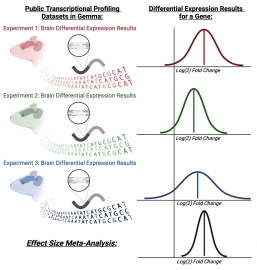
Sep 18, 2024

Brain Data Alchemy Project: Meta-Analysis of Preprocessed Public Transcriptional Profiling Data in the Gemma Database (v.2023)



Forked from Brain Data Alchemy Project: Meta-Analysis of Re-Analyzed Public Transcriptional Profiling Data in the Gemma Database

DOI
dx.doi.org/10.17504/protocols.io.x54v925d4l3e/v1

Megan Hagenauer[1], Elizabeth Flandreau[2], Toni Duan[3], Anne Bader[3], Christabel Mclain[4], Trevonn Gyles[4], Ashlee Lewis[5], Duy Manh Nguyen[3]

[1]University of Michigan; [2]Grand Valley State University; [3]Grinnell College; [4]Icahn School of Medicine at Mount Sinai; [5]University of Detroit Mercy

Megan Hagenauer
University of Michigan

OPEN ACCESS

DOI: dx.doi.org/10.17504/protocols.io.x54v925d4l3e/v1

Protocol Citation: Megan Hagenauer, Elizabeth Flandreau, Toni Duan, Anne Bader, Christabel Mclain, Trevonn Gyles, Ashlee Lewis, Duy Manh Nguyen 2024. Brain Data Alchemy Project: Meta-Analysis of Preprocessed Public Transcriptional Profiling Data in the Gemma Database (v.2023). **protocols.io https://dx.doi.org/10.17504/protocols.io.x54v925d4l3e/v1**

Protocol status: Working
This protocol was used for the 2023 cohort of the Brain Data Alchemy Project.

Created: August 21, 2024

Last Modified: September 18, 2024

Protocol Integer ID: 106122

Keywords: Transcriptional Profiling, RNA-Seq, Microarray, Gemma, Meta-Analysis, Gene Expression, Summer Program, R Programming, PRISMA, Systematic Meta-Analysis

Funders Acknowledgement:
Hope for Depression Research Foundation (HDRF)
Grant ID: Data Center
Pritzker Neuropsychiatric Disorders Research Foundation
Grant ID: NA
National Institute on Drug Abuse (NIDA)
Grant ID: U01DA043098
Grinnell College Center for Careers, Life, and Service
Grant ID: NA
International Brain Research Organization (IBRO)
Grant ID: Travel Grant

Disclaimer

This protocol provides meta-analysis guidelines and sample code, but not a full reproducible analysis pipeline and coding environment. It was meant to be adapted to the needs of each individual project.

Abstract

Background: Over the past two decades, transcriptional profiling has become an increasingly common tool for investigating the effects of diseases and experimental manipulations on the nervous system. Within transcriptional profiling experiments, microarray or sequencing technologies are used to measure the relative amount of RNA transcript for each of the thousands of genes expressed in a sample. The primary objective of these experiments is to identify genes that are differentially expressed in response to conditions of interest. However, transcriptional profiling experiments have traditionally been conducted with small sample sizes due to expense (e.g., n=3-10/group), resulting in low statistical power. Due to low power, these experiments are prone to capturing large technical artifacts and false positives rather than smaller biological effects of interest.
Methods: To address this issue, we developed a 10-week summer research program (*The Brain Data Alchemy Project*) that guides participants through the process of performing a meta-analysis of differential expression effect sizes (Log2 Fold Change or Log2FC) extracted from publicly available transcriptional profiling datasets. To conduct our meta-analyses, we leverage the efforts of the Gemma project, which has curated, preprocessed, and re-analyzed over 19,000 publicly available datasets (https://gemma.msl.ubc.ca/home.html). Participants learn the fundamental principles of systematic review and R programming to conduct the dataset search, result extraction, and whole transcriptome meta-analysis.
Version Information: This protocol outlines the methods used during the second pilot year of the program in 2023. These methods differ from the 2022 methods (**dx.doi.org/10.17504/protocols.io.j8nlk84jxl5r/v1**): The 2023 project methods import the preprocessed gene expression data and metadata from Gemma, whereas the 2022 project methods import the differential expression results.

Image Attribution

Graphical abstract created with BioRender.com

Guidelines

Any questions regarding this protocol should be directed to Dr. Megan Hagenauer (University of Michigan: hagenaue@umich.edu).

This protocol outlines the meta-analysis methods recommended to participants within the Brain Data Alchemy Program during the second pilot year of the program (2023). These methods were updated for later years - for updates, see the associated forks for this protocol and for the protocol from the first pilot year of the program (2022: **dx.doi.org/10.17504/protocols.io.j8nlk84jxl5r/v1**).

Materials

A computer and internet access.

Safety warnings

N/A

Project Preparation: Environment Set-Up

1 Install R and RStudio

Note

R is a programming language for statistical computing and data visualization. We will be running our analyses using R.
RStudio is an integrated development environment for R (i.e., a software interface that facilitates working in R). We will use the RStudio interface to make writing and editing code easier.

1.1 Install R

Software

R programming language	NAME
The R Foundation	DEVELOPER
Comprehensive R Archive Network	SOURCE LINK

Note

Go to: https://cran.r-project.org
- Choose the precompiled binary distributions of the base system and contributed packages
- If there is an option, use the CRAN “mirror” nearest to you to minimize network load.
- This protocol was designed using the version of R (R v.4.3.0) available on 06/01/2023. It is likely to work using other versions.

1.2 Install R Studio

Software

NAME	DEVELOPER
R Studio Desktop	The R Studio, Inc.

Note

Go to: **https://posit.co/download/rstudio-desktop/**
- Install the appropriate version for your operating system
- This protocol was designed using the version of RStudio (v.2023.03.2) available on 06/01/2023. It is likely to work using other versions.

2 Follow the Brain Data Alchemy code repository on Github

Note

Github is a website (and desktop app) that enables easy code sharing, collaboration, and version control. We use Github in its most basic format for this project (easy code sharing).

2.1 Sign up for a free account on Github

Note

Go to: **https://github.com/**

2.2 Follow the Brain Data Alchemy code repository

Note

Go to: https://github.com/hagenaue/BrainDataAlchemy
In the upper right hand corner, click on either "watch" or "star" so that you can easily navigate back. "Watch" means that you will receive updates whenever anything in the repository is changed, whereas "Star" means that you have saved the repository to a list (sort of like a bookmark).

2.3 Optional: Install GitHub Desktop

Software

NAME	DEVELOPER
GitHub Desktop	GitHub

Note

If you want to use Github more fully (for actual version control), GitHub Desktop can make that easier:
Go to: **https://desktop.github.com/**

- Install the version of Github desktop that is compatible with your operating system

Dataset Identification: Pre-specification of Search Strategy

3 We will first plan the search strategy that we will use to identify useful datasets for the meta-analysis.

Note

In order to avoid biasing our meta-analysis, it is important that we pre-specify the search strategy that we will use to identify our datasets *before we learn anything about the results from that search*.
Full guidelines about how to conduct a systematic meta-analysis can be found here: **https://www.prisma-statement.org/**

CITATION

Moher D, Liberati A, Tetzlaff J, Altman DG, PRISMA Group (2009). Preferred reporting items for systematic reviews and meta-analyses: the PRISMA statement..
LINK
https://doi.org/10.1136/bmj.b2535

CITATION

Liberati A, Altman DG, Tetzlaff J, Mulrow C, Gøtzsche PC, Ioannidis JP, Clarke M, Devereaux PJ, Kleijnen J, Moher D (2009). The PRISMA statement for reporting systematic reviews and meta-analyses of studies that evaluate healthcare interventions: explanation and elaboration..
LINK
https://doi.org/10.1136/bmj.b2700

3.1 The scope of our meta-analysis search is pre-specified below. The scope will be the same for each meta-analysis conducted using the Brain Data Alchemy analysis pipeline.

Note

For our meta-analysis, we will only be considering:
1) Publicly-available transcriptional profiling datasets (gene expression microarray or RNA-Sequencing) from laboratory rodents (rats, mice).
2) We will only use datasets that profiled RNA extracted from bulk tissue dissections (i.e., not from a particular cell type or small tissue subregion).
3) We will only use datasets that profiled either the full transcriptome or a large representation of the full transcriptome. We will not use datasets targeted at a particular subpopulation of transcripts (e.g., Chip-Seq, miRNA, TRAP-Seq, etc).

3.2 The information source will be the same for all meta-analyses: The Gemma Database.

Note

We will be leveraging the efforts of the Gemma project to curate, preprocess, and re-analyze publicly-available transcriptional profiling datasets:
https://gemma.msl.ubc.ca/home.html
The Gemma database contains >19,000 re-analyzed transcriptional profiling datasets.

Gemma performs the following data preprocessing and analysis steps:
1) Probe alignment or read mapping is redone whenever a new genome assembly is available.
2) Identification and removal of outlier samples.
3) Filtering out genes (rows) with minimal variance (either zero variance or <70% distinct values).
4) Batch correction: When confounded, batches are split into separate analyses.
5) Manual curation of common issues.
6) Differential expression output from omnibus (full dataset) tests examining the effects of the primary variables of interest as well as individual statistical contrasts.

For the current (2023) version of the Brain Data Alchemy Project, we will only be directly building upon some of Gemma's analysis efforts (steps #1, #5), but will be reapplying analysis methods that are similar to those used by Gemma for steps #2-4 & #6.

CITATION

Zoubarev A, Hamer KM, Keshav KD, McCarthy EL, Santos JR, Van Rossum T, McDonald C, Hall A, Wan X, Lim R, Gillis J, Pavlidis P (2012). Gemma: a resource for the reuse, sharing and meta-analysis of expression profiling data..

LINK

https://doi.org/10.1093/bioinformatics/bts430

CITATION

Lim N, Tesar S, Belmadani M, Poirier-Morency G, Mancarci BO, Sicherman J, Jacobson M, Leong J, Tan P, Pavlidis P (2021). Curation of over 10 000 transcriptomic studies to enable data reuse..

LINK

https://doi.org/10.1093/database/baab006

3.3 Choose your meta-analysis topic and primary research question.

Note

Not all topics or research questions have been studied extensively using brain transcriptional profiling data. At this point, before pursuing the meta-analysis further, it would be helpful to have a second researcher who is not directly involved in the project (e.g., research mentor) quickly double-check whether there appears to be datasets in the Gemma database related to the chosen topic.

This quick, initial review should only use a handful of the most basic search terms for the topic. It is often advisable to wait to specify the tissue of interest (e.g., brain region) until later after you have determined how many the available datasets will survive the initial inclusion/exclusion criteria (protocol section 6).

The 2023 cohort of the Brain Data Alchemy Project chose to examine the effects of chronic pain, early life stress, chronic stress, and individual factors modulating chronic stress responses (sex, stress susceptibility vs. stress resilience).

3.4 Pre-specify the search terms for your topic.

Note

The search terms should be thoroughly brainstormed to catch any dataset related to the meta-analysis topic. e.g.,
- Topic
- Synonyms for Topic
- Abbreviations for Topic
- Subtypes of Topic
- Umbrella Categories for Topic

Since this is a critical step, feedback on search terms is important. For the 2023 cohort of the Brain Data Alchemy Project, these terms were reviewed by the mentor and whole cohort.

3.5 Examine the search terms more carefully and specify the formal search syntax.

Note

We will be conducting our dataset search using an R coding package that accesses the API (application programming interface) for the Gemma database. Gemma's API uses formal search syntax (i.e., search term construction). This syntax is not as flexible (or smart!) as a Google search, and differs in some ways from more conventional databases such as PubMed or Embase.

Examine your search terms carefully to define your formal syntax:

- Are there a variety of potential endings for any of the terms? (e.g., "stress" vs. "stressful" vs. "stressed" vs. "stressing") If so, you may be able to just include the shared component of the term with an asterisk to indicate term expansion (e.g., "stress*")
- Are any of these terms likely to identify datasets that are not related to your topic? If so, you might want to rule them out with NOT, e.g., "stress* NOT distress")
- If you have a term that is a phrase that includes more than one word, you may need to add quotations around that phrase if the exact phrase is desired/necessary ("chronic social defeat stress").
- If you need two or more words in your phrase to be present, but the exact order/spacing is not important, you can use AND (e.g., "chronic AND social AND stress") or you can just include the words in parentheses "(chronic social stress)"
- Independent search terms can be combined into a longer list using OR (e.g., "stress* OR advers* ")

Defining search syntax is also a critical step, and feedback is important. For the 2023 cohort of the Brain Data Alchemy Project, these terms were reviewed by the mentor.

Command

Example R code: Search syntax for Gemma's API

```
SearchTerms<-"(dentate gyrus stress*) OR
(hippocamp* stress*) OR
(Ammon's Horn stress*)"
```

Note

Note that the SearchTerms object is a single string value containing the full search term syntax (i.e., the full syntax is wrapped in quotations - " ").

Dataset Identification: Coding Steps

4 Dataset Identification: Coding Steps. This coding is all performed within the R environment using Rstudio.

Note

Example code for these steps can be found here: https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_GemmaDatasets%20/2023_Antidepressants_New_Code_forGemmaSearch.R

4.1 Install the R package Devtools using: *install.packages("devtools")*

Note

- This protocol was designed using the version of Devtools (v.2.4.4) available on 06/01/2023. It is likely to work using other versions.

Software

R Package devtools NAME

Wickham, Hester, Chan, and Bryan DEVELOPER

http://cran.r-project.org/web/packages/devtools/index.html SOURCE LINK

4.2 Install the gemma.R package.

Software

Gemma.R

NAME

Javier Castillo-Arnemann, Jordan Sicherman, Ogan Mancarci, Guillaume Poirier-Morency

DEVELOPER

http://github.com/PavlidisLab/gemma.R

SOURCE LINK

Command

Example R Code: Install gemma.R

```
if(!requireNamespace("devtools", quietly=T)){
  install.packages("devtools")
}

devtools:: install_github("PavlidisLab/gemma.R", force=T)
```

Note

- This protocol was designed using the version of gemma.R (v.0.99.41) available on 06/01/2023. *It is unlikely to work using later versions.*
- There is some nice documentation for Gemma.R: https://pavlidislab.github.io/gemma.R/articles/gemma.R.html

4.3 Use the function *search_datasets()* to determine how many transcriptional profiling datasets within the Gemma database contain your prespecified keywords.

Command

Example R code: Determining how many datasets in Gemma contain your search terms

```
result <- gemma.R ::search_datasets(SearchTerms, taxon =
'mouse',limit = 1)

print(attributes(result)$totalElements)
#[1] 111

result <- gemma.R ::search_datasets(SearchTerms, taxon = 'rat',limit
= 1)

print(attributes(result)$totalElements)
#[1] 36
```

Note

- SearchTerms is the object created by the code in step 3.5. It is a single string value containing the full search term syntax (i.e., the full syntax is wrapped in quotations - " ").
- Taxon specifies the species for the search (e.g., 'mouse' or 'rat').
- For this purpose (counting datasets), limit is set as 1
- print(attributes(result)$total elements) shows the number of datasets found for your search - for the first example we found 111 datasets, for the second example we found 36.

4.4 Use the function *search_datasets()* to locate transcriptional profiling datasets within the Gemma database that contain your prespecified keywords.

Command

Example R code: Identifying datasets from mice and rats with our search terms

```
#Identifying datasets that fit our search terms in mice:

#First, determine how many datasets from mice fit our search:

result <- gemma.R ::search_datasets(SearchTerms, taxon =
'mouse',limit = 1)

print(attributes(result)$totalElements)
#[1]  111
#There are 111 datasets that fit this search.

#Gemma.R can only download 100 dataset metadata records at a time, so
if we have more than 100 results, we will need to perform the search
interatively, with the number of iterations determined by the total
number of datasets divided by 100:

SearchIterations<-floor(attributes(result)$totalElements/100)

#We can run the search for the first 100 records using the basic
syntax:

TempResult<-gemma.R ::search_datasets(SearchTerms, taxon =
'mouse',limit = 100)

#Then we run a loop to gather the next sets of 100 records, and add
those records into the same dataframe as additional rows:

for(i in c(1:SearchIterations)){
  TempResult<-rbind.data.frame(TempResult,(gemma.R
::search_datasets(SearchTerms, taxon = 'mouse',limit = 100,
offset=i*100)))
}

#I recommend renaming the object afterwards to make it easier to keep
track of things later:
results_Stress_Hippocampus_Mice<-TempResult

#Cleaning up our workspace of objects that aren't needed anymore:
rm(result, TempResult, SearchIterations)
```

```
rm(result, TempResult, SearchIterations)

############

#Identifying datasets with our search terms in rats:

result <- gemma.R ::search_datasets(SearchTerms, taxon = 'rat',limit
= 1)

print(attributes(result)$totalElements)
#[1] 36
#There are less than 100 datasets that fit this search, so we do not
need to iterate our search.

TempResult<-gemma.R ::search_datasets(SearchTerms, taxon =
'rat',limit = 100)

#I renamed the object containing all of the rat datasets to make it
easier to keep track of later:
results_Stress_Hippocampus_Rat<-TempResult

#Cleaning up our workspace of objects that aren't needed anymore:
rm(result, TempResult, SearchIterations)
```

Note

- The first argument in the *search_datasets()* function is your search term syntax. The full syntax for the search terms will need to be placed in quotations as if it were a single string value.
- The argument "taxon" allows you to filter by species.
- Gemma.R can't extract more than 100 records at a time (the "limit")
- If we have more than 100 results, we will need to perform the search interatively, with the number of iterations determined by the total number of datasets divided by 100.
-If we run the search for the first 100 records using the basic syntax, we can then run a for loop to gather the next sets of 100 records, and add those records into the same data frame as additional rows.

4.5 Compile the metadata records from these identified datasets into a single data frame.

Command

Example R code: Combining search results into a single data frame

```
combined<-rbind.data.frame(results_Stress_Hippocampus_Mice,
results_Stress_Hippocampus_Rat)
```

4.6 Eliminate any duplicated records in the data frame.

Command

Example R code: Removing duplicated metadata records

```
unique_combined<- unique(combined)
```

Initial Dataset Filtering Using MetaData

5 The metadata records should be filtered using pre-specified inclusion and exclusion criteria.

5.1 Exclude any datasets that Gemma has marked as "troubled".

Command

Example R code: Excluding datasets marked as "troubled"

```
#How many datasets are marked by Gemma as troubled?
table(unique_combined$experiment.Troubled)

#Excluding datasets marked as troubled:
unique_combined<-
unique_combined[unique_combined$experiment.Troubled==FALSE,]
```

5.2 Export the data frame containing the remaining metadata records from R as a .csv file. This data frame of metadata can then be more easily explored and annotated in a spreadsheet software program like Microsoft Excel or Google Sheets.

Command

Example R code: Writing out metadata records as a .csv file

```
write.csv(unique_combined, "Metadata_forDatasets_toReview.csv")
```

5.3 The titles, abstracts, and metadata for the datasets should then be scanned to determine which datasets should be excluded based on our pre-specified criteria.

Note

Pre-specified exclusion criteria:
1) Brain tissue was not collected.
2) The experimental manipulation was unrelated to the meta-analysis topic.
3) The brain tissue used in the experiment was not from a bulk dissection, and instead represented a particular cell type or small subregion.
4) The experiment targeted a particular subpopulation of transcripts (e.g., Chip-Seq, miRNA, TRAP-Seq, etc.).
5) The experiment used subjects from a developmental stage other than the timeframe of interest.
6) The metadata record on Gemma has critical issues, including duplication/overlap with another record or critical missing information (minimal methodological information due to no associated publication and a minimalist metadata record).

5.4 This is a critical step. Following this initial filtering by the individual experimenter, all inclusion/exclusion choices should be reviewed and approved by a second researcher (such as a research mentor).

Specification of Tissue of Interest (If Not Pre-Specified)

6 If a particular tissue or brain region of interest (ROI) had not been pre-specified in the search criteria, the research question should now be narrowed to a particular tissue or brain ROI based on dataset availability, and the dataset records filtered accordingly.

Note

At this point, since the titles and abstracts for the datasets have already been viewed, it is important to run decisions about tissue/ROI definition past a second researcher who is less invested in the project (e.g., a research mentor) to guard against bias. These decisions may include whether to include datasets from samples that only focus on a subregion of the larger ROI, or whether to include tissue cultures as well as fresh-collected tissue.

Secondary Dataset Filtering Using Detailed Methodological Review

7 The remaining dataset metadata records should now be subjected to a detailed review of accompanying experimental methods and available file formats.

Note

For this level of filtering, the publications associated with the datasets need to be referenced in addition to the metadata available on Gemma.

Only the methodological information in the publication and Gemma record should be used to determine study/dataset inclusion/exclusion, because we want our inclusion/exclusion criteria to be independent of the results reported in the original publication as much as possible.

Ideally, this would mean that only the methods section of the publication (or supplementary methods) would be read in depth. However, sometimes it is necessary to reference other sections of the paper in order to interpret or locate methodological information. For example, it may be necessary to read the introduction to understand the terminology and abbreviations used in the paper. Likewise, sometimes the experimental design will be overviewed in a figure, or final quality control decisions (and final sample sizes) discussed at the beginning of the results section or in the figure legends.

7.1 When reviewing the methodological information in the publications and metadata for the datasets, studies should be excluded using pre-specified inclusion/exclusion criteria.

Note

Pre-specified inclusion/exclusion criteria:
1) The study was retracted.
2) The experimental design was confounded or lacked a proper control group for the experimental manipulation of interest.
3) The subjects or protocol used in the study were dramatically different from all other included datasets.

7.2 Depending on the research topic, other inclusion/exclusion criteria may also be applied. Ideally, these criteria should be pre-specified as much as possible. If not pre-specified, this deviation from the planned protocol should be reported.

7.3 Following this secondary filtering by the individual experimenter, the experimental design for the remaining datasets and inclusion/exclusion choices should be reviewed and approved by a second researcher (such as a research mentor).

Note

Since this is a critical step, feedback is important. For the 2023 cohort of the Brain Data Alchemy Project, these inclusion/exclusion decisions were reviewed by the mentor and whole cohort.

7.4 Following all filtering, the full search strategy and inclusion/exclusion procedure should be documented using a PRISMA flow diagram.

Note

A template PRISMA flow diagram in an editable Google Slides format can be found here: https://docs.google.com/presentation/d/1Dzw8MQsgo__EkXoIFSFf0ilavvpxX3Zr/edit?usp=sharing&ouid=106595687423493776462&rtpof=true&sd=true

This template should be updated to match the search procedure and the inclusion/exclusion criteria used for your specific project.

7.5 Following all filtering, the essential characteristics of the final datasets that will be included in the meta-analysis should be summarized in a table that can be included with your results.

Note

A template for the table summarizing the characteristics of the included studies in an editable Google Sheets format can be found here: https://docs.google.com/spreadsheets/d/1KptuIeWQH7B6SDk5z7K-Dub2beMSnmSa/edit?usp=sharing&ouid=106595687423493776462&rtpof=true&sd=true

Dataset Import

8 The gene expression data, gene (row) annotation, and sample metadata for **each dataset** will need to be read into R. This will need to be done sequentially - i.e., complete all of the dataset import and analysis steps for one dataset, and then move on to the next.
It may be easiest to conduct these processing steps for each dataset within a its own dedicated R workspace and working directory.

Ideally, this would mean that we import a "tidy" summarized experiment object for the filtered gene expression data, but we found that not all datasets had summarized experiment objects available. **We have provided the code for both scenarios below.**

Note

This vignette is a useful introduction to the concept of a summarized experiment object: https://www.bioconductor.org/packages/release/bioc/vignettes/SummarizedExperiment/inst/doc/SummarizedExperiment.html#:~:text=SummarizedExperiment%20is%20a%20matrix%2Dlike,of%20numeric%20or%20other%20mode.

8.1 This code will make use of several R code packages which may need to be installed before being loaded. One package is plyr, the other packages are part of a series called "the tidyverse" which can be installed and loaded together using a single command.

Software

R package plyr — NAME

Wickham — DEVELOPER

http://cran.r-project.org/web/packages/plyr/index.html — SOURCE LINK

Software

R package tidyverse — NAME

Wickham H — DEVELOPER

http://cran.r-project.org/web/packages/tidyverse/index.html — SOURCE LINK

Command

R code: Install and load the R packages tidyverse and plyr

```
install.packages("tidyverse")

library(tidyverse)


install.packages("plyr")

library(plyr)
```

CITATION

Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R, Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller E, Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V, Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H (2019). "Welcome to the tidyverse". Journal of Open Source Software.
LINK
doi:10.21105/joss.01686

CITATION

Wickham H (2011). The Split-Apply-Combine Strategy for Data Analysis . Journal of Statistical Software.
LINK
10.18637/jss.v040.i01

8.2 If there is a summarized experiment object for a dataset, this code can be adapted to import the summarized experiment object for the dataset from Gemma into our R environment:

Command

Example R code: Importing a summarized experiment object from Gemma into R

```
#This code imports the summarized experiment object for dataset
"GSE81672":

SummarizedExperiment_Filtered<-
gemma.R::get_dataset_object("GSE81672", type = 'se', filter=TRUE,
consolidate="average")
```

Note

- Detailed instructions for using the get_dataset_object function can be found here: https://rdrr.io/github/PavlidisLab/Gemma-API/man/get_dataset_object.html
- The first parameter in the function is the dataset ID number (typically starting with "GSE...")
- The second parameter specifies the type of object that we wish to import. We have chosen 'se', which stands for "summarized experiment"
- The third parameter specifies whether we want to filter the gene expression data to drop genes that have low variability. Gemma applies a standardized filter across all platforms (RNA-Seq/microarray) to remove genes with minimal variance. These genes are likely to suffer from severe floor effects (very low expression/unmeasurable) or may not actually be expressed at all and the measurements that we have are just noise. The filter used by Gemma is quite permissive - it only filters out data that cause problems for the analysis. There are two stages to the variance filter:

1. Remove genes with zero variance.
2. Remove genes that have too many identical values. "Too many" is currently defined as <70% distinct values, but Dr. Paul Pavlidis says that they have tweaked this over time (*personal communication*).

- The fourth parameter (consolidate="average") consolidates the data for probes representing the same gene.

Note

Additional example code for this step can be found here: https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_GemmaDatasets%20/2023_Example_ExploringADataSet_MoreDetail.R

8.3 If there is *not* a summarized experiment object for a dataset, we can make one by adapting the code in this example:

Command

Example R code: Importing a dataset that is not available as a summarized experiment

```
#If a summarized experiment object isn't available for a dataset (in
this case dataset "GSE81672"), this is the error code that you will
get:

SummarizedExperiment_Filtered<-
gemma.R::get_dataset_object("GSE81672", type = 'se', filter=TRUE,
consolidate="average")
# Error in `dplyr::mutate()`:
#   ! Can't transform a data frame with `NA` or `""` names.
# Run `rlang::last_trace()` to see where the error occurred.


#The expression data should be able to be imported into R using a
different function:

GSE85136_Expression<-gemma.R::get_dataset_expression("GSE85136",
filter=TRUE)
#There isn't an option to consolidate the data representing the same
gene for this version of reading in data

str(GSE85136_Expression)
#Classes ‘data.table’ and 'data.frame': 23821 obs. of  28 variables:


#Renaming the object so the downstream code doesn't break:
#Note: Some of the downstream dimensions will be different
GSE85136_Expression_Filtered<-GSE85136_Expression


#We still need the metadata for this dataset object:


GSE85136_Samples<-gemma.R::get_dataset_design("GSE85136")

str(GSE85136_Samples)
#'data.frame':  24 obs. of  4 variables:

#Double-checking that it is in the same order as the expression
data frame:
```

```
data.frame.
row.names(GSE85136_Samples)
# [1] "ControlMale3"   "ControlFemale2" "GFPFemale1"     "GFPMale1"
    "StressFemale2"  "CREMale3"
# [7] "StressMale1"    "CREFemale1"     "StressFemale1"  "GFPMale3"
    "CREFemale2"     "GFPMale2"
# [13] "StressFemale3"  "StressMale2"    "ControlFemale3"
"ControlFemale1" "CREMale2"       "GFPFemale3"
# [19] "ControlMale2"   "GFPFemale2"     "StressMale3"
"ControlMale1"   "CREMale1"       "CREFemale3"

colnames(GSE85136_Expression_Filtered)
# [1] "Probe"            "GeneSymbol"       "GeneName"
"NCBIid"           "Control Male 3"   "Control Male 1"
# [7] "Control Male 2"   "Stress Male 2"    "Stress Male 1"
"Stress Male 3"    "GFP Male 3"       "GFP Male 2"
# [13] "GFP Male 1"       "CRE Male 3"       "CRE Male 2"       "CRE
Male 1"       "Control Female 3" "Control Female 1"
# [19] "Control Female 2" "Stress Female 3"  "Stress Female 2"
"Stress Female 1"  "GFP Female 3"     "GFP Female 2"
# [25] "GFP Female 1"     "CRE Female 3"     "CRE Female 2"     "CRE
Female 1"

#Interestingly, it is *not* in the same order, and the names don't
match.
#To make a version that matches, we would need to remove spaces

ExpressionColnames_NoSpaces<- gsub(" ", "",
colnames(GSE85136_Expression_Filtered)[-c(1:4)])

#What the names look like with the spaces removed:
ExpressionColnames_NoSpaces
# [1] "ControlMale3"   "ControlMale1"   "ControlMale2"
"StressMale2"    "StressMale1"    "StressMale3"
# [7] "GFPMale3"       "GFPMale2"       "GFPMale1"       "CREMale3"
    "CREMale2"       "CREMale1"
# [13] "ControlFemale3" "ControlFemale1" "ControlFemale2"
"StressFemale3"  "StressFemale2"  "StressFemale1"
# [19] "GFPFemale3"     "GFPFemale2"     "GFPFemale1"
"CREFemale3"     "CREFemale2"     "CREFemale1"

#To get the metadata in the same order, we could use join or merge:

#To do this, we need to load the plyr package:
library(plyr)

#We also need to make the gene expression data column names it's own
```

```
object (a data frame) with a column name ("x"):
ExpressionColnames NoSpaces toJoin<-
```

Note

Additional example code for this step can be found here: **http://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_GemmaDatasets%20/2023_DealingWithNoSummarizedExperiment_ForChristabel.R**

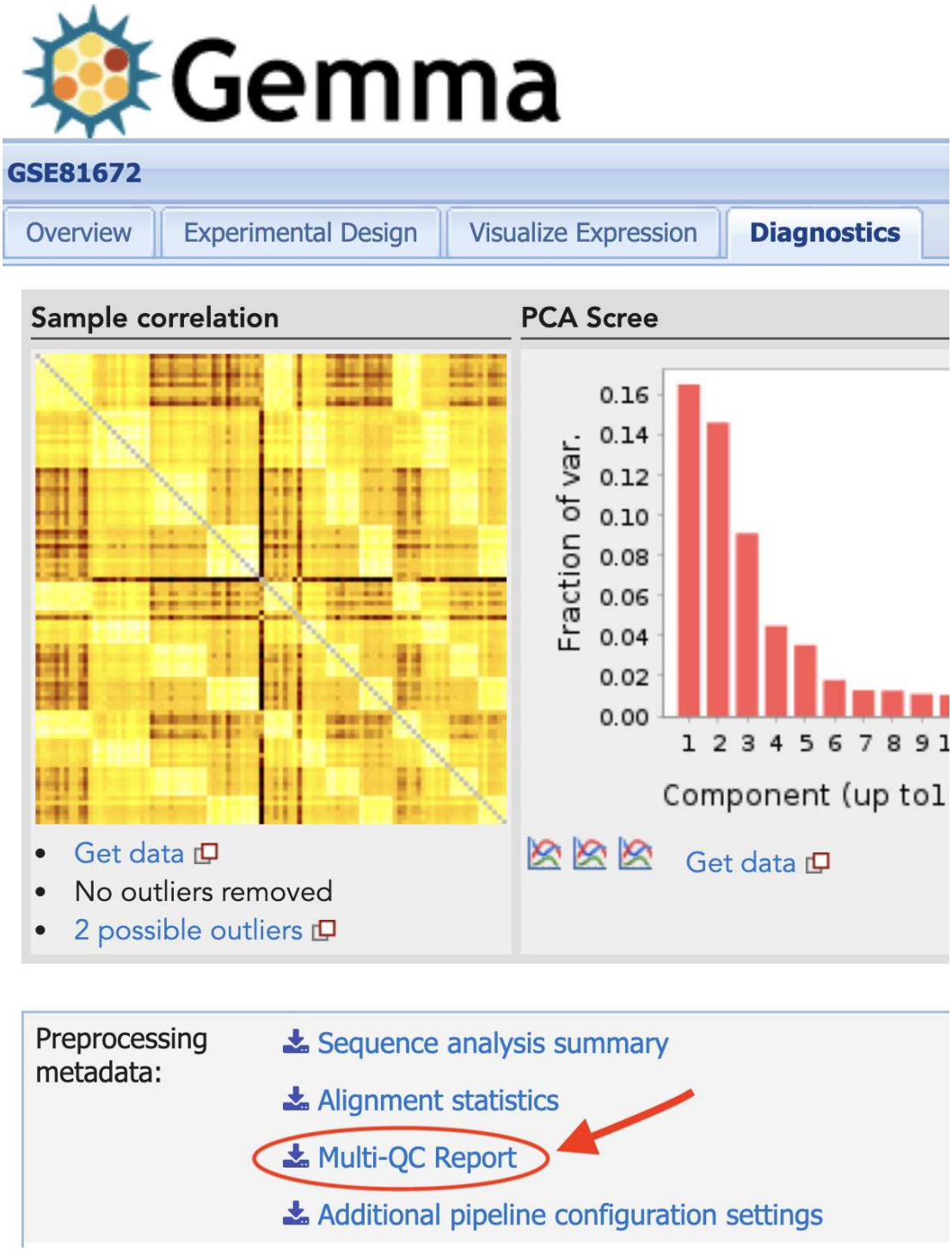
```
#We then need to make a data frame for the metadata what has a column
of names that is also named "x"
GSE85136_Samples<-data.frame(x=colnames(GSE85136_Samples),
GSE85136_Samples)
```

8.4 If the dataset was originally collected using RNA-Seq (vs. microarray), we will need some additional metadata about the number of read counts collected for each sample (library size).

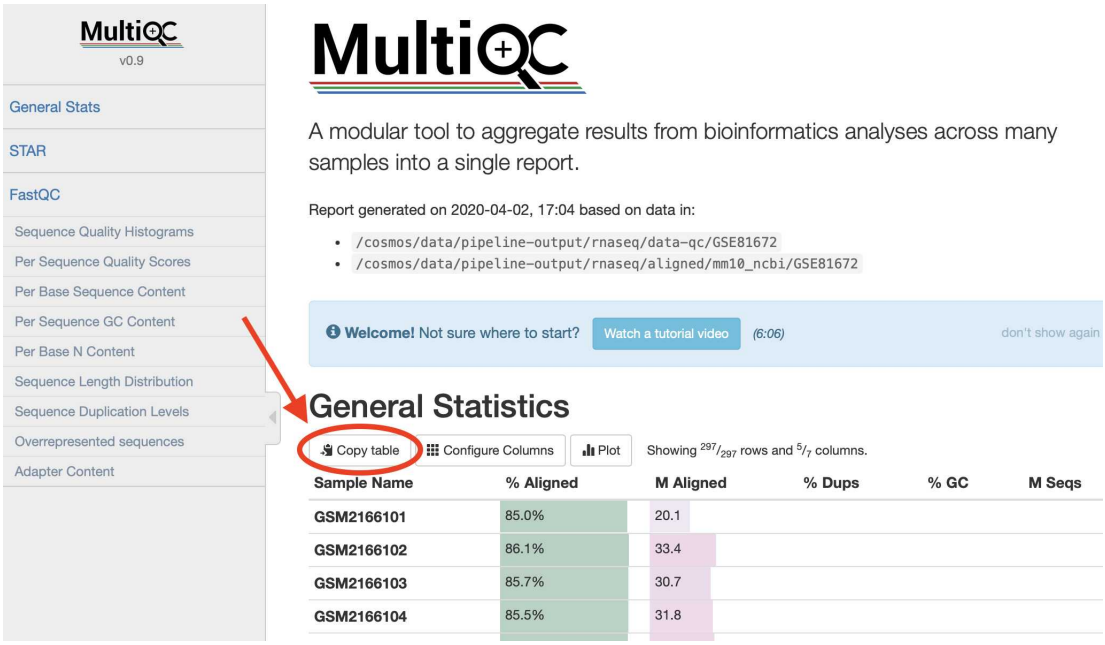
Note

RNA-Seq data is count data - i.e., it is technically *discrete* data with a hard lower boundary (floor). These properties are particularly important for running statistics for genes with lower expression values (closer to the floor), especially for samples with a smaller number of counts overall (smaller library size). Therefore, most analysis pipelines for RNA-Seq data take into account the overall mean-variance trend within the dataset as well as differences in sample library size.

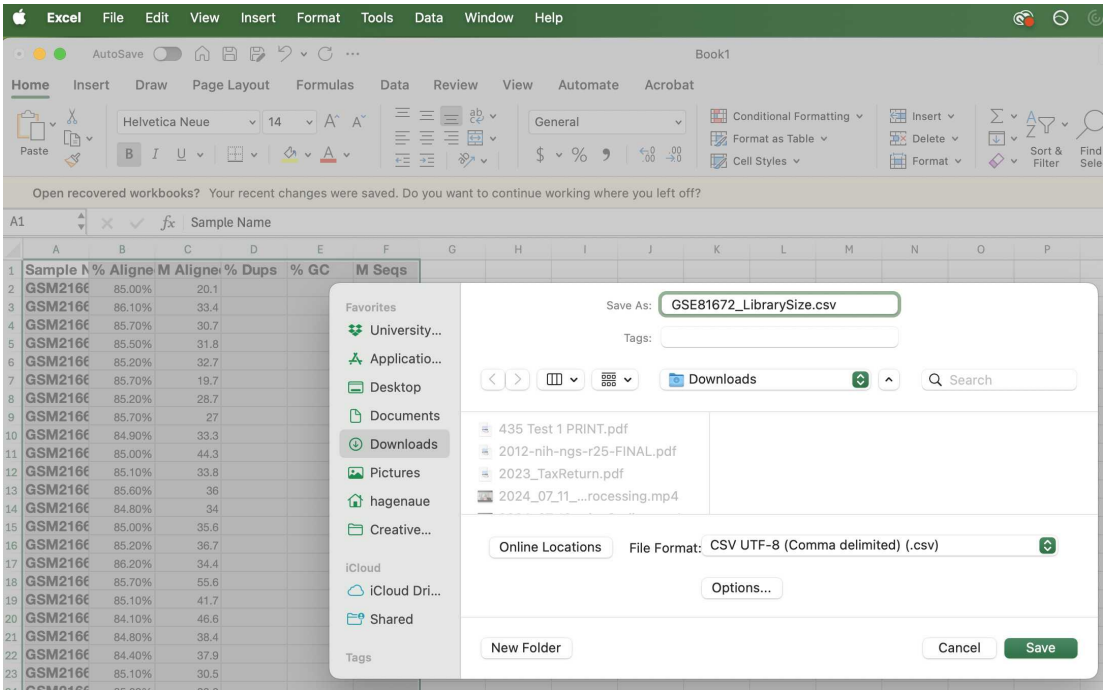
Ideally, this information would be determined by importing the aligned read count data for each sample, but we found that count data was not reliably available within Gemma. Therefore, we ended up importing sample library size information from the MultiQC Report within the Gemma Diagnostics tab.



To access the MultiQC report for an RNA-Seq dataset within Gemma, go to the Gemma Diagnostics tab. Clicking on "Multi-QC Report" will download an html file containing the report onto your computer.



Open the downloaded Multi-QC Report html file (the file will be named after the dataset, e.g., "GSE81672.multiqc.report.html"). Click on "Copy table".



Paste the table within an excel file (or other spreadsheet program, like Google Sheets). Save as a .csv file in the working directory for your project.
(If you don't know the working directory for your project, you can find out by using the command getwd() in R)

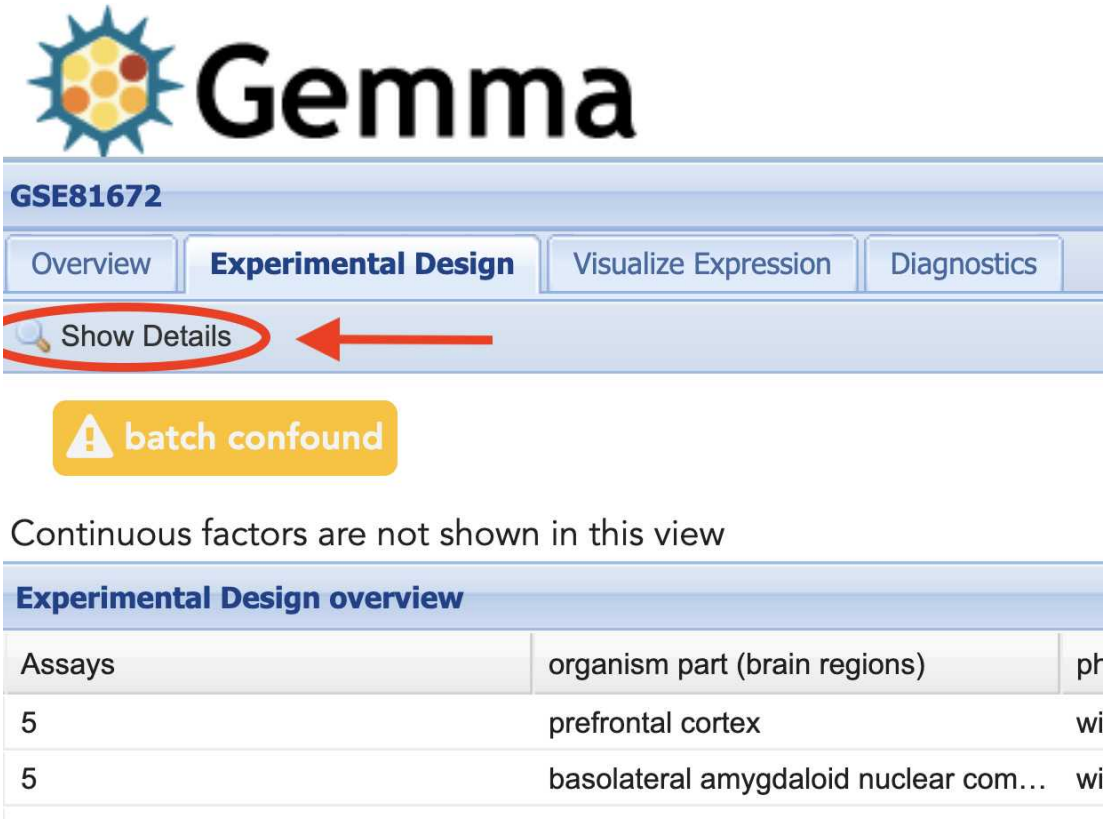
Command

Example R code: Adding library size information to a summarized experiment object

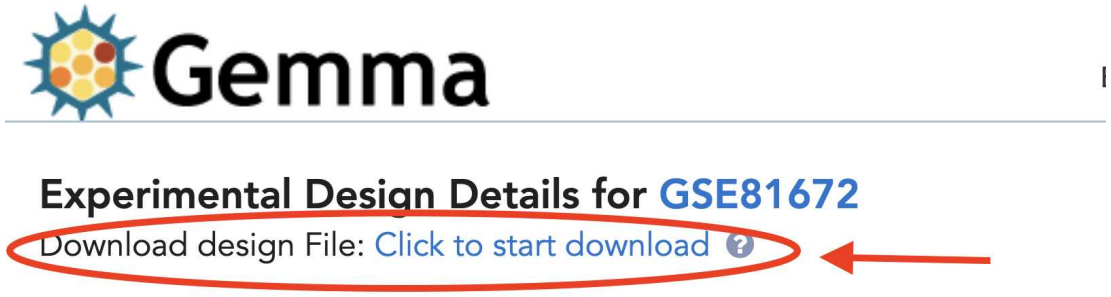
```
#Read in the .csv file containing the MultiQC report information into
R:
GSE81672_LibrarySize<-read.csv("GSE81672_LibrarySize.csv",
header=TRUE, stringsAsFactors = FALSE)

#The library size information is annotated with sample information
using GSM number.
head(GSE81672_LibrarySize)
# ExternalID X..Aligned M.Aligned X..Dups X..GC M.Seqs
# 1 GSM2166101     85.00%      20.1                   NA
# 2 GSM2166102     86.10%      33.4                   NA

#As far as I can tell, this identifier is not in the Summarized
experiment object
#So we need to read in the experimental design info from the Gemma
website, which contains GSM#
```



The library size information from the MultiQC Report is annotated with sample information using GSM number. As far as I can tell, this identifier is not in the Summarized experiment object, so we need to read in the experimental design info from the Gemma website, which contains GSM#. To do this, we need to go to the Experimental Design tab for the dataset within Gemma and click on "Show Details".



Within the Experimental Design details, click on "download design file". This will download a tab-delimited text file which will be named after the dataset (e.g., "13458_GSE81672_expdesign.data.txt"). Save it in the working directory for your project.

Command

Example R code: Reading in the experimental design file containing the GSM#s for Samples

```
#To read in the experimental design file that contains the GSM#s for
the samples:

GSE81672_expdesign<-read.delim("13458_GSE81672_expdesign.data.txt",
sep="\t", comment.char="#", header=TRUE, stringsAsFactors = FALSE)


#This column has the GSM #
GSE81672_expdesign$ExternalID
# [1] "GSM2166189" "GSM2166195" "GSM2166180"

#This column includes the Name... and a bunch of other stuff:
GSE81672_expdesign$Bioassay
# [1]
"GSE81672_Biomat_64___BioAssayId=450411Name=Sample102.NAC_susceptible_
ketamine_non_responder"
# [2]
"GSE81672_Biomat_58___BioAssayId=450412Name=Sample108.HIP_susceptible_
saline"
#...
```

Command

Example R code: Reformatting the experimental design sample ids to match the Summarized Experiment

```
#This column includes the Name... and a bunch of other stuff:
GSE81672_expdesign$Bioassay
# [1]
"GSE81672_Biomat_64___BioAssayId=450411Name=Sample102.NAC_susceptible_
ketamine_non_responder"
# [2]
"GSE81672_Biomat_58___BioAssayId=450412Name=Sample108.HIP_susceptible_
saline"
#...

#This code splits these long names up at the "Name=" and then
combines things back into a matrix again
temp<-do.call(rbind.data.frame, strsplit(GSE81672_expdesign$Bioassay,
"Name="))

#This adds the name information to the expdesign matrix:
GSE81672_expdesign$SampleName<-temp[,2]
GSE81672_expdesign$SampleName

#For this dataset, The formatting is still a little different from
the Summarized Experiment object, e.g.
GSE81672_expdesign$SampleName[1]
#"Sample102.NAC_susceptible_ketamine_non_responder"

row.names(colData(SummarizedExperiment_Filtered[[1]]))[1]
#"Sample 18: AMY_resilient_saline"
#One version uses a ., the other uses a ": "

#the ": " version is straight off of Gene Expression Omnibus (GEO),
so I'm going to assume that for some reason Gemma doesn't like : in
the design data.frame

#This is likely to be a dataset specific issue - I'm not sure how to
generalize this code.
#For many datasets, things may be fine at this point.

#So we essentially need a find and replace - in R, these are often
done with sub() or gsub()
#https://www.digitalocean.com/community/tutorials/sub-and-gsub-
```

```
#https://www.digitalocean.com/community/tutorials/sub-and-gsub
function-r
#https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/gr
ep

#this is a little funky, because we're not replacing simple
alphanumeric characters, but punctuation that also has a meaning for
coding
temp<-gsub("[.]", ": ", GSE81672_expdesign$SampleName)

#But it turns out that the names also differ because of an extra
space.
#To take care of this we just removed all spaces from both vectors
because it is easier:

GSE81672_expdesign$SampleName_toJoin<-gsub(" ", "", temp)

SampleName_toJoin<-gsub(" ", "",
row.names(colData(SummarizedExperiment_Filtered[[1]])))
```

Command

Example R code: Adding library size info to the Summarized Experiment object

```
#We can't just add library size as a column to experimental design
because the samples aren't in the same order
#We have to use a "join" or "merge" function to align them instead

#To join the design matrix with the library size, we need to have two
data.frames that have columns that have the same name:
str(GSE81672_LibrarySize)

#In this data.frame, the GSM# is called Sample.Name, whereas in the
exp. design data.frame it is called ExternalID, so I will rename it
ExternalID:
colnames(GSE81672_LibrarySize)[1]<-"ExternalID"

#Now we can join by these columns (the function "merge" also works
for this):
GSE81672_expdesign_wLibrarySize<-join(GSE81672_expdesign,
GSE81672_LibrarySize, by="ExternalID", type="left")


#Now we have to add this information into the Summarized Experiment
object
#Which also has a different sample order
#I'm going to grab that sample order information and name it after
the column with Sample Names in the experimental design object

SamplesOrderedLikeSummarizedExperiment<-
data.frame(SampleName_toJoin=SampleName_toJoin)

GSE81672_expdesign_wLibrarySize_Ordered<-
join(SamplesOrderedLikeSummarizedExperiment,
GSE81672_expdesign_wLibrarySize, by="SampleName_toJoin", type="left")



#Double checking that things are actually in the same order
cbind(row.names(colData(SummarizedExperiment_Filtered[[1]])),
GSE81672_expdesign_wLibrarySize_Ordered$SampleName_toJoin)

#Then I can finally add library size to the Summarized Experiment
```

```
#then I can finally add library size to the Summarized Experiment
object:
colData(SummarizedExperiment_Filtered[[1]])$LibrarySize<-
GSE81672_expdesign_wLibrarySize_Ordered$M.Aligned
```

8.5 To increase the reproducibility of our results, save a copy of the Gemma Summarized Experiment data in an Excel-friendly file format (.csv).

Note

We need to document the version of the data that we used for our analysis because if someone tries to reproduce our results in the future by re-running our code to extract datasets from Gemma, they may download a slightly different dataset (e.g., if Gemma reannotates the microarray probes or realigns the RNA-Seq reads to an updated genome). We will save our R Workspace (.Rdata) which will preserve a copy of the dataset in its current form, but that file type is less accessible to folks who don't work in R.

Command

Example R Code: Save a copy of the dataset in an Excel-friendly format (.csv)

```
write.csv(rowData(SummarizedExperiment_Filtered[[1]]),
"Example_Annotation.csv")

write.csv(colData(SummarizedExperiment_Filtered[[1]])[,-1],
"Example_MetaData.csv")

write.csv(ExpressionData_Filtered, "ExpressionData_Filtered.csv")
```

Double Checking Data Normalization: Externally-Derived Datasets

9 After we ran all of the meta-analyses for the Summer 2022 cohort of the Brain Data Alchemy Project, we realized that datasets that were marked “external” in the Gemma database had

sometimes been imported into Gemma in a format that was incompatible with Gemma's analysis pipeline.

Note

The analysis pipeline in Gemma typically begins with raw data extracted from public databases (e.g., Gene Expression Omnibus (GEO), **https://www.ncbi.nlm.nih.gov/geo/**, (Lim et al., 2021)). After we ran all of the meta-analyses for the Summer 2022 cohort of the Brain Data Alchemy Project, we realized that datasets that were marked "external" in the Gemma database had been subjected to a slightly different analysis pipeline. These datasets lacked available raw data, and were therefore imported into Gemma from external databases as the probe- or gene-level summarized expression data for each subject as generated by the original dataset contributors.

This imported external data sometimes appeared to be in formats that were incompatible with Gemma's analysis pipeline. This seemed to be particularly true for datasets generated with Agilent transcriptional profiling platforms, which, for proprietary reasons, always lacked raw expression data in the GEO database, and appeared to have summarized gene expression data per sample that was generated by an unstandardized analysis pipeline (*i.e.,* GEO records included a variety of normalization methods and formats). For these Agilent records, we regularly found that the original units in GEO were non-standard or uninterpretable (*e.g.,* just labeled "normalized") or had something appearing to follow a standard Log2 expression unit format but then had units within the Gemma database that had clearly been log transformed a second time (*e.g.,* the range of Log2 expression units in Gemma was highly restricted, such as units ranging between 2-4).

CITATION

Lim N, Tesar S, Belmadani M, Poirier-Morency G, Mancarci BO, Sicherman J, Jacobson M, Leong J, Tan P, Pavlidis P (2021). Curation of over 10 000 transcriptomic studies to enable data reuse.. LINK
https://doi.org/10.1093/database/baab006

9.1 Due to these irregularities, all datasets - especially "external" datasets - should be re-reviewed for evidence of irregular units at import or accidental double log2 transformation (as signaled by units with a highly restricted range - such as gene expression values only ranging from 2-4).

Command

Example R code: Double-checking the range of gene expression values in a dataset

```
#Pull out the matrix of gene expression data from the Summarized
Experiment object:
ExpressionData_Filtered<-assay(SummarizedExperiment_Filtered[[1]])

#To observe the distribution of gene expression values, plot a
histogram of the gene expression data:
hist(ExpressionData_Filtered)

#The range for the gene expression data can also be explored by
examining the minimum and maximum values:
min(ExpressionData_Filtered)
max(ExpressionData_Filtered)
```

9.2 If the dataset appears to have been clearly accidentally log2 transformed twice over (as signaled by gene expression units with a highly restricted range, e.g., from 2-4), the second log2 transformation can be reversed using $y=2^x$.

Dataset Subsetting

10 Subset the dataset down to the samples that you plan to use.

Note

For datasets that include samples from more than one tissue or brain region, this subsetting step will include filtering out samples that are not from your tissue or region of interest.

There may be other samples that you wish to filter out because there are theoretical reasons to believe that they will not properly represent the effects of your variable of interest. For example, if you are studying chronic stress, you may want to filter out samples that were treated with interventions that were intended to counteract the effects of chronic stress (e.g., antidepressant treatment, genetic knock-downs of the glucocorticoid receptor, etc).

10.1 Determine the characteristics of the samples in the dataset and how many samples have each characteristic of interest.

Command

Example code: Exploring sample characteristics in the summarized experiment object

```
#The sample data - organism part, treatment, batch, etc:
colData(SummarizedExperiment_Filtered[[1]])
head(colData(SummarizedExperiment_Filtered[[1]]))

#To view the distribution of samples from different tissues or brain
regions:

table(SummarizedExperiment_Filtered[[1]]$`organism part`)
#Example output:
# basolateral amygdaloid nuclear complex                     nucleus
accumbens
#           25
                              25
# prefrontal cortex                   ventral,Ammon's horn
#            25                                      24

#Other common types of variables that you may want to view the
distribution of samples for:

#Phenotype:
table(SummarizedExperiment_Filtered[[1]]$phenotype)

#Treatment:
table(SummarizedExperiment_Filtered[[1]]$treatment)
```

10.2 Filter out the samples that have unwanted characteristics.

Note

Helpful information for coding this filter:

We don't use equals to mean "equals" in R, we use it to define objects, e.g.:
MynewObject=c(1,2,3,4)
means the same thing as...
MynewObject<-c(1,2,3,4)

So "==" means equals, whereas "!=" means doesn't equal

When combining together criteria:
"&" means "AND"
"|" (also called the pipe) means "OR"

Command

Example code: Subsetting the summarized experiment object down to samples with desired characteristics

```
#For an example, I'm using an experiment with a weird/complex design

#This experiment includes samples from many tissues. I am only
interested in the hippocampus ("ventral,Ammon's horn").

#I'm interested in antidepressant effects, and it looks like only the
stress susceptible subjects were treated with antidepressants
#So I am going to subset down to just the stress susceptible subjects

#I'm going to keep both antidepressant responders and non-responders
#because I'm looking at antidepressant effects (not antidepressant
effectiveness)
#and most of my other datasets will not screen for antidepressant
effectiveness


SampleFilter<-
  SummarizedExperiment_Filtered[[1]]$`organism
part`=="ventral,Ammon's horn" &
  (SummarizedExperiment_Filtered[[1]]$phenotype=="susceptible
toward,Responder"|
     SummarizedExperiment_Filtered[[1]]$phenotype=="Non-
responder,susceptible toward"|
     SummarizedExperiment_Filtered[[1]]$phenotype=="susceptible
toward")

#Subsetting the data to have only subjects with desired
characteristics:
SummarizedExperiment_Subset<-SummarizedExperiment_Filtered[[1]]
[,SampleFilter]


#Sanity Check: Double-checking that the subsetting worked properly:

str(SummarizedExperiment_Subset)
# class: SummarizedExperiment
# dim: 22782 16
#This dataset is much smaller now
```

```
table(SummarizedExperiment_Subset$`organism part`)
#ventral,Ammon's horn
#16
#Only the hippocampal samples remain.

table(SummarizedExperiment_Subset$phenotype,
SummarizedExperiment_Subset$treatment)
#                                  imipramine ketamine reference
substance role,saline
# Non-responder,susceptible toward   4        3
        0
# susceptible toward                                0        0
                     3
# susceptible toward,Responder           3        3
            0
```

Removing outlier samples from the dataset

11 Check for outlier samples and remove them if they are present.

> **Note**
>
> Within transcriptional profiling analysis, outlier samples are normally defined at the level of the whole genome instead of at the level of individual measurements for particular genes. If the distribution of all gene expression measurements for a sample looks completely different from the other samples in the dataset, it is likely that the dramatic differences are due to a technical problem and that sample should be removed from the dataset.

11.1 Using the full expression data, examine the pattern of sample-sample correlations to identify any outlier samples.

Note

Within Gemma's algorithm, samples are considered potential outliers if their adjusted median sample-sample correlation is outside the interquartile range of sample-sample correlations for the sample with the closest median correlation to them (Lim et al. 2021).

Gemma's algorithm is therefore optimized to detect single outliers, but can miss pairs or groups of outlier samples. To double-check that there weren't still pairs or groups of outliers remaining in the dataset, we visualized the distribution of sample-sample correlations within each dataset using a correlation matrix and boxplots.

Command

Example code: Examining the sample-sample correlations within the gene expression data

```
#Pulling out a matrix of gene expression data for this subset (to use
in functions that require matrices)

ExpressionData_Subset<-assay(SummarizedExperiment_Subset)

#Creating a matrix showing how each sample correlates with every
other sample:
CorMatrix<-cor(ExpressionData_Subset)

#Writing that matrix out to the working directory to save it:
write.csv(CorMatrix, "CorMatrix.csv")

#Creating a hierarchically clustered heatmap illustrating the sample-
sample correlation matrix:
heatmap(CorMatrix)

#Creating a boxplot illustrating the distribution of sample-sample
correlations for each sample:
boxplot(CorMatrix)
#Outlier samples should be obvious at this point
#Gemma's algorithm uses the cut-off that the median sample-sample
correlation for any particular sample should not fall outside the
interquartile range (boxes) for all of the other samples.
#We double-checked that there weren't pairs or small clusters of
samples that fell outside the the interquartile range (boxes) for all
of the other samples.
```

11.2 If there is an outlier sample, remove it from the summarized experiment object.

Command

Example code: Removing an outlier sample from the summarized experiment object

```
#If there is an outlier sample, you can remove it using subsetting
similar to above by identifying it's column name

#E.g.
OutlierFilter<-colnames(ExpressionData_Subset)!="Sample 15:
HIP_susceptible_imipramine_non_responder"

OutlierFilter
#[1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
TRUE  TRUE  TRUE  TRUE  TRUE  TRUE

SummarizedExperiment_Subset_noBad<-
SummarizedExperiment_Subset[,OutlierFilter]

#Sanity check: the Summarized experiment object should now be one
column smaller

SummarizedExperiment_Subset_noBad
# class: SummarizedExperiment
# dim: 22782 15

#Then recreate the ExpressionData Matrix from our new summarized
experiment object that has had the outlier samples removed:

ExpressionData_Subset_noBad<-assay(SummarizedExperiment_Subset_noBad)
```

11.3 If there isn't an outlier sample, just rename the summarized experiment object so that the downstream code works.

Command

Example code: Renaming the summarized experiment object to acknowledge no outliers

```
#If we *don't* have any outliers to remove, we can just rename our
object so it works with the downstream code:

SummarizedExperiment_Subset_noBad<-SummarizedExperiment_Subset

#Then recreate the ExpressionData Matrix from our new summarized
experiment object:

ExpressionData_Subset_noBad<-assay(SummarizedExperiment_Subset_noBad)
```

Filtering out genes lacking variability in expression measurements in experimental subgroups

12 Filter out genes (rows) in the dataset that lack variability in expression measurements in any of the experimental groups.

Note

Now that we have subsetted our data, we have a new problem:
To prevent unwanted artifacts within statistical calculations, Gemma filtered out genes that lacked variability in expression measurements in the full dataset... but that doesn't mean that the remaining genes have variability in our chosen subset of samples. Genes lacking variability in expression measurements are going to cause problems - you can't run stats on a variable with no variability!

But there is more:
We may still have issues with genes that lack any variability *for any particular subgroup of interest* as well, since a lack of variability in any experimental subgroup can cause problems for later statistical tests.

12.1 To filter out genes (rows) that lack variability in the experimental subgroups, we will need to know how to define our experimental subgroups of interest.

If only one variable is of interest (for example, antidepressant treatment), this will be relatively easy.
If more than one variable is of interest (for example, the interaction of antidepressant treatment with treatment responder vs. treatment non-responder status), then we will need to identify genes that lack variability in any subgroup created by any combination of those two variables (treatment*phenotype).

Command

Example code: Filtering genes (rows) that lack variability in the experimental subgroups

```
#We will need to calculate the sd for each of our experimental groups
of interest.
#For example, for this dataset, I might be interested in the
experimental groups defined by treatment.

#This function calculates the sd for each treatment group for a
particular gene (row of data):
tapply(ExpressionData_Subset_noBad[1,],
SummarizedExperiment_Subset_noBad$treatment, sd)
# imipramine                          ketamine reference substance
role,saline
# 0.6547312                         0.9273463
0.7106250

#We want the minimum sd for all of our groups to not be zero, e.g.:
min(tapply(ExpressionData_Subset_noBad[1,],
SummarizedExperiment_Subset_noBad$treatment, sd))!=0
#[1] TRUE

#... and we need to know that for all rows.
GenesToFilter<-apply(ExpressionData_Subset_noBad, 1, function(y)
(min(tapply(y, SummarizedExperiment_Subset_noBad$treatment, sd))!=0))
head(GenesToFilter)

#This tells us how many genes (rows) we'll end up keeping:
sum(GenesToFilter)
#[1] 21686

SummarizedExperiment_Subset_noBad_Filtered<-
SummarizedExperiment_Subset_noBad[GenesToFilter,]
SummarizedExperiment_Subset_noBad_Filtered

#Remaking the expression set from the filtered summarized experiment
object:
ExpressionData_Subset_noBad_Filtered<-
assay(SummarizedExperiment_Subset_noBad_Filtered)
str(ExpressionData_Subset_noBad_Filtered)
```

Checking for batch confounds

13 Check for confounding variability due to technical processing batches.

Note

Processing batches are often the largest sources of variability in gene expression datasets. They are unfortunately also often distributed in an unbalanced manner in regards to variables of interest. This can cause the batch effects to be mistaken for the effects of the variable of interest, confounding results.

Gemma's analysis pipeline screens for confounding batch effects in the dataset as a whole. If a dataset is found to have a variable that is highly confounded by a processing batch (e.g., brain region), Gemma breaks apart the dataset based on that variable, and runs separate differential expression analyses on each of the components.

Unfortunately, this procedure does not:
1) Rule out confounding batch effects in our particular subset of samples
2) Control for large batch effects in designs that are more moderately imbalanced.

13.1 View the distribution of the batch variable across samples and determine if it needs to be divided into subcomponents (e.g., device, run, flow cell, lane, etc).

Note

Currently, Gemma has all of the processing batch information for RNA-Seq lumped into one variable (device, run, flow cell, lane) called "block".

Command

Example R code: View the distribution of the batch variable across samples

```
table(SummarizedExperiment_Subset_noBad_Filtered$block)

# Device=11V6WR1:Run=206:Flowcell=C3HBLACXX:Lane=1
Device=11V6WR1:Run=206:Flowcell=C3HBLACXX:Lane=3
# 1                                                    3
# Device=HWI-D00147:Run=59:Flowcell=C2GWMACXX:Lane=2  Device=HWI-
D00147:Run=59:Flowcell=C2GWMACXX:Lane=4
# 3                                                    2
# Device=HWI-D00147:Run=59:Flowcell=C2GWMACXX:Lane=5  Device=HWI-
D00147:Run=59:Flowcell=C2GWMACXX:Lane=6
# 1                                                    1
# Device=HWI-D00147:Run=59:Flowcell=C2GWMACXX:Lane=7  Device=HWI-
D00147:Run=59:Flowcell=C2GWMACXX:Lane=8
# 1                                                    1
# Device=HWI-ST1147:Run=158:Flowcell=C34A7ACXX:Lane=3 Device=HWI-
ST1147:Run=158:Flowcell=C34A7ACXX:Lane=7
# 1                                                    1
# Device=HWI-ST276:Run=345:Flowcell=C3H70ACXX:Lane=1
# 1
```

13.2 If the batch variable (block) has subcomponents (e.g., device, run, flow cell, lane, etc), split them apart into individual variables.

Command

Example R Code: Splitting apart the batch variable (block) into its subcomponents

```
#This function breaks apart these "blocks" into specific batch-
related variables (device, run, flow cell, lane):
strsplit(SummarizedExperiment_Subset_noBad_Filtered$block, ":")

#To make that into an easier-to-use data.frame
BatchVariables<-do.call(rbind.data.frame,
strsplit(SummarizedExperiment_Subset_noBad_Filtered$block, ":"))

str(BatchVariables)

#I'm going to rename these (note - this will be dataset specific):
colnames(BatchVariables)<-c("Device", "Run", "FlowCell", "Lane")
```

13.3 Determine whether any of the batch-related variables are distributed in an unbalanced manner across your variables of interest.

Command

Example R code: Determining whether batch-related variables are distributed unevenly across experimental groups

```
table(SummarizedExperiment_Subset_noBad_Filtered$treatment,
BatchVariables$Device)
#                                  Device=11V6WR1 Device=HWI-D00147
Device=HWI-ST1147 Device=HWI-ST276
# imipramine                                    2                 4
             1                0
# ketamine                                      1                 4
             1                0
# reference substance role,saline               1                 1
             0                1

table(SummarizedExperiment_Subset_noBad_Filtered$treatment,
BatchVariables$Run)
#                                  Run=158 Run=206 Run=345 Run=59
# imipramine                           1       2       0      4
# ketamine                             1       1       0      4
# reference substance role,saline      0       1       1      1

#Looks like run and device may be redundant
table(BatchVariables$Device, BatchVariables$Run)
#                     Run=158 Run=206 Run=345 Run=59
# Device=11V6WR1           0       4       0      0
# Device=HWI-D00147        0       0       0      9
# Device=HWI-ST1147        2       0       0      0
# Device=HWI-ST276         0       0       1      0

table(SummarizedExperiment_Subset_noBad_Filtered$treatment,
BatchVariables$FlowCell)
#Looks also potentially redundant

table(BatchVariables$FlowCell, BatchVariables$Run)
#                       Run=158 Run=206 Run=345 Run=59
# Flowcell=C2GWMACXX        0       0       0      9
# Flowcell=C34A7ACXX        2       0       0      0
# Flowcell=C3H70ACXX        0       0       1      0
# Flowcell=C3HBLACXX        0       4       0      0
#yep

table(SummarizedExperiment_Subset_noBad_Filtered$treatment
```

```
table(summarizedExperiment_subset_noBad_filtered$treatment,
BatchVariables$Lane)
#                                  Lane=1 Lane=2 Lane=3 Lane=4
Lane=5 Lane=6 Lane=7 Lane=8
# imipramine                          1      1      1      1      0
    1      1      1
# ketamine                            0      1      2      1      1
    0      1      0
# reference substance role,saline     1      1      1      0      0
    0      0      0
#Too many Lanes to take into consideration
```

Principal Components Analysis (PCA)

14 Use Principal Components Analysis (PCA) to identify biological or technical co-variates that may be highly impacting the gene expression data.

Note

Principal components analysis (PCA) is a "dimension reduction" analysis method that identifies the main gradients of variation within a multi-dimensional dataset. Within transcriptional profiling analysis, PCA and related methods (sometimes also called "multidimensional scaling (MDS)") are used to identify either genes or samples that show highly correlated expression. This can be useful for identifying biological functions associated with variables of interest (e.g., experimental treatment), but is also often used to help identify impactful sources of technical variation (e.g., processing batch effects) or outlier samples (samples that generally don't correlate with any other samples).

For more background on PCA, I really like the general description given here:
http://ordination.okstate.edu/PCA.htm

This is also a good interactive website:
http://setosa.io/ev/principal-component-analysis/

For those of you who want to go deeper into the math of PCA:
An awesome introduction to matrix algebra:
https://faculty.utrgv.edu/eleftherios.gkioulekas/Teaching/LectureNotes/MathForEEandLinAlg/books/book_tapia_linear_algebra.pdf
(Eigenvectors and Eigenvalues are Chapter 9)

14.1 Load the R stats package: *library(stats)*

Note

The stats package typically comes pre-installed with R (base R).
- This protocol was designed using the version of stats (v.4.3.0) available on 06/01/2023. It is likely to work with other versions of the package.

Software

The R stats package NAME

R Core Team and contributors worldwide DEVELOPER

https://stat.ethz.ch/R-manual/R-devel/library/stats/html/00Index.html SOURCE LINK

14.2 Run PCA on the transposed gene expression matrix.
Transposing a matrix means to switch the orientation of the matrix so that the original rows are now columns and the original columns are now rows.

Command

Example R Code: Run PCA on the transposed gene expression matrix

```
#Run PCA on the transposed gene expression matrix
#In this case, ExpressionData_Subset_noBad_Filtered is the object
containing the gene expression matrix
pca_output<-prcomp(t(ExpressionData_Subset_noBad_Filtered),
scale=TRUE)
```

14.3 Create a scree plot for the PCA results. This plot illustrates the proportion of variance explained by each principal component of variation in the dataset.

Command

Example R Code: Output a scree plot for the PCA results

```
#Output a scree plot for the PCA results:

#This plot illustrates the proportion of variance explained by each
principal component (PC):
png("10 PCA Scree Plot1.png")
plot(summary(pca_output)$importance[2,]~
(c(1:length(summary(pca_output)$importance[2,]))), main="Variance
Explained by Each Principal Component", xlab="PC #", ylab="Proportion
of Variance Explained", col=2)
dev.off()
```

14.4 Extract out the first four principal components of variation (PC1-PC4) and make them named R objects for graphing purposes. These values show you the weighted contribution of each of your samples to the principal components of variation in the dataset.

Command

Example R Code: Make PC1-PC4 named R objects

```
#Extract out the first four principal components and make them named
R objects for graphing purposes:
PC1<-pca_output$x[,1]
PC2<-pca_output$x[,2]
PC3<-pca_output$x[,3]
PC4<-pca_output$x[,4]
```

14.5 Extract out the first four eigenvectors and export them as a .csv file for later reference. The eigenvectors show the weighted contribution of each of the genes in your gene expression matrix to each of the principal components of variation in the dataset.

Command

Example R Code: Export eigenvectors for PC1-4 as a .csv file

```
#Extract the eigenvectors for PC1-4 and make them a named R object
PCeigenvectors<-pca_output$rotation[ ,c(1:4)]

#Add the gene names (rowData) from the gene expression matrix to the
object with the eigenvectors
PCeigenvectors2<-cbind(PCeigenvectors,
rowData(SummarizedExperiment_Subset_noBad_Filtered))

#Write this information out as a .csv file:
write.csv(PCeigenvectors2, "PCeigenvectors.csv")
```

14.6 Output a scatterplot illustrating the relationship between Principal components 1 & 2 (PC1 & PC2).

Note

Any remaining outlier samples should be highly visible in this plot. If the samples are strongly clustered in any way, it may also be visible here. The datapoints in the scatterplots can be colored to reflect different categorical variables in the dataset - this can help explain the main sources of variation in the data.

Command

Example R Code: Output a scatterplot illustrating PC1 vs. PC2 as a .png file

```
#Output a scatterplot illustrating the relationship between Principal
components 1 & 2 (PC1 & PC2):
png("10 PC1 vs PC2.png")
plot(PC1~PC2, main="Principal Components Analysis")
dev.off()
```

14.7 Examine the relationship between the top PCs (PC1-4) and each of the potential biological and technical co-variates.

Command

Example R Code: Plotting a PC versus a categorical variable

```
#Examining the relationship between PC1 and a categorical variable
using a boxplot
#In this case, the plotting code is referring back to the summarized
experiment object containing the gene expression matrix used in the
PCA analysis


boxplot(PC1~SummarizedExperiment_Subset_noBad_Filtered$treatment,
las=2, xlab="")

#The relationship between the PC and continuous numeric variables can
be plotted as a scatterplot using the "plot" function instead of
"boxplot"
```

Double checking the model: Examining the expression data from a single gene vs. the variable of interest

15 Before progressing forward with the differential expression analysis, it can be helpful to examine the data from a single gene in comparison to the variable of interest. This can be used to confirm that the intended differential expression model makes sense and isn't causing error messages.

15.1 If needed, rename the conditions for the variable(s) of interest so that the names are concise and informative.

Command

Example R Code: Renaming conditions for the variable of interest

```
#Renaming conditions
#Some of the levels for my variables are a mouthful.
#I'm going to rename them:

#The names of the conditions (levels) for the variable with the
number of subjects in each condition:
table(SummarizedExperiment_Subset_noBad_Filtered$treatment)
#imipramine               ketamine reference substance role,saline
#7                               6                              11

#I'm going to rename the control condition because it is a mouthful:
SummarizedExperiment_Subset_noBad_Filtered$treatment[SummarizedExperim
ent_Subset_noBad_Filtered$treatment=="reference substance
role,saline"]<-"saline"

#The new names:
table(SummarizedExperiment_Subset_noBad_Filtered$treatment)
# imipramine   ketamine     saline
# 7          6         11

#Renaming the phenotype conditions as well:
SummarizedExperiment_Subset_noBad_Filtered$phenotype[SummarizedExperim
ent_Subset_noBad_Filtered$phenotype=="Non-responder,susceptible
toward"]<-"non-responder"
SummarizedExperiment_Subset_noBad_Filtered$phenotype[SummarizedExperim
ent_Subset_noBad_Filtered$phenotype=="susceptible toward,Responder"]<-
"responder"
SummarizedExperiment_Subset_noBad_Filtered$phenotype[SummarizedExperim
ent_Subset_noBad_Filtered$phenotype=="susceptible toward"]<-
"untreated"

#The new names:
table(SummarizedExperiment_Subset_noBad_Filtered$phenotype)
# non-responder     responder     untreated
# 7             6             3
```

15.2 Set up the categorical variables of interest so that they have an intuitive reference group

Command

Example R Code: Setting the reference group for a categorical variable

```
#Setting up our categorical variables so that they have intuitive
reference groups:

#First, we need to define the categorical variable as a factor:
#I tend to make this a new variable in the Summarized experiment
object instead of overwriting the old categorical variable:
SummarizedExperiment_Subset_noBad_Filtered$treatment_factor<-
as.factor(SummarizedExperiment_Subset_noBad_Filtered$treatment)

#R automatically assigns a reference level for this new factor that
may or may not make intuitive sense
#When viewing the levels for a factor, the reference variable is
whatever is listed first:
levels(SummarizedExperiment_Subset_noBad_Filtered$treatment_factor)
#[1] "imipramine" "ketamine"   "saline"

#We want saline to be our control (reference) group:
SummarizedExperiment_Subset_noBad_Filtered$treatment_factor<-
relevel(SummarizedExperiment_Subset_noBad_Filtered$treatment_factor,
ref="saline")
```

15.3 Illustrate the relationship between the expression of a single gene and the variable of interest.

Command

Example R Code: A boxplot illustrating the relationship between the expression for a single gene and a categorical variable of interest

```
#Plotting data for a particular gene

#Getting the expression data for a particular gene (in this case,
parvalbumin):
Pvalb<-assay(SummarizedExperiment_Subset_noBad_Filtered)
[rowData(SummarizedExperiment_Subset_noBad_Filtered)$GeneSymbol=="Pval
b",]

#Plotting a boxplot of the expression of Pvalb Log2 CPM across
antidepressant groups:
boxplot(Pvalb~SummarizedExperiment_Subset_noBad_Filtered$treatment,
xlab="", ylab="Log2 CPM", main="Pvalb", las=2)

#We can add jittered data points to our graph so that we can see the
values for the individual samples in each group.
#To do this, we use the function stripchart and the exact same y~x
formula that we used for the boxplot
#The parameter pch determines the size of the data points.
#The parameter "add" places the points on top of the boxplot that we
already created.
stripchart(Pvalb~SummarizedExperiment_Subset_noBad_Filtered$treatment,
 pch = 19, method = "jitter", jitter = 0.2, vertical = TRUE, add=TRUE)
```

15.4 Design the statistical model that will be used for running the differential expression analyses. This model should include the variable of interest (e.g., experimental treatment) as well as any desired biological or technical co-variates.

Command

Example R Code: Defining the model matrix for the differential expression analysis

```
#Example 1: Creating a statistical model matrix that includes
experimental treatment (which is a factor, with the reference level
defined as saline) and the technical covariate of Library Size:
design <- model.matrix(~treatment_factor+LibrarySize,
data=colData(SummarizedExperiment_Subset_noBad_Filtered))

#Example 2: Creating a statistical model matrix that includes
experimental treatment (which is a factor, with the reference level
defined as saline), the technical covariate of Library Size, and
their interaction:
design2 <- model.matrix(~treatment_factor+LibrarySize +
treatment_factor*LibrarySize,
data=colData(SummarizedExperiment_Subset_noBad_Filtered))
```

15.5 Test out the differential expression model using a simple Type III linear regression analysis. Within the output:
- The Intercept should be the reference group (in this example, saline)
- The "Estimate" is the Log2FC (the difference between the group of interest and the reference group in Log2 expression units). A positive value means that the group of interest has greater expression than the reference group, a negative value means that the group of interest has lower expression than the reference group. This should match what you see in your boxplot.
- The "Pr(>|t|)" is the p-value showing the probability of this effect arising under conditions of random chance. Ignore this for now - a real differential expression analysis performed on gene expression data normally has additional analysis steps to control for extreme values or mean-variance trends in the data.

Command

Example R code: Running a basic linear regression as to check that the differential expression model is functioning

```
#Running a basic linear regression to double check our differential
expression model:

#We have to tell the linear model (lm) function to not add an
intercept because the model matrix already defines the intercept
summary.lm(lm(Pvalb~0+design))
```

Differential Expression Analysis

16 To run the differential expression analysis, we will use the *limma* differential expression analysis pipeline (for microarray data) or *limma-trend* differential expression analysis pipeline (for RNA-Seq data).

Note

Similar to Gemma, we have chosen to use the *limma* pipeline because it allows us to run analogous analysis procedures for the microarray and RNA-Seq datasets.
Limma also has a variety of other desirable features, including the ability to use more complex differential expression models (if desired).

CITATION

Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W, Smyth GK (2015). limma powers differential expression analyses for RNA-sequencing and microarray studies..
LINK
https://doi.org/10.1093/nar/gkv007

16.1 Install and load the R package *limma:*

Note

This protocol was designed using the version of *limma* (v.3.17) available on 06/01/2023. It is likely to work with other versions of the package.

Command

Example R Code: Install and Load the package limma

```
if (!require("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("limma")

library(limma)
```

Software

R package Limma NAME

Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W, Smyth GK DEVELOPER

https://bioconductor.org/packages/release/bioc/html/limma.html SOURCE LINK

16.2 Fit the differential expression model to the entire log2 gene expression dataset and apply an empirical Bayes correction using the eBayes() function to reduce the impact of outliers in small samples

The code for this step will depend on whether your data is from a microarray or RNA-Seq experiment. For the RNA-Seq datasets, we include a correction for the mean-variance relationship present in the dataset (trend=TRUE).

Command

Example R Code: Running a differential expression analysis on RNA-Seq data using limma-trend

```
#Applying the differential expression model to all genes using limma:

fit <- lmFit(ExpressionData_Subset_noBad_Filtered, design)

#Adding an eBayes correction to help reduce the influence of
outliers/small sample size on estimates
efit <- eBayes(fit, trend=TRUE)
```

Command

Example R Code: Running a differential expression analysis on microarray data using limma

```
#Applying the differential expression model to all genes using limma:

fit <- lmFit(ExpressionData_Subset_noBad_Filtered, design)

#Adding an eBayes correction to help reduce the influence of
outliers/small sample size on estimates
efit <- eBayes(fit)
```

16.3 The limma results need to be written out into the working directory, along with their accompanying gene annotation.
Note that these results and annotation should either be written out into a directory specific to this dataset or given dataset-specific filenames - otherwise the files will be overwritten when another dataset is analyzed!

Note

Please note that the p-values are corrected for multiple comparisons (false discovery rate or FDR) as part of this code, but only the Log2 Fold Changes (Log2FC) and T-statistics will be used in later meta-analysis steps.

Command

Example R Code: Applying a FDR correction and writing out the limma differential expression results

```
#Applying a false discovery rate (FDR) correction and writing out the
results as a .txt file:
write.fit(efit, adjust="BH", file="Limma_results.txt")

#Writing out the gene annotation for the results as a .csv file:
write.csv(rowData(SummarizedExperiment_Subset_noBad_Filtered),
"Annotation_LimmaResults.csv")
```

Result Extraction: Differential Expression Results from Each Dataset

17 Extract the differential expression results for each dataset and prepare the results for the meta-analysis. The following code is intended to be applied to the differential expression results from one dataset at a time.
It may be easiest to conduct these steps within a new R workspace.

Note

A detailed version of example code can be found here: https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_GemmaDatasets%20/2023_Formatting_LimmaResults_forMetaAnalysis.R

17.1 The differential expression results for the dataset should be read into R from the working directory.

Command

Example R Code: Reading in the differential expression results for a dataset from the working directory

```
#Example code: Reading in a file containing the differential
expression results for dataset GSE92718
#Note: This code will only work if the working directory is set to
the folder that contains this file of differential expression results
#Note: Depending on file format, you may need to use read.delim
instead of read.csv:
TempResultsJoined<-
read.csv("Annotation_LimmaResults_GSE92718_all.csv", header=TRUE,
stringsAsFactors = FALSE)
```

17.2 Within the joined file containing the differential expression results for a dataset, rows that lack unambiguous gene symbol annotation (including “”, “null”, '\\|' in the gene symbol) need to be counted and excluded.

Command

R Function: Remove rows of data that have missing or ambiguous Entrez ID gene annotation

```
FilteringDEResults_GoodAnnotation<-function(TempResultsJoined){

  print("# of rows in results")
  print(nrow(TempResultsJoined))

  print("# of rows with missing NCBI annotation:")

print(sum(TempResultsJoined$NCBIid==""|TempResultsJoined$NCBIid=="null
"))

  print("# of rows with NA NCBI annotation:")
  print(sum(is.na(TempResultsJoined$NCBIid)))

  print("# of rows with missing Gene Symbol annotation:")

print(sum(TempResultsJoined$GeneSymbol==""|TempResultsJoined$GeneSymbo
l=="null"))

  print("# of rows mapped to multiple NCBI_IDs:")
  print(length(grep('\\|', TempResultsJoined$NCBIid)))

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

  #I only want the subset of data which contains rows that do not
contain an NCBI EntrezID of ""
  TempResultsJoined_NoNA<-
TempResultsJoined[(TempResultsJoined$NCBIid==""|TempResultsJoined$NCBI
id=="null")==FALSE & is.na(TempResultsJoined$NCBIid)==FALSE,]

  if(length(grep('\\|', TempResultsJoined_NoNA$NCBIid))==0){
    TempResultsJoined_NoNA_NoMultimapped<<-TempResultsJoined_NoNA
  }else{
    #I only want rows annotated with a single Gene Symbol (no pipe):
    TempResultsJoined_NoNA_NoMultimapped<<-TempResultsJoined_NoNA[-
(grep('\\|', TempResultsJoined_NoNA$NCBIid)),]
  }

  print("# of rows with good annotation")
```

```
  print("# of rows with good annotation")
  print(nrow(TempResultsJoined_NoNA_NoMultimapped))

  #For record keeping (sometimes useful for troubleshooting later)
  write.csv(TempResultsJoined_NoNA_NoMultimapped,
"TempResultsJoined_NoNA_NoMultimapped.csv")

  rm(TempResultsJoined_NoNA)

  print("Outputted object: TempResultsJoined_NoNA_NoMultimapped")
}
```

Command

Example Usage: Remove rows of data that have missing or ambiguous Entrez ID gene annotation

```
FilteringDEResults_GoodAnnotation(TempResultsJoined)
# [1] "# of rows with missing NCBI annotation:"
# [1] NA
# [1] "# of rows with missing Gene Symbol annotation:"
# [1] 5
# [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] 16867
# [1] "Outputted object: TempResultsJoined_NoNA_NoMultimapped"
```

17.3 The specific statistical contrasts of interest (group comparisons) that are related to your research question should be identified within the differential expression results. The Log2 Fold Changes (Log2FC - i.e., group differences in log2 gene expression) and t-statistics for these statistical contrasts need to be extracted into their own matrix. The columns in these two matrices are then given descriptive names that include the dataset ids.

Command

Example R Code: Extract and rename relevant Log2FC and Tstatistic columns

```
#This code will be dataset specific
#We need to extract the Log2FC ("Coef") and T-statistic ("t.")
columns for the statistical contrasts relevant to our meta-analysis
and place them into their own matrix

TempResultsJoined_NoNA_NoMultimapped_FoldChanges<-
cbind(TempResultsJoined_NoNA_NoMultimapped$Coef.interaction.stress.x.f
emale)

TempResultsJoined_NoNA_NoMultimapped_Tstats<-
cbind(TempResultsJoined_NoNA_NoMultimapped$t.interaction.stress.x.fema
le)


#Making the row names for the Log2FC and Tstat matrices the Entrez ID
gene annotation:
row.names(TempResultsJoined_NoNA_NoMultimapped_FoldChanges)<-
TempResultsJoined_NoNA_NoMultimapped$NCBIid

row.names(TempResultsJoined_NoNA_NoMultimapped_Tstats)<-
TempResultsJoined_NoNA_NoMultimapped$NCBIid

#Let's rename our columns to something nicer describing the effect of
interest:
#Note - we later discovered that this name needs to include the
dataset identifier (GSEID#) for later joining and plotting purposes

ComparisonsOfInterest<-c("GSE85136_StressEffects_InteractionWSex")

colnames(TempResultsJoined_NoNA_NoMultimapped_FoldChanges)<-
ComparisonsOfInterest
colnames(TempResultsJoined_NoNA_NoMultimapped_Tstats)<-
ComparisonsOfInterest
```

17.4 The standard error (SE) can be calculated using the Log2FC and T-statistic (Tstat) in the differential expression output for each row. *(This will be performed by the function in step 17.6)*

Note

Log2FC/Tstat=SE

17.5 The sampling variance (SV) for each Log2FC can then be calculated by squaring the standard error (SE^2). *(This will be performed by the function in step 17.6)*

Note

As discussed in the online materials for the *metafor* package: *https://www.metafor-project.org/doku.php/tips:input_to_rma_function*, Viechtbauer 2024.

17.6 The differential expression results for the dataset are then placed into an object in the environment that is named "DEResults_" followed by the dataset name.

Command

R Function: Extracting dataset differential expression results for meta-analysis

```
ExtractingDEResults<-function(GSE_ID,
TempResultsJoined_NoNA_NoMultimapped_FoldChanges,
TempResultsJoined_NoNA_NoMultimapped_Tstats){

  #We calculate the standard error by dividing the log2FC by the tstat
  TempResultsJoined_NoNA_NoMultimapped_SE<-
TempResultsJoined_NoNA_NoMultimapped_FoldChanges/TempResultsJoined_NoN
A_NoMultimapped_Tstats
  str(TempResultsJoined_NoNA_NoMultimapped_SE)

  #For running our meta-analysis, we are actually going to need the
sampling variance instead of the standard error
  #The sampling variance is just the standard error squared.

  TempResultsJoined_NoNA_NoMultimapped_SV<-
(TempResultsJoined_NoNA_NoMultimapped_SE)^2
  str(TempResultsJoined_NoNA_NoMultimapped_SV)

  TempMasterResults<-
list(Log2FC=TempResultsJoined_NoNA_NoMultimapped_FoldChanges,
Tstat=TempResultsJoined_NoNA_NoMultimapped_Tstats,
SE=TempResultsJoined_NoNA_NoMultimapped_SE,
SV=TempResultsJoined_NoNA_NoMultimapped_SV)

  assign(paste("DEResults", GSE_ID, sep="_"), TempMasterResults,
envir = as.environment(1))

  print(paste("Output: Named DEResults", GSE_ID, sep="_"))

  rm(TempMasterResults, TempResultsJoined_NoNA_NoMultimapped_SV,
TempResultsJoined_NoNA_NoMultimapped_SE,
TempResultsJoined_NoNA_NoMultimapped_FoldChanges,
TempResultsJoined_NoNA_NoMultimapped_Tstats)

}
```

Command

Example R Function Usage: Extracting dataset differential expression results for meta-analysis

```
#Example usage for the extracting results function:

ExtractingDEResults(GSE_ID="GSE92718",
TempResultsJoined_NoNA_NoMultimapped_FoldChanges,
TempResultsJoined_NoNA_NoMultimapped_Tstats)
# num [1:16862, 1] 0.1655 0.0943 0.1069 0.1443 0.0382 ...
# - attr(*, "dimnames")=List of 2
# ..$ : chr [1:16862] "13654" "215418" "13656" "19252" ...
# ..$ : chr "GSE92718_CuffOperation_vs_Ctrl"
# num [1:16862, 1] 0.02739 0.00889 0.01142 0.02083 0.00146 ...
# - attr(*, "dimnames")=List of 2
# ..$ : chr [1:16862] "13654" "215418" "13656" "19252" ...
# ..$ : chr "GSE92718_CuffOperation_vs_Ctrl"
# [1] "Output: Named DEResults_GSE92718"


#################

#Cleaning up our environment before moving on to the next dataset:

rm(TempResultsJoined, TempResultsJoined_NoNA_NoMultimapped,
TempResultsJoined_NoNA_NoMultimapped_FoldChanges,
TempResultsJoined_NoNA_NoMultimapped_Tstats, ComparisonsOfInterest)
```

Command

Example R Function Output: Extracting dataset differential expression results for meta-analysis

```
str(DEResults_GSE92718)
# List of 4
# $ Log2FC: num [1:16862, 1] 1.386 0.668 0.72 0.974 0.245 ...
# ..- attr(*, "dimnames")=List of 2
# .. ..$ : chr [1:16862] "13654" "215418" "13656" "19252" ...
# .. ..$ : chr "GSE92718_CuffOperation_vs_Ctrl"
# $ Tstat : num [1:16862, 1] 8.37 7.09 6.74 6.75 6.41 ...
# ..- attr(*, "dimnames")=List of 2
# .. ..$ : chr [1:16862] "13654" "215418" "13656" "19252" ...
# .. ..$ : chr "GSE92718_CuffOperation_vs_Ctrl"
# $ SE    : num [1:16862, 1] 0.1655 0.0943 0.1069 0.1443 0.0382 ...
# ..- attr(*, "dimnames")=List of 2
# .. ..$ : chr [1:16862] "13654" "215418" "13656" "19252" ...
# .. ..$ : chr "GSE92718_CuffOperation_vs_Ctrl"
# $ SV    : num [1:16862, 1] 0.02739 0.00889 0.01142 0.02083 0.00146
...
# ..- attr(*, "dimnames")=List of 2
# .. ..$ : chr [1:16862] "13654" "215418" "13656" "19252" ...
# .. ..$ : chr "GSE92718_CuffOperation_vs_Ctrl"

#Sanity Check:

#Calculating the SE from the Log2FC and Tstat for the first gene(row)
1.386/8.37
#[1] 0.1655914

#Calculating the SV for the first gene
0.1655914^2
#[1] 0.02742051
#Looks good (the decimal differences are probably just due to
rounding)

#Calculating the SE from the Log2FC and Tstat for the second gene(row)
0.668/7.09
#[1] 0.09421721

#Calculating the SV for the second gene
0.09421721^2
#[1] 0.008876883
```

```
#[1] 0.0000000005
#Looks good (the decimal differences are probably just due to
rounding)
```

Aligning Results Across Datasets

18 The results (Log2FC, SV) from the different datasets are then compiled into a single data frame.

18.1 Install the *plyr* package.

Note

- This protocol was designed using the version of plyr (v.1.8.7) available on 06/01/2022. *It may not work using later versions.*

Software

R package plyr NAME

Wickham DEVELOPER

http://cran.r-project.org/web/packages/plyr/index.html SOURCE LINK

CITATION

Wickham H (2011). The Split-Apply-Combine Strategy for Data Analysis. Journal of Statistical Software.

LINK

10.18637/jss.v040.i01

18.2 The results (Log2FC, SV) from the different statistical contrasts of interest and datasets should then be compiled into a single data frame. To do this, results from studies performed on the same species are first aligned (joined) using Entrez ID (“NCBI_ID”).

Command

R Functions: Aligning datasets from the same species (rats, mice)

```
AligningRatDatasets<-function(ListOfRatDEResults){

  Rat_MetaAnalysis_FoldChange_Dfs<-list()

  for(i in c(1:length(ListOfRatDEResults))){
    Rat_MetaAnalysis_FoldChange_Dfs[[i]]<-
data.frame(Rat_EntrezGene.ID=row.names(ListOfRatDEResults[[i]]
[[1]]),ListOfRatDEResults[[i]][[1]], stringsAsFactors=FALSE)
  }

  print("Rat_MetaAnalysis_FoldChange_Dfs:")
  print(str(Rat_MetaAnalysis_FoldChange_Dfs))

  Rat_MetaAnalysis_FoldChanges<<-
join_all(Rat_MetaAnalysis_FoldChange_Dfs, by="Rat_EntrezGene.ID",
type="full")
  #This function could be join_all (if there are more than 2
datasets) or merge/merge_all (if the plyr package isn't working)

  print("Rat_MetaAnalysis_FoldChanges:")
  print(str(Rat_MetaAnalysis_FoldChanges))

  Rat_MetaAnalysis_SV_Dfs<-list()

  for(i in c(1:length(ListOfRatDEResults))){
    Rat_MetaAnalysis_SV_Dfs[[i]]<-
data.frame(Rat_EntrezGene.ID=row.names(ListOfRatDEResults[[i]]
[[4]]),ListOfRatDEResults[[i]][[4]], stringsAsFactors=FALSE)
  }

  print("Rat_MetaAnalysis_SV_Dfs:")
  print(str(Rat_MetaAnalysis_SV_Dfs))

  Rat_MetaAnalysis_SV<<-join_all(Rat_MetaAnalysis_SV_Dfs,
by="Rat_EntrezGene.ID", type="full")
  #This function could be join_all (if there are more than 2
datasets) or merge/merge_all (if the plyr package isn't working)

  print("Rat_MetaAnalysis_SV:")
  print(str(Rat_MetaAnalysis_SV))
```

```
  rm(Rat_MetaAnalysis_SV_Dfs, Rat_MetaAnalysis_FoldChange_Dfs)
}



###########

AligningMouseDatasets<-function(ListOfMouseDEResults){

  Mouse_MetaAnalysis_FoldChange_Dfs<-list()

  for(i in c(1:length(ListOfMouseDEResults))){
    Mouse_MetaAnalysis_FoldChange_Dfs[[i]]<-
data.frame(Mouse_EntrezGene.ID=row.names(ListOfMouseDEResults[[i]]
[[1]]),ListOfMouseDEResults[[i]][[1]], stringsAsFactors=FALSE)
  }

  print("Mouse_MetaAnalysis_FoldChange_Dfs:")
  print(str(Mouse_MetaAnalysis_FoldChange_Dfs))

  Mouse_MetaAnalysis_FoldChanges<<-
join_all(Mouse_MetaAnalysis_FoldChange_Dfs, by="Mouse_EntrezGene.ID",
type="full")
  #This function could be join_all (if there are more than 2
datasets) or merge/merge_all (if the plyr package isn't working)

  print("Mouse_MetaAnalysis_FoldChanges:")
  print(str(Mouse_MetaAnalysis_FoldChanges))

  Mouse_MetaAnalysis_SV_Dfs<-list()

  for(i in c(1:length(ListOfMouseDEResults))){
    Mouse_MetaAnalysis_SV_Dfs[[i]]<-
data.frame(Mouse_EntrezGene.ID=row.names(ListOfMouseDEResults[[i]]
[[4]]),ListOfMouseDEResults[[i]][[4]], stringsAsFactors=FALSE)
  }

  print("Mouse_MetaAnalysis_SV_Dfs:")
  print(str(Mouse_MetaAnalysis_SV_Dfs))

  Mouse_MetaAnalysis_SV<<-join_all(Mouse_MetaAnalysis_SV_Dfs,
by="Mouse_EntrezGene.ID", type="full")
  #This function could be join_all (if there are more than 2
datasets) or merge/merge_all (if the plyr package isn't working)
```

```
  print("Mouse_MetaAnalysis_SV:")
  print(str(Mouse_MetaAnalysis_SV))
```

Command

```
}
```

Example R Function Usage: Aligning datasets from the same species (rats, mice)

```
#Example usage for rat datasets:

ListOfRatDEResults<-list(DEResults_GSE172133)

AligningRatDatasets(ListOfRatDEResults)


#Example usage for mouse datasets:

ListOfMouseDEResults<-list(DEResults_GSE85136, DEResults_GSE92718)

AligningMouseDatasets(ListOfMouseDEResults)
```

18.3 Next studies performed on different species (rats vs. mice) need to be aligned (joined) using a gene orthology database.

Note

To align results between species, we referenced the Jackson Labs Mouse ortholog database (downloaded from **https://www.informatics.jax.org/downloads/reports/HOM_AllOrganism.rpt** on July 27, 2023). In preparation for the analysis, Dr. Hagenauer extracted rows of annotation from the ortholog database for laboratory mice and rats, and aligned this species-specific annotation by database key ("DB.Class.Key") using join(x, type= "full", match= "all") from the plyr package. If a gene mapped to more than two orthologs in the other species, that orthology relationship (row) was removed from the data frame. Then the column containing the Entrez ID information from each species was extracted, and combined into a third column ("MouseEntrezID _RatEntrezID") to create a single combined identifier for the orthology relationship.

This orthology database can be downloaded here:
https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_GemmaDatasets%20/MouseVsRat_NCBI_Entrez_JacksonLab_20230727.csv

The code for preparing the orthology database for our analyses can be found here:
https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_GemmaDatasets%20/2023_FormattingRatMouseOrthologDatabase.R

Command

Example R Code: Aligning differential expression results from different species (rats vs. mice)

```
#Read in the rat vs. mouse gene orthology database from the working
directory:
MouseVsRat_NCBI_Entrez<-
read.csv("MouseVsRat_NCBI_Entrez_JacksonLab_20230727.csv",
header=TRUE, stringsAsFactors = FALSE, row.names=1,
colClasses=c("character", "character", "character"))

#Join the rat vs. mouse gene orthology information to the mouse
Log2FC results:
Mouse_MetaAnalysis_FoldChanges_wOrthologs<-
join(MouseVsRat_NCBI_Entrez, Mouse_MetaAnalysis_FoldChanges,
by="Mouse_EntrezGene.ID", type="full")

#Join the rat vs. mouse gene orthology information to the mouse
sampling variance results:
Mouse_MetaAnalysis_SV_wOrthologs<-join(MouseVsRat_NCBI_Entrez,
Mouse_MetaAnalysis_SV, by="Mouse_EntrezGene.ID", type="full")


#If there are rat datasets, join the mouse results with the rat
results (for both the Log2FC and SV)
MetaAnalysis_FoldChanges<-
join(Mouse_MetaAnalysis_FoldChanges_wOrthologs,
Rat_MetaAnalysis_FoldChanges, by="Rat_EntrezGene.ID", type="full")

MetaAnalysis_SV<-join(Mouse_MetaAnalysis_SV_wOrthologs,
Rat_MetaAnalysis_SV, by="Rat_EntrezGene.ID", type="full")


#If there *aren't* any rat datasets, just rename the mouse Log2FC and
SV results in a manner so that the downstream code will still work:

MetaAnalysis_FoldChanges<-Mouse_MetaAnalysis_FoldChanges_wOrthologs

MetaAnalysis_SV<-Mouse_MetaAnalysis_SV_wOrthologs


#For simplicity's sake, I'm going to replace that Mouse-Rat Entrez
annotation
```

```
annotation
#Because it is missing entries for any genes in the datasets that
*don't* have orthologs
MetaAnalysis_FoldChanges$MouseVsRat_EntrezGene.ID<-
paste(MetaAnalysis_FoldChanges$Mouse_EntrezGene.ID,
MetaAnalysis_FoldChanges$Rat_EntrezGene.ID, sep="_")
MetaAnalysis_SV$MouseVsRat_EntrezGene.ID<-
paste(MetaAnalysis_SV$Mouse_EntrezGene.ID,
MetaAnalysis_SV$Rat_EntrezGene.ID, sep="_")
```

Meta-Analysis of Differential Expression Results

19 Meta-Analysis of Differential Expression Results: This coding is all performed within the R environment using Rstudio.

Note

Example code can be found here: https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_GemmaDatasets%20/2023_MetaAnalysisCode.R

19.1 Because we are considering differential expression results that are derived from a variety of different transcriptional profiling platforms with varying transcript/probe representation and sensitivity, each dataset will include differential expression results for a slightly different set of genes.

Decide on a minimum number of differential expression results that need to be present to run a meta-analysis for each gene (i.e., a minimum number of Log2FC values that are not NAs).

Note

This decision should be guided by considerations related to sample size and the minimum number of datasets necessary to represent the diversity of subjects and methods available in your datasets. If you only have a small number of datasets included in your meta-analysis (<5 datasets), you may want to require that a gene be represented in all of them to be included in the meta-analysis.

19.2 Install the R package metafor

Software

R package metafor

NAME

Wolfgang Viechtbauer

DEVELOPER

http://CRAN.R-project.org/package=metafor

SOURCE LINK

Note

This protocol was designed using the version of metafor (v.4.0-0) available on 06/01/2023. It is likely to work using other versions of the package.

CITATION

Viechtbauer, W (2010). Conducting meta-analyses in R with the metafor package. Journal of Statistical Software.

LINK

https://doi.org/10.18637/jss.v036.i03

19.3 To run a meta-analysis for every gene meeting our criteria for a minimum number of differential expression results, we use a *for loop* to run the same series of calculations for each row of our Log2FC data frame and accompanying SV data frame.

Note

(This process is performed by the function in 19.5)

19.4 For each row (gene), we first determine whether the gene meets the criteria of having our pre-specified minimum number of differential expression results (Log2FC values). This is done using a conditional *if-else* statement.

Note

(This process is performed by the function in 19.5)

19.5 If the row (gene) meets our criteria, we use the function *rma()* from the metafor package to fit a random effects meta-analysis model to the Log2FC values (y_i) and accompanying SV (v_i).

Note

Due to the relatively small number of differential expression results that are available for most topics, we have chosen to use the simplest model possible for the meta-analysis (an intercept-only model) as our main outcome.

The downside to this approach is an inadequate accounting for the covariance of results derived from the same dataset (*e.g.,* a dataset with multiple drug treatments compared to a control treatment).

This approach also means that we do not attempt to quantify potentially impactful sources of
heterogeneity within the subject characteristics or methodology of the studies. If there is a larger number of datasets (e.g., 4-5 datasets representing each of the potential sources of heterogeneity), it may be worth exploring models including these variables as secondary outcomes.

An example of a more complicated meta-analysis that was designed to include a predictor variable (sex) can be found here:
https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_GemmaDatasets%20/2023_ExampleCode_FancierMetaAnalysis_Christabel.R

Command

R Function: Running a simple (intercept-only) meta-analysis on the data frame of differential expression results

```
RunBasicMetaAnalysis<-function(NumberOfComparisons, CutOffForNAs,
MetaAnalysis_FoldChanges, MetaAnalysis_SV){

  MetaAnalysis_FoldChanges_NAsPerRow<-
apply(MetaAnalysis_FoldChanges[,-c(1:3)], 1, function(y)
sum(is.na(y)))

  print("Table of # of NAs per Row (Gene):")
  print(table(MetaAnalysis_FoldChanges_NAsPerRow))

  MetaAnalysis_FoldChanges_ForMeta<<-
MetaAnalysis_FoldChanges[MetaAnalysis_FoldChanges_NAsPerRow<CutOffForN
As,]
  MetaAnalysis_SV_ForMeta<<-
MetaAnalysis_SV[MetaAnalysis_FoldChanges_NAsPerRow<CutOffForNAs,]

  print("MetaAnalysis_FoldChanges_ForMeta:")
  print(str(MetaAnalysis_FoldChanges_ForMeta))

  #I'm going to make an empty matrix to store the results of my meta-
analysis:
  metaOutput<-matrix(NA, nrow(MetaAnalysis_FoldChanges_ForMeta), 6)

  #And then run a loop that run's a meta-analysis on the differential
expression results (columns 2-10) for each gene (row):
  for(i in c(1:nrow(MetaAnalysis_FoldChanges_ForMeta))){

    effect<-as.numeric(MetaAnalysis_FoldChanges_ForMeta[i,-c(1:3)])
    var<-as.numeric(MetaAnalysis_SV_ForMeta[i,-c(1:3)])

    #I added a function tryCatch that double-checks that the meta-
analysis function (rma) doesn't produce errors (which breaks the
loop):
    skip_to_next <- FALSE
    tryCatch(TempMeta<-rma(effect, var), error = function(e)
{skip_to_next <<- TRUE})

    if(skip_to_next){}else{
      TempMeta<-rma(effect, var)
```

```
      TempMeta<-rma(effect, var)
      metaOutput[i, 1]<-TempMeta$b #gives estimate Log2FC
      metaOutput[i, 2]<-TempMeta$se #gives standard error
      metaOutput[i, 3]<-TempMeta$pval #gives pval
      metaOutput[i, 4]<-TempMeta$ci.lb #gives confidence interval
lower bound
      metaOutput[i, 5]<-TempMeta$ci.ub #gives confidence interval
upper bound
      metaOutput[i, 6]<-NumberOfComparisons-sum(is.na(effect))#Number
of comparisons with data
      rm(TempMeta)
    }
    rm(effect, var)
  }

  colnames(metaOutput)<-c("Log2FC_estimate", "SE", "pval", "CI_lb",
"CI_ub", "Number_Of_Comparisons")
  row.names(metaOutput)<-MetaAnalysis_FoldChanges_ForMeta[,3]

  metaOutput<<-metaOutput
  MetaAnalysis_Annotation<<-MetaAnalysis_FoldChanges_ForMeta[,c(1:3)]
  return(metaOutput)
  return(MetaAnalysis_Annotation)

  print("metaOutput:")
  print(str(metaOutput))

  print("Top of metaOutput:")
  print(head(metaOutput))

  print("Bottom of metaOutput")
  print(tail(metaOutput))

}
```

Command

Example Usage: Running a simple (intercept-only) meta-analysis on the data frame of differential expression results

```
#We can only run a meta-analysis if there are differential expression results from more than one comparison.
#Since I know that the differential expression results from the same study (dataset) are artificially correlated, I would actually prefer that there are results from more than one dataset.

#How many genes satisfy this criteria?

#This code caculates the number of NAs in each row:
MetaAnalysis_FoldChanges_NAsPerRow<-apply(MetaAnalysis_FoldChanges[,-c(1:3)], 1, function(y) sum(is.na(y)))

#I'm going to make a histogram of the results because I'm curious to see how they are distributed
hist(MetaAnalysis_FoldChanges_NAsPerRow)

#Or, since there are a limited number of integer answers (0-3), I could make a table of the results:
table(MetaAnalysis_FoldChanges_NAsPerRow)
# 0     1     2     3
# 11043  4823 13346   875

#Let's try running a meta-analysis using genes that were found in at least 2 sets of differential expression results
#Since there are 3 sets of differential expression results, that means that the genes that we are including need to have 1 or fewer NAs in their results
#I set this conservatively, because there are so few studies in this meta-analysis.
#2 NA is too many


#Example Usage:

NumberOfComparisons=3
CutOffForNAs=2
#I have 3 comparisons
#2 NA is too many
```

```
metaOutput<-RunBasicMetaAnalysis(NumberOfComparisons, CutOffForNAs,
MetaAnalysis_FoldChanges, MetaAnalysis_SV)
#Note: this function can take a while to run, especially if you have
a lot of data
#Plug in your computer, take a break, grab some coffee...

# [1] "Table of # of NAs per Row (Gene):"
# MetaAnalysis_FoldChanges_NAsPerRow
# 0     1     2     3
# 11043  4823 13346   875
# [1] "MetaAnalysis_FoldChanges_ForMeta:"
# 'data.frame': 15866 obs. of  6 variables:
# $ Rat_EntrezGene.ID            : chr  "498097" "114521" "360576"
"24628" ...
# $ Mouse_EntrezGene.ID          : chr  "68980" "21665" "237858"
"18591" ...
# $ MouseVsRat_EntrezGene.ID     : chr  "68980_498097" "21665_114521"
"237858_360576" "18591_24628" ...
# $ StressEffects_InteractionWSex: num  0.0831 -0.0504 -0.7557
-0.0782 0.161 ...
# $ CuffOperation_vs_Ctrl        : num  0.01656 0.00964 -0.0427
0.01155 NA ...
# $ ChronicConstriction_vs_Ctrl  : num  -0.0608 0.0631 -0.4516 0.4982
1.4642 ...
# NULL

#Example Output:

str(metaOutput)
# num [1:15866, 1:6] 0.012 0.0143 -0.1497 0.157 1.3608 ...
# - attr(*, "dimnames")=List of 2
# ..$ : chr [1:15866] "68980_498097" "21665_114521" "237858_360576"
"18591_24628" ...
# ..$ : chr [1:6] "Log2FC_estimate" "SE" "pval" "CI_lb" ...

head(metaOutput)
# Log2FC_estimate         SE       pval       CI_lb      CI_ub
# 68980_498097       0.01203600 0.05083559 0.81284048 -0.08759993
0.11167192
# 21665_114521       0.01426912 0.03385763 0.67343034 -0.05209062
0.08062887
# 237858_360576     -0.14968156 0.17385300 0.38925664 -0.49042717
0.19106405
# 18591_24628        0.15695379 0.17760171 0.37683643 -0.19113918
0.50504675
# 229323_688737      1.36081008 0.45349519 0.00269346  0.47197584
```

```
2.24964432
# 240549_309227        0.04938634 0.14315287 0.73010174 -0.23118814
```

19.6 If a random effects model for the Log2FC values for a gene (row) produces a convergence error (i.e., there is not a stable model fit), that row of results will be recorded as NA.

Applying a False Discovery Rate (FDR) Correction to the Meta-Analysis Results

20 False Discovery Rate Correction: Since we have run meta-analyses for thousands of genes, we correct the p-values for false discovery rate (FDR) using the the Benjamini-Hochberg method. This coding is all performed within the R environment using Rstudio.

Note

To properly interpret the p-values produced by our meta-analyses for each gene, we need to take into account the fact that we have run thousands of statistical tests (i.e., one statistical test for each gene for thousands of genes). Therefore, we are likely to get a large number of results that are "significant" using a traditional p-value threshold (alpha=0.05) just due to random chance. This is called a multiple comparisons correction or p-value adjustment. We use a type of correction called False Discovery Rate (FDR) or q-value. It is also sometimes called a "Benjamini–Hochberg" adjustment after its originators.

20.1 Install the R package *multtest*

Software

R package multtest

NAME

Katherine S. Pollard, Houston N. Gilbert, Yongchao Ge, Sandra Taylor, Sandrine Dudoit

DEVELOPER

http://bioconductor.org/packages/release/bioc/html/multtest.html

SOURCE LINK

Note

This protocol was designed using the version of multtest (v.2.8.0) available on 06/01/2023. It is likely to work using other versions of the package.

CITATION

Pollard K.S., Dudoit S., van der Laan M.J. (2005). Multiple Testing Procedures: R multtest Package and Applications to Genomics, in Bioinformatics and Computational Biology Solutions Using R and Bioconductor. Springer: Bioinformatics and Computational Biology Solutions Using R and Bioconductor .
LINK
10.1007/0-387-29362-0

Command

Example R code: Installing R package multtest

```
#Installing and loading code package multtest

if (!require("BiocManager", quietly = TRUE))
install.packages("BiocManager")
BiocManager::install("multtest")
library(multtest)
```

20.2 We can add some additional gene annotation to our results during this step too.

Note

This database of additional gene annotation can be downloaded here: https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_GemmaDatasets%20/HOM_MouseVsRat.csv

This additional gene annotation was obtained from the Jackson Labs Mouse ortholog database (downloaded from **https://www.informatics.jax.org/downloads/reports/HOM_AllOrganism.rpt** on July 27, 2023) and reformatted for easy use while preparing the rat-mouse ortholog database used in protocol step 18.3 above.

The code used to prepare the database for our analyses can be found here: https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_GemmaDatasets%20/2023_FormattingRatMouseOrthologDatabase.R

Command

Example R code: Reading in additional mouse/rat gene annotation

```
#We can add some additional gene annotation at this point too:

HOM_MouseVsRat <- read.csv("HOM_MouseVsRat.csv", header = TRUE,
row.names = 1)
HOM_MouseVsRat$Mouse_EntrezGene.ID <-
as.character(HOM_MouseVsRat$Mouse_EntrezGene.ID)
HOM_MouseVsRat$Rat_EntrezGene.ID <-
as.character(HOM_MouseVsRat$Rat_EntrezGene.ID)
```

Note

Note: It is important to make sure that R recognizes that Entrez ID is a character vector and not a numeric variable (i.e., the numbers used as Entrez IDs are not a meaningful quantity they are a label for the gene).

20.3 We correct the p-values for false discovery rate (FDR) using the the Benjamini-Hochberg method as applied by the *mt.rawp2adjp()* function within the multtest package.

Command

R Function: Applying an FDR correction to meta-analysis p-values

```
FalseDiscoveryCorrection<-function(metaOutput){

  tempPvalAdjMeta<-mt.rawp2adjp(metaOutput[,3], proc=c("BH"))

  metaPvalAdj<-tempPvalAdjMeta$adjp[order(tempPvalAdjMeta$index),]

  metaOutputFDR<-cbind(metaOutput, metaPvalAdj[,2])

  colnames(metaOutputFDR)[7]<-"FDR"

  metaOutputFDR<<-metaOutputFDR

  print("metaOutputFDR:")
  print(str(metaOutputFDR))

  write.csv(metaOutputFDR, "metaOutputFDR.csv")

  #a version of the output in order by p-value:
  metaOutputFDR_OrderbyPval<<-metaOutputFDR[order(metaOutputFDR[,3]),]

  #Let's write out a version of the output in order by p-value:
  write.csv(metaOutputFDR_OrderbyPval,
"metaOutputFDR_orderedByPval_wHDRFData.csv")

  print("Do we have any genes that are statistically significant
following false discovery rate correction?")
  print(sum(metaOutputFDR[,7]<0.10, na.rm=TRUE))

  print("What are the top results?")
  print(head(metaOutputFDR[order(metaOutputFDR[,3]),]))

  rm(tempPvalAdjMeta, metaPvalAdj)

}
```

Command

Example usage: Applying an FDR correction to meta-analysis p-values

```
#Example usage:

FalseDiscoveryCorrection(metaOutput)
# [1] "metaOutputFDR:"
# num [1:11562, 1:7] 0.06022 -0.01543 -0.06567 -0.07498 -0.00186 ...
# - attr(*, "dimnames")=List of 2
# ..$ : chr [1:11562] "A4galt" "Aaas" "Aacs" "Aadac" ...
# ..$ : chr [1:7] "Log2FC_estimate" "SE" "pval" "CI_lb" ...
# NULL
# [1] "Do we have any genes that are statistically significant
following false discovery rate correction?"
# [1] 797
# [1] "What are the top results?"
# Log2FC_estimate         SE         pval      CI_lb       CI_ub
Number_Of_Comparisons          FDR
# Parm1       -0.1798498 0.02389052 5.149049e-14 -0.2266743
-0.13302524                     7 5.953331e-10
# Lpl          0.2138084 0.03015666 1.341870e-12  0.1547024
0.27291434                     7 7.757348e-09
# Gabrr2       0.1965106 0.02859386 6.309657e-12  0.1404677
0.25255358                     7 2.431742e-08
# Pla2g5       0.1593702 0.02375177 1.948623e-11  0.1128176
0.20592280                     7 5.632494e-08
# Akap8l      -0.1174562 0.01848935 2.116479e-10 -0.1536947
-0.08121775                     7 4.894145e-07
# Dusp1       -0.4415588 0.07101670 5.045755e-10 -0.5807489
-0.30236858                     7 9.723171e-07
```

20.4 We will consider results to be statistically significant if they survive a threshold of 5% false discovery (FDR<0.05). If our meta-analyses do not produce any results surviving this threshold, we may consider results at a weaker threshold (FDR<0.10) with the understanding that a higher percent of these results are likely to be false discovery (10%).

Exploring Meta-Analysis Results

21 To explore our meta-analysis results, we first rank the results by p-value and examine the results surviving our false discovery rate correction (FDR<0.05). We will refer to these results as our "top genes".

21.1 Within the results surviving false discovery rate correction, count how many show increased expression ("upregulation") in response to our variable of interest and how many show decreased expression ("down-regulation").

Note

Within the meta-analysis results, a negative "estimate" (estimated Log2FC) means that the gene typically shows lower expression in response to our variable of interest, whereas a positive estimate (estimated Log2FC) means that a gene typically shows higher expression in response to our variable of interest.

21.2 To explore the meta-analysis results for any particular gene in more detail, use the function *forest.rma()* from the *metafor* package to make a forest plot illustrating the effect sizes (Log2FC) and confidence intervals for each study.

Command

R Function: Making forest plots to illustrate differential expression results across studies for a gene

```
MakeForestPlots<-function(EntrezIDAsCharacter){

  pdf(paste("ForestPlot_", EntrezIDAsCharacter, ".pdf", sep=""),
height=5, width=8)

  effect<-
as.numeric(MetaAnalysis_FoldChanges_ForMeta[MetaAnalysis_FoldChanges_F
orMeta$Mouse_EntrezGene.ID==EntrezIDAsCharacter,-c(1:3)])
  var<-
as.numeric(MetaAnalysis_SV_ForMeta[MetaAnalysis_FoldChanges_ForMeta$Mo
use_EntrezGene.ID==EntrezIDAsCharacter,-c(1:3)])

  forest.rma(rma(effect,
var),slab=colnames(MetaAnalysis_FoldChanges_ForMeta)[-c(1:3)],
xlim=c(-3, 3))

  mtext(paste(EntrezIDAsCharacter), line=-1.5, cex=2)
  dev.off()
}
```

Command

Example Usage: Making forest plots to illustrate differential expression results across studies for a gene

```
#Example Usage:

MakeForestPlots("215418")
```

Note

Note - this function currently uses Mouse Entrez ID (NCBI ID) as it's input.
It needs to be formatted as a character (not an integer) to work.
In the future, I will need to make a version of this function that accepts rat Entrez ID.

21.3 For the sake of easily including results from the project in future posters, presentations and publications, we have a template for quickly summarizing meta-analysis results.

Note

A Google Docs template for quickly illustrating and summarizing the meta-analysis results can be found here:
https://docs.google.com/presentation/d/1fAirnPu9zdu4uHRIuESeK5RTDxwFqiD8/edit?usp=sharing&ouid=106595687423493776462&rtpof=true&sd=true

21.4 Quick ways to learn about the functions associated with our top genes:

Basic functional summary:
GeneCards: **https://www.genecards.org/**
Rat Genome Database: **https://rgd.mcw.edu/**

Cell types that express the gene in the brain:
DropViz: **http://dropviz.org/**
MouseBrain.Org: **http://www.mousebrain.org/**

Regional distribution of the expression for the gene in the brain:
https://mouse.brain-map.org/search/index

CITATION

Saunders A, Macosko EZ, Wysoker A, Goldman M, Krienen FM, de Rivera H, Bien E, Baum M, Bortolin L, Wang S, Goeva A, Nemesh J, Kamitaki N, Brumbaugh S, Kulp D, McCarroll SA (2018). Molecular Diversity and Specializations among the Cells of the Adult Mouse Brain..
LINK
https://doi.org/10.1016/j.cell.2018.07.028

CITATION

Shimoyama M, De Pons J, Hayman GT, Laulederkind SJ, Liu W, Nigam R, Petri V, Smith JR, Tutaj M, Wang SJ, Worthey E, Dwinell M, Jacob H (2015). The Rat Genome Database 2015: genomic, phenotypic and environmental variations and disease..

LINK

https://doi.org/10.1093/nar/gku1026

CITATION

Zeisel A, Hochgerner H, Lönnerberg P, Johnsson A, Memic F, van der Zwan J, Häring M, Braun E, Borm LE, La Manno G, Codeluppi S, Furlan A, Lee K, Skene N, Harris KD, Hjerling-Leffler J, Arenas E, Ernfors P, Marklund U, Linnarsson S (2018). Molecular Architecture of the Mouse Nervous System..

LINK

https://doi.org/10.1016/j.cell.2018.06.021

CITATION

Lein ES, Hawrylycz MJ, Ao N, Ayres M, Bensinger A, Bernard A, Boe AF, Boguski MS, Brockway KS, Byrnes EJ, Chen L, Chen L, Chen TM, Chin MC, Chong J, Crook BE, Czaplinska A, Dang CN, Datta S, Dee NR, Desaki AL, Desta T, Diep E, Dolbeare TA, Donelan MJ, Dong HW, Dougherty JG, Duncan BJ, Ebbert AJ, Eichele G, Estin LK, Faber C, Facer BA, Fields R, Fischer SR, Fliss TP, Frensley C, Gates SN, Glattfelder KJ, Halverson KR, Hart MR, Hohmann JG, Howell MP, Jeung DP, Johnson RA, Karr PT, Kawal R, Kidney JM, Knapik RH, Kuan CL, Lake JH, Laramee AR, Larsen KD, Lau C, Lemon TA, Liang AJ, Liu Y, Luong LT, Michaels J, Morgan JJ, Morgan RJ, Mortrud MT, Mosqueda NF, Ng LL, Ng R, Orta GJ, Overly CC, Pak TH, Parry SE, Pathak SD, Pearson OC, Puchalski RB, Riley ZL, Rockett HR, Rowland SA, Royall JJ, Ruiz MJ, Sarno NR, Schaffnit K, Shapovalova NV, Sivisay T, Slaughterbeck CR, Smith SC, Smith KA, Smith BI, Sodt AJ, Stewart NN, Stumpf KR, Sunkin SM, Sutram M, Tam A, Teemer CD, Thaller C, Thompson CL, Varnam LR, Visel A, Whitlock RM, Wohnoutka PE, Wolkey CK, Wong VY, Wood M, Yaylaoglu MB, Young RC, Youngstrom BL, Yuan XF, Zhang B, Zwingman TA, Jones AR (2007). Genome-wide atlas of gene expression in the adult mouse brain..
LINK
https://doi.org/

21.5 *Optional:* To learn about the functions associated with our entire collection of differential expression results, we can use a functional ontology analysis. There are many different varieties of functional ontology analysis tools; I find that these two are an easy place to start out:

GOrilla:
https://cbl-gorilla.cs.technion.ac.il/

EnrichR:
https://maayanlab.cloud/Enrichr/

Note

When running functional ontology analyses on differential expression results from brain tissue, it is particularly important to either use a tool that either considers the full ranked list of results (i.e., all significant (FDR<0.05) and non-significant results) or that compares the significant results (FDR<0.05) to a "background" of all genes contained within the results. Do not use the full genome as the "background" for your functional comparison. Only a subset of the genome is expressed in brain tissue - therefore, by definition, the genes included in your results are already enriched for brain functions, regardless of their relationship to your variable of interest.

CITATION

Eden E, Navon R, Steinfeld I, Lipson D, Yakhini Z (2009). GOrilla: a tool for discovery and visualization of enriched GO terms in ranked gene lists..
LINK
https://doi.org/10.1186/1471-2105-10-48

CITATION

Kuleshov MV, Jones MR, Rouillard AD, Fernandez NF, Duan Q, Wang Z, Koplev S, Jenkins SL, Jagodnik KM, Lachmann A, McDermott MG, Monteiro CD, Gundersen GW, Ma'ayan A (2016). Enrichr: a comprehensive gene set enrichment analysis web server 2016 update..
LINK
https://doi.org/10.1093/nar/gkw377

Wrapping Up the Project

22 In order to wrap up the meta-analysis project in a manner that can be easily followed up on later by yourself or others, it is important to preserve input, output, code, and workspaces.

22.1 First, make sure that you save both your code and workspace in R.

Note

The names for the code files will end with the file extension ".R".
The names for the workspace files will end with the file extension ".Rdata".

22.2 To preserve and share code, it is useful to upload the final R code and workspaces to a Github repository.

Note

Here are instructions for initiating a Github repository: https://docs.github.com/en/repositories/creating-and-managing-repositories/quickstart-for-repositories

(if you haven't already been using Github for version control throughout the duration of the project)

22.3 To make your analyses reproducible, make sure that you save the folders containing the Gemma differential expression results for each dataset that you used as input for your meta-analysis.

Note

Gemma updates gene annotation each time a new genome assembly is released, therefore a scientist following the same extraction and analysis procedure a year from now might get different results if they do not have your exact input.

22.4 In addition to your PRISMA search diagram, table of final datasets, and results summary, make sure to save the full meta-analysis results for all genes. The full results can be provided as a supplementary table in a publication.

22.5 A README describing the organization for all of the meta-analysis files can help you (or others) navigate the files later.

Protocol references

Eden, E., Navon, R., Steinfeld, I., Lipson, D., Yakhini, Z., 2009. GOrilla: a tool for discovery and visualization of enriched GO terms in ranked gene lists. BMC Bioinformatics 10, 48. https://doi.org/10.1186/1471-2105-10-48

Kuleshov, M.V., Jones, M.R., Rouillard, A.D., Fernandez, N.F., Duan, Q., Wang, Z., Koplev, S., Jenkins, S.L., Jagodnik, K.M., Lachmann, A., McDermott, M.G., Monteiro, C.D., Gundersen, G.W., Ma'ayan, A., 2016. Enrichr: a comprehensive gene set enrichment analysis web server 2016 update. Nucleic Acids Res 44, W90-97. https://doi.org/10.1093/nar/gkw377

Lein, E.S., Hawrylycz, M.J., Ao, N., Ayres, M., Bensinger, A., Bernard, A., Boe, A.F., Boguski, M.S., Brockway, K.S., Byrnes, E.J., Chen, Lin, Chen, Li, Chen, T.-M., Chin, M.C., Chong, J., Crook, B.E., Czaplinska, A., Dang, C.N., Datta, S., Dee, N.R., Desaki, A.L., Desta, T., Diep, E., Dolbeare, T.A., Donelan, M.J., Dong, H.-W., Dougherty, J.G., Duncan, B.J., Ebbert, A.J., Eichele, G., Estin, L.K., Faber, C., Facer, B.A., Fields, R., Fischer, S.R., Fliss, T.P., Frensley, C., Gates, S.N., Glattfelder, K.J., Halverson, K.R., Hart, M.R., Hohmann, J.G., Howell, M.P., Jeung, D.P., Johnson, R.A., Karr, P.T., Kawal, R., Kidney, J.M., Knapik, R.H., Kuan, C.L., Lake, J.H., Laramee, A.R., Larsen, K.D., Lau, C., Lemon, T.A., Liang, A.J., Liu, Y., Luong, L.T., Michaels, J., Morgan, J.J., Morgan, R.J., Mortrud, M.T., Mosqueda, N.F., Ng, L.L., Ng, R., Orta, G.J., Overly, C.C., Pak, T.H., Parry, S.E., Pathak, S.D., Pearson, O.C., Puchalski, R.B., Riley, Z.L., Rockett, H.R., Rowland, S.A., Royall, J.J., Ruiz, M.J., Sarno, N.R., Schaffnit, K., Shapovalova, N.V., Sivisay, T., Slaughterbeck, C.R., Smith, S.C., Smith, K.A., Smith, B.I., Sodt, A.J., Stewart, N.N., Stumpf, K.-R., Sunkin, S.M., Sutram, M., Tam, A., Teemer, C.D., Thaller, C., Thompson, C.L., Varnam, L.R., Visel, A., Whitlock, R.M., Wohnoutka, P.E., Wolkey, C.K., Wong, V.Y., Wood, M., Yaylaoglu, M.B., Young, R.C., Youngstrom, B.L., Yuan, X.F., Zhang, B., Zwingman, T.A., Jones, A.R., 2007. Genome-wide atlas of gene expression in the adult mouse brain. Nature 445, 168–176. https://doi.org/10.1038/nature05453

Liberati, A., Altman, D.G., Tetzlaff, J., Mulrow, C., Gøtzsche, P.C., Ioannidis, J.P.A., Clarke, M., Devereaux, P.J., Kleijnen, J., Moher, D., 2009. The PRISMA Statement for Reporting Systematic Reviews and Meta-Analyses of Studies That Evaluate Health Care Interventions: Explanation and Elaboration. PLOS Medicine 6, e1000100. https://doi.org/10.1371/journal.pmed.1000100

Lim, N., Tesar, S., Belmadani, M., Poirier-Morency, G., Mancarci, B.O., Sicherman, J., Jacobson, M., Leong, J., Tan, P., Pavlidis, P., 2021. Curation of over 10 000 transcriptomic studies to enable data reuse. Database (Oxford) 2021, baab006. https://doi.org/10.1093/database/baab006

Moher, D., Liberati, A., Tetzlaff, J., Altman, D.G., 2010. Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement. International Journal of Surgery 8, 336–341. https://doi.org/10.1016/j.ijsu.2010.02.007

Pollard, K.S., Dudoit, S., Laan, M.J. van der, 2005. Multiple Testing Procedures: the multtest Package and Applications to Genomics, in: Bioinformatics and Computational Biology Solutions Using R and Bioconductor, Statistics for Biology and Health. Springer, New York, NY, pp. 249–271. https://doi.org/10.1007/0-387-29362-0_15

Ritchie, M.E., Phipson, B., Wu, D., Hu, Y., Law, C.W., Shi, W., Smyth, G.K., 2015. limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic Acids Res. 43, e47. https://doi.org/10.1093/nar/gkv007

Saunders, A., Macosko, E.Z., Wysoker, A., Goldman, M., Krienen, F.M., de Rivera, H., Bien, E., Baum, M., Bortolin, L., Wang, S., Goeva, A., Nemesh, J., Kamitaki, N., Brumbaugh, S., Kulp, D., McCarroll, S.A., 2018. Molecular Diversity and Specializations among the Cells of the Adult Mouse Brain. Cell 174, 1015-1030.e16. https://doi.org/10.1016/j.cell.2018.07.028

Shimoyama, M., De Pons, J., Hayman, G.T., Laulederkind, S.J.F., Liu, W., Nigam, R., Petri, V., Smith, J.R., Tutaj, M., Wang, S.-J., Worthey, E., Dwinell, M., Jacob, H., 2015. The Rat Genome Database 2015: genomic, phenotypic and environmental variations and disease. Nucleic Acids Res. 43, D743-750. https://doi.org/10.1093/nar/gku1026

Viechtbauer, W., 2010. Conducting Meta-Analyses in R with The metafor Package. Journal of Statistical Software 36. https://doi.org/10.18637/jss.v036.i03

Wickham, H., 2023. plyr: Tools for Splitting, Applying and Combining Data.

Wickham, H., Hester, J., Chang, W., Bryan, J., RStudio, 2022. devtools: Tools to Make Developing R Packages Easier.

Zeisel, A., Hochgerner, H., Lönnerberg, P., Johnsson, A., Memic, F., van der Zwan, J., Häring, M., Braun, E., Borm, L.E., La Manno, G., Codeluppi, S., Furlan, A., Lee, K., Skene, N., Harris, K.D., Hjerling-Leffler, J., Arenas, E., Ernfors, P., Marklund, U., Linnarsson, S., 2018. Molecular Architecture of the Mouse Nervous System. Cell 174, 999-1014.e22. https://doi.org/10.1016/j.cell.2018.06.021

Zoubarev, A., Hamer, K.M., Keshav, K.D., McCarthy, E.L., Santos, J.R.C., Van Rossum, T., McDonald, C., Hall, A., Wan, X., Lim, R., Gillis, J., Pavlidis, P., 2012. Gemma: a resource for the reuse, sharing and meta-analysis of expression profiling data. Bioinformatics 28, 2272–2273. https://doi.org/10.1093/bioinformatics/bts430

Citations

Step 16

Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W, Smyth GK. limma powers differential expression analyses for RNA-sequencing and microarray studies.
https://doi.org/10.1093/nar/gkv007

Step 18.1

Wickham H. The Split-Apply-Combine Strategy for Data Analysis
10.18637/jss.v040.i01

Step 19.2

Viechtbauer, W. Conducting meta-analyses in R with the metafor package
https://doi.org/10.18637/jss.v036.i03

Step 20.1

Pollard K.S., Dudoit S., van der Laan M.J.. Multiple Testing Procedures: R multtest Package and Applications to Genomics, in Bioinformatics and Computational Biology Solutions Using R and Bioconductor
10.1007/0-387-29362-0

Step 21.4

Zeisel A, Hochgerner H, Lönnerberg P, Johnsson A, Memic F, van der Zwan J, Häring M, Braun E, Borm LE, La Manno G, Codeluppi S, Furlan A, Lee K, Skene N, Harris KD, Hjerling-Leffler J, Arenas E, Ernfors P, Marklund U, Linnarsson S. Molecular Architecture of the Mouse Nervous System.
https://doi.org/10.1016/j.cell.2018.06.021

Step 21.4

Lein ES, Hawrylycz MJ, Ao N, Ayres M, Bensinger A, Bernard A, Boe AF, Boguski MS, Brockway KS, Byrnes EJ, Chen L, Chen L, Chen TM, Chin MC, Chong J, Crook BE, Czaplinska A, Dang CN, Datta S, Dee NR, Desaki AL, Desta T, Diep E, Dolbeare TA, Donelan MJ, Dong HW, Dougherty JG, Duncan BJ, Ebbert AJ, Eichele G, Estin LK, Faber C, Facer BA, Fields R, Fischer SR, Fliss TP, Frensley C, Gates SN, Glattfelder KJ, Halverson KR, Hart MR, Hohmann JG, Howell MP, Jeung DP, Johnson RA, Karr PT, Kawal R, Kidney JM, Knapik RH, Kuan CL, Lake JH, Laramee AR, Larsen KD, Lau C, Lemon TA, Liang AJ, Liu Y, Luong LT, Michaels J, Morgan JJ, Morgan RJ, Mortrud MT, Mosqueda NF, Ng LL, Ng R, Orta GJ, Overly CC, Pak TH, Parry SE, Pathak SD, Pearson OC, Puchalski RB, Riley ZL, Rockett HR, Rowland SA, Royall JJ, Ruiz MJ, Sarno NR, Schaffnit K, Shapovalova NV, Sivisay T, Slaughterbeck CR, Smith SC, Smith KA, Smith BI, Sodt AJ, Stewart NN, Stumpf KR, Sunkin SM, Sutram M, Tam A, Teemer CD, Thaller C, Thompson CL, Varnam LR, Visel A, Whitlock RM, Wohnoutka PE, Wolkey CK, Wong VY, Wood M, Yaylaoglu MB, Young RC, Youngstrom BL, Yuan XF, Zhang B, Zwingman TA, Jones AR. Genome-wide atlas of gene expression in the adult mouse brain.
https://doi.org/

Step 21.4

Saunders A, Macosko EZ, Wysoker A, Goldman M, Krienen FM, de Rivera H, Bien E, Baum M, Bortolin L, Wang S, Goeva A, Nemesh J, Kamitaki N, Brumbaugh S, Kulp D, McCarroll SA. Molecular Diversity and Specializations among the Cells of the Adult Mouse Brain.
https://doi.org/10.1016/j.cell.2018.07.028

Step 21.4

Shimoyama M, De Pons J, Hayman GT, Laulederkind SJ, Liu W, Nigam R, Petri V, Smith JR, Tutaj M, Wang SJ, Worthey E, Dwinell M, Jacob H. The Rat Genome Database 2015: genomic, phenotypic and environmental variations and disease.
https://doi.org/10.1093/nar/gku1026

Step 21.5

Eden E, Navon R, Steinfeld I, Lipson D, Yakhini Z. GOrilla: a tool for discovery and visualization of enriched GO terms in ranked gene lists.
https://doi.org/10.1186/1471-2105-10-48

Step 21.5

Kuleshov MV, Jones MR, Rouillard AD, Fernandez NF, Duan Q, Wang Z, Koplev S, Jenkins SL, Jagodnik KM, Lachmann A, McDermott MG, Monteiro CD, Gundersen GW, Ma'ayan A. Enrichr: a comprehensive gene set enrichment analysis web server 2016 update.
https://doi.org/10.1093/nar/gkw377

Step 3

Moher D, Liberati A, Tetzlaff J, Altman DG, PRISMA Group. Preferred reporting items for systematic reviews and meta-analyses: the PRISMA statement.
https://doi.org/10.1136/bmj.b2535

Step 3

Liberati A, Altman DG, Tetzlaff J, Mulrow C, Gøtzsche PC, Ioannidis JP, Clarke M, Devereaux PJ, Kleijnen J, Moher D. The PRISMA statement for reporting systematic reviews and meta-analyses of studies that evaluate healthcare interventions: explanation and elaboration.
https://doi.org/10.1136/bmj.b2700

Step 3.2

Zoubarev A, Hamer KM, Keshav KD, McCarthy EL, Santos JR, Van Rossum T, McDonald C, Hall A, Wan X, Lim R, Gillis J, Pavlidis P. Gemma: a resource for the reuse, sharing and meta-analysis of expression profiling data.
https://doi.org/10.1093/bioinformatics/bts430

Step 3.2

Lim N, Tesar S, Belmadani M, Poirier-Morency G, Mancarci BO, Sicherman J, Jacobson M, Leong J, Tan P, Pavlidis P. Curation of over 10 000 transcriptomic studies to enable data reuse.

https://doi.org/10.1093/database/baab006

Step 8.1

Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R, Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller E, Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V, Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H . “Welcome to the tidyverse”
doi:10.21105/joss.01686

Step 8.1

Wickham H. The Split-Apply-Combine Strategy for Data Analysis
10.18637/jss.v040.i01

Step 9

Lim N, Tesar S, Belmadani M, Poirier-Morency G, Mancarci BO, Sicherman J, Jacobson M, Leong J, Tan P, Pavlidis P. Curation of over 10 000 transcriptomic studies to enable data reuse.
https://doi.org/10.1093/database/baab006