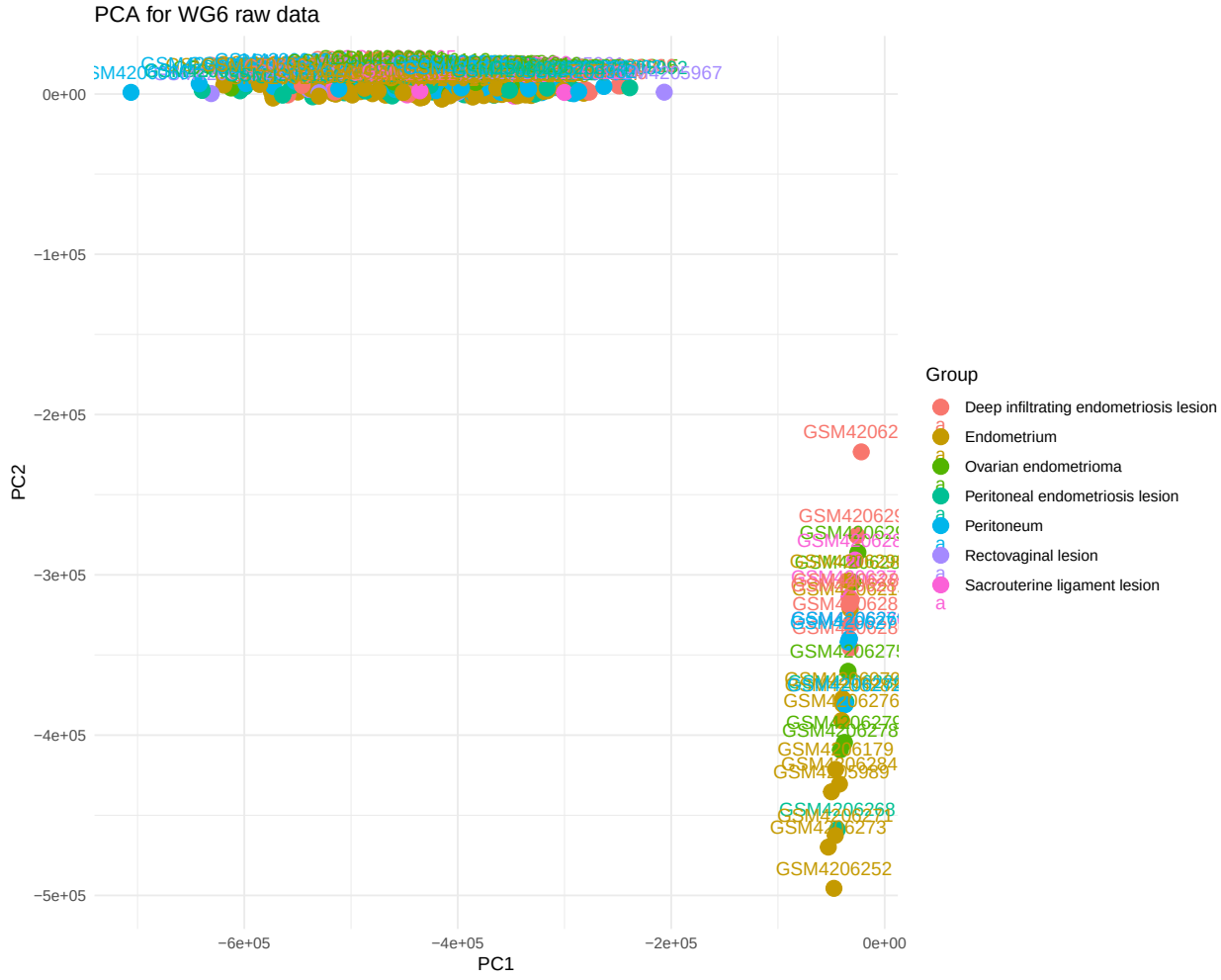


To further illustrate the distribution of the two raw datasets, we generate boxplots. The results reveal significant differences in distribution, particularly in terms of range, between the datasets. This disparity suggests the need for normalization prior to their integration.

To demonstrate the batch effect present in the raw data, we conduct principal component analysis (PCA) and produce corresponding plots. The normalization techniques employed include Z-score and Min-Max

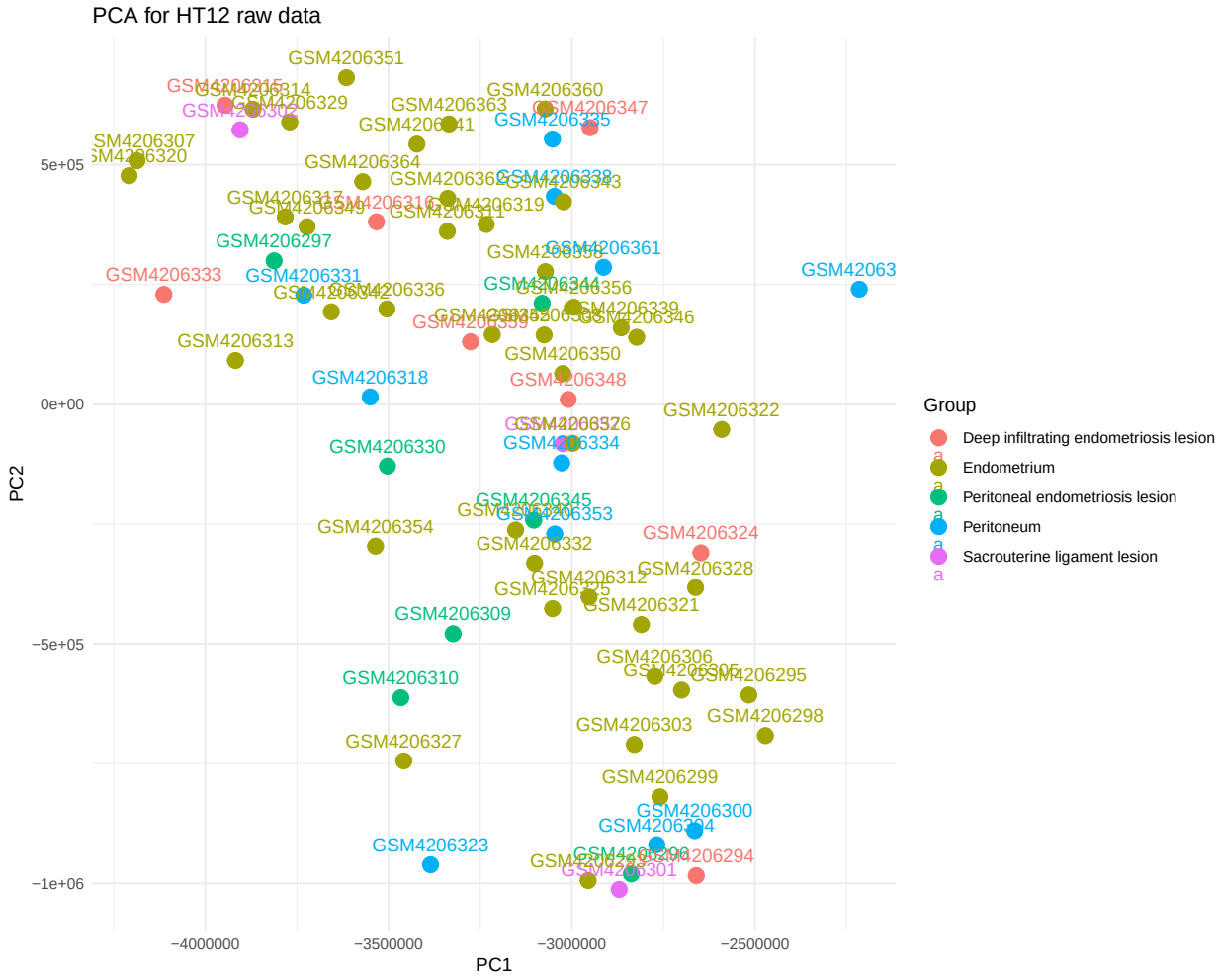
normalization. Importantly, these normalizations are applied horizontally, indicating that they are performed on a gene-based level.

```
# PCA for WG-6 raw data
group_labels_WG6 <- as.character(WG6[2, 3:dim(WG6)[2]])
group_labels_WG6 <- gsub(" Replicate", "", group_labels_WG6)
num_WG6[is.na(num_WG6)] <- 0
# In PCA, every row is a sample and every columns is a feature(variable)
pca_result <- prcomp(t(num_WG6), center = FALSE, scale. = FALSE)
pca_data <- data.frame(pca_result$x, Group = group_labels_WG6)
ggplot(pca_data, aes(x = PC1, y = PC2, color = Group, label = rownames(pca_data))) +
  geom_point(size = 4) +
  geom_text(vjust = -1) +
  labs(title = "PCA for WG6 raw data", x = "PC1", y = "PC2") +
  theme_minimal()
```




```
# PCA for HT-12 raw data
group_labels_HT12 <- as.character(HT12[2, 3:dim(HT12)[2]])
group_labels_HT12 <- gsub(" Replicate", "", group_labels_HT12)
num_HT12[is.na(num_HT12)] <- 0
# In PCA, every row is a sample and every columns is a feature(variable)
```

```r
pca_result <- prcomp(t(num_HT12), center = FALSE, scale. = FALSE)
pca_data <- data.frame(pca_result$x, Group = group_labels_HT12)
ggplot(pca_data, aes(x = PC1, y = PC2, color = Group, label = rownames(pca_data))) +
  geom_point(size = 4) +
  geom_text(vjust = -1) +
  labs(title = "PCA for HT12 raw data", x = "PC1", y = "PC2") +
  theme_minimal()
```

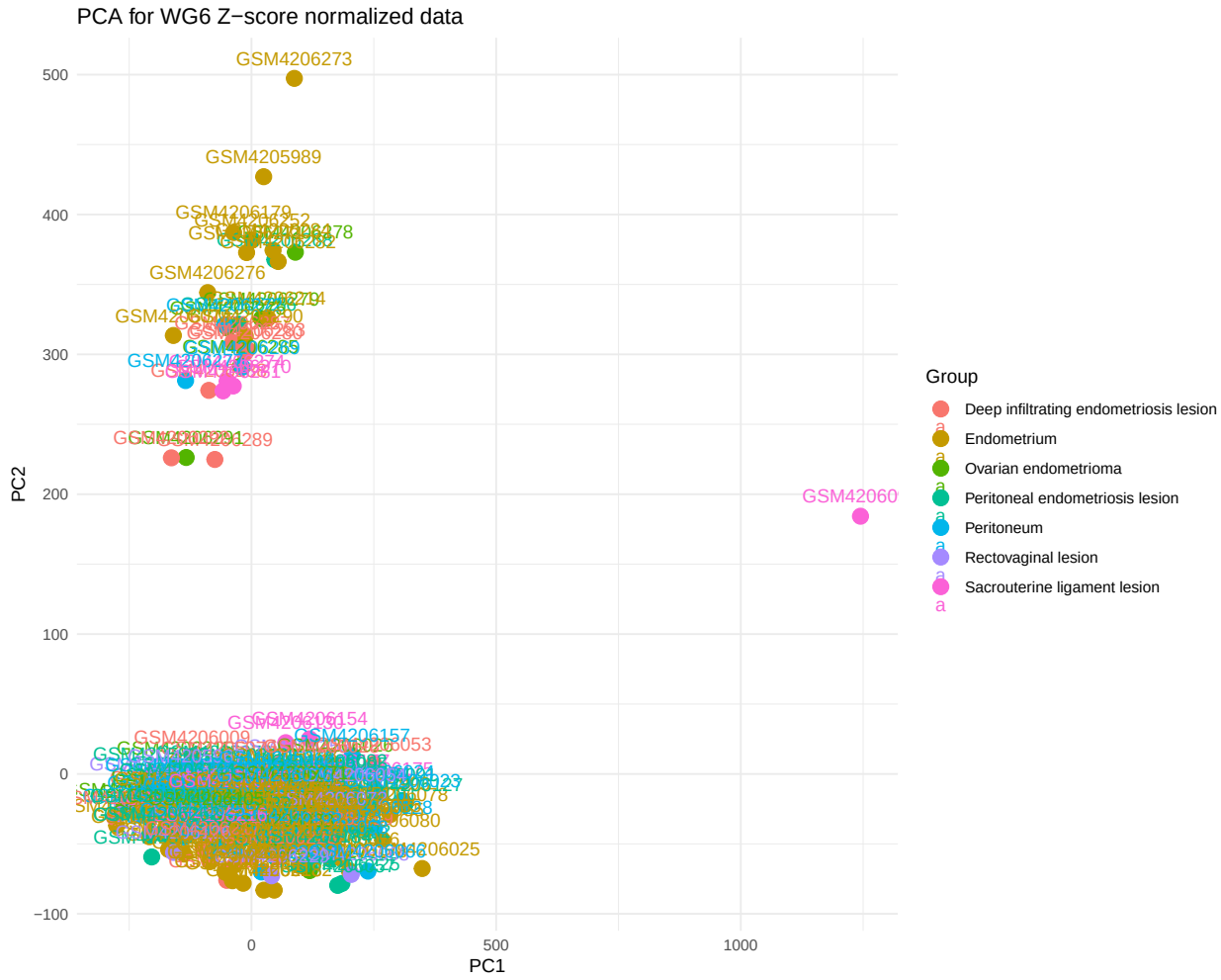


From the PCA results, it is evident that the raw data from the WG6 platform exhibit two distinct clusters, while the data from the HT12 platform does not show significant clusters.

PCA with Z-score normalization

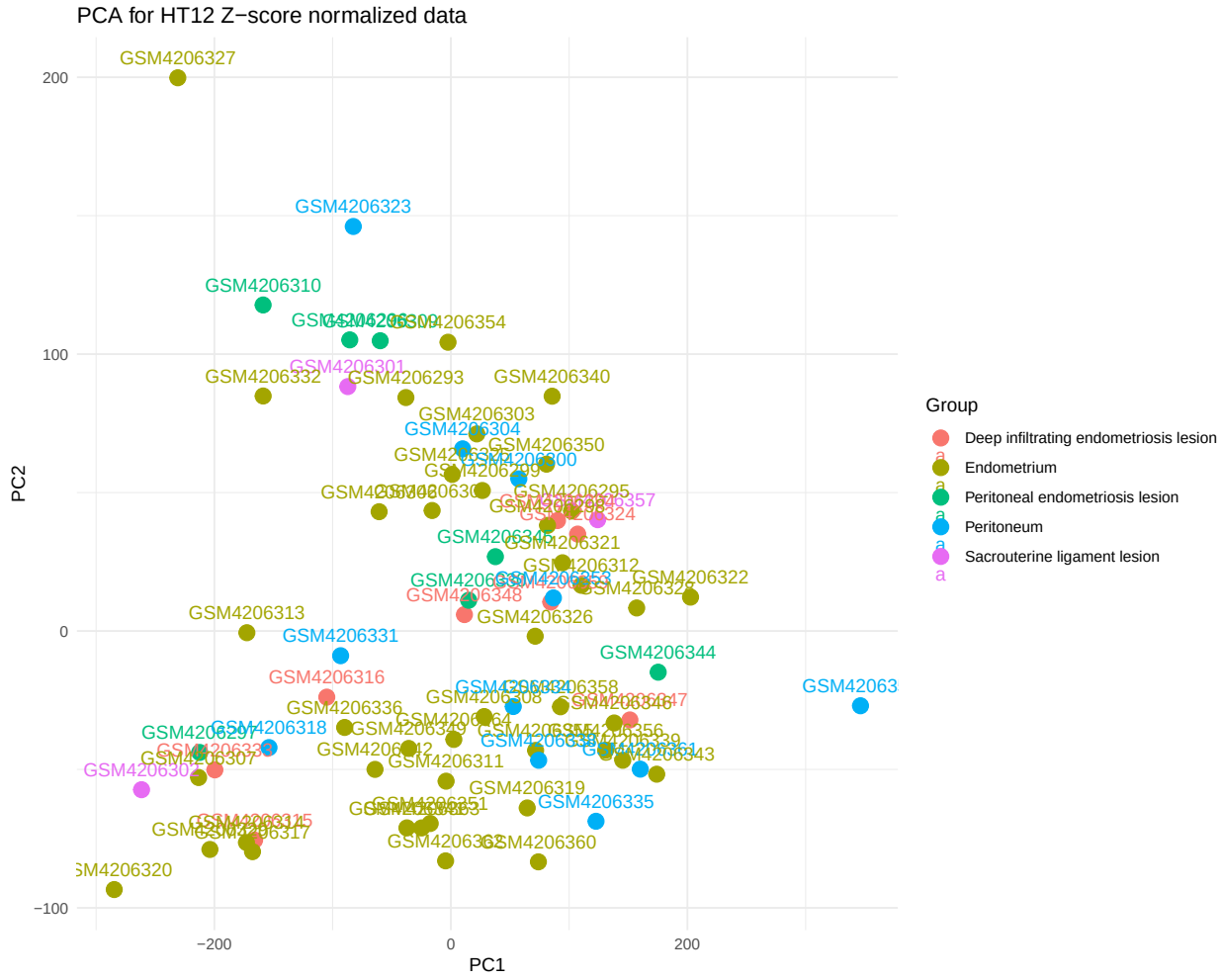
```r
# Normalize separately
# Z-score
# Function to normalize rows
normalize_row_Z <- function(row) {
  return((row - mean(row)) / sd(row))
}
```

```r
# Apply the normalization function to each row
num_WG6_norm <- t(apply(num_WG6, 1, normalize_row_Z))
num_HT12_norm <- t(apply(num_HT12, 1, normalize_row_Z))

# PCA for WG-6 normalized data
# remove all the NA values
num_WG6_norm[is.na(num_WG6_norm)] <- 0

# In PCA, every row is a sample and every columns is a feature(variable)
pca_result <- prcomp(t(num_WG6_norm), center = FALSE, scale. = FALSE)
pca_data <- data.frame(pca_result$x, Group = group_labels_WG6)
ggplot(pca_data, aes(x = PC1, y = PC2, color = Group, label = rownames(pca_data))) +
  geom_point(size = 4) +
  geom_text(vjust = -1) +
  labs(title = "PCA for WG6 Z-score normalized data", x = "PC1", y = "PC2") +
  theme_minimal()
```




```r
# PCA for HT-12 normalized data
num_HT12_norm[is.na(num_HT12_norm)] <- 0
# In PCA, every row is a sample and every columns is a feature(variable)
pca_result <- prcomp(t(num_HT12_norm), center = FALSE, scale. = FALSE)
```

```r
pca_data <- data.frame(pca_result$x, Group = group_labels_HT12)
ggplot(pca_data, aes(x = PC1, y = PC2, color = Group, label = rownames(pca_data))) +
  geom_point(size = 4) +
  geom_text(vjust = -1) +
  labs(title = "PCA for HT12 Z-score normalized data", x = "PC1", y = "PC2") +
  theme_minimal()
```

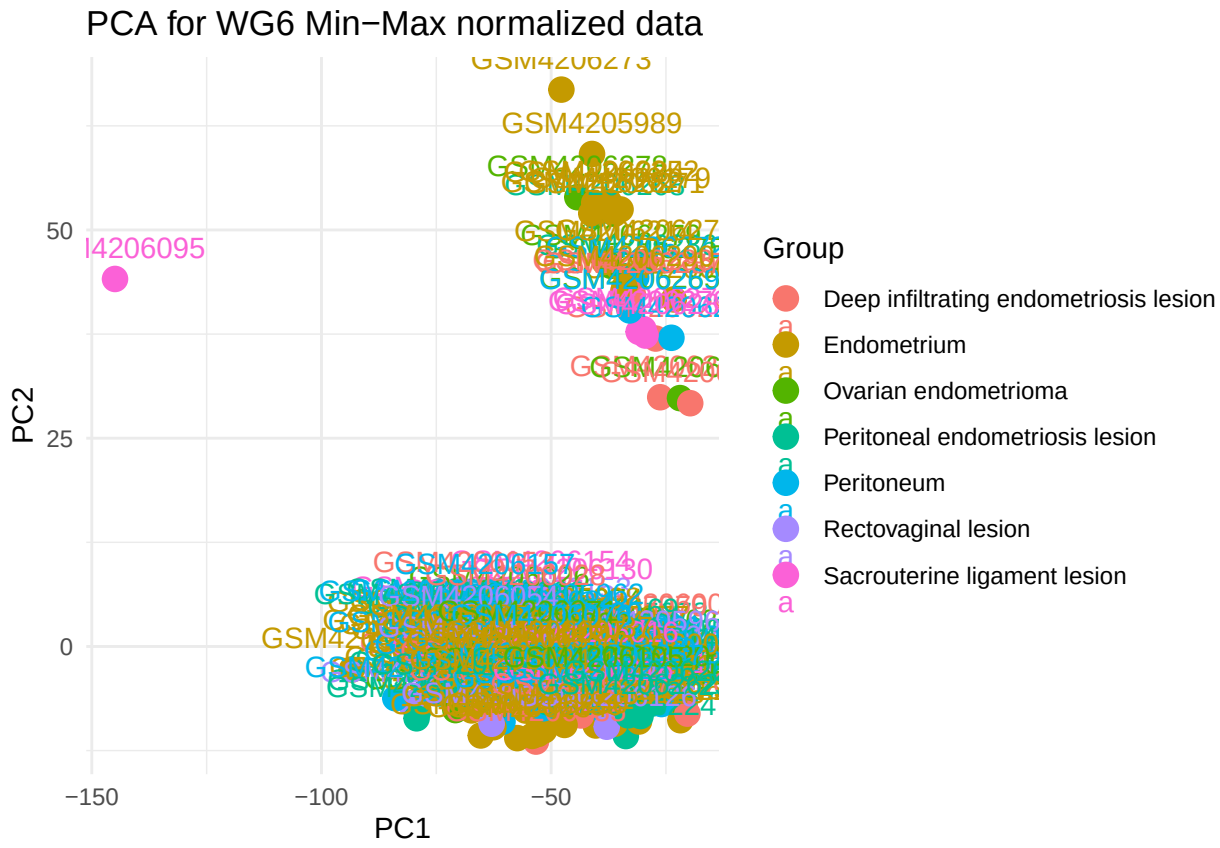


From the PCA plot, it seems like that the Z-score normalization reduces the linearity in PCA result plot of the WG6 raw data, yet it does not eliminate the batch effect. Additionally, one sample (GSM4206095) appears to be an outlier after normalization. However, upon reviewing the literature, the original study does not categorize this sample as an outlier, and it passed the Quality Control during the data processing phase conducted by the authors.

PCA with Min-max normalization

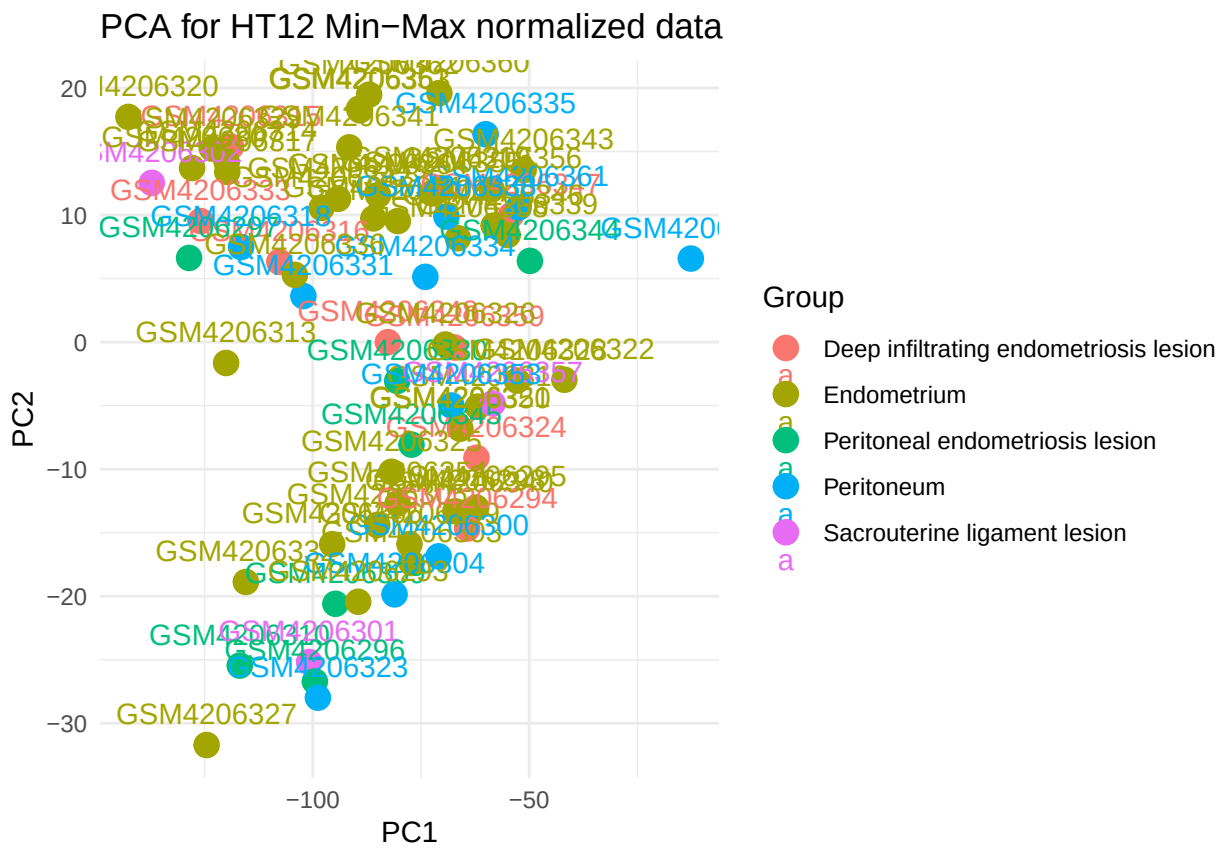
```r
# Min-Max

min_max_normalize <- function(row) {
  return((row - min(row)) / (max(row) - min(row)))
```

```
}

# Apply the normalization function to each row
num_WG6_norm <- t(apply(num_WG6, 1, min_max_normalize))
num_HT12_norm <- t(apply(num_HT12, 1, min_max_normalize))

# PCA for WG-6 normalized data
# remove all the NA values
num_WG6_norm[is.na(num_WG6_norm)] <- 0

# In PCA, every row is a sample and every columns is a feature(variable)
pca_result <- prcomp(t(num_WG6_norm), center = FALSE, scale. = FALSE)
pca_data <- data.frame(pca_result$x, Group = group_labels_WG6)
ggplot(pca_data, aes(x = PC1, y = PC2, color = Group, label = rownames(pca_data))) +
  geom_point(size = 4) +
  geom_text(vjust = -1) +
  labs(title = "PCA for WG6 Min-Max normalized data", x = "PC1", y = "PC2") +
  theme_minimal()
```




```
# PCA for HT-12 normalized data
num_HT12_norm[is.na(num_HT12_norm)] <- 0
# In PCA, every row is a sample and every columns is a feature(variable)
pca_result <- prcomp(t(num_HT12_norm), center = FALSE, scale. = FALSE)
pca_data <- data.frame(pca_result$x, Group = group_labels_HT12)
ggplot(pca_data, aes(x = PC1, y = PC2, color = Group, label = rownames(pca_data))) +
```

```
  geom_point(size = 4) +
  geom_text(vjust = -1) +
  labs(title = "PCA for HT12 Min-Max normalized data", x = "PC1", y = "PC2") +
  theme_minimal()
```



The results are quite similar.