DIFFERENTIAL GENE EXPRESSION ANALYSIS

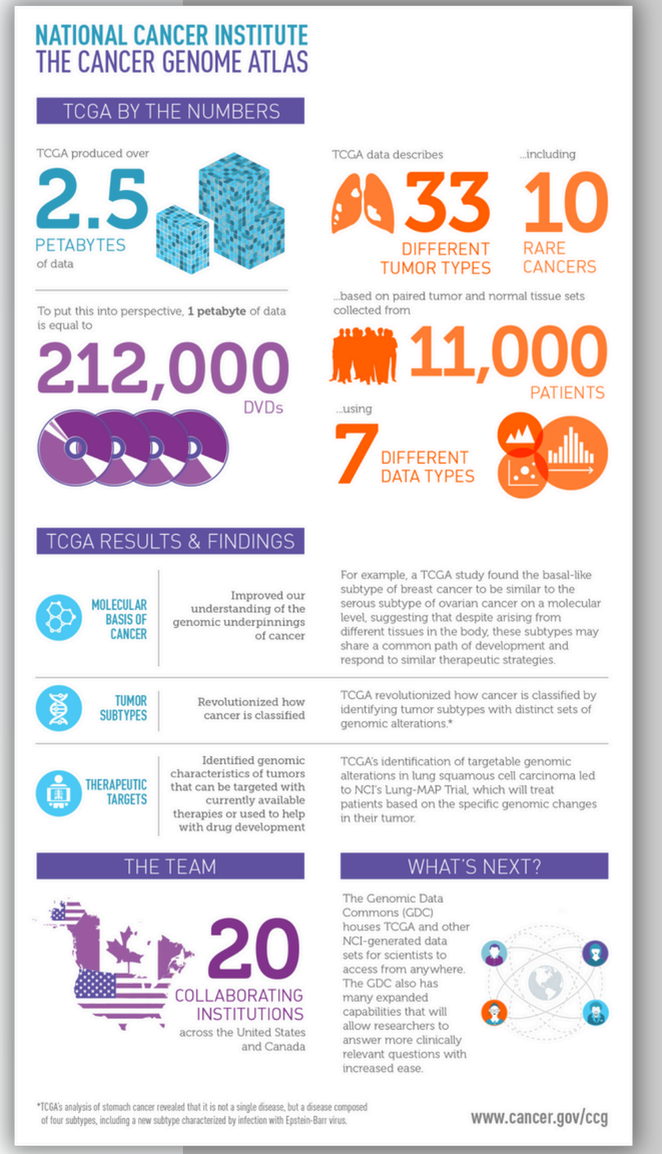
SARTHAK RAY

TCGA

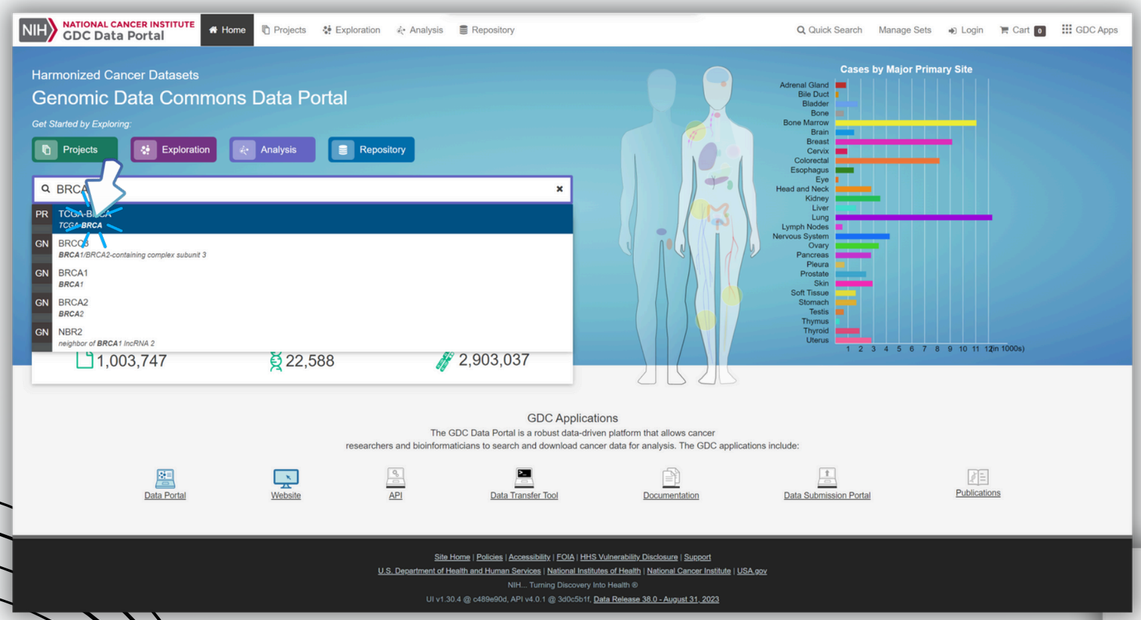
THE CANCER GENOME ATLAS

The Cancer Genome Atlas (TCGA) is a project to catalogue the genetic mutations responsible for cancer using genome sequencing and bioinformatics.

This joint effort between NCI and the National Human Genome Research Institute began in 2006, bringing together researchers from diverse disciplines and multiple institutions.

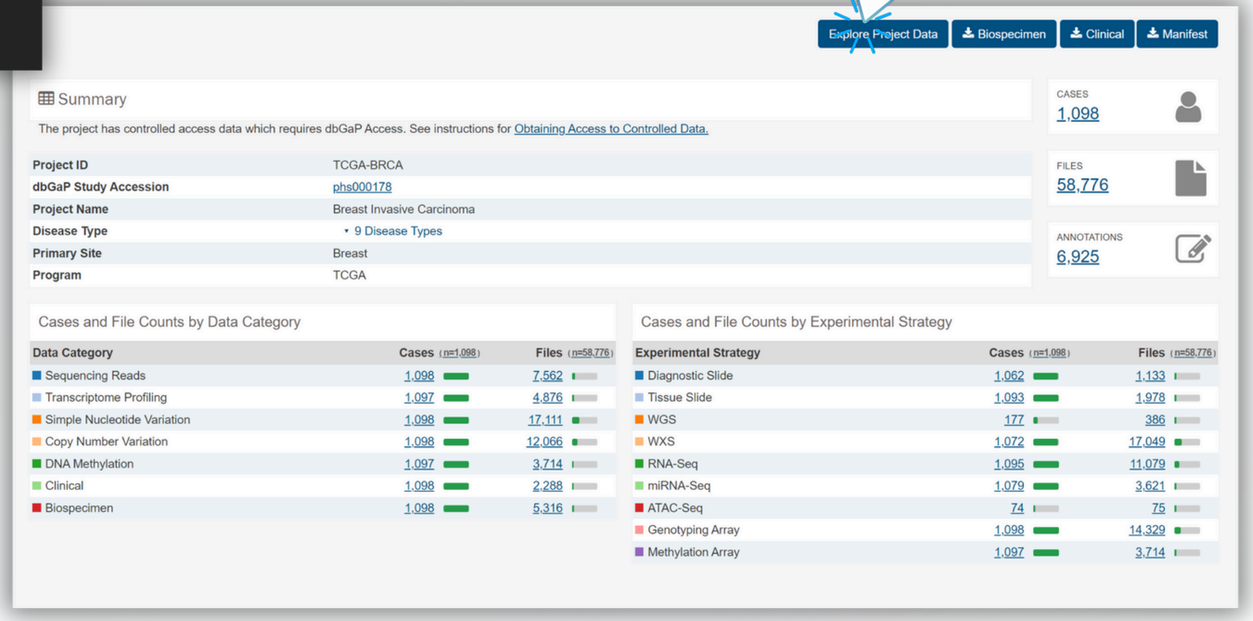


GDC DATA PORTAL

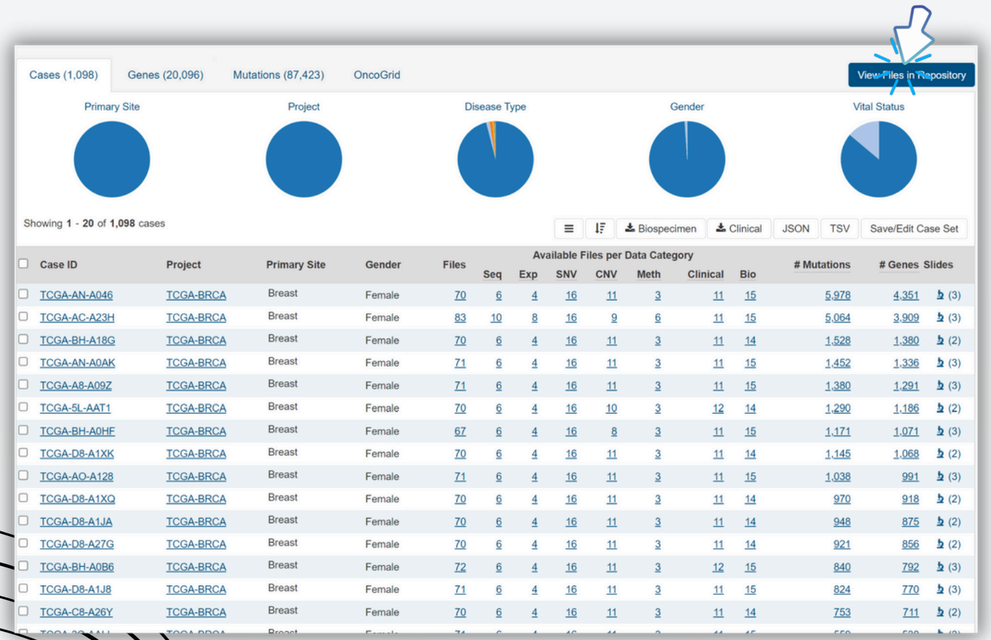


- Visit **https://portal.gdc.cancer.gov/**
- Look for TCGA-BRCA in the search bar.

- You land here in the summary page which describes the project.
- In our case we'll study **Breast Invasive Carcinoma**.
- Go to "Explore Project Data".

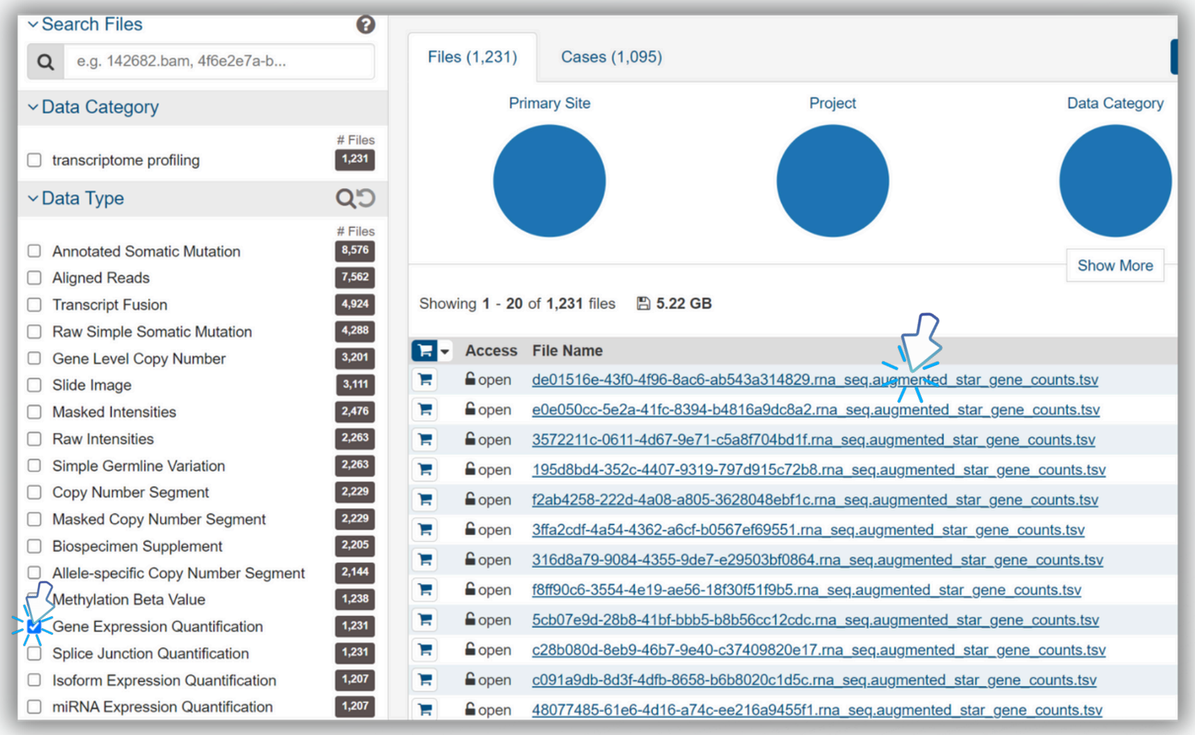


NEXT STEPS...



- This is the Explore page with a list of all Case IDs.
- Next go to “View Files in Repository”.

- This page gives a list of all kinds of data files pertaining to the BRCA project.
- Select **Gene Expression Quantification** in the filter and download all the tsv files.
- Visit the first file to understand the nuiances.

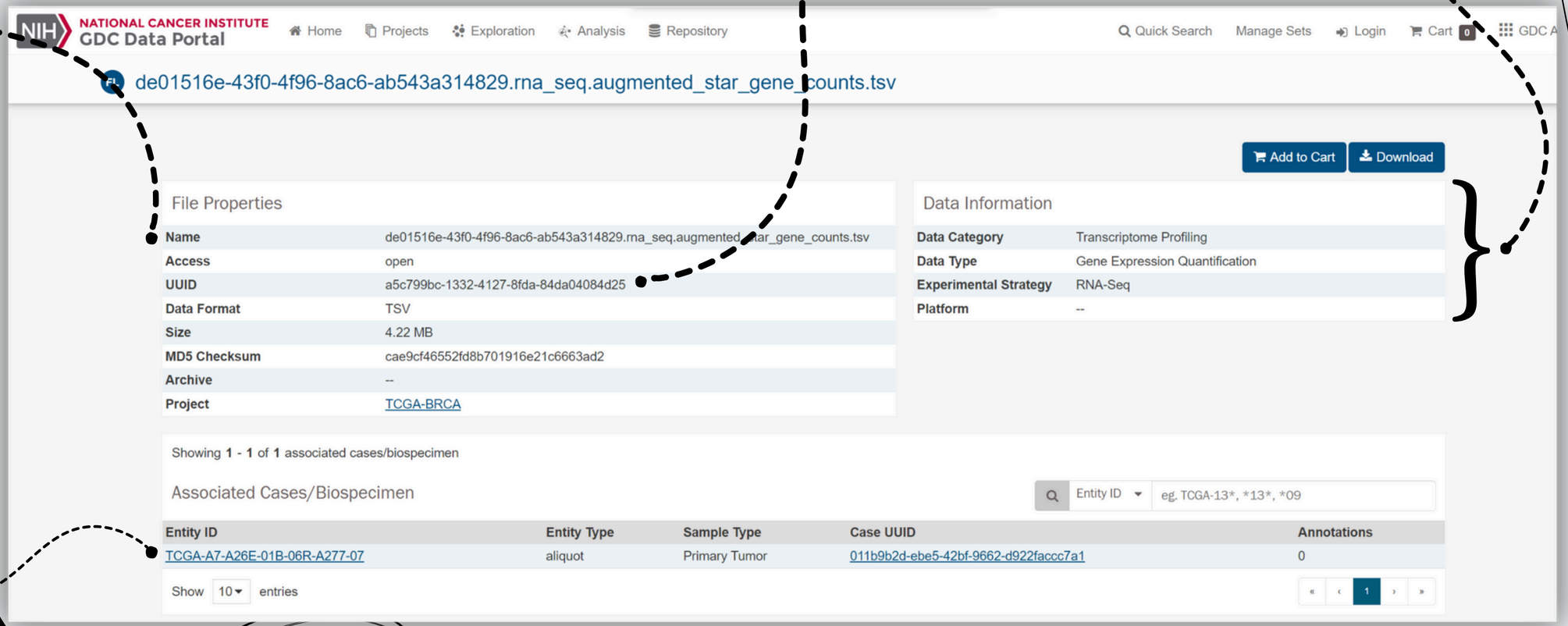


KEY IDENTIFIERS

Name: This will be the name of your respective file upon dowloading.

UUID: This will be the name of the folder which will contain your file..

Data Information: This bit will be used to later in R to make queries to the GDC server.



Entity ID: This will be your particular sample for the chosen case file.

ENTITY ID

For our purposes, we only need the following four parts for our analysis:

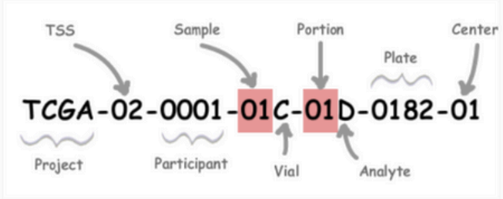
- Analyte
- Plate
- Portion
- Vial

Entity ID
TCGA-A7-A26E-01B-06R-A277-07

TCGA-A7-A26E-01B

Reading Barcodes

A TCGA barcode is composed of a collection of identifiers. Each specifically identifies a TCGA data element. Refer to the following figure for an illustration of how metadata identifiers comprise a barcode. An aliquot barcode, an example of which shows in the illustration, contains the highest number of identifiers.



Label	Identifier for	Value	Value Description	Possible Values
Analyte	Molecular type of analyte for analysis	D	The analyte is a DNA sample	See Code Tables Report
Plate	Order of plate in a sequence of 96-well plates	182	The 182nd plate	4-digit alphanumeric value
Portion	Order of portion in a sequence of 100 - 120 mg sample portions	1	The first portion of the sample	01-99
Vial	Order of sample in a sequence of samples	C	The third vial	A to Z
Project	Project name	TCGA	TCGA project	TCGA
Sample	Sample type	1	A solid tumor	Tumor types range from 01 - 09, normal types from 10 - 19 and control samples from 20 - 29. See Code Tables Report for a complete list of sample codes
Center	Sequencing or characterization center that will receive the aliquot for analysis	1	The Broad Institute GCC	See Code Tables Report
Participant	Study participant	1	The first participant from MD Anderson for GBM study	Any alpha-numeric value
TSS	Tissue source site	2	GBM (brain tumor) sample from MD Anderson	See Code Tables Report

CODE

FORMING COMBINED DATAFRAME

```r
# Query GDC metadata without UUID filter
query <- GDCquery(project = "TCGA-BRCA",
                  data.category = "Transcriptome Profiling",
                  data.type = "Gene Expression Quantification")
```

- This query fetches all the metadata from TCGA-BRCA project with the mentioned data category and type(shown previously).
- This will enable us to obtain the Entity IDs for the corresponding file names which we downloaded.

```
Data
query                          1 obs. of 10 variables
    $ results                  :List of 1
     ..$ :'data.frame':        1231 obs. of  29 variables:
     .. ..$ id                            : chr  "a5c799bc-1332-4127-8fda-84d
     .. ..$ data_format                   : chr  "TSV" "TSV" "TSV" "TSV" ...
     .. ..$ cases                         : chr  "TCGA-A7-A26E-01B-06R-A277-0
     .. ..$ access                        : chr  "open" "open" "open" "open"
     .. ..$ file_name                     : chr  "de01516e-43f0-4f96-8ac6-ab5
     .. ..$ submitter_id                  : chr  "162ed5c3-add6-4485-8d99-37c
     .. ..$ data_category                 : chr  "Transcriptome Profiling" "T
     .. ..$ type                          : chr  "gene_expression" "gene_expr
     .. ..$ file_size                     : int  4220759 4242201 4240106 4248
     .. ..$ created_datetime              : chr  "2021-12-13T22:37:49.340267-
     .. ..$ md5sum                        : chr  "cae9cf46552fd8b701916e21c66
     .. ..$ updated_datetime              : chr  "2022-01-19T12:00:30.311996-
     .. ..$ file_id                       : chr  "a5c799bc-1332-4127-8fda-84d
     .. ..$ data_type                     : chr  "Gene Expression Quantificat
     .. ..$ state                         : chr  "released" "released" "relea
     .. ..$ experimental_strategy         : chr  "RNA-Seq" "RNA-Seq" "RNA-Seq
     .. ..$ version                       : chr  "1" "1" "1" "1" ...
     .. ..$ data_release                  : chr  "32.0 - 38.0" "32.0 - 38.0"
     .. ..$ project                       : chr  "TCGA-BRCA" "TCGA-BRCA" "TCG
     .. ..$ analysis_id                   : chr  "1d06d9a6-4145-4c38-a35d-c23
     .. ..$ analysis_state                : chr  "released" "released" "relea
     .. ..$ analysis_submitter_id         : chr  "de01516e-43f0-4f96-8ac6-ab5
     .. ..$ analysis_workflow_link        : chr  "https://github.com/NCI-GDC/
     .. ..$ analysis_workflow_type        : chr  "STAR - Counts" "STAR - Coun
     .. ..$ analysis_workflow_version: chr  "5d8c131bbff59fb0c969217fc1d
     .. ..$ sample_type                   : chr  "Primary Tumor" "Primary Tum
     .. ..$ is_ffpe                       : logi  NA NA NA NA NA NA ...
     .. ..$ cases.submitter_id            : chr  "TCGA-A7-A26E" "TCGA-A2-A0CU
     .. ..$ sample.submitter_id           : chr  "TCGA-A7-A26E-01B" "TCGA-A2-
```

CODE

```r
# Extract sample submitter IDs and UUIDs from the query result
sample_submitter_ids <- query[[1]][[1]]$sample.submitter_id
uuids <- query[[1]][[1]]$id

# Create a dataframe with the extracted information
sample_uuid_df <- data.frame(Sample_Submitter_ID = sample_submitter_ids, UUID = uuids)

write_xlsx(sample_uuid_df, "C:/Users/sarth/Downloads/idvssample.xlsx")
```

- This code extracts the required two columns from our query result namely sample.submitter_id and id essentially creating an id vs. sample table.
- Then it writes and saves them in a excel file for further usage.

Sample_Submitter_ID	UUID
TCGA-B6-A0RH-01A	6b62d6dd-bb6f-4f39-9041-992c47b875f3
TCGA-BH-A1FU-11A	eddb2dc6-2b72-43a8-a7fc-3dd09dde68af
TCGA-BH-A1FU-01A	90eefb33-6808-45c3-9ab1-6eca0decdc9d
TCGA-AR-A0TX-01A	eb166054-ff70-4a86-883b-9c25a7d2b0e5
TCGA-A1-A0SE-01A	f66aace4-01a4-4def-8e71-26d9aaf13d49
TCGA-BH-A1FC-11A	248ac510-a5af-4608-9cdc-5c8673633b82
TCGA-OL-A5D6-01A	ff5f8ada-17c5-497e-9182-63a05e3ab4c5
TCGA-E2-A1IK-01A	4bfe1281-919d-47ee-afc7-2b373b113674
TCGA-AC-A2FM-11B	52151eca-7819-496e-bf31-4875b68d429d
TCGA-AN-A0FT-01A	2ff175d2-100f-4ce8-8d1e-b3b95b41ae56
TCGA-A2-A3KD-01A	5599b7dd-7c2a-4037-ad77-11a0b4ddcb54
TCGA-A2-A0EO-01A	31e1dbf5-8c97-4e4d-a488-d6737ab304c2
TCGA-A2-A04P-01A	c95a37a1-d3cd-421c-bc5f-0f8dc448f64c
TCGA-AC-A5EH-01A	99367019-5583-4bcc-aa62-25916515fe17
TCGA-A7-A0DC-01B	3e8cff80-ed17-479c-8327-ca8e71ee7482
TCGA-OK-A5Q2-01A	c25c7ca5-8d03-4bba-b42b-c8b1042c0d48
TCGA-BH-A0DO-01B	69bb06b6-74b3-441b-82d7-c6783aa88cdd
TCGA-BH-A0DO-11A	4df2233b-3bbc-4d20-9abc-2a09b3f37383
TCGA-BH-A18J-01A	561bb38a-f9b8-4a0a-b03b-fbe70859869a
TCGA-A7-A0CJ-01A	c55e6460-fea1-468e-951e-bf45bbddd7a6
TCGA-A8-A09V-01A	1300bf72-ead0-4c03-b140-fca26302b43c
TCGA-C8-A275-01A	f35ae5c7-6e88-410d-b86f-c5bdae11a5f4
TCGA-E2-A1BC-11A	c6a4afd8-8044-475f-b4fd-a1b4cb922976

CODE

# gene-model: GENCODE v36	
gene_id	**gene_name**
N_unmapped	
N_multimapping	
N_noFeature	
N_ambiguous	
ENSG00000000003.15	TSPAN6
ENSG00000000005.6	TNMD
ENSG00000000419.13	DPM1
ENSG00000000457.14	SCYL3
ENSG00000000460.17	C1orf112
ENSG00000000938.13	FGR
ENSG00000000971.16	CFH
ENSG00000001036.14	FUCA2
ENSG00000001084.13	GCLC
ENSG00000001167.14	NFYA
ENSG00000001460.18	STPG1
ENSG00000001461.17	NIPAL3
ENSG00000001497.18	LAS1L
ENSG00000001561.7	ENPP4
ENSG00000001617.12	SEMA3F
ENSG00000001626.16	CFTR
ENSG00000001629.10	ANKIB1
ENSG00000001630.17	CYP51A1

```r
# Specify the main folder containing subfolders whcih contain TSV files
main_folder <- "D:/btp/TCGA/GDCdata/TCGA-BRCA/Transcriptome_Profiling/Gene_Expression_Quantification"

# List subfolders in the main folder
subfolders <- list.files(main_folder, full.names = TRUE)

# Initialize gene_ids and gene_names using the first TSV file
first_tsv_file <- list.files(subfolders[1], pattern = "*.tsv", full.names = TRUE)[1]
gene_ids <- read.table(first_tsv_file, header = TRUE, sep = "\t")$gene_id
gene_names <- read.table(first_tsv_file, header = TRUE, sep = "\t")$gene_name

gene_ids <- gene_ids[-(1:4)]
gene_names <- gene_names[-(1:4)]

# Set gene_ids as row names for the final_data dataframe
final_data <- data.frame(row.names = gene_ids)
final_data$GeneNames <- gene_names
```

- We now enter the first file(arbitrary) and extract all the gene_ids and their corresponding gene_names.
- We exclude the first 4 irrelevant columns and then form the structure of our dataframe which will combine count data from all the files.

CODE

```r
# Loop through each subfolder
for (subfolder in subfolders) {
  # Extract the subfolder name from the path
  subfolder_name <- tools::file_path_sans_ext(basename(subfolder))

  # Find the corresponding sample ID from the subfolder name(id) in sample_uuid_df
  sample_id <- sample_uuid_df$Sample_Submitter_ID[sample_uuid_df$UUID == subfolder_name]

  # List the TSV file in the subfolder
  tsv_file <- list.files(subfolder, pattern = "*.tsv", full.names = TRUE)

  # Select the count data of the unstranded column from the tsv file
  unstranded_column <- read.table(tsv_file, header = TRUE, sep = "\t")$unstranded
  unstranded_column <- unstranded_column[-(1:4)]

  # Add this data under the corresponding sample ID
  final_data[sample_id] <- unstranded_column
}


# Write the combined data to an Excel file
write.xlsx(final_data, "C:/Users/sarth/Downloads/final_data.xlsx", row.names = TRUE)
```

- We now run a for loop; extracting the subfolder name which will give us the sample ID and opening the tsv file in each of them.
- We will use the count data for unstranded column and thus extract and refine this column.

GeneIds	GeneNames	TCGA-A8-A086-01A	TCGA-D8-A	TCGA-AN-A
ENSG00000000003.15	TSPAN6	4263	4370	2443
ENSG00000000005.6	TNMD	9	7	144
ENSG00000000419.13	DPM1	2071	2625	2322
ENSG00000000457.14	SCYL3	1101	3005	1466
ENSG00000000460.17	C1orf112	717	1578	409
ENSG00000000938.13	FGR	312	599	1179
ENSG00000000971.16	CFH	2840	4864	11555
ENSG00000001036.14	FUCA2	2812	1944	2770
ENSG00000001084.13	GCLC	4188	1958	2260
ENSG00000001167.14	NFYA	2886	4597	2448
ENSG00000001460.18	STPG1	770	669	750
ENSG00000001461.17	NIPAL3	3410	3220	1922
ENSG00000001497.18	LAS1L	3809	3766	2862
ENSG00000001561.7	ENPP4	1029	3504	2457
ENSG00000001617.12	SEMA3F	7335	4516	3711
ENSG00000001626.16	CFTR	164	12	12

- We will add this column under the respective sample ID.
- Finally we will export it into an excel file.

CODE

INTEGRATING ER_STATUS VALUES

```
# Import clinical data
clinicaldata <- TCGA.BRCA
remove(TCGA.BRCA)
```

- Now we need to import clinical data which will have all kinds of metadata.
- We will import the dataframe and assign it to the variable clinical_data.
- Observe that the sampleIDs here don't have the symbol for vial at the end.

sampleID	AJCC_Stage	Age_at_Ini	CN_Cluster	Converted_	Days_to_D	Days_to_d	ER_Status_
TCGA-3C-AAAU-01							
TCGA-3C-AALI-01							
TCGA-3C-AALJ-01							
TCGA-3C-AALK-01							
TCGA-4H-AAAK-01							
TCGA-5L-AAT0-01							
TCGA-5L-AAT1-01							
TCGA-5T-A9QA-01							
TCGA-A1-A0SB-01	Stage I	70	1	Stage I	259		Positive
TCGA-A1-A0SD-01	Stage IIA	59	2	Stage IIA	437		Positive
TCGA-A1-A0SE-01	Stage I	56	2	Stage I	1320		Positive
TCGA-A1-A0SF-01	Stage IIA	54	3	Stage IIA	1463		Positive
TCGA-A1-A0SG-01	Stage IIB	61	4	Stage IIB	433		Positive
TCGA-A1-A0SH-01	Stage IIA	39	5	Stage IIA	1437		Negative
TCGA-A1-A0SI-01	Stage IIB	52	3	Stage IIB	634		Positive
TCGA-A1-A0SJ-01	Stage IIIA	39	3	Stage IIIA	426		Positive
TCGA-A1-A0SK-01	Stage IIA	54	1	Stage IIA	594	967	Negative
TCGA-A1-A0SM-01	Stage IIA	77	2	Stage IIA	242		Positive
TCGA-A1-A0SN-01	Stage IIIA	50	5	Stage IIA	1196		Positive

CODE

```r
# Extract sample IDs without the last character(representing vial)
sample_ids <- substr(sample_uuid_df$Sample_Submitter_ID, 1,
                     nchar(sample_uuid_df$Sample_Submitter_ID) - 1)

# Find corresponding ER_status values from clinical data using sample_ids
er_status_values <- clinicaldata$ER_Status_nature2012[match(
  sample_ids, clinicaldata$sampleID)]

# Add the ER_status column to sample_uuid_df
sample_uuid_df$ER_status <- er_status_values
er_status <- sample_uuid_df
write_xlsx(er_status, "C:/Users/sarth/Downloads/erstatus.xlsx")
```

- We wil obtain all the samples from our sample_uuid_df and remove the last character in each which corresponds to the vial, as the clinical data doesn't have that.
- Then we will search for the sample_ids in the clinical data and obtain their er_status.
- Finally, we will add that column to the sample_uuid_df, rename it to er_status and download an excel sheet of it if required.

	Sample_Submitter_ID	UUID	ER_status
1	TCGA-A7-A26E-01B	a5c799bc-1332-4127-8fda-84da04084d25	Positive
2	TCGA-A2-A0CU-01A	1dd898ab-3b85-4932-b889-0eb0a494f2b6	Positive
3	TCGA-PL-A8LV-01A	7286a685-17ef-4ef5-9b7b-2696c6e7e1a6	
4	TCGA-BH-A0BC-01A	de6f1503-33d7-4f86-b835-bdffba7ea4e3	Positive
5	TCGA-AR-A1AX-01A	4c46a463-f0e3-4e19-ad95-5dab0231d1c6	Positive
6	TCGA-AC-A2FO-01A	586d9c21-e58a-4171-9755-30c0e8b16293	
7	TCGA-BH-A1FU-11A	eddb2dc6-2b72-43a8-a7fc-3dd09dde68af	
8	TCGA-AQ-A0Y5-01A	c2515ef2-5cfc-4cc2-9541-ae072616d361	Positive
9	TCGA-AC-A3EH-01A	d93c4b85-2617-43d9-8017-ed3a82dc4117	
10	TCGA-AC-A5EH-01A	99367019-5583-4bcc-aa62-25916515fe17	
11	TCGA-D8-A142-01A	c8c7b9cc-7015-4405-a7f5-f60587de285a	Negative
12	TCGA-D8-A1XB-01A	176e739b-87e2-49b7-80e1-c27a6a632576	Positive

CODE

```r
# Find samples with empty or "Indeterminate" ER status
samples_with_empty_er <- er_status$Sample_Submitter_ID[er_status$ER_status == ""
                                                       | er_status$ER_status == "Indeterminate"]

# Remove columns from final_data where samples have empty or "Indeterminate" ER status
final_data_filtered <- final_data[, !(colnames(final_data) %in% samples_with_empty_er)]
final_data_filtered <- final_data_filtered[, -1]


# Reorder er_status based on the order of samples in final_data_filtered
er_status_filtered <- er_status[match(colnames(final_data_filtered),
                                      er_status$Sample_Submitter_ID), ]
er_status_filtered <- er_status_filtered[, -2]

# Convert ER_status to factor form
er_status_filtered$ER_status <- as.factor(er_status_filtered$ER_status)

write_xlsx(er_status_filtered, "C:/Users/sarth/Downloads/coldata.csv")
write_xlsx(final_data_filtered, "C:/Users/sarth/Downloads/countdata.csv")
```

- Now, we remove the samples whose Er_status is either empty or indeterminate.
- We filter out the empty samples from both final_data and er_status and reorder er_status according to final_data.
- We also convert the Er_status column to factor form.

final_data_filtered

	TCGA-D8-A1XO-01A	TCGA-AN-A0FN-01A	TCGA-BH-A18M-01A	TCGA-E2-A15T-01A	TCGA-AR-A0TU-01A
ENSG00000000003.15	4370	2443	1635	3456	8943
ENSG00000000005.6	7	144	101	22	10
ENSG00000000419.13	2625	2322	1565	1779	2621
ENSG00000000457.14	3005	1466	1183	2176	1804
ENSG00000000460.17	1578	409	419	864	2349
ENSG00000000938.13	599	1179	416	250	1050
ENSG00000000971.16	4864	11555	2957	1100	736

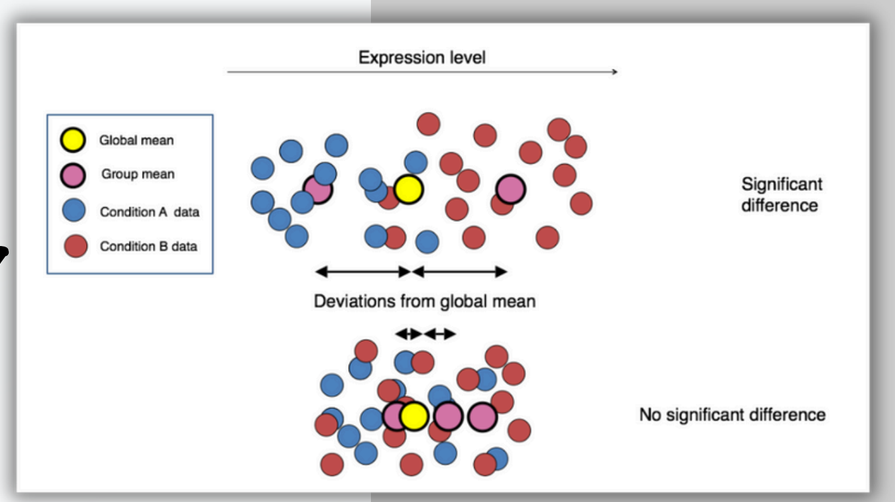
er_status_filtered

	Sample_Submitter_ID	ER_status
701	TCGA-D8-A1XO-01A	Positive
970	TCGA-AN-A0FN-01A	Positive
686	TCGA-BH-A18M-01A	Positive
304	TCGA-E2-A15T-01A	Positive
284	TCGA-AR-A0TU-01A	Negative
313	TCGA-EW-A1IX-01A	Positive

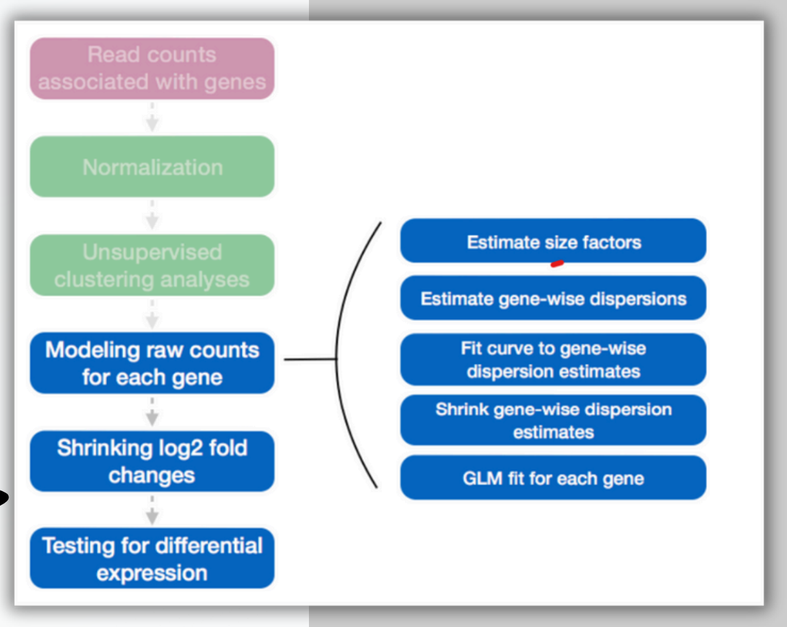
DGE

DESEQ2 (R)

Our goal is to essentially determine whether the mean expression levels of different sample groups are significantly different.



Differential expression analysis with DESeq2 involves multiple steps as displayed in the flowchart aside in blue.



NORMALIZATION

```
#Normalization to account for different sequencing depth
dds <- estimateSizeFactors(dds)
```

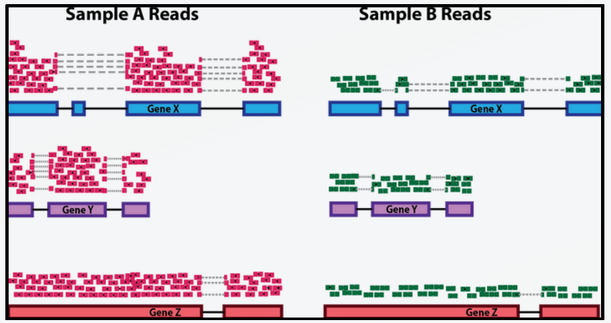
To normalize the count data, DESeq2 calculates size factors for each sample using the median of ratios method

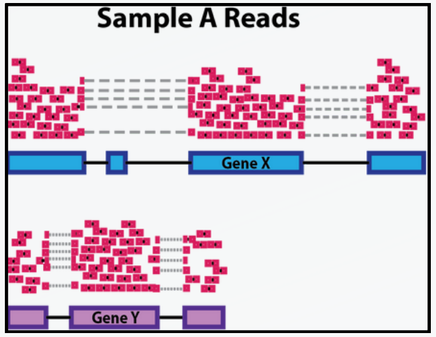
DESeq2's **median of ratios** [1]	counts divided by sample-specific size factors determined by median ratio of gene counts relative to geometric mean per gene	sequencing depth and RNA composition	gene count comparisons between samples and for **DE analysis**; **NOT for within sample comparisons**

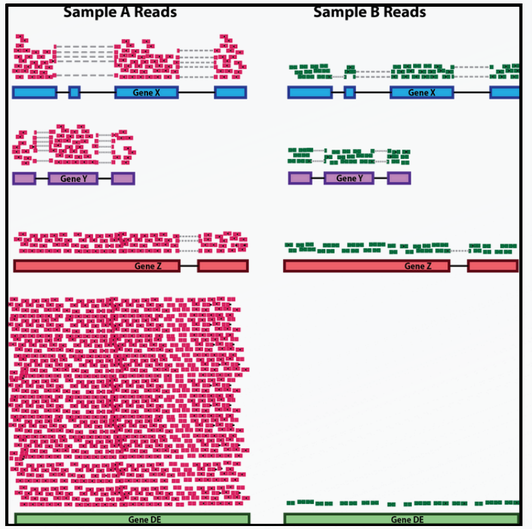
Sequencing depth: Each gene appears to have doubled in expression in Sample A relative to Sample B, however this is a consequence of Sample A having double the sequencing depth.

Gene length: Gene X and Gene Y have similar levels of expression, but the number of reads mapped to Gene X would be many more than the number mapped to Gene Y because Gene X is longer.

RNA composition: If we were to divide each sample by the total number of counts to normalize, the counts would be greatly skewed by the DE gene, which takes up most of the counts for Sample A, but not Sample B.



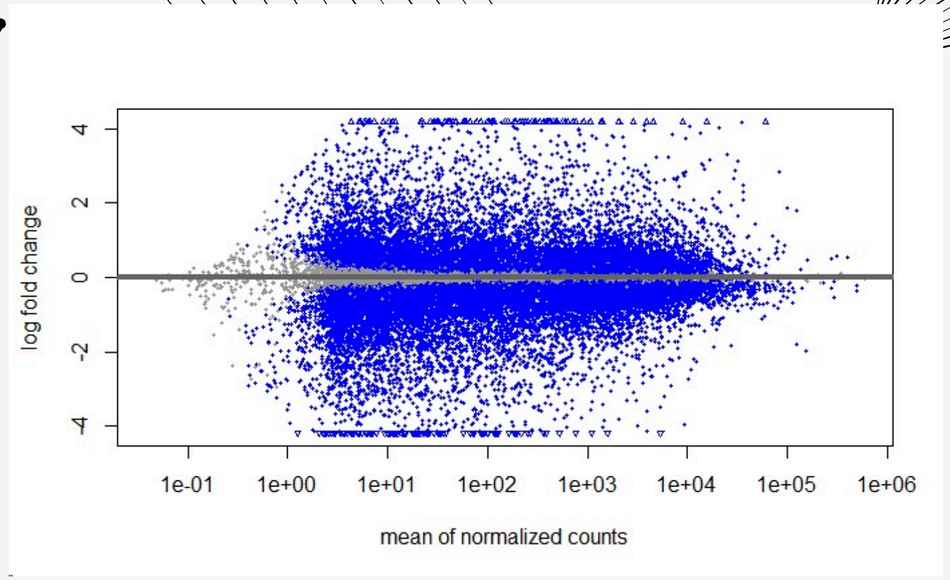




CODE

DESEQ ANALYSIS

```r
### DESeq Analysis
# Create the dds object
dds <- DESeqDataSetFromMatrix(countData = final_data_filtered, colData = er_status_filtered,
                              design = ~ ER_status)

# Filtering genes; different ways
# Calculate the 50th percentile of the sum of counts
threshold <- quantile(rowSums(counts(dds)), probs = 0.5)

# Filter genes based on the threshold
dds <- dds[rowSums(counts(dds)) > threshold, ]

# Normalization to account for different sequencing depth
dds <- estimateSizeFactors(dds)

dds <- DESeq(dds)

contrast_results <- results(dds, contrast=c("ER_status", "Positive", "Negative"))

# MA Plot
plotMA(contrast_results)
```



- First, create the dds object.
- Then you can create thresholds to filter out necessary genes.
- Finally, obtain the results contrasted against ER_status values.
- We can make an MA plot to display log fold change vs Average count values.

dds	S4 [30328 x 787] (DESeq2::DESec	S4 object of class DESeqDataSet
design	formula	~ER_status
dispersionFunction	function	function(q) { ... }
rowRanges	S4 (GenomicRanges::Compresse(	S4 object of class CompressedGRangesList
colData	S4 [787 x 4] (S4Vectors::DFrame)	S4 object of class DFrame
assays	S4 [30328 x 787] (SummarizedEx	S4 object of class SimpleAssays
NAMES	NULL	Pairlist of length 0
elementMetadata	S4 [30328 x 0] (S4Vectors::DFram	S4 object of class DFrame
metadata	list [1]	List of length 1

CODE

VIEWING THE RESULTS

```r
### Filtering results based on custom criteria
# Apply filtering criteria
filtered_results <- contrast_results[abs(contrast_results$log2FoldChange) >= 3
                                     & contrast_results$padj < 0.01, ]

# Add a new column "gene_name" to filtered_results with gene names from final_data
filtered_results$gene_name <- final_data[row.names(filtered_results), 1]

# View the filtered results
filtered_results

write_xlsx(as.data.frame(filtered_results), "C:/Users/sarth/Downloads/genes.csv")
```

- Finally, we can choose appropriate filtering criteria to select our differentially expressed genes.
- We can add the corresponding gene names to their ids.
- Export the results into an excel file.

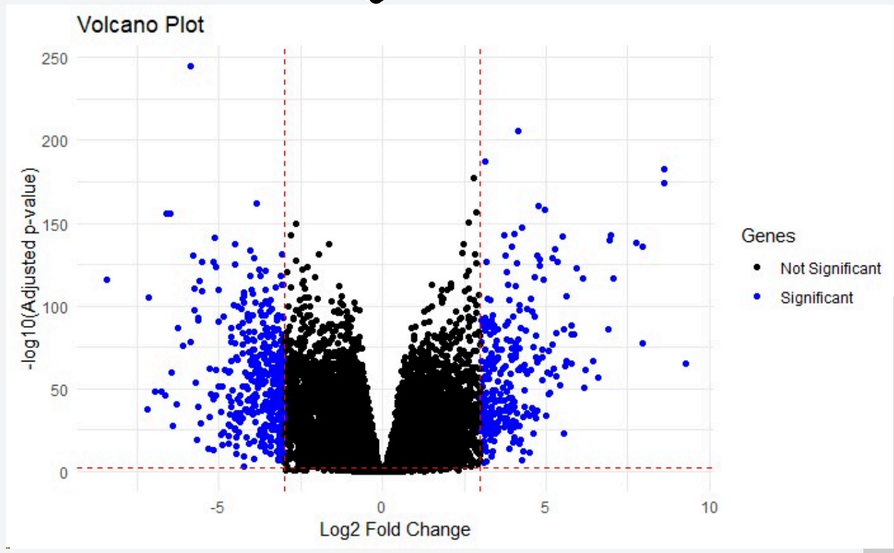
```
> filtered_results
log2 fold change (MLE): ER_status Positive vs Negative
Wald test p-value: ER status Positive vs Negative
DataFrame with 5765 rows and 7 columns
                    baseMean log2FoldChange     lfcSE      stat      pvalue        padj   gene_name
                   <numeric>      <numeric> <numeric> <numeric>   <numeric>   <numeric> <character>
ENSG00000001617.12  6375.7639        1.01870 0.0636284  16.01016 1.08528e-57 3.35518e-56       SEMA3F
ENSG00000001626.16    50.7170       -1.47765 0.1764493  -8.37436 5.55246e-17 2.42469e-16         CFTR
ENSG00000002079.14    16.2457       -1.22199 0.1168966 -10.45364 1.41012e-25 9.84488e-25        MYH16
ENSG00000003989.18 16913.2301        3.14582 0.1580578  19.90299 3.83360e-88 4.65062e-86       SLC7A2
ENSG00000004468.13   397.8802       -1.56685 0.1429086 -10.96397 5.69468e-28 4.49410e-27         CD38
...                       ...            ...       ...       ...         ...         ...          ...
ENSG00000288610.1     2.43880       -1.21340  0.188006  -6.45403 1.08915e-10 3.16760e-10  AL161669.4
ENSG00000288611.1     9.95262       -2.01531  0.214169  -9.40991 4.96572e-21 2.70669e-20       NPBWR1
ENSG00000288648.1     2.37882       -3.22087  0.350799  -9.18152 4.25039e-20 2.20013e-19  AL139190.1
ENSG00000288657.1     2.23697       -3.08894  0.445909  -6.92727 4.29026e-12 1.37528e-11  AL139280.3
ENSG00000288658.1    23.90743       -1.41714  0.180217  -7.86351 3.73507e-15 1.46334e-14  AC010980.1
```

CODE

VOLCANO PLOT

```r
### Create a volcano plot to visualize differential expression results
# Convert DESeqResults to a data frame
contrast_results_df <- as.data.frame(contrast_results)
# Plot
volcano_plot <- ggplot(contrast_results_df, aes(x = log2FoldChange, y = -log10(padj))) +
  geom_point(aes(color = ifelse(padj < 0.01 & abs(log2FoldChange) >= 3, "Significant", "Not Significant")))+
  geom_hline(yintercept = -log10(0.01), linetype = "dashed", color = "red") +
  geom_vline(xintercept = c(-3, 3), linetype = "dashed", color = "red") +
  labs(x = "Log2 Fold Change", y = "-log10(Adjusted p-value)") +
  theme_minimal() +
  ggtitle("Volcano Plot") +
  scale_color_manual(name = "Genes", values = c("Significant" = "blue", "Not Significant" = "black"))

print(volcano_plot)
```

- We can observe our significant genes by plotting them in a volcano plot.
- Here, we set the thresholds as Padj<0.01 and log2FoldCHange>3.



CODE

HEATMAP

```r
### Creating a heatmap for the significant genes
# Use variance stabilizing transformation
dds_vst <- vst(dds, blind = FALSE)
dds_vst <- assay(dds_vst)
dds_vst <- as.data.frame(dds_vst)

# Keep only the significantly differentiated genes where the fold-change was at least 3
sigGenes <- rownames(filtered_results)
dds_vst <- dds_vst[rownames(dds_vst) %in% sigGenes,]

# Create a new data frame with gene names as the first column
genes_column <- data.frame(genes = filtered_results$gene_name)
# Add the remaining columns from dds_vst
dds_vst <- cbind(genes_column, dds_vst)

# Convert it to long format using melt
dds_vst_melted <- melt(dds_vst, id.vars = "genes")
```

- To create a heatmap of the dds object we'll first run variance stabilising transformation(VST) on the object.
- Then from the previously obtained significant genes we filter the dds_vst object. We are left with around 673 genes.
- We melt the data into long format which is required for getting a heatmap.

dds_vst

	genes	TCGA-D8-A1XO-01A	TCGA-AN-A0FN-01A	TCGA-BH-A18M-01A
ENSG00000003989.18	SLC7A2	10.934817	11.079144	15.148822
ENSG00000005513.10	SOX8	5.164461	6.707382	5.353675
ENSG00000007372.24	PAX6	5.896562	5.838647	4.197148
ENSG00000010438.17	PRSS3	3.699067	3.301242	3.754560
ENSG00000047936.11	ROS1	4.825674	4.962999	5.425173
ENSG00000054598.9	FOXC1	7.550064	10.285325	8.733843
ENSG00000054938.16	CHRDL2	6.721673	6.245702	7.635202

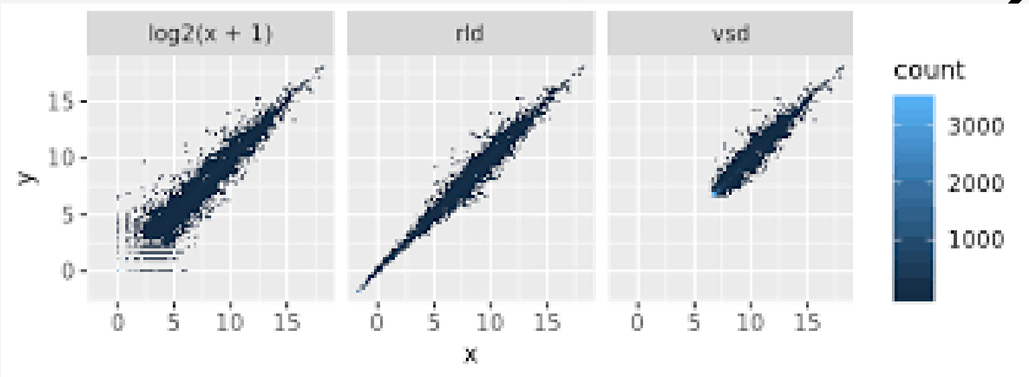
dds_vst_melted

genes	variable	value
SLC7A2	TCGA-D8-A1XO-01A	10.934817
SOX8	TCGA-D8-A1XO-01A	5.164461
PAX6	TCGA-D8-A1XO-01A	5.896562
PRSS3	TCGA-D8-A1XO-01A	3.699067
ROS1	TCGA-D8-A1XO-01A	4.825674
FOXC1	TCGA-D8-A1XO-01A	7.550064
CHRDL2	TCGA-D8-A1XO-01A	6.721673
SERPINB3	TCGA-D8-A1XO-01A	3.301242
ARSF	TCGA-D8-A1XO-01A	3.699067
ROPN1	TCGA-D8-A1XO-01A	3.301242

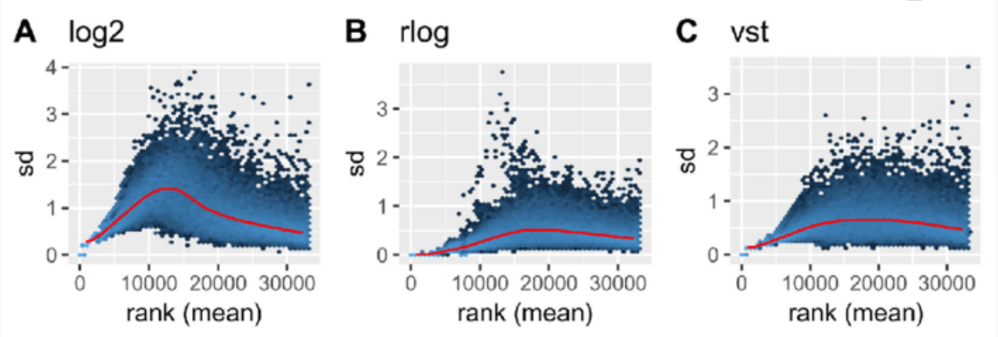
TRANSFORMATIONS ON VARIANCE

Here are some differences between VST and rlog:

- **Speed:** VST runs faster than rlog.
- **Sensitivity to outliers:** VST is less sensitive to high count outliers than rlog.
- **Data:** distribution: rlog relies more heavily on the assumption of the data distribution.
- **Size factor:** VST takes into account both size factor and mean dispersion across the sample, while rlog only accounts for size factor.
- **Library size:** rlog transformation is a better option than VST if the library size of the samples and therefore their size factors vary widely.
- **Dataset size:** rlog tends to work well on small datasets (n < 30).



- All of them aim to remove the dependence of the variance on the mean.
- We use VST in our example because it works well with larger sample size and is also faster.



Variance Stabilising Transformations(VST): VST is used to stabilize the variance across different expression levels. It models the relationship between the mean and variance of gene expression.

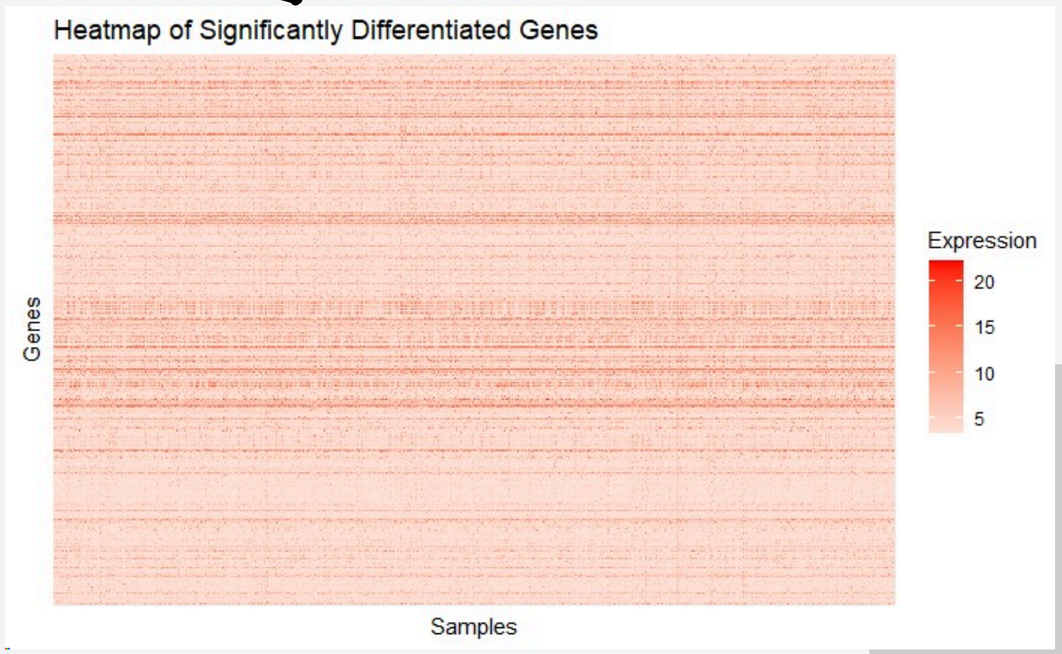
rlog transformation: It aims to stabilize the variance and make the data more suitable for statistical analysis. It also has a regularized component that helps with small sample sizes.

log2 transformation: The log2 transformation is a simple logarithmic transformation that takes the base-2 logarithm of the data. It is commonly used to reduce the dynamic range of data and make it more amenable to statistical analysis.

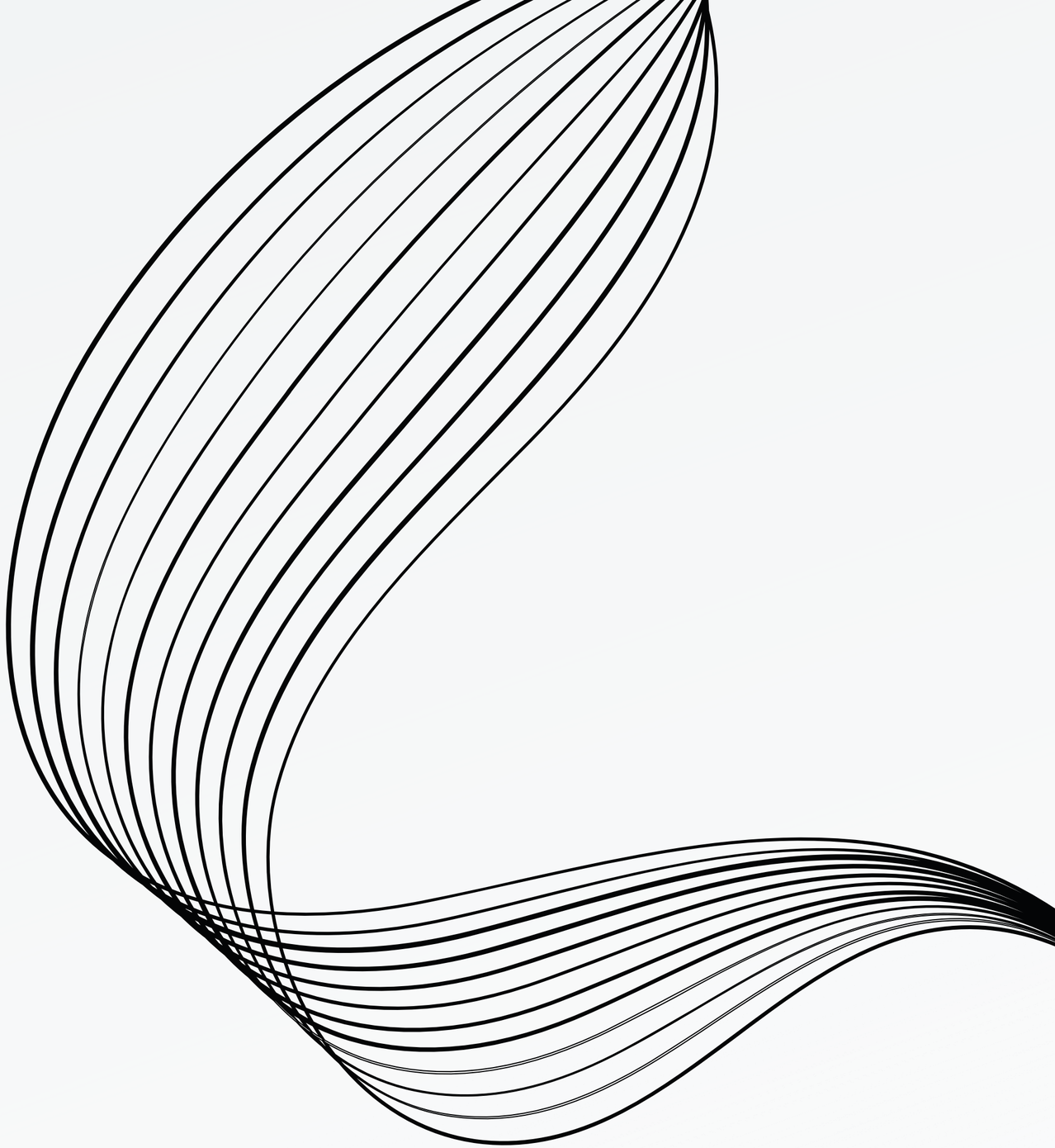
CODE

```r
# Plot
heatmap_plot <- ggplot(dds_vst_melted, aes(x = variable, y = genes, fill = value)) +
  geom_tile() +
  scale_fill_gradient2(low = "blue", high = "red", midpoint = 0) +
  labs(x = "Samples", y = "Genes", fill = "Expression") +  # Set axis labels
  theme_minimal() +
  theme(axis.text.x = element_blank(),  # Remove text on x-axis
        axis.text.y = element_blank()) + # Remove text on y-axis
 ggtitle("Heatmap of Significantly Differentiated Genes")

# Print the heatmap
print(heatmap_plot)
```

- Finally, we obtain the heatmap from the transformed object.
- This heatmap consists of all the significant genes only.



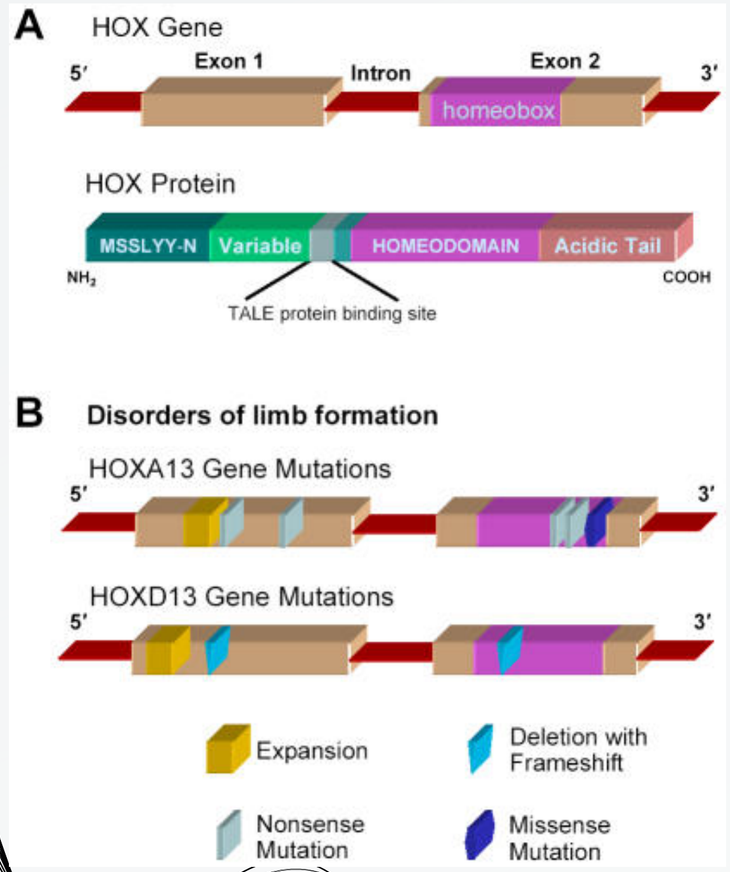
THANK YOU

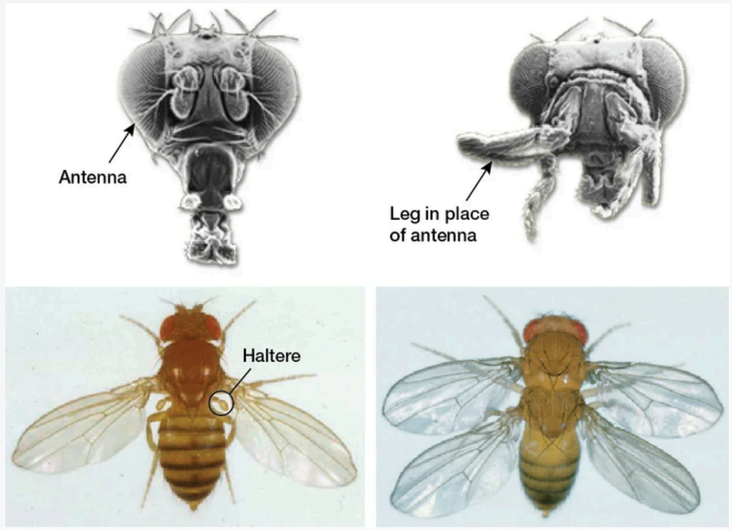


ANALYSIS OF HOXB2 AND MMP11 GENES

SARTHAK RAY

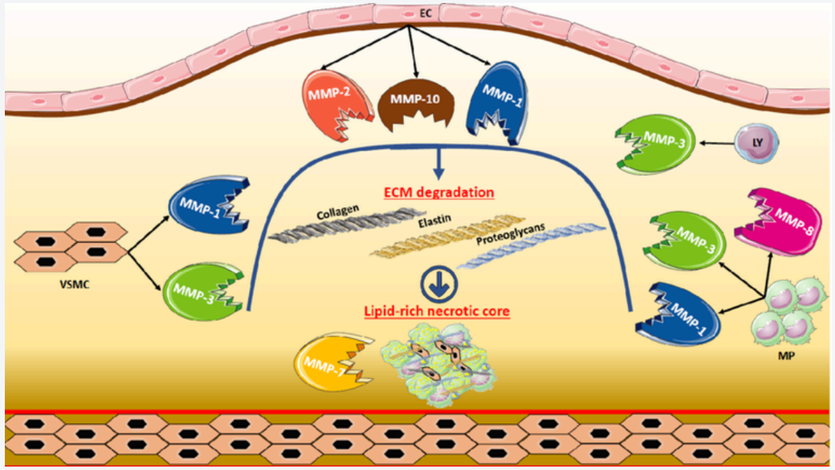
HOX GENES

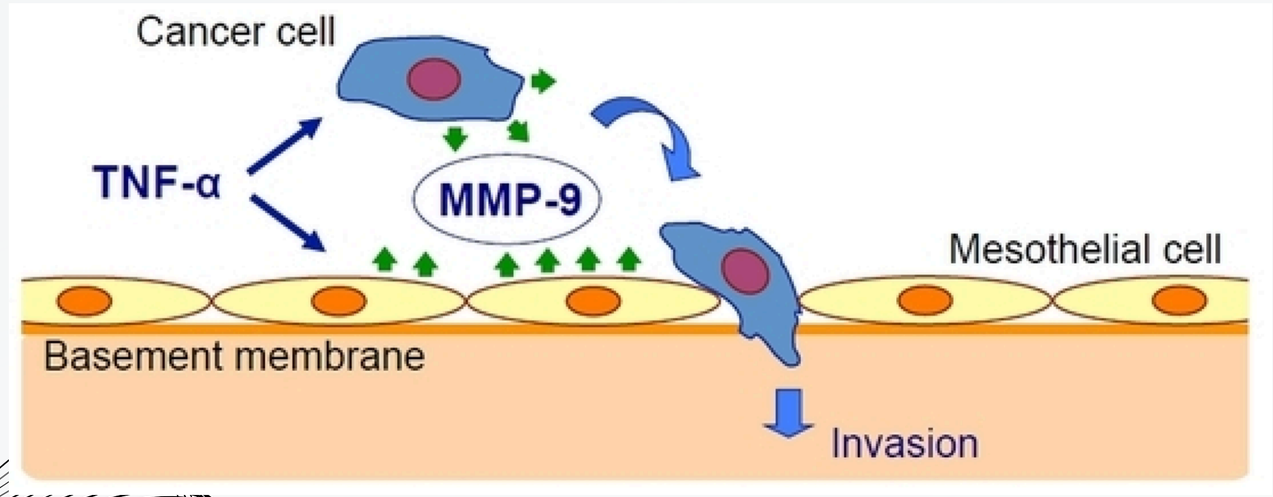




- Scientists discovered homeotic genes by studying **strange transformations** in fruit flies, including flies with feet instead of mouthparts or extra pairs of wings.
- Researchers discovered that many of these transformations were caused by defects in single genes, which they termed **homeotic or Hox**.
- This work demonstrated that antennal cells carry all of the information necessary to become leg cells. This is a general principle: every cell in an organism carries, within its DNA, all of the information necessary to build the entire organism.

MMP GENES





- **Matrix metalloproteinases (MMPs)** are a family of zinc-dependent extracellular matrix (ECM) remodelling endopeptidases that can degrade almost every ECM component.
- When the expression of MMPs is altered, it can generate abnormal degradation of the ECM. This process has an association with neurodegeneration and **cancer progression**.
- MMPs exert these effects by **cleaving** a diverse group of substrates like growth-factor-binding proteins, growth-factor precursors, receptor tyrosine kinases, cell-adhesion molecules and other proteinases.

GENE LISTS

HOX GENES

ENSG00000005073.6	HOXA11
ENSG00000037965.6	HOXC8
ENSG00000078399.19	HOXA9
ENSG00000105991.10	HOXA1
ENSG00000105996.7	HOXA2
ENSG00000105997.22	HOXA3
ENSG00000106004.5	HOXA5
ENSG00000106006.6	HOXA6
ENSG00000106031.9	HOXA13
ENSG00000108511.10	HOXB6
ENSG00000120068.7	HOXB8
ENSG00000120075.5	HOXB5
ENSG00000120093.11	HOXB3
ENSG00000120094.9	HOXB1
ENSG00000122592.8	HOXA7
ENSG00000123364.5	HOXC13
ENSG00000123388.4	HOXC11
ENSG00000123407.4	HOXC12
ENSG00000128645.15	HOXD1
ENSG00000128652.11	HOXD3
ENSG00000128709.13	HOXD9
ENSG00000128710.6	HOXD10
ENSG00000128713.14	HOXD11
ENSG00000128714.6	HOXD13
ENSG00000159184.8	HOXB13
ENSG00000170166.6	HOXD4
ENSG00000170178.7	HOXD12
ENSG00000170689.10	HOXB9
ENSG00000172789.3	HOXC5
ENSG00000173917.10	HOXB2
ENSG00000175879.9	HOXD8
ENSG00000180806.5	HOXC9
ENSG00000180818.5	HOXC10
ENSG00000182742.6	HOXB4
ENSG00000197576.14	HOXA4
ENSG00000197757.8	HOXC6
ENSG00000198353.8	HOXC4
ENSG00000230148.8	HOXB-AS1
ENSG00000233101.10	HOXB-AS3
ENSG00000237380.7	HOXD-AS2
ENSG00000239552.2	HOXB-AS2
ENSG00000240990.10	HOXA11-AS
ENSG00000242207.1	HOXB-AS4
ENSG00000249641.2	HOXC13-AS
ENSG00000250133.2	HOXC-AS2
ENSG00000250451.5	HOXC-AS1
ENSG00000251151.2	HOXC-AS3
ENSG00000253187.2	HOXA10-AS
ENSG00000253293.5	HOXA10
ENSG00000253552.8	HOXA-AS2
ENSG00000254369.6	HOXA-AS3
ENSG00000260027.5	HOXB7

MMP GENES

ENSG00000008516.17	MMP25
ENSG00000087245.13	MMP2
ENSG00000099953.10	MMP11
ENSG00000100985.7	MMP9
ENSG00000102996.5	MMP15
ENSG00000118113.12	MMP8
ENSG00000123342.16	MMP19
ENSG00000125966.10	MMP24
ENSG00000126005.17	MMP24OS
ENSG00000137673.9	MMP7
ENSG00000137674.4	MMP20
ENSG00000137675.5	MMP27
ENSG00000137745.12	MMP13
ENSG00000149968.12	MMP3
ENSG00000154485.5	MMP21
ENSG00000156103.16	MMP16
ENSG00000157227.13	MMP14
ENSG00000166670.10	MMP10
ENSG00000167346.7	MMP26
ENSG00000189409.14	MMP23B
ENSG00000196611.5	MMP1
ENSG00000198598.6	MMP17
ENSG00000215914.4	MMP23A
ENSG00000260135.7	MMP2-AS1
ENSG00000261971.8	MMP25-AS1
ENSG00000262406.3	MMP12
ENSG00000271447.6	MMP28

CODE

DATA SEGREGATION

```r
hox_genes <- final_data_filtered[grepl("^HOX", final_data_filtered$GeneNames), ]

transposed_data <- t(hox_genes[, -1])  # Exclude the column with gene names

# Set the transposed gene names as column names
colnames(transposed_data) <- hox_genes$GeneNames

transposed_data <- data.frame(transposed_data)
transposed_data$status <- er_status_filtered$ER_status

# Reorder columns to place 'status' at the beginning
transposed_data <- transposed_data[, c('status', setdiff(names(transposed_data), 'status'))]

hox_data <- transposed_data
```

- We use the grep keyword to filter out the required genes and make two separate data frames.
- We make the necessary transitions and added the ER-alpha status column to the data
- Finally, we transpose the data to keep samples as rows and genes as columns.

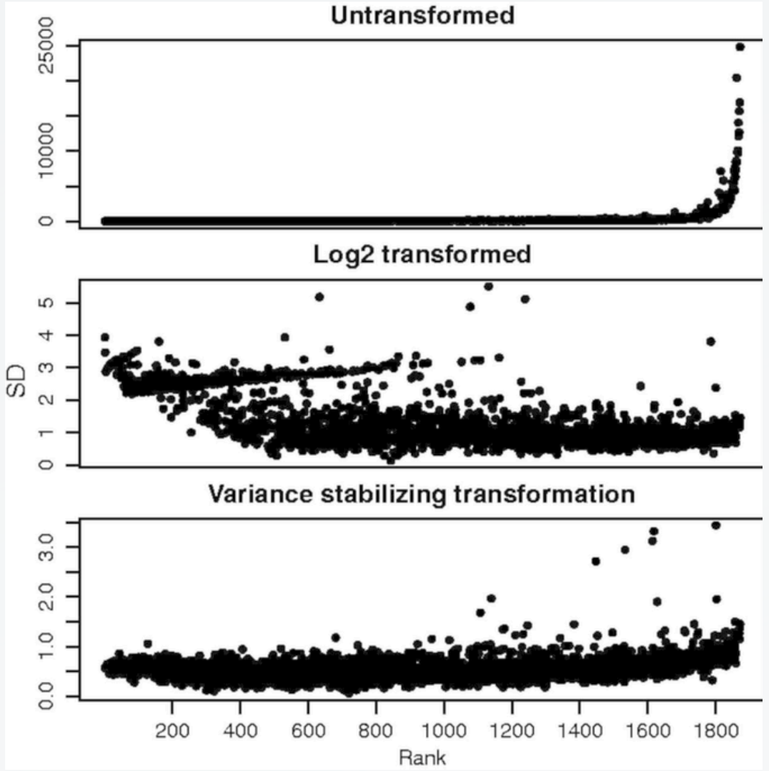
HOXB2

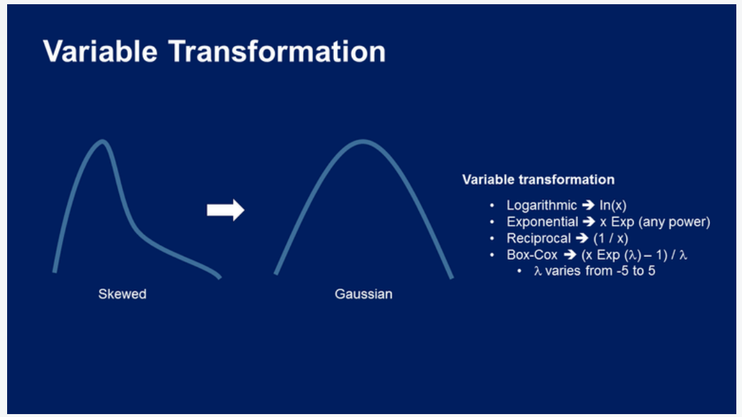
	status	HOXA11	HOXC8	HOXA9	HOXA1	HOXA2	HOXA3
TCGA-D8-A1XO-01A	Positive	118	586	70	79	14	85
TCGA-AN-A0FN-01A	Positive	305	906	108	142	66	329
TCGA-BH-A18M-01A	Positive	26	304	78	50	35	103
TCGA-E2-A15T-01A	Positive	4	448	13	5	4	15
TCGA-AR-A0TU-01A	Negative	19	252	8	37	77	422
TCGA-EW-A1IX-01A	Positive	123	380	64	56	31	93
TCGA-E2-A1IE-01A	Positive	21	193	50	26	20	51
TCGA-E2-A10E-01A	Positive	30	213	46	54	8	55
TCGA-BH-A0H5-01A	Positive	8	241	725	80	40	127
TCGA-E2-A14W-01A	Positive	26	438	18	21	13	80
TCGA-AQ-A04L-01B	Positive	73	480	7058	130	259	822

MMP11

	status	MMP25	MMP2	MMP11	MMP9
TCGA-D8-A1XO-01A	Positive	65	63469	34960	173
TCGA-AN-A0FN-01A	Positive	127	145328	89569	4109
TCGA-BH-A18M-01A	Positive	48	41191	35480	280
TCGA-E2-A15T-01A	Positive	524	6779	2776	3204
TCGA-AR-A0TU-01A	Negative	161	4141	2874	3062
TCGA-EW-A1IX-01A	Positive	148	63058	38984	3221
TCGA-E2-A1IE-01A	Positive	27	12208	17144	896
TCGA-E2-A10E-01A	Positive	108	17573	55537	1003

VARIANCE STABILISING TRANSFORMATION





- In applied statistics, a **variance-stabilizing transformation** is a data transformation that is specifically chosen to simplify considerations in graphical exploratory data analysis and to allow the analysis of variance techniques.
- **RNA-seq data** often exhibit **heteroscedasticity**, meaning that the variance of the counts depends on the mean count, which violates the assumptions of many statistical methods.
- In R, the varianceStabilizingTransformation function calculates a VST from **dispersion-mean relations** and transforms count data to produce a matrix of values with constant variance.

CODE

VARIANCE STABILIZATION TRANSFORMATION

```r
# Extract expression matrix (exclude the ER status column)
expression_matrix <- as.matrix(hox_data[, -1])

# Perform variance stabilizing transformation directly
vst_data <- varianceStabilizingTransformation(expression_matrix)

# If you want to add ER status column back to the vst_data
hox_vst_data <- cbind(hox_data[, 1, drop = FALSE], vst_data)
```

- Our data frames do not have a consistent variance.
- Therefore, we use the variance stabilizing transformation function built in R.
- This helps us normalise the values, and you can see the resulting dataframes in the adjacent images.

	status	HOXA11	HOXC8	HOXA9	HOXA1
TCGA-D8-A1XO-01A	Positive	6.81046197	5.824990269	5.0270011	5.9666634
TCGA-AN-A0FN-01A	Positive	8.41180036	6.784593538	6.0625935	7.1917123
TCGA-BH-A18M-01A	Positive	3.32159698	4.246199321	5.2900059	4.8680875
TCGA-E2-A15T-01A	Positive	-0.07788552	5.179997043	1.4321983	0.2788887
TCGA-AR-A0TU-01A	Negative	2.65891400	3.812009597	0.5961335	4.1486142
TCGA-EW-A1IX-01A	Positive	6.89279877	4.779938578	4.8092657	5.1436404
TCGA-E2-A1IE-01A	Positive	2.86577504	3.220367115	4.2188919	3.3505337

HOXB2

	status	MMP25	MMP2	MMP11	MMP9
TCGA-D8-A1XO-01A	Positive	2.4012854904	8.542208482	6.56554306	-0.29484509
TCGA-AN-A0FN-01A	Positive	4.1130801296	10.249074784	9.00454987	7.05794692
TCGA-BH-A18M-01A	Positive	1.7299618063	7.438763179	6.60902904	0.42149321
TCGA-E2-A15T-01A	Positive	8.1621986756	2.417079361	0.39455843	6.33086683
TCGA-AR-A0TU-01A	Negative	4.7866249585	1.358957179	0.45081672	6.19556249
TCGA-EW-A1IX-01A	Positive	4.5442167042	8.526843370	6.88378618	6.34662400
TCGA-E2-A1IE-01A	Positive	0.6416023459	3.907360293	4.45885739	2.76358130
TCGA-E2-A10E-01A	Positive	3.6716267319	4.938836683	7.85898629	3.04249664

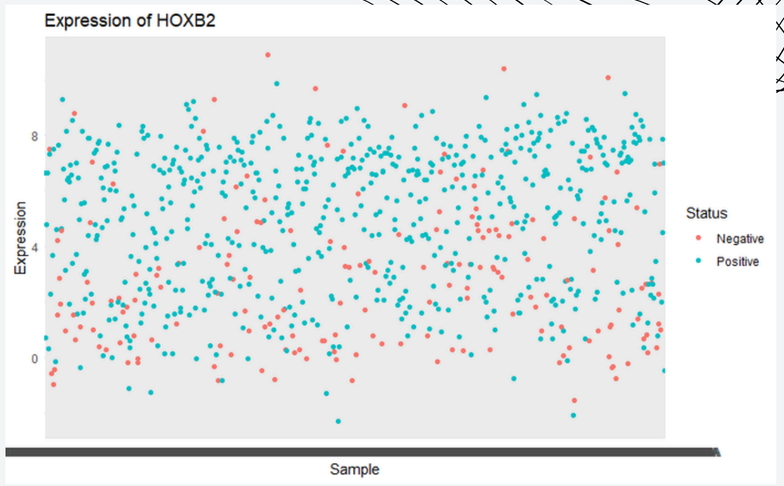
MMP11

CODE

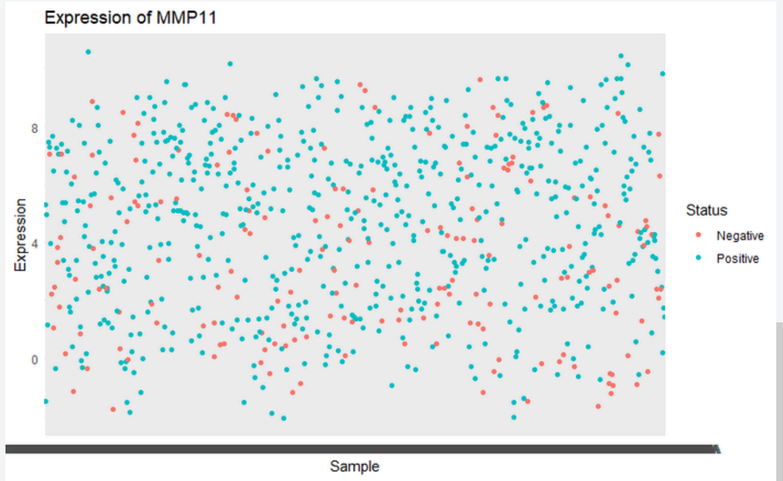
DATA REPRESENTATION

```r
# Initialize a list to store plots
gene_plots <- list()

# Iterate over each gene and create separate plots
for (gene in colnames(gene_expression_data)) {
  # Create a dataframe for the current gene
  gene_data <- data.frame(Sample = rownames(hox_vst_data), Gene = rep(gene, nrow(hox_vst_data)),
                          Expression = gene_expression_data[, gene], Status = status_column)

  # Plot using ggplot2
  gene_plot <- ggplot(gene_data, aes(x = Sample, y = Expression, color = Status)) +
    geom_point() +
    labs(title = paste("Expression of", gene),
         x = "Sample", y = "Expression") +
    theme_minimal()

  # Save the plot to the list
  gene_plots[[gene]] <- gene_plot

}
```

- We can now use the ggplot2 package to get a graphical representation of the data frames.
- This helps us get a rough idea of the distribution between positive and negative samples.
- As we can see, in both genes, the positive samples tend to have a higher expression value.

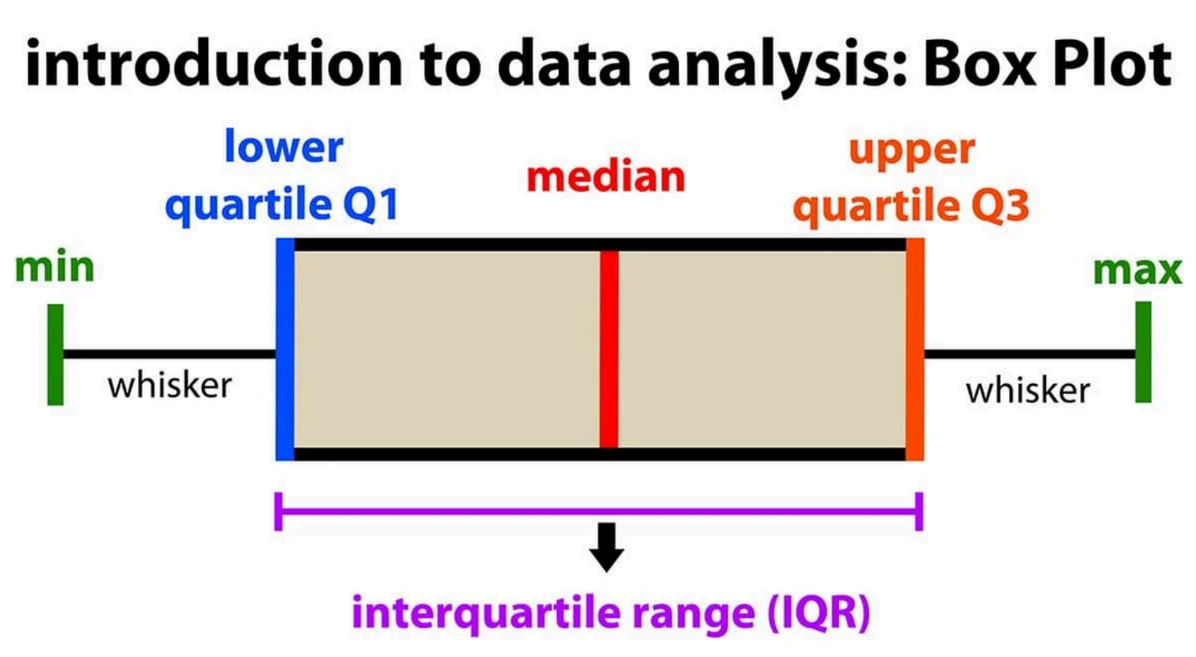


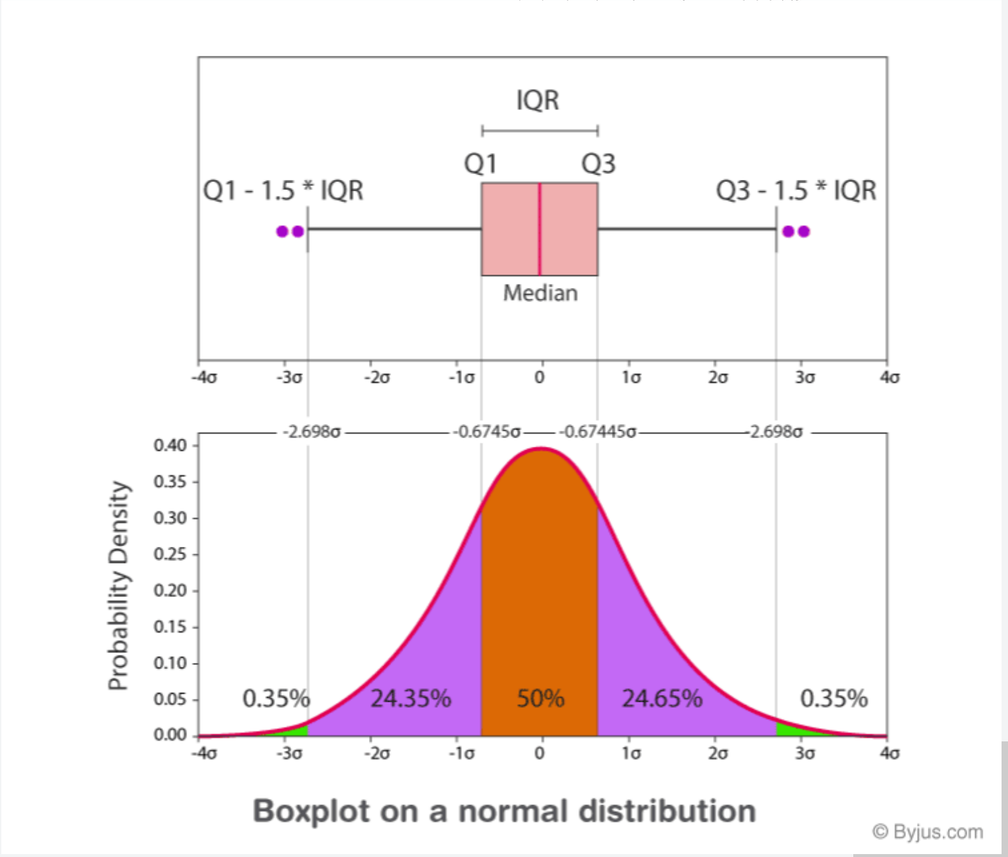
HOXB2



MMP11

BOX PLOT


introduction to data analysis: Box Plot
lower
quartile Q1
median
upper
quartile Q3
min
max
whisker
whisker
interquartile range (IQR)

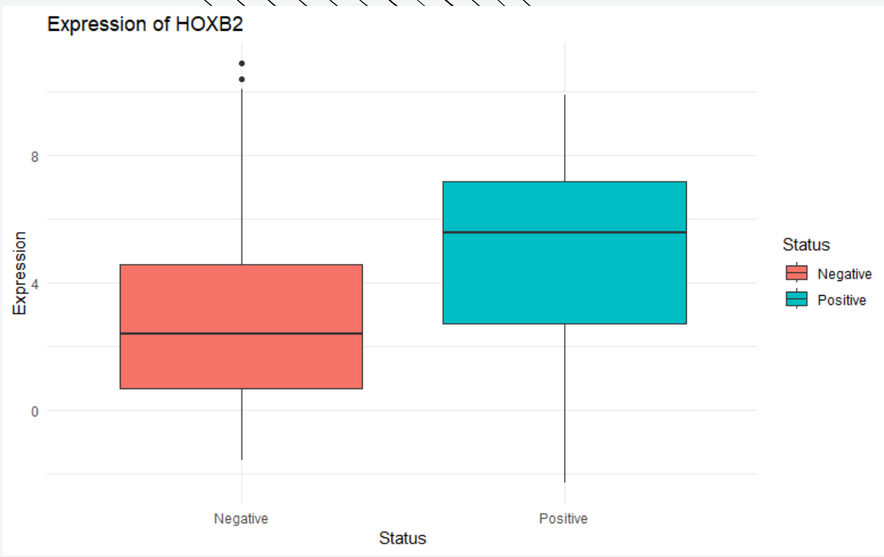

IQR
Q1
Q3
Q1 - 1.5 * IQR
Q3 - 1.5 * IQR
Median
-4σ
-3σ
-2σ
-1σ
0
1σ
2σ
3σ
4σ
-2.698σ
-0.6745σ
-0.67445σ
-2.698σ
Probability Density
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0.35%
24.35%
50%
24.65%
0.35%
Boxplot on a normal distribution
© Byjus.com

CODE

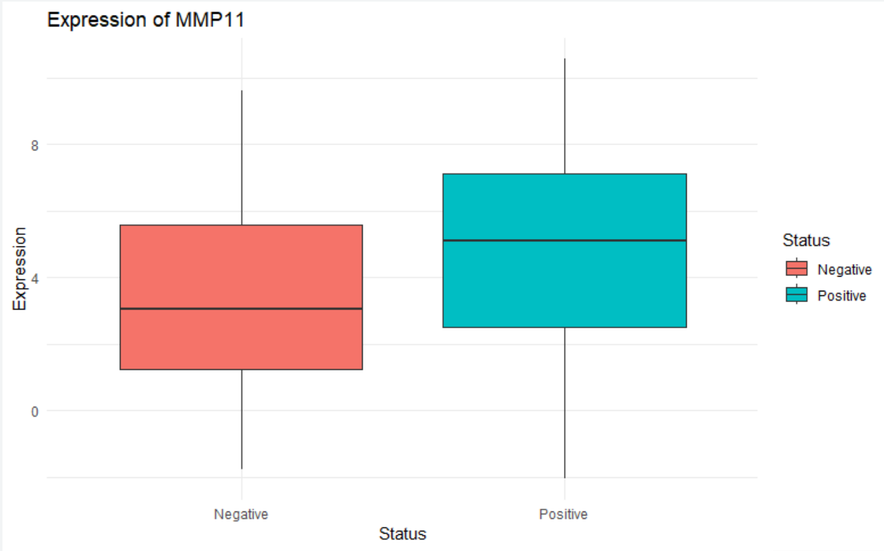
BOX PLOTS

```r
# Initialize a list to store box plots
box_plots <- list()

# Iterate over each gene and create separate box plots
for (gene in colnames(gene_expression_data)) {
  # Create a dataframe for the current gene
  gene_data <- data.frame(Gene = rep(gene, nrow(gene_expression_data)),
                          Expression = gene_expression_data[, gene],
                          Status = status_column)

  # Plot using ggplot2
  box_plot <- ggplot(gene_data, aes(x = Status, y = Expression, fill = Status)) +
    geom_boxplot() +
    labs(title = paste("Expression of", gene),
         x = "Status", y = "Expression") +
    theme_minimal()

  # Save the box plot to the list
  box_plots[[gene]] <- box_plot

}
```

- Box plots are an efficient way to visualise key features in a dataset.
- Using ggplot2, we make box plots of both the genes and contrast between the positive and negative samples.
- It gives us an idea of the median, range, interquartile regions, and any existing outliers.

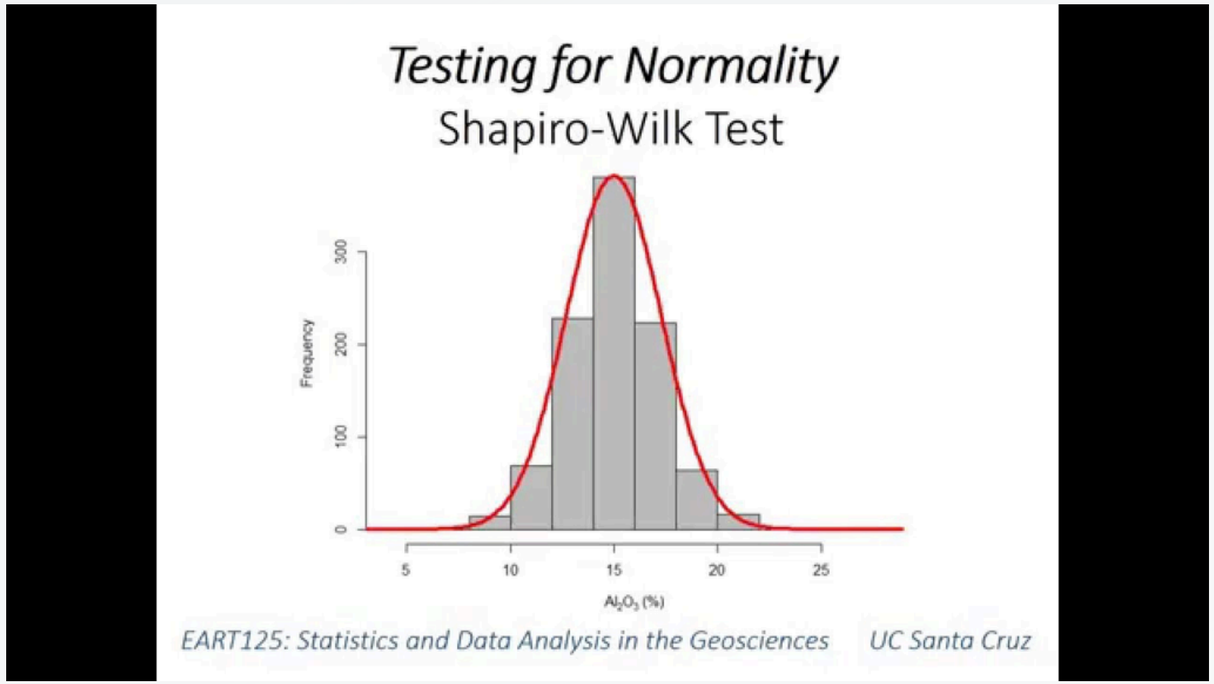


HOXB2



MMP11

SHAPIRO-WILK'S TEST (NORMALITY)



$$W = \frac{\left\{\sum_{i-1}^{n} a_i (x_{(n-i+1):n} - x_{i:n})\right\}^2}{\sum_{i=1}^{n} (x_i - \bar{x})_2},$$

- The **Shapiro–Wilk test** is a **nonparametric** statistical tool that evaluates whether a data set is normally distributed. It's a goodness-of-fit test that examines how closely sample data fit to a **normal distribution**.

- The null hypothesis of the Shapiro–Wilk test is that the data set is normally distributed. A **large p-value** indicates that the data set is **normally distributed,** while a low p-value indicates that it isn't.

- The Shapiro-Wilk test is sensitive to departures from normality in the distribution's tails, making it effective for detecting deviations from normality.

CODE

NORMALITY TEST - SHAPIRO WILK'S TEST

```r
hoxb2_data <- data.frame(Gene = rep("HOXB2", nrow(gene_expression_data)),
                         Expression = gene_expression_data[, "HOXB2"],
                         Status = status_column)

# Extract expression data for positive and negative status
positive_data <- hoxb2_data$Expression[hoxb2_data$Status == "Positive"]
negative_data <- hoxb2_data$Expression[hoxb2_data$Status == "Negative"]

# Perform Shapiro-Wilk test for normality
shapiro_positive <- shapiro.test(positive_data)
shapiro_negative <- shapiro.test(negative_data)

# Print the test results
print("Shapiro-Wilk Test for Normality - Positive Status:")
print(shapiro_positive)
print("Shapiro-Wilk Test for Normality - Negative Status:")
print(shapiro_negative)
```

```
> # Print the test results
> print("Shapiro-Wilk Test for Normality - Positive Status:")
[1] "Shapiro-Wilk Test for Normality - Positive Status:"
> print(shapiro_positive)

        Shapiro-Wilk normality test

data:  positive_data
W = 0.94991, p-value = 1.817e-13

> print("Shapiro-Wilk Test for Normality - Negative Status:")
[1] "Shapiro-Wilk Test for Normality - Negative Status:"
> print(shapiro_negative)

        Shapiro-Wilk normality test

data:  negative_data
W = 0.94135, p-value = 9.362e-07
```

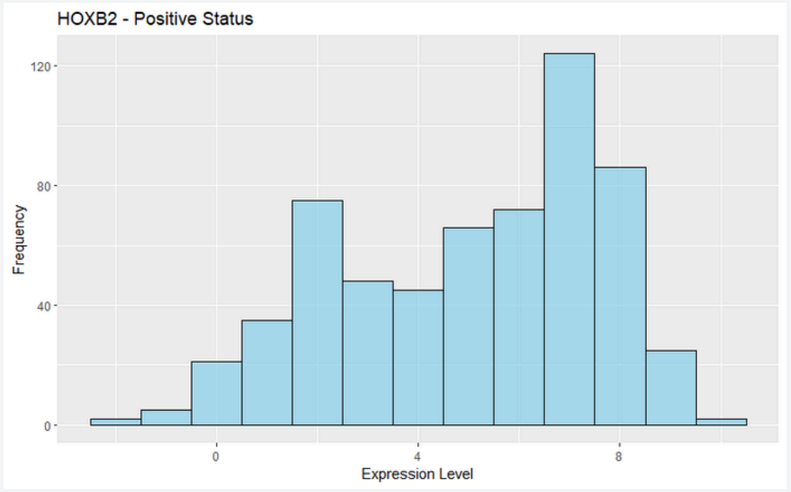
HOXB2

- To conduct a test of normality, we use the Shapiro-Wilk's test.
- The results of this test give us a test statistic and a p-value, which helps us determine whether or not the data is normal.
- We test both genes' divisions and get all four p-values to be less than our chosen significance value of 0.05.
- Hence, the data sets are not normally distributed.

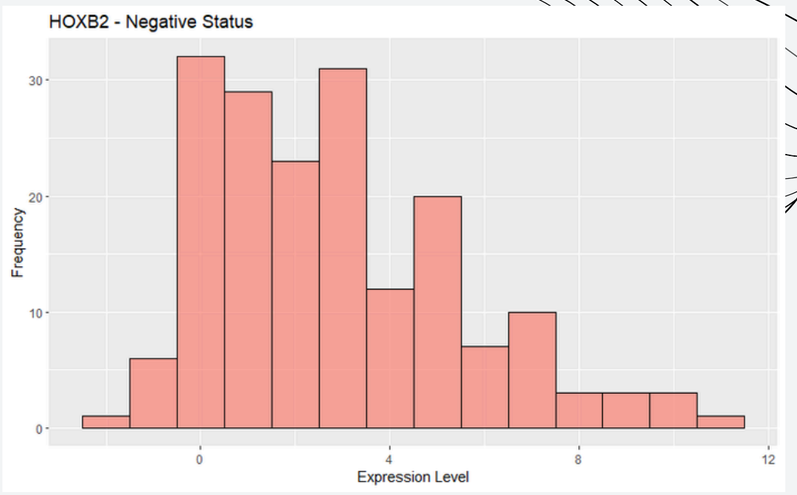
```
> # Print the test results
> print("Shapiro-Wilk Test for Normality - Positive Status:")
[1] "Shapiro-Wilk Test for Normality - Positive Status:"
> print(shapiro_positive)

        Shapiro-Wilk normality test

data:  positive_data
W = 0.97366, p-value = 5.654e-09

> print("Shapiro-Wilk Test for Normality - Negative Status:")
[1] "Shapiro-Wilk Test for Normality - Negative Status:"
> print(shapiro_negative)

        Shapiro-Wilk normality test

data:  negative_data
W = 0.96764, p-value = 0.0003303
```

MMP11

HISTOGRAMS

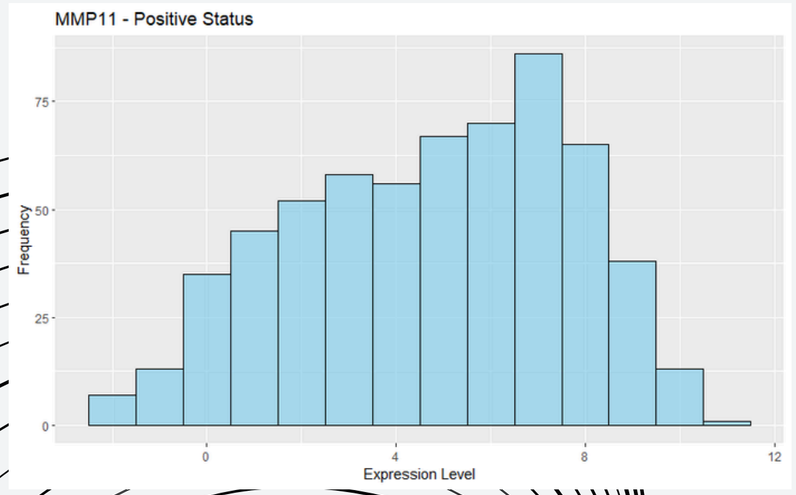


HOXB2

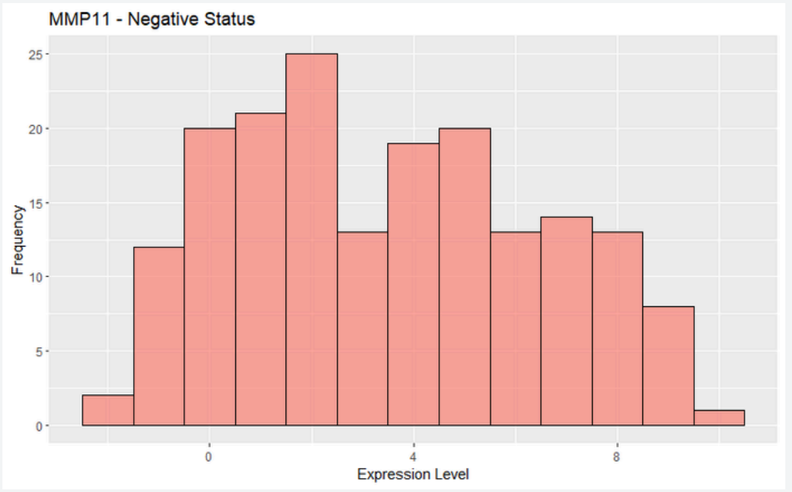


POSITIVE STATUS

NEGATIVE STATUS



MMP11



SIGNIFICANCE TESTS

T-TEST

- The t-test assumes that the data are approximately normally distributed within each group and that the variances of the two groups are equal.
- The t-test compares the means of two groups and calculates a test statistic (t-value) based on the difference between the means and the standard errors of the means.
- A significant result (small p-value) indicates evidence of a difference in means between the two groups.
- The t-test is more sensitive and powerful when the large sample size and the data are normally distributed.

Two-Sample T-Test

$$t = \frac{(\bar{x}_1 - \bar{x}_2)}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

$\bar{x}_1$ = observed mean of 1st sample
$\bar{x}_2$ = observed mean of 2nd sample
s_1 = standard deviation of 1st sample
s_2 = standard deviation of 2nd sample
n_1 = sample size of 1st sample
n_2 = sample size of 2nd sample

Wilcoxon Rank-Sum Test

Test statistic:

Combine the data from both samples and rank it.
R = the sum of the ranks for the smaller sample.
Find the z-score for the value of R.

where

$$z = \frac{R - \mu_R}{\sigma_R}$$

$$\mu_R = \frac{n_1(n_1 + n_2 + 1)}{2} \qquad \sigma_R = \sqrt{\frac{n_1 n_2(n_1 + n_2 + 1)}{12}}$$

WILCOXON RANK-SUM TEST

- The Wilcoxon rank-sum test is a non-parametric test that does not rely on assumptions about the data distribution.
- The Wilcoxon rank-sum test compares the medians of two groups by assigning ranks to the combined data and calculating a test statistic (U or W) based on the sum of the ranks in each group.
- A significant result (small p-value) indicates evidence of a difference in medians between the two groups.
- The Wilcoxon rank-sum test is robust to the data distribution and is often preferred when the sample size is small or when the data are not normally distributed.

CODE

SIGNIFICANCE TEST - WILCOXON RANK-SUM TEST

```r
# Perform Wilcoxon rank-sum test
wilcox_test_result <- wilcox.test(positive_data, negative_data)
print(wilcox_test_result)
```

```
        Wilcoxon rank sum test with continuity correction

data:  positive_data and negative_data
W = 80509, p-value < 2.2e-16
alternative hypothesis: true location shift is not equal to 0
```

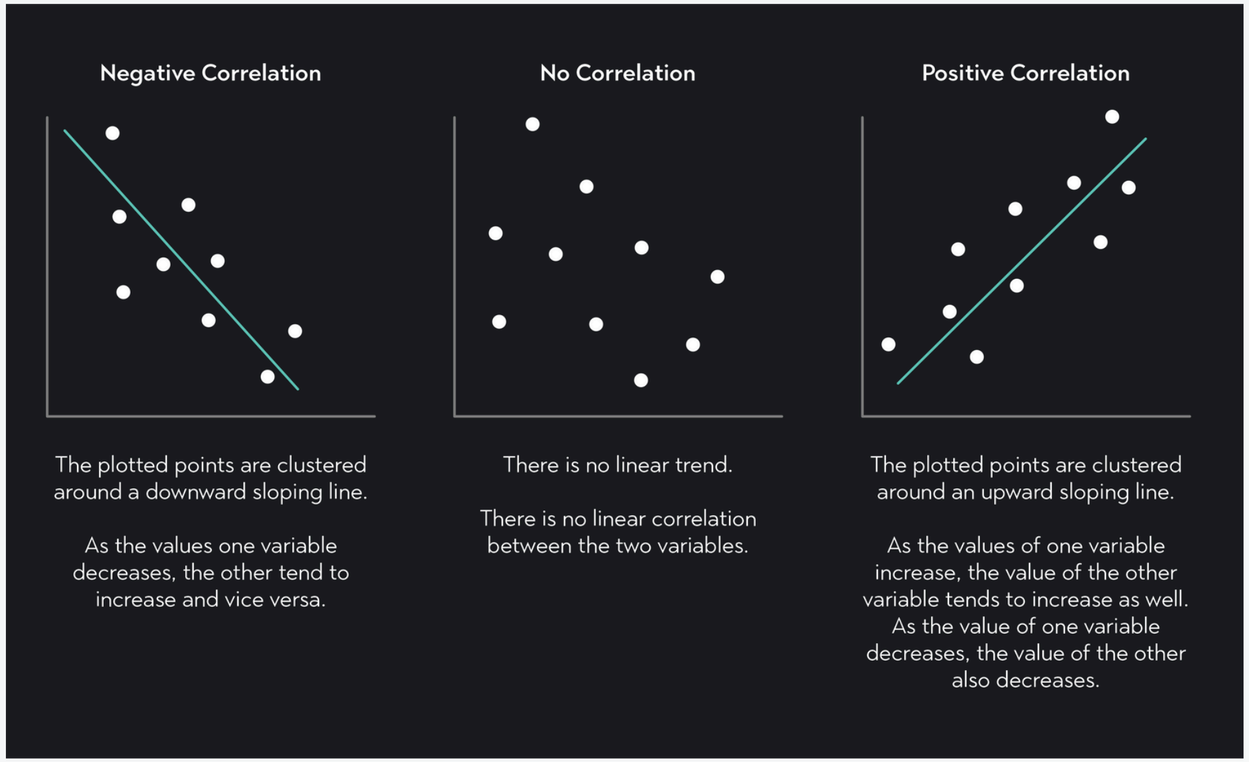
HOXB2

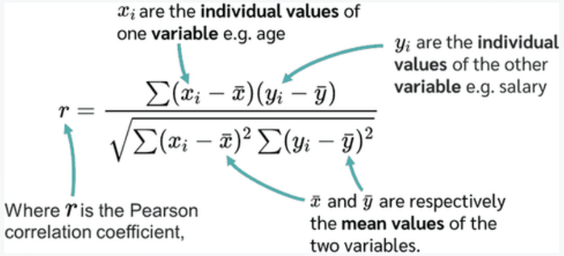
- Since the data was not normally distributed, we used the Wilcoxon rank-sum test to check the significance of the p-values.
- The test gives us a test statistic and a p-value.
- As we can see, both the p-values are less than 0.05, and thus, we can conclude that there is a significant difference between the positive and negative values.

```
        Wilcoxon rank sum test with continuity correction

data:  positive_data and negative_data
W = 68174, p-value = 6.805e-07
alternative hypothesis: true location shift is not equal to 0
```

MMP11

PEARSON'S CORRELATION TEST





- **Pearson's correlation** is a statistical method that measures the strength and direction of a linear relationship between two continuous variables.

- The Pearson correlation coefficient, denoted by r, ranges between -1 and 1, where:

1. r**=1**: Indicates a perfect positive linear relationship, meaning that as one variable increases, the other also increases proportionally.
2. r**=−1**: Indicates a perfect negative linear relationship, meaning that as one variable increases, the other variable decreases proportionally.
3. r**=0**: Indicates no linear relationship between the variables.

CODE

CORRELATION

```r
# Create an empty data frame to store correlation results
correlation_results <- data.frame(Gene = character(), Correlation = numeric(), stringsAsFactors = FALSE)

# Iterate over each gene and compute correlation with status
for (gene in colnames(gene_expression_data)) {
  # Create a dataframe for the current gene
  gene_data <- data.frame(Expression = gene_expression_data[, gene], Status = status_column)

  # Convert 'Status' to a binary variable
  gene_data$Status_binary <- ifelse(gene_data$Status == "Positive", 1, 0)

  # Compute correlation between status and gene expression
  correlation_value <- cor.test(gene_data$Expression, gene_data$Status_binary)$estimate

  # Append results to the data frame
  correlation_results <- rbind(correlation_results, data.frame(Gene = gene, Correlation = correlation_value))
}

correlation_results <- correlation_results[order(-(correlation_results$Correlation)), ]
```

- We can also find the correlation between our chosen genes and all the other genes.
- We use the built-in cor.test function, which uses Pearson's test.
- We convert the status column into binary values, conduct the test, and sort them in the required orders.

CODE

CORRELATION RESULTS

	Gene	HOXB2_Correlation	p_value
cor13605	HOXB2	1.0000000	0.000000e+00
cor5007	HOXB3	0.6955738	7.672181e-115
cor29231	HOXB.AS1	0.6852758	3.340457e-110
cor15312	HOXB4	0.5347363	2.065520e-59
cor47806	AC036222.1	0.4901554	8.414913e-49
cor35191	HOXB.AS2	0.4605725	1.406403e-42
cor39971	RNU6.863P	0.4522896	6.081057e-41
cor12038	PRR15L	0.3933254	1.612232e-30
cor42633	TRIM51DP	0.3864863	1.936574e-29
cor25017	AC092648.1	0.3826213	7.690841e-29
cor5005	HOXB5	0.3815634	1.118233e-28
cor7274	CALCOCO2	0.3575664	3.789903e-25

HOXB2

POSITIVE CORRELATION

NEGATIVE CORRELATION

	Gene	HOXB2_Correlation	p_value
cor9098	FAM171A1	-0.2069920	4.594922e-09
cor15179	RGMA	-0.2049490	6.548909e-09
cor15619	PRKX	-0.1983764	1.997970e-08
cor730	FOXC1	-0.1967956	2.598209e-08
cor9274	CNKSR2	-0.1939109	4.173111e-08
cor9776	SRSF12	-0.1938799	4.194263e-08
cor28218	SNHG26	-0.1931562	4.718270e-08
cor2496	FERMT1	-0.1902332	7.556586e-08
cor9048	ZNF462	-0.1889767	9.231639e-08

MMP11

	Gene	MMP11_Correlation	p_value
cor2140	MMP11	1.0000000	0.000000e+00
cor3339	AEBP1	0.7566974	4.799297e-147
cor20275	CYS1	0.7526983	1.185813e-144
cor10077	MMP14	0.7448981	4.090054e-140
cor15298	NTM	0.7425146	9.233677e-139
cor5393	COL10A1	0.7357444	5.368370e-135
cor19660	PLPP4	0.7267734	3.488142e-130
cor12628	ANTXR1	0.7217749	1.385867e-127
cor7537	ITGA11	0.7191409	3.079714e-126
cor5315	PLAU	0.7047463	3.836250e-119
cor821	COL11A1	0.7035819	1.377453e-118

	Gene	MMP11_Correlation	p_value
cor60395	AL353135.2	-0.2066402	4.885291e-09
cor4969	BCL11A	-0.1941567	4.009122e-08
cor8826	PM20D2	-0.1923320	5.392412e-08
cor52193	AC006946.2	-0.1907413	6.966206e-08
cor7394	IL33	-0.1810804	3.150676e-07
cor32475	SOX9.AS1	-0.1784314	4.700337e-07
cor3264	MINDY4	-0.1780676	4.963530e-07
cor8094	TAF4B	-0.1774811	5.417902e-07
cor58760	AL356776.2	-0.1771041	5.730895e-07
cor26192	CADM3.AS1	-0.1759890	6.761626e-07
cor5643	SOX9	-0.1715314	1.296235e-06
cor52277	AC027449.1	-0.1695686	1.717372e-06

THANK YOU