BDA_2024_MetaAnalysis_Glioblastoma

Duy Nguyen

2024-06-25

Research Flow of Meta-Analysis of Transcriptomics Profile in Glioblastoma

Step 1: Project set up with necessary packages

Step 2: Search for dataset in Gemma using search terms

Step 3: Extract annotations for every class in the datasets, including:

1. Organism part
2. Cell type
3. Developmental stage
4. Treatment
5. Disease
6. Disease model
7. Genotype
8. Strain
9. Biological sex

(These annotations would be helpful in dataset selection based on inclusion/exclusion criteria regarding these classes)

Step 4: Extract annotations for experimental design in the datasets, including:

1. Factor categories
2. Experimental factors
3. Baseline factors

(These annotations would be helpful in dataset selection as they show which variables were actually manipulated as part of the experiment, especially for datasets in large studies containing multiple experiments)

Step 5: Extract statistical contrasts of differential analysis in the datasets

(These statistical contrasts allow us to check if the differential analyses are:

- Subsetted by brain region
- Included unwanted subjects

- Set up in an appropriate manner with the reference group

Based on the statistical contrasts, we might need to re-run the differential analysis)

Step 6: Assess gene expression range

(Some datasets, especially Agilent microarray datasets, were normalized incorrectly, which may require re-normalization)

1) Project Set Up

```
# Load packages
library("gemma.R")
library("plyr")
library("stringr")
library("metafor")
library("tidyverse")
library("writexl")
library("pheatmap")
library("RColorBrewer")
library("viridis")
library("multtest")
library("rentrez")
library("clusterProfiler")
library("org.Hs.eg.db")
library("ReactomePA")
library("fgsea")
library("pathview")

# Check loaded packages
(.packages())
```

```
##  [1] "pathview"        "fgsea"           "ReactomePA"      "org.Hs.eg.db"
##  [5] "AnnotationDbi"   "IRanges"         "S4Vectors"       "stats4"
##  [9] "clusterProfiler" "rentrez"         "multtest"        "Biobase"
## [13] "BiocGenerics"    "viridis"         "viridisLite"     "RColorBrewer"
## [17] "pheatmap"        "writexl"         "lubridate"       "forcats"
## [21] "dplyr"           "purrr"           "readr"           "tidyr"
## [25] "tibble"          "ggplot2"         "tidyverse"       "metafor"
## [29] "numDeriv"        "metadat"         "Matrix"          "stringr"
## [33] "plyr"            "gemma.R"         "stats"           "graphics"
## [37] "grDevices"       "utils"           "datasets"        "methods"
## [41] "base"
```

2) Gemma Dataset Search

- Datasets in Gemma are annotated using structured terms of ontologies. Many of the ontological terms are hierarchical.
- **Ontologies**: a set of concepts and categories in a subject area or domain that shows their properties and the relations between them.

- Within an ontological hierarchy, the broader, umbrella term is called the *"parent"* while the term representing the more specific subset is called the *"child"*. When requesting Gemma records with particular annotation, a parent term can often pull up all records with the child terms.

```
# Search for ontological annotations of "glioblastoma"
Gbm_onto_annotation <- search_annotations("glioblastom*")

# Count usage of ontological terms and filter terms that are used in Gemma
Gbm_onto_annotation_in_Gemma <- Gbm_onto_annotation %>%
  # Add new column "counts" in which the function is being applied to every
  # single value.URI from gbm_annotation
  mutate(counts = sapply(value.URI, function(uri){
    # This function retrieves "totalElements" attribute from the datasets
    # associated with given input URI
    # The "totalElements" represent the total number of records
    attributes(get_datasets(uris = uri))$totalElements
  })) %>%
  # Filter to keep URI with at least 1 records
  filter(counts > 0)

# Check child terms for "glioblastoma"
get_child_terms("http://purl.obolibrary.org/obo/MONDO_0018177")
```

```
##  [1] "http://purl.obolibrary.org/obo/MONDO_0000457"
##  [2] "http://purl.obolibrary.org/obo/MONDO_0000458"
##  [3] "http://purl.obolibrary.org/obo/MONDO_0000459"
##  [4] "http://purl.obolibrary.org/obo/MONDO_0000460"
##  [5] "http://purl.obolibrary.org/obo/MONDO_0002501"
##  [6] "http://purl.obolibrary.org/obo/MONDO_0004363"
##  [7] "http://purl.obolibrary.org/obo/MONDO_0016681"
##  [8] "http://purl.obolibrary.org/obo/MONDO_0016682"
##  [9] "http://purl.obolibrary.org/obo/MONDO_0018177"
## [10] "http://purl.obolibrary.org/obo/MONDO_0020690"
## [11] "http://purl.obolibrary.org/obo/MONDO_0850335"
```

```
# Search dataset using search terms
Gbm_datasets_searchterm <- get_datasets(query = "glioblastom* OR
                                                 gbm OR gb OR gsc OR btic OR
                                                 \"glioblastoma multiforme\"
                                                 \"grade IV astrocytic tumor\" OR
                                                 \"grade IV astrocytoma\" OR
                                                 \"WHO grade IV glioma\" OR
                                                 \"glioblastoma stem cells\" OR
                                                 \"self-renewing glioblastoma stem cells\" OR
                                                 \"brain tumor initiating cells\" " ,
                                        # Focus on human datasets
                                        taxa = c("human")) %>%
  # Get full set of results
  get_all_pages()

# Check if abbreviations of "gbm" and "gb affect the search
Gbm_datasets_searchterm_noabbre <- get_datasets(query = "glioblastom* OR
                                                 \"glioblastoma multiforme\"
```

```
                                                \"grade IV astrocytic tumor\" OR
                                                \"grade IV astrocytoma\" OR
                                                \"WHO grade IV glioma\" OR
                                                \"glioblastoma stem cells\" OR
                                                \"self-renewing glioblastoma stem cells\" OR
                                                \"brain tumor initiating cells\" ",
                                        # Focus on human datasets
                                        taxa = c("human")) %>%
  # Get full set of results
  get_all_pages()

# Search dataset using parent term on ontological hierarchy
Gbm_datasets_uris <- get_datasets(uris = "http://purl.obolibrary.org/obo/MONDO_0018177",
                                  # Focus on human datasets
                                  taxa = c("human")) %>%
  # Get full set of results
  get_all_pages()


# Check the number of datasets in all dataframes
nrow(Gbm_datasets_searchterm)
```

```
## [1] 352
```

```
nrow(Gbm_datasets_searchterm_noabbre)
```

```
## [1] 339
```

```
nrow(Gbm_datasets_uris)
```

```
## [1] 197
```

```
# Check if the datasets are duplicated in both methods of searching
Gbm_datasets_check <- Gbm_datasets_searchterm %>%
  mutate(available_in_both_dataframes = ifelse(experiment.shortName %in%
                                                 Gbm_datasets_uris$experiment.shortName,
                                                 TRUE, FALSE))

# Check the number of datasets
nrow(Gbm_datasets_check)
```

```
## [1] 352
```

```
# Check the quality of the datasets
# (FALSE - data are of good quality)
table(Gbm_datasets_check$experiment.troubled)
```

```
##
## FALSE
##   352
```

```r
# Check if raw data were available
# (1 - raw data available | -1 - raw data not available)
table(Gbm_datasets_check$experiment.rawData)
```

```
##
##  -1   1
##  79 270
```

```r
# Filter for datasets with good quality and raw data available
Gbm_datasets_filtered <- Gbm_datasets_check %>%
  filter(experiment.troubled == FALSE & experiment.rawData == 1)

# Check the filtered datasets
str(Gbm_datasets_filtered)
```

```
## Classes 'data.table' and 'data.frame':   270 obs. of  24 variables:
##  $ experiment.shortName        : chr  "GSE71224" "GSE140409" "GSE234762" "GSE162614" ...
##  $ experiment.name             : chr  "Inhibition of 13-cis retinoic acid-induced gene expression of
##  $ experiment.ID               : int  9396 29149 32522 35210 3178 5217 5438 5651 5768 5867 ...
##  $ experiment.description      : chr  "The cell differentiation potential of 13-cis retinoic acid (RA
##  $ experiment.troubled         : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ experiment.accession        : chr  "GSE71224" "GSE140409" "GSE234762" "GSE162614" ...
##  $ experiment.database         : chr  "GEO" "GEO" "GEO" "GEO" ...
##  $ experiment.URI              : chr  "https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE71224"
##  $ experiment.sampleCount      : int  12 4 4 4 22 117 180 12 46 12 ...
##  $ experiment.lastUpdated      : POSIXct, format: "2023-12-20 22:15:52" "2023-12-16 16:49:34" ...
##  $ experiment.batchEffectText  : chr  "SINGLE_BATCH_SUCCESS" "SINGLE_BATCH_SUCCESS" "PROBLEMATIC_BAT
##  $ experiment.batchCorrected   : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
##  $ experiment.batchConfound    : num  1 1 0 0 0 -1 -1 1 1 1 ...
##  $ experiment.batchEffect      : num  1 1 0 0 0 0 0 0 1 0 ...
##  $ experiment.rawData          : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ geeq.qScore                 : num  0.569 0.853 0.281 0.283 0.422 ...
##  $ geeq.sScore                 : num  0.875 0.625 0.5 0.5 1 0.625 1 0.75 1 0.875 ...
##  $ taxon.name                  : chr  "human" "human" "human" "human" ...
##  $ taxon.scientific            : chr  "Homo sapiens" "Homo sapiens" "Homo sapiens" "Homo sapiens" ..
##  $ taxon.ID                    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ taxon.NCBI                  : int  9606 9606 9606 9606 9606 9606 9606 9606 9606 9606 ...
##  $ taxon.database.name         : chr  "hg38" "hg38" "hg38" "hg38" ...
##  $ taxon.database.ID           : int  87 87 87 87 87 87 87 87 87 87 ...
##  $ available_in_both_dataframes: logi  TRUE FALSE TRUE FALSE TRUE TRUE ...
##  - attr(*, ".internal.selfref")=<externalptr>
##  - attr(*, "call_origin")= chr "https://gemma.msl.ubc.ca/rest/v2/datasets/?&offset=0&limit=20&sort=%
##  - attr(*, "env_origin")=<environment: 0x1336236f0>
##  - attr(*, "env")=<environment: 0x13361bc78>
```

```r
# Create a function to extract annotations
ExtractAnnotations <- function(dataset_shortname, annotation_class_name){
  # Extract annotations for the dataset experiment.shortNameentified by dataset_shortname
  exp_anno <- get_dataset_annotations(dataset = dataset_shortname)
```

```
  # Check if there are any annotations of the specified annotation_class_name
  if(any(exp_anno$class.name == annotation_class_name)){
    # If yes, return the term names associated with the specified annotation_class_name
    return(exp_anno$term.name[exp_anno$class.name == annotation_class_name])
  } else {
    # If no, return NA
    return(NA)
  }
}
```

```
# Function use

# Extract annotation for organism part
Gbm_annotations_organism_part <- sapply(Gbm_datasets_filtered$experiment.shortName,
                                        ExtractAnnotations,
                                        annotation_class_name = "organism part") %>%
  # Transform into a dataframe
  enframe(name = "experiment.shortName", value = "Organism_part")

# Extract annotation for cell type
Gbm_annotations_cell_type <- sapply(Gbm_datasets_filtered$experiment.shortName,
                                    ExtractAnnotations,
                                    annotation_class_name = "cell type") %>%
  # Transform into a dataframe
  enframe(name = "experiment.shortName", value = "Cell_type")

# Extract annotation for developmental stage
Gbm_annotations_dev_stage <- sapply(Gbm_datasets_filtered$experiment.shortName,
                                    ExtractAnnotations,
                                    annotation_class_name = "developmental stage") %>%
  # Transform into a dataframe
  enframe(name = "experiment.shortName", value = "Developmental_stage")

# Extract annotation for treatment
Gbm_annotations_treatment <- sapply(Gbm_datasets_filtered$experiment.shortName,
                                    ExtractAnnotations,
                                    annotation_class_name = "treatment") %>%
  # Transform into a dataframe
  enframe(name = "experiment.shortName", value = "Treatment")

# Extract annotation for disease
Gbm_annotations_disease <- sapply(Gbm_datasets_filtered$experiment.shortName,
                                  ExtractAnnotations,
                                  annotation_class_name = "disease") %>%
  # Transform into a dataframe
  enframe(name = "experiment.shortName", value = "Disease")

# Extract annotation for disease model
Gbm_annotations_dis_model <- sapply(Gbm_datasets_filtered$experiment.shortName,
                                    ExtractAnnotations,
                                    annotation_class_name = "Disease model") %>%
  # Transform into a dataframe
```

```
  enframe(name = "experiment.shortName", value = "Disease_model")

# Extract annotation for genotype
Gbm_annotations_geno <- sapply(Gbm_datasets_filtered$experiment.shortName,
                               ExtractAnnotations,
                               annotation_class_name = "genotype") %>%
  # Transform into a dataframe
  enframe(name = "experiment.shortName", value = "Genotype")

# Extract annotation for strain
Gbm_annotations_strain <- sapply(Gbm_datasets_filtered$experiment.shortName,
                                 ExtractAnnotations,
                                 annotation_class_name = "strain") %>%
  # Transform into a dataframe
  enframe(name = "experiment.shortName", value = "Strain")

# Extract annotation for biological sex
Gbm_annotations_sex <- sapply(Gbm_datasets_filtered$experiment.shortName,
                              ExtractAnnotations,
                              annotation_class_name = "biological sex") %>%
  # Transform into a dataframe
  enframe(name = "experiment.shortName", value = "Biological_sex")
```

3) Extraction of Annotations for Each Class in Each Dataset

4) Extraction of Factors in Experimental Design of Each Dataset Sometimes it is difficult to determine from the abstract and dataset annotation which variables were *actually manipulated* as part of the experiment.

This can be especially true for datasets that were part of much larger studies containing multiple experiments.

Gemma's information about the experimental design for the transcriptional profiling experiment can clarify this.

```
# Create a function to extract factors in experimental design
ExtractFactors <- function(dataset_shortname){
  # Create a dataframe to store the results
  factor_df <- data.frame(
    experiment.shortName = dataset_shortname,
    factor_info = vector(mode = "character",
                         length = length(dataset_shortname)),
    treatment_factor_info = vector(mode = "character",
                                   length = length(dataset_shortname)),
    baseline_factor_info = vector(mode = "character",
                                  length = length(dataset_shortname)),
    stringsAsFactors = FALSE
  )

  # Loop through each of the dataset
  for (i in c(1:length(dataset_shortname))){

    # Extract the differential expression analysis from the datasets
    design <- get_dataset_differential_expression_analyses(dataset_shortname[i])
```

```
    if(nrow(design) > 0){
    # Concatenate the factor categories into a single string
    factor_df$factor_info[i] <- paste(design$factor.category, collapse = ";")

    # Concatenate the experimental factors into a single string
    treatment_factor <- sapply(design$experimental.factors, function(f) f$summary)
    factor_df$treatment_factor_info[i] <- paste(treatment_factor, collapse = ";")

    # Concatenate the baseline factors into a single string
    baseline_factor <- sapply(design$baseline.factors, function(f) f$summary)
    factor_df$baseline_factor_info[i] <- paste(baseline_factor, collapse = ";")
    }
  }
  # Return the final dataframe
  return(factor_df)
}
```

```
# Function use
Gbm_annotations_factor <- ExtractFactors(Gbm_datasets_filtered$experiment.shortName)
```

```
# Combine all annotations by experiment.shortName and create extra columns in
# the final dataframe
Gbm_annotations_all <- Gbm_datasets_filtered %>%
  left_join(Gbm_annotations_organism_part, by = "experiment.shortName") %>%
  left_join(Gbm_annotations_cell_type, by = "experiment.shortName") %>%
  left_join(Gbm_annotations_dev_stage, by = "experiment.shortName") %>%
  left_join(Gbm_annotations_treatment, by = "experiment.shortName") %>%
  left_join(Gbm_annotations_disease, by = "experiment.shortName") %>%
  left_join(Gbm_annotations_dis_model, by = "experiment.shortName") %>%
  left_join(Gbm_annotations_geno, by = "experiment.shortName") %>%
  left_join(Gbm_annotations_strain, by = "experiment.shortName") %>%
  left_join(Gbm_annotations_sex, by = "experiment.shortName") %>%
  left_join(Gbm_annotations_factor, by = "experiment.shortName") %>%

  # Add empty columns to store screening notes
  mutate(Manipulation_unrelated_to_topic =
           vector(mode = "character", length = nrow(.)),
         Incorrect_developmental_stage =
           vector(mode = "character", length = nrow(.)),
         Not_bulk_dissection_Particular_Cell_type_or_Subregion =
           vector(mode = "character", length = nrow(.)),
         Not_full_transcriptome =
           vector(mode = "character", length = nrow(.)),
         Metadata_issues_Missing_info_NoPub_Retracted_Duplicated =
           vector(mode = "character", length = nrow(.)),
         Excluded =
           vector(mode = "character", length = nrow(.)),
         Excluded_reason =
           vector(mode = "character", length = nrow(.)),
         Included =
           vector(mode = "character", length = nrow(.)))

# Transform the annotations into characters
```

```
Gbm_annotations_all$Organism_part <- as.character(Gbm_annotations_all$Organism_part)
Gbm_annotations_all$Cell_type <- as.character(Gbm_annotations_all$Cell_type)
Gbm_annotations_all$Developmental_stage <- as.character(Gbm_annotations_all$Developmental_stage)
Gbm_annotations_all$Treatment <- as.character(Gbm_annotations_all$Treatment)
Gbm_annotations_all$Disease <- as.character(Gbm_annotations_all$Disease)
Gbm_annotations_all$Disease_model <- as.character(Gbm_annotations_all$Disease_model)
Gbm_annotations_all$Genotype <- as.character(Gbm_annotations_all$Genotype)
Gbm_annotations_all$Strain <- as.character(Gbm_annotations_all$Strain)
Gbm_annotations_all$Biological_sex <- as.character(Gbm_annotations_all$Biological_sex)

# Extract the final dataframe into CSV
write_csv(Gbm_annotations_all, "All_datasets_with_annotations.csv")
# str(gbm_annotations_all)
```

Through initial selection process, 6 datasets are possible, including: GSE154958, GSE15209, GSE45899, GSE135306, GSE4536, GSE75147

5) Gene Expression Range Assessment

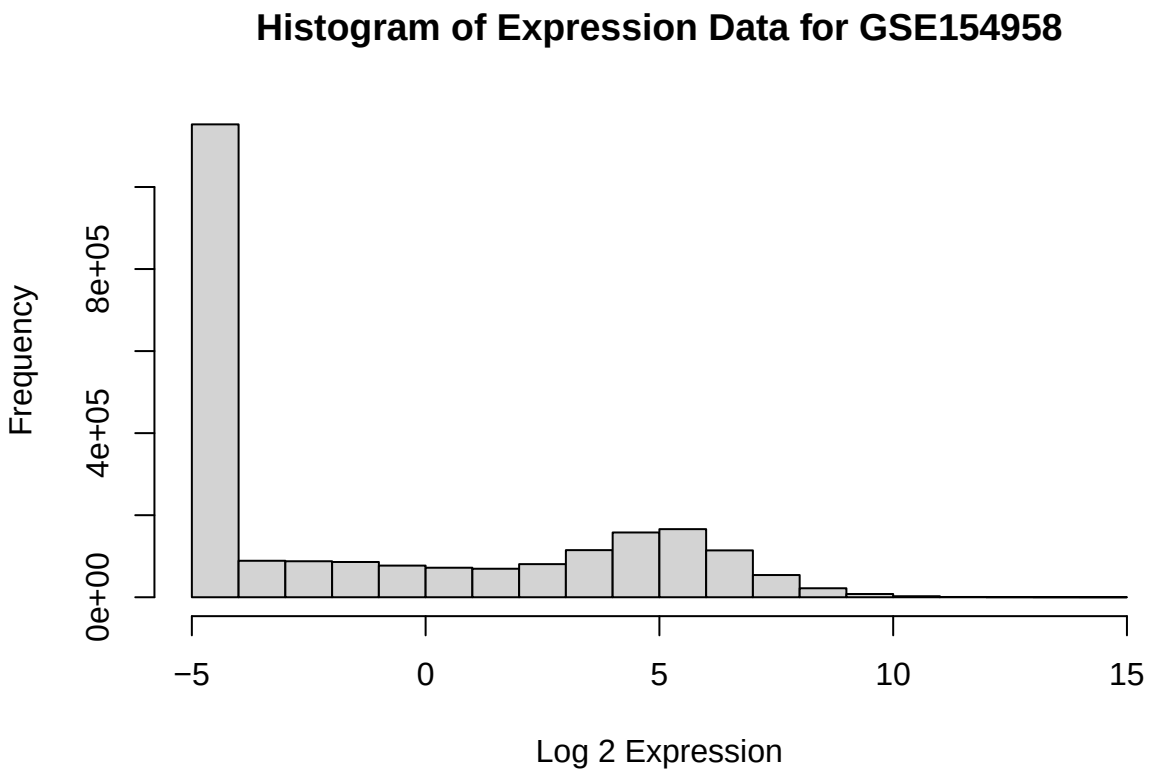
```
# Create a function to check the expression range of the dataset
# (especially for the Agilent microarray datasets)
CheckGeneExpressionRange <- function(dataset_shortname){

  # Retrieve the processed expression data for the given dataset
  expression_data <- get_dataset_processed_expression(dataset_shortname)

  # Print the structure of the expression data
  print(str(expression_data))

  # The first four columns are row metadata: Probe, GeneSymbol, GeneName, NCBI ID
  # The rest of the columns are gene expression values for each subject

  # Exclude metadata row (row 1-4) and convert the gene expression columns to a
  # matrix for further analysis
  expression_matrix <- as.matrix(expression_data[,-1:-4])

  # Create a histogram of the expression data
  hist(expression_matrix,
       main = paste("Histogram of Expression Data for", dataset_shortname),
       # The y-axis is the gene frequency
       # The x-axis is log 2 gene expression - log 2 counts per million
       xlab = "Log 2 Expression", ylab = "Frequency")

       # The large spike on the left side of the histogram ("floor effect") are
       # all of the genes that are not truly expressed or have too low of expression
       # to be measurable

  # Log 2 RNA-seq dataset has the range between -5 to 12
  # Log 2 Microarray dataset has the range between 4 to 15

  # Calculate the min, median, max values of the expression data
```

```
  min_val <- min(expression_matrix, na.rm = TRUE)
  median_val <- median(expression_matrix, na.rm = TRUE)
  max_val <- max(expression_matrix, na.rm = TRUE)

  # Print the calculated values
  print(paste("Minimum value:", min_val))
  print(paste("Median value:", median_val))
  print(paste("Maximum value:", max_val))
}
```

```
# Function use
CheckGeneExpressionRange("GSE154958") # Min: -4.9 | Max: 14.1 | RNA-seq
```

```
## Classes 'data.table' and 'data.frame':   38609 obs. of  68 variables:
##  $ Probe                              : int  1 10 100 1000 10000 100008586 100008587 100008588 10000
##  $ GeneSymbol                         : chr  "A1BG" "NAT2" "ADA" "CDH2" ...
##  $ GeneName                           : chr  "alpha-1-B glycoprotein" "N-acetyltransferase 2" "adenos
##  $ NCBIid                             : int  1 10 100 1000 10000 100008586 100008587 100008588 10000
##  $ DURA061_IPSC_N4_P5 [262136]        : num  0.12 -1.63 2.33 6.56 4.05 ...
##  $ DURA061_NSC_N4_P2 [241186]         : num  0.758 -3.585 2.63 8.759 5.436 ...
##  $ GBM050_P7n8 [227164]               : num  2.1 -3.3 4.75 9.72 5.16 ...
##  $ GBM031_P4 [205149]                 : num  2.31 -1.996 0.934 9.807 6.435 ...
##  $ GBM018_P10 [201101]                : num  -4.92 -3.11 3.28 9.72 5.55 ...
##  $ DURA050_IPSC_N12_P5 [264112]       : num  -0.0479 -2.2688 2.5607 6.8505 4.1232 ...
##  $ GBM017_P3 [242174]                 : num  2.52 -2.05 2.9 9.98 5.68 ...
##  $ DURA019_NSC_N5C1_P2 [267171]       : num  -0.724 -2.477 3.086 8.18 5.386 ...
##  $ GBM017_P4 [243162]                 : num  2.75 -3.23 3.67 9.82 5.86 ...
##  $ GBM030_P5 [223117]                 : num  2.51 -4.92 3.21 9.3 6.05 ...
##  $ GBM050_P9 [228152]                 : num  1.88 -1.71 4.73 9.83 5.4 ...
##  $ GBM054_P6 [230128]                 : num  2.19 -2.21 3.14 9.3 5.93 ...
##  $ DURA054_FB_P5 [252149]             : num  2.38 -4.91 2.59 5.96 6.66 ...
##  $ GBM052_P4n5 [234175]               : num  -2.21 -2.21 4.64 9.88 5.35 ...
##  $ GBM061_P5 [232104]                 : num  2.69 -3.21 4.59 9.87 6.06 ...
##  $ DURA030_FB_P8 [277146]             : num  2.22 -4.91 3.49 4.95 6.65 ...
##  $ DURA031_FB_P7 [276158]             : num  0.966 -4.914 3.42 6.266 6.676 ...
##  $ DURA019_FB_P7 [275170]             : num  2.85 -0.775 4.468 4.041 6.587 ...
##  $ DURA026_FB_P8 [246126]             : num  2.38 -4.92 4.02 4.46 6.62 ...
##  $ GBM031_P7 [238127]                 : num  2.96 -2.07 0.486 9.949 6.497 ...
##  $ DURA031_NSC_N44_P3 [265195]        : num  0.99 -4 2.82 8.51 5.55 ...
##  $ DURA061_FB_P7 [253137]             : num  2.63 -4.92 3.91 7.16 6.65 ...
##  $ DURA017_FB_P7 [255113]             : num  2.95 -1.88 2.82 6.13 6.4 ...
##  $ NH16_677_SP1A [259165]             : num  0.363 -4.726 -2.289 6.486 7.782 ...
##  $ DURA054_IPSC_N3C_P11 [261148]      : num  0.882 -3.319 2.929 8.779 5.24 ...
##  $ DURA052_FB_P6 [254125]             : num  2.94 -4.92 2.77 6.73 6.73 ...
##  $ GBM018_P12 [217189]                : num  -2.81 -4.92 3.07 9.58 6 ...
##  $ DURA030_NSC_N16B6_P1 [224105]      : num  1.44 -1.13 3.62 8.31 5.57 ...
##  $ DURA031_IPSC_N44B_P10 [247114]     : num  1.17 -2.07 2.53 6.81 3.56 ...
##  $ NH16_616DEF1B [249112]             : num  4.05 -4.92 -2.75 6.27 6.28 ...
##  $ NH16_2806DEF3A1 [255184]           : num  -1.03 -4.92 -4.58 7.47 7.4 ...
##  $ NH16_2214_DEF1A [256196]           : logi  NA NA NA NA NA NA ...
##  $ NH16_2255DEF1B2 [254172]           : num  -0.135 -4.529 -0.647 7.065 6.833 ...
##  $ DURA030_IPSC_N16B6_P13 [248102]    : num  1.614 -0.467 2.702 6.146 3.679 ...
##  $ DURA017_NSC_N3C5_P4 [244150]       : num  1.32 -4.92 3.69 9.33 5.86 ...
```

```
##  $ DURA018_NSC_N2_P6 [211126]         : num  2.298 -0.581 3.593 7.197 4.879 ...
##  $ DURA061_NSC_N1_P3 [257196]         : num  0.0928 -3.2836 3.1608 8.4635 5.8019 ...
##  $ DURA031_NSC_N44B_P2 [215174]       : num  0.65 -3.61 2.81 8.19 5.48 ...
##  $ GBM030_P9n10 [237139]              : num  2.92 -4.92 2.68 9.28 5.12 ...
##  $ NH16_2063_DEF1Areplacement [251136]: logi  NA NA NA NA NA NA ...
##  $ DURA019_NSC_N8C_P2 [212138]        : num  1.32 -3.11 3.6 8.76 6.07 ...
##  $ DURA019_IPSC_N8_P13 [245138]       : num  0.129 -1.334 3.565 4.88 3.466 ...
##  $ DURA030_NSC_N9_P2 [266183]         : num  2.675 0.815 3.974 5.848 3.995 ...
##  $ DURA054_NSC_N2E_P1 [240103]        : num  0.938 -2.302 3.046 8.781 5.546 ...
##  $ GBM054_P4 [229140]                 : num  1.55 -3.06 3.25 8.97 5.46 ...
##  $ GBM019_P3n6 [239115]               : num  2.96 -4.92 4.23 8.68 4.74 ...
##  $ NH16_270_DEF1Ereplacement [249112] : logi  NA NA NA NA NA NA ...
##  $ NH15_2101_DEF1A [231121]           : num  2.51 -4.83 -2.61 6.53 6.19 ...
##  $ DURA026_NSC_N31D_P5 [214162]       : num  1.18 -1.84 3.62 8.52 5.83 ...
##  $ DURA018_NSC_N4_P4 [222129]         : num  1.875 -0.636 3.726 7.414 5.031 ...
##  $ GBM052_P6n7 [233187]               : num  -4.92 -4.92 4.63 10.04 4.93 ...
##  $ DURA054_NSC_N3C_P2 [239115]        : num  1.21 -3.31 3.31 8.76 5.31 ...
##  $ DURA052_NSC_N4_P3 [252149]         : num  2.62 -4.89 5.35 8.55 4.92 ...
##  $ DURA050_NSC_N12_P3 [237139]        : num  0.892 -2.836 3.234 8.483 5.585 ...
##  $ DURA052_NSC_N5_P2 [244150]         : num  0.478 -4.918 2.498 9.081 5.992 ...
##  $ DURA018_FB_P6 [244150]             : num  0.823 -4.915 2.815 5.17 6.695 ...
##  $ GBM061_P3 [231116]                 : num  2.55 -3.21 3.95 9.62 5.93 ...
##  $ NH15_1877_SP1C [211171]            : num  -0.803 -4.676 -2.964 6.554 6.932 ...
##  $ GBM019_P4 [202113]                 : num  2.63 -4.92 3.66 8.53 4.83 ...
##  $ GBM026_P3n4 [216106]               : num  -3.02 -4.92 3.62 9.19 5.17 ...
##  $ DURA050_FB_P7 [251161]             : num  2.77 -4.89 3.19 6.59 7.16 ...
##  $ NH15_1661DEF2C [250124]            : num  3.71 -4.92 -3.82 6.38 5.7 ...
##  $ GBM026_P8 [204137]                 : num  -3.33 -4.92 4.21 9.18 4.71 ...
##  $ DURA050_NSC_N16_P4 [238127]        : num  0.568 -2.611 3.623 8.642 6.112 ...
##  - attr(*, ".internal.selfref")=<externalptr>
##  - attr(*, "call")= chr "https://gemma.msl.ubc.ca/rest/v2/datasets/GSE154958/data/processed"
##  - attr(*, "env")=<environment: 0x1113c23e0>
## NULL
```

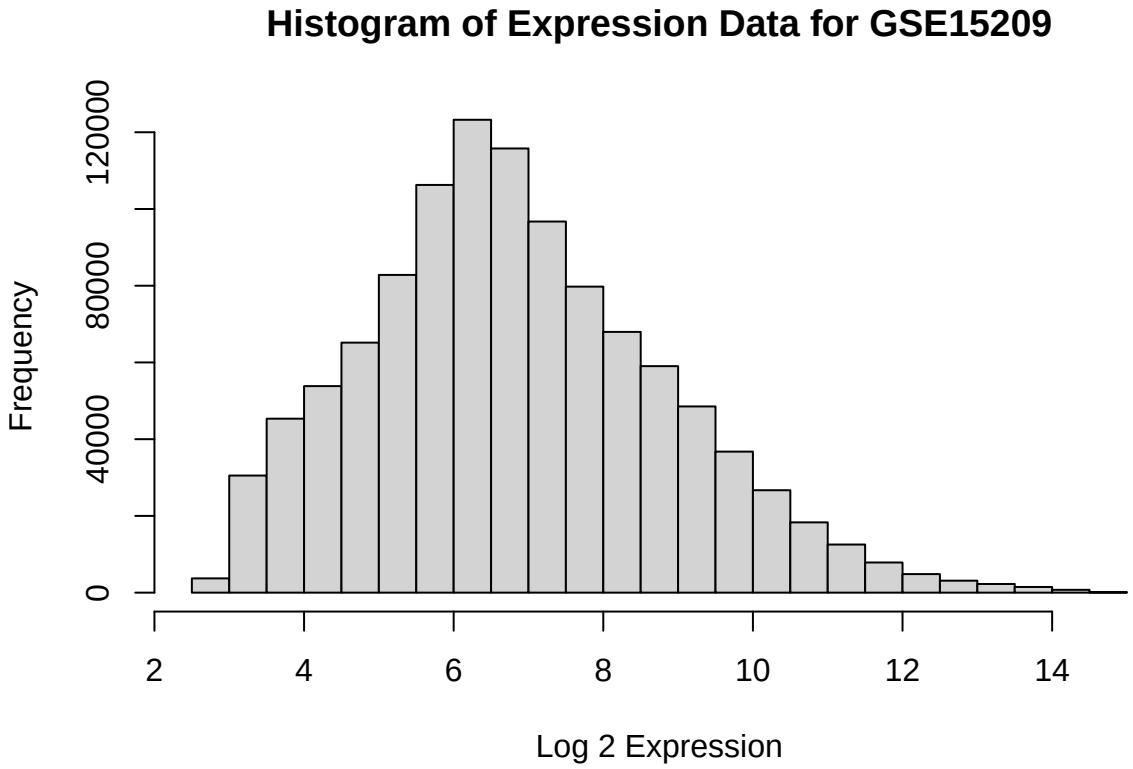



```
## [1] "Minimum value: -4.924"
## [1] "Median value: -3.5236"
## [1] "Maximum value: 14.1592"
```

```
CheckGeneExpressionRange("GSE15209") # Min: 2.7 | Max: 14.8 | Affymetrix Array
```

```
## Classes 'data.table' and 'data.frame':   54675 obs. of  24 variables:
##  $ Probe           : chr  "1007_s_at" "1053_at" "117_at" "121_at" ...
##  $ GeneSymbol      : chr  "DDR1" "RFC2" "HSPA7|HSPA6" "PAX8" ...
##  $ GeneName        : chr  "discoidin domain receptor tyrosine kinase 1" "replication factor C subuni
##  $ NCBIid          : chr  "780" "5982" "3311|3310" "7849" ...
##  $ G166-NS_A       : num  10.19 8.26 6.89 9.84 4.53 ...
##  $ G144-NS_B       : num  11.41 9.69 7.14 9.47 4.34 ...
##  $ G144-NS_A       : num  11.63 9.3 6.94 9.59 4.41 ...
##  $ GliNS2          : num  13.01 8.42 7.03 9.45 4.37 ...
##  $ G144-ED (GliNS1): num  12.18 8.13 7.43 9.49 4.42 ...
##  $ hf289-NS        : num  12.57 8.3 7.04 9.65 6.1 ...
##  $ hf286-NS        : num  12.97 7.97 7.12 9.84 4.6 ...
##  $ hf240-NS-C      : num  12.23 8.89 6.54 9.44 4.43 ...
##  $ hf240-NS-B      : num  12.33 8.87 6.57 9.56 4.35 ...
##  $ hf240-NS-A      : num  12.29 8.54 6.83 9.52 4.18 ...
##  $ Norm Brain#102  : num  11.15 6.27 6.51 9.93 5.1 ...
##  $ NORM#41         : num  11.02 6.76 6.93 9.58 5.37 ...
##  $ Norm Brain#36   : num  10.51 7.05 6.64 9.52 5.07 ...
##  $ NORM#31         : num  11.88 7.29 7.25 9.74 4.61 ...
```

```
##  $ NORM#12         : num  11.31 6.87 6.88 9.86 6.4 ...
##  $ Normal Brain #4 : num  10.62 6.37 6.91 9.92 6.29 ...
##  $ G179-NS_B       : num  11.54 9.4 6.75 9.43 4.1 ...
##  $ G179-NS_A       : num  11.26 8.72 6.96 9.46 4.38 ...
##  $ G174-NS         : num  11.9 8.74 6.7 9.55 4.4 ...
##  $ G166-NS_B       : num  9.99 9.35 7.06 9.62 4.24 ...
##  - attr(*, ".internal.selfref")=<externalptr>
##  - attr(*, "call")= chr "https://gemma.msl.ubc.ca/rest/v2/datasets/GSE15209/data/processed"
##  - attr(*, "env")=<environment: 0x167884d38>
## NULL
```

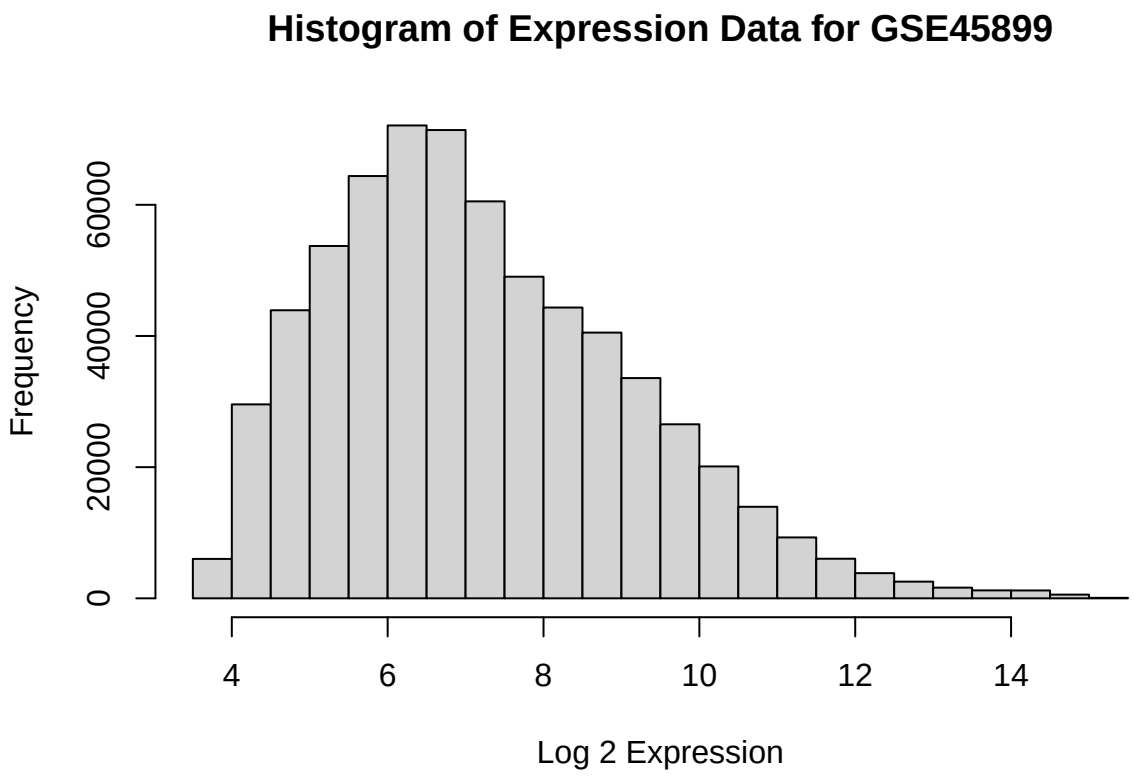



```
## [1] "Minimum value: 2.6704"
## [1] "Median value: 6.6478"
## [1] "Maximum value: 14.7956"
```

```
CheckGeneExpressionRange("GSE45899") # Min: 3.6 | Max: 15.2 | Affymetrix Array
```

```
## Classes 'data.table' and 'data.frame':   54675 obs. of  16 variables:
##  $ Probe     : chr  "1007_s_at" "1053_at" "117_at" "121_at" ...
##  $ GeneSymbol: chr  "DDR1" "RFC2" "HSPA7|HSPA6" "PAX8" ...
##  $ GeneName  : chr  "discoidin domain receptor tyrosine kinase 1" "replication factor C subunit 2" "
##  $ NCBIid    : chr  "780" "5982" "3311|3310" "7849" ...
##  $ NS_rep2   : num  12.44 10 7.46 9.93 5.71 ...
##  $ NS_rep1   : num  12.39 9.83 7.94 10.08 5.32 ...
##  $ MGG8_rep4 : num  10.18 10.77 7.63 7.96 7.02 ...
```

```
##  $ MGG8_rep3 : num  10.11 10.75 7.7 7.91 6.51 ...
##  $ MGG8_rep2 : num  10.91 9.7 8.25 10.35 4.8 ...
##  $ MGG8_rep1 : num  10.85 10.11 8.34 10.24 5.05 ...
##  $ MGG6_rep2 : num  12.35 9.61 7.76 10.18 4.72 ...
##  $ MGG6_rep1 : num  12.45 9.6 7.83 9.97 4.58 ...
##  $ MGG4_rep2 : num  10.04 9.97 6.61 8.14 4.79 ...
##  $ MGG4_rep1 : num  10.03 9.92 6.33 8.04 4.66 ...
##  $ MGG23_rep2: num  11.6 9.68 6.88 7.91 4.71 ...
##  $ MGG23_rep1: num  11.43 9.98 6.53 7.96 4.84 ...
##  - attr(*, ".internal.selfref")=<externalptr>
##  - attr(*, "call")= chr "https://gemma.msl.ubc.ca/rest/v2/datasets/GSE45899/data/processed"
##  - attr(*, "env")=<environment: 0x1114f79c8>
## NULL
```

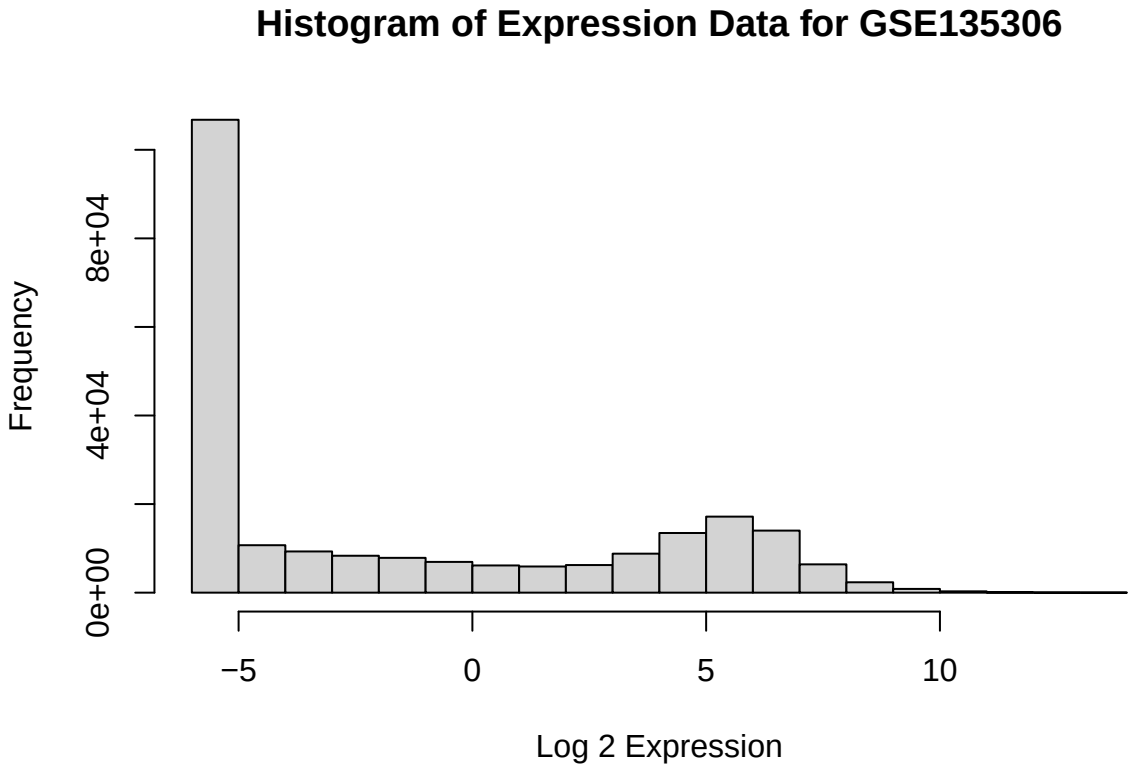



```
## [1] "Minimum value: 3.5717"
## [1] "Median value: 6.9036"
## [1] "Maximum value: 15.1961"
```

```
CheckGeneExpressionRange("GSE135306") # Min: -5.9 | Max: 13.5 | RNA-Seq
```

```
## Classes 'data.table' and 'data.frame':   38609 obs. of  10 variables:
##  $ Probe        : int  1 10 100 1000 10000 100008586 100008587 100008588 100008589 100009606 ...
##  $ GeneSymbol   : chr  "A1BG" "NAT2" "ADA" "CDH2" ...
##  $ GeneName     : chr  "alpha-1-B glycoprotein" "N-acetyltransferase 2" "adenosine deaminase" "cadhe
##  $ NCBIid       : int  1 10 100 1000 10000 100008586 100008587 100008588 100008589 100009606 ...
```

```
##  $ 3565_shEYA2-2: num  2.8 -5.9 5.35 8.97 5.88 ...
##  $ 3691_shEYA2-2: num  3.76 -2.37 5.34 9.17 5.61 ...
##  $ 3565_shNT    : num  3.19 -5.9 6.66 8.64 5.39 ...
##  $ 3565_shEYA2-1: num  3.11 -4.31 4.77 9.08 4.25 ...
##  $ 3691_shNT    : num  3.4 -2.37 6.8 8.78 5.05 ...
##  $ 3691_shEYA2-1: num  3.94 -1.33 5.74 9.12 4.16 ...
##  - attr(*, ".internal.selfref")=<externalptr>
##  - attr(*, "call")= chr "https://gemma.msl.ubc.ca/rest/v2/datasets/GSE135306/data/processed"
##  - attr(*, "env")=<environment: 0x113345d68>
## NULL
```



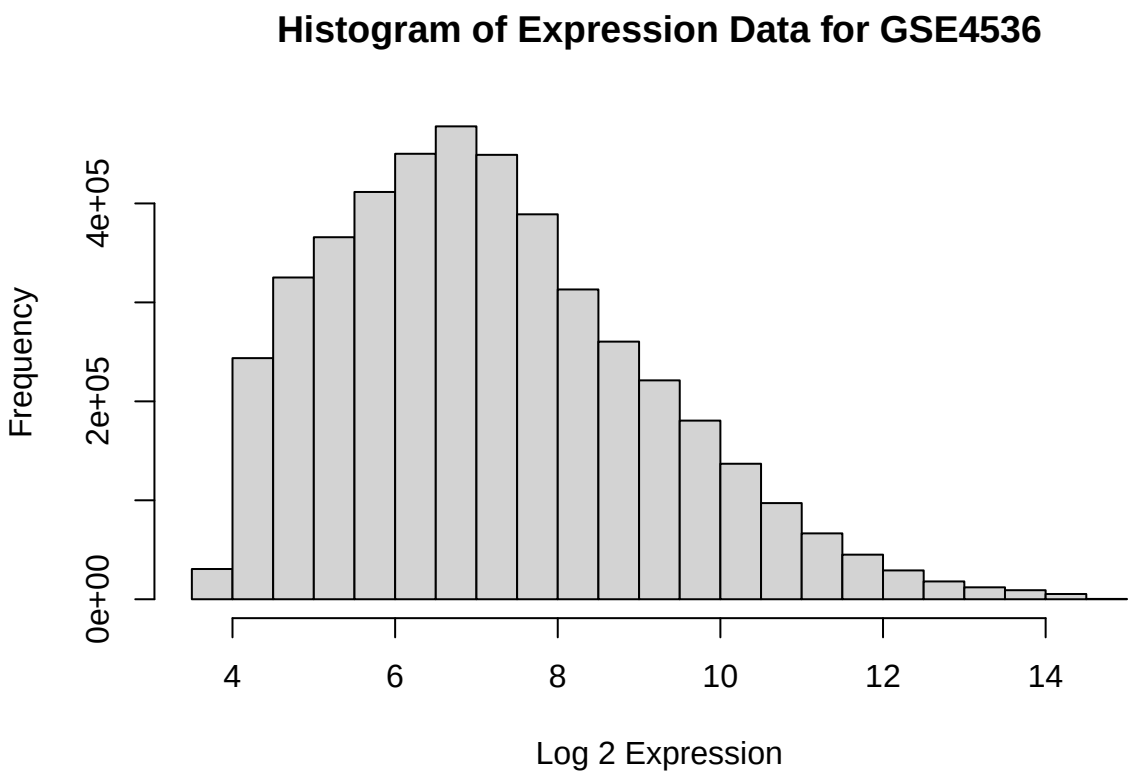

```
## [1] "Minimum value: -5.8986"
## [1] "Median value: -4.3137"
## [1] "Maximum value: 13.5267"
```

```
CheckGeneExpressionRange("GSE4536") # Min: 3.7 | Max: 14.6 | Affymetrix Array
```

```
## Classes 'data.table' and 'data.frame':   54675 obs. of  87 variables:
##  $ Probe              : chr  "1007_s_at" "1053_at" "117_at" "121_at" ...
##  $ GeneSymbol         : chr  "DDR1" "RFC2" "HSPA7|HSPA6" "PAX8" ...
##  $ GeneName           : chr  "discoidin domain receptor tyrosine kinase 1" "replication factor C subu
##  $ NCBIid             : chr  "780" "5982" "3311|3310" "7849" ...
##  $ HF0445_U133P2      : num  12.35 9.24 8.05 9.84 4.71 ...
##  $ U87_2              : num  9.95 9.53 7.39 9.95 4.68 ...
##  $ U87_1              : num  9.98 9.49 7.5 10.06 4.62 ...
```

```
##  $ U87_SC_2           : num  10.89 9.91 7.41 9.99 4.8 ...
##  $ U251_IC_2          : num  12.24 9.39 7.24 10.18 4.94 ...
##  $ U387_2             : num  10.89 10.66 7.37 9.56 4.58 ...
##  $ U387_1             : num  10.83 10.74 7.4 9.57 4.69 ...
##  $ U373_2             : num  11.34 10.27 7.28 9.6 4.68 ...
##  $ U373_1             : num  11.28 10.29 7.27 9.74 4.75 ...
##  $ U251_2             : num  10.7 8.02 8.31 9.88 4.63 ...
##  $ U251_1             : num  10.81 7.94 8.48 9.91 4.67 ...
##  $ U251_SC_2          : num  12.59 9.1 8.13 10.05 4.73 ...
##  $ U251_SC_1          : num  12.47 9.26 7.29 10 4.83 ...
##  $ U87_SC_1           : num  10.8 9.87 7.75 9.78 4.64 ...
##  $ U251_IC_1          : num  12.26 9.29 7.23 9.87 4.74 ...
##  $ U138_2             : num  10.79 9.83 7.42 10.08 4.69 ...
##  $ U138_1             : num  10.77 9.74 7.48 10.21 4.54 ...
##  $ U118_2             : num  9.91 9.64 7.37 10.1 4.82 ...
##  $ U118_1             : num  9.77 9.61 7.47 10.17 4.8 ...
##  $ T98_2              : num  10.73 10.15 7.33 9.72 4.83 ...
##  $ T98_1              : num  10.68 10.05 7.4 9.8 4.61 ...
##  $ SW1088_2           : num  9.97 10.16 7.53 9.63 4.77 ...
##  $ SW1088_1           : num  10.18 10.2 7.54 9.82 4.56 ...
##  $ NOB0308_S_p25_IC_2 : num  12.01 9.29 7.35 9.9 4.73 ...
##  $ NOB0308_S_p25_IC_1 : num  12.17 9.09 7.22 9.88 4.53 ...
##  $ NOB0308_S_p20_2    : num  10.5 9.9 7.14 9.79 4.83 ...
##  $ NOB0308_S_p20_1    : num  11.03 9.79 7.18 9.74 4.77 ...
##  $ NOB0308_NBE_p20_2  : num  12.88 9.37 7.83 9.84 4.86 ...
##  $ NOB0308_NBE_p20_1  : num  12.92 9.43 7.79 10.09 4.87 ...
##  $ NOB0308_NBE_p35_IC : num  12.81 9.72 7.72 9.68 4.63 ...
##  $ NOB0308_NBE_ssc_IC : num  12.8 9.46 7.54 9.85 4.53 ...
##  $ NOB0308_NBE_p10_IC : num  13.03 9.63 7.71 9.98 4.7 ...
##  $ NOB0308_NBE_p1_IC  : num  12.4 9.92 7.42 9.74 4.69 ...
##  $ NOB1228_S_p3_2     : num  12.48 9.1 7.87 9.65 4.66 ...
##  $ NOB1228_S_p3_1     : num  12.86 9 7.85 9.75 4.57 ...
##  $ NOB1228_NBE_p3_2   : num  12.98 9.65 7.48 9.95 4.91 ...
##  $ NOB1228_NBE_p3_1   : num  12.92 9.85 7.39 9.32 4.8 ...
##  $ NOB1228            : num  13.11 9.07 7.73 9.41 4.63 ...
##  $ NOB0308_2          : num  12.77 9.55 8.53 9.63 4.65 ...
##  $ NOB0308_1          : num  12.76 9.47 8.59 9.62 4.66 ...
##  $ NSC_p13_2          : num  12.95 9.4 7.25 9.62 4.67 ...
##  $ NSC_p13_1          : num  12.84 9.49 7.24 9.4 4.75 ...
##  $ NOB1228_S_p8_4     : num  11.96 9.45 7.28 9.54 4.48 ...
##  $ NOB1228_S_P8_3     : num  10.92 9.21 7.18 9.5 4.58 ...
##  $ NOB1228_S_p3_4     : num  12.56 9.07 7.79 9.68 4.6 ...
##  $ NOB1228_S_p3_3     : num  12.79 8.91 7.99 9.66 4.61 ...
##  $ NOB1228_S_p13_3    : num  11.19 9.31 7.28 9.79 4.61 ...
##  $ NOB1228_S_P13_2    : num  11.06 9.3 7.46 9.71 4.63 ...
##  $ NOB1228_NBE_P8_3   : num  13.08 9.51 7.26 9.3 4.78 ...
##  $ NOB1228_NBE_P8_2   : num  13.03 9.4 7.34 9.36 4.63 ...
##  $ NOB1228_NBE_P8_1   : num  13.22 9.36 7.4 9.52 4.63 ...
##  $ NOB1228_NBE_P3_4   : num  12.94 9.91 7.42 9.27 4.79 ...
##  $ NOB1228_NBE_P3_3   : num  12.87 9.87 7.52 9.32 4.81 ...
##  $ NOB1228_NBE_P13_3  : num  13.1 9.46 7.38 9.46 4.68 ...
##  $ NOB1228_NBE_P13_2  : num  13.1 9.66 7.3 9.28 4.74 ...
##  $ NOB1228_NBE_P13_1  : num  13.05 9.46 7.19 9.3 4.7 ...
##  $ NOB0308_S_P35_3    : num  9.23 10.31 7.64 9.86 4.8 ...
```

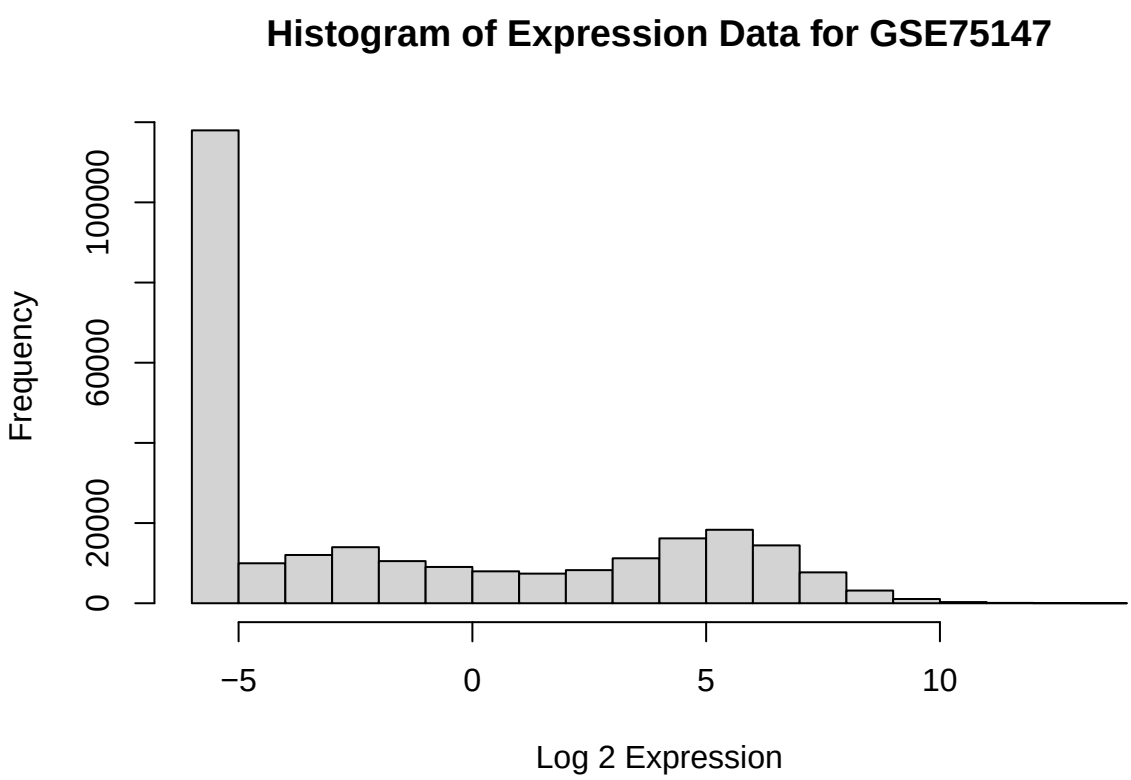
```
##  $ NOB0308_S_P35_2    : num  9.4 10.41 7.45 9.89 4.77 ...
##  $ NOB0308_S_P35_1    : num  9.53 10.26 7.76 9.91 4.82 ...
##  $ NOB0308_S_P20_4    : num  11.22 9.9 7.32 9.88 4.69 ...
##  $ NOB0308_S_P20_3    : num  11.17 9.87 7.18 9.67 4.71 ...
##  $ NOB1228_S_P13_1    : num  11.16 9.34 7.23 9.72 4.62 ...
##  $ NOB0308_NBE_P5_4   : num  13.05 9.73 7.5 9.74 4.73 ...
##  $ NOB0308_NBE_P5_3   : num  13.13 9.62 7.48 9.77 4.81 ...
##  $ NOB0308_NBE_P35_2  : num  12.72 9.81 7.87 9.72 4.63 ...
##  $ NOB0308_NBE_P35_1  : num  12.69 9.8 7.89 9.78 4.79 ...
##  $ NOB0308_NBE_P20_4  : num  13 9.46 7.78 9.55 4.75 ...
##  $ NOB0308_NBE_P20_3  : num  12.89 9.44 7.85 9.64 4.9 ...
##  $ NSC_p19            : num  12.93 9.33 7.4 9.53 4.78 ...
##  $ NOB0308_S_p10_2    : num  12.53 8.83 7.4 9.89 4.65 ...
##  $ NOB0308_S_p10_1    : num  12.43 8.92 7.47 9.94 4.74 ...
##  $ NOB0308_NBE_p5_2   : num  13.11 9.79 7.46 9.68 4.7 ...
##  $ NOB0308_NBE_p5_1   : num  13.05 9.67 7.63 9.81 4.81 ...
##  $ LN229_2            : num  10.78 10.09 7.45 9.87 4.73 ...
##  $ LN229_1            : num  10.79 10.02 7.51 9.96 4.64 ...
##  $ NOB1228_NBE_p1_IC_4: num  13.49 9.78 7.53 9.78 4.77 ...
##  $ NOB1228_NBE_p1_IC_3: num  13.24 9.61 7.74 9.83 4.72 ...
##  $ NOB1228_NBE_p1_IC_2: num  13.26 9.64 7.51 9.85 4.75 ...
##  $ NOB1228_NBE_p1_IC_1: num  13.16 9.68 7.61 9.55 4.68 ...
##  $ A172_2             : num  11.03 9.23 7.33 9.63 4.64 ...
##  $ A172_1             : num  11.03 9.02 7.62 9.88 4.56 ...
##  $ NOB1228_S_p8_2     : num  11.91 9.21 7.43 9.62 4.58 ...
##  $ NOB1228_S_p8_1     : num  10.85 9.25 7.08 9.48 4.6 ...
##  - attr(*, ".internal.selfref")=<externalptr>
##  - attr(*, "call")= chr "https://gemma.msl.ubc.ca/rest/v2/datasets/GSE4536/data/processed"
##  - attr(*, "env")=<environment: 0x29079c8a0>
## NULL
```




```
## [1] "Minimum value: 3.6876"
## [1] "Median value: 6.9621"
## [1] "Maximum value: 14.5943"
```

```
CheckGeneExpressionRange("GSE75147") # Min: -5.9 | Max: 13.3 | RNA-Seq
```

```
## Classes 'data.table' and 'data.frame':   38525 obs. of  11 variables:
##  $ Probe             : int  1 10 100 1000 10000 100008586 100008587 100008588 100008589 100009606 ..
##  $ GeneSymbol        : chr  "A1BG" "NAT2" "ADA" "CDH2" ...
##  $ GeneName          : chr  "alpha-1-B glycoprotein" "N-acetyltransferase 2" "adenosine deaminase" "
##  $ NCBIid            : int  1 10 100 1000 10000 100008586 100008587 100008588 100008589 100009606 ..
##  $ G166.phf5a_kd_402 : num  1.65 -4.62 5.34 8.3 6.64 ...
##  $ 0827.phf5a_kd_861 : num  3.25 -1.62 5.11 9.07 5.35 ...
##  $ CB660.control_kd  : num  1.82 -5.96 7.38 9.19 6.91 ...
##  $ G166.phf5a_kd_861 : num  2.36 -5.51 4.25 7.93 6.67 ...
##  $ CB660.phf5a_kd_402: num  1.47 -5.96 7.23 9.47 6.85 ...
##  $ 0827.control_kd   : num  3.02 -2.94 5.77 8.81 5.46 ...
##  $ G166.control_kd   : num  2.37 -5.96 6.4 7.98 6.82 ...
##  - attr(*, ".internal.selfref")=<externalptr>
##  - attr(*, "call")= chr "https://gemma.msl.ubc.ca/rest/v2/datasets/GSE75147/data/processed"
##  - attr(*, "env")=<environment: 0x115408a40>
## NULL
```




```
## [1] "Minimum value: -5.962"
## [1] "Median value: -3.5053"
## [1] "Maximum value: 13.2761"
```

Gene expression of all 6 datasets seem to be within normal range and have appropriate distribution.

6) Extraction of Statistical Contrasts of Differential Analysis in Each Dataset

For the meta-analysis, we will be extracting the differential expression results from Gemma. The differential expression results for each dataset may include multiple result sets (e.g., one result set for the subset of the data from the hippocampus, one result set for frontal cortex). Each of these result sets may have multiple statistical contrasts (e.g., drug1 vs. vehicle, drug2 vs. vehicle). Therefore, each of the statistical contrasts is labeled with a result ID and contrast ID within the Gemma database. We will need to know which of these IDs are relevant to our project goals to easily extract their results.

We will also need to double-check that these statistical contrasts are set up in a manner that makes sense for our experiments:

1. For experiments that include more than one brain region, we will need to double-check that the results have been subsetted by brain region (instead of including brain region ("OrganismPart") as a factor in the model). If they have not been subsetted by region, we will probably need to re-run the differential expression analysis.
2. Depending on the goals of the meta-analysis, we may also need to re-run the differential expression analysis to remove other unwanted subjects (e.g., removing subjects with genotypes that might interfere with our results).

3. We will need to double-check that the comparisons include an appropriate reference group - sometimes they are reversed in Gemma (e.g., having the drug treatment set as the baseline, with vehicle as the manipulation). If this is the case, we will need to invert the effects when we input them into our meta-analysis (multiply the effects by -1).

```
# Create a function to extract statistical contrasts of differential analysis
# in each dataset
GettingResultSetInfoForDatasets <- function(ExperimentIDs) {

  # Create a dataframe to store the results
  resultsets_toscreen <- data.frame(
    ExperimentIDs = character(),
    ResultSetIDs = character(),
    ContrastIDs = character(),
    FactorCategory = character(),
    ExperimentalFactors = character(),
    BaselineFactors = character(),
    Subsetted = logical(),
    SubsetBy = character(),
    stringsAsFactors = FALSE
  )

  # Loop through each of the datasets
  for (i in c(1:length(ExperimentIDs))) {

    # Extract the differential expression analysis from the datasets
    design <- get_dataset_differential_expression_analyses(ExperimentIDs[i])

    # Check if the design dataframe has rows
    if (nrow(design) > 0) {

      # Create the vectors to store the experimental and baseline factors
      experimental_factors <- vector(mode = "character", length = nrow(design))
      baseline_factors <- vector(mode = "character", length = nrow(design))

      # Loop through each result.ID in the design dataframe
      for (j in c(1:nrow(design))) {

        # Concatenate the experimental factors into a single string
        experimental_factors[j] <- paste(design$experimental.factors[[j]]$summary,
                                         collapse = ";")

        # Concatenate the baseline factors into a single string
        baseline_factors[j] <- paste(design$baseline.factors[[j]]$summary,
                                     collapse = ";")
      }

      # Create a vector to store subset information
      SubsetBy <- vector(mode = "character", length = nrow(design))

      # Check if the design dataframe is subsetted or not
      if (design$isSubset[1] == TRUE) {

        # Loop through each result.ID to extract and concatenate subset information
```

```
        for (j in c(1:nrow(design))) {
          SubsetBy[j] <- paste(design$subsetFactor[[j]]$summary, collapse = ";")
        }
      } else {
        # If not subsetted, fill the SubsetBy vector with NA values
        SubsetBy <- rep(NA, length(design$result.ID))
      }

      # Create a temporary dataframe to store extracted info
      resultsets_for_experiment <- data.frame(
        ExperimentIDs = ExperimentIDs[i],
        ResultSetIDs = design$result.ID,
        ContrastIDs = design$contrast.ID,
        FactorCategory = design$factor.category,
        ExperimentalFactors = experimental_factors,
        BaselineFactors = baseline_factors,
        Subsetted = design$isSubset,
        SubsetBy = SubsetBy,
        stringsAsFactors = FALSE
      )

      # Append the temporary dataframe with the dataframe created initially
      resultsets_toscreen <- rbind(resultsets_toscreen, resultsets_for_experiment)

    }
  }

  # Add empty columns to store screening notes
  resultsets_toscreen <- cbind(
    resultsets_toscreen,
    Include = vector(mode = "character", length = nrow(resultsets_toscreen)),
    WrongBaseline = vector(mode = "character", length = nrow(resultsets_toscreen)),
    ResultsNotRegionSpecific = vector(mode = "character", length = nrow(resultsets_toscreen)),
    ReAnalyze = vector(mode = "character", length = nrow(resultsets_toscreen)),
    stringsAsFactors = FALSE
  )

  # Export the final dataframe into the working directory
  write.csv(resultsets_toscreen, "ResultSets_toScreen.csv")

  # Print a message to indicate that the results have been saved
  print("Output object: ResultSets_toScreen.csv")
}
```

```
# Set up a vector of the names of all datasets
ExperimentIDs <- c("GSE154958", "GSE15209", "GSE45899",
                   "GSE135306", "GSE4536","GSE75147")

# Function use
GettingResultSetInfoForDatasets(ExperimentIDs)
```

```
## [1] "Output object: ResultSets_toScreen.csv"
```

Through secondary selection process, contrast 40036 of GSE15209, contrast 94018 of GSE45899, contrast

121770 and 121772 of GSE75147 were included in the meta-analysis.

7) Downloading the DE Results from Each Dataset

```
# Create a function to download DE results and extract Log2FC and T-statistics
# for contrasts of interest
DownloadingDEResults <- function(ResultSets_contrasts){

  # Identify the unique ResultSet IDs
  # Some ResultSets may have multiple statistical contrasts, but we only want unique ResultSet IDs
  UniqueResultSetIDs <- unique(ResultSets_contrasts$ResultSetIDs)

  # Print the identified unique ResultSet IDs
  print("ResultSets identified as being of interest:")
  print(UniqueResultSetIDs)

  # Download DE results for each unique ResultSet ID
  differentials <- UniqueResultSetIDs %>%
    # The function returns a list because single experiment may have multiple
    # resultSets
    # Only take the first element of the output
    # The "resultSet" argument is used to directly access the results we need
    lapply(function(x) {get_differential_expression_values(resultSets = x)[[1]]})

  # Some datasets might not have all the advertised DE results due to a variety of
  # so we need to remove empty differential (only keep differential with rows present)
  non_missing_contrasts <- sapply(differentials, function(df) nrow(df) > 0)

  # Return the "differentials" object that contains the DE results of contrast of interest
  differentials <<- differentials[non_missing_contrasts]
  UniqueResultSetIDs <<- UniqueResultSetIDs[non_missing_contrasts]

  # Print the ResultSet IDs that had DE results
  print("ResultSets that had DE results:")
  print(UniqueResultSetIDs)

  # Print a message to inform the structure of the output "differentials"
  print("Your DE results for each of the ResultSets are stored in object differentials.")
  print("This object is structured as a list of dataframes.")
  print("Each element in the list represents a ResultSet, with the dataframe containing DE results")

  # Extract the effect sizes of Log2FC of contrasts of interest
  print("Columns of effect sizes (Log2FC) for contrasts of interest:")
  Contrasts_Log2FC <- paste("contrast_", ResultSets_contrasts$ContrastIDs,
                            "_log2fc", sep = "")
  print(Contrasts_Log2FC)

  # Extract the T-statistics of contrasts of interest for calculating sampling variances
  print("Columns of T-statistics for contrasts of interest:")
  Contrasts_Tstat <- paste("contrast_", ResultSets_contrasts$ContrastIDs,
                           "_tstat", sep = "")
  print(Contrasts_Tstat)
```

```
  # Remove the temporary Log2FC and T-statistics of contrast of interest
  rm(Contrasts_Log2FC, Contrasts_Tstat)
}
```

```
# Import the CSV file with contrasts of interest
ResultSets_contrasts <- read.csv("ResultSets_Screened.csv",
                                 header = TRUE, stringsAsFactors = FALSE)
# Function use
DownloadingDEResults(ResultSets_contrasts)
```

```
## [1] "ResultSets identified as being of interest:"
## [1] 521206 515008 524285
## [1] "ResultSets that had DE results:"
## [1] 521206 515008 524285
## [1] "Your DE results for each of the ResultSets are stored in object differentials."
## [1] "This object is structured as a list of dataframes."
## [1] "Each element in the list represents a ResultSet, with the dataframe containing DE results"
## [1] "Columns of effect sizes (Log2FC) for contrasts of interest:"
## [1] "contrast_40037_log2fc"  "contrast_40036_log2fc"  "contrast_94078_log2fc"
## [4] "contrast_121772_log2fc" "contrast_121770_log2fc"
## [1] "Columns of T-statistics for contrasts of interest:"
## [1] "contrast_40037_tstat"  "contrast_40036_tstat"  "contrast_94078_tstat"
## [4] "contrast_121772_tstat" "contrast_121770_tstat"
```

```
# Contrast 94078 in GSE45889 compares neural stem cells (NSC) vs glioblastoma stem
# cells (GSC), which is opposite in direction to the other datasets. To align the
# comparisons (GSC vs NSC), we reverse the coefficient, T-statistic, and Log2FC
# for consistency across all datasets.

# Access DE results of GSE45889 from "differentials" object
GSE45889 <- differentials[[2]]

# Reverse the coefficient, Tstat, and Log2FC of contrast 94074
GSE45889$contrast_94074_coefficient <- -1*GSE45889$contrast_94074_coefficient
GSE45889$contrast_94074_log2fc <- -1*GSE45889$contrast_94074_log2fc
GSE45889$contrast_94074_tstat <- -1*GSE45889$contrast_94074_tstat

# Reverse the coefficient, Tstat, and Log2FC of contrast 94076
GSE45889$contrast_94076_coefficient <- -1*GSE45889$contrast_94076_coefficient
GSE45889$contrast_94076_log2fc <- -1*GSE45889$contrast_94076_log2fc
GSE45889$contrast_94076_tstat <- -1*GSE45889$contrast_94076_tstat

# Reverse the coefficient, Tstat, and Log2FC of contrast 94077
GSE45889$contrast_94077_coefficient <- -1*GSE45889$contrast_94077_coefficient
GSE45889$contrast_94077_log2fc <- -1*GSE45889$contrast_94077_log2fc
GSE45889$contrast_94077_tstat <- -1*GSE45889$contrast_94077_tstat

# Reverse the coefficient, Tstat, and Log2FC of contrast 94078
GSE45889$contrast_94078_coefficient <- -1*GSE45889$contrast_94078_coefficient
GSE45889$contrast_94078_log2fc <- -1*GSE45889$contrast_94078_log2fc
GSE45889$contrast_94078_tstat <- -1*GSE45889$contrast_94078_tstat
```

```r
# Replace the original dataset with the reversed one
differentials[[2]] <- GSE45889

# Remove the temporary reversed dataset
rm(GSE45889)

# Check the structure of the "differentials" object
str(differentials)
```

```
## List of 3
##  $ :Classes 'data.table' and 'data.frame':   35587 obs. of  19 variables:
##   ..$ Probe                    : chr [1:35587] "207174_at" "221008_s_at" "231771_at" "204570_at" ..
##   ..$ NCBIid                   : chr [1:35587] "2262" "64850" "10804" "1346" ...
##   ..$ GeneSymbol               : chr [1:35587] "GPC5" "ETNPPL" "GJB6" "COX7A1" ...
##   ..$ GeneName                 : chr [1:35587] "glypican 5" "ethanolamine-phosphate phospho-lyase" "
##   ..$ pvalue                   : num [1:35587] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ corrected_pvalue         : num [1:35587] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ rank                     : num [1:35587] 2e-04 2e-04 2e-04 2e-04 2e-04 2e-04 2e-04 2e-04 2e-04 2e-04
##   ..$ contrast_40036_coefficient: num [1:35587] -0.118 0.9397 0.2132 -0.0313 -0.1611 ...
##   ..$ contrast_40036_log2fc    : num [1:35587] -0.118 0.9397 0.2132 -0.0313 -0.1611 ...
##   ..$ contrast_40036_tstat     : num [1:35587] -0.731 3.521 1.222 -0.174 -0.722 ...
##   ..$ contrast_40036_pvalue    : num [1:35587] 0.4744 0.0024 0.2372 0.8637 0.4794 ...
##   ..$ contrast_40037_coefficient: num [1:35587] -0.0044 0.1794 0.1772 -0.1156 -0.0572 ...
##   ..$ contrast_40037_log2fc    : num [1:35587] -0.0044 0.1794 0.1772 -0.1156 -0.0572 ...
##   ..$ contrast_40037_tstat     : num [1:35587] -0.0389 0.9604 1.4517 -0.9195 -0.3665 ...
##   ..$ contrast_40037_pvalue    : num [1:35587] 0.969 0.349 0.164 0.37 0.718 ...
##   ..$ contrast_40038_coefficient: num [1:35587] 3.57 7.17 5.76 4.14 5.89 ...
##   ..$ contrast_40038_log2fc    : num [1:35587] 3.57 7.17 5.76 4.14 5.89 ...
##   ..$ contrast_40038_tstat     : num [1:35587] 30.5 37.1 45.6 31.8 36.5 ...
##   ..$ contrast_40038_pvalue    : num [1:35587] 0 0 0 0 0 0 0 0 0 0 ...
##   ..- attr(*, ".internal.selfref")=<externalptr>
##   ..- attr(*, "call")= chr "https://gemma.msl.ubc.ca/rest/v2/resultSets/521206"
##   ..- attr(*, "env")=<environment: 0x117880600>
##  $ :Classes 'data.table' and 'data.frame':   35591 obs. of  23 variables:
##   ..$ Probe                    : chr [1:35591] "203131_at" "224579_at" "226213_at" "202291_s_at" ..
##   ..$ NCBIid                   : chr [1:35591] "5156" "81539" "2065" "4256" ...
##   ..$ GeneSymbol               : chr [1:35591] "PDGFRA" "SLC38A1" "ERBB3" "MGP" ...
##   ..$ GeneName                 : chr [1:35591] "platelet derived growth factor receptor alpha" "solu
##   ..$ pvalue                   : num [1:35591] 1.13e-12 1.27e-12 5.83e-13 6.93e-13 8.60e-13 ...
##   ..$ corrected_pvalue         : num [1:35591] 7.71e-09 7.71e-09 7.71e-09 7.71e-09 7.71e-09 ...
##   ..$ rank                     : num [1:35591] 1.00e-04 2.00e-04 5.49e-05 7.32e-05 9.15e-05 ...
##   ..$ contrast_94074_coefficient: num [1:35591] -3.364 1.854 3.376 -2.244 0.331 ...
##   ..$ contrast_94074_log2fc    : num [1:35591] -3.364 1.854 3.376 -2.244 0.331 ...
##   ..$ contrast_94074_tstat     : num [1:35591] -18.96 11.88 20.82 -15.06 2.16 ...
##   ..$ contrast_94074_pvalue    : num [1:35591] 3.25e-09 2.97e-07 1.29e-09 3.07e-08 5.63e-02 ...
##   ..$ contrast_94076_coefficient: num [1:35591] -1.34 7.49 6.52 4.31 -0.17 ...
##   ..$ contrast_94076_log2fc    : num [1:35591] -1.34 7.49 6.52 4.31 -0.17 ...
##   ..$ contrast_94076_tstat     : num [1:35591] -7.53 48.05 40.23 28.93 -1.1 ...
##   ..$ contrast_94076_pvalue    : num [1:35591] 1.90e-05 3.07e-13 1.82e-12 4.92e-11 2.95e-01 ...
##   ..$ contrast_94077_coefficient: num [1:35591] -3.734 0.346 0.378 4.252 -0.234 ...
##   ..$ contrast_94077_log2fc    : num [1:35591] -3.734 0.346 0.378 4.252 -0.234 ...
##   ..$ contrast_94077_tstat     : num [1:35591] -24.3 2.56 2.69 32.95 -1.76 ...
##   ..$ contrast_94077_pvalue    : num [1:35591] 2.80e-10 2.82e-02 2.25e-02 1.35e-11 1.09e-01 ...
```

```
##   ..$ contrast_94078_coefficient: num [1:35591] 4.63 1.36 6.9 4.25 7.01 ...
##   ..$ contrast_94078_log2fc     : num [1:35591] 4.63 1.36 6.9 4.25 7.01 ...
##   ..$ contrast_94078_tstat      : num [1:35591] 26.1 8.7 42.6 28.5 45.7 ...
##   ..$ contrast_94078_pvalue     : num [1:35591] 1.36e-10 5.30e-06 1.03e-12 5.72e-11 5.11e-13 ...
##   ..- attr(*, ".internal.selfref")=<externalptr>
##   ..- attr(*, "call")= chr "https://gemma.msl.ubc.ca/rest/v2/resultSets/515008"
##   ..- attr(*, "env")=<environment: 0x1154f6090>
##  $ :Classes 'data.table' and 'data.frame':   21273 obs. of  15 variables:
##   ..$ Probe                     : int [1:21273] 1281 1956 83543 440738 1293 51050 2977 26002 6299 23
##   ..$ NCBIid                    : int [1:21273] 1281 1956 83543 440738 1293 51050 2977 26002 6299 23
##   ..$ GeneSymbol                : chr [1:21273] "COL3A1" "EGFR" "AIF1L" "MAP1LC3C" ...
##   ..$ GeneName                  : chr [1:21273] "collagen type III alpha 1 chain" "epidermal growth
##   ..$ pvalue                    : num [1:21273] 1.08e-11 4.57e-12 1.08e-11 7.23e-11 8.30e-11 ...
##   ..$ corrected_pvalue          : num [1:21273] 7.90e-08 7.90e-08 7.90e-08 8.54e-08 8.54e-08 ...
##   ..$ rank                      : num [1:21273] 9.08e-05 4.54e-05 1.00e-04 7.00e-04 9.00e-04 ...
##   ..$ contrast_121770_coefficient: num [1:21273] -18.05 4.11 8.76 -8.06 -8.49 ...
##   ..$ contrast_121770_log2fc    : num [1:21273] -18.05 4.11 8.76 -8.06 -8.49 ...
##   ..$ contrast_121770_tstat     : num [1:21273] -23.7 26.2 23.6 -22.1 -21.2 ...
##   ..$ contrast_121770_pvalue    : num [1:21273] 2.84e-10 1.05e-10 3.02e-10 5.71e-10 8.64e-10 ...
##   ..$ contrast_121772_coefficient: num [1:21273] -17.37 -1.44 0.82 -10.51 -8.29 ...
##   ..$ contrast_121772_log2fc    : num [1:21273] -17.37 -1.44 0.82 -10.51 -8.29 ...
##   ..$ contrast_121772_tstat     : num [1:21273] -29.8 -8.49 1.95 -24.02 -24.78 ...
##   ..$ contrast_121772_pvalue    : num [1:21273] 2.87e-11 6.06e-06 7.92e-02 2.52e-10 1.84e-10 ...
##   ..- attr(*, ".internal.selfref")=<externalptr>
##   ..- attr(*, "call")= chr "https://gemma.msl.ubc.ca/rest/v2/resultSets/524285"
##   ..- attr(*, "env")=<environment: 0x1154b88f0>
```

```r
# Create a function to save DE results for each ResultSet
SavingGemmaDEResults_forEachResultSet <- function(differentials,
                                                  UniqueResultSetIDs,
                                                  ResultSets_contrasts){

  # Loop through each dataset (list element) in "differentials" object
  for (i in c(1:length(differentials))){

    # Get the current ResultSet ID
    ThisResultSet <- UniqueResultSetIDs[i]

    # Get the dataset ID corresponding to the current ResultSet ID
    # Some datasets have multiple ResultSets, so take the dataset ID from the
    # first matching entry
    ThisDataSet <- ResultSets_contrasts$ExperimentID[ResultSets_contrasts$ResultSetIDs == ThisResultSet]

    # Export the DE results for the current ResultSet ID into the working directory
    write.csv(differentials[[i]], paste("DEResults_", ThisDataSet, "_", ThisResultSet, ".csv", sep=""))

    # Remove the temporary IDs
    rm(ThisDataSet, ThisResultSet)
   }

  # Print a message to indicate the DE results have been exported into working directory
  print("Output object: DEResults_Dataset ID_ResultSet ID")
}
```

```
# Function use
SavingGemmaDEResults_forEachResultSet(differentials,
                                      UniqueResultSetIDs,
                                      ResultSets_contrasts)
```

```
## [1] "Output object: DEResults_Dataset ID_ResultSet ID"
```

8) Filter of DE results for rows with good gene annotation

```
# Create a function to filter for rows with good gene annotation
FilteringDEResults_GoodAnnotation <- function(DE_Results){

  # Print the total number of rows in the DE results
  print("# of rows in results")
  print(nrow(DE_Results))

  # Print the number of rows with missing NCBI annotation
  print("# of rows with missing NCBI annotation:")
  print(sum(DE_Results$NCBIid == "" | DE_Results$NCBIid == "null", na.rm = TRUE))

  # Print the number of rows with NA NCBI annotation
  print("# of rows with NA NCBI annotation:")
  print(sum(is.na(DE_Results$NCBIid)))

  # Print the number of rows with missing Gene Symbol annotation
  print("# of rows with missing Gene Symbol annotation:")
  print(sum(DE_Results$GeneSymbol == ""| DE_Results$GeneSymbol == "null", na.rm = TRUE))

  # Print the number of rows mapped to multiple NCBI IDs
  print("# of rows mapped to multiple NCBI IDs:")
  print(length(grep('\\|', DE_Results$NCBIid)))

  # Print the number of rows mapped to multiple Gene Symbols
  print("# of rows mapped to multiple Gene Symbols:")
  print(length(grep('\\|', DE_Results$GeneSymbol)))

  # Subset data containing rows with valid NCBI EntrezID (non-empty and non-null)
  DE_Results_NoNA <- DE_Results[(DE_Results$NCBIid == "" |
                                 DE_Results$NCBIid == "null") == FALSE &
                                 is.na(DE_Results$NCBIid) == FALSE,]

  # Subset data annotated with a single gene (not ambiguously mapped to more
  # than one gene)
  if(length(grep('\\|', DE_Results_NoNA$NCBIid)) == 0){
    # If there are no rows with multiple NCBI IDs, use the current subset
    DE_Results_GoodAnnotation <- DE_Results_NoNA
  } else {
    # Extract only rows annotated with a single Gene Symbol (no pipe character '|')
    DE_Results_GoodAnnotation <- DE_Results_NoNA[-(grep('\\|', DE_Results_NoNA$NCBIid)),]
  }
```

```
  # Print the number of rows with good annotation
  print("# of rows with good annotation")
  print(nrow(DE_Results_GoodAnnotation))

  # Get the name of the input object as a string for file naming
  ID <- deparse(substitute(DE_Results))

  # Export the DE results with good annotations into the working directory
  write.csv(DE_Results_GoodAnnotation, paste(ID, "_GoodAnnotation.csv", sep = ""))

  # Remove the temporary DE result objects
  rm(DE_Results_NoNA, DE_Results)

  # Print a message to the DE results with good annotations have been exported
  # into working directory
  print(paste("Output object:", ID, "_GoodAnnotation.csv", sep = ""))

  # Return the DE results with good annotation into the environment
  return(DE_Results_GoodAnnotation)
}
```

```
# Separate the DE results from object "differentials"
DEResults_GSE15209 <- differentials[[1]]
DEResults_GSE45899 <- differentials[[2]]
DEResults_GSE75147 <- differentials[[3]]


# Function use
DE_Results_GSE15209_GoodAnnotation <- FilteringDEResults_GoodAnnotation(DEResults_GSE15209)
```

```
## [1] "# of rows in results"
## [1] 35587
## [1] "# of rows with missing NCBI annotation:"
## [1] 0
## [1] "# of rows with NA NCBI annotation:"
## [1] 0
## [1] "# of rows with missing Gene Symbol annotation:"
## [1] 0
## [1] "# of rows mapped to multiple NCBI IDs:"
## [1] 0
## [1] "# of rows mapped to multiple Gene Symbols:"
## [1] 0
## [1] "# of rows with good annotation"
## [1] 35587
## [1] "Output object:DEResults_GSE15209_GoodAnnotation.csv"
```

```
DE_Results_GSE45899_GoodAnnotation <- FilteringDEResults_GoodAnnotation(DEResults_GSE45899)
```

```
## [1] "# of rows in results"
## [1] 35591
## [1] "# of rows with missing NCBI annotation:"
## [1] 0
```

```
## [1] "# of rows with NA NCBI annotation:"
## [1] 0
## [1] "# of rows with missing Gene Symbol annotation:"
## [1] 0
## [1] "# of rows mapped to multiple NCBI IDs:"
## [1] 0
## [1] "# of rows mapped to multiple Gene Symbols:"
## [1] 0
## [1] "# of rows with good annotation"
## [1] 35591
## [1] "Output object:DEResults_GSE45899_GoodAnnotation.csv"
```

```
DE_Results_GSE75147_GoodAnnotation <- FilteringDEResults_GoodAnnotation(DEResults_GSE75147)
```

```
## [1] "# of rows in results"
## [1] 21273
## [1] "# of rows with missing NCBI annotation:"
## [1] 0
## [1] "# of rows with NA NCBI annotation:"
## [1] 0
## [1] "# of rows with missing Gene Symbol annotation:"
## [1] 0
## [1] "# of rows mapped to multiple NCBI IDs:"
## [1] 0
## [1] "# of rows mapped to multiple Gene Symbols:"
## [1] 0
## [1] "# of rows with good annotation"
## [1] 21273
## [1] "Output object:DEResults_GSE75147_GoodAnnotation.csv"
```

```
# Check the structure of the DE results with good annotations
# str(DE_Results_GSE15209_GoodAnnotation)
# str(DE_Results_GSE45899_GoodAnnotation)
# str(DE_Results_GSE75147_GoodAnnotation)
```

9) Extraction of DE results for the contrasts of interest

```
# Create a function to extract the contrast ID from FC column names
GetContrastIDsforResultSet <- function(NamesOfFoldChangeColumns){

  # Split the column names using the underscore as a delimiter
  # The result is a list where each element is a vector of the split parts of
  # each column name
  ColumnNames_BrokenUp <- strsplit(NamesOfFoldChangeColumns, "_")

  # Convert the list of split names to a matrix
  MatrixOfColumnNames_BrokenUp <- do.call(rbind, ColumnNames_BrokenUp)

  # Extract the contrast IDs in the second column
  ContrastIDs_inCurrentDF <- MatrixOfColumnNames_BrokenUp[, 2]
```

```
  # Return the extracted contrast ID
  return(ContrastIDs_inCurrentDF)
}
```

```
# Create a function to extract DE results for contrasts of interest
ExtractingDEResultsForContrasts <- function(DE_Results_GoodAnnotation,
                                            Contrasts_Log2FC,
                                            Contrasts_Tstat,
                                            ResultSet_contrasts){

    # Print the column names in the DE results with good annotations for the
    # current ResultSet
    print("Columns in the DE results for the current ResultSet:")
    print(colnames(DE_Results_GoodAnnotation))

    # Print the column names that correspond to Log2FC values for contrasts of interest
    print("Columns of Log2FC for contrasts of interest within the DE results for the current ResultSet:
    NamesOfFoldChangeColumns <- colnames(DE_Results_GoodAnnotation)[colnames(DE_Results_GoodAnnotation)
    print(NamesOfFoldChangeColumns)

    # Print the column names that correspond to T-statistics values for contrasts of interest
    print("Columns of T-statistics for contrasts of interest within the DE results for the current Resul
    NamesOfTstatColumns <- colnames(DE_Results_GoodAnnotation)[colnames(DE_Results_GoodAnnotation) %in%
    print(NamesOfTstatColumns)

    # Extract contrast IDs from FC column names using the function "GetContrastIDsforResultSet"
    ContrastIDs_inCurrentDF <- GetContrastIDsforResultSet(NamesOfFoldChangeColumns)
    print("Contrast IDs for contrasts of interest within the current ResultSet:")
    print(ContrastIDs_inCurrentDF)

    # Extract dataset IDs for contrasts of interest
    DatasetIDs <- ResultSet_contrasts$ExperimentID[ResultSet_contrasts$ContrastIDs %in% ContrastIDs_inCu
    print("Dataset ID for the ResultSet and contrasts:")
    print(DatasetIDs)

    # Extract experimental factors for contrasts of interest
    Factors_inCurrentDF <- ResultSet_contrasts$ExperimentalFactors[ResultSet_contrasts$ContrastIDs %in%
    print("Experimental factors for ResultSet and contrasts:")
    print(Factors_inCurrentDF)

    # Combine dataset IDs and experimental factors to create unique identifiers
    # for each statistical comparison
    ComparisonsOfInterest <- paste(DatasetIDs, Factors_inCurrentDF, sep = "_")
    print("Current names of contrasts of interest")
    print(ComparisonsOfInterest)

    # Create a list to store extracted DE results and relevant metadata
    DE_result_contrast <- list(
      All_Columns = colnames(DE_Results_GoodAnnotation),
      NamesOfFoldChangeColumns = NamesOfFoldChangeColumns,
      NamesOfTstatColumns = NamesOfTstatColumns,
      Contrast_ID = ContrastIDs_inCurrentDF,
      Dataset_ID = DatasetIDs,
```

```
      Experimental_Factor = Factors_inCurrentDF,
      ComparisonsOfInterest = ComparisonsOfInterest
  )
    # Return the list of all extracted info
    return(DE_result_contrast)
}
```

```
# Create a function to extract the Log2FC and T-statistics of contrast of interest
GetContrastStatColumns <- function(ResultSets_contrasts){

  # Extract the effect sizes (Log2FC) of contrasts of interest
  Contrasts_Log2FC <- paste("contrast_", ResultSets_contrasts$ContrastIDs, "_log2fc", sep = "")

  # Extract the T-statistics of contrasts of interest for calculating sampling variances
  Contrasts_Tstat <- paste("contrast_", ResultSets_contrasts$ContrastIDs, "_tstat", sep = "")

  # Create a dataframe to store the extracted information
  Contrasts_Stat <- data.frame(
    ExperimentID = ResultSets_contrasts$ExperimentIDs,
    ContrastID = ResultSets_contrasts$ContrastIDs,
    Log2FC_Column = Contrasts_Log2FC,
    Tstat_Column = Contrasts_Tstat,
    stringsAsFactors = FALSE
  )

  # Return the resulting dataframe
  return(Contrasts_Stat)
}
```

```
# Function use
Contrasts_Stat_Columns <- GetContrastStatColumns(ResultSets_contrasts)
```

```
# Prepare "Contrast_Log2FC" and "Contrast_Tstat" arguments for extracting DE results
# of contrasts of interest

# GSE15209
Contrasts_Log2FC_GSE15209 <- Contrasts_Stat_Columns %>%
  filter(ExperimentID == "GSE15209") %>%
  pull(Log2FC_Column)

Contrasts_Tstat_GSE15209 <- Contrasts_Stat_Columns %>%
  filter(ExperimentID == "GSE15209") %>%
  pull(Tstat_Column)

# GSE45899
Contrasts_Log2FC_GSE45899 <- Contrasts_Stat_Columns %>%
  filter(ExperimentID == "GSE45899") %>%
  pull(Log2FC_Column)

Contrasts_Tstat_GSE45899 <- Contrasts_Stat_Columns %>%
  filter(ExperimentID == "GSE45899") %>%
  pull(Tstat_Column)
```

```r
# GSE75147
Contrasts_Log2FC_GSE75147 <- Contrasts_Stat_Columns %>%
  filter(ExperimentID == "GSE75147") %>%
  pull(Log2FC_Column)

Contrasts_Tstat_GSE75147 <- Contrasts_Stat_Columns %>%
  filter(ExperimentID == "GSE75147") %>%
  pull(Tstat_Column)
```

```r
# Function use

# GSE15209
DE_Results_Contrasts_GSE15209 <- ExtractingDEResultsForContrasts(DE_Results_GSE15209_GoodAnnotation,
                                                                 Contrasts_Log2FC_GSE15209,
                                                                 Contrasts_Tstat_GSE15209,
                                                                 ResultSets_contrasts)
```

```
## [1] "Columns in the DE results for the current ResultSet:"
##  [1] "Probe"                      "NCBIid"
##  [3] "GeneSymbol"                 "GeneName"
##  [5] "pvalue"                     "corrected_pvalue"
##  [7] "rank"                       "contrast_40036_coefficient"
##  [9] "contrast_40036_log2fc"      "contrast_40036_tstat"
## [11] "contrast_40036_pvalue"      "contrast_40037_coefficient"
## [13] "contrast_40037_log2fc"      "contrast_40037_tstat"
## [15] "contrast_40037_pvalue"      "contrast_40038_coefficient"
## [17] "contrast_40038_log2fc"      "contrast_40038_tstat"
## [19] "contrast_40038_pvalue"
## [1] "Columns of Log2FC for contrasts of interest within the DE results for the current ResultSet:"
## [1] "contrast_40036_log2fc" "contrast_40037_log2fc"
## [1] "Columns of T-statistics for contrasts of interest within the DE results for the current ResultS
## [1] "contrast_40036_tstat" "contrast_40037_tstat"
## [1] "Contrast IDs for contrasts of interest within the current ResultSet:"
## [1] "40036" "40037"
## [1] "Dataset ID for the ResultSet and contrasts:"
## [1] "GSE15209" "GSE15209"
## [1] "Experimental factors for ResultSet and contrasts:"
## [1] "glioblastoma"     "oligoastrocytoma"
## [1] "Current names of contrasts of interest"
## [1] "GSE15209_glioblastoma"     "GSE15209_oligoastrocytoma"
```

```r
# GSE45899
DE_Results_Contrasts_GSE45899 <- ExtractingDEResultsForContrasts(DE_Results_GSE45899_GoodAnnotation,
                                                                 Contrasts_Log2FC_GSE45899,
                                                                 Contrasts_Tstat_GSE45899,
                                                                 ResultSets_contrasts)
```

```
## [1] "Columns in the DE results for the current ResultSet:"
##  [1] "Probe"                      "NCBIid"
##  [3] "GeneSymbol"                 "GeneName"
##  [5] "pvalue"                     "corrected_pvalue"
##  [7] "rank"                       "contrast_94074_coefficient"
```

```
##  [9] "contrast_94074_log2fc"      "contrast_94074_tstat"
## [11] "contrast_94074_pvalue"      "contrast_94076_coefficient"
## [13] "contrast_94076_log2fc"      "contrast_94076_tstat"
## [15] "contrast_94076_pvalue"      "contrast_94077_coefficient"
## [17] "contrast_94077_log2fc"      "contrast_94077_tstat"
## [19] "contrast_94077_pvalue"      "contrast_94078_coefficient"
## [21] "contrast_94078_log2fc"      "contrast_94078_tstat"
## [23] "contrast_94078_pvalue"
## [1] "Columns of Log2FC for contrasts of interest within the DE results for the current ResultSet:"
## [1] "contrast_94078_log2fc"
## [1] "Columns of T-statistics for contrasts of interest within the DE results for the current ResultS
## [1] "contrast_94078_tstat"
## [1] "Contrast IDs for contrasts of interest within the current ResultSet:"
## [1] "94078"
## [1] "Dataset ID for the ResultSet and contrasts:"
## [1] "GSE45899"
## [1] "Experimental factors for ResultSet and contrasts:"
## [1] "neural stem cell"
## [1] "Current names of contrasts of interest"
## [1] "GSE45899_neural stem cell"
```

```
# GSE75147
DE_Results_Contrasts_GSE75147 <- ExtractingDEResultsForContrasts(DE_Results_GSE75147_GoodAnnotation,
                                                                 Contrasts_Log2FC_GSE75147,
                                                                 Contrasts_Tstat_GSE75147,
                                                                 ResultSets_contrasts)
```

```
## [1] "Columns in the DE results for the current ResultSet:"
##  [1] "Probe"                      "NCBIid"
##  [3] "GeneSymbol"                 "GeneName"
##  [5] "pvalue"                     "corrected_pvalue"
##  [7] "rank"                       "contrast_121770_coefficient"
##  [9] "contrast_121770_log2fc"     "contrast_121770_tstat"
## [11] "contrast_121770_pvalue"     "contrast_121772_coefficient"
## [13] "contrast_121772_log2fc"     "contrast_121772_tstat"
## [15] "contrast_121772_pvalue"
## [1] "Columns of Log2FC for contrasts of interest within the DE results for the current ResultSet:"
## [1] "contrast_121770_log2fc" "contrast_121772_log2fc"
## [1] "Columns of T-statistics for contrasts of interest within the DE results for the current ResultS
## [1] "contrast_121770_tstat" "contrast_121772_tstat"
## [1] "Contrast IDs for contrasts of interest within the current ResultSet:"
## [1] "121770" "121772"
## [1] "Dataset ID for the ResultSet and contrasts:"
## [1] "GSE75147" "GSE75147"
## [1] "Experimental factors for ResultSet and contrasts:"
## [1] "G166 derives from patient having disease glioblastoma"
## [2] "0827 derives from patient having disease glioblastoma"
## [1] "Current names of contrasts of interest"
## [1] "GSE75147_G166 derives from patient having disease glioblastoma"
## [2] "GSE75147_0827 derives from patient having disease glioblastoma"
```

10) Collapse of DE results to one result per gene & Calculation of standard error and sampling variance

Gene expression can be measured using multiple probes (microarray). Therefore, DE results need to be collapsed to one result per gene.

Standard error of Log2FC = Log2FC/T-statistics

Sampling variance = Average of standard error of each gene^2

```
# Create a function to average Log2FC, Tstat, standard error to one unique gene
# per dataset
CollapsingDEResults_OneResultPerGene<-function(GSE_ID,
                                               DE_Results_GoodAnnotation,
                                               ComparisonsOfInterest,
                                               NamesOfFoldChangeColumns,
                                               NamesOfTstatColumns){

  # Print a message to check if the vectors containing FC and T-stat column names
  # are in the same order as the comparisons of interest
  print("Check if the vectors containing FC and Tstat column names contain the same order as the compari

  # Print the number of unique NCBI IDs in the DE results
  print("# of rows with unique NCBI IDs:")
  print(length(unique(DE_Results_GoodAnnotation$NCBIid)))

  # Print the number of unique Gene Symbols in the DE results
  print("# of rows with unique Gene Symbols:")
  print(length(unique(DE_Results_GoodAnnotation$GeneSymbol)))

  # Create a folder named after the dataset ID to store results
  dir.create(paste("./", "Collapsing_DEResults_", GSE_ID, sep=""))

  # Set the working directory to the newly created folder
  setwd(paste("./", "Collapsing_DEResults_", GSE_ID, sep=""))

  # Create lists to store results
  DE_Results_GoodAnnotation_FoldChange_Average <- list()
  DE_Results_GoodAnnotation_Tstat_Average <- list()
  DE_Results_GoodAnnotation_SE_Average <- list()

  # Loop through each column containing FC and T-statistic info for the contrasts of interest
  for(i in c(1:length(NamesOfFoldChangeColumns))){

    # Select the Log2FC column of interest
    FoldChangeColumn <- dplyr::select(DE_Results_GoodAnnotation, NamesOfFoldChangeColumns[i])

    # Select the T-stat column of interest
    TstatColumn <- dplyr::select(DE_Results_GoodAnnotation, NamesOfTstatColumns[i])

    # Calculate the standard error (SE)
    DE_Results_GoodAnnotation_SE <- FoldChangeColumn[[1]]/TstatColumn[[1]]

    # Calculate the average Log2FC per gene
```

```
  DE_Results_GoodAnnotation_FoldChange_Average[[i]] <- tapply(FoldChangeColumn[[1]],
                                                              DE_Results_GoodAnnotation$NCBIid,
                                                              mean)

  # Calculate the average T-statistics per gene
  DE_Results_GoodAnnotation_Tstat_Average[[i]] <- tapply(TstatColumn[[1]],
                                                         DE_Results_GoodAnnotation$NCBIid,
                                                         mean)

  # Calculate the average SE per gene
  DE_Results_GoodAnnotation_SE_Average[[i]] <- tapply(DE_Results_GoodAnnotation_SE,
                                                      DE_Results_GoodAnnotation$NCBIid,
                                                      mean)

}

# Combine averaged Log2FC values into a single dataframe
DE_Results_GoodAnnotation_FoldChange_AveragedByGene <- do.call(cbind, DE_Results_GoodAnnotation_FoldC

# Print the dimensions of the averaged Fold Change matrix
print("Dimensions of Fold Change matrix, averaged by gene symbol:")
print(dim(DE_Results_GoodAnnotation_FoldChange_AveragedByGene))

# Name the columns in the dataframe to describe the contrast of interest
colnames(DE_Results_GoodAnnotation_FoldChange_AveragedByGene) <- ComparisonsOfInterest

# Export the averaged Log2FC results into the working directory
write.csv(DE_Results_GoodAnnotation_FoldChange_AveragedByGene,
          paste("DEResults_", GSE_ID, "_GoodAnnotation_FoldChange_AveragedByGene.csv", sep = ""))

# Combine T-statistics values into a single dataframe
DE_Results_GoodAnnotation_Tstat_AveragedByGene<-do.call(cbind, DE_Results_GoodAnnotation_Tstat_Averag

# Name the columns in the dataframe to describe the contrast of interest
colnames(DE_Results_GoodAnnotation_Tstat_AveragedByGene) <- ComparisonsOfInterest

# Export the averaged T-statistics results into the working directory
write.csv(DE_Results_GoodAnnotation_Tstat_AveragedByGene,
          paste("DEResults_", GSE_ID, "_GoodAnnotation_Tstat_AveragedByGene.csv", sep = ""))

# Combine SE values into a single dataframe
DE_Results_GoodAnnotation_SE_AveragedByGene<-do.call(cbind, DE_Results_GoodAnnotation_SE_Average)

# Name the columns in the dataframe to describe the contrast of interest
colnames(DE_Results_GoodAnnotation_SE_AveragedByGene) <- ComparisonsOfInterest

# Export the averaged SE results into the working directory
write.csv(DE_Results_GoodAnnotation_SE_AveragedByGene,
          paste("DEResults_", GSE_ID, "_GoodAnnotation_SE_AveragedByGene.csv", sep = ""))

# Calculate the sampling variance (SV) by squaring the SE
DE_Results_GoodAnnotation_SV <- (DE_Results_GoodAnnotation_SE_AveragedByGene)^2
```

```
  # Export the SV results into the working directory
  write.csv(DE_Results_GoodAnnotation_SV,
            paste("DEResults_", GSE_ID, "_GoodAnnotation_SV.csv", sep = ""))

  # Compile all averaged results into a single list
  TempMasterResults<-list(Log2FC = DE_Results_GoodAnnotation_FoldChange_AveragedByGene,
                          Tstat = DE_Results_GoodAnnotation_Tstat_AveragedByGene,
                          SE = DE_Results_GoodAnnotation_SE_AveragedByGene,
                          SV = DE_Results_GoodAnnotation_SV)

  # Print the name of the output
  print(paste("Output: Collapsing_DEResults", GSE_ID, sep="_"))

  # Clean up the environment by removing temporary results
  rm(DE_Results_GoodAnnotation, DE_Results_GoodAnnotation_SV,
     DE_Results_GoodAnnotation_SE, DE_Results_GoodAnnotation_FoldChange_AveragedByGene,
     DE_Results_GoodAnnotation_FoldChange_Average, DE_Results_GoodAnnotation_Tstat_AveragedByGene,
     DE_Results_GoodAnnotation_Tstat_Average, DE_Results_GoodAnnotation_SE_Average,
     FoldChangeColumn, TstatColumn, GSE_ID, ComparisonsOfInterest, NamesOfFoldChangeColumns,
     NamesOfTstatColumns)

  # Set the working directory back to the parent directory
  setwd("../")

  # Return the compiled results
  return(TempMasterResults)
}
```

```
# Prepare "NamesOfFoldChangeColumns" and "NamesOfTstatColumns" arguments for
# extracting DE results of contrasts of interest

# GSE15209
# GSC cell lines: G144 (GBM), G166 (GBM), GliN2 (GBM), G179 (Giant cell GBM),
# G174 (anaplastic oligoastrocytoma)
# NSC cell line: Fetal neural stem cells
NamesOfFoldChangeColumns_GSE15209 <- DE_Results_Contrasts_GSE15209[[2]]
NamesOfTstatColumns_GSE15209 <- DE_Results_Contrasts_GSE15209[[3]]
ComparisonsOfInterest_GSE15209 <- c("GSE15209_G144.GBM_vs_Fetal.NSC", "GSE15209_G174.AnaOlig_vs_Fetal.NS

# GSE45899
# GSC cell lines: MGG4 (GBM), MGG6 (GBM), MGG8 (GBM)
# NSC cell line: Human embryonic stem-derived neural stem cells
NamesOfFoldChangeColumns_GSE45899 <- DE_Results_Contrasts_GSE45899[[2]]
NamesOfTstatColumns_GSE45899 <- DE_Results_Contrasts_GSE45899[[3]]
ComparisonsOfInterest_GSE45899 <- c("GSE45899_MGG4.GBM_vs_Human.embryonic.NSC")

# GSE75147
# GSC cell lines: G144 (GBM), GSC-0827 (GBM)
# NSC cell line: Fetal neural stem cells (CB660)
NamesOfFoldChangeColumns_GSE75147 <- DE_Results_Contrasts_GSE75147[[2]]
NamesOfTstatColumns_GSE75147 <- DE_Results_Contrasts_GSE75147[[3]]
ComparisonsOfInterest_GSE75147 <- c("GSE75147_G166.GBM_vs_Fetal.NSC", "GSE75147_GSC-0827.GBM_vs_Fetal.NS
```

```
# Function use

# GSE15209
Collapsing_DEResults_GSE15209 <- CollapsingDEResults_OneResultPerGene("GSE15209",
                                                                       DE_Results_GSE15209_GoodAnnotatio
                                                                       ComparisonsOfInterest_GSE15209,
                                                                       NamesOfFoldChangeColumns_GSE15209
                                                                       NamesOfTstatColumns_GSE15209)
```

```
## [1] "Check if the vectors containing FC and Tstat column names contain the same order as the compari
## [1] "# of rows with unique NCBI IDs:"
## [1] 19550
## [1] "# of rows with unique Gene Symbols:"
## [1] 19550
## [1] "Dimensions of Fold Change matrix, averaged by gene symbol:"
## [1] 19550     2
## [1] "Output: Collapsing_DEResults_GSE15209"
```

```
# GSE45899
Collapsing_DEResults_GSE45899 <- CollapsingDEResults_OneResultPerGene("GSE45899",
                                                                       DE_Results_GSE45899_GoodAnnotatio
                                                                       ComparisonsOfInterest_GSE45899,
                                                                       NamesOfFoldChangeColumns_GSE45899
                                                                       NamesOfTstatColumns_GSE45899)
```

```
## [1] "Check if the vectors containing FC and Tstat column names contain the same order as the compari
## [1] "# of rows with unique NCBI IDs:"
## [1] 19552
## [1] "# of rows with unique Gene Symbols:"
## [1] 19552
## [1] "Dimensions of Fold Change matrix, averaged by gene symbol:"
## [1] 19552     1
## [1] "Output: Collapsing_DEResults_GSE45899"
```

```
# GSE75147
Collapsing_DEResults_GSE75147 <- CollapsingDEResults_OneResultPerGene("GSE75147",
                                                                       DE_Results_GSE75147_GoodAnnotatio
                                                                       ComparisonsOfInterest_GSE75147,
                                                                       NamesOfFoldChangeColumns_GSE75147
                                                                       NamesOfTstatColumns_GSE75147)
```

```
## [1] "Check if the vectors containing FC and Tstat column names contain the same order as the compari
## [1] "# of rows with unique NCBI IDs:"
## [1] 21273
## [1] "# of rows with unique Gene Symbols:"
## [1] 21273
## [1] "Dimensions of Fold Change matrix, averaged by gene symbol:"
## [1] 21273     2
## [1] "Output: Collapsing_DEResults_GSE75147"
```

11) Alignment of DE results from human cell line models

Each dataset has DE results from a slightly different list of genes.

Depending on the exact tissue dissected, the sensitivity of the transcriptional profiling platform, the representation on the transcriptional profiling platform (for microarray), and the experimental conditions, the DE results from different datasets will also be in a slightly different order.

We want to align these results so that the DE results from each dataset are columns, with each row representing a different gene.

```
# Create a function to align all mouse DE results from different datasets into
# a single dataframe for Log2FC and SV
AligningHumanDatasets <- function(ListOfHumanDEResults){

  # Create a list to store the log2FC
  Human_MetaAnalysis_FoldChange_Dfs <- list()

  # Loop through all mouse DE results
  for(i in c(1:length(ListOfHumanDEResults))){

    # Extract the Log2FC values from each dataset and put them into separate list
    # The element from the "Collapsing_DEResults_" has the format of row names
    # as Entrez Gene ID and columns containing Log2FC values
    Human_MetaAnalysis_FoldChange_Dfs[[i]] <- data.frame(Human_EntrezGene.ID = row.names(ListOfHumanDER
                                                         ListOfHumanDEResults[[i]][[1]],
                                                         stringsAsFactors = FALSE)
  }

  # Print the structure of the Log2FC lists
  print("Human_MetaAnalysis_FoldChange_Dfs:")
  print(str(Human_MetaAnalysis_FoldChange_Dfs))

  # ALign the DE results by Entrez Gene ID and turn them into a single dataframe
  # join_all can be used for object of list class
  Human_MetaAnalysis_FoldChanges <<- join_all(Human_MetaAnalysis_FoldChange_Dfs,
                                              by = "Human_EntrezGene.ID",
                                              type = "full")

  # Print the structure of the aligned Log2FC
  print("Human_MetaAnalysis_FoldChanges:")
  print(str(Human_MetaAnalysis_FoldChanges))

  # Create a list to store the SV
  Human_MetaAnalysis_SV_Dfs <- list()

  # Loop through all mouse DE results
  for(i in c(1:length(ListOfHumanDEResults))){

    # Extract the SV values from each dataset and put them into separate list
    # The element from the "Collapsing_DEResults_" has the format of row names
    # as Entrez Gene ID and columns containing SV values
    Human_MetaAnalysis_SV_Dfs[[i]] <- data.frame(Human_EntrezGene.ID = row.names(ListOfHumanDEResults[[
                                                 ListOfHumanDEResults[[i]][[4]],
                                                 stringsAsFactors = FALSE)
  }

  # Print the structure of the SV lists
  print("Human_MetaAnalysis_SV_Dfs:")
```

```
  print(str(Human_MetaAnalysis_SV_Dfs))

  # ALign the DE results by Entrez Gene ID and turn them into a single dataframe
  # join_all can be used for object of list class
  Human_MetaAnalysis_SV <<- join_all(Human_MetaAnalysis_SV_Dfs,
                                     by = "Human_EntrezGene.ID",
                                     type = "full")

  # Print the structure of the aligned SV
  print("Human_MetaAnalysis_SV:")
  print(str(Human_MetaAnalysis_SV))

  # Remove the temporary lists of Log2FC and SV
  rm(Human_MetaAnalysis_SV_Dfs, Human_MetaAnalysis_FoldChange_Dfs)
}
```

```
# Prepare the list of collapsed mouse DE results
ListOfHumanDEResults <- list(Collapsing_DEResults_GSE15209,
                             Collapsing_DEResults_GSE45899,
                             Collapsing_DEResults_GSE75147)

# Function use
AligningHumanDatasets(ListOfHumanDEResults)
```

```
## [1] "Human_MetaAnalysis_FoldChange_Dfs:"
## List of 3
##  $ :'data.frame':    19550 obs. of  3 variables:
##   ..$ Human_EntrezGene.ID               : chr [1:19550] "1" "10" "100" "1000" ...
##   ..$ GSE15209_G144.GBM_vs_Fetal.NSC    : num [1:19550] -0.267 -0.11 -0.279 -0.325 -0.53 ...
##   ..$ GSE15209_G174.AnaOlig_vs_Fetal.NSC: num [1:19550] -0.1018 -0.0046 -0.1766 -0.8076 -0.2849 ...
##  $ :'data.frame':    19552 obs. of  2 variables:
##   ..$ Human_EntrezGene.ID                     : chr [1:19552] "1" "10" "100" "1000" ...
##   ..$ GSE45899_MGG4.GBM_vs_Human.embryonic.NSC: num [1:19552] 0.9247 -1.127 -0.6016 -1.6222 -0.0666
##  $ :'data.frame':    21273 obs. of  3 variables:
##   ..$ Human_EntrezGene.ID               : chr [1:21273] "1" "2" "9" "10" ...
##   ..$ GSE75147_G166.GBM_vs_Fetal.NSC    : num [1:21273] 1.49 4.45 0.58 3.71 8.16 ...
##   ..$ GSE75147_GSC.0827.GBM_vs_Fetal.NSC: num [1:21273] 0.534 -4.927 0.623 0.412 2.619 ...
## NULL
## [1] "Human_MetaAnalysis_FoldChanges:"
## 'data.frame':    24871 obs. of  6 variables:
##  $ Human_EntrezGene.ID                     : chr  "1" "10" "100" "1000" ...
##  $ GSE15209_G144.GBM_vs_Fetal.NSC          : num  -0.267 -0.11 -0.279 -0.325 -0.53 ...
##  $ GSE15209_G174.AnaOlig_vs_Fetal.NSC      : num  -0.1018 -0.0046 -0.1766 -0.8076 -0.2849 ...
##  $ GSE45899_MGG4.GBM_vs_Human.embryonic.NSC: num  0.9247 -1.127 -0.6016 -1.6222 -0.0666 ...
##  $ GSE75147_G166.GBM_vs_Fetal.NSC          : num  1.49 3.71 -1.85 -0.39 -1.48 ...
##  $ GSE75147_GSC.0827.GBM_vs_Fetal.NSC      : num  0.534 0.412 -1.808 -1.303 -0.149 ...
## NULL
## [1] "Human_MetaAnalysis_SV_Dfs:"
## List of 3
##  $ :'data.frame':    19550 obs. of  3 variables:
##   ..$ Human_EntrezGene.ID               : chr [1:19550] "1" "10" "100" "1000" ...
##   ..$ GSE15209_G144.GBM_vs_Fetal.NSC    : num [1:19550] 0.13307 0.00773 0.09807 0.08896 0.16494 ...
##   ..$ GSE15209_G174.AnaOlig_vs_Fetal.NSC: num [1:19550] 0.06523 0.00372 0.04804 0.04357 0.08077 ...
```

```
##  $ :'data.frame':    19552 obs. of  2 variables:
##   ..$ Human_EntrezGene.ID                    : chr [1:19552] "1" "10" "100" "1000" ...
##   ..$ GSE45899_MGG4.GBM_vs_Human.embryonic.NSC: num [1:19552] 0.1037 0.0949 0.1086 0.0526 0.0696 ...
##  $ :'data.frame':    21273 obs. of  3 variables:
##   ..$ Human_EntrezGene.ID               : chr [1:21273] "1" "2" "9" "10" ...
##   ..$ GSE75147_G166.GBM_vs_Fetal.NSC    : num [1:21273] 0.147 0.71 0.19 0.727 0.914 ...
##   ..$ GSE75147_GSC.0827.GBM_vs_Fetal.NSC: num [1:21273] 0.146 1.778 0.169 0.854 1.126 ...
## NULL
## [1] "Human_MetaAnalysis_SV:"
## 'data.frame':    24871 obs. of  6 variables:
##  $ Human_EntrezGene.ID                     : chr  "1" "10" "100" "1000" ...
##  $ GSE15209_G144.GBM_vs_Fetal.NSC          : num  0.13307 0.00773 0.09807 0.08896 0.16494 ...
##  $ GSE15209_G174.AnaOlig_vs_Fetal.NSC      : num  0.06523 0.00372 0.04804 0.04357 0.08077 ...
##  $ GSE45899_MGG4.GBM_vs_Human.embryonic.NSC: num  0.1037 0.0949 0.1086 0.0526 0.0696 ...
##  $ GSE75147_G166.GBM_vs_Fetal.NSC          : num  0.147 0.727 0.1462 0.027 0.0339 ...
##  $ GSE75147_GSC.0827.GBM_vs_Fetal.NSC      : num  0.1462 0.8535 0.1248 0.0245 0.026 ...
## NULL
```

```
# Rename the output objects for the pipeline
MetaAnalysis_FoldChanges <- Human_MetaAnalysis_FoldChanges
MetaAnalysis_SV <- Human_MetaAnalysis_SV
```

12) Comparison of Log2FC across datasets

```
# Check the column names in the MetaAnalysis dataframes
colnames(MetaAnalysis_FoldChanges)
```

```
## [1] "Human_EntrezGene.ID"
## [2] "GSE15209_G144.GBM_vs_Fetal.NSC"
## [3] "GSE15209_G174.AnaOlig_vs_Fetal.NSC"
## [4] "GSE45899_MGG4.GBM_vs_Human.embryonic.NSC"
## [5] "GSE75147_G166.GBM_vs_Fetal.NSC"
## [6] "GSE75147_GSC.0827.GBM_vs_Fetal.NSC"
```

There are different ways to plot relationships across datasets:

1. Rank-rank hypergeometric overlap plots
2. Hierarchically clustered heatmaps

We can generally compare the DE results associated with different contrasts using a scatter plot and correlation analysis.

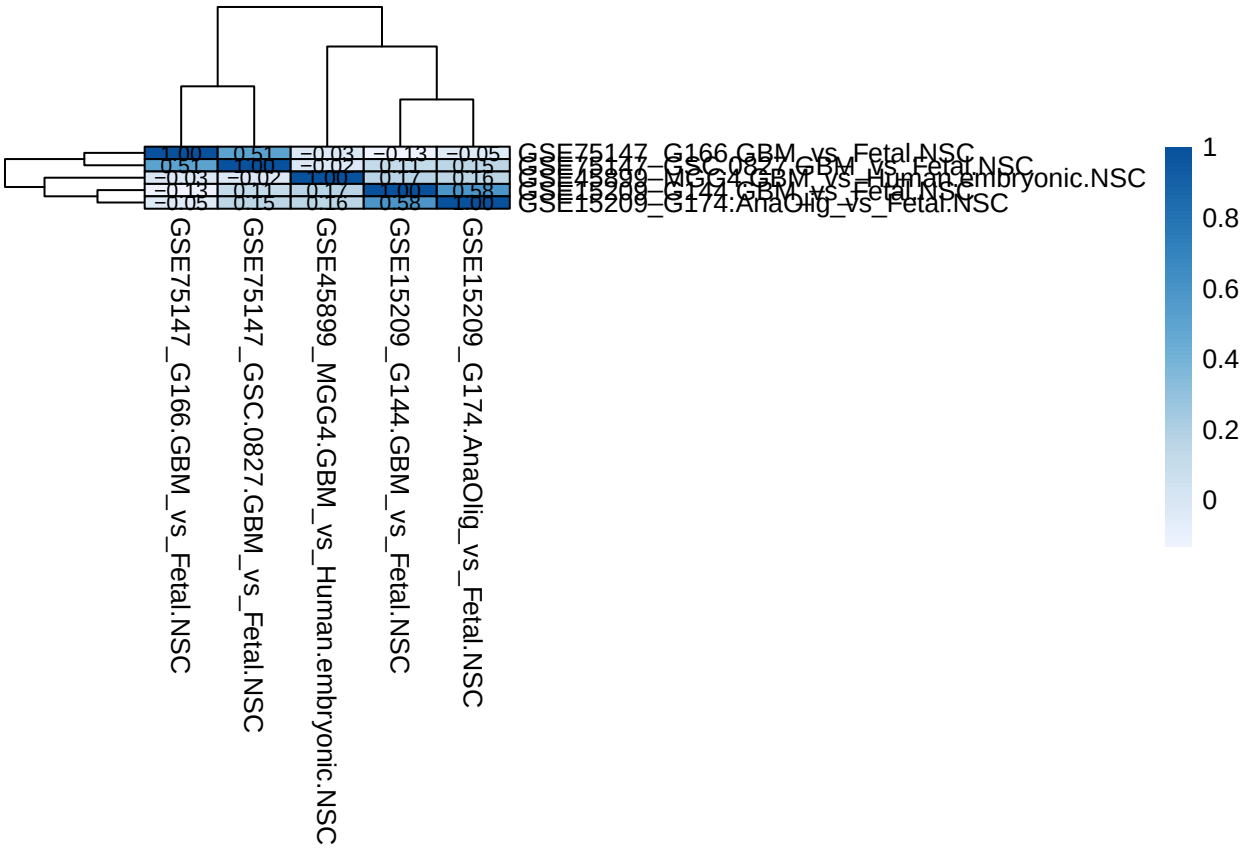
```
# Create a correlation matrix for Log2FC across datasets
# "pairwise.complete.obs" ignore any rows of DE results that do not have Log2FC
# for one of our columns
MetaAnalysis_CorMatrix_FoldChanges <- cor(as.matrix(MetaAnalysis_FoldChanges[,-c(1)]),
                                          use = "pairwise.complete.obs",
                                          method = "spearman")
# Check the correlation matrix
MetaAnalysis_CorMatrix_FoldChanges
```

```
##                                          GSE15209_G144.GBM_vs_Fetal.NSC
## GSE15209_G144.GBM_vs_Fetal.NSC                                1.0000000
## GSE15209_G174.AnaOlig_vs_Fetal.NSC                            0.5810458
## GSE45899_MGG4.GBM_vs_Human.embryonic.NSC                      0.1717834
## GSE75147_G166.GBM_vs_Fetal.NSC                               -0.1325160
## GSE75147_GSC.0827.GBM_vs_Fetal.NSC                            0.1117278
##                                          GSE15209_G174.AnaOlig_vs_Fetal.NSC
## GSE15209_G144.GBM_vs_Fetal.NSC                                   0.58104585
## GSE15209_G174.AnaOlig_vs_Fetal.NSC                               1.00000000
## GSE45899_MGG4.GBM_vs_Human.embryonic.NSC                         0.15650085
## GSE75147_G166.GBM_vs_Fetal.NSC                                  -0.04677386
## GSE75147_GSC.0827.GBM_vs_Fetal.NSC                               0.15128451
##                                          GSE45899_MGG4.GBM_vs_Human.embryonic.NSC
## GSE15209_G144.GBM_vs_Fetal.NSC                                         0.17178342
## GSE15209_G174.AnaOlig_vs_Fetal.NSC                                     0.15650085
## GSE45899_MGG4.GBM_vs_Human.embryonic.NSC                               1.00000000
## GSE75147_G166.GBM_vs_Fetal.NSC                                        -0.02831877
## GSE75147_GSC.0827.GBM_vs_Fetal.NSC                                    -0.02359925
##                                          GSE75147_G166.GBM_vs_Fetal.NSC
## GSE15209_G144.GBM_vs_Fetal.NSC                              -0.13251604
## GSE15209_G174.AnaOlig_vs_Fetal.NSC                          -0.04677386
## GSE45899_MGG4.GBM_vs_Human.embryonic.NSC                    -0.02831877
## GSE75147_G166.GBM_vs_Fetal.NSC                               1.00000000
## GSE75147_GSC.0827.GBM_vs_Fetal.NSC                           0.50501842
##                                          GSE75147_GSC.0827.GBM_vs_Fetal.NSC
## GSE15209_G144.GBM_vs_Fetal.NSC                                   0.11172782
## GSE15209_G174.AnaOlig_vs_Fetal.NSC                               0.15128451
## GSE45899_MGG4.GBM_vs_Human.embryonic.NSC                        -0.02359925
## GSE75147_G166.GBM_vs_Fetal.NSC                                   0.50501842
## GSE75147_GSC.0827.GBM_vs_Fetal.NSC                               1.00000000
```

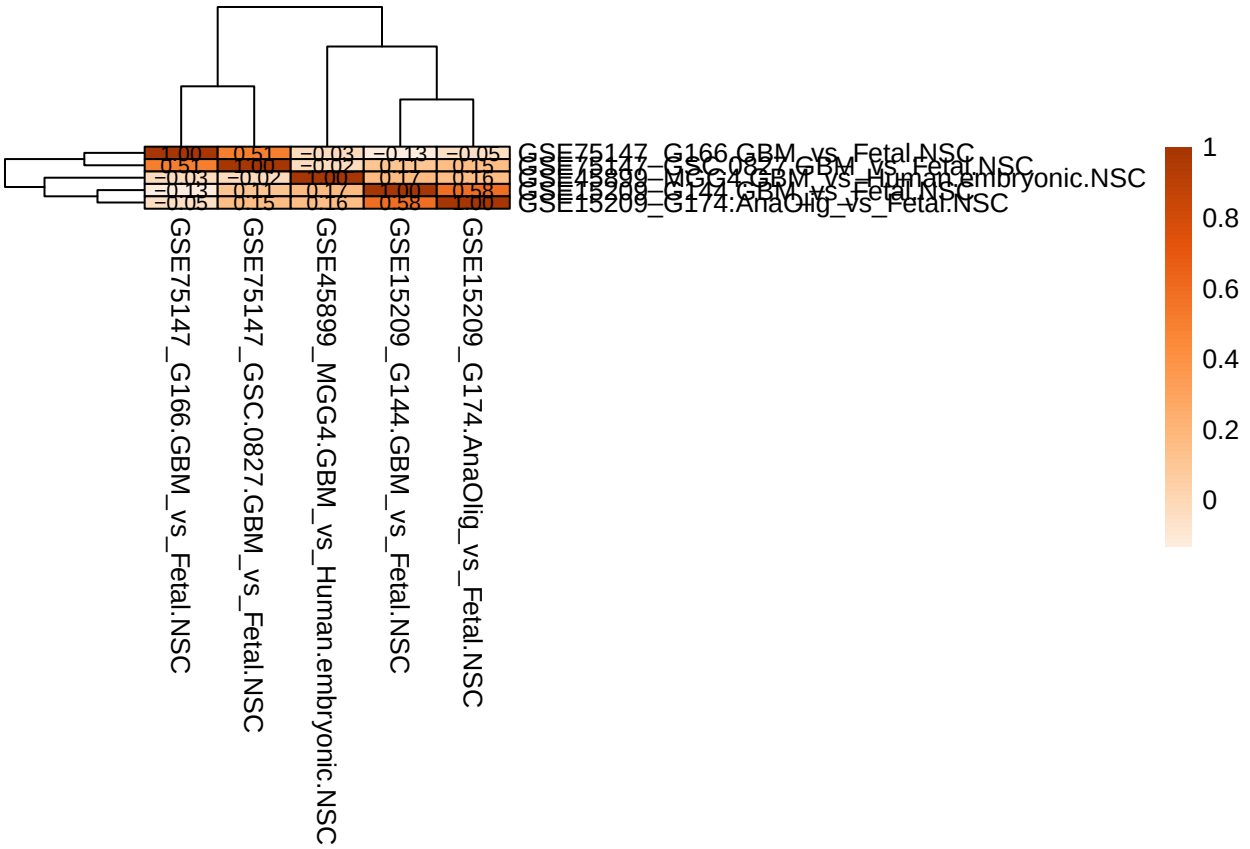
Each cell includes the correlation coefficient reflecting the similarity of the effect sizes for the comparison of between the contrast in the row and the according contrast in the column.

Correlation coefficient ranges between -1 to 1, with -1 being a perfect negative correlation and +1 being a perfect positive correlation

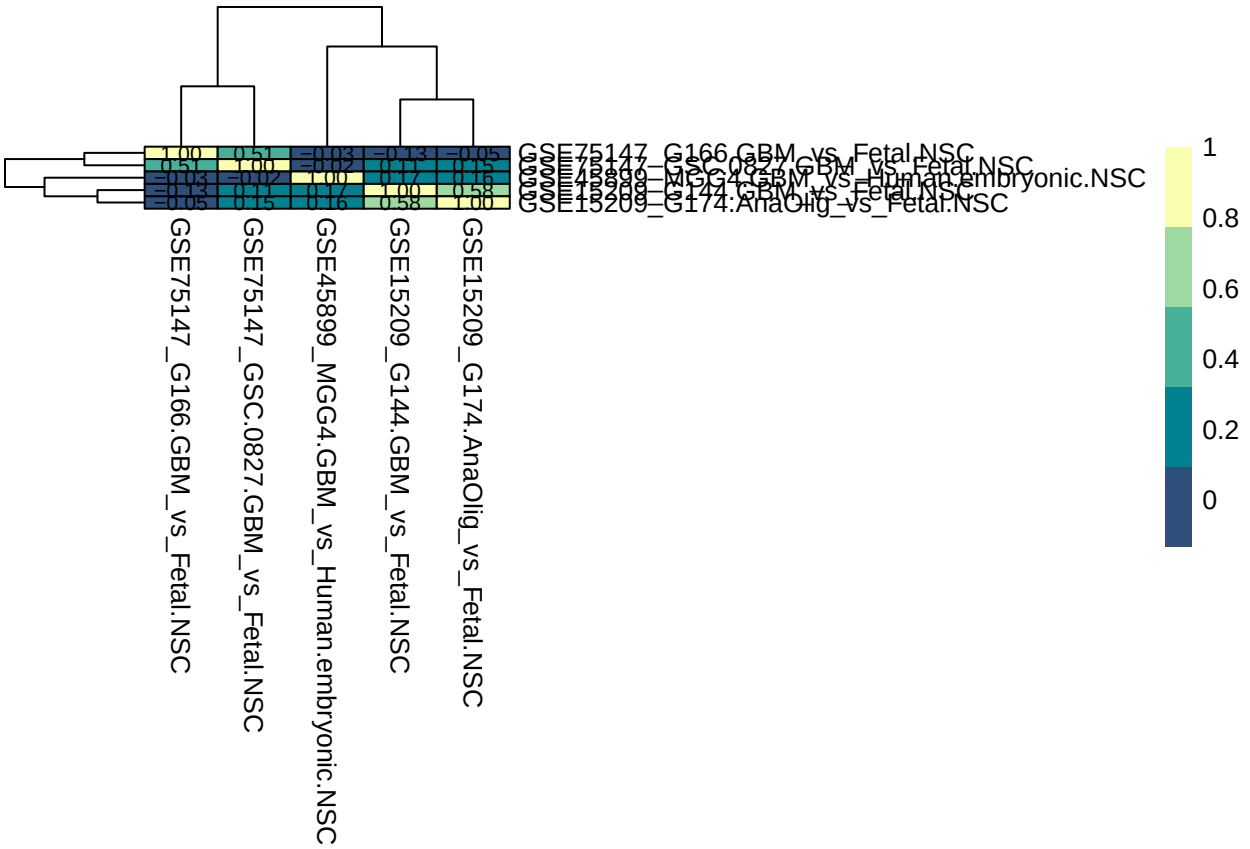
```r
# Illustrate the correlation matrix with hierarchically clustered heatmap

# Blues Palette
# png("MetaAnalysis_CorMatrix_FoldChanges_1.png", 8, 5, "in", res = 300)
pheatmap(MetaAnalysis_CorMatrix_FoldChanges,
         display_numbers = TRUE,
         fontsize = 10,
         number_format = "%.2f",
         color = colorRampPalette(brewer.pal(5, "Blues"))(100),
         border_color = "black",
         number_color = "black")
```



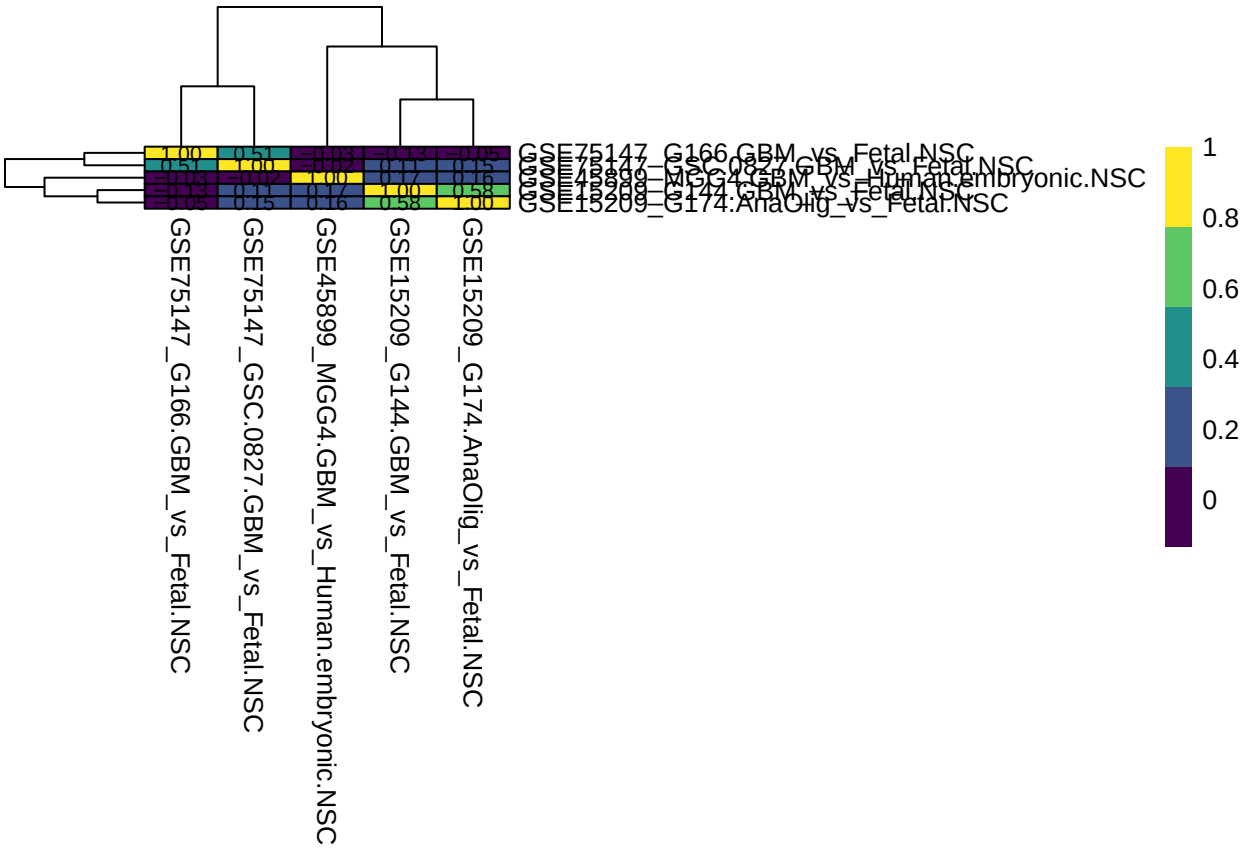

```
# dev.off()

# Oranges Palette
# png("MetaAnalysis_CorMatrix_FoldChanges_2.png", 8, 5, "in", res = 300)
pheatmap(MetaAnalysis_CorMatrix_FoldChanges,
         display_numbers = TRUE,
         fontsize = 10,
         number_format = "%.2f",
         color = colorRampPalette(brewer.pal(5, "Oranges"))(100),
         border_color = "black",
         number_color = "black")
```




```
# dev.off()

# Blues-Yellows Palette
# png("MetaAnalysis_CorMatrix_FoldChanges_3.png", 8, 5, "in", res = 300)
pheatmap(MetaAnalysis_CorMatrix_FoldChanges,
         display_numbers = TRUE,
         fontsize = 10,
         number_format = "%.2f",
         color = hcl.colors(5, "BluYl"),
         border_color = "black",
         number_color = "black")
```




```
# dev.off()

# Viridis Palette
# png("MetaAnalysis_CorMatrix_FoldChanges_4.png", 8, 5, "in", res = 300)
pheatmap(MetaAnalysis_CorMatrix_FoldChanges,
         display_numbers = TRUE,
         fontsize = 10,
         number_format = "%.2f",
         color = viridis(5),
         border_color = "black",
         number_color = "black")
```




```
# dev.off()

# The groups are placed in order by similarity, as determined by hierarchical clustering
# The lines ("tree branches") on the left and top illustrate that similarity (clustering)
# using a "dendrogram"
```

13) Meta-analysis with random effect models

The meta-analysis is performed using the effect sizes (Log2FC) and sampling variances (SV) for each gene stored in the objects *MetaAnalysis_FoldChanges* and *MetaAnalysis_SV*.

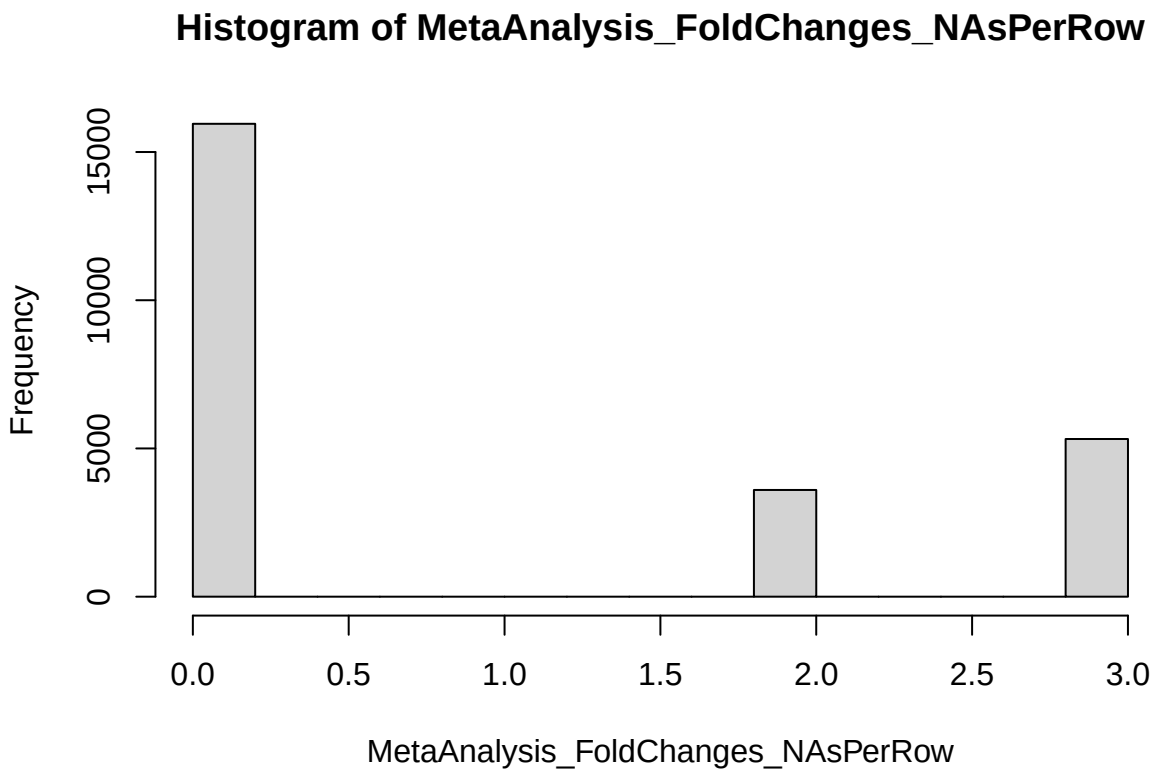
For any particular gene, it is likely that some datasets may be missing DE results. This is especially true for genes that have low levels of expression and may not be detected by less sensitive assays. It is also likely to be true for genes that were discovered more recently (i.e., not targeted by older microarray platforms) or that lack a clear ortholog in rat/mouse.

We can only run a meta-analysis if there DE results from more than 1 statistical contrast. Since the DE results from the same study (dataset) are often artificially correlated (especially if they use the same control group as the comparison), we would prefer that there are results from more than 1 dataset (not just more than 1 statistical contrast)

Before the meta-analysis, we need to decide the minimum number of DE results allowed for a gene to be included.

```
# Calculate the number of NAs (the number of statistical contrasts lacking DE
# results) in each row (for each gene)
```

```r
MetaAnalysis_FoldChanges_NAsPerRow <- apply(MetaAnalysis_FoldChanges[,-c(1)],
                                            1,
                                            function(y) sum(is.na(y)))

# Visualize the distribution of the NAs with histogram
hist(MetaAnalysis_FoldChanges_NAsPerRow)
```




```r
# Viusalize the distribution of the NAs with table
table(MetaAnalysis_FoldChanges_NAsPerRow)
```

```
## MetaAnalysis_FoldChanges_NAsPerRow
##     0     2     3
## 15952  3600  5319
```

```r
# This table tells us how many genes (rows) contain each number of NAs
# Ex: There are 15952 genes that contain no NA, meaning that these 15952 genes can
# be found across all datasets
# Ex: There are 3600 genes that contain 2 NA in 2 of the 5 contrasts
```

```r
# Create a function to run the meta-analysis
RunBasicMetaAnalysis <- function(NumberofComparisons,
                                 CutOffForNAs,
                                 MetaAnalysis_FoldChanges,
                                 MetaAnalysis_SV){
```

```r
# Calculate the number of NAs (the number of statistical contrasts lacking DE
# results) in each row (for each gene)
MetaAnalysis_FoldChanges_NAsPerRow <- apply(MetaAnalysis_FoldChanges[,-c(1)],
                                            1,
                                            function(y) sum(is.na(y)))

# Print the number of NAs per gene
print("Table of # of NAs per Row (Gene):")
print(table(MetaAnalysis_FoldChanges_NAsPerRow))

# Filter the genes with too many NAs
MetaAnalysis_FoldChanges_ForMeta <- MetaAnalysis_FoldChanges[MetaAnalysis_FoldChanges_NAsPerRow < Cut
MetaAnalysis_SV_ForMeta <- MetaAnalysis_SV[MetaAnalysis_FoldChanges_NAsPerRow < CutOffForNAs,]

# Print the structure of the filtered FC
print("MetaAnalysis_FoldChanges_ForMeta:")
print(str(MetaAnalysis_FoldChanges_ForMeta))

# Print the structure of the filtered SV
print("MetaAnalysis_SV_ForMeta:")
print(str(MetaAnalysis_SV_ForMeta))

# Create a matrix with 6 columns filled with NAs to store the meta-analysis results
metaOutput <- matrix(NA, nrow(MetaAnalysis_FoldChanges_ForMeta), 6)

# Loop through each gene to perform meta-analysis
for(i in c(1:nrow(MetaAnalysis_FoldChanges_ForMeta))){

  # Extract the Log2FC and SV value for the current gene in numeric format
  # Remove the annotation columns prior to extraction
  fc <- as.numeric(MetaAnalysis_FoldChanges_ForMeta[i, -c(1)])
  sv <- as.numeric(MetaAnalysis_SV_ForMeta[i, -c(1)])

  # Perform meta-analysis using random-effect model that treat Log2FC across
  # studies as random effects

  # Set the flag as FALSE
  # This flag is used to determine if the current iteration should be skipped
  # due to an error
  skip_to_next <- FALSE

  # The "tryCatch" function will execute the code within "{}" block. If an error
  # occurs, the code in the "error" block is executed
  tryCatch({
    # Perform meta-analysis that treat Log2FC across studies as random effects
    TempMeta <- rma(fc, sv)
    # Store estimated Log2FC
    metaOutput[i,1] <- TempMeta$b
    # Store standard error of estimate
    metaOutput[i,2] <- TempMeta$se
    # Store p-value
    metaOutput[i,3] <- TempMeta$pval
    # Store lower bound of the confidence interval
```

```
      metaOutput[i,4] <- TempMeta$ci.lb
      # Store the upper bound of the confidence interval
      metaOutput[i,5] <- TempMeta$ci.ub
      # Store the number of comparisons with available data
      metaOutput[i,6] <- NumberOfComparisons-sum(is.na(fc))
    },
      # If an error occurs during the meta-analysis, the "error" function sets
      # "skip_to_next" to TRUE
      error = function(e){skip_to_next <<- TRUE})

    # If "skip_to_next" is TRUE, the "next" statement skips the current iteration
    # and move to the next gene
    if(skip_to_next) next

    # Remove temporary variables
    rm(fc, sv)
  }

  # Name the columns in the output
  colnames(metaOutput) <- c("Log2FC_estimate",
                            "SE",
                            "pval",
                            "CI_lb",
                            "CI_ub",
                            "Number_Of_Comparisons")

  # Assign the human entrez ID as row names of the output
  row.names(metaOutput) <- MetaAnalysis_FoldChanges_ForMeta[,1]

  # Print the structure of the output
  print("metaOutput:")
  print(str(metaOutput))

  # Print the top of the output
  print("Top of metaOutput:")
  print(head(metaOutput))

  # Print the bottom of the output
  print("Bottom of metaOutput:")
  print(tail(metaOutput))

  # Change the format of the output to dataframe for the final list
  metaOutput <- as.data.frame(metaOutput)

  # Return the output and the annotation as separated lists
  return(list(metaOutput = metaOutput,
              MetaAnalysis_Annotation = MetaAnalysis_FoldChanges_ForMeta[, c(1)],
              MetaAnalysis_FoldChanges_ForMeta = MetaAnalysis_FoldChanges_ForMeta,
              MetaAnalysis_SV_ForMeta = MetaAnalysis_SV_ForMeta))
}
```

```
# Set up number of comparisons and cut off for NAs
# The number of NA = CutOffForNAs - 1
```

```r
# Ex: 0 NA => CutOffForNA = 1
# Ex: 1 NA => CutOffForNA = 2

# There are 5 contrasts and we want genes that are available in all 5 contrasts
NumberOfComparisons = 5
CutOffForNAs = 1

# Function use
# Meta-analysis with 0 NA
MetaAnalysis_Results_ONA <- suppressWarnings(
  RunBasicMetaAnalysis(NumberOfComparisons,
                       CutOffForNAs,
                       MetaAnalysis_FoldChanges,
                       MetaAnalysis_SV)
)
```

```
## [1] "Table of # of NAs per Row (Gene):"
## MetaAnalysis_FoldChanges_NAsPerRow
##     0     2     3
## 15952  3600  5319
## [1] "MetaAnalysis_FoldChanges_ForMeta:"
## 'data.frame':    15952 obs. of  6 variables:
##  $ Human_EntrezGene.ID                     : chr  "1" "10" "100" "1000" ...
##  $ GSE15209_G144.GBM_vs_Fetal.NSC          : num  -0.267 -0.11 -0.279 -0.325 -0.53 ...
##  $ GSE15209_G174.AnaOlig_vs_Fetal.NSC      : num  -0.1018 -0.0046 -0.1766 -0.8076 -0.2849 ...
##  $ GSE45899_MGG4.GBM_vs_Human.embryonic.NSC: num  0.9247 -1.127 -0.6016 -1.6222 -0.0666 ...
##  $ GSE75147_G166.GBM_vs_Fetal.NSC          : num  1.49 3.71 -1.85 -0.39 -1.48 ...
##  $ GSE75147_GSC.0827.GBM_vs_Fetal.NSC      : num  0.534 0.412 -1.808 -1.303 -0.149 ...
## NULL
## [1] "MetaAnalysis_SV_ForMeta:"
## 'data.frame':    15952 obs. of  6 variables:
##  $ Human_EntrezGene.ID                     : chr  "1" "10" "100" "1000" ...
##  $ GSE15209_G144.GBM_vs_Fetal.NSC          : num  0.13307 0.00773 0.09807 0.08896 0.16494 ...
##  $ GSE15209_G174.AnaOlig_vs_Fetal.NSC      : num  0.06523 0.00372 0.04804 0.04357 0.08077 ...
##  $ GSE45899_MGG4.GBM_vs_Human.embryonic.NSC: num  0.1037 0.0949 0.1086 0.0526 0.0696 ...
##  $ GSE75147_G166.GBM_vs_Fetal.NSC          : num  0.147 0.727 0.1462 0.027 0.0339 ...
##  $ GSE75147_GSC.0827.GBM_vs_Fetal.NSC      : num  0.1462 0.8535 0.1248 0.0245 0.026 ...
## NULL
## [1] "metaOutput:"
##  num [1:15952, 1:6] 0.498 0.448 -0.913 -0.9 -0.513 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:15952] "1" "10" "100" "1000" ...
##   ..$ : chr [1:6] "Log2FC_estimate" "SE" "pval" "CI_lb" ...
## NULL
## [1] "Top of metaOutput:"
##           Log2FC_estimate        SE         pval      CI_lb       CI_ub
## 1               0.4979527 0.3214663 0.1213806802 -0.1321097  1.12801500
## 10              0.4476195 0.7614027 0.5566074495 -1.0447023  1.93994141
## 100            -0.9126776 0.3661007 0.0126680019 -1.6302218 -0.19513340
## 1000           -0.8996414 0.2496734 0.0003142351 -1.3889923 -0.41029056
## 10000          -0.5127080 0.2783734 0.0655045176 -1.0583099  0.03289392
## 100009676       0.5208251 0.3942594 0.1864942629 -0.2519092  1.29355930
##           Number_Of_Comparisons
```

```
## 1                            5
## 10                           5
## 100                          5
## 1000                         5
## 10000                        5
## 100009676                    5
## [1] "Bottom of metaOutput:"
##      Log2FC_estimate        SE        pval      CI_lb      CI_ub
## 9990     -0.71560713 0.2727421 0.008696765 -1.2501719 -0.1810424
## 9991     -0.30011292 0.2212315 0.174922286 -0.7337186  0.1334928
## 9992     -0.15329446 0.2735637 0.575233089 -0.6894694  0.3828804
## 9993     -0.04721288 0.2694899 0.860927580 -0.5754033  0.4809776
## 9994      0.17294801 0.1497134 0.248011040 -0.1204849  0.4663809
## 9997      1.01283679 0.3951674 0.010375491  0.2383229  1.7873507
##      Number_Of_Comparisons
## 9990                     5
## 9991                     5
## 9992                     5
## 9993                     5
## 9994                     5
## 9997                     5
```

```r
# Separate the meta-analysis output and annotation
# 0 NA
metaOutput_ONA <- MetaAnalysis_Results_ONA[[1]]
MetaAnalysis_Annotation_ONA <- as.data.frame(MetaAnalysis_Results_ONA[[2]])
colnames(MetaAnalysis_Annotation_ONA) <- "Human_EntrezGene.ID"
MetaAnalysis_FoldChanges_ForMeta_ONA <- MetaAnalysis_Results_ONA[[3]]
MetaAnalysis_SV_ForMeta_ONA <- MetaAnalysis_Results_ONA[[4]]
```

14) Correction of p-value using Benjamini-Hochberg method

```r
# Create a function to correct FDR and extract genes with certain threshold of FDR
# and Log2FC
FalseDiscoveryCorrection <- function(NumberOfNAs,
                                     metaOutput,
                                     MetaAnalysis_Annotation){

  # Calculate the FDR (q-value) for each p-value using the Benjamini-Hochberg method
  tempPvalAdjMeta <- mt.rawp2adjp(metaOutput[, 3], proc = c("BH"))

  # Re-order the FDR to match with the original order
  metaPvalAdj <- tempPvalAdjMeta$adjp[order(tempPvalAdjMeta$index), ]

  # Add the FDR column to the meta-analysis output
  metaOutputFDR <- cbind(metaOutput, FDR = metaPvalAdj[, 2])

  # Rename the column to "FDR"
  colnames(metaOutputFDR)[7] <- "FDR"

  # Print the structure of meta-analysis output with FDR
  print("Meta-analysis output with FDR:")
```

```r
print(str(metaOutputFDR))

# Add annotations to the output
TempDF <- cbind(metaOutputFDR, MetaAnalysis_Annotation)

# Save the annotated results to the meta-analysis output
metaOutputFDR_annotated <- TempDF

# Export the annotated meta-analysis output into the working directory
# (remove the row names as entrez ID have been merged into the dataframe)
write.csv(metaOutputFDR_annotated,
          paste("metaOutputFDR_Annotated_", NumberOfNAs, "NA.csv", sep = ""),
          row.names = FALSE)

# Order the results by p-value
# The results are ordered by p-values because close p-values can result in the same
# FDR due to the restriction of the algorithm (the bending of FDR)
metaOutputFDR_OrderbyPval <- metaOutputFDR_annotated[order(metaOutputFDR_annotated$pval), ]

# Filter the ordered results by Log2FC threshold at 2
metaOutputFDR_OrderbyPval_Log2FC_2 <- metaOutputFDR_annotated %>%
  filter(abs(Log2FC_estimate) >= 2) %>%
  arrange(pval)

# Filter the ordered results by Log2FC threshold at 1
metaOutputFDR_OrderbyPval_Log2FC_1 <- metaOutputFDR_annotated %>%
  filter(abs(Log2FC_estimate) >= 1) %>%
  arrange(pval)

# Filter the ordered results by Log2FC threshold at 0.5
metaOutputFDR_OrderbyPval_Log2FC_0.5 <- metaOutputFDR_annotated %>%
  filter(abs(Log2FC_estimate) >= 0.5) %>%
  arrange(pval)

# Export the ordered, annotated meta-analysis output into the working directory
# (remove the row names as entrez ID have been merged into the dataframe)
write.csv(metaOutputFDR_OrderbyPval,
          paste("metaOutputFDR_OrderedByPval_", NumberOfNAs, "NA.csv",sep = ""),
          row.names = FALSE)

# Export the ordered, annotated, filtered results at Log2FC of 2
write.csv(metaOutputFDR_OrderbyPval_Log2FC_2,
          paste("metaOutputFDR_OrderedByPval_Log2FC_2_", NumberOfNAs, "NA.csv",sep = ""),
          row.names = FALSE)

# Export the ordered, annotated, filtered results at Log2FC of 1
write.csv(metaOutputFDR_OrderbyPval_Log2FC_1,
          paste("metaOutputFDR_OrderedByPval_Log2FC_1_", NumberOfNAs, "NA.csv",sep = ""),
          row.names = FALSE)

# Export the ordered, annotated, filtered results at Log2FC of 0.5
write.csv(metaOutputFDR_OrderbyPval_Log2FC_0.5,
          paste("metaOutputFDR_OrderedByPval_Log2FC_0.5_", NumberOfNAs, "NA.csv",sep = ""),
```

```
            row.names = FALSE)

# Print genes with FDR < 0.1
print("# of genes that are statistically significant following loose FDR correction (FDR < 0.10):")
print(sum(metaOutputFDR_annotated$FDR < 0.10, na.rm = TRUE))

print("# of upregulated genes that are statistically significant following FDR < 0.10:")
print(sum(metaOutputFDR_annotated$FDR < 0.10 & metaOutputFDR_annotated$Log2FC_estimate > 0, na.rm = TR

print("# of downregulated genes that are statistically significant following FDR < 0.10:")
print(sum(metaOutputFDR_annotated$FDR < 0.10 & metaOutputFDR_annotated$Log2FC_estimate < 0, na.rm = TR

# Print genes with FDR < 0.05
print("# of genes that are statistically significant following traditional FDR correction (FDR < 0.05)
print(sum(metaOutputFDR_annotated$FDR < 0.05, na.rm = TRUE))

print("# of upregulated genes that are statistically significant following FDR < 0.05:")
print(sum(metaOutputFDR_annotated$FDR < 0.05 & metaOutputFDR_annotated$Log2FC_estimate > 0, na.rm = TR

print("# of downregulated genes that are statistically significant following FDR < 0.05:")
print(sum(metaOutputFDR_annotated$FDR < 0.05 & metaOutputFDR_annotated$Log2FC_estimate < 0, na.rm = TR

print("# of genes that are statistically significant following FDR < 0.05 and |Log2FC|   2:")
print(sum(metaOutputFDR_OrderbyPval_Log2FC_2$FDR < 0.05, na.rm = TRUE))

print("# of upregulated genes that are statistically significant following FDR < 0.05 and |Log2FC|   2
print(sum(metaOutputFDR_OrderbyPval_Log2FC_2$FDR < 0.05 & metaOutputFDR_OrderbyPval_Log2FC_2$Log2FC_es

print("# of downregulated genes that are statistically significant following FDR < 0.05 and |Log2FC|
print(sum(metaOutputFDR_OrderbyPval_Log2FC_2$FDR < 0.05 & metaOutputFDR_OrderbyPval_Log2FC_2$Log2FC_es

print("# of genes that are statistically significant following FDR < 0.05 and |Log2FC|   1:")
print(sum(metaOutputFDR_OrderbyPval_Log2FC_1$FDR < 0.05, na.rm = TRUE))

print("# of upregulated genes that are statistically significant following FDR < 0.05 and |Log2FC|   1
print(sum(metaOutputFDR_OrderbyPval_Log2FC_1$FDR < 0.05 & metaOutputFDR_OrderbyPval_Log2FC_1$Log2FC_es

print("# of downregulated genes that are statistically significant following FDR < 0.05 and |Log2FC|
print(sum(metaOutputFDR_OrderbyPval_Log2FC_1$FDR < 0.05 & metaOutputFDR_OrderbyPval_Log2FC_1$Log2FC_es

print("# of genes that are statistically significant following FDR < 0.05 and |Log2FC|   0.5:")
print(sum(metaOutputFDR_OrderbyPval_Log2FC_0.5$FDR < 0.05, na.rm = TRUE))

print("# of upregulated genes that are statistically significant following FDR < 0.05 and |Log2FC|   0
print(sum(metaOutputFDR_OrderbyPval_Log2FC_0.5$FDR < 0.05 & metaOutputFDR_OrderbyPval_Log2FC_0.5$Log2

print("# of downregulated genes that are statistically significant following FDR < 0.05 and |Log2FC|
print(sum(metaOutputFDR_OrderbyPval_Log2FC_0.5$FDR < 0.05 & metaOutputFDR_OrderbyPval_Log2FC_0.5$Log2

# Print top results
print("Top 20 results ordered by p-values:")
print(head(metaOutputFDR_OrderbyPval$Human_EntrezGene.ID, 20))
```

```
  # Print top results with |Log2FC|  2
  print("Top 20 results ordered by p-values with |Log2FC|  2:")
  print(head(metaOutputFDR_OrderbyPval_Log2FC_1$Human_EntrezGene.ID, 20))

  # Print top results with |Log2FC|  1
  print("Top 20 results ordered by p-values with |Log2FC|  1:")
  print(head(metaOutputFDR_OrderbyPval_Log2FC_1$Human_EntrezGene.ID, 20))

  # Print top results with |Log2FC|  0.5
  print("Top 20 results ordered by p-values with |Log2FC|  0.5:")
  print(head(metaOutputFDR_OrderbyPval_Log2FC_0.5$Human_EntrezGene.ID, 20))

  # Return the annotated output and ordered output with FDR
  return(list(metaOutputFDR = metaOutputFDR,
              metaOutputFDR_annotated = metaOutputFDR_annotated,
              metaOutputFDR_OrderbyPval = metaOutputFDR_OrderbyPval,
              metaOutputFDR_OrderbyPval_Log2FC_2 = metaOutputFDR_OrderbyPval_Log2FC_2,
              metaOutputFDR_OrderbyPval_Log2FC_1 = metaOutputFDR_OrderbyPval_Log2FC_1,
              metaOutputFDR_OrderbyPval_Log2FC_0.5 = metaOutputFDR_OrderbyPval_Log2FC_0.5))

  # Remove temporary objects
  rm(tempPvalAdjMeta, metaPvalAdj)
}
```

```
# Function use

# Meta-analysis with 0 NA
metaOutputFDR_all_0NA <- FalseDiscoveryCorrection(0,
                                                  metaOutput_0NA,
                                                  MetaAnalysis_Annotation_0NA)
```

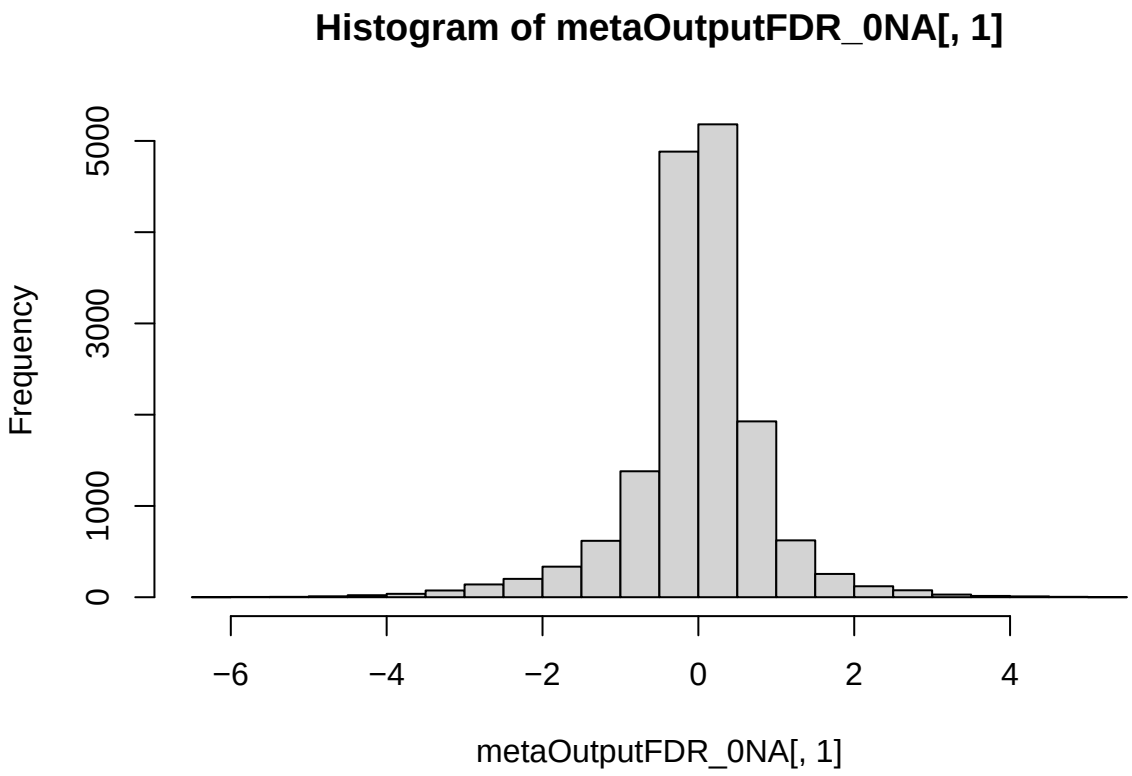
```
## [1] "Meta-analysis output with FDR:"
## 'data.frame':    15952 obs. of  7 variables:
##  $ Log2FC_estimate      : num  0.498 0.448 -0.913 -0.9 -0.513 ...
##  $ SE                   : num  0.321 0.761 0.366 0.25 0.278 ...
##  $ pval                 : num  0.121381 0.556607 0.012668 0.000314 0.065505 ...
##  $ CI_lb                : num  -0.132 -1.045 -1.63 -1.389 -1.058 ...
##  $ CI_ub                : num  1.128 1.9399 -0.1951 -0.4103 0.0329 ...
##  $ Number_Of_Comparisons: num  5 5 5 5 5 5 5 5 5 5 ...
##  $ FDR                  : num  0.42258 0.76905 0.12706 0.00885 0.32789 ...
## NULL
## [1] "# of genes that are statistically significant following loose FDR correction (FDR < 0.10):"
## [1] 1389
## [1] "# of upregulated genes that are statistically significant following FDR < 0.10:"
## [1] 761
## [1] "# of downregulated genes that are statistically significant following FDR < 0.10:"
## [1] 628
## [1] "# of genes that are statistically significant following traditional FDR correction (FDR < 0.05)
## [1] 1023
## [1] "# of upregulated genes that are statistically significant following FDR < 0.05:"
## [1] 553
## [1] "# of downregulated genes that are statistically significant following FDR < 0.05:"
## [1] 470
```

```
## [1] "# of genes that are statistically significant following FDR < 0.05 and |Log2FC|   2:"
## [1] 66
## [1] "# of upregulated genes that are statistically significant following FDR < 0.05 and |Log2FC|   2:
## [1] 34
## [1] "# of downregulated genes that are statistically significant following FDR < 0.05 and |Log2FC|
## [1] 32
## [1] "# of genes that are statistically significant following FDR < 0.05 and |Log2FC|   1:"
## [1] 249
## [1] "# of upregulated genes that are statistically significant following FDR < 0.05 and |Log2FC|   1:
## [1] 139
## [1] "# of downregulated genes that are statistically significant following FDR < 0.05 and |Log2FC|
## [1] 110
## [1] "# of genes that are statistically significant following FDR < 0.05 and |Log2FC|   0.5:"
## [1] 719
## [1] "# of upregulated genes that are statistically significant following FDR < 0.05 and |Log2FC|   0.
## [1] 403
## [1] "# of downregulated genes that are statistically significant following FDR < 0.05 and |Log2FC|
## [1] 316
## [1] "Top 20 results ordered by p-values:"
##  [1] "7020"   "51373"  "23366"  "10549"  "221830" "54838"  "221895" "11097"
##  [9] "54543"  "115416" "27175"  "282991" "10765"  "5230"   "2535"   "1371"
## [17] "6603"   "3708"   "10491"  "7422"
## [1] "Top 20 results ordered by p-values with |Log2FC|   2:"
##  [1] "7020"   "51373"  "10549"  "221830" "54838"  "221895" "11097"  "54543"
##  [9] "5230"   "2535"   "1371"   "3708"   "7422"   "2297"   "84282"  "1051"
## [17] "388722" "5321"   "23643"  "285016"
## [1] "Top 20 results ordered by p-values with |Log2FC|   1:"
##  [1] "7020"   "51373"  "10549"  "221830" "54838"  "221895" "11097"  "54543"
##  [9] "5230"   "2535"   "1371"   "3708"   "7422"   "2297"   "84282"  "1051"
## [17] "388722" "5321"   "23643"  "285016"
## [1] "Top 20 results ordered by p-values with |Log2FC|   0.5:"
##  [1] "7020"   "51373"  "23366"  "10549"  "221830" "54838"  "221895" "11097"
##  [9] "54543"  "115416" "27175"  "282991" "10765"  "5230"   "2535"   "1371"
## [17] "6603"   "3708"   "10491"  "7422"
```

```
# Separate the FDR outputs into individual objects

# Meta-analysis with FDR output only
metaOutputFDR_ONA <- metaOutputFDR_all_ONA[[1]]

# Meta-analysis with FDR output with annotations
metaOutputFDR_Annotated_ONA <- metaOutputFDR_all_ONA[[2]]

# Meta-analysis with FDR output ordered by p-values
metaOutputFDR_OrderByPval_ONA <- metaOutputFDR_all_ONA[[3]]

# Meta-analysis with FDR output ordered by p-values filtered by Log2FC at 2
metaOutputFDR_OrderbyPval_Log2FC_2_ONA <- metaOutputFDR_all_ONA[[4]]

# Meta-analysis with FDR output ordered by p-values filtered by Log2FC at 1
metaOutputFDR_OrderbyPval_Log2FC_1_ONA <- metaOutputFDR_all_ONA[[5]]

# Meta-analysis with FDR output ordered by p-values filtered by Log2FC at 0.5
metaOutputFDR_OrderbyPval_Log2FC_0.5_ONA <- metaOutputFDR_all_ONA[[6]]
```

15) Forest plots for statistically significant genes

```
# Check the range of Log2FC values
hist(metaOutputFDR_0NA[, 1], breaks = 40)
```




```
# Create a function to generate forest plot
MakeForestPlots <- function(metaOutputFDR_annotated,
                            GeneEntrezID,
                            MetaAnalysis_FoldChanges_ForMeta,
                            MetaAnalysis_SV_ForMeta){

  # Extract the row corresponding to the provided Gene Entrez ID
  gene_row <- metaOutputFDR_annotated[metaOutputFDR_annotated$Human_EntrezGene.ID == GeneEntrezID, ]

  # Check if the gene exists
  if (nrow(gene_row) == 0){
    stop(paste("No gene found for GeneEntrezID:", GeneEntrezID))
  }

  # Extract Log2FC and SV
  fc <- as.numeric(MetaAnalysis_FoldChanges_ForMeta[MetaAnalysis_FoldChanges_ForMeta$Human_EntrezGene.I
  sv <- as.numeric(MetaAnalysis_SV_ForMeta[MetaAnalysis_FoldChanges_ForMeta$Human_EntrezGene.ID == Gene

  # Create the filename for the output plot
```

```
  filename <- paste("ForestPlot_GeneEntrezID_", GeneEntrezID, ".png", sep = "")

  # Open a PNG device with specified resolution
  png(filename, height = 5, width = 8, units = "in", res = 300)

  # Create the forest plot
  forest.rma(rma(fc, sv),
             slab = colnames(MetaAnalysis_FoldChanges_ForMeta)[-c(1)],
             xlim = c(-10, 10),
             cex = 0.65)

  # Add labels to the plot
  mtext(paste("GeneEntrezID:", GeneEntrezID, sep = ""),
        line = -1.5,
        cex = 1.5)

  # Close the PNG device
  dev.off()
}
```

```
# Function use

# Top genes ordered by p-values with |Log2FC|  2

# "7020"
MakeForestPlots(metaOutputFDR_Annotated_ONA,
                "7020",
                MetaAnalysis_FoldChanges_ForMeta_ONA,
                MetaAnalysis_SV_ForMeta_ONA)
```

```
## pdf
##   2
```

```
# "51373"
MakeForestPlots(metaOutputFDR_Annotated_ONA,
                "51373",
                MetaAnalysis_FoldChanges_ForMeta_ONA,
                MetaAnalysis_SV_ForMeta_ONA)
```

```
## pdf
##   2
```

```
# "10549"
MakeForestPlots(metaOutputFDR_Annotated_ONA,
                "10549",
                MetaAnalysis_FoldChanges_ForMeta_ONA,
                MetaAnalysis_SV_ForMeta_ONA)
```

```
## pdf
##   2
```

```
# "221830"
MakeForestPlots(metaOutputFDR_Annotated_ONA,
                "221830",
                MetaAnalysis_FoldChanges_ForMeta_ONA,
                MetaAnalysis_SV_ForMeta_ONA)
```

```
## pdf
##   2
```

```
# "54838"
MakeForestPlots(metaOutputFDR_Annotated_ONA,
                "54838",
                MetaAnalysis_FoldChanges_ForMeta_ONA,
                MetaAnalysis_SV_ForMeta_ONA)
```

```
## pdf
##   2
```

```
# "221895"
MakeForestPlots(metaOutputFDR_Annotated_ONA,
                "221895",
                MetaAnalysis_FoldChanges_ForMeta_ONA,
                MetaAnalysis_SV_ForMeta_ONA)
```

```
## pdf
##   2
```

```
# "11097"
MakeForestPlots(metaOutputFDR_Annotated_ONA,
                "11097",
                MetaAnalysis_FoldChanges_ForMeta_ONA,
                MetaAnalysis_SV_ForMeta_ONA)
```

```
## pdf
##   2
```

```
# "54543"
MakeForestPlots(metaOutputFDR_Annotated_ONA,
                "54543",
                MetaAnalysis_FoldChanges_ForMeta_ONA,
                MetaAnalysis_SV_ForMeta_ONA)
```

```
## pdf
##   2
```

```
# "5230"
MakeForestPlots(metaOutputFDR_Annotated_ONA,
                "5230",
                MetaAnalysis_FoldChanges_ForMeta_ONA,
                MetaAnalysis_SV_ForMeta_ONA)
```

```
## pdf
##   2
```

```
# "2535"
MakeForestPlots(metaOutputFDR_Annotated_ONA,
                "2535",
                MetaAnalysis_FoldChanges_ForMeta_ONA,
                MetaAnalysis_SV_ForMeta_ONA)
```

```
## pdf
##   2
```

Interpretation of Forest Plots

1. Study Labels (Left Side)

- The labels on the left side of the plot indicate the individual contrasts included in the meta-analysis.

2. Effect Sizes and Confidence Intervals (Right side)

- Each horizontal line represents the effect size (Log2FC) for a particular contrast along with its confidence interval. The squares represent the point of estimate of the effect size for each contrast. The horizontal lines extending from the squares are the confidence intervals for the effect size.
- The size of the squares represent the weight of each contrast in the meta-analysis, with larger square indicating more weight.

3. Overall Effect (Bottom)

- The diamond shape at the bottom of the plot represents the overall effect size calculated from the meta-analysis (random-effects model).
- The width of the diamond represents the confidence interval for the overall effect size.

4. X-axis (Observed Outcome)

- The x-axis shows the scale/range for the effect sizes (Log2FC).
- The vertical dash line at 0 indicate no effect. Effect sizes to the left of this line suggest a decrease, while those to the right suggest an increase.

16) Retrieval the function of statistically significant gene

```
# Create a function to get gene function from NCBI using Entrez Gene ID
GetGeneFunctionByID <- function(NumberOfNAs, Log2FCThreshold, EntrezGeneList) {

  # Create an empty list to store the result
  gene_info_list <- list()

  # Loop through each Entrez Gene ID
  for(i in EntrezGeneList){
```

```
  # Print the message indicating which gene is being processed
  print(paste("Processing Entrez Gene ID:", i, sep = ""))

  # Retrieve the gene summary using the Entrez Gene ID
  gene_summary <- tryCatch({
    entrez_summary(db = "gene", id = i)
  },

  # In case of error, print a warning and return NULL
  error = function(e){
    warning(paste("Error in retrieving data for Entrez Gene ID:", i, sep = "", e$message))
    return(NULL)
  })

  # Check if the gene summary was successfully retrieved
  if(!is.null(gene_summary)){
    # If the gene summaray is not NULL, extract relevant info
    gene_info_list[[i]] <- list(
      # ID
      Entrez_ID = gene_summary$uid,
      # Gene symbol
      Gene_symbol = gene_summary$name,
      # Full name of gene
      Gene_description = gene_summary$description,
      # Location on chromosome
      Chromosome = gene_summary$chromosome,
      # Detailed location
      Map_location = gene_summary$maplocation,
      # Scientific name of model
      Model_scientific_name = gene_summary$organism$scientificname,
      # Common name of model
      Model_common_name = gene_summary$organism$commonname,
      # Function of the gene
      Gene_function = gene_summary$summary
    )
  } else {
    # If the gene summary is NULL, return NA for all columns
    gene_info_list[[i]] <- list(
      Entrez_ID = i,
      Gene_symbol = NA,
      Gene_description = NA,
      Chromosome = NA,
      Map_location = NA,
      Model_scientific_name = NA,
      Model_common_name = NA,
      Gene_function = NA
    )
  }
  # Introduce a delay of 0.5 seconds between request to avoid hitting API rate limit
  Sys.sleep(0.5)
}

# Convert the final list into a dataframe
```

```
  gene_info_df <- do.call(rbind, lapply(gene_info_list, as.data.frame))
  rownames(gene_info_df) <- NULL

  # Export the dataframe of gene function into working directory
  write.csv(gene_info_df,
            paste("Gene_function_Log2FC_", Log2FCThreshold, "_", NumberOfNAs, "NA.csv", sep = ""))

  # Return the dataframe of gene function
  return(gene_info_df)
}
```

```
# Extract gene entrez ID for genes with FDR < 0.05 & |Log2FC|   1
EntrezGeneList_OrderbyPval_Log2FC_1_0NA <- metaOutputFDR_OrderbyPval_Log2FC_1_0NA %>%
  filter(FDR < 0.05) %>%
  pull(Human_EntrezGene.ID)

# Extract gene entrez ID for genes with FDR < 0.05 & |Log2FC|   2
EntrezGeneList_OrderbyPval_Log2FC_2_0NA <- metaOutputFDR_OrderbyPval_Log2FC_2_0NA %>%
  filter(FDR < 0.05) %>%
  pull(Human_EntrezGene.ID)

# Function use
GeneFunction_Log2FC_1_0NA <- GetGeneFunctionByID(0,
                                                 1,
                                                 EntrezGeneList_OrderbyPval_Log2FC_1_0NA)
```

```
## [1] "Processing Entrez Gene ID:7020"
## [1] "Processing Entrez Gene ID:51373"
## [1] "Processing Entrez Gene ID:10549"
## [1] "Processing Entrez Gene ID:221830"
## [1] "Processing Entrez Gene ID:54838"
## [1] "Processing Entrez Gene ID:221895"
## [1] "Processing Entrez Gene ID:11097"
## [1] "Processing Entrez Gene ID:54543"
## [1] "Processing Entrez Gene ID:5230"
## [1] "Processing Entrez Gene ID:2535"
## [1] "Processing Entrez Gene ID:1371"
## [1] "Processing Entrez Gene ID:3708"
## [1] "Processing Entrez Gene ID:7422"
## [1] "Processing Entrez Gene ID:2297"
## [1] "Processing Entrez Gene ID:84282"
## [1] "Processing Entrez Gene ID:1051"
## [1] "Processing Entrez Gene ID:388722"
## [1] "Processing Entrez Gene ID:5321"
## [1] "Processing Entrez Gene ID:23643"
## [1] "Processing Entrez Gene ID:285016"
## [1] "Processing Entrez Gene ID:4254"
## [1] "Processing Entrez Gene ID:100506325"
## [1] "Processing Entrez Gene ID:976"
## [1] "Processing Entrez Gene ID:26230"
## [1] "Processing Entrez Gene ID:3315"
## [1] "Processing Entrez Gene ID:144577"
## [1] "Processing Entrez Gene ID:2990"
```

```
## [1] "Processing Entrez Gene ID:97"
## [1] "Processing Entrez Gene ID:8985"
## [1] "Processing Entrez Gene ID:3673"
## [1] "Processing Entrez Gene ID:5634"
## [1] "Processing Entrez Gene ID:84836"
## [1] "Processing Entrez Gene ID:928"
## [1] "Processing Entrez Gene ID:9989"
## [1] "Processing Entrez Gene ID:160335"
## [1] "Processing Entrez Gene ID:55752"
## [1] "Processing Entrez Gene ID:10579"
## [1] "Processing Entrez Gene ID:29902"
## [1] "Processing Entrez Gene ID:119032"
## [1] "Processing Entrez Gene ID:150967"
## [1] "Processing Entrez Gene ID:121512"
## [1] "Processing Entrez Gene ID:135154"
## [1] "Processing Entrez Gene ID:8996"
## [1] "Processing Entrez Gene ID:55744"
## [1] "Processing Entrez Gene ID:5799"
## [1] "Processing Entrez Gene ID:8497"
## [1] "Processing Entrez Gene ID:9260"
## [1] "Processing Entrez Gene ID:4741"
## [1] "Processing Entrez Gene ID:92181"
## [1] "Processing Entrez Gene ID:57407"
## [1] "Processing Entrez Gene ID:3385"
## [1] "Processing Entrez Gene ID:64101"
## [1] "Processing Entrez Gene ID:136"
## [1] "Processing Entrez Gene ID:55506"
## [1] "Processing Entrez Gene ID:4071"
## [1] "Processing Entrez Gene ID:205"
## [1] "Processing Entrez Gene ID:2774"
## [1] "Processing Entrez Gene ID:4826"
## [1] "Processing Entrez Gene ID:1846"
## [1] "Processing Entrez Gene ID:6119"
## [1] "Processing Entrez Gene ID:2729"
## [1] "Processing Entrez Gene ID:27092"
## [1] "Processing Entrez Gene ID:100093630"
## [1] "Processing Entrez Gene ID:25999"
## [1] "Processing Entrez Gene ID:23522"
## [1] "Processing Entrez Gene ID:4094"
## [1] "Processing Entrez Gene ID:1612"
## [1] "Processing Entrez Gene ID:7041"
## [1] "Processing Entrez Gene ID:9315"
## [1] "Processing Entrez Gene ID:1978"
## [1] "Processing Entrez Gene ID:10079"
## [1] "Processing Entrez Gene ID:54995"
## [1] "Processing Entrez Gene ID:5087"
## [1] "Processing Entrez Gene ID:7570"
## [1] "Processing Entrez Gene ID:3397"
## [1] "Processing Entrez Gene ID:54874"
## [1] "Processing Entrez Gene ID:563"
## [1] "Processing Entrez Gene ID:57717"
## [1] "Processing Entrez Gene ID:29887"
## [1] "Processing Entrez Gene ID:2180"
## [1] "Processing Entrez Gene ID:1741"
```

```
## [1] "Processing Entrez Gene ID:900"
## [1] "Processing Entrez Gene ID:1690"
## [1] "Processing Entrez Gene ID:9054"
## [1] "Processing Entrez Gene ID:54733"
## [1] "Processing Entrez Gene ID:9540"
## [1] "Processing Entrez Gene ID:2697"
## [1] "Processing Entrez Gene ID:100287482"
## [1] "Processing Entrez Gene ID:10135"
## [1] "Processing Entrez Gene ID:25805"
## [1] "Processing Entrez Gene ID:516"
## [1] "Processing Entrez Gene ID:7464"
## [1] "Processing Entrez Gene ID:9891"
## [1] "Processing Entrez Gene ID:55244"
## [1] "Processing Entrez Gene ID:9590"
## [1] "Processing Entrez Gene ID:79789"
## [1] "Processing Entrez Gene ID:10622"
## [1] "Processing Entrez Gene ID:7108"
## [1] "Processing Entrez Gene ID:114908"
## [1] "Processing Entrez Gene ID:79778"
## [1] "Processing Entrez Gene ID:4747"
## [1] "Processing Entrez Gene ID:91419"
## [1] "Processing Entrez Gene ID:79590"
## [1] "Processing Entrez Gene ID:2157"
## [1] "Processing Entrez Gene ID:85028"
## [1] "Processing Entrez Gene ID:9518"
## [1] "Processing Entrez Gene ID:6506"
## [1] "Processing Entrez Gene ID:55691"
## [1] "Processing Entrez Gene ID:5925"
## [1] "Processing Entrez Gene ID:56963"
## [1] "Processing Entrez Gene ID:3632"
## [1] "Processing Entrez Gene ID:51012"
## [1] "Processing Entrez Gene ID:57508"
## [1] "Processing Entrez Gene ID:54898"
## [1] "Processing Entrez Gene ID:230"
## [1] "Processing Entrez Gene ID:56994"
## [1] "Processing Entrez Gene ID:27244"
## [1] "Processing Entrez Gene ID:5780"
## [1] "Processing Entrez Gene ID:84262"
## [1] "Processing Entrez Gene ID:23255"
## [1] "Processing Entrez Gene ID:23305"
## [1] "Processing Entrez Gene ID:23564"
## [1] "Processing Entrez Gene ID:10560"
## [1] "Processing Entrez Gene ID:10409"
## [1] "Processing Entrez Gene ID:1410"
## [1] "Processing Entrez Gene ID:10367"
## [1] "Processing Entrez Gene ID:79675"
## [1] "Processing Entrez Gene ID:65251"
## [1] "Processing Entrez Gene ID:4501"
## [1] "Processing Entrez Gene ID:79937"
## [1] "Processing Entrez Gene ID:8804"
## [1] "Processing Entrez Gene ID:64981"
## [1] "Processing Entrez Gene ID:23090"
## [1] "Processing Entrez Gene ID:101928847"
## [1] "Processing Entrez Gene ID:202781"
```

```
## [1] "Processing Entrez Gene ID:5027"
## [1] "Processing Entrez Gene ID:100507557"
## [1] "Processing Entrez Gene ID:374969"
## [1] "Processing Entrez Gene ID:2149"
## [1] "Processing Entrez Gene ID:9603"
## [1] "Processing Entrez Gene ID:10063"
## [1] "Processing Entrez Gene ID:57546"
## [1] "Processing Entrez Gene ID:9314"
## [1] "Processing Entrez Gene ID:1809"
## [1] "Processing Entrez Gene ID:100289098"
## [1] "Processing Entrez Gene ID:1102"
## [1] "Processing Entrez Gene ID:1026"
## [1] "Processing Entrez Gene ID:9828"
## [1] "Processing Entrez Gene ID:10397"
## [1] "Processing Entrez Gene ID:125988"
## [1] "Processing Entrez Gene ID:8642"
## [1] "Processing Entrez Gene ID:387640"
## [1] "Processing Entrez Gene ID:8829"
## [1] "Processing Entrez Gene ID:23677"
## [1] "Processing Entrez Gene ID:7018"
## [1] "Processing Entrez Gene ID:645"
## [1] "Processing Entrez Gene ID:101927597"
## [1] "Processing Entrez Gene ID:4430"
## [1] "Processing Entrez Gene ID:1652"
## [1] "Processing Entrez Gene ID:7414"
## [1] "Processing Entrez Gene ID:54756"
## [1] "Processing Entrez Gene ID:1002"
## [1] "Processing Entrez Gene ID:8601"
## [1] "Processing Entrez Gene ID:793"
## [1] "Processing Entrez Gene ID:9388"
## [1] "Processing Entrez Gene ID:283131"
## [1] "Processing Entrez Gene ID:55714"
## [1] "Processing Entrez Gene ID:121551"
## [1] "Processing Entrez Gene ID:7349"
## [1] "Processing Entrez Gene ID:54212"
## [1] "Processing Entrez Gene ID:79630"
## [1] "Processing Entrez Gene ID:5064"
## [1] "Processing Entrez Gene ID:81849"
## [1] "Processing Entrez Gene ID:8537"
## [1] "Processing Entrez Gene ID:965"
## [1] "Processing Entrez Gene ID:8612"
## [1] "Processing Entrez Gene ID:339229"
## [1] "Processing Entrez Gene ID:80313"
## [1] "Processing Entrez Gene ID:94241"
## [1] "Processing Entrez Gene ID:22882"
## [1] "Processing Entrez Gene ID:23646"
## [1] "Processing Entrez Gene ID:57569"
## [1] "Processing Entrez Gene ID:729852"
## [1] "Processing Entrez Gene ID:51491"
## [1] "Processing Entrez Gene ID:3157"
## [1] "Processing Entrez Gene ID:89792"
## [1] "Processing Entrez Gene ID:80704"
## [1] "Processing Entrez Gene ID:23220"
## [1] "Processing Entrez Gene ID:6616"
```

```
## [1] "Processing Entrez Gene ID:6876"
## [1] "Processing Entrez Gene ID:4162"
## [1] "Processing Entrez Gene ID:5214"
## [1] "Processing Entrez Gene ID:56521"
## [1] "Processing Entrez Gene ID:6663"
## [1] "Processing Entrez Gene ID:91373"
## [1] "Processing Entrez Gene ID:8934"
## [1] "Processing Entrez Gene ID:55568"
## [1] "Processing Entrez Gene ID:10650"
## [1] "Processing Entrez Gene ID:205251"
## [1] "Processing Entrez Gene ID:8908"
## [1] "Processing Entrez Gene ID:54438"
## [1] "Processing Entrez Gene ID:100506311"
## [1] "Processing Entrez Gene ID:64409"
## [1] "Processing Entrez Gene ID:5274"
## [1] "Processing Entrez Gene ID:231"
## [1] "Processing Entrez Gene ID:26872"
## [1] "Processing Entrez Gene ID:7163"
## [1] "Processing Entrez Gene ID:10570"
## [1] "Processing Entrez Gene ID:1280"
## [1] "Processing Entrez Gene ID:2913"
## [1] "Processing Entrez Gene ID:145482"
## [1] "Processing Entrez Gene ID:220979"
## [1] "Processing Entrez Gene ID:64118"
## [1] "Processing Entrez Gene ID:6764"
## [1] "Processing Entrez Gene ID:1678"
## [1] "Processing Entrez Gene ID:94121"
## [1] "Processing Entrez Gene ID:10938"
## [1] "Processing Entrez Gene ID:1958"
## [1] "Processing Entrez Gene ID:196500"
## [1] "Processing Entrez Gene ID:3939"
## [1] "Processing Entrez Gene ID:6875"
## [1] "Processing Entrez Gene ID:1381"
## [1] "Processing Entrez Gene ID:81931"
## [1] "Processing Entrez Gene ID:3823"
## [1] "Processing Entrez Gene ID:10799"
## [1] "Processing Entrez Gene ID:145853"
## [1] "Processing Entrez Gene ID:11098"
## [1] "Processing Entrez Gene ID:4282"
## [1] "Processing Entrez Gene ID:1730"
## [1] "Processing Entrez Gene ID:1368"
## [1] "Processing Entrez Gene ID:147495"
## [1] "Processing Entrez Gene ID:3251"
## [1] "Processing Entrez Gene ID:4504"
## [1] "Processing Entrez Gene ID:27010"
## [1] "Processing Entrez Gene ID:221035"
## [1] "Processing Entrez Gene ID:26577"
## [1] "Processing Entrez Gene ID:5122"
## [1] "Processing Entrez Gene ID:26022"
## [1] "Processing Entrez Gene ID:10439"
## [1] "Processing Entrez Gene ID:57685"
## [1] "Processing Entrez Gene ID:80223"
## [1] "Processing Entrez Gene ID:100996740"
## [1] "Processing Entrez Gene ID:4782"
```

```
## [1] "Processing Entrez Gene ID:22915"
## [1] "Processing Entrez Gene ID:10418"
## [1] "Processing Entrez Gene ID:63979"
## [1] "Processing Entrez Gene ID:2890"
## [1] "Processing Entrez Gene ID:9361"
## [1] "Processing Entrez Gene ID:152330"
```

```
GeneFunction_Log2FC_2_0NA <- GetGeneFunctionByID(0,
                                                 2,
                                                 EntrezGeneList_OrderbyPval_Log2FC_2_0NA)
```

```
## [1] "Processing Entrez Gene ID:7020"
## [1] "Processing Entrez Gene ID:221895"
## [1] "Processing Entrez Gene ID:2297"
## [1] "Processing Entrez Gene ID:84282"
## [1] "Processing Entrez Gene ID:1051"
## [1] "Processing Entrez Gene ID:5321"
## [1] "Processing Entrez Gene ID:23643"
## [1] "Processing Entrez Gene ID:285016"
## [1] "Processing Entrez Gene ID:928"
## [1] "Processing Entrez Gene ID:4741"
## [1] "Processing Entrez Gene ID:55506"
## [1] "Processing Entrez Gene ID:4071"
## [1] "Processing Entrez Gene ID:205"
## [1] "Processing Entrez Gene ID:4826"
## [1] "Processing Entrez Gene ID:1846"
## [1] "Processing Entrez Gene ID:3397"
## [1] "Processing Entrez Gene ID:563"
## [1] "Processing Entrez Gene ID:29887"
## [1] "Processing Entrez Gene ID:2697"
## [1] "Processing Entrez Gene ID:100287482"
## [1] "Processing Entrez Gene ID:25805"
## [1] "Processing Entrez Gene ID:55244"
## [1] "Processing Entrez Gene ID:4747"
## [1] "Processing Entrez Gene ID:9518"
## [1] "Processing Entrez Gene ID:5925"
## [1] "Processing Entrez Gene ID:56963"
## [1] "Processing Entrez Gene ID:54898"
## [1] "Processing Entrez Gene ID:10409"
## [1] "Processing Entrez Gene ID:23090"
## [1] "Processing Entrez Gene ID:101928847"
## [1] "Processing Entrez Gene ID:5027"
## [1] "Processing Entrez Gene ID:9314"
## [1] "Processing Entrez Gene ID:100289098"
## [1] "Processing Entrez Gene ID:10397"
## [1] "Processing Entrez Gene ID:8642"
## [1] "Processing Entrez Gene ID:8829"
## [1] "Processing Entrez Gene ID:4430"
## [1] "Processing Entrez Gene ID:793"
## [1] "Processing Entrez Gene ID:9388"
## [1] "Processing Entrez Gene ID:121551"
## [1] "Processing Entrez Gene ID:54212"
## [1] "Processing Entrez Gene ID:79630"
## [1] "Processing Entrez Gene ID:81849"
```

```
## [1] "Processing Entrez Gene ID:8612"
## [1] "Processing Entrez Gene ID:94241"
## [1] "Processing Entrez Gene ID:23220"
## [1] "Processing Entrez Gene ID:6616"
## [1] "Processing Entrez Gene ID:6876"
## [1] "Processing Entrez Gene ID:6663"
## [1] "Processing Entrez Gene ID:100506311"
## [1] "Processing Entrez Gene ID:64409"
## [1] "Processing Entrez Gene ID:26872"
## [1] "Processing Entrez Gene ID:7163"
## [1] "Processing Entrez Gene ID:1280"
## [1] "Processing Entrez Gene ID:2913"
## [1] "Processing Entrez Gene ID:6875"
## [1] "Processing Entrez Gene ID:1381"
## [1] "Processing Entrez Gene ID:11098"
## [1] "Processing Entrez Gene ID:1730"
## [1] "Processing Entrez Gene ID:147495"
## [1] "Processing Entrez Gene ID:4504"
## [1] "Processing Entrez Gene ID:26577"
## [1] "Processing Entrez Gene ID:5122"
## [1] "Processing Entrez Gene ID:26022"
## [1] "Processing Entrez Gene ID:22915"
## [1] "Processing Entrez Gene ID:2890"
```

```
# Check the structure of the output
str(GeneFunction_Log2FC_1_0NA)
```

```
## 'data.frame':    249 obs. of  8 variables:
##  $ Entrez_ID            : chr  "7020" "51373" "10549" "221830" ...
##  $ Gene_symbol          : chr  "TFAP2A" "MRPS17" "PRDX4" "POLR1F" ...
##  $ Gene_description     : chr  "transcription factor AP-2 alpha" "mitochondrial ribosomal protein S1
##  $ Chromosome           : chr  "6" "7" "X" "7" ...
##  $ Map_location         : chr  "6p24.3" "7p11.2" "Xp22.11" "7p21.1" ...
##  $ Model_scientific_name: chr  "Homo sapiens" "Homo sapiens" "Homo sapiens" "Homo sapiens" ...
##  $ Model_common_name    : chr  "human" "human" "human" "human" ...
##  $ Gene_function        : chr  "The protein encoded by this gene is a transcription factor that binds
```

```
str(GeneFunction_Log2FC_2_0NA)
```

```
## 'data.frame':    66 obs. of  8 variables:
##  $ Entrez_ID            : chr  "7020" "221895" "2297" "84282" ...
##  $ Gene_symbol          : chr  "TFAP2A" "JAZF1" "FOXD1" "RNF135" ...
##  $ Gene_description     : chr  "transcription factor AP-2 alpha" "JAZF zinc finger 1" "forkhead box I
##  $ Chromosome           : chr  "6" "7" "5" "17" ...
##  $ Map_location         : chr  "6p24.3" "7p15.2-p15.1" "5q13.2" "17q11.2" ...
##  $ Model_scientific_name: chr  "Homo sapiens" "Homo sapiens" "Homo sapiens" "Homo sapiens" ...
##  $ Model_common_name    : chr  "human" "human" "human" "human" ...
##  $ Gene_function        : chr  "The protein encoded by this gene is a transcription factor that binds
```

17) Enrichment analysis (EA) with KEGG/GO/Reactome databases

The difference between Enrichment Analysis, GSEA, and fGSEA:

1. Enrichment Analysis (EA)

- *Definition:* Enrichment analysis refers to a collection of methods used to determine whether predefined gene sets (such as biological pathways or functional categories) are overrepresented in a list of genes of interest, often derived from differential expression studies.
- *Goal:* The goal is to identify if specific gene sets are significantly enriched in the gene list compared to a background set of genes. This can highlight biological processes or pathways that are impacted in the condition of interest.
- *Method:* Enrichment analysis typically involves comparing the gene list to a background gene set using various statistical tests. The most common methods include Fisher's exact test, hypergeometric test,... The analysis can be performed on a gene set level without requiring a pre-ranked list.
- *Flexibility:* This approach can be used with different types of gene set databases (e.g., KEGG, GO, Reactome). The choice of the background gene set can affect the results, and the method may not be sensitive to the ranking of genes.

2. Gene Set Enrichment Analysis (GSEA)

- *Definition:* GSEA is a computational method used to determine whether a predefined set of genes shows statistically significant differences between two biological states or conditions based on a ranked list of genes.
- *Goal:* The goal of GSEA is to identify whether specific gene sets are overrepresented or enriched in a ranked list of genes, which is derived from differential expression analysis. GSEA is particularly useful for analyzing genome-wide expression profiles to determine whether gene sets are preferentially located at the top or bottom of the ranked list.
- *Method:*
  - **Step 1: Calculation of Enrichment Score (ES)**

    GSEA calculates an enrichment score that reflects the degree to which a gene set is overrepresented at the extremes (top or bottom) of the ranked list. The score is derived from a running-sum statistic, which increases when encountering genes in the set and decreases otherwise, with the magnitude of the increment based on the gene's correlation with the phenotype.
  - **Step 2: Estimation of Significance Level of ES**

    Significance is estimated using permutation-based tests, where phenotype labels are permuted to create a null distribution for the ES. The empirical p-value is calculated based on this distribution.
  - **Step 3: Adjustment for Multiple Hypothesis Testing** The method normalizes the ES for each gene set to yield a normalized enrichment score (NES). The false discovery rate (FDR) is calculated to control the proportion of false positives.
- *Sensitivity:* GSEA is sensitive to the choice of background gene set and requires careful selection to avoid misleading results. It uses the entire ranked list of genes and does not rely on arbitrary cut-offs due to its rank-based. However, this sometimes results in gene sets/pathways that end up showing the strongest enrichment are driven by genes that would not necessarily meet the traditional FDR cut-off. These genes are normally referred to as **"leading genes"**.
- *Input:* GSEA requires a full, pre-ranked list of genes based on metrics such as p-values or Log2FC.

3.Fast Gene Set Enrichment Analysis (fGSEA)

- *Definition:* fGSEA is an adaptation of GSEA designed to provide faster and computationally efficient enrichment analysis using approximation techniques.

- *Goal:* The goal of fGSEA is to maintain the accuracy of GSEA while significantly reducing the computation time required for large datasets.
- *Method:* fGSEA employs fast approximation methods for calculating enrichment scores, which simplifies the process
- *Flexibility:* fGSEA is more robust to variations in the background gene set due to its approximation techniques but can still benefit from a well-chosen background for accurate interpretation. The output of fGSEA are leading genes, which is similar to GSEA.
- *Input:* fGSEA requires a full, pre-ranked list of genes based on metrics such as p-values or Log2FC.

Feature	Enrichment Analysis	GSEA	fGSEA
Requires ranked gene list	No	Yes	Yes
Statistical test	Various (e.g., Fisher's exact, hypergeometric)	Permutation-based	Approximation-based
Computational efficiency	Fast	Slow	Fast
Sensitivity to background	Less sensitive	Sensitive	Less sensitive

Ref for Overview of GSEA: https://www.pnas.org/doi/full/10.1073/pnas.0506580102

Ref for enrichment analysis with GO database using `clusterProfiler`: https://yulab-smu.top/biomedical-knowledge-mining-book/clusterprofiler-go.html

Ref for enrichment analysis with KEGG database using `clusterProfiler`: https://yulab-smu.top/biomedical-knowledge-mining-book/clusterprofiler-kegg.html

Ref for enrichment analysis with KEGG database using `clusterProfiler`: https://yulab-smu.top/biomedical-knowledge-mining-book/reactomepa.html

```
# Create a function to perform EA on common databases, including KEGG, GO, and
# Reactome
EAPerform_KEGG_GO_Reactome <- function(EntrezGeneList,
                                       metaOuput,
                                       pvalueCutOff,
                                       qvalueCutOff){

  # Perform EA with KEGG database
  KEGG_EA_result <- enrichKEGG(gene = EntrezGeneList,
                               # Organism code for Homo sapiens
                               organism = "hsa",
                               # Address the background gene set (output of meta-analysis)
                               universe = metaOuput,
                               # p-value cut off
                               pvalueCutoff = pvalueCutOff,
                               # Choose the method for FDR of GSEA (Benjamini-Hochberg)
                               pAdjustMethod = "BH",
                               # FDR cut off
                               qvalueCutoff = qvalueCutOff)

  # Perform EA with GO database
  GO_EA_result <- enrichGO(gene = EntrezGeneList,
```

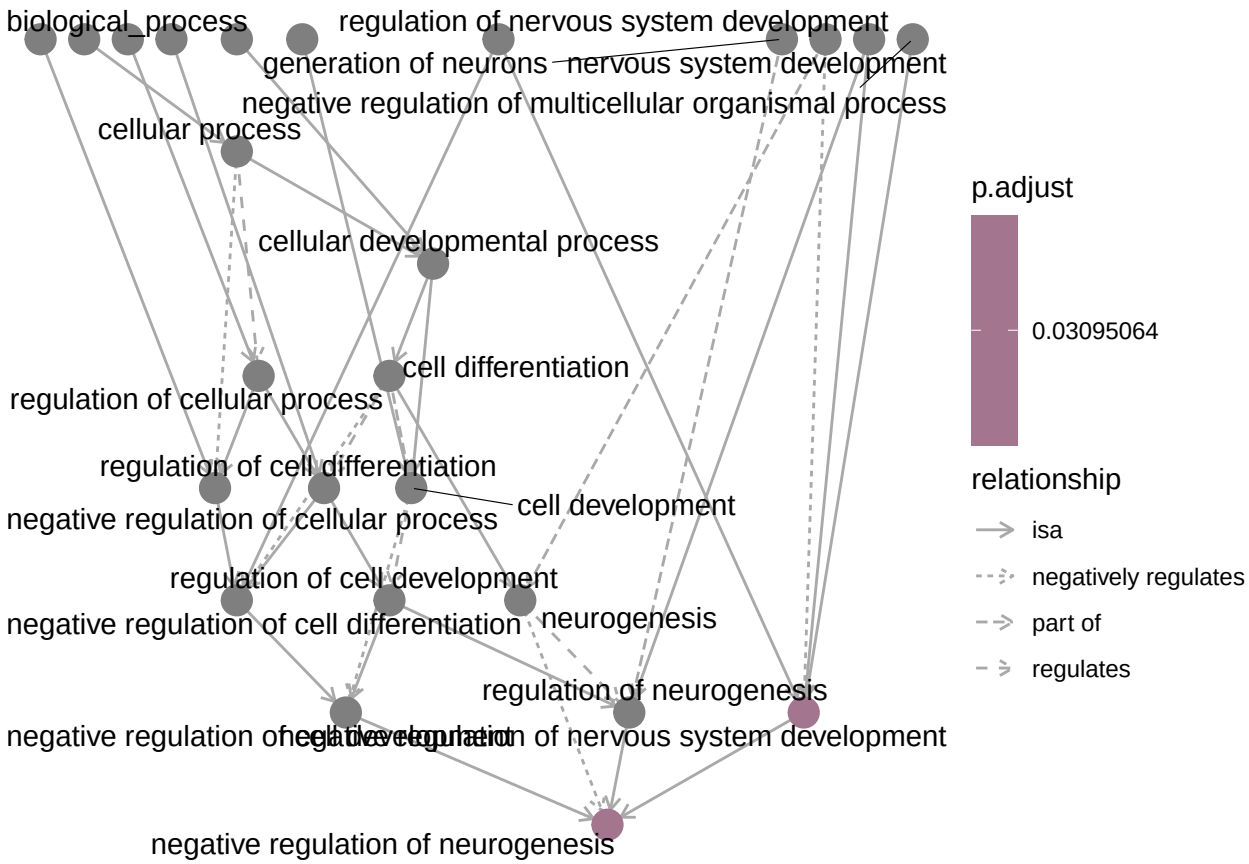
```
                            # Select database for Homo sapiens
                            OrgDb = org.Hs.eg.db,
                            # Ontology for biological process
                            # Other ontologies include "MF" for Molecular Function
                            # and "CC" for Cellular Component
                            ont = "BP",
                            # Address the background gene set (output of meta-analysis)
                            universe = metaOuput,
                            # p-value cut off
                            pvalueCutoff = pvalueCutOff,
                            # Choose the method for FDR of GSEA (Benjamini-Hochberg)
                            pAdjustMethod = "BH",
                            # FDR cut off
                            qvalueCutoff = qvalueCutOff)

  # Perform EA with Reactome database
  Reactome_EA_result <- enrichPathway(gene = EntrezGeneList,
                                      # Organism code for Homo sapiens
                                      organism = "human",
                                      # Address the background gene set (output of meta-analysis)
                                      universe = metaOuput,
                                      # p-value cut off
                                      pvalueCutoff = pvalueCutOff,
                                      # Choose the method for FDR of GSEA (Benjamini-Hochberg)
                                      pAdjustMethod = "BH",
                                      # FDR cut off
                                      qvalueCutoff = qvalueCutOff)

  # Return the list of results
  return(list(KEGG = KEGG_EA_result,
              GO = GO_EA_result,
              Reactome = Reactome_EA_result))
}
```

```
# Function use
EntrezGeneList_metaOutputFDR_ONA <- as.character(metaOutputFDR_Annotated_ONA$Human_EntrezGene.ID)

EA_Log2FC_1_ONA <- EAPerform_KEGG_GO_Reactome(EntrezGeneList_OrderbyPval_Log2FC_1_ONA,
                                              EntrezGeneList_metaOutputFDR_ONA,
                                              0.05,
                                              0.05)

EA_Log2FC_2_ONA <- EAPerform_KEGG_GO_Reactome(EntrezGeneList_OrderbyPval_Log2FC_2_ONA,
                                              EntrezGeneList_metaOutputFDR_ONA,
                                              0.05,
                                              0.05)
```

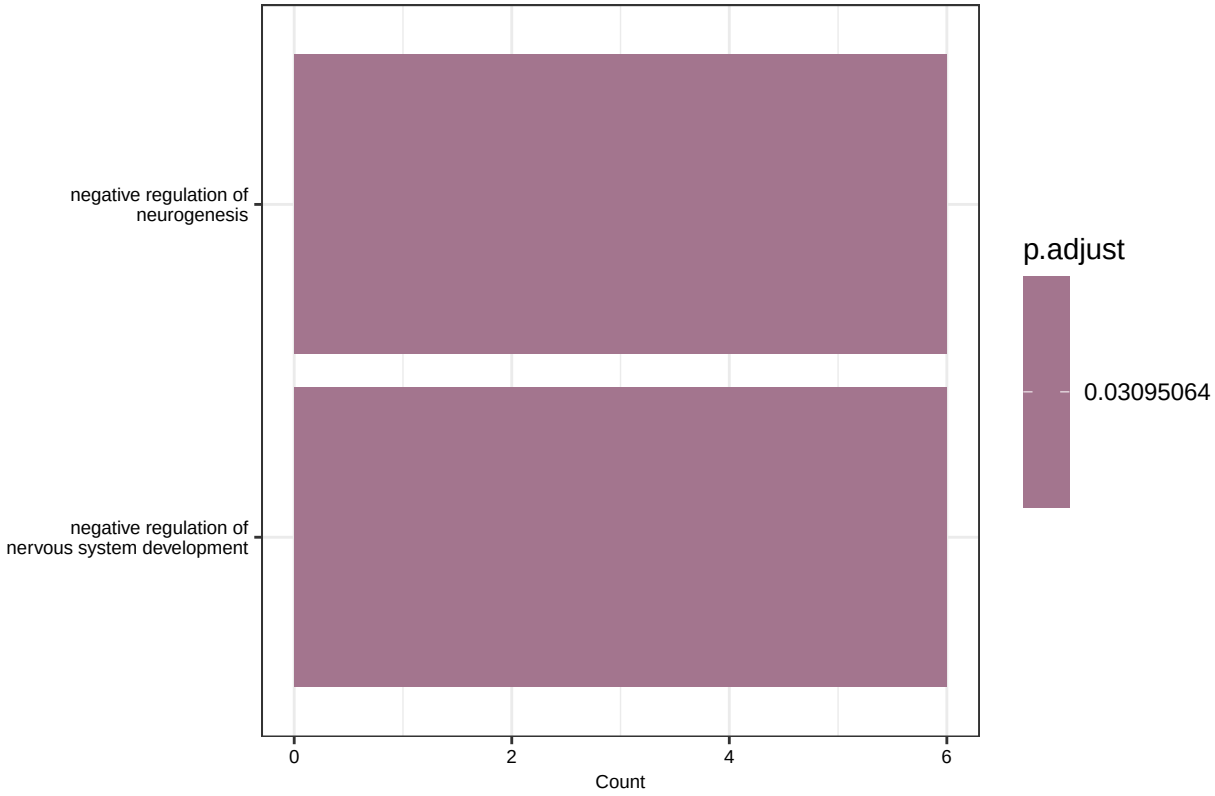
```
# Visualize the EA results with KEGG database

# There is only 1 enriched pathway on KEGG at loose p-value < 0.1 and loose FDR < 0.1
# with EA_Log2FC_1_ONA: HIF-1 signaling pathway (hsa04066)

# There is no enriched pathway on KEGG at loose p-value < 0.1 and loose FDR < 0.1
```

```r
# with EA_Log2FC_2_ONA

# There is no enriched pathway on KEGG at p-value < 0.05 and FDR < 0.05 with EA_Log2FC_1_ONA

# There is no enriched pathway on KEGG at p-value < 0.05 and FDR < 0.05 with EA_Log2FC_2_ONA

# Visualize individual enriched pathways

# Filter genes with FDR < 0.05 in genes of meta-analysis output with |Log2FC|   1
KEGG_metaOutputFDR_Log2FC_1_ONA_FDR_0.05 <- metaOutputFDR_OrderbyPval_Log2FC_1_ONA %>%
  filter(FDR < 0.05)

# Prepare the named vector
# Names = Mouse_EntrezGene.ID | Values  = Log2FC estimate
KEGGgenedata_metaOutputFDR_Log2FC_1_ONA <- setNames(
  KEGG_metaOutputFDR_Log2FC_1_ONA_FDR_0.05$Log2FC_estimate,
  KEGG_metaOutputFDR_Log2FC_1_ONA_FDR_0.05$Human_EntrezGene.ID
)

# Generate pathway view for "hsa04066" with genes of FDR < 0.05 and |Log2FC|   1
hsa04066 <- pathview(gene.data = KEGGgenedata_metaOutputFDR_Log2FC_1_ONA,
                     pathway.id = "hsa04066",
                     species = "hsa",
                     limit = list(gene = c(min(KEGGgenedata_metaOutputFDR_Log2FC_1_ONA),
                                           max(KEGGgenedata_metaOutputFDR_Log2FC_1_ONA)),
                                  cpd = 1))
```

```r
# Visualize the EA results with GO database

# There is no enriched pathway on GO at p-value < 0.05 and FDR < 0.05 with EA_Log2FC_1_ONA

# There are 2 enriched pathway on GO at p-value < 0.05 and FDR < 0.05 with EA_Log2FC_2_ONA:
# Negative regulation of neurogenesis (GO:0050768) and Negative regulation of nervous
# system development (GO:0051961)

# Visualize the results with directed acyclic graph

# png("EA_Log2FC_2_ONA_GO_Network.png", 8, 5, "in", res = 300)
goplot(EA_Log2FC_2_ONA$GO)
```




```
# dev.off()

# 2 pathways using barplot presentation for genes with FDR < 0.05 and with |Log2FC|   2

# png("EA_Log2FC_2_ONA_GO.png", 8, 5, "in", res = 300)
barplot(EA_Log2FC_2_ONA$GO, showCategory = 10, font.size = 7)
```




```
# dev.off()
```

```
# There is no enriched pathway on Reactome at p-value < 0.05 and FDR < 0.05 with EA_Log2FC_1_ONA

# There is no enriched pathway on Reactome at p-value < 0.05 and FDR < 0.05 with EA_Log2FC_2_ONA
```

```
# Transform the matched pathways results into dataframes and export into the working
# directory

EA_Log2FC_2_ONA_GO <- as.data.frame(EA_Log2FC_2_ONA$GO)
row.names(EA_Log2FC_2_ONA_GO) <- NULL
write.csv(EA_Log2FC_2_ONA_GO, "EA_Log2FC_2_ONA_GO.csv")
```

18) Fast Gene Set Enrichment Analysis (fGSEA) with Brain.GMT database [Need to fix]

The Brain.GMT database, developed by Hagenauer et al. (2024), is designed to improve the interpretation of brain-derived differential expression results, especially in studies focused on neuropsychiatric disorders. The Brain.GMT can be used within common pipelines (GSEA, limma, edgeR) to interpret results from 3 species of rat, mouse, human. The database includes 918 gene sets derived from various sources, each curated to be relevant to brain functions, brain cell types, co-expression networks, and regional gene expression signatures.

Gene sets included in the Brain.GMT:

1. MSigDB Gene SetS

- Curated Gene Sets: 158 gene sets related to the nervous system.

- Cell Type Signature Gene Sets: 211 gene sets related to the nervous system and blood.

2. BrainInABlender

- 39 gene sets related to cell type-enriched expression, particularly in the cortex.

3. DropViz

- 13 gene sets focused on hippocampal expression, and 12 gene sets for the nucleus accumbens.

4. HippoSeq

- 14 gene sets related to regional enriched expression in the hippocampus.

5. Coexpression Analyses

- 55 gene sets derived from coexpression networks in the hippocampus.

6. Gene Weaver

- 33 gene sets for the hippocampus and 6 gene sets for the nucleus accumbens related to stress, environmental enrichment, affective behavior, and mood disorder.

7. Gemma

- 29 gene sets derived from reanalysis pipelines focusing on stress, environmental enrichment, affective behavior, and mood disorder in the nucleus accumbens.

Ref for Brain.GMT: https://doi.org/10.1016/j.mex.2024.102788

Ref for orginal GSEA: https://doi.org/10.1073/pnas.0506580102

Although the Brain.GMT focuses on brain-related genes, the impact of viral infection might involve specific pathways or immune responses that are not well-represented in this database.

Use all of the meta-analysis output and ordered it by Log2FC values (pre-ranked lists)

```
# fGSEA uses the entirety of the gene list but need to pre-ranked with either p-values
# or Log2FC. For Brain.GMT, we use gene list pre-ranked by Log2FC.

# Retrieve gene symbols and gene functions using Entrez Gene ID for fGSEA
# (This takes approximately 5 hours for 15952 genes so please import the "GeneFunction_metaOutputFDR_ON
# csv file or load the RData file to save time)
GeneFunction_metaOutputFDR_ONA <- GetGeneFunctionByID(0,
                                                      0,
                                                      EntrezGeneList_metaOutputFDR_ONA)

# Export the gene symbols and gene functions into the working directory
write.csv(GeneFunction_metaOutputFDR_ONA, "GeneFunction_metaOutputFDR_ONA.csv")

# Rename the "Entrez_ID" column to "Human_EntrezGene.ID"
colnames(GeneFunction_metaOutputFDR_ONA)[colnames(GeneFunction_metaOutputFDR_ONA) == "Entrez_ID"] <- "Hu
```

```r
# Merge gene function data with meta-analysis output based on Human_EntrezGene.ID
BrainGMT_genesummary_metaOutputFDR_ONA <- left_join(metaOutputFDR_Annotated_ONA,
                                                    GeneFunction_metaOutputFDR_ONA,
                                                    by = "Human_EntrezGene.ID")

# Convert gene symbols to normal text with the first letter capitalized
BrainGMT_genesummary_metaOutputFDR_ONA$Gene_symbol <- str_to_title(BrainGMT_genesummary_metaOutputFDR_O


# Create a named vector from meta-analysis output
# Names = Human gene symbol | Values = Log2FC estimate
BrainGMTgenedata_metaOutputFDR_ONA <- setNames(
  BrainGMT_genesummary_metaOutputFDR_ONA$Log2FC_estimate,
  BrainGMT_genesummary_metaOutputFDR_ONA$Gene_symbol
)

# Remove genes with NA or duplicated names
BrainGMTgenedata_metaOutputFDR_ONA_cleaned <- BrainGMTgenedata_metaOutputFDR_ONA %>%
  # Convert to dataframe for easier manipulation
  enframe(name = "gene_name", value = "Log2FC_estimate") %>%
  # Remove rows with NA names
  filter(!is.na(gene_name)) %>%
  # Remove rows with finite Log2FC values
  filter(is.finite(Log2FC_estimate)) %>%
  # Remove duplicated names, keeping the first instance
  distinct(gene_name, .keep_all = TRUE) %>%
  # Convert back to named vector
  deframe()

# Order the vector by Log2FC in ascending order
BrainGMTgenedata_metaOutputFDR_OrderedbyLog2FC_ONA <- BrainGMTgenedata_metaOutputFDR_ONA_cleaned[order(

# Import the Brain.GMT for the specific species of interest
# Ignore the warning about incomplete line
BrainGMT_Human <- gmtPathways("BrainGMTv2_wGO_HumanOrthologs.gmt.txt")

# Perform fast fGSEA with Brain.GMT
fGSEA_ONA_BrainGMT <- fgseaSimple(BrainGMT_Human,
                                  BrainGMTgenedata_metaOutputFDR_OrderedbyLog2FC_ONA,
                                  # Number of permutations for the GSEA
                                  nperm = 10000,
                                  # Minimum size of gene sets to consider
                                  minSize = 10,
                                  # Maximum size of gene sets to consider
                                  maxSize = 1000)

# Converts the list of leading edge genes (genes contributing to enrichment) in
# the fGSEA results to a comma-separated string
fGSEA_ONA_BrainGMT$leadingEdge <- vapply(fGSEA_ONA_BrainGMT$leadingEdge,
                                         paste,
                                         collapse= ",",
                                         character(1L))
```

```r
# Order the fGSEA results based on adjusted p-values and count the number of leading
# edges in each pathways
fGSEA_ONA_BrainGMT <- fGSEA_ONA_BrainGMT %>%
  arrange(padj) %>%
  mutate(Num_leadingEdge = sapply(strsplit(leadingEdge, ","), length))

# Export the matched pathways into the working directory
write.csv(fGSEA_ONA_BrainGMT, "fGSEA_ONA_BrainGMT.csv")
```

Interpretation of fGSEA

1. padj (Adjusted p-value)

- *Definition:* This is the p-value adjusted for multiple comparisons for the False Discovery Rate (FDR) using the family-wise error rate method. It controls for the proportion of false positives among the significant results.
- *Explanation:* A low padj value (e.g., < 0.05) indicates that the enrichment of a gene set is statistically significant after correction for multiple testing, reducing the likelihood of false positives.

2. ES (Enrichment Score)

- *Definition:* The Enrichment Score (ES) represents the degree to which a gene set is overrepresented at the top or bottom of a ranked list of genes. It is calculated by walking down the list of pre-ranked genes and increasing a running-sum statistic when a gene in the gene set is encountered and decreasing it when a gene not in the gene set is encountered.
- *Explanation:*
  - A positive ES suggests that the gene set is enriched among genes that are upregulated (i.e., genes at the top of the ranked list).
  - A negative ES indicates that the gene set is enriched among genes that are downregulated (i.e., genes at the bottom of the ranked list).

3. NES (Normalized Enrichment Score)

- *Definition:* The Normalized Enrichment Score (NES) is the enrichment score (ES) that has been normalized across all gene sets to account for differences in gene set size. It allows for comparison between gene sets of different sizes.
- *Explanation:*
  - NES adjusts the ES so that it can be compared across gene sets of different sizes:
    - Positive NES: Indicates the gene set is upregulated in your data.
    - Negative NES: Indicates the gene set is downregulated in your data.

4. leadingEdge

- *Definition:* The leading-edge subset is the subset of genes in the gene set that contribute most to the enrichment score. These are the genes that appear before the peak enrichment score is reached and thus drive the observed enrichment.
- *Explanation:* This variable contains the "core" genes responsible for the enrichment signal. Understanding which genes are in the leading edge can provide insights into which genes are most responsible for the phenotype of interest.

5. nMoreExtreme

- *Definition:* nMoreExtreme represents the number of permutations where the observed ES was more extreme (either higher or lower) than the ES obtained from random permutations of the gene set.
- *Explanation:* This variable helps estimate the empirical p-value for the observed ES. A lower nMoreExtreme value indicates that few random permutations achieved an ES as extreme as the observed one, suggesting that the observed enrichment is unlikely to be due to chance.

6. Size

- *Definition:* Size indicates the number of genes in the gene set that overlap with the pre-ranked list of genes used in the enrichment analysis.
- *Explanation:* This is the effective size of the gene set in the context of the analysis. Larger gene sets might have more opportunities to show enrichment, but smaller gene sets can also provide strong signals if their genes are strongly associated with the phenotype of interest.