

# HGEN 48800 homework: Unsupervised learning methods applied to some genomics problems

Lin\_Yu

2024-04-29

## Problem 1: Batch effect

### Dependencies

This document depends on the following packages:

```
library(devtools)
library(Biobase)
library(sva)
library(bladderbatch)
library(snpStats)
library(factoextra)
```

To install these packages you can use the code (or if you are compiling the document, remove the `eval=FALSE` from the chunk.)

```
#install.packages(c("devtools", "factoextra"))
library(BiocManager)
#BiocManager::install(c("Biobase", "sva", "bladderbatch", "snpStats"))
```

### Download the data

The analyses performed in this experiment are based on gene expression measurements from a bladder cancer study: Gene expression in the urinary bladder: a common carcinoma in situ gene expression signature exists disregarding histopathological classification. The data can be loaded from the bladderbatch data package.

```
data(bladderdata)
```

### Set up the data

```
pheno = pData(bladderEset)
edata = exprs(bladderEset)
colnames(edata) <- gsub(".CEL", "", colnames(edata))
colnames(edata) <- paste0("B", pheno$batch, "_", colnames(edata))
table(pheno$batch, pheno$outcome)
```

## Perform PCA

```
resPC <- prcomp(t(edata))
batch.id <- pheno$batch
outcome <- pheno$outcome

PC1 <- resPC$x[, 1]
PC2 <- resPC$x[, 2]

# Plot the PCA
pca_df <- data.frame(PC1 = PC1, PC2 = PC2, Batch = batch.id, Outcome = outcome)

library(ggplot2)

ggplot(pca_df, aes(x = PC1, y = PC2, color = Batch, shape = Outcome)) +
  geom_point(size = 3) +
  labs(title = "PCA Plot", x = "PC1", y = "PC2") +
  theme_minimal() +
  theme(legend.position = "right")
```

## Perform hierarchical clustering

```
d <- dist(t(edata), method = "euclidean")

# Hierarchical clustering using Complete Linkage
d <- dist(t(edata), method = "euclidean")

# Hierarchical clustering using Complete Linkage
hc <- hclust(d, method = "complete")

# Plot the obtained dendrogram
plot(hc, main = "Hierarchical Clustering", xlab = "Samples", sub = "", cex = 0.5)
```

## Visualize the clustering results in a scatter plot

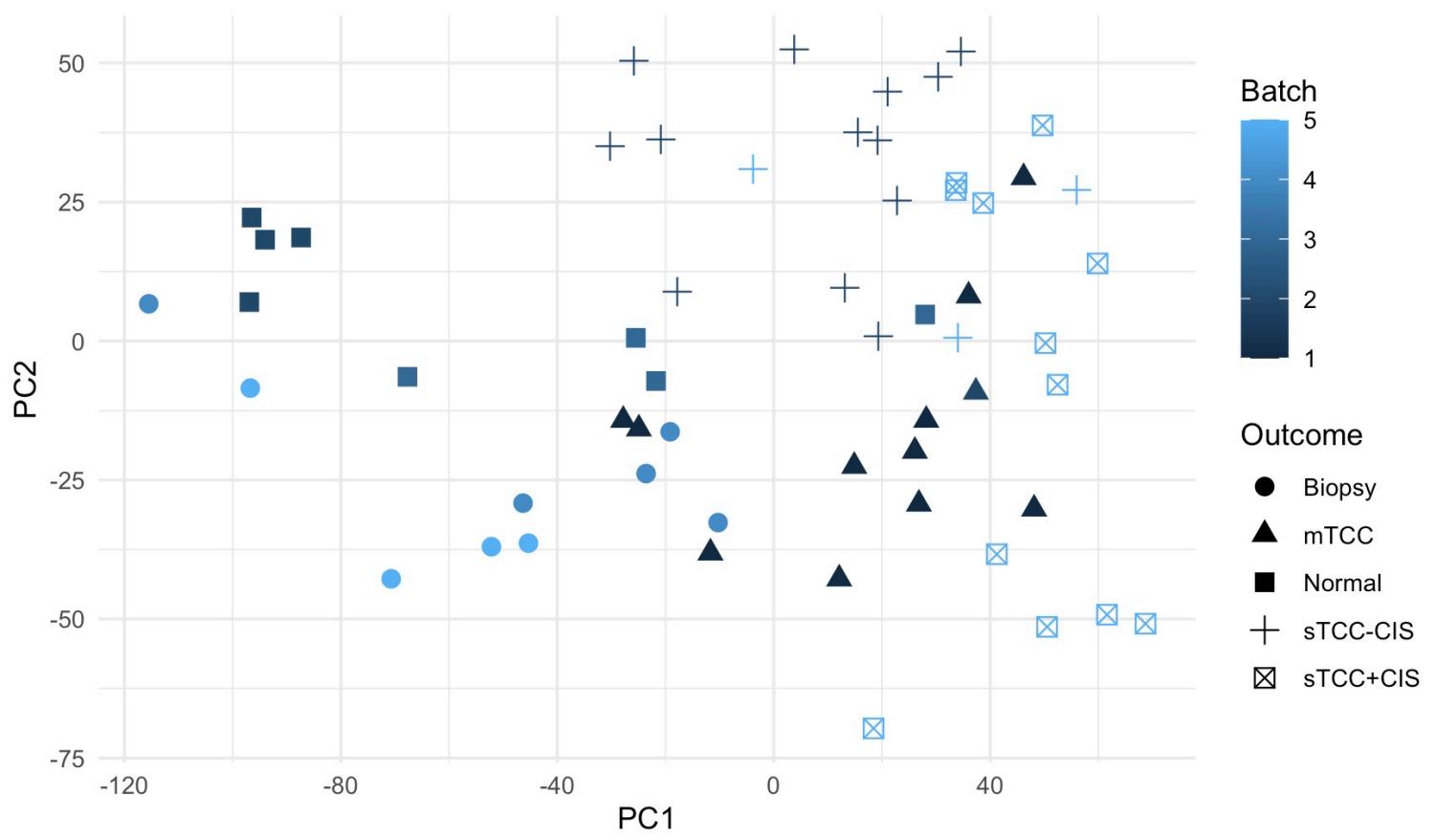
```
# using function fviz_cluster()

# Cut the tree into a desired number of clusters, e.g., k = 5
clusters <- hcut(t(edata), k=5, hc_method="complete")

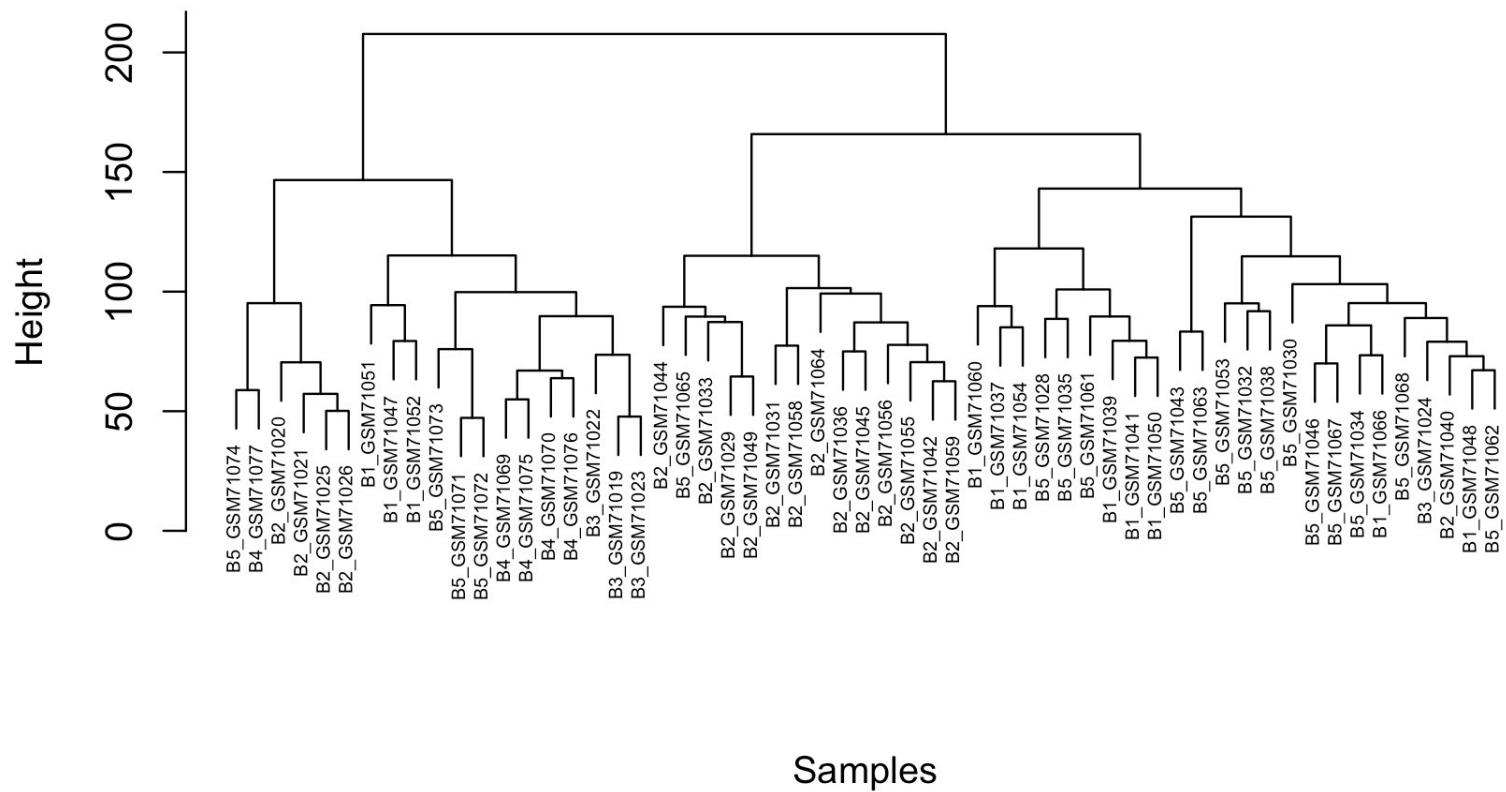
# using function fviz_cluster() to visualize the clusters
fviz_cluster(clusters,
             geom = "point",           # Type of plot (points)
             ellipse.type = "norm")    # Color palette
```

## Questions:

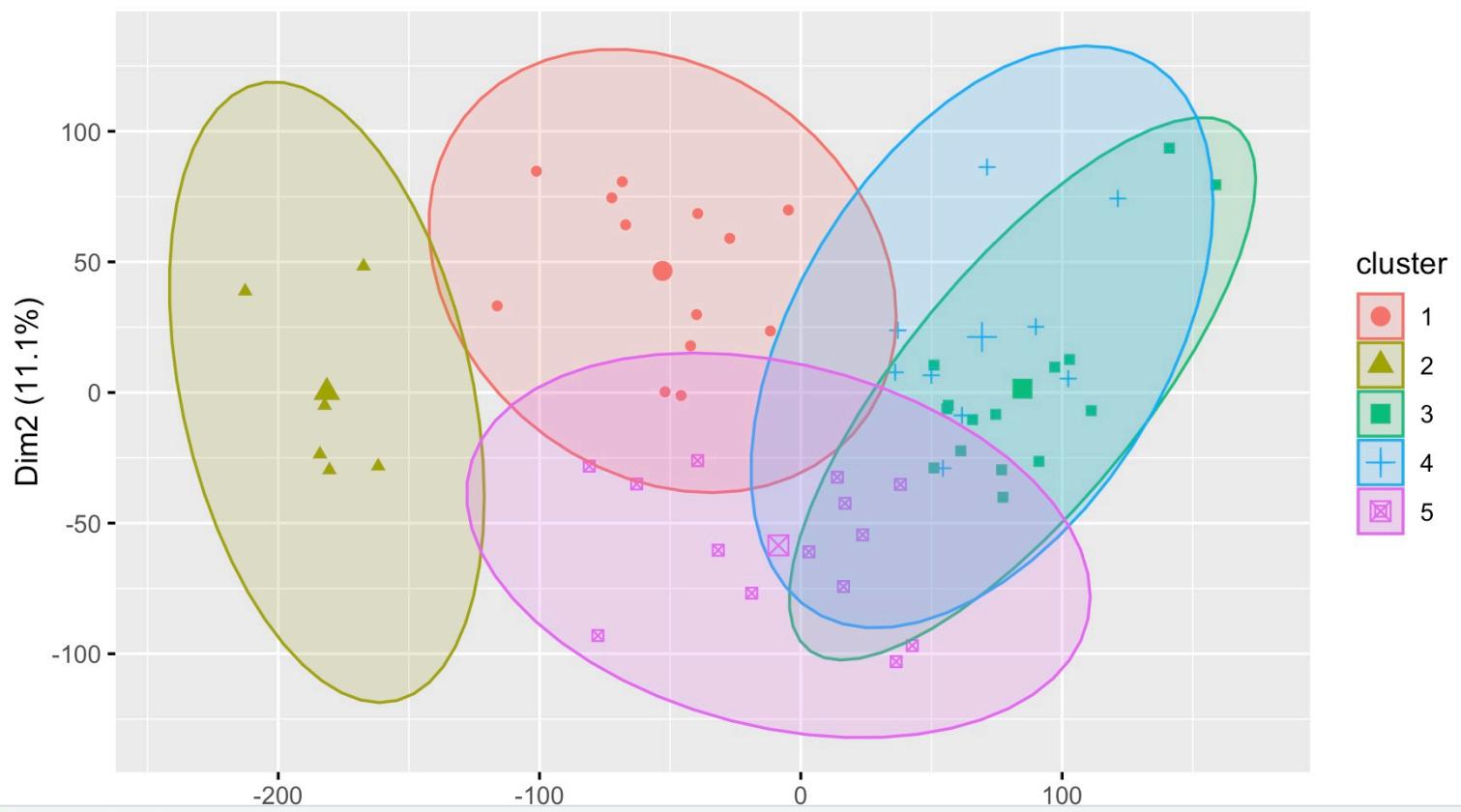
## PCA Plot



## Hierarchical Clustering



Cluster plot



- Finish the code above and report output of each step. Answer:
  1. Can you make comments on the experimental design based on the table of batch and outcome? Any problems? How can you improve the design?
  2. By looking at PCA, do you observe batch effects in this dataset? Why or why not?
  3. When only looking at the above hierarchical clustering results, can you make judgements on the batch effects? If not, what else analysis do you need to do?

#### Answers:

1. The table shows a significant imbalance in clinical outcomes across batches, leading to potential biases. For instance, batch 5 contains mostly sTCC+CIS samples, while Batches 2,3 contains mostly Normal samples, indicating a skewed distribution. Additionally, certain outcomes like Biopsy and mTCC are only present in select batches, making it harder to control for batch effects. To enhance the design, samples should be randomly distributed across batches, ensuring a balanced representation of clinical outcomes. Adding replicates for each outcome across batches could also be helpful when identifying batch effects. Employing batch effect correction methods, conducting pilot studies, and standardizing protocols can also help minimize biases and improve the study's reliability.
2. The PCA plot suggests minimal batch effects, as evidenced by the lack of strong clustering based on color (batch information), indicating batches do not strongly influence the primary data variations.
3. By comparing the PCA plots with hierarchical clustering plots, we observe a batch effect in hierarchical clustering. The clusters closely align with batches 1 to 5. This pattern is also evident in the dendrogram, where in certain branches, all leaves come from the same batch.

## Problem 2: Technical Variability Vs. Biological Variability

### Introduction

In the following sections we will cover inference in the context of genomics experiments. Here we introduce a concept that is particularly important in the analysis of genomics data: the distinction between biological and technical variability.

In general, the variability we observe across biological units, such as individuals, within a population is referred to as **biological**. We refer to the variability we observe across measurements of the same biological unit, such as aliquots from the same biological sample, as **technical**. Because newly developed measurement technologies are common in genomics, technical replicates are used many times to assess experimental data. By generating measurements from samples that are designed to be the same, we are able to measure and assess technical variability. We also use the terminology **biological replicates** and **technical replicates** to refer to samples from which we can measure biological and technical variability respectively.

It is important not to confuse biological and technical variability when performing statistical inference as the interpretation is quite different. For example, when analyzing data from technical replicates, the population is just the one sample from which these come from as opposed to more general population such as healthy humans or control mice. Here we explore this concept with a experiment that was designed to include both technical and biological replicates.

### Pooling experiment data

The dataset we will study includes data from gene expression arrays. In this experiment, RNA was extracted from 12 randomly selected mice from two strains. All 24 samples were hybridized to microarrays but we also

formed pools, including two pools from with the RNA from all twelve mice from each of the two strains. Other pools were also created, as we will see below, but we will ignore these here.

We will need the following library which you need to install if you have not done so already:

```
library(devtools)
install_github("genomicsclass/maPooling")
install_github("ririzarr/rafalib")
```

We can see the experimental design using the `pData` function. Each row represents a sample and the column are the mice. A 1 in cell  $i, j$  indicates that RNA from mouse  $j$  was included in sample  $i$ . The strain can be identified from the row names (this is not a recommended approach, you can add additional variables to the `phenoData` to make strain information explicit.)

```
library(BioBase)
library(maPooling)
data(maPooling)
head(pData(maPooling))
```

Below we create an image to illustrate which mice were included in which samples:

```
library(rafalib)
mypar()
flipt <- function(m) t(m[nrow(m):1,])
myimage <- function(m,...) {
  image(flipt(m),xaxt="n",yaxt="n",...)
}
myimage(as.matrix(pData(maPooling)),col=c("white","black"),
       xlab="experiments",
       ylab="individuals",
       main="phenoData")
```

Note that ultimately we are interested in detecting genes that are differentially expressed between the two strains of mice which we will refer to as strain 0 and 1. We can apply tests to the technical replicates of pooled samples or the data from 12 individual mice. We can identify these pooled samples because all mice from each strain were represented in these samples and thus the sum of the rows of experimental design matrix add up to 12:

```
data(maPooling)
pd=pData(maPooling)
pooled=which(rowSums(pd)==12)
```

We can determine the strain from the column names:

```
factor(as.numeric(grep("b",names(pooled))))
```

If we compare the mean expression between groups for each gene we find several showing consistent differences. Here are two examples:

```
###look at 2 pre-selected genes for illustration
i=11425;j=11878
pooled_y=exprs(maPooling[,pooled])
```

```

pooled_g=factor(as.numeric(grep1("b",names(pooled))))
mypar(1,2)
stripchart(split(pooled_y[i,],pooled_g),vertical=TRUE,method="jitter",col=c(1,2),
           main="Gene 1",xlab="Group",pch=15)
stripchart(split(pooled_y[j,],pooled_g),vertical=TRUE,method="jitter",col=c(1,2),
           main="Gene 2",xlab="Group",pch=15)

```

Note that if we compute a t-test from these values we obtain highly significant results

```

library(genefilter)
pooled_tt=rowttests(pooled_y,pooled_g)
pooled_tt$p.value[i]
pooled_tt$p.value[j]

```

Observe that what is being replicated here is the experimental protocol. We have created four **technical replicates** for each pooled sample. Gene 1 may be a highly variable gene within strain of mice while Gene 2 a stable one, but we have no way of seeing this, because mouse-to-mouse variability is submerged in the act of pooling.

We also have microarray data for each individual mouse. For each strain we have 12 **biological replicates**. We can find them by looking for rows with just one 1.

```
individuals=which(rowSums(pd)==1)
```

It turns out that some technical replicates were included for some individual mice so we remove them to illustrate an analysis with only biological replicates:

```

##remove replicates
individuals=individuals[-grep("tr",names(individuals))]
y=exprs(maPooling)[,individuals]
g=factor(as.numeric(grep1("b",names(individuals))))

```

We can compute the sample variance for each gene and compare to the standard deviation obtained with the technical replicates.

```

technicalsds <- rowSds(pooled_y[,pooled_g==0])
biologicalsds <- rowSds(y[,g==0])
LIM=range(c(technicalsds,biologicalsds))
mypar(1,1)
boxplot(technicalsds,biologicalsds,names=c("technical","biological"),ylab="standard deviation")

```

Note the biological variance is much larger than the technical variance. And also that the variability of variances is also larger for biological variance. Here are the two genes we showed above but now we show expression values measured on each individual mouse

```

mypar(1,2)
stripchart(split(y[i,],g),vertical=TRUE,method="jitter",col=c(1,2),xlab="Gene 1",pch=15)
points(c(1,2),tapply(y[i,],g,mean),pch=4,cex=1.5)
stripchart(split(y[j,],g),vertical=TRUE,method="jitter",col=c(1,2),xlab="Gene 2",pch=15)
points(c(1,2),tapply(y[j,],g,mean),pch=4,cex=1.5)

```

Now the p-values tell a different story

```
library(genefilter)
tt=rownames(y,g)
tt$p.value[i]
tt$p.value[j]
```

### Questions:

- Read the context and run the code above. Answer:
  1. What does the biological and technical variability mean in this example?
  2. Which of these two genes do we feel more confident reporting as being differentially expressed between strains?
  3. If another investigator takes another random sample of mice and tries the same experiment, which gene do you think will be identified?

### Answer:

1. Biological variability means differences in gene expression observed between individual mice, which captures the natural variation present in the population. Technical variability means differences in measurements taken from pooled RNA of the same mice. And these differences are introduced by the measurement technology and experimental protocol
2. After adjusting for technical variability in the graph, we are more confident that gene 2 is differentially expressed between strains.
3. I believe gene 2 should be identified as differentially expressed, as the graph suggests that biological variability plays a more significant role. And I have evident that, without technical variability, gene 2 is differentially expressed between strains.

## Problem 3: Unsupervised visualization on gene expression data

Machine learning is a very broad topic and a highly active research area. In the life sciences, much of what is described as **precision medicine** is an application of machine learning to biomedical data. The general idea is to predict or discover outcomes from measured predictors. Clustering is a major machine learning component that can be used to answer questions like following: can we discover new types of cancer from gene expression profiles? Can we separate cell lines or tissues based on gene expression markers?

### Clustering

We will demonstrate the concepts and code needed to perform clustering analysis with the tissue gene expression data:

```
load("tissuesGeneExpression.rda")
```

**Hierarchical clustering** With the distance between each pair of samples computed, we need clustering algorithms to join them into groups. Hierarchical clustering is one of the many clustering algorithms available to do this. Each sample is assigned to its own group and then the algorithm continues iteratively, joining the two most similar clusters at each step, and continuing until there is just one group. While we have defined distances between samples, we have not yet defined distances between groups. There are various ways this

can be done and they all rely on the individual pairwise distances. The helpfile for `hclust` includes detailed information.

We can perform hierarchical clustering based on the distances defined above using the `hclust` function. This function returns an `hclust` object that describes the groupings that were created using the algorithm described above. The `plot` method represents these relationships with a tree or dendrogram:

```
d <- dist( t(e) )
library(rafalib)
mypar()
hc <- hclust(d)
hc
plot(hc, labels=tissue, cex=0.5)
```

Does this technique “discover” the clusters defined by the different tissues? In this plot, it is not easy to see the different tissues so we add colors by using the `myplclust` function from the `rafalib` package.

```
myplclust(hc, labels=tissue, lab.col=as.fumeric(tissue), cex=0.5)
```

Visually, it does seem as if the clustering technique has discovered the tissues. However, hierarchical clustering does not define specific clusters, but rather defines the dendrogram above. From the dendrogram we can decipher the distance between any two groups by looking at the height at which the two groups split into two.

**Questions:** Use `cutree` function:

- Can you cut the tree into 8 clusters? Can you examine how the clusters overlap with the actual tissues using function `table`? Yes, I can. Based on the table, we observe that endometrium and kidney tissues predominantly cluster in cluster 1. Cerebellum tissues cluster primarily in cluster 3, while colon tissues mainly cluster in cluster 4. Hippocampus tissues are distributed across clusters 2 and 3, and liver tissues predominantly cluster in cluster 5
- Instead of specifying number of clusters, we can also define clusters by “cutting the tree” at some distance and group all samples that are within that distance into groups below. Can you cut the tree at the height of 120? How the clusters overlap with the actual tissues?
- Can you perform hierarchical clustering using a different agglomeration method? Check `method` option in the `hclust` function for possibilities. Do you get same result using an alternative method? What does it imply in real data analysis when you have different choices of agglomeration methods and distance metrics?

**K-means** We can also cluster with the `kmeans` function to perform k-means clustering. We can see this in the first plot above. If we instead perform k-means clustering using all of the genes, we obtain a much improved result. To visualize this, we can use an MDS plot:

```
km <- kmeans(t(e), centers=7)
mds <- cmdscale(d)
## make the plot
```

**Questions:**

- What do you observe from the MDS plot? Do we obtain a similar answer to that obtained with hierarchical clustering? Confirm by tabulating the results.

**\*\*Questions\*\*:** Use `cutree` function:

- Can you cut the tree into 8 clusters? Can you examine how the clusters overlap with the actual tissues using function `table`?

```
```{r}
# Cut the tree into 8 clusters
# Perform hierarchical clustering and cut the tree into 8 clusters
hc <- hclust(d)
clusters_8 <- cutree(hc, k = 8)

# Compare clusters with actual tissues
table(clusters_8, tissue)
````
```

| tissue | cerebellum | colon | endometrium | hippocampus | kidney | liver | placenta |
|--------|------------|-------|-------------|-------------|--------|-------|----------|
| 1      | 0          | 0     | 15          | 0           | 37     | 0     | 0        |
| 2      | 0          | 0     | 0           | 12          | 0      | 0     | 0        |
| 3      | 31         | 0     | 0           | 19          | 0      | 0     | 0        |
| 4      | 0          | 34    | 0           | 0           | 0      | 0     | 0        |
| 5      | 0          | 0     | 0           | 0           | 0      | 24    | 0        |
| 6      | 2          | 0     | 0           | 0           | 2      | 2     | 0        |
| 7      | 5          | 0     | 0           | 0           | 0      | 0     | 0        |
| 8      | 0          | 0     | 0           | 0           | 0      | 0     | 6        |

Based on the table above, we observe that endometrium and kidney tissues predominantly cluster in cluster 1. Cerebellum tissues cluster primarily in cluster 3, while colon tissues mainly cluster in cluster 4. Hippocampus tissues are distributed across clusters 2 and 3, and liver tissues predominantly cluster in cluster 5

325  
324 - Instead of specifying number of clusters, we can also define clusters by "cutting the tree" at some distance and group all samples  
that are within that distance into groups below. Can you cut the tree at the height of 120? How the clusters overlap with the actual  
tissues?

325  
326 ````{r}  
327 # Cut the tree at height 120  
328 clusters\_height <- cutree(hc, h = 120)  
329  
330 # Compare clusters with actual tissues  
331 table(clusters\_height, tissue)  
332  
333 ````

| clusters_height | cerebellum | colon | endometrium | hippocampus | kidney | liver | placenta |
|-----------------|------------|-------|-------------|-------------|--------|-------|----------|
| 1               | 0          | 0     | 0           | 0           | 9      | 0     | 0        |
| 2               | 0          | 0     | 0           | 0           | 18     | 0     | 0        |
| 3               | 0          | 0     | 0           | 12          | 0      | 0     | 0        |
| 4               | 0          | 0     | 0           | 19          | 0      | 0     | 0        |
| 5               | 31         | 0     | 0           | 0           | 0      | 0     | 0        |
| 6               | 0          | 0     | 0           | 0           | 10     | 0     | 0        |
| 7               | 0          | 34    | 0           | 0           | 0      | 0     | 0        |
| 8               | 0          | 0     | 0           | 0           | 0      | 24    | 0        |
| 9               | 2          | 0     | 0           | 0           | 2      | 0     | 0        |
| 10              | 0          | 0     | 0           | 0           | 0      | 2     | 0        |
| 11              | 0          | 0     | 15          | 0           | 0      | 0     | 0        |
| 12              | 5          | 0     | 0           | 0           | 0      | 0     | 0        |
| 13              | 0          | 0     | 0           | 0           | 0      | 0     | 2        |
| 14              | 0          | 0     | 0           | 0           | 0      | 0     | 4        |

334 Based on the table above, we observe that kidney tissues predominantly cluster in cluster 1,2,6. Cerebellum tissues cluster primarily  
in cluster 5, while colon tissues mainly cluster in cluster 7. Hippocampus tissues are distributed across clusters 3 and 4, and liver  
tissues predominantly cluster in cluster 8. Additionally, endometrium tissues cluster only in cluster 11

## Cut the tree into 8 clusters

```
336 - Can you perform hierarchical clustering using a different agglomeration method? Check `method` option in the `hclust` function for possibilities. Do you get same result using an alternative method? What does it imply in real data analysis when you have different choices of agglomeration methods and distance metrics?
```

```
337  
338 ``{r}  
339 hc_average <- hclust(d, method = "average")  
340 ``  
341  
342 ``{r}  
343  
344 # Cut the tree into 8 clusters  
345 clusters_8_average<- cutree(hc_average, k = 8)  
346  
347 # Compare clusters with actual tissues  
348 table(clusters_8_average, tissue)  
349 ``
```

| tissue | cerebellum | colon | endometrium | hippocampus | kidney | liver | placenta |
|--------|------------|-------|-------------|-------------|--------|-------|----------|
| 1      | 0          | 0     | 0           | 0           | 37     | 0     | 0        |
| 2      | 31         | 0     | 0           | 31          | 0      | 0     | 0        |
| 3      | 0          | 34    | 0           | 0           | 0      | 0     | 0        |
| 4      | 0          | 0     | 0           | 0           | 0      | 24    | 0        |
| 5      | 2          | 0     | 0           | 0           | 2      | 2     | 0        |
| 6      | 0          | 0     | 15          | 0           | 0      | 0     | 0        |
| 7      | 5          | 0     | 0           | 0           | 0      | 0     | 0        |
| 8      | 0          | 0     | 0           | 0           | 0      | 0     | 6        |

```
350 ``{r}  
351 hc_single <- hclust(d, method = "single")  
352 # Cut the tree into 8 clusters  
353 clusters_8_single<- cutree(hc_single, k = 8)  
354
```

```

```{r}
hc_single <- hclust(d, method = "single")
# Cut the tree into 8 clusters
clusters_8_single<- cutree(hc_single, k = 8)

# Compare clusters with actual tissues
table(clusters_8_single, tissue)
```

```

|   | tissue     |       |             |             |        |       |          |
|---|------------|-------|-------------|-------------|--------|-------|----------|
|   | cerebellum | colon | endometrium | hippocampus | kidney | liver | placenta |
| 1 | 0          | 0     | 15          | 0           | 39     | 0     | 0        |
| 2 | 31         | 0     | 0           | 31          | 0      | 0     | 0        |
| 3 | 0          | 34    | 0           | 0           | 0      | 0     | 0        |
| 4 | 0          | 0     | 0           | 0           | 0      | 24    | 0        |
| 5 | 2          | 0     | 0           | 0           | 0      | 0     | 0        |
| 6 | 0          | 0     | 0           | 0           | 0      | 2     | 0        |
| 7 | 5          | 0     | 0           | 0           | 0      | 0     | 0        |
| 8 | 0          | 0     | 0           | 0           | 0      | 0     | 6        |

```

```{r}
#Results of hierarchy clustering with complete method
table(clusters_8, tissue)
```

```

|   | tissue     |       |             |             |        |       |          |
|---|------------|-------|-------------|-------------|--------|-------|----------|
|   | cerebellum | colon | endometrium | hippocampus | kidney | liver | placenta |
| 1 | 0          | 0     | 15          | 0           | 37     | 0     | 0        |
| 2 | 0          | 0     | 0           | 12          | 0      | 0     | 0        |
| 3 | 31         | 0     | 0           | 19          | 0      | 0     | 0        |
| 4 | 0          | 34    | 0           | 0           | 0      | 0     | 0        |
| 5 | 0          | 0     | 0           | 0           | 0      | 24    | 0        |
| 6 | 2          | 0     | 0           | 0           | 2      | 2     | 0        |

```

```{r}
# Hierarchical clustering with average linkage (WPGMA)
hc_mcquitty <- hclust(d, method = "mcquitty")

clusters_8_mcquitty <- cutree(hc_mcquitty, k = 8)

# Compare clusters with actual tissues
table(clusters_8_mcquitty, tissue)
```

```

|   | tissue     |       |             |             |        |       |          |
|---|------------|-------|-------------|-------------|--------|-------|----------|
|   | cerebellum | colon | endometrium | hippocampus | kidney | liver | placenta |
| 1 | 0          | 0     | 0           | 0           | 37     | 0     | 0        |
| 2 | 31         | 0     | 0           | 31          | 0      | 0     | 0        |
| 3 | 0          | 34    | 0           | 0           | 0      | 0     | 0        |
| 4 | 0          | 0     | 0           | 0           | 0      | 24    | 0        |
| 5 | 2          | 0     | 0           | 0           | 2      | 2     | 0        |
| 6 | 0          | 0     | 15          | 0           | 0      | 0     | 0        |
| 7 | 5          | 0     | 0           | 0           | 0      | 0     | 0        |
| 8 | 0          | 0     | 0           | 0           | 0      | 0     | 6        |

While comparing the results obtained from different agglomeration methods, we notice similarities but not exact matches. The predominant clustering of each tissue type remains consistent across the methods. For example, kidney tissues consistently cluster into a single group across all three cases. Similarly, cerebellum, liver, and colon tissues maintain their predominant clustering patterns. However, the combinations of tissue clusters within each group may vary. For instance, kidney tissues cluster with endometrium tissues in both the complete and single methods, but appear as an isolated cluster in the average method.

While the variations between different agglomeration methods and distance metrics may not be substantial in this instance, it remains crucial to explore various combinations of these parameters. This exploration helps to grasp the robustness of the clustering outcomes, ensuring a comprehensive understanding of the data's structure.

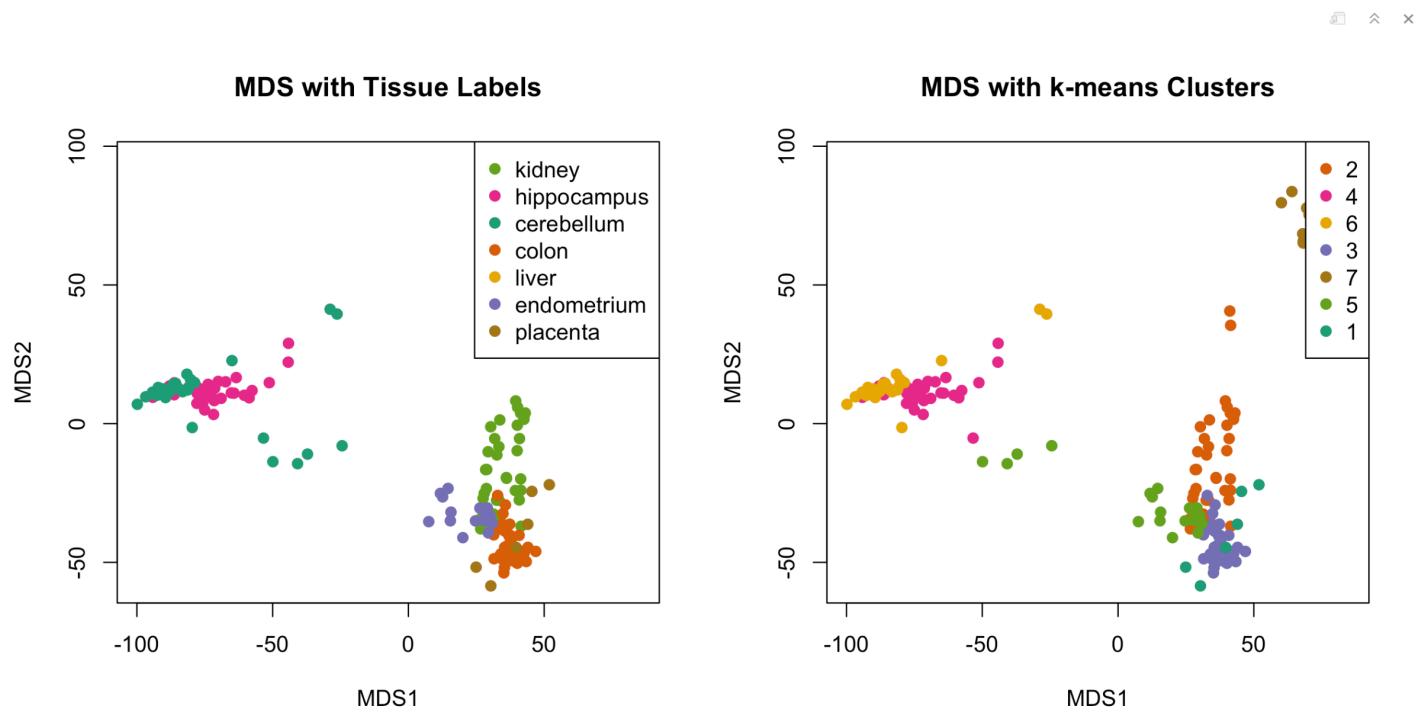
#### #### K-means

We can also cluster with the `kmeans` function to perform k-means clustering. We can see this in the first plot above. If we instead perform k-means clustering using all of the genes, we obtain a much improved result. To visualize this, we can use an MDS plot:

```
```{r kmeans_mds, fig.cap="Plot of gene expression for first two PCs with color representing tissues (left) and clusters found using all genes (right).",fig.width=10.5,fig.height=5.25, eval=FALSE}
km <- kmeans(t(e), centers=7)
mds <- cmdscale(d)
# Make the plot
par(mfrow = c(1, 2)) # Set up plotting area for side-by-side plots

# Plot MDS with actual tissue labels
plot(mds, col = as.factor(tissue), pch = 19, main = "MDS with Tissue Labels",
      xlab = "MDS1", ylab = "MDS2")
legend("topright", legend = unique(tissue), col = unique(as.factor(tissue)), pch = 19)

# Plot MDS with k-means clusters
plot(mds, col = km$cluster, pch = 19, main = "MDS with k-means Clusters",
      xlab = "MDS1", ylab = "MDS2")
legend("topright", legend = unique(km$cluster), col = unique(km$cluster), pch = 19)
````
```



```
```{r}
table(km$cluster, tissue)
```



	tissue	cerebellum	colon	endometrium	hippocampus	kidney	liver	placenta
1		0	0	0	0	0	0	6
2		0	0	0	0	39	0	0
3		0	34	0	0	0	0	0
4		1	0	0	31	0	0	0
5		4	0	15	0	0	0	0
6		33	0	0	0	0	0	0
7		0	0	0	0	0	26	0

```
```{r}
# Cut the tree into 7 clusters
clusters_7 <- cutree(hc, k = 7)
```



```
# Compare clusters with actual tissues
table(clusters_7, tissue)
````
```

|   | tissue | clusters_7 | cerebellum | colon | endometrium | hippocampus | kidney | liver | placenta |
|---|--------|------------|------------|-------|-------------|-------------|--------|-------|----------|
| 1 |        |            | 0          | 0     | 15          | 0           | 37     | 0     | 0        |
| 2 |        |            | 0          | 0     | 0           | 12          | 0      | 0     | 0        |
| 3 |        |            | 36         | 0     | 0           | 19          | 0      | 0     | 0        |
| 4 |        |            | 0          | 34    | 0           | 0           | 0      | 0     | 0        |
| 5 |        |            | 0          | 0     | 0           | 0           | 0      | 24    | 0        |
| 6 |        |            | 2          | 0     | 0           | 0           | 2      | 2     | 0        |
| 7 |        |            | 0          | 0     | 0           | 0           | 0      | 0     | 6        |

**\*\*Questions\*\*:**

- What do you observe from the MDS plot? Do we obtain a similar answer to that obtained with hierarchical clustering? Confirm by tabulating the results.

The MDS plot shows a strong overlap between k-means clusters and tissue labels, indicating k-means has effectively grouped the tissues based on gene expression data.

Upon examining the two tables above, we confirms that both clustering methods largely identify similar groupings, although some variations in cluster assignments are expected due to methodological differences. For instance, kidney tissues consistently cluster into a single group in both cases. Similarly, cerebellum, liver, endometrium, placenta, and colon tissues maintain their predominant clustering patterns. However, there is one notable difference: hippocampus clusters into two groups in hierarchical clustering, whereas it forms only one group in k-means clustering

**Heatmaps** Heatmaps are ubiquitous in the genomics literature. They are very useful plots for visualizing the measurements for a subset of rows over all the samples. A *dendrogram* is added on top and on the side that is created with hierarchical clustering. We will demonstrate how to create heatmaps from within R. Let's begin by defining a color palette:

```
library(RColorBrewer)
hmcol <- colorRampPalette(brewer.pal(9, "GnBu"))(100)
```

Now, pick the genes with the top variance over all samples:

```
library(genefilter)
rv <- rowVars(e)
idx <- order(-rv)[1:40]
```

While a `heatmap` function is included in R, we recommend the `heatmap.2` function from the `gplots` package on CRAN because it is a bit more customized. For example, it stretches to fill the window. Here we add colors to indicate the tissue on the top:

```
library(gplots) ##Available from CRAN
cols <- palette(brewer.pal(8, "Dark2"))[as.fumeric(tissue)]
head(cbind(colnames(e),cols))
heatmap.2(e[idx,], labCol=tissue,
          trace="none",
          ColSideColors=cols,
          col=hmcol)
```

We did not use tissue information to create this heatmap, and we can quickly see, with just 40 genes, good separation across tissues.

### Questions:

- Can you create heatmaps using the 100 most variable genes and the 200 most variable genes, respectively? How do your results compare to the heatmap based on 40 genes?
- How does the *dendrogram* on top of your heatmaps compare to that from previous hierarchical clustering analysis?

## Reference

This homework contains online materials written by Rafael Irizarry, Jeff Leek, Ethan Cerami, Dave Tang.

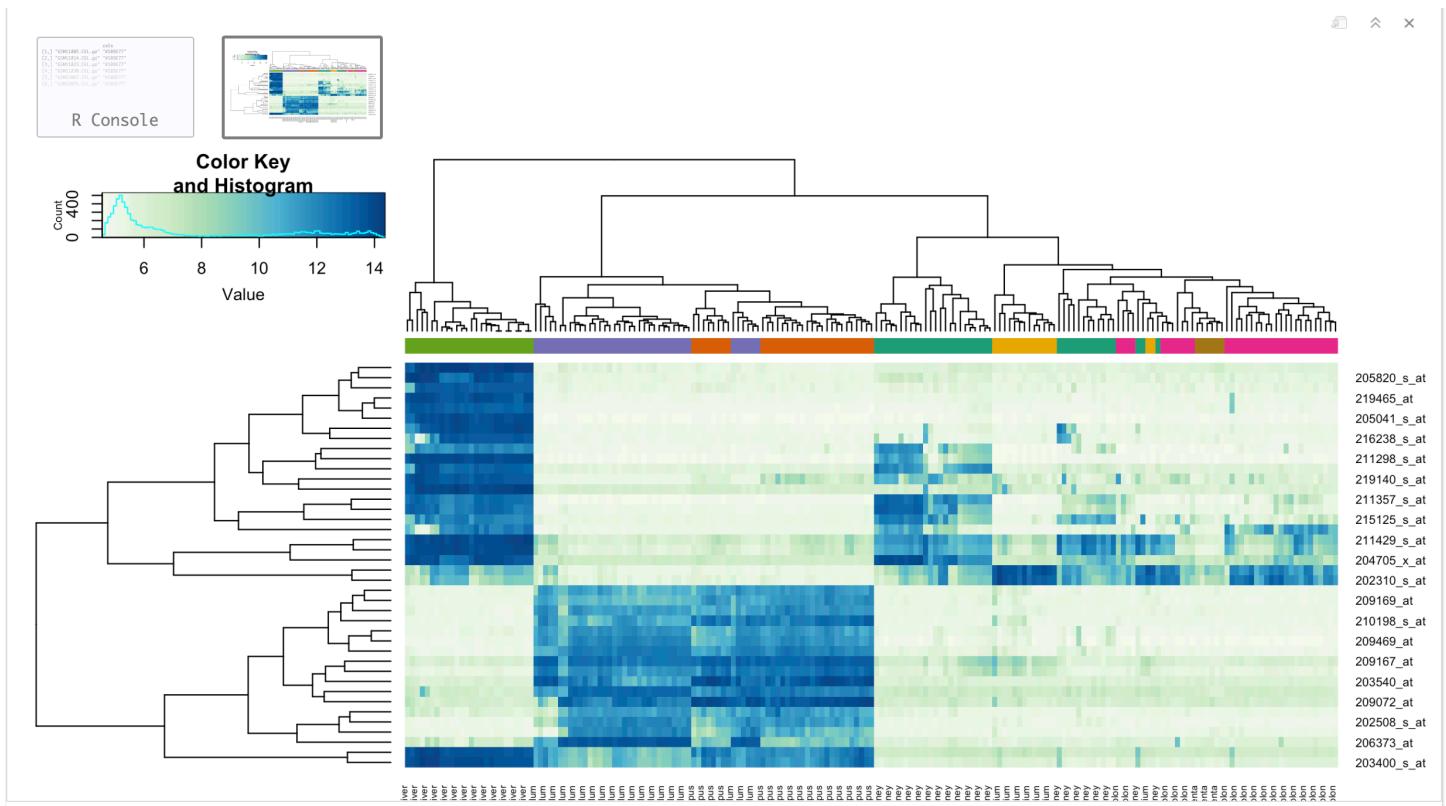
## Session information

Here is the session information.

```
devtools::session_info()

## - Session info -----
##   setting  value
##   version  R version 4.4.0 (2024-04-24)
##   os        macOS Monterey 12.2
```

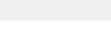
## 40 variables



**\*\*Questions\*\*:**

- Can you create heatmaps using the 100 most variable genes and the 200 most variable genes, respectively? How do your results compare to the heatmap based on 40 genes?

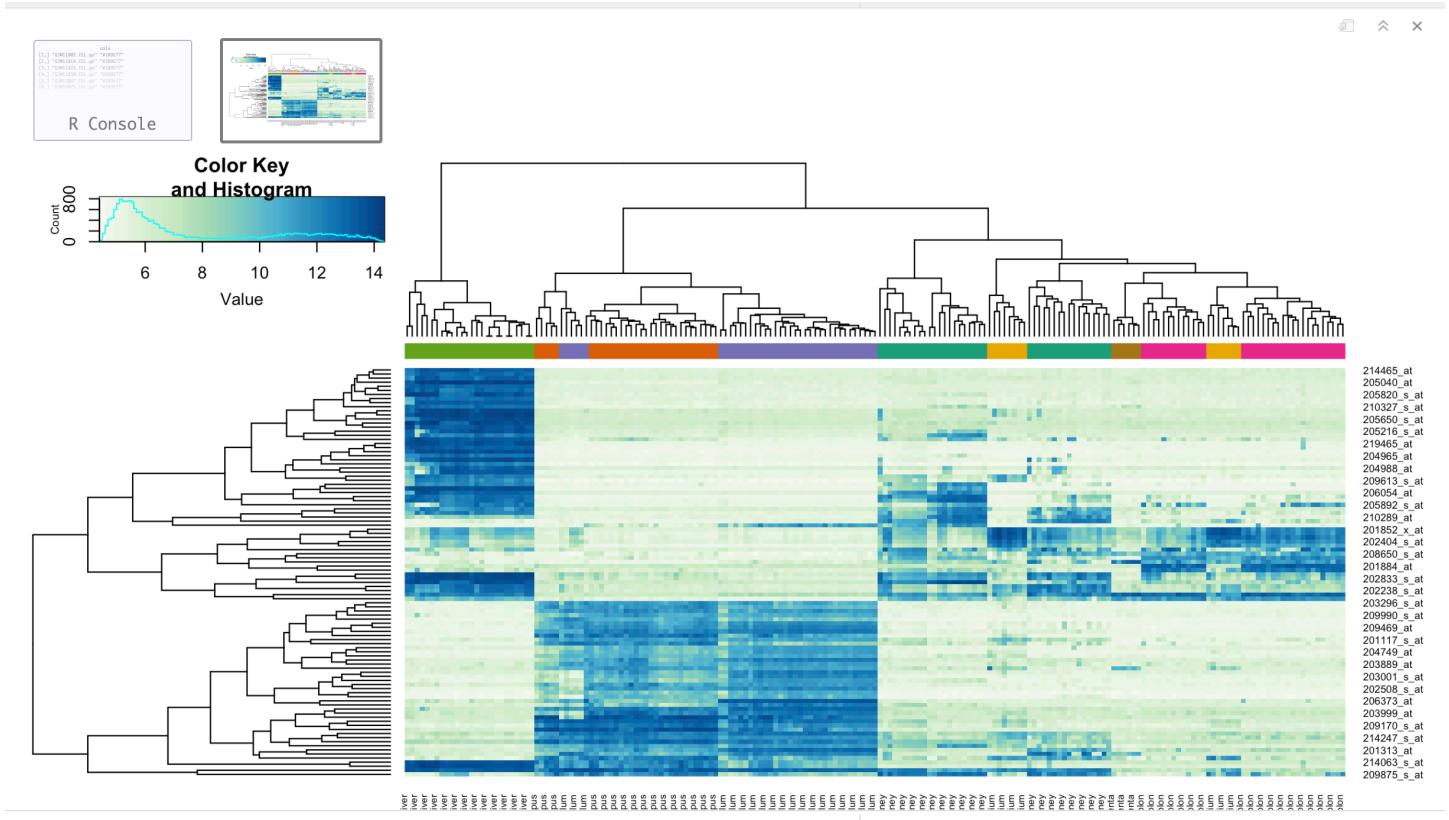
```
```{r}
idx_100 <- order(-rv)[1:100]
````
```



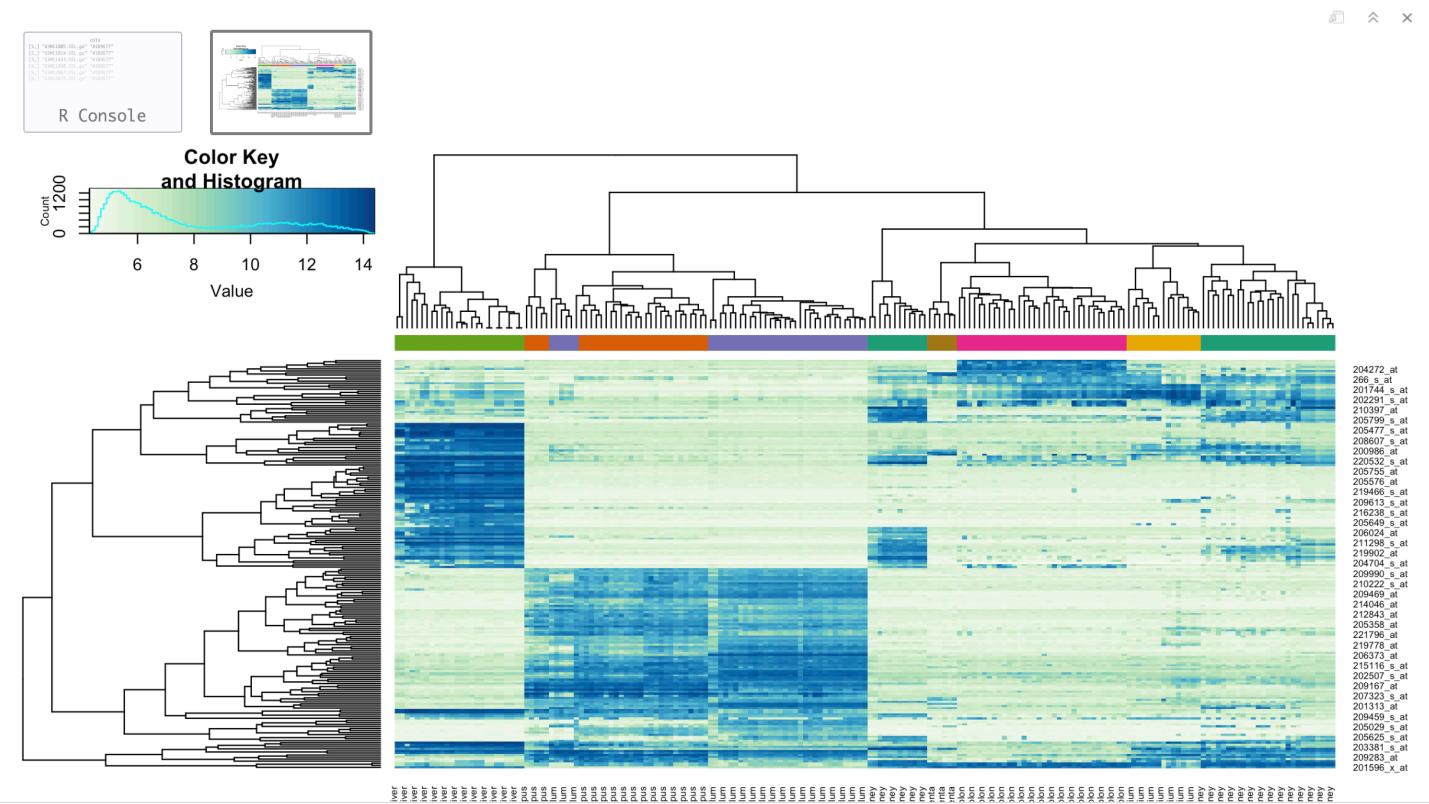
```
```{r heatmap.2,message=FALSE,fig.cap="Heatmap created using the 100 most variable genes and the function heatmap.2.",
fig.width=10.5,fig.height=5.25,eval=FALSE}
library(gplots) ##Available from CRAN
cols <- palette(brewer.pal(8, "Dark2"))[as.fumeric(tissue)]
head(cbind(colnames(e),cols))
heatmap.2(e[idx_100,], labCol=tissue,
          trace="none",
          ColSideColors=cols,
          col=hmcol)
````
```



100



200



```
```{r}
idx_200 <- order(-rv)[1:200]
```
```{r heatmap.2,message=FALSE,fig.cap="Heatmap created using the 100 most variable genes and the function heatmap.2.",fig.width=10.5,fig.height=5.25,eval=FALSE}
library(gplots) ##Available from CRAN
cols <- palette(brewer.pal(8, "Dark2"))[as.numeric(tissue)]
head(cbind(colnames(e),cols))
heatmap.2(e[idx_200,], labCol=tissue,
          trace="none",
          ColSideColors=cols,
          col=hmc)
```

```

#### #### Observation:

Using the 100 most variable genes to create a heatmap provides a clearer and more detailed view than the heatmap based on 40 genes. With 100 genes, the clustering is more pronounced, but the clusters remain similar to those observed in the 40-gene heatmap.

When using the 200 most variable genes, the heatmap shows more complexity and detail due to the increased data, but some clusters appear less distinct. The additional data introduces noise, causing some clusters to blend together or lose their clear separation, which might obscure patterns that were previously evident with fewer genes.

- How does the \*dendrogram\* on top of your heatmaps compare to that from previous hierarchical clustering analysis?

#### #### Observation:

The dendrogram\* on top of our heatmaps looks very similar to previous hierarchical clustering analysis.

```

##  system    aarch64, darwin20
##  ui        X11
##  language  (EN)
##  collate   en_US.UTF-8
##  ctype     en_US.UTF-8
##  tz        America/Chicago
##  date      2024-05-06
##  pandoc   3.1.1 @ /Applications/RStudio.app/Contents/Resources/app/quarto/bin/tools/ (via rmarkdown)
##
## - Packages -----
##   package * version date (UTC) lib source
##   cachem   1.0.8   2023-05-01 [1] CRAN (R 4.4.0)
##   cli       3.6.2   2023-12-11 [1] CRAN (R 4.4.0)
##   devtools  2.4.5   2022-10-11 [1] CRAN (R 4.4.0)
##   digest    0.6.35  2024-03-11 [1] CRAN (R 4.4.0)
##   ellipsis  0.3.2   2021-04-29 [1] CRAN (R 4.4.0)
##   evaluate  0.23    2023-11-01 [1] CRAN (R 4.4.0)
##   fastmap   1.1.1   2023-02-24 [1] CRAN (R 4.4.0)
##   fs        1.6.4   2024-04-25 [1] CRAN (R 4.4.0)
##   glue      1.7.0   2024-01-09 [1] CRAN (R 4.4.0)
##   htmltools  0.5.8.1 2024-04-04 [1] CRAN (R 4.4.0)
##   htmlwidgets 1.6.4   2023-12-06 [1] CRAN (R 4.4.0)
##   httpuv    1.6.15  2024-03-26 [1] CRAN (R 4.4.0)
##   knitr     1.46    2024-04-06 [1] CRAN (R 4.4.0)
##   later     1.3.2   2023-12-06 [1] CRAN (R 4.4.0)
##   lifecycle 1.0.4   2023-11-07 [1] CRAN (R 4.4.0)
##   magrittr   2.0.3   2022-03-30 [1] CRAN (R 4.4.0)
##   memoise   2.0.1   2021-11-26 [1] CRAN (R 4.4.0)
##   mime      0.12    2021-09-28 [1] CRAN (R 4.4.0)
##   miniUI    0.1.1.1 2018-05-18 [1] CRAN (R 4.4.0)
##   pkgbuild  1.4.4   2024-03-17 [1] CRAN (R 4.4.0)
##   pkgload   1.3.4   2024-01-16 [1] CRAN (R 4.4.0)
##   profvis   0.3.8   2023-05-02 [1] CRAN (R 4.4.0)
##   promises  1.3.0   2024-04-05 [1] CRAN (R 4.4.0)
##   purrr    1.0.2   2023-08-10 [1] CRAN (R 4.4.0)
##   R6        2.5.1   2021-08-19 [1] CRAN (R 4.4.0)
##   Rcpp      1.0.12  2024-01-09 [1] CRAN (R 4.4.0)
##   remotes   2.5.0   2024-03-17 [1] CRAN (R 4.4.0)
##   rlang     1.1.3   2024-01-10 [1] CRAN (R 4.4.0)
##   rmarkdown  2.26    2024-03-05 [1] CRAN (R 4.4.0)
##   rstudioapi 0.16.0  2024-03-24 [1] CRAN (R 4.4.0)
##   sessioninfo 1.2.2   2021-12-06 [1] CRAN (R 4.4.0)
##   shiny     1.8.1.1  2024-04-02 [1] CRAN (R 4.4.0)
##   stringi   1.8.3   2023-12-11 [1] CRAN (R 4.4.0)
##   stringr   1.5.1   2023-11-14 [1] CRAN (R 4.4.0)
##   urlchecker 1.0.1   2021-11-30 [1] CRAN (R 4.4.0)
##   usethis   2.2.3   2024-02-19 [1] CRAN (R 4.4.0)
##   vctrs     0.6.5   2023-12-01 [1] CRAN (R 4.4.0)
##   xfun      0.43    2024-03-25 [1] CRAN (R 4.4.0)
##   xtable    1.8-4   2019-04-21 [1] CRAN (R 4.4.0)
##   yaml      2.3.8   2023-12-11 [1] CRAN (R 4.4.0)
##
## [1] /Library/Frameworks/R.framework/Versions/4.4-arm64/Resources/library
##

```

## -----