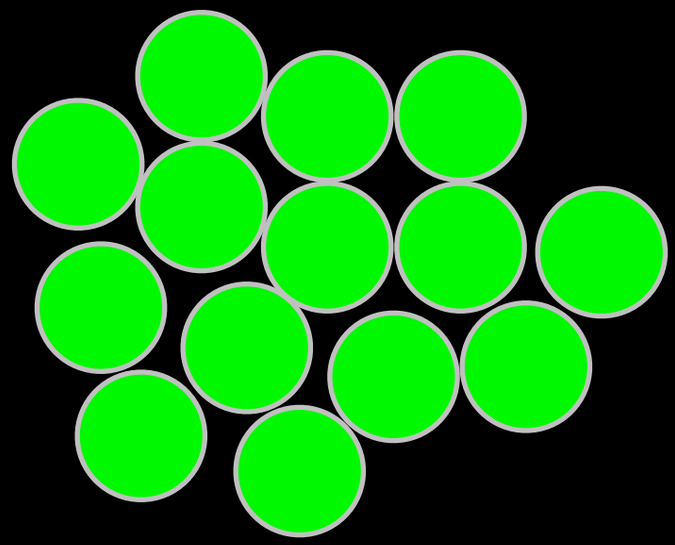
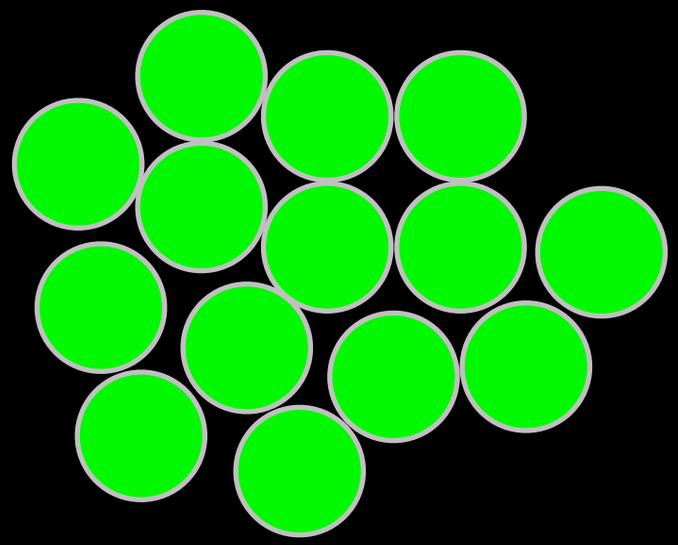
Please do your formal course evaluations!

PCA: The absolute basics

Bunch of “normal” cells

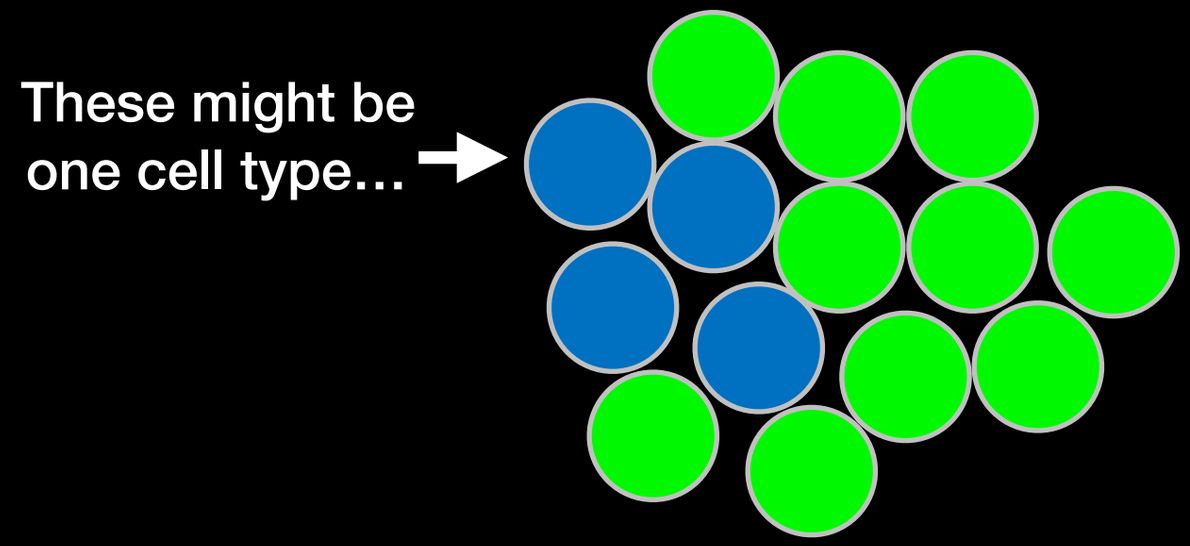


Bunch of “normal” cells



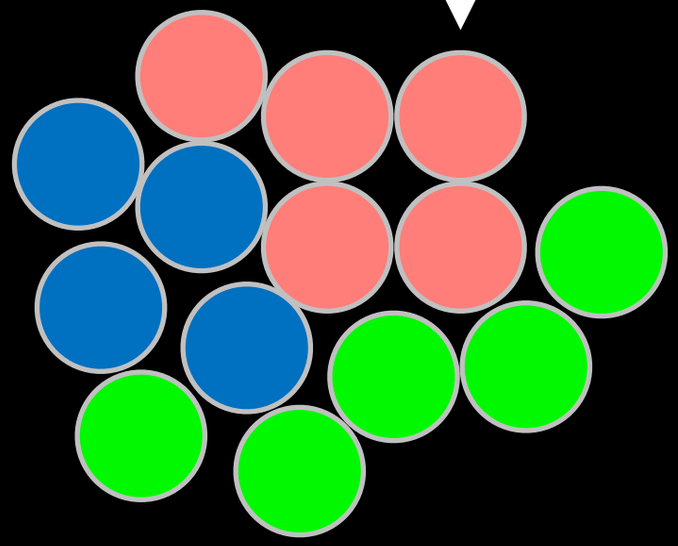
Even though they look the same we suspect that there are differences…

These might be one cell type…



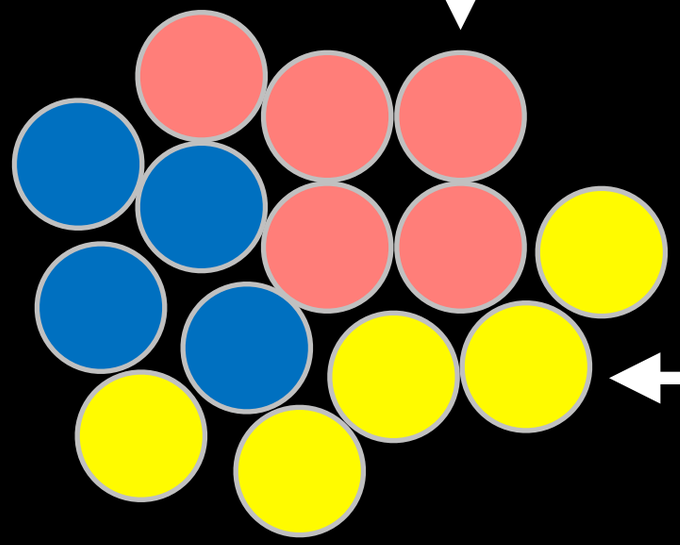
These another
cell type…

These might be
one cell type…

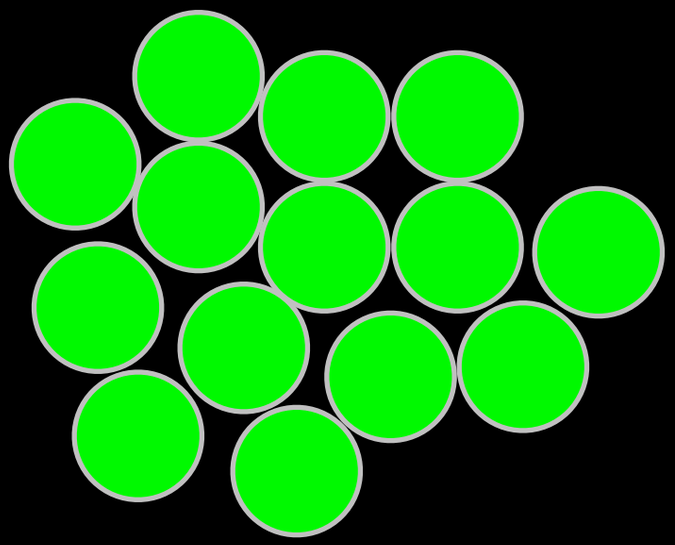


These another cell type…

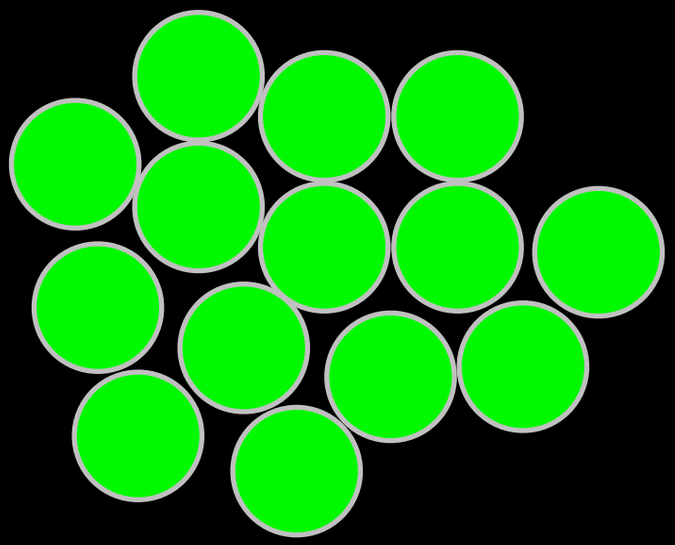
These might be one cell type…



These might be a third cell type…



Unfortunately we can’t observe
the differences visually



Unfortunately we can’t observe
the differences visually

So we sequence the mRNA in each
cell to identify which genes are
active and at what levels.

Here is the data…

	Cell1	Cell2	Cell3	Cell4	…
Gene1	3	0.25	2.8	0.1	…
Gene2	2.9	0.8	2.2	1.8	…
Gene3	2.2	1	1.5	3.2	…
Gene4	2	1.4	2	0.3	…
Gene5	1.3	1.6	1.6	0	…
Gene6	1.5	2	2.1	3	…
Gene7	1.1	2.2	1.2	2.8	…
Gene8	1	2.7	0.9	0.3	…
Gene9	0.4	3	0.6	0.1	…

Each column shows how much each gene is transcribed in each cell

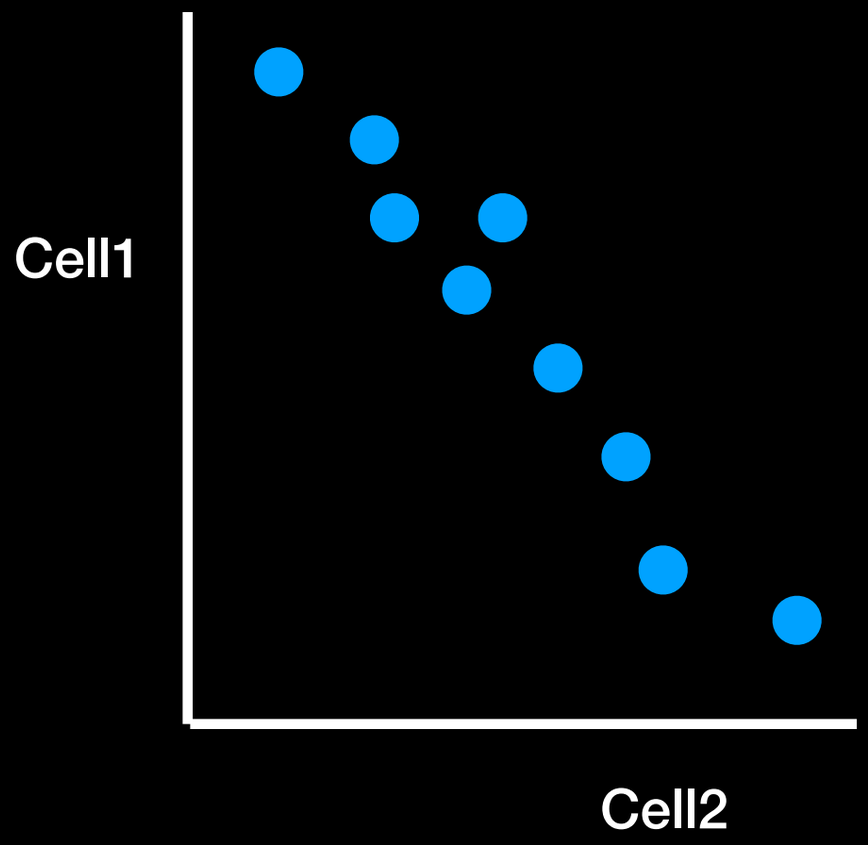
Here is the data…

	Cell1	Cell2	Cell3	Cell4	…
Gene1	3	0.25	2.8	0.1	…
Gene2	2.9	0.8	2.2	1.8	…
Gene3	2.2	1	1.5	3.2	…
Gene4	2	1.4	2	0.3	…
Gene5	1.3	1.6	1.6	0	…
Gene6	1.5	2	2.1	3	…
Gene7	1.1	2.2	1.2	2.8	…
Gene8	1	2.7	0.9	0.3	…
Gene9	0.4	3	0.6	0.1	…

For now lets consider only two cells

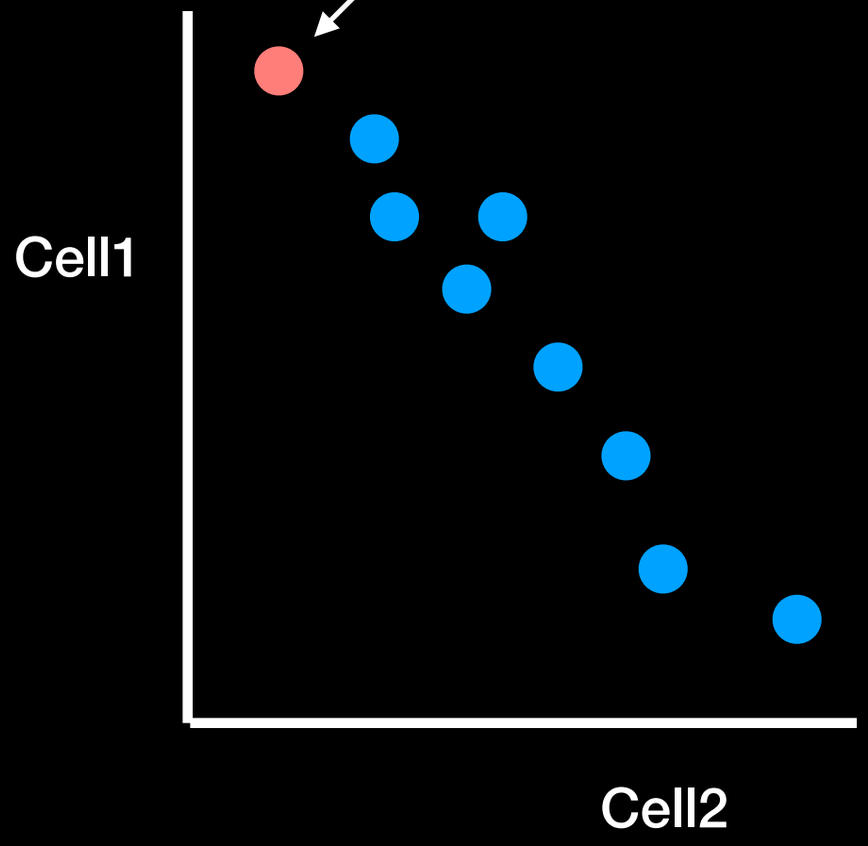
	Cell1	Cell2
Gene1	3	0.25
Gene2	2.9	0.8
Gene3	2.2	1
Gene4	2	1.4
Gene5	1.3	1.6
Gene6	1.5	2
Gene7	1.1	2.2
Gene8	1	2.7
Gene9	0.4	3

We have just 2 cells so we can plot the measurements for each gene



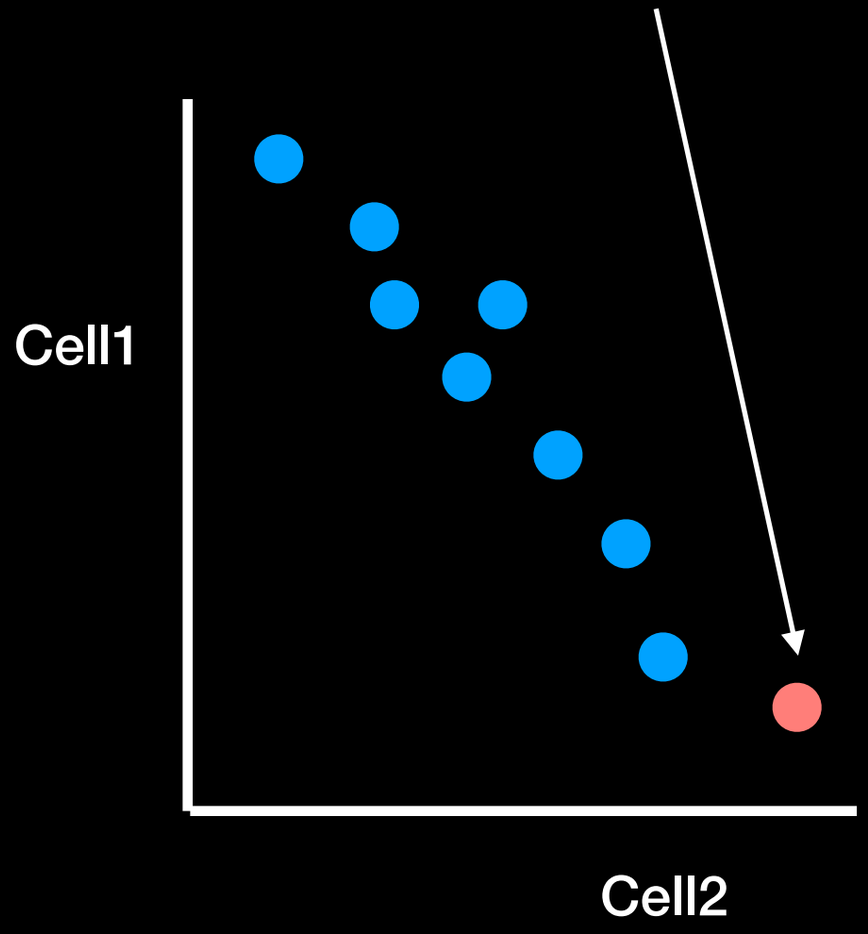
	Cell1	Cell2
Gene1	3	0.25
Gene2	2.9	0.8
Gene3	2.2	1
Gene4	2	1.4
Gene5	1.3	1.6
Gene6	1.5	2
Gene7	1.1	2.2
Gene8	1	2.7
Gene9	0.4	3

This gene (Gene1) is highly transcribed in Cell1 and lowly transcribed in Cell2…



	Cell1	Cell2
Gene1	3	0.25
Gene2	2.9	0.8
Gene3	2.2	1
Gene4	2	1.4
Gene5	1.3	1.6
Gene6	1.5	2
Gene7	1.1	2.2
Gene8	1	2.7
Gene9	0.4	3

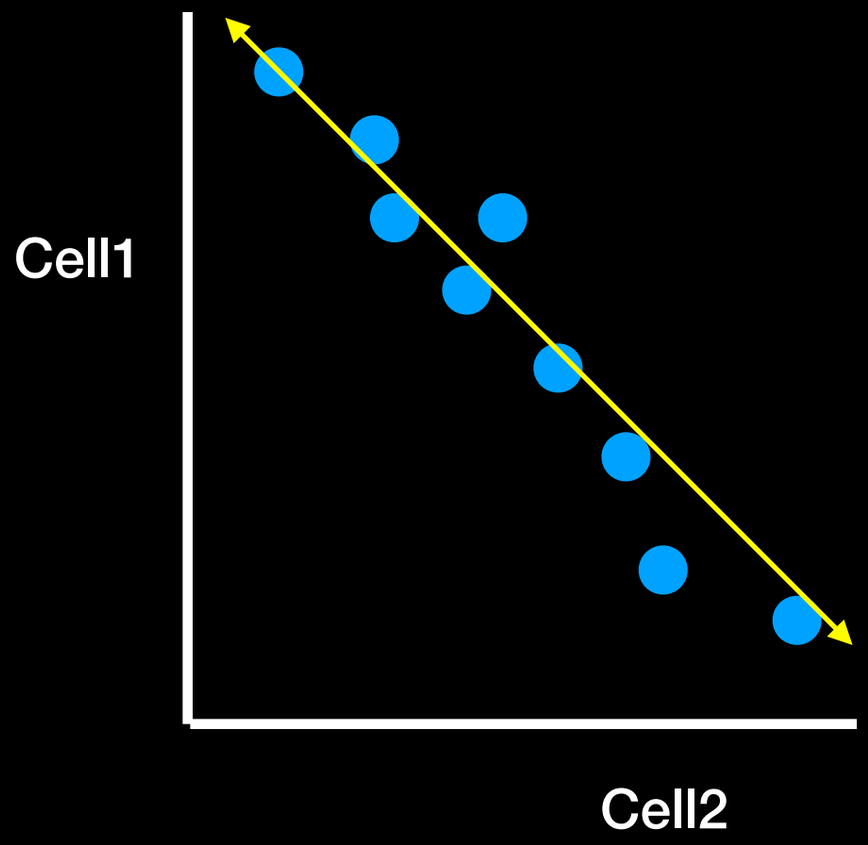
This gene (Gene9) is lowly transcribed in Cell1 and highly transcribed in Cell2…



	Cell1	Cell2
Gene1	3	0.25
Gene2	2.9	0.8
Gene3	2.2	1
Gene4	2	1.4
Gene5	1.3	1.6
Gene6	1.5	2
Gene7	1.1	2.2
Gene8	1	2.7
Gene9	0.4	3

In generel, Cell1 and Cell2 have an inverse correlation.

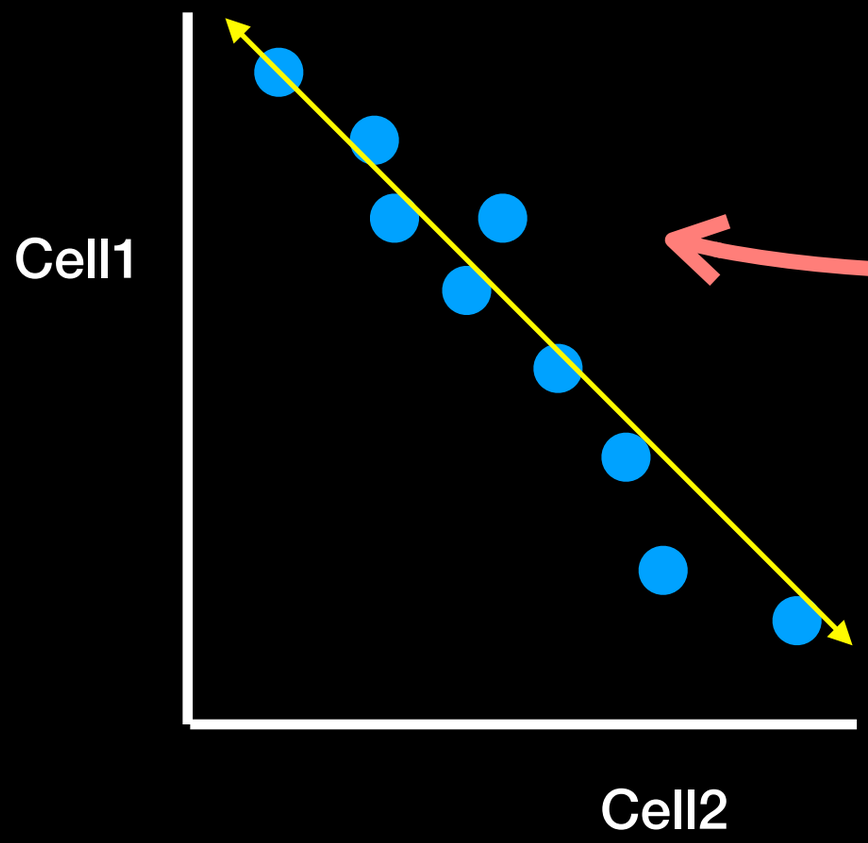
This suggests that they may be two different types of cells as they are using different genes



	Cell1	Cell2
Gene1	3	0.25
Gene2	2.9	0.8
Gene3	2.2	1
Gene4	2	1.4
Gene5	1.3	1.6
Gene6	1.5	2
Gene7	1.1	2.2
Gene8	1	2.7
Gene9	0.4	3

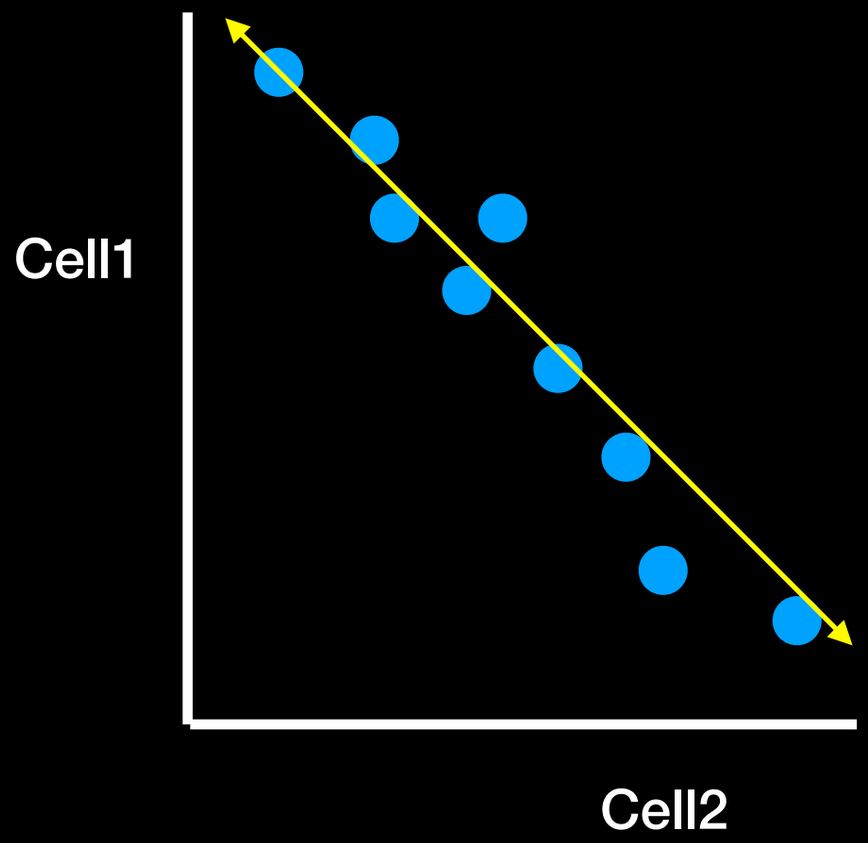
Now lets imagine there are three cells

	Cell1	Cell2	Cell3
Gene1	3	0.25	2.8
Gene2	2.9	0.8	2.2
Gene3	2.2	1	1.5
Gene4	2	1.4	2
Gene5	1.3	1.6	1.6
Gene6	1.5	2	2.1
Gene7	1.1	2.2	1.2
Gene8	1	2.7	0.9
Gene9	0.4	3	0.6



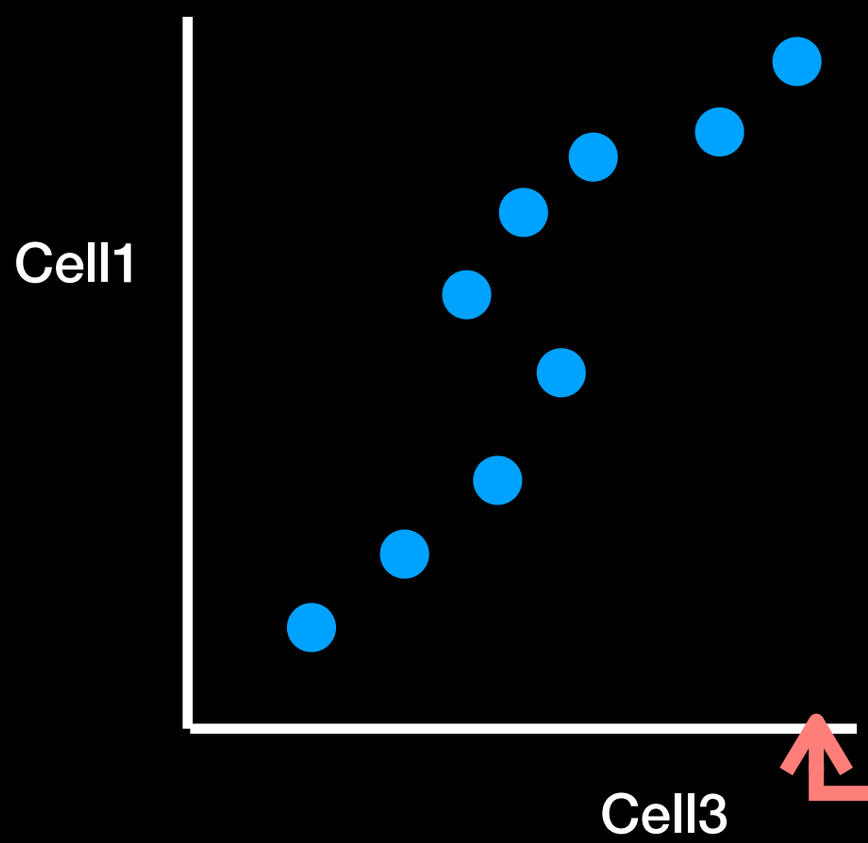
We have already seen how we can plot the first two cells to see how closely related they are

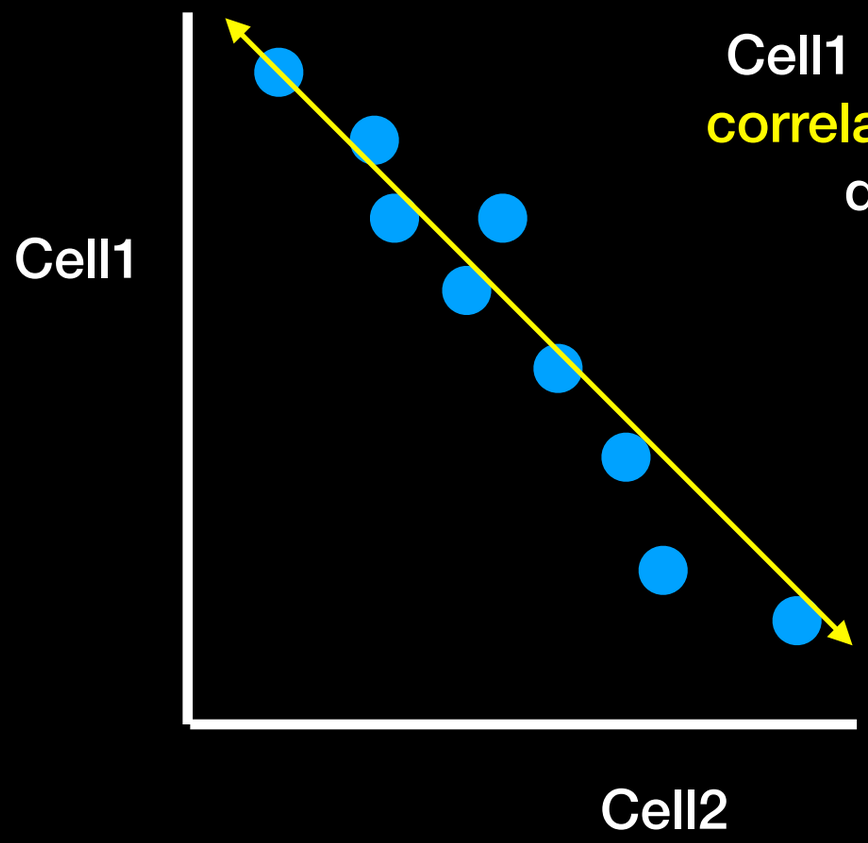
	Cell1	Cell2	Cell3
Gene1	3	0.25	2.8
Gene2	2.9	0.8	2.2
Gene3	2.2	1	1.5
Gene4	2	1.4	2
Gene5	1.3	1.6	1.6
Gene6	1.5	2	2.1
Gene7	1.1	2.2	1.2
Gene8	1	2.7	0.9
Gene9	0.4	3	0.6



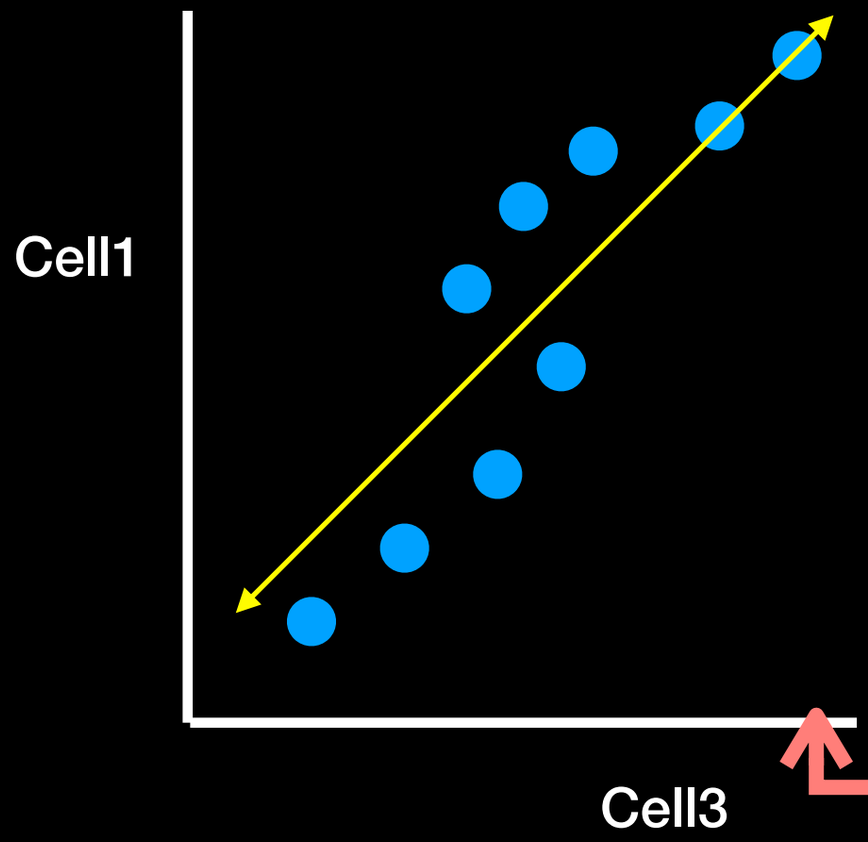
Now we can also compare
Cell1 to Cell3

	Cell1	Cell2	Cell3
Gene1	3	0.25	2.8
Gene2	2.9	0.8	2.2
Gene3	2.2	1	1.5
Gene4	2	1.4	2
Gene5	1.3	1.6	1.6
Gene6	1.5	2	2.1
Gene7	1.1	2.2	1.2
Gene8	1	2.7	0.9
Gene9	0.4	3	0.6

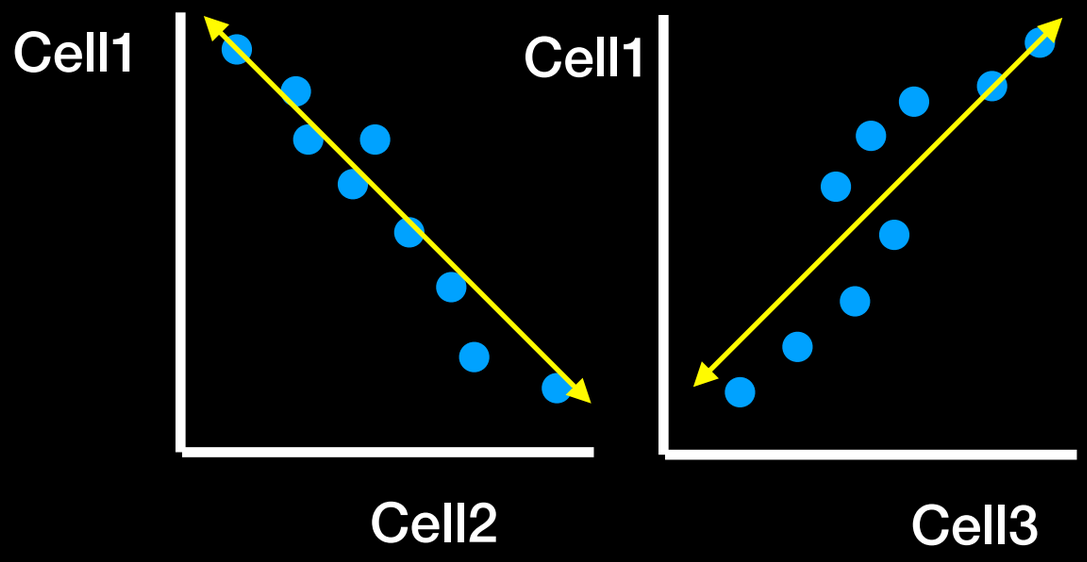




Cell1 and Cell3 are positively correlated suggesting they are doing similar things



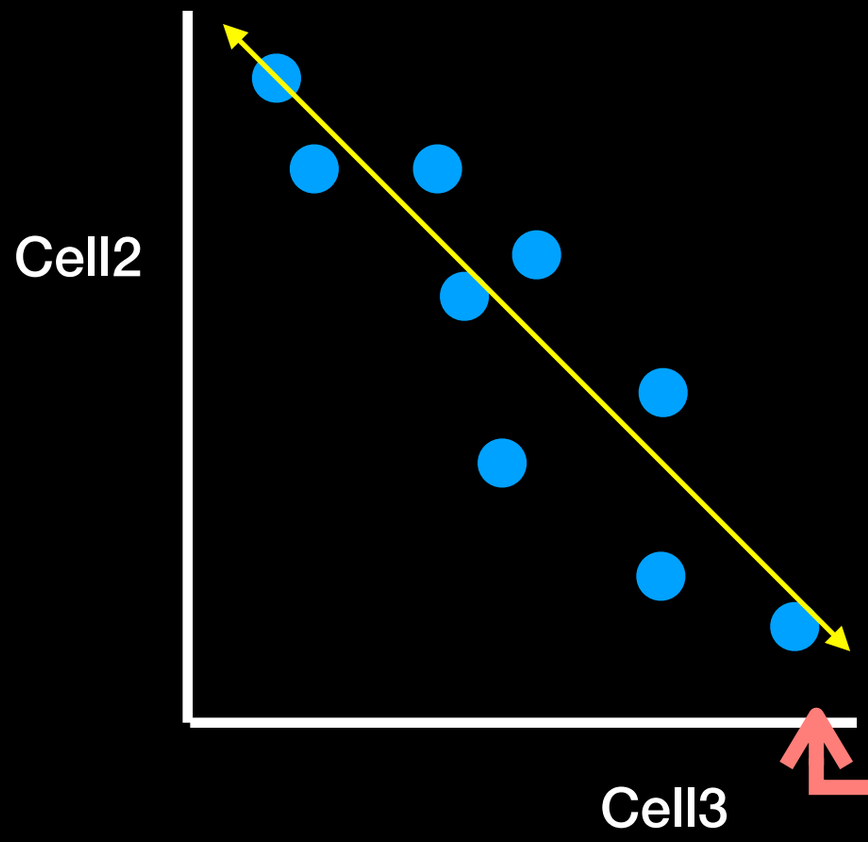
	Cell1	Cell2	Cell3
Gene1	3	0.25	2.8
Gene2	2.9	0.8	2.2
Gene3	2.2	1	1.5
Gene4	2	1.4	2
Gene5	1.3	1.6	1.6
Gene6	1.5	2	2.1
Gene7	1.1	2.2	1.2
Gene8	1	2.7	0.9
Gene9	0.4	3	0.6



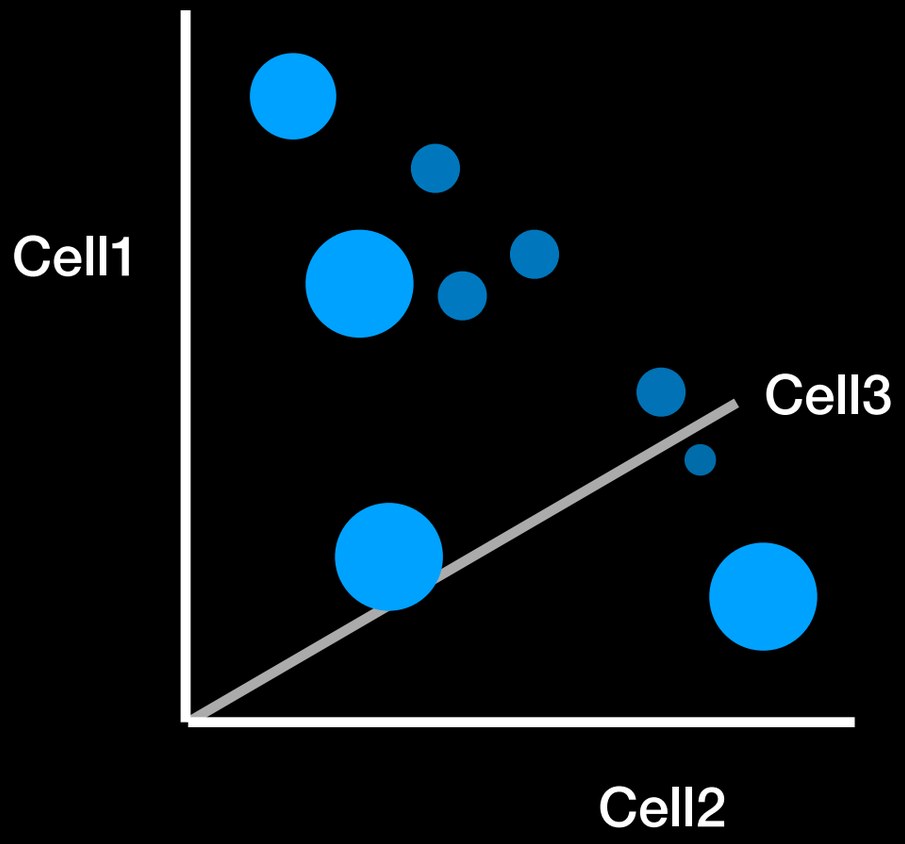
We can also compare Cell2 to Cell3…

	Cell1	Cell2	Cell3
Gene1	3	0.25	2.8
Gene2	2.9	0.8	2.2
Gene3	2.2	1	1.5
Gene4	2	1.4	2
Gene5	1.3	1.6	1.6
Gene6	1.5	2	2.1
Gene7	1.1	2.2	1.2
Gene8	1	2.7	0.9
Gene9	0.4	3	0.6

The inverse correlation suggests that Cell2 is doing something different from Cell3



Alternatively, we could try to plot all 3 cells at once on a 3-dimensional graph.



	Cell1	Cell2	Cell3
Gene1	3	0.25	2.8
Gene2	2.9	0.8	2.2
Gene3	2.2	1	1.5
Gene4	2	1.4	2
Gene5	1.3	1.6	1.6
Gene6	1.5	2	2.1
Gene7	1.1	2.2	1.2
Gene8	1	2.7	0.9
Gene9	0.4	3	0.6

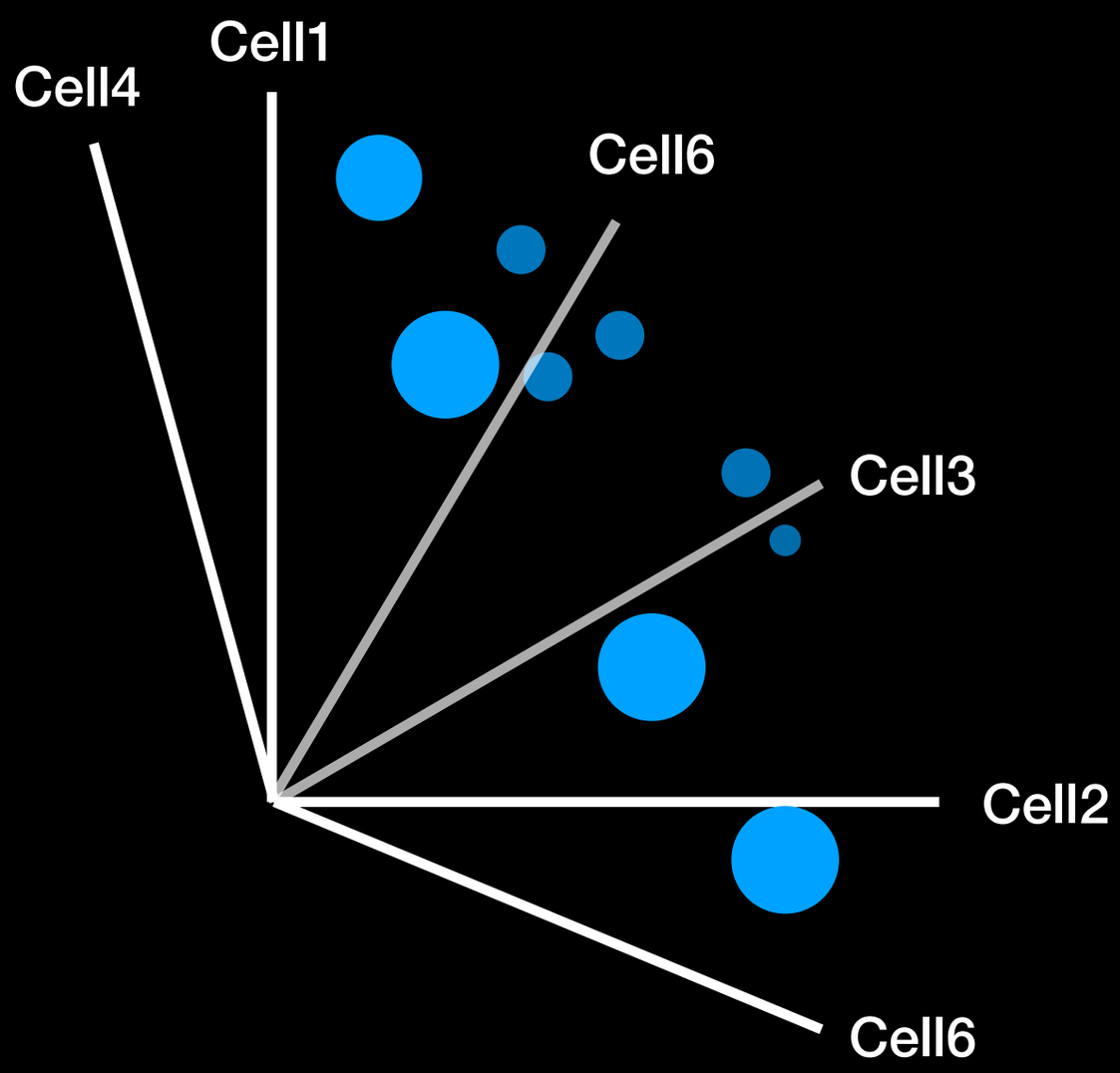
But what if we have 4 or more Cells?

	Cell1	Cell2	Cell3	Cell4	…
Gene1	3	0.25	2.8	0.1	…
Gene2	2.9	0.8	2.2	1.8	…
Gene3	2.2	1	1.5	3.2	…
Gene4	2	1.4	2	0.3	…
Gene5	1.3	1.6	1.6	0	…
Gene6	1.5	2	2.1	3	…
Gene7	1.1	2.2	1.2	2.8	…
Gene8	1	2.7	0.9	0.3	…
Gene9	0.4	3	0.6	0.1	…

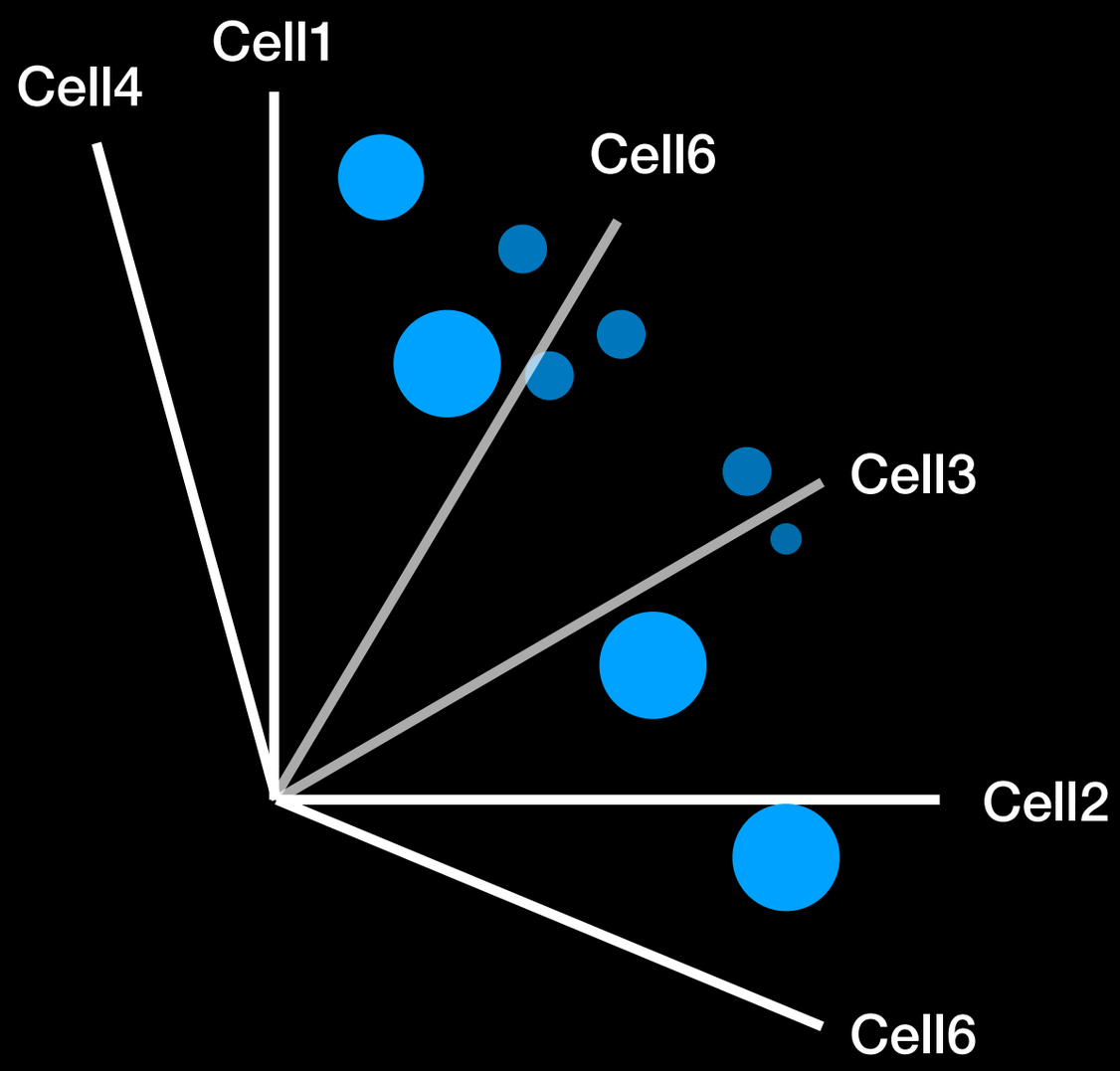
Draw lots of 2 cell plots and try to make sense of them all?

	Cell1	Cell2	Cell3	Cell4	…
Gene1	3	0.25	2.8	0.1	…
Gene2	2.9	0.8	2.2	1.8	…
Gene3	2.2	1	1.5	3.2	…
Gene4	2	1.4	2	0.3	…
Gene5	1.3	1.6	1.6	0	…
Gene6	1.5	2	2.1	3	…
Gene7	1.1	2.2	1.2	2.8	…
Gene8	1	2.7	0.9	0.3	…
Gene9	0.4	3	0.6	0.1	…

Or draw some crazy graph that has an axis for each cell and makes or brains hurt!

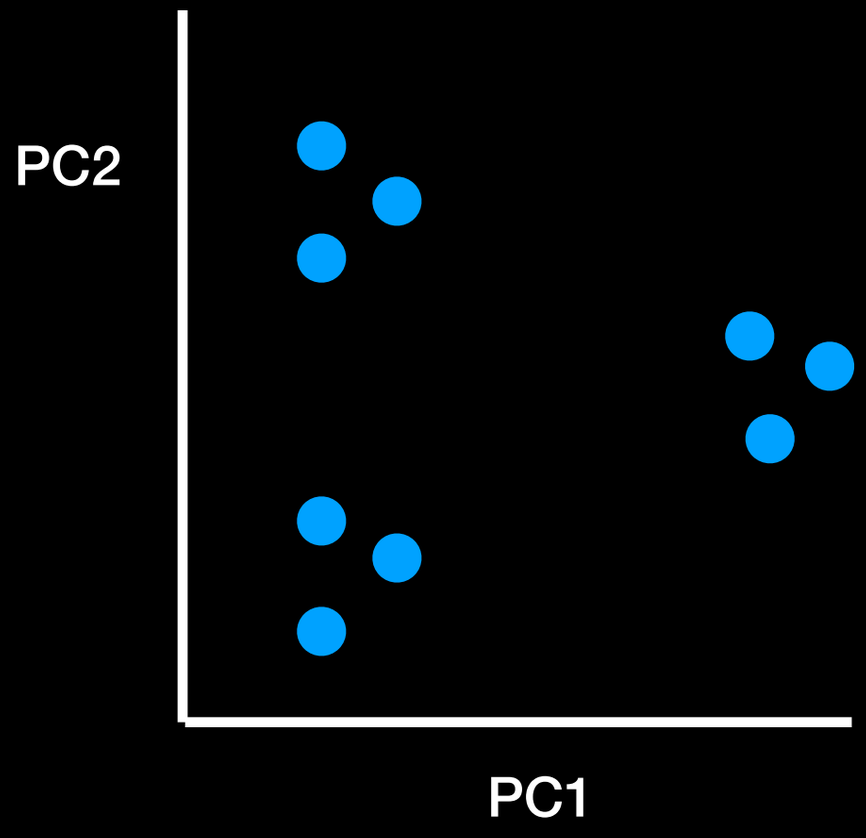


	Cell1	Cell2	Cell3	Cell4	…
Gene1	3	0.25	2.8	0.1	…
Gene2	2.9	0.8	2.2	1.8	…
Gene3	2.2	1	1.5	3.2	…
Gene4	2	1.4	2	0.3	…
Gene5	1.3	1.6	1.6	0	…
Gene6	1.5	2	2.1	3	…
Gene7	1.1	2.2	1.2	2.8	…
Gene8	1	2.7	0.9	0.3	…
Gene9	0.4	3	0.6	0.1	…



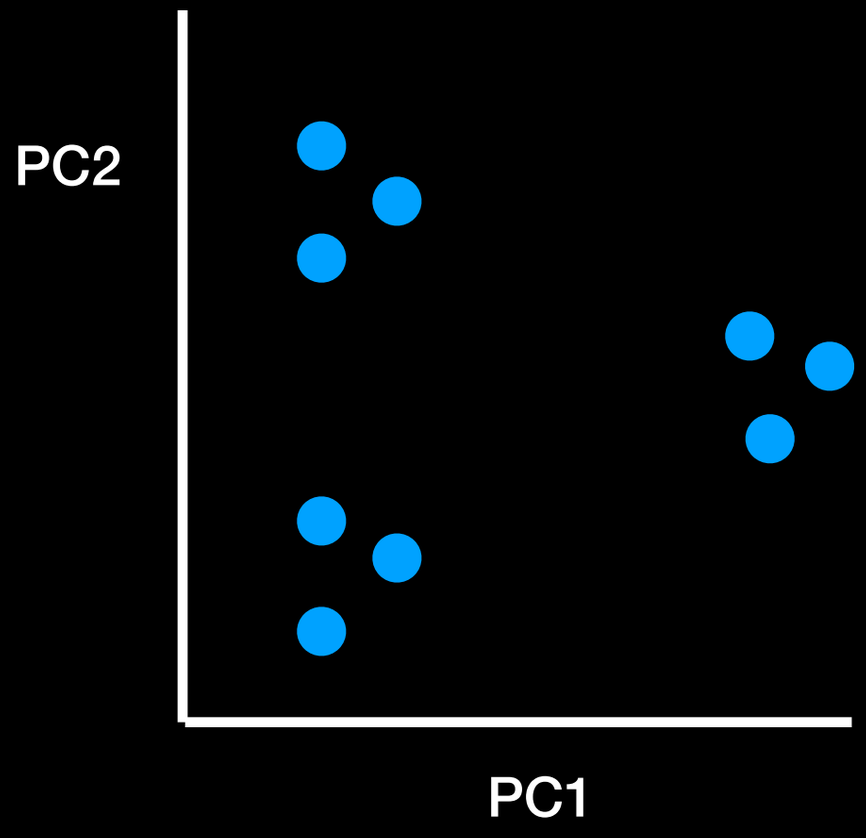
	Cell1	Cell2	Cell3	Cell4	…
Gene1	3	0.25	2.8	0.1	…
Gene2	2.9	0.8	2.2	1.8	…
Gene3	2.2	1	1.5	3.2	…
Gene4	2	1.4	2	0.3	…
Gene5	1.3	1.6	1.6	0	…
Gene6	1.5	2	2.1	3	…
Gene7	1.1	2.2	1.2	2.8	…
Gene8	1	2.7	0.9	0.3	…
Gene9	0.4	3	0.6	0.1	…

Enter Principal Component Analysis (PCA)



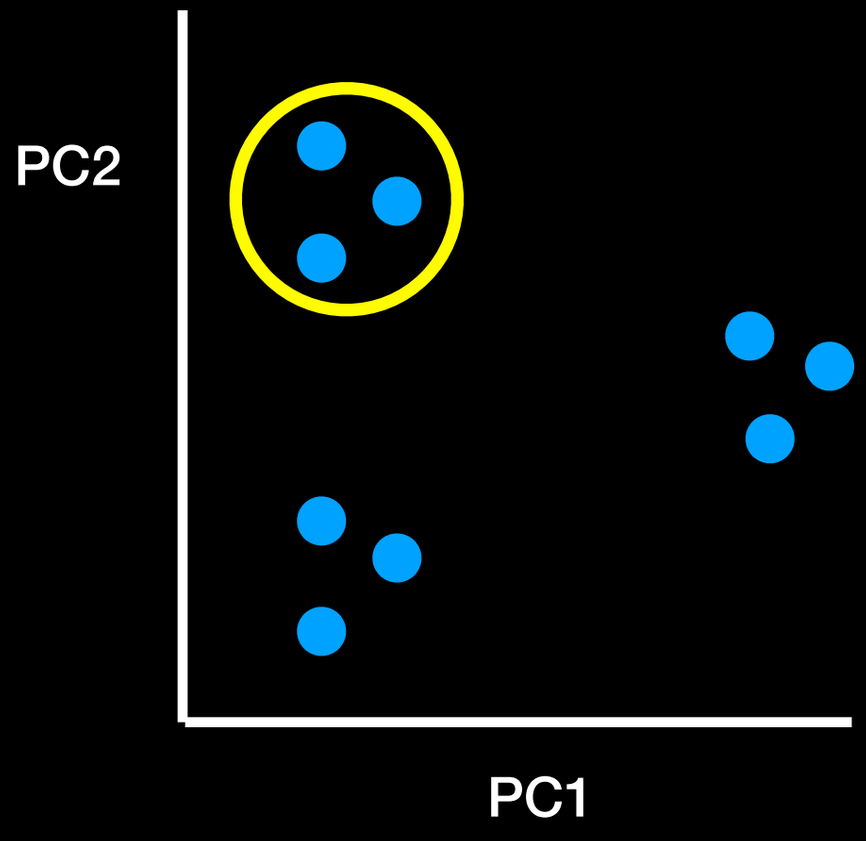
	Cell1	Cell2	Cell3	Cell4	…
Gene1	3	0.25	2.8	0.1	…
Gene2	2.9	0.8	2.2	1.8	…
Gene3	2.2	1	1.5	3.2	…
Gene4	2	1.4	2	0.3	…
Gene5	1.3	1.6	1.6	0	…
Gene6	1.5	2	2.1	3	…
Gene7	1.1	2.2	1.2	2.8	…
Gene8	1	2.7	0.9	0.3	…
Gene9	0.4	3	0.6	0.1	…

PCA converts the correlations (or lack there of) among all cells into a representation we can more readily interpret (e.g. a 2D graph!)



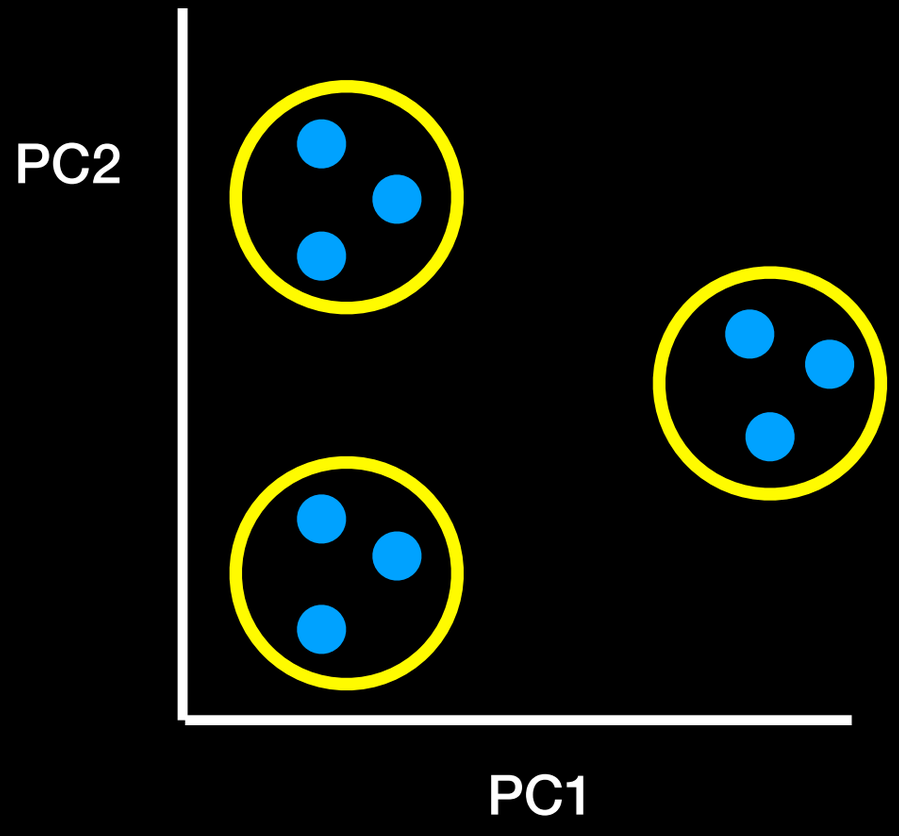
	Cell1	Cell2	Cell3	Cell4	…
Gene1	3	0.25	2.8	0.1	…
Gene2	2.9	0.8	2.2	1.8	…
Gene3	2.2	1	1.5	3.2	…
Gene4	2	1.4	2	0.3	…
Gene5	1.3	1.6	1.6	0	…
Gene6	1.5	2	2.1	3	…
Gene7	1.1	2.2	1.2	2.8	…
Gene8	1	2.7	0.9	0.3	…
Gene9	0.4	3	0.6	0.1	…

Cells that are highly correlated cluster together



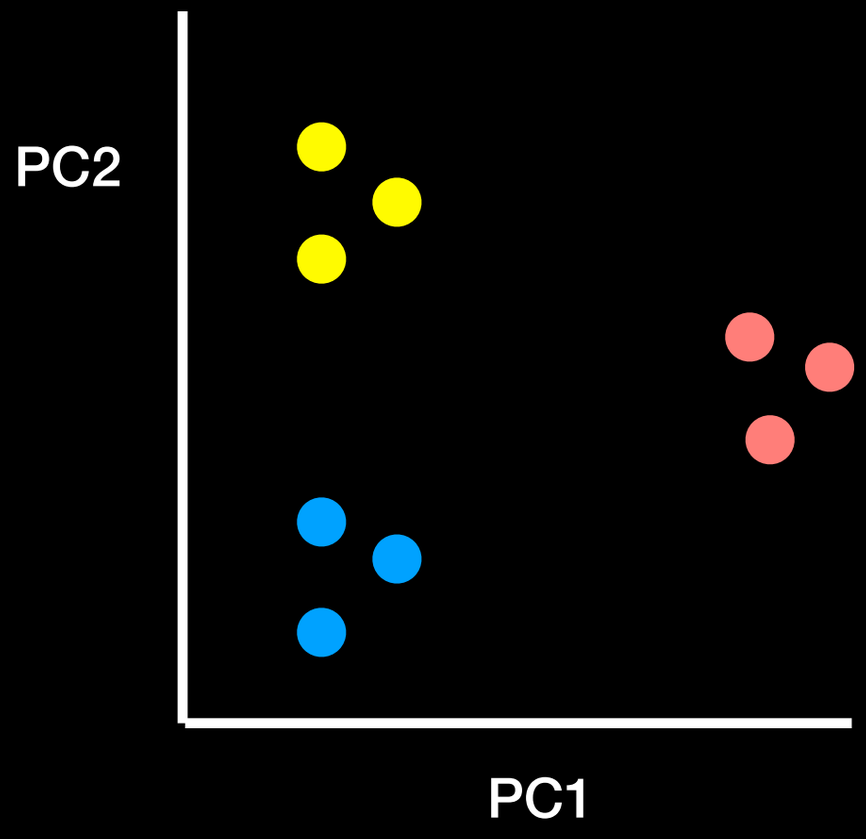
	Cell1	Cell2	Cell3	Cell4	…
Gene1	3	0.25	2.8	0.1	…
Gene2	2.9	0.8	2.2	1.8	…
Gene3	2.2	1	1.5	3.2	…
Gene4	2	1.4	2	0.3	…
Gene5	1.3	1.6	1.6	0	…
Gene6	1.5	2	2.1	3	…
Gene7	1.1	2.2	1.2	2.8	…
Gene8	1	2.7	0.9	0.3	…
Gene9	0.4	3	0.6	0.1	…

Cells that are highly correlated cluster together



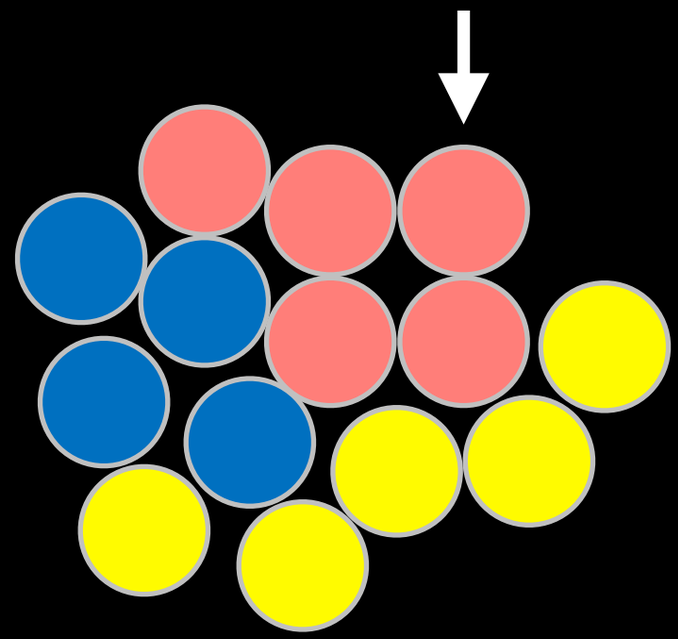
	Cell1	Cell2	Cell3	Cell4	…
Gene1	3	0.25	2.8	0.1	…
Gene2	2.9	0.8	2.2	1.8	…
Gene3	2.2	1	1.5	3.2	…
Gene4	2	1.4	2	0.3	…
Gene5	1.3	1.6	1.6	0	…
Gene6	1.5	2	2.1	3	…
Gene7	1.1	2.2	1.2	2.8	…
Gene8	1	2.7	0.9	0.3	…
Gene9	0.4	3	0.6	0.1	…

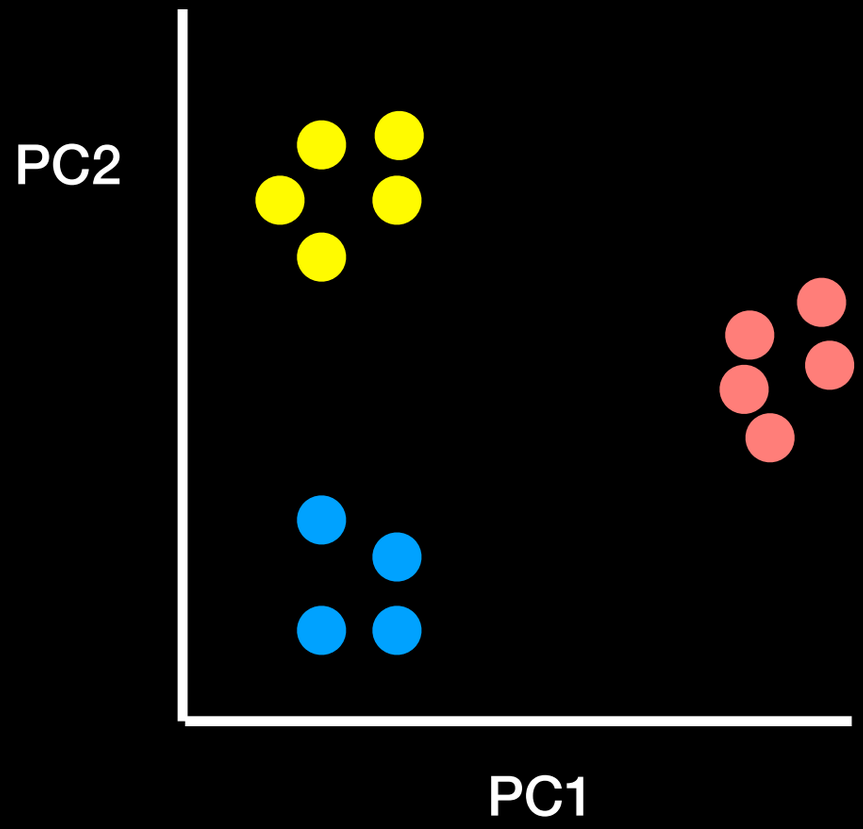
To make the clusters easier to see
we can color code them…

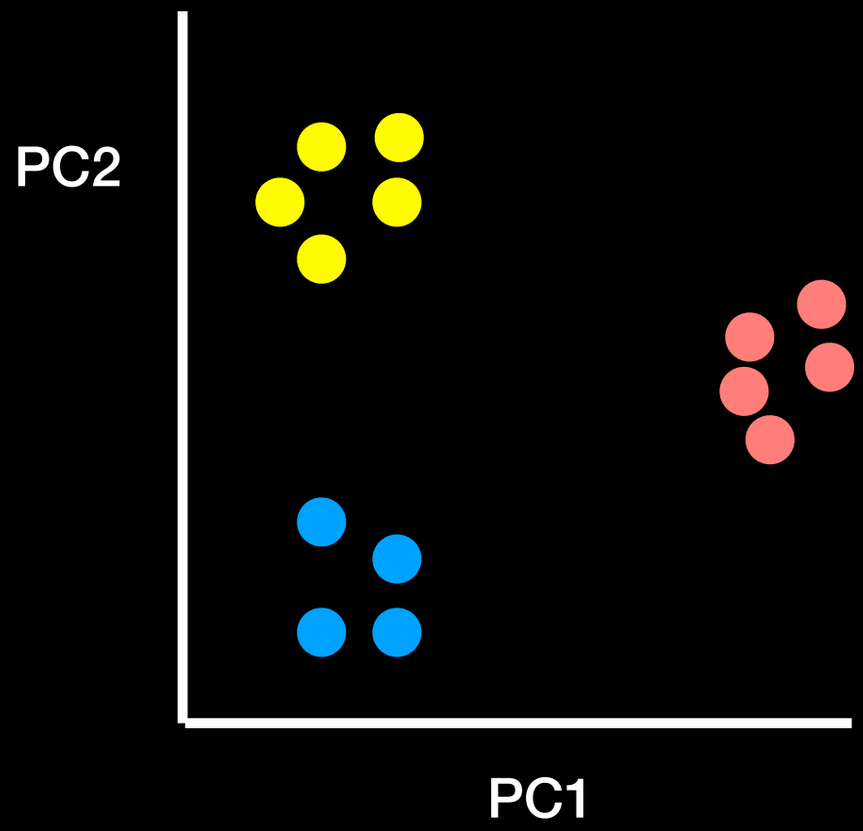


	Cell1	Cell2	Cell3	Cell4	…
Gene1	3	0.25	2.8	0.1	…
Gene2	2.9	0.8	2.2	1.8	…
Gene3	2.2	1	1.5	3.2	…
Gene4	2	1.4	2	0.3	…
Gene5	1.3	1.6	1.6	0	…
Gene6	1.5	2	2.1	3	…
Gene7	1.1	2.2	1.2	2.8	…
Gene8	1	2.7	0.9	0.3	…
Gene9	0.4	3	0.6	0.1	…

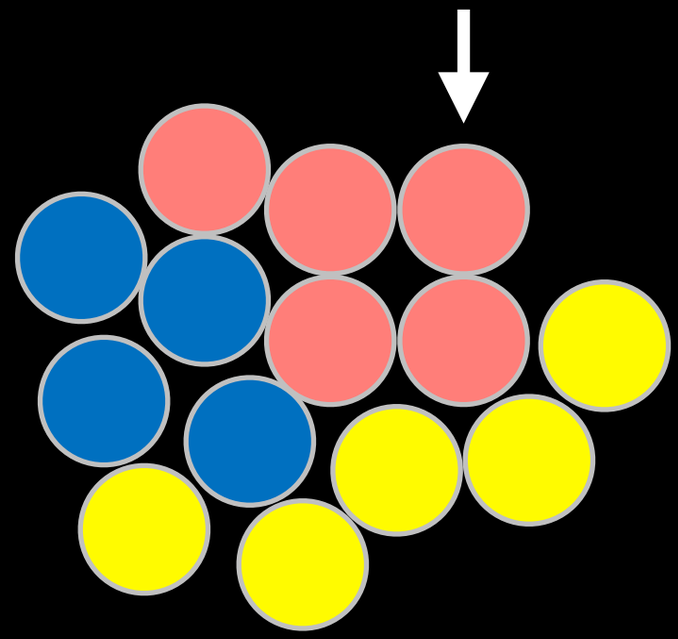
Once we have identified the clusters from our PCA results, we can go back to or original cells…



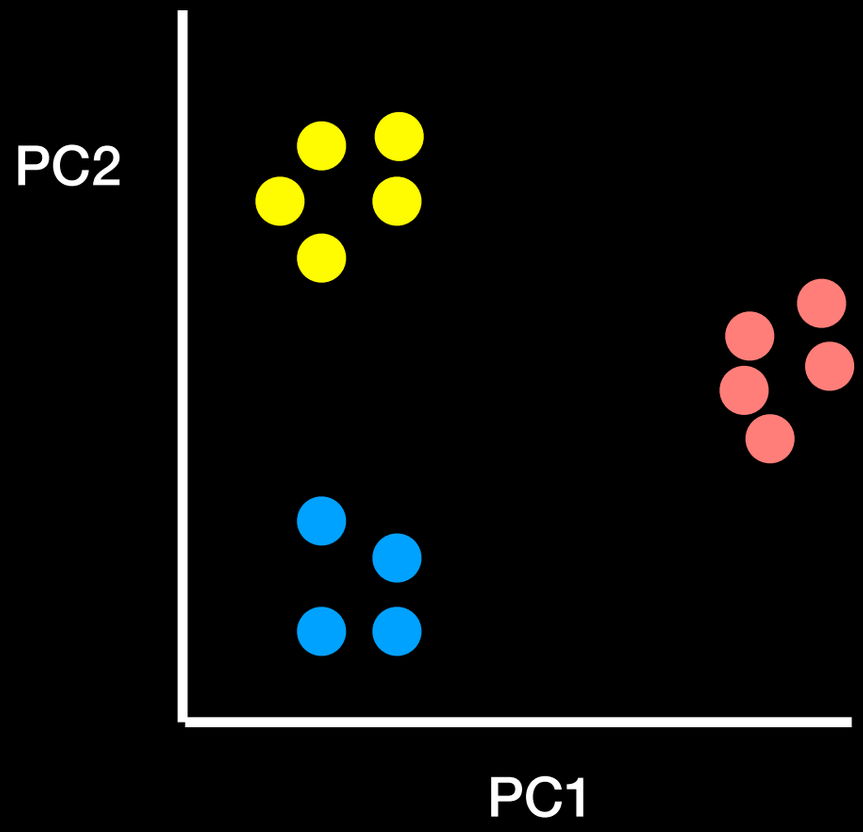




Once we have identified the clusters from our PCA results, we can go back to or original cells…



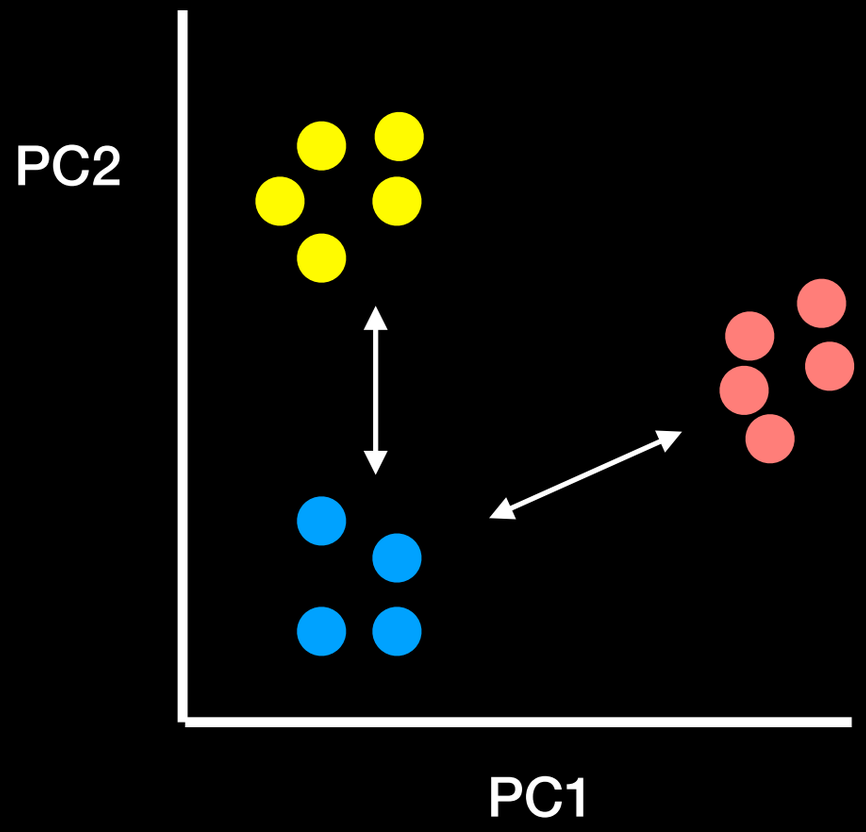
…and see they represent three different types of cells doing three different things with their genes!



<u>Some key points</u>:

The PCs (i.e. new plot axis) are ranked by their importance

So PC1 is more important than PC2 which in turn is more important than PC3 etc.

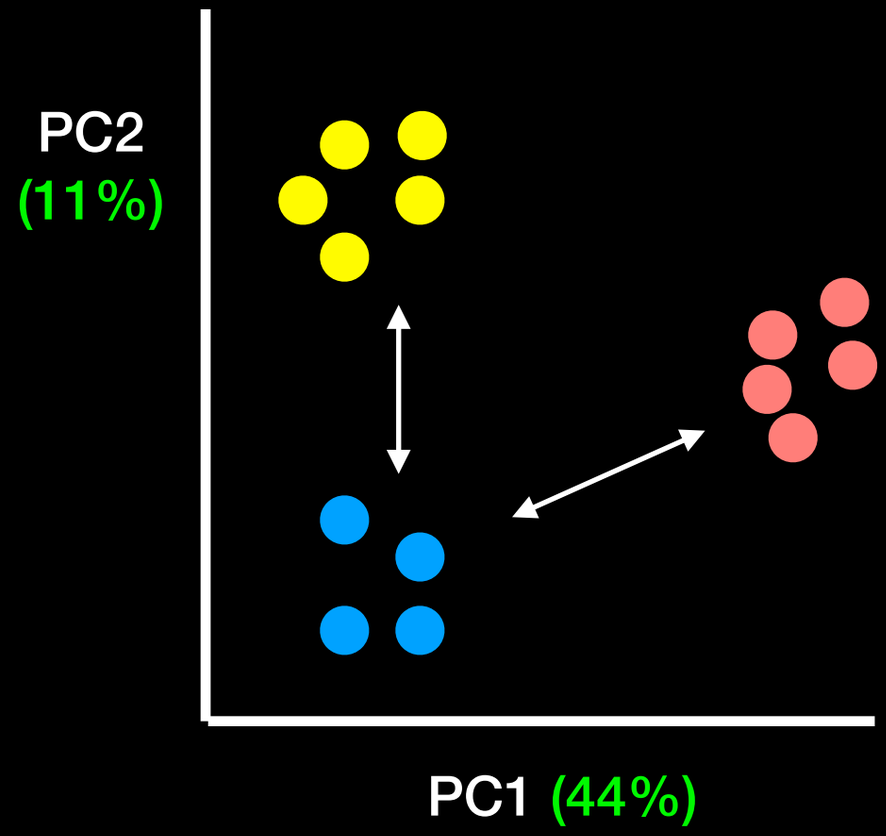


<u>Some key points</u>:

The PCs (i.e. new plot axis) are ranked by their importance

So PC1 is more important than PC2 which in turn is more important than PC3 etc.

So the red and blue cluster are more dissimilar than the yellow and blue clusters



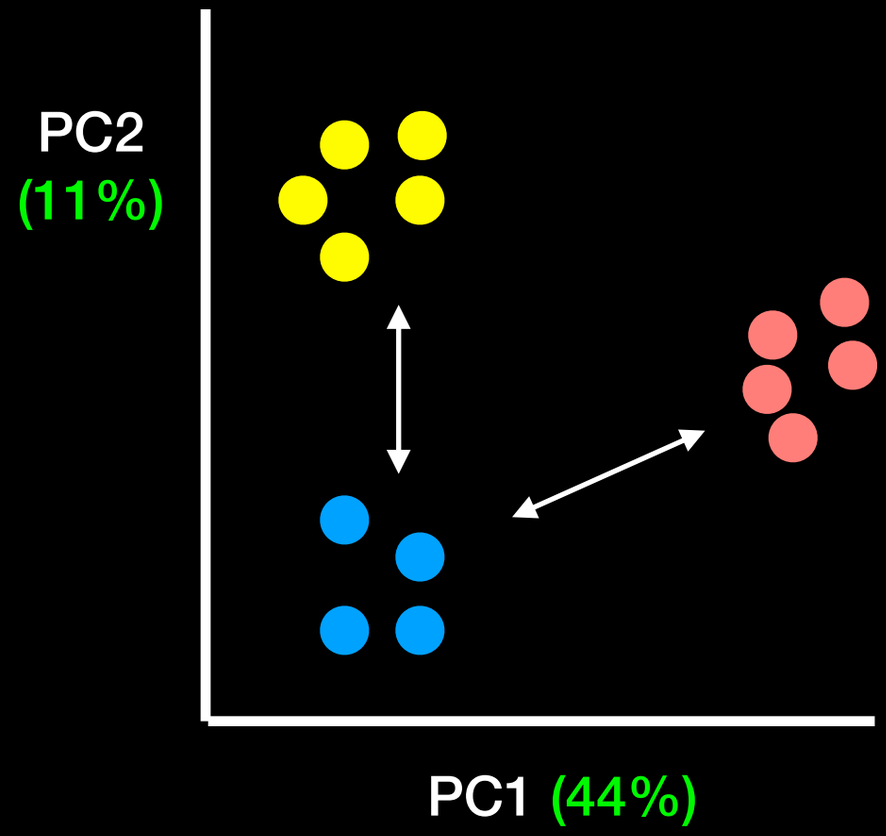
<u>Some key points</u>:

The PCs (i.e. new plot axis) are ranked by their importance

So PC1 is more important than PC2 which in turn is more important than PC3 etc.

So the red and blue cluster are more dissimilar than the yellow and blue clusters

The PCs (i.e. new plot axis) are ranked by the amount of variance in the original data (i.e. gene expression values) that they “capture”



<u>Some key points</u>:

The PCs (i.e. new plot axis) are ranked by their importance

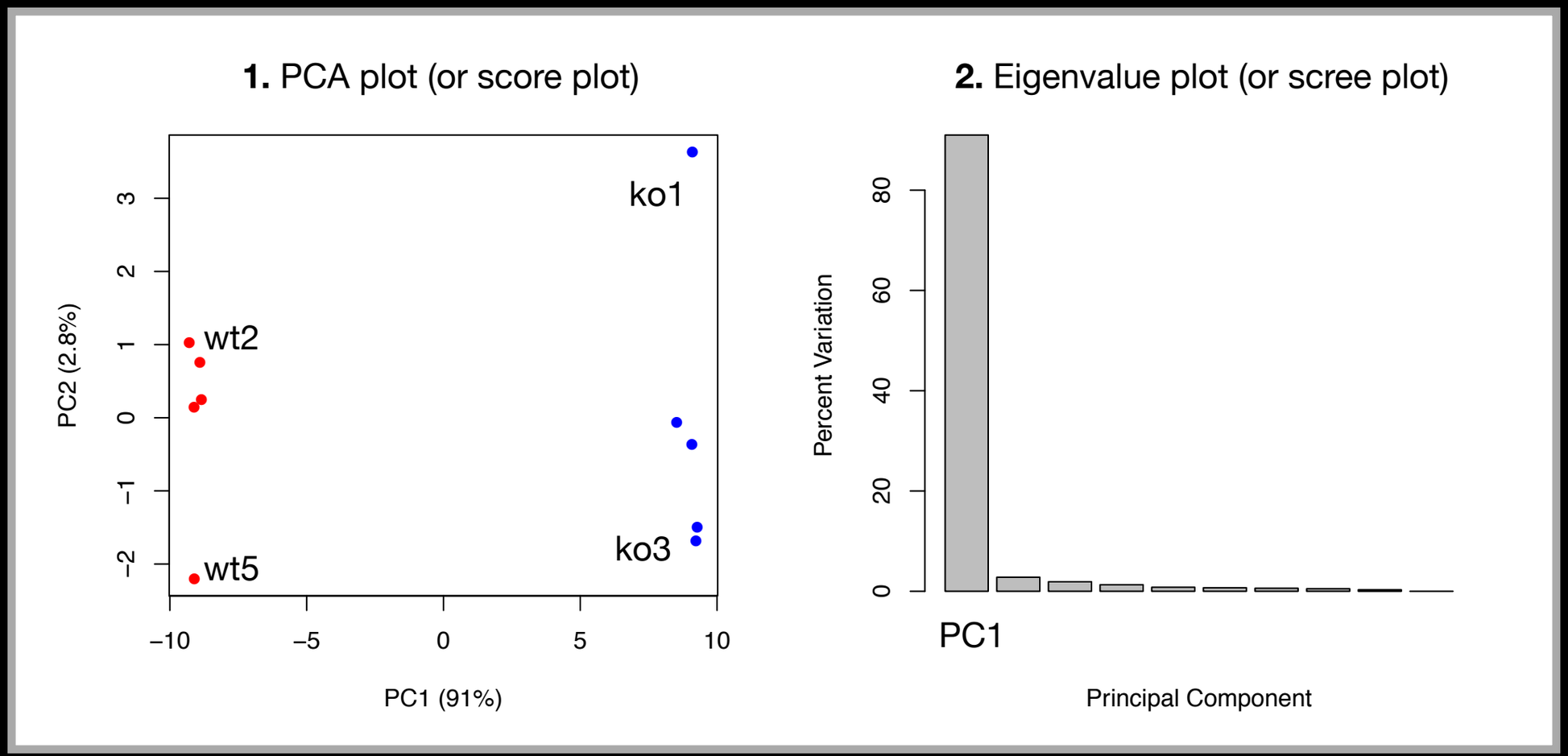
So PC1 is more important than PC2 which in turn is more important than PC3 etc.

So the red and blue cluster are more dissimilar than the yellow and blue clusters

The PCs (i.e. new plot axis) are ranked by the amount of variance in the original data (i.e. gene expression values) that they “capture”

In this example PC1 ‘captures’ 4x more of the original variance than PC2 (44/11 = 4)

- We actually get two main things out of a typical PCA
  - The new axis (called PCs or Eigenvectors) and
  - Eigenvalues that detail the amount of variance captured by each PC



- Another cool thing we can get out of PCA is a quantitive report on how the original variables contributed to each PC

    - In other words, which were the most important genes that lead to the observed clustering in PC-space

- These are often called the loadings and we can plot them to see which are the most important genes for the observed separation as well as outputting ranked lists of genes that act to discriminate the samples

```
    gene64     gene39
0.1047968  0.1047629
```

```
     gene7      gene65
-0.1047629 -0.1047443
```

Do it Yourself!

Hands-on time!

http://setosa.io/ev/principal-component-analysis/

Outline: How to do PCA in R

- How to use the **prcomp()** function to do PCA.
- How to draw and interpret PCA plots
- How to determine how much variation each principal component accounts for and the the “intrinsic dimensionality” useful for further analysis
- How to examine the **loadings** (or loading scores) to determine what variables have the largest effect on the graph and are thus important for discriminating samples.

- First lets generate some example data to work with.

```
## Initialize a blank 100 row by 10 column matrix
mydata <- matrix(nrow=100, ncol=10)
```

- First lets generate some example data to work with.

```
## Initialize a blank 100 row by 10 column matrix
mydata <- matrix(nrow=100, ncol=10)

## Lets label the rows gene1, gene2 etc. to gene100
rownames(mydata) <- paste("gene", 1:100, sep="")
```

- First lets generate some example data to work with.

```
## Initialize a blank 100 row by 10 column matrix
mydata <- matrix(nrow=100, ncol=10)

## Lets label the rows gene1, gene2 etc. to gene100
rownames(mydata) <- paste("gene", 1:100, sep="")

## Lets label the first 5 columns wt1, wt2, wt3, wt4 and wt5
##   and the last 5 ko1, ko2 etc. to ko5 (for "knock-out")
colnames(mydata) <- c( paste("wt", 1:5, sep=""),
                       paste("ko", 1:5, sep="") )
```

- First lets generate some example data to work with.

```
## Initialize a blank 100 row by 10 column matrix
mydata <- matrix(nrow=100, ncol=10)

## Lets label the rows gene1, gene2 etc. to gene100
rownames(mydata) <- paste("gene", 1:100, sep="")

## Lets label the first 5 columns wt1, wt2, wt3, wt4 and wt5
##   and the last 5 ko1, ko2 etc. to ko5 (for "knock-out")
colnames(mydata) <- c( paste("wt", 1:5, sep=""),
                       paste("ko", 1:5, sep="") )

## Fill in some fake read counts
for(i in 1:nrow(mydata)) {
  wt.values <- rpois(5, lambda=sample(x=10:1000, size=1))
  ko.values <- rpois(5, lambda=sample(x=10:1000, size=1))

  mydata[i,] <- c(wt.values, ko.values)
}

head(mydata)
```

- **NOTE**: the samples are columns, and the genes are rows!

```
mydata <- matrix(nrow=100, ncol=10)

rownames(mydata) <- paste("gene", 1:100, sep="")

colnames(mydata) <- c( paste("wt", 1:5, sep=""),
                       paste("ko", 1:5, sep="") )

for(i in 1:nrow(mydata)) {
  wt.values <- rpois(5, lambda=sample(x=10:1000, size=1))
  ko.values <- rpois(5, lambda=sample(x=10:1000, size=1))
  mydata[i,] <- c(wt.values, ko.values)
}

head(mydata)
```

```
      wt1 wt2 wt3 wt4 wt5 ko1 ko2 ko3 ko4 ko5
gene1 147 171 160 175 187  63  57  58  55  59
gene2 151 134 148 126 134 838 831 894 847 830
gene3 702 672 653 681 701 593 579 644 596 610
gene4 319 297 310 296 304 754 807 734 750 774
```

- Now we have our data we call **prcomp()** to do PCA
  - NOTE: **prcomp()** expects the samples to be rows and genes to be columns so we need to first transpose the matrix with the **t()** function!

```
## lets do PCA
pca <- prcomp(t(mydata), scale=TRUE)
```

- Now we have our data we call **prcomp()** to do PCA
  - NOTE: **prcomp()** expects the samples to be rows and genes to be columns so we need to first transpose the matrix with the **t()** function!

```
## lets do PCA
pca <- prcomp(t(mydata), scale=TRUE)

## See what is returned by the prcomp() function
attributes(pca)

# $names
#[1] "sdev"     "rotation" "center"   "scale"    "x"
#
# $class
#[1] "prcomp"
```

- The returned pca$x here contains the principal components (PCs) for drawing our first graph.
  - Here we will take the first two columns in pca$x (corresponding to PC1 and PC2) to draw a 2-D plot

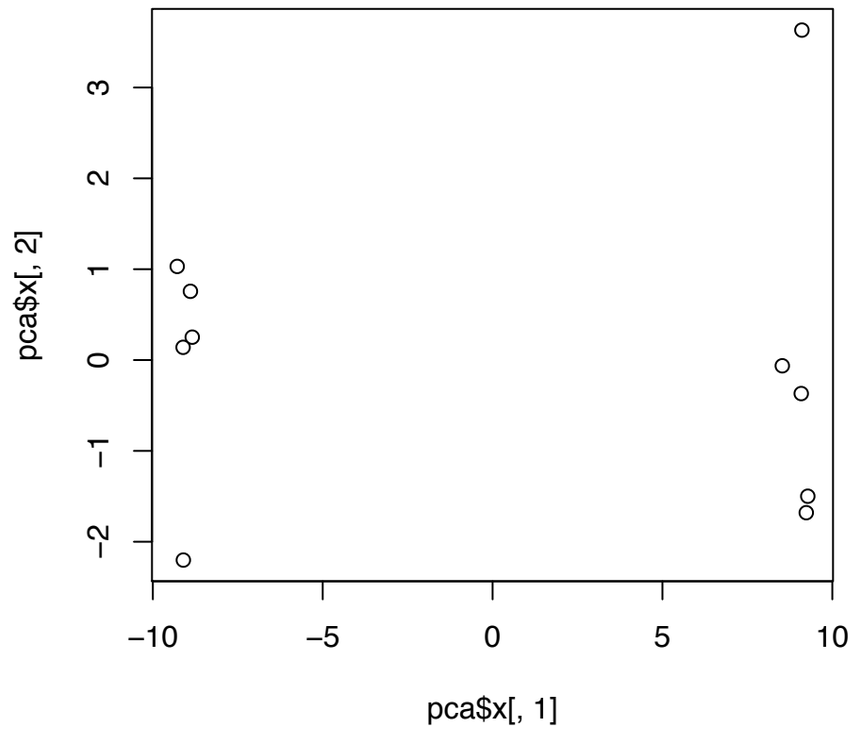
```
## lets do PCA
pca <- prcomp(t(mydata), scale=TRUE)

## See what is returned by the prcomp() function
attributes(pca)

# $names
#[1] "sdev"     "rotation" "center"   "scale"    "x"
#
# $class
#[1] "prcomp"
```

- The returned pca$x here contains the principal components (PCs) for drawing our first graph.
  - Here we will take the first two columns in pca$x (corresponding to PC1 and PC2) to draw a 2-D plot

```
## lets do PCA
pca <- prcomp(t(mydata), scale=TRUE)

## A basic PC1 vs PC2 2-D plot
plot(pca$x[,1], pca$x[,2])
```

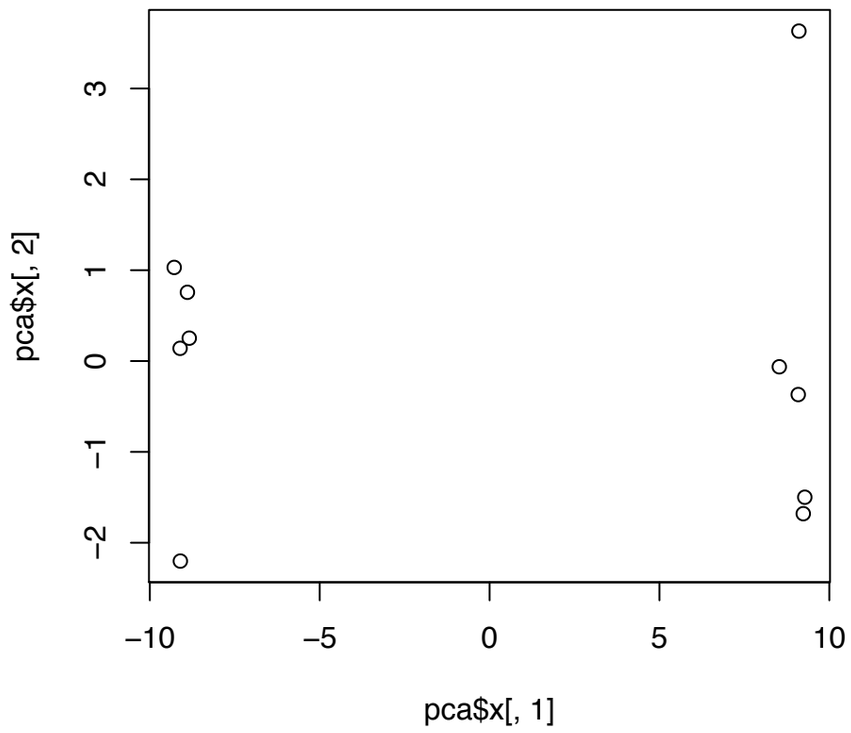
- The returned pca$x here contains the principal components (PCs) for drawing our first graph.
  - Here we will take the first two columns in pca$x (corresponding to PC1 and PC2) to draw a 2-D plot

```
## lets do PCA
pca <- prcomp(t(mydata), scale=TRUE)

## A basic PC1 vs PC2 2-D plot
plot(pca$x[,1], pca$x[,2])
```



- Looks interesting with a nice separation of samples into two groups of 5 samples each
  - Now we can use the square of pca$sdev , which stands for “standard deviation”, to calculate how much variation in the original data each PC accounts for

```
## lets do PCA
pca <- prcomp(t(mydata), scale=TRUE)

## A basic PC1 vs PC2 2-D plot
plot(pca$x[,1], pca$x[,2])

## Variance captured per PC
pca.var <- pca$sdev^2
```



- Looks interesting with a nice separation of samples into two groups of 5 samples each
  - Now we can use the square of pca$sdev , which stands for “standard deviation”, to calculate how much variation in the original data each PC accounts for

```
## lets do PCA
pca <- prcomp(t(mydata), scale=TRUE)

## A basic PC1 vs PC2 2-D plot
plot(pca$x[,1], pca$x[,2])

## Precent variance is often more informative to look at
pca.var <- pca$sdev^2
pca.var.per <- round(pca.var/sum(pca.var)*100, 1)
```

- Looks interesting with a nice separation of samples into two groups of 5 samples each
  - Now we can use the square of pca$sdev , which stands for “standard deviation”, to calculate how much variation in the original data each PC accounts for

```
## lets do PCA
pca <- prcomp(t(mydata), scale=TRUE)

## A basic PC1 vs PC2 2-D plot
plot(pca$x[,1], pca$x[,2])

## Precent variance is often more informative to look at
pca.var <- pca$sdev^2
pca.var.per <- round(pca.var/sum(pca.var)*100, 1)

pca.var.per
```

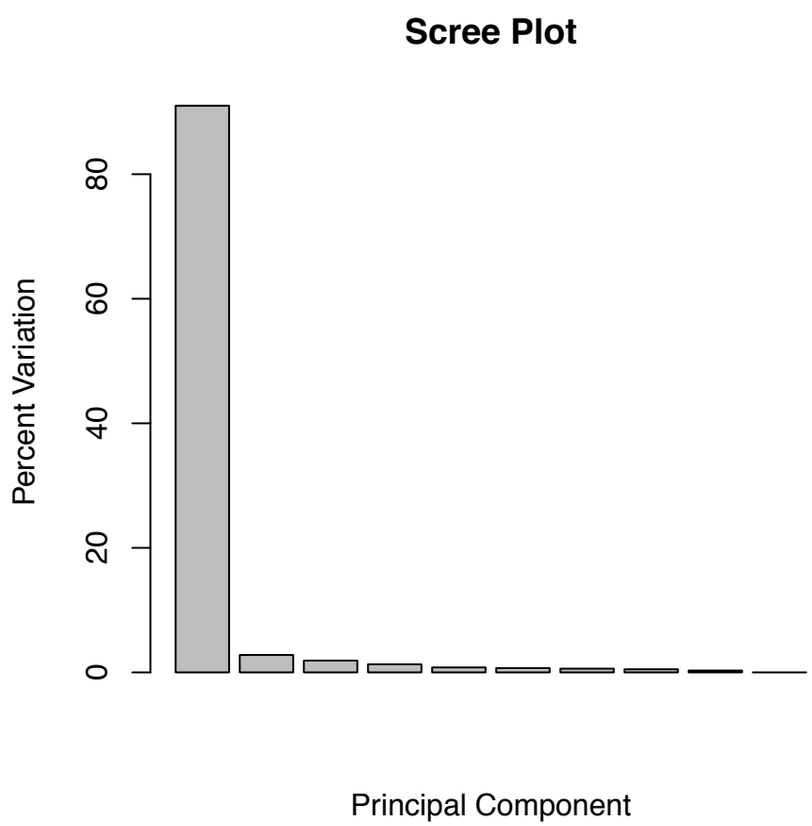
```
 [1] 91.0  2.8  1.9  1.3  0.8  0.7  0.6  0.5  0.3  0.0
```

- Looks interesting with a nice separation of samples into two groups of 5 samples each
  - Now we can use the square of pca$sdev , which stands for “standard deviation”, to calculate how much variation in the original data each PC accounts for

```
pca.var <- pca$sdev^2
pca.var.per <- round(pca.var/sum(pca.var)*100, 1)

barplot(pca.var.per, main="Scree Plot",
        xlab="Principal Component", ylab="Percent Variation")
```

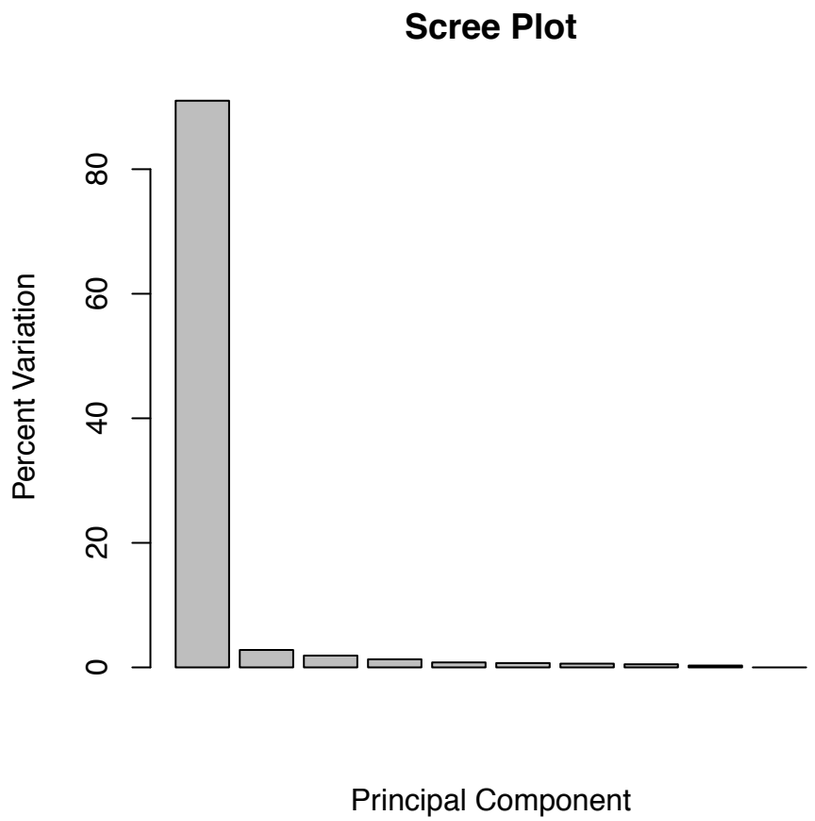
- From the "scree plot" it is clear that **PC1** accounts for almost all of the variation in the data!

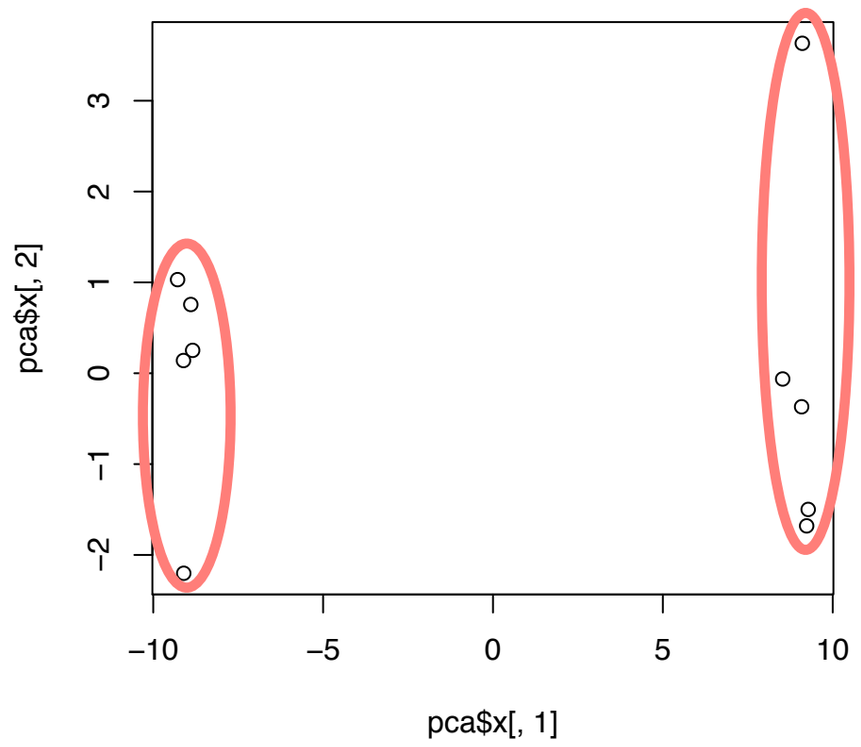
```
pca.var <- pca$sdev^2
pca.var.per <- round(pca.var/sum(pca.var)*100, 1)

barplot(pca.var.per, main="Scree Plot",
        xlab="Principal Component", ylab="Percent Variation")
```



- Which means there are big differences between these two groups that are separated along the PC1 axis…

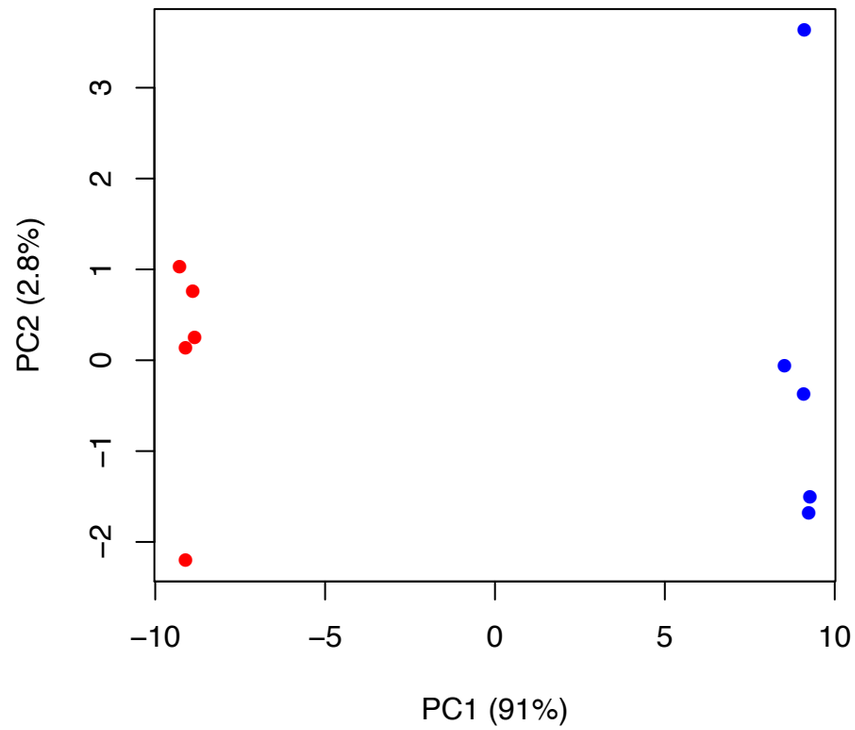
```
pca.var <- pca$sdev^2
pca.var.per <- round(pca.var/sum(pca.var)*100, 1)

barplot(pca.var.per, main="Scree Plot",
        xlab="Principal Component", ylab="Percent Variation")
```





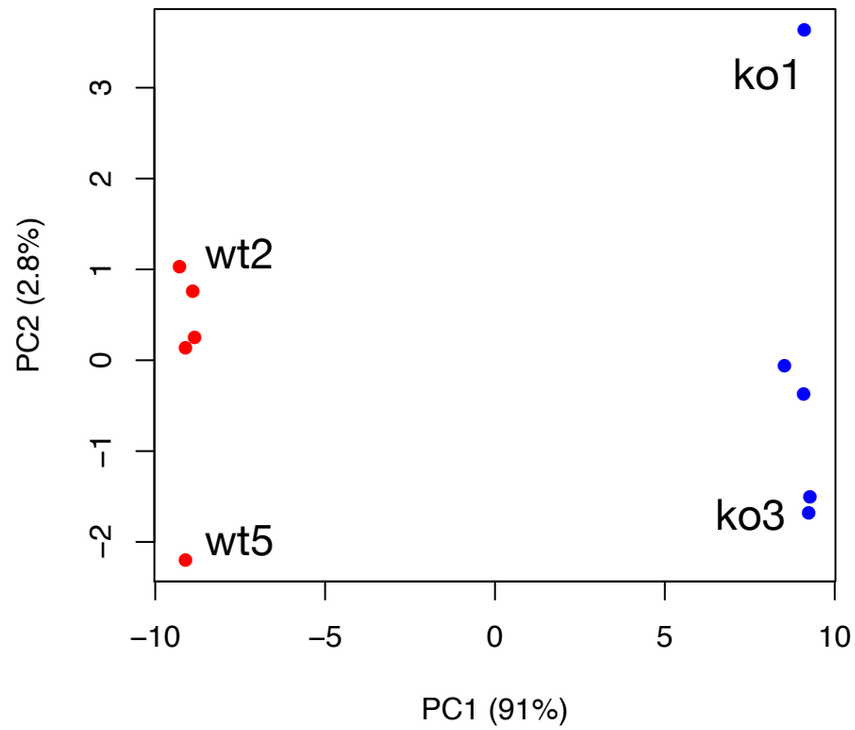
- Lets make our plot a bit more useful…

```
## A vector of colors for wt and ko samples
colvec <- colnames(mydata)
colvec[grep("wt", colvec)] <- "red"
colvec[grep("ko", colvec)] <- "blue"

plot(pca$x[,1], pca$x[,2], col=colvec, pch=16,
     xlab=paste0("PC1 (", pca.var.per[1], "%)"),
     ylab=paste0("PC2 (", pca.var.per[2], "%)"))
```



- And add some labels…

```
plot(pca$x[,1], pca$x[,2], col=colvec, pch=16,
     xlab=paste0("PC1 (", pca.var.per[1], "%)"),
     ylab=paste0("PC2 (", pca.var.per[2], "%)"))

## Click to identify which sample is which
identify(pca$x[,1], pca$x[,2], labels=colnames(mydata))

## Press ESC to exit…
```



- Finally, lets look at how to use the loading scores to determine which genes have the largest effect on where samples are plotted in the PCA plot
    - The prcomp() function calls loading scores $rotation

```
## Lets focus on PC1 as it accounts for > 90% of variance
loading_scores <- pca$rotation[,1]
```

- Finally, lets look at how to use the loading scores to determine which genes have the largest effect on where samples are plotted in the PCA plot
    - The prcomp() function calls loading scores $rotation

```
## Lets focus on PC1 as it accounts for > 90% of variance
loading_scores <- pca$rotation[,1]

summary(loading_scores)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.104763 -0.104276 -0.068784 -0.005656  0.103926  0.104797

## We are interested in the magnitudes of both plus
##   and minus contributing genes
gene_scores <- abs(loading_scores)
```

- Finally, lets look at how to use the loading scores to determine which genes have the largest effect on where samples are plotted in the PCA plot
    - The prcomp() function calls loading scores $rotation

```
loading_scores <- pca$rotation[,1]

gene_scores <- abs(loading_scores)

## Sort by magnitudes from high to low
gene_score_ranked <- sort(gene_scores, decreasing=TRUE)
```

- Finally, lets look at how to use the loading scores to determine which genes have the largest effect on where samples are plotted in the PCA plot
    - The prcomp() function calls loading scores $rotation

```
loading_scores <- pca$rotation[,1]

gene_scores <- abs(loading_scores)

## Sort by magnitudes from high to low
gene_score_ranked <- sort(gene_scores, decreasing=TRUE)

## Find the names of the top 5 genes
top_5_genes <- names(gene_score_ranked[1:5])
```

- Finally, lets look at how to use the loading scores to determine which genes have the largest effect on where samples are plotted in the PCA plot
    - The prcomp() function calls loading scores $rotation

```
loading_scores <- pca$rotation[,1]

gene_scores <- abs(loading_scores)

## Sort by magnitudes from high to low
gene_score_ranked <- sort(gene_scores, decreasing=TRUE)

## Find the names of the top 5 genes
top_5_genes <- names(gene_score_ranked[1:5])

## Show the scores (with +/- sign)
pca$rotation[top_5_genes,1]
```

- Here we see genes with the largest positive loading scores that effectively ‘push’ the “ko” samples to the right positive side of the plot.
- And the genes with high negative scores that push “wt” samples to the left side of the plot.

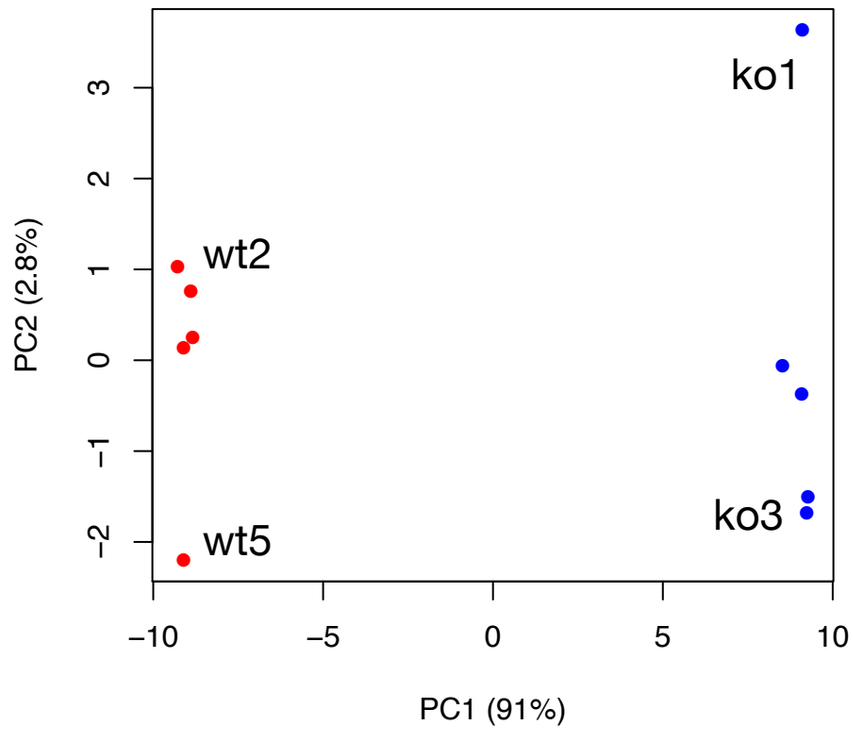
```
loading_scores <- pca$rotation[,1]

gene_scores <- abs(loading_scores)

## Sort by magnitudes from high to low
gene_score_ranked <- sort(gene_scores, decreasing=TRUE)

## Find the names of the top 5 genes
top_5_genes <- names(gene_score_ranked[1:5])

## Show the scores (with +/- sign)
pca$rotation[top_5_genes,1]
```

```
    gene64     gene39      gene7     gene60     gene65
 0.1047968  0.1047629 -0.1047629  0.1047601 -0.1047443
```

- Here we see genes with the largest positive loading scores that effectively ‘push’ the “ko” samples to the right positive side of the plot.
- And the genes with high negative scores that push “wt” samples to the left side of the plot.

```
pca$rotation[top_5_genes,1]
```

```
    gene64     gene39
 0.1047968  0.1047629

     gene7     gene65
-0.1047629 -0.1047443
```



PCA Recap

- PCA is classic "multivariate statistical technique" used to reduce the dimensionality of a complex data set to a more manageable number (typically 2D or 3D)

- For a matrix of *m* genes x *n* samples, we mean center (i.e. subtract the sample mean from each sample column), optionally rescale the values for each sample column, then calculate a new covariance matrix of size *n* x *n*

- We finally diagonalize the covariance matrix to yield our *n* Eigenvectors (called principal components or PCs) and *n* Eigenvalues.

- The top PCs (with largest Eigenvalues) retain the essential features of the original data and represent a useful subspace for further analysis (e.g. visualization, clustering, feature extraction, outlier detection etc…)

PCA objectives in a nutshell

- to reduce dimensionality
- to visualize multidimensional data
- to choose the most useful variables (features)
- to identify groupings of objects (e.g. genes/samples)
- to identify outliers

Do it Yourself!

Your turn!

Perform a PCA on the UK foods dataset

UK_foods.csv

Input: read, View/head,
PCA: prcomp,
Plots: PCA plot
scree plot,
loadings plot.

https://bioboot.github.io/bggn213_f17/class-material/UK_food_pca/

Please do your formal class evaluations!