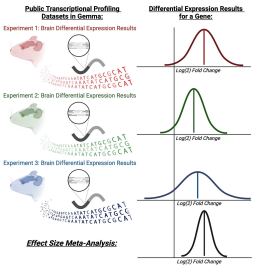
Sep 24, 2024

Brain Data Alchemy Project: Meta-Analysis of Re-Analyzed Public Transcriptional Profiling Data in the Gemma Database (v.2024)



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.yxmvmejkng3p/v1

Megan Hagenauer[1], Duy Manh Nguyen[2], Elizabeth Flandreau[3], Eva Geoghegan[4], Sophie Mensch[1], Mubashshir Ra'eed Bhuiyan[1], Lakshmi Thirupatamma Chennupati[3], Sophia Espinoza[5], Ashlee Lewis[6], Anna Drozman[7], Brandon W. Hughes, M.S., Ph.D.[8]

[1]University of Michigan; [2]Grinnell College; [3]Grand Valley State University; [4]Columbia University; [5]Michigan State University; [6]University of Detroit Mercy; [7]Duke University; [8]Mount Sinai - Icahn School of Medicine

Brain Data Alchemy Project

Megan Hagenauer
University of Michigan

DOI: dx.doi.org/10.17504/protocols.io.yxmvmejkng3p/v1

Protocol Citation: Megan Hagenauer, Duy Manh Nguyen, Elizabeth Flandreau, Eva Geoghegan, Sophie Mensch, Mubashshir Ra'eed Bhuiyan, Lakshmi Thirupatamma Chennupati, Sophia Espinoza, Ashlee Lewis, Anna Drozman, Brandon W. Hughes, M.S., Ph.D. 2024. Brain Data Alchemy Project: Meta-Analysis of Re-Analyzed Public Transcriptional Profiling Data in the Gemma Database (v.2024). **protocols.io https://dx.doi.org/10.17504/protocols.io.yxmvmejkng3p/v1**

Protocol status: Working
We successfully used this protocol 6/2024-8/2024.

Created: September 09, 2024

Last Modified: September 24, 2024

Protocol Integer ID: 107212

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
University of Michigan Undergraduate Research Opportunities Program (UROP)
Grant ID: NA

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

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.

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.

This protocol outlines the methods used during the third pilot year of the program in 2024. These methods differ from the 2022 methods (**dx.doi.org/10.17504/protocols.io.j8nlk84jxl5r/v1**) in terms of the sophistication of the search strategy and inclusion/exclusion procedure, use of Entrez ID annotation and formal mouse/rat gene orthology, and the functionalization of most necessary coding steps. These methods also differ from the 2023 project methods - the 2023 methods imported the preprocessed gene expression data and metadata from Gemma, whereas the 2022 and 2024 project methods imported 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 used by participants within the Brain Data Alchemy Program during the third pilot year of the program (2024). For updates, see the associated forks for this protocol.

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 (v.4.4.0) available on 06/01/2024. It is likely to work using other versions.

1.2 Install R Studio

Software

NAME

R Studio Desktop

DEVELOPER

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.2022.04.2) available on 06/01/2024. 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: **https://github.com/**

Note

To learn the basics of Github:
https://docs.github.com/en/get-started/quickstart/hello-world

DataCamp also has an interactive Github introductory training:
Github Concepts: 2 hrs
https://www.datacamp.com/courses/github-concepts

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

2.4 Create a Github repository for your project. Give the repository a name that clearly signals the goals of your specific project (e.g., "SleepDeprivationMetaAnalysis").

Note

Link to detailed instructions for creating a Github repository:
https://docs.github.com/en/repositories/creating-and-managing-repositories/quickstart-for-repositories

2.5 Within R Studio, set your Github credentials.

Command

Example R Code: Setting Github credentials within RStudio

```
#Setting GitHub Credentials
#Megan Hagenauer, July 1, 2024

#Checking on my GitHub settings:
library("usethis")
git_sitrep()

#Example Output:
# ── Git global (user)
# • Name: 'Megan Hastings Hagenauer'
# • Email: 'hagenaue@umich.edu'
# • Global (user-level) gitignore file: <unset>
#   • Vaccinated: FALSE
# ℹ See `?git_vaccinate` to learn more
# ℹ Defaulting to 'https' Git protocol
# • Default Git protocol: 'https'
# • Default initial branch name: <unset>
#
#   ── GitHub user
# • Default GitHub host: 'https://github.com'
# • Personal access token for 'https://github.com': <unset>
#   • To create a personal access token, call `create_github_token()`
# • To store a token for current and future use, call
`gitcreds::gitcreds_set()`
# ℹ Read more in the 'Managing Git(Hub) Credentials' article:
#   https://usethis.r-lib.org/articles/articles/git-credentials.html
#
# ── Active usethis project:
'/Users/hagenaue/Documents/Example2024_BrainDataAlchemy' ──
#
# ── Git local (project)
# • Name: 'Megan Hastings Hagenauer'
# • Email: 'hagenaue@umich.edu'
# • Default branch: 'main'
# • Current local branch -> remote tracking branch:
#   'main' -> 'origin/main'
#
# ── GitHub project
# • Type = 'maybe_ours_or_theirs'
# • Host = 'https://github.com'
```

```
# • Config supports a pull request = NA
# • origin = 'hagenaue/Example2024_BrainDataAlchemy'
# • upstream = <not configured>
#   • Desc = 'origin' is a GitHub repo and 'upstream' is either not
configured or is not a GitHub repo.
#
# We may be offline or you may need to configure a GitHub personal
access
# token. `gh_token_help()` can help with that.
#
# Read more about what this GitHub remote configurations means at:
#   'https://happygitwithr.com/common-remote-setups.html'

create_github_token()

gitcreds::gitcreds_set()
# -> Adding new credentials...
# -> Removing credentials from cache...
# -> Done.
```

Note

This blog post by David Keyes (2021) includes a nice overview of how to set Github credentials within RStudio (Section: "Connect RStudio and GitHub"): https://rfortherestofus.com/2021/02/how-to-use-git-github-with-r

2.6 Within R Studio, initiate an R project.

Note

- Choose the version control option, and enter the URL for the Github repository that you created for the project.
- When you set the working directory, make sure that you choose a logical place on your computer to store your project. This folder is where you will place any input files for your project and also where you will find any output files from your project.
- Check the box for "Use renv with this project". This will start tracking the R environment used for your project (e.g., code package versions).
- Check the box for "Open in new session". This will open up a new Rstudio session for your project.

Note

This chapter "6 Starting your *R* projects" from the online book-in-progress "R for Non-Programmers: A Guide for Social Scientists" by *Daniel Dauber (2024-02-05)* provides a nice overview of how to initiate an R project within R studio:
https://bookdown.org/daniel_dauber_io/r4np_book/starting-your-r-projects.html

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.

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 5).

The 2024 cohort of the Brain Data Alchemy Project chose to examine the effects of antidepressant treatment, antipsychotic treatment, chronic stress, estrus cycle stage, animal models of ADHD, glioblastoma, and viral encephalitis.

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 2024 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"). This is a little awkward within Gemma's API because the full query will eventually be wrapped in quotations, so we end up having to add a backslash in front of the quotations around our exact phrase: \"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").
- Independent search terms can be combined into a longer list using OR (e.g., "stress* OR advers* "). Just listing a variety of terms also seems to function similarly to OR (e.g., "stress* advers*").

Defining search syntax is also a critical step, and feedback is important. For the 2024 cohort of the Brain Data Alchemy Project, these terms were reviewed by a mentor.

Command

Example R code: Formally defining your search terms

```
#Example of a complex query using proper Gemma.R API syntax:

MyQueryTerms<-"(stress* OR \"chronic social defeat\" ) NOT
(\"oxidative stress\" OR \"ER Stress\")"
```

Dataset Identification: Coding Steps

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

Note

Example code with more detail for all of these steps can be found here:

https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_GemmaDEResults_2024/2024_Example_Code_forGemmaSearch_Part1.R

https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_GemmaDEResults_2024/2024_Example_Code_forGemmaSearch_Part2.R

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

Note

- This protocol was designed using the version of Devtools (v.2.4.5) available on 06/01/2024. 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 06/01/2024. *It is unlikely to work using later versions.*

4.3 Install and load the *plyr* package and *dplyr* package. The dplyr package can be installed as part of the full "tidyverse".

Note

- This protocol was designed using the versions of plyr (v.3.7.8) and tidyverse (v.2.0.0) available on 06/01/2024. *It may not work using later versions.*

Command

Example R code: Install and load package plyr and tidyverse

```
install.packages("plyr")

library(plyr)

install.packages("tidyverse")

library(tidyverse)
```

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

4.4 Use the function *get_datasets()* to locate transcriptional profiling datasets within the Gemma database that contain your pre-specified keywords.

Command

Example R Code: Searching for Gemma Datasets related to Antidepressants

```
MyQueryTerms<-"SSRI fluoxetine sertraline paroxetine citalopram
escitalopram fluvoxamine vilazodone vortioxetine amitriptyline
imipramine amoxapine desipramine nortriptyline clomipramine
trimipramine protriptyline doxepin maprotiline trazadone nefazodone
mirtazapine phenelzine nialamide isocarboxazid hydracarbazine
tranylcypromine selegiline venlafaxine desvenlafaxine duloxetine
levomilnacipran bupropion duloxetine levomilnacipran ketamine
esketamine tianeptine brexanolone"


result_MyQueryTerms_RatsMice<- gemma.R
::get_datasets(query=MyQueryTerms, taxa = c("mouse", "rat")) %>%
  gemma.R:::get_all_pages()


#To examine the output of the search:
str(result_MyQueryTerms_RatsMice)
#Classes ‘data.table’ and 'data.frame': 111 obs. of  23 variables:

#You can also peruse the results in Rstudio by going to the
Environment tab in the righthand corner and clicking on the object
```

Note

- The first argument in this function is query. Inputting basic text to this argument means that a search will be performed of the entire Gemma record for that exact text - i.e., it will search the dataset title, description, annotations, and all of the other fields.
- As described in protocol step 3.5, we have defined the query search terms in a separate line of code making an object *MyQueryTerms,* which we then use as the input to query in the *get_datasets()* function. This makes the code easier to read.
- The argument "taxa" allows you to filter by species. We will only be using data from the two most common laboratory rodents in our meta-analysis: rats and mice.
- The search defaults to 20 records at a time - adding the *get_all_pages()* function expands the results to include all records.

Command

Example R code: Getting help with an R function

```
#If you are struggling with getting an R function to work, you can
get help using this code:

help("gemma.R ::get_datasets")
??gemma.R ::get_datasets
```

4.5 Filter the results by data quality to remove "troubled" datasets.

Command

Example R Code: Identifying and filtering out "troubled" datasets from the Gemma results

```
#To determine how many datasets in our search results are marked as
"troubled" within the Gemma database we can make a table:

table(result_MyQueryTerms_NoFilter$experiment.troubled)
# FALSE
# 194
#We only want the non-troubled datasets (so FALSE) - looks good!

#We can filter out any troubled datasets using this code (even if you
don't have troubled datasets, run this code anyway to make sure the
downstream code works):

result_MyQueryTerms_Filtered<-result_MyQueryTerms_RatsMice %>%
filter(experiment.troubled==FALSE)
```

Note

Filtering out troubled datasets can probably also be done within the search, but I chose to instead subset the search results so that we could first count how many of the results were troubled and being filtered out.

4.6 Output the results of the search in a format that can be easily opened in a standard spreadsheet program (comma separated variable file format: .csv).

Command

Example R Code: Output the search results to a comma separate variable (.csv) file

```
#Outputting the search results in comma separate variable (.csv) file
format
#CSV files are easily read into commonly used spreadsheet programs
like Excel or GoogleSheets
write.csv(result_MyQueryTerms_Filtered,
"result_MyQueryTerms_Filtered.csv")

#Where did my file go?
getwd()
```

4.7 Survey the results to determine how many datasets come from each brain region. The datasets in Gemma are reliably annotated with either the "organism parts" or "cell types" that were sampled for transcriptional profiling for each dataset. Brain regions are defined as "organism parts", whereas "cell types" are typically either purified cells (e.g., by flow cytometry) or cell culture.

Command

Example R code: Surveying the results to determine how many datasets come from each brain region

```
#Let's find out what brain regions we have:

MyResults<-result_MyQueryTerms_Filtered

#Note: some datasets have data from more than one "organism part"
#That means that if we survey all of the organism parts in our
datasets
#we may end up with a vector of organism parts that is of unknown
length
#and potentially longer than our number of datasets

#Creating an empty vector to save our results
#We can just add on to this using concatenate
#That way we don't need to pre-specify length

OrganismPartAnnotations_All<-vector(mode="character")

#Looping over all of the datasets:

#How many datasets do we have?
length(MyResults$experiment.shortName)
nrow(MyResults)

for(i in c(1:nrow(MyResults))){

  ExperimentName<-MyResults$experiment.shortName[i]

  ExperimentAnnotations<-
get_dataset_annotations(dataset=ExperimentName)
  rm(ExperimentName)

  #Making sure that there is "organism part" annotation for the
dataset before attempting to extract it:

  #Do we have annotation for organism part? Here is a true/false
vector
  #ExperimentAnnotations$class.name=="organism part"
  #This is the column of terms for the annotation
  #ExperimentAnnotations$term.name
```

```
  #ExperimentAnnotations$term.name
  #Example of subsetting for organism part:

#ExperimentAnnotations$term.name[ExperimentAnnotations$class.name=="or
ganism part"]

  #How many annotations do we have?

length(ExperimentAnnotations$term.name[ExperimentAnnotations$class.nam
e=="organism part"])


if(length(ExperimentAnnotations$term.name[ExperimentAnnotations$class.
name=="organism part"])>0){

  OrganismPartAnnotations<-
ExperimentAnnotations$term.name[ExperimentAnnotations$class.name=="org
anism part"]
  rm(ExperimentAnnotations)

  OrganismPartAnnotations_All<-c(OrganismPartAnnotations_All,
OrganismPartAnnotations)
  rm(OrganismPartAnnotations)

  #if there isn't organism part annotation, we just tell the loop to
move on to the next dataset entry:

  }else{
    rm(ExperimentAnnotations)
  }
}

#This is what our result looks like:
OrganismPartAnnotations_All

#How many datasets have each organism part?  Let's make a summary
table!

table(OrganismPartAnnotations_All)

#That is probably worth saving:

write.csv(table(OrganismPartAnnotations_All),
"Table_OrganismPartAnnotations.csv")
```

4.8 If a particular tissue or ROI was pre-specified during the original conception of the project, jump to step 4.9.

If a particular tissue or brain region of interest (ROI) was not pre-specified during the conception of the project, the research question should now be narrowed to a particular tissue or brain ROI based on dataset availability.

Note

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.

4.9 If there were very few datasets identified by the search (e.g., <50 datasets), it may be possible to filter the datasets down to those that include the targeted tissue/ROI by just looking at the metadata records. If so, jump to step 4.12.

Otherwise, to narrow the search to a particular brain region, it is best if we leverage the formal ontology used for defining "organism part" annotation within the Gemma database.

Note

Why use formal ontology terms to narrow our search to a particular brain region?

Narrowing our search by just adding the brain region to the query terms suffers from two problems:

A. They require us to brainstorm every possible variation on "hippocampus" that could be used in the title, description, or metadata for a dataset (e.g., Ammon's horn, dentate gyrus, CA1, etc).

B. The datasets that we discover may include the term hippocampus in their Gemma record but not actually represent an experiment studying hippocampal tissue (e.g., in the abstract, the authors might describe how they are following up on previous investigations in the hippocampus by studying the amygdala...)

To get around these problems, we can make use of the fact that the datasets in Gemma are reliably annotated with either "organism part" or "cell type" using formal, existing ontologies.
Ontologies are "a set of concepts and categories in a subject area or domain that shows their properties and the relations between them."

You can get a glimpse of the ontological terms used to annotate the datasets using the point-and-click dataset search GUI on the Gemma website (**https://gemma.msl.ubc.ca/browse/#/**).
- To see the ontological terms: after running a search, at the bottom of the screen is "Dataset download code"
- Under this download code tab, the option "Gemma.R" provides the R code for the search.

Many of the ontological terms are hierarchical (e.g., dentate gyrus is part of the hippocampus)

Within an ontological hierarchy:
- the broader, umbrella term is called the "parent" (in the previous example, the hippocampus is the parent)
- the term representing the more specific subset is called the "child" (in the previous example, the dentate gyrus is the child)

Why hierarchical ontology is useful:
- When requesting Gemma records with particular annotation, a parent term can often pull up all records with the child terms

This is an easy way to browse the hierarchies for the ontological terms:
https://www.ebi.ac.uk/ols4
- Gemma considers the children of a term to be the terms listed under "is a"/"subclass of" and "part of"

To specifically browse the hierarchies used for tissues ("organism parts"):
https://www.ebi.ac.uk/ols4/ontologies/uberon

Here are several other categories of terms that we regularly reference - I found this ontological annotation within the Gemma database to be less reliable and useful for the purposes of our meta-analyses:

Experimental Factors: **https://www.ebi.ac.uk/efo/**

Drugs - note: Gemma does *not* infer children from parent terms within this ontological framework: https://www.ebi.ac.uk/ols4/ontologies/chebi

There is also annotation for developmental stage, but I wouldn't recommend filtering using it - many of the datasets seem to lack this annotation.

Command

Example R Code: Identifying useful "organism part" annotation ontology terms

```
#We can search for annotations related to hippocampus using the
search_annotations function
#e.g.,
annots<-gemma.R ::search_annotations("hippocamp*")

#To view the results you can click on the annots object in the
Environment tab

#That search provides ontological annotations that include terms that
start with hippocamp*
#if we want to determine which of these annotations are actually used
in the Gemma database, we can run this code to count their usage:

annot_counts <- annots$value.URI %>% sapply(\(x){
  attributes(get_datasets(uris = x))$totalElements
})
annots$counts = annot_counts

#Then we can filter down to just the ontological terms that are
actually used in the Gemma database:
annots_wRecords<-annots %>% filter(counts>0)

#You'll note that one of the frequently used terms is an "organism
part" hippocampal formation
#That implies that this annotation is used to characterize the actual
tissue used in the experiment
#It has the annotation ontological URI:
http://purl.obolibrary.org/obo/UBERON_0002421

#If we use "UBERON_0002421" (hippocampal formation) in our search for
datasets
# we will get not just results for hippocampal formation, but results
matching its child terms as well

#If we wanted to see the hierarchy of ontological terms encompassed
by "hippocampal formation" and/or encompassing "hippocampal formation"
#We can go to EMBL-EBI Ontology Lookup Service:
# https://www.ebi.ac.uk/ols4
#And put in the search box UBERON_0002421
```

```
#And put in the search box UBERON_0002421

#EMBL-EBI will tell us all of the various terms/concepts encompassed
by "hippocampal formation" in the "Related from" section under "part
of"
#e.g., layer of hippocampus, subiculum, fornix, Ammon's horn, etc

#Whereas to see if "hippocampal formation" can be broadened to
include other useful nearby areas, we can click above it in the
hierarchy ("cerebral cortex") to see our options

#We can also double check that those child terms for "hippocampal
formation" are actually used within Gemma using this function:
get_child_terms("http://purl.obolibrary.org/obo/UBERON_0002421")
```

4.10 Re-run the search, using the formal ontology term for the "organism part".

Command

Example R Code: Re-running the search to narrow our datasets down to a particular tissue or ROI

```
#Re-running the search to narrow our datasets down to a particular
tissue or ROI:

#Example: Hippocampal formation:

#I'm going to use the URI ontology term for the hippocampus so that I
catch all of datasets annotated with the child terms as well:
result_MyQueryTerms_RatsMice_ROI <- gemma.R
::get_datasets(query=MyQueryTerms, filter =
'allCharacteristics.valueUri in
(http://purl.obolibrary.org/obo/UBERON_0002421)', taxa = c("mouse",
"rat")) %>%
  gemma.R:::get_all_pages()

#Output:
str(result_MyQueryTerms_RatsMice_ROI)
#Classes ‘data.table’ and 'data.frame': 31 obs. of  23 variables:
```

4.11 Filter the results again down to high quality data (no "troubled" datasets).

Command

Example R Code: Filtering the results for a particular region down to high quality data

```
#Filtering out the "troubled" results again:

result_MyQueryTerms_RatsMice_ROI_Filtered<-
result_MyQueryTerms_RatsMice_ROI[result_MyQueryTerms_RatsMice_ROI$expe
riment.troubled==FALSE,]

#Output:

str(result_MyQueryTerms_RatsMice_ROI_Filtered)
#Classes ‘data.table’ and 'data.frame': 31 obs. of  23 variables:
```

4.12 To make the dataset records easier to triage using our standardized inclusion/exclusion criteria, let's add additional annotation to the dataset metadata results outputted by our search, including information about the organism parts, developmental stages, experimental factors used in the experiment.

Command

Example R Code: Adding additional annotation to the dataset metadata records

```
#Adding additional annotation to this basic output from the
getDatasets function
#e.g., info about the organism parts, developmental stage annotation,
experimental factors used in the analyses

#To begin with, I renamed my results to something generic to make
this code easier to functionalize later...
MyResults<-result_MyQueryTerms_RatsMice_ROI_Filtered

#Let's make some empty vectors that are the same length as the
columns in our results
#These empty vectors will be used to store our annotations while we
loop through the rows of datasets:
OrganismParts<-vector(mode="character", length=nrow(MyResults))
CellTypes<-vector(mode="character", length=nrow(MyResults))
DevelopmentalStages<-vector(mode="character", length=nrow(MyResults))
Treatments<-vector(mode="character", length=nrow(MyResults))
Diseases<-vector(mode="character", length=nrow(MyResults))
DiseaseModels<-vector(mode="character", length=nrow(MyResults))
Genotypes<-vector(mode="character", length=nrow(MyResults))
Strains<-vector(mode="character", length=nrow(MyResults))
Sex<-vector(mode="character", length=nrow(MyResults))

#I'm going to loop over all of the rows (row number =i) in my results
(i.e., dataset metadata)
#And collect all of this annotation information
#And then format it in a way so that it can be added into my simple
dataframe of results
#And then outputted and read easily in a spreadsheet program like
excel

for(i in c(1:nrow(MyResults))){

  #Pulling out the name for the dataset in a row (row number=i):
  ExperimentName<-MyResults$experiment.shortName[i]

  #Accessing the annotations for the dataset:
  ExperimentAnnotations<-
get_dataset_annotations(dataset=ExperimentName)
```

```
get_dataset_annotations(dataset=ExperimentName)
  #The number and type of annotations for the datasets is quite
variable

  rm(ExperimentName)

  #Determining whether there is any annotation for organism part:


if(length(ExperimentAnnotations$term.name[ExperimentAnnotations$class.
name=="organism part"])>0){

    #If there is organism part annotation, I'm grabbing it:

    Annotations<-
ExperimentAnnotations$term.name[ExperimentAnnotations$class.name=="org
anism part"]

    #And then collapsing that vector of annotations into a single
string
    #that can be easily stashed in a single cell in a data.frame (or
Excel spreadsheet)
    #This will eventually become part of the the row for that dataset
in the results
    # e.g., "annotation 1; annotation 2; annotation 3"
    OrganismParts[i]<-paste(Annotations, collapse="; ")
    rm(Annotations)

  #If there isn't any annotation for organism part, we move on to the
next type of annotation:
  }else{}

  #Now grabbing the annotation for cell type in a similar manner:

if(length(ExperimentAnnotations$term.name[ExperimentAnnotations$class.
name=="cell type"])>0){

    Annotations<-
ExperimentAnnotations$term.name[ExperimentAnnotations$class.name=="cel
l type"]

    CellTypes[i]<-paste(Annotations, collapse="; ")
    rm(Annotations)

  }else{ }

  #Now grabbing the annotation for developmental stage in a similar
```

manner:

4.13 Add some empty columns to our dataset metadata data.frame for taking inclusion/exclusion notes and output the data frame as a file format that is easily viewed using a standard spreadsheet program (comma separated variable file: .csv).

Command

Example R code: Adding some empty columns to our dataset metadata data frame for taking inclusion/exclusion notes and outputting as a .csv

```
#Let's add some empty columns for taking inclusion/exclusion notes:

ManipulationUnrelatedToTopic<-vector(mode="character",
length=nrow(MyResults))
IncorrectDevelopmentalStage<-vector(mode="character",
length=nrow(MyResults))
NotBulkDissection_ParticularCellTypeOrSubRegion<-
vector(mode="character", length=nrow(MyResults))
NotFullTranscriptome_ChipSeq_TRAP_miRNA<-vector(mode="character",
length=nrow(MyResults))
MetadataIssues_MissingInfo_NoPub_Retracted_Duplicated<-
vector(mode="character", length=nrow(MyResults))

Excluded<-vector(mode="character", length=nrow(MyResults))
WhyExcluded<-vector(mode="character", length=nrow(MyResults))

MyResults_Annotated<-cbind.data.frame(MyResults_Annotated,
ManipulationUnrelatedToTopic, IncorrectDevelopmentalStage,
NotBulkDissection_ParticularCellTypeOrSubRegion,
NotFullTranscriptome_ChipSeq_TRAP_miRNA,
MetadataIssues_MissingInfo_NoPub_Retracted_Duplicated, Excluded,
WhyExcluded)

#And then write out the results so that we can snoop through them in
a spreadsheet program like Excel:
write.csv(MyResults_Annotated, "MyResults_Annotated.csv")
```

Note

The outputted file will be called "MyResults_annotated.csv" and will be located in your current working directory. It can be opened using a standard spreadsheet software program, like Excel or Google Sheets.

Initial Dataset Filtering Using MetaData

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

5.1 Open the file containing the dataset metadata ("MyResults_annotated.csv") using a standard spreadsheet software program, like Excel or Google Sheets. Immediately resave the file in a spreadsheet format (e.g., .xlsx) - otherwise any changes that you make to the formatting (like color) will be lost when you close the file (.csv files do not include formatting).

Note

The file "MyResults_annotated.csv" will be located in your current working directory.
We are switching over to looking at the dataset metadata in a spreadsheet software program instead of R because it is makes it easier to explore and annotate the metadata records. That said, because we are switching over to making changes to the dataset metadata by hand (instead of using coding), make sure that you take good notes about what changes you make (and why!).

5.2 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. Make notes in the empty columns provided for the common exclusion criteria. For each dataset that is excluded, also make notes in the "Excluded" and "WhyExcluded" column.

Note

Pre-specified exclusion criteria:
1) The targeted tissue/ROI was not sampled (if you haven't already filtered by ROI in your search, make notes in the "WhyExcluded" column).
2) The experimental manipulation was unrelated to the meta-analysis topic (note in column: "ManipulationUnrelatedToTopic").
3) The brain tissue used in the experiment was not from a bulk dissection, and instead represented a particular cell type or small subregion (note in column: "NotBulkDissection_ParticularCellTypeOrSubRegion").
4) The experiment targeted a particular subpopulation of transcripts (e.g., Chip-Seq, miRNA, TRAP-Seq, etc., note in column: NotFullTranscriptome_ChipSeq_TRAP_miRNA)
5) The experiment used subjects from a developmental stage other than the timeframe of interest (note in column: "IncorrectDevelopmentalStage").
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, note in column: "MetadataIssues_MissingInfo_NoPub_Retracted_Duplicated").

5.3 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).

5.4 Copy the metadata record worksheet and use it to make a new worksheet in the same file. For this new worksheet, delete all metadata records for datasets that were excluded so that it is easier to easily view/read the remaining potential datasets.

Note

Removing the excluded datasets within the new worksheet can be easily done by sorting the worksheet by the "Excluded" column.

Secondary Dataset Filtering Using Detailed Methodological Review

6 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.

6.1 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 (note in column: "MetadataIssues_MissingInfo_NoPub_Retracted_Duplicated").
2) The experimental design was confounded or lacked a proper control group for the experimental manipulation of interest (note in column: "WhyExcluded").
3) The subjects or protocol used in the study were dramatically different from all other included datasets (note in column: "WhyExcluded").

6.2 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 documented.

7 Next, double check whether the transcriptional profiling data for each dataset appears to have been imported into Gemma properly (especially datasets marked “external”).
This step is best completed for one dataset at a time.

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

7.1 Download the processed gene expression data from Gemma for the potential dataset.

Command

Example R Code: Download processed gene expression data from Gemma for a dataset

```
#We can download the processed expression data for any particular
dataset using this code:
Expression<-gemma.R::get_dataset_processed_expression("GSE81672")
#In this case "GSE81672" is the dataset id
#For reference, this example is an RNA-Seq dataset

#Structure of the new Expression object:
str(Expression)
#Classes ‘data.table’ and 'data.frame': 35205 obs. of  103 variables:
#The first four columns are row metadata: Probe, GeneSymbol,
GeneName, NCBIid
#The rest of the columns are gene expression values for each subject
```

7.2 Double-check the range of the processed gene expression data for the dataset.

- For log2 RNA-Seq gene expression data, the range is typically between -5 to 12
- For log2 microarray gene expression datasets, the range is often between 4-15

If you see a highly restricted range (e.g., 2-4), the dataset should be excluded (note in column: "WhyExcluded").

Command

Example R Code: Visualize the distribution of the gene expression data for a dataset

```
#If you want to visualize the distribution of gene expression data
for the entire study, you will need to grab all of the columns that
aren't row metadata (i.e., not rows 1-4) and force them into the
format of a numeric matrix first:

ExpressionMatrix<-as.matrix(Expression[,-c(1:4)])

#You can visualize the distribution of the gene expression data for
the dataset using a histogram
#We can make this pretty by adding a title and x-axis label
#You can also change the color and scaling
hist(ExpressionMatrix, main="Histogram", xlab="Log2 Expression",
col="red", cex.axis=1.3, cex.lab=1.3)

#The range of the x-axis is the range of the log2 gene expression
values
#The large spike on the left side of the histogram ("floor effect")
are all of the genes that aren't truly expressed or have too low of
expression to be measurable

#You can save the histogram using "export" in the Plots window

#We can also pull out numeric values summarizing the distribution,
e.g.:
min(ExpressionMatrix, na.rm=TRUE)
#[1] -5.8601
median(ExpressionMatrix, na.rm=TRUE)
#[1] -2.1651
max(ExpressionMatrix, na.rm=TRUE)
#[1] 12.312

#Or to get more of an overview:
summary(ExpressionMatrix)
```

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 2024 cohort of the Brain Data Alchemy Project, these inclusion/exclusion decisions were reviewed by the mentor (Dr. Hagenauer).

7.4 Copy the remaining metadata record worksheet and use it to make a new (third) worksheet in the same file. For this new worksheet, again delete all metadata records for datasets that were excluded so that it is easier to easily view/read the remaining potential datasets.

Note

Removing the excluded datasets within the new worksheet can be easily done by sorting the worksheet by the "Excluded" column.

7.5 Copy the metadata for this final set of selected datasets into a new .csv file and save it in your working directory with the name "MyDatasets_Screened.csv".

7.6 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.

7.7 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

Tertiary Filtering of Datasets: Availability of Relevant Differential Expression Output

8 Identify the differential expression output (result set and statistical contrast) that is relevant to your research question within the Gemma database for each of the selected datasets.

Note

For the meta-analysis, we will be extracting the differential expression results from Gemma.
The differential expression results for each dataset may include multiple result sets (e.g., one result set for the subset of the data from the hippocampus, one result set for frontal cortex).
Each of these result sets may have multiple statistical contrasts (e.g., drug1 vs. vehicle, drug2 vs. vehicle). Therefore, each of the statistical contrasts is labeled with a result id and contrast id within the Gemma database.

We will need to know which of these ids are relevant to our project goals to easily extract their differential expression results for the meta-analysis.

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

- First, for experiments that include more than one brain region, we will need to double-check that the results have been subsetted by brain region (instead of including brain region ("OrganismPart") as a factor in the differential expression model). If they haven't been subsetted by region, we will probably need to re-run the differential expression analysis. (Depending on the goals of the meta-analysis, we may also need to re-run the differential expression analysis to remove unwanted subjects - e.g., removing subjects with genotypes that might interfere with our results)
- Second, we will need to double-check that the comparisons include an appropriate reference group (baseline). Sometimes the reference and experimental groups are reversed in Gemma (e.g., having the drug treatment set as the baseline, with vehicle as the manipulation). If this is the case, we will need to invert the effects (Log2 Fold Changes) when we input them into our meta-analysis (multiply the effects by -1).

8.1 Input the spreadsheet containing your final chosen datasets into R and extract their dataset ids (GSE#s).

Command

Example R code: Inputting the spreadsheet of selected datasets and extracting their ids

```
MyDatasets_Screened<-read.csv("MyDatasets_Screened.csv",
stringsAsFactors = FALSE, header=TRUE)

#The format of the new object:

str(MyDatasets_Screened)
# 'data.frame': 3 obs. of  24 variables:
#   $ X                          : int  1 2 3
# $ experiment.shortName      : chr  "GSE126678" "GSE181285"
"GSE205325"
# $ experiment.name           : chr  "Enduring and sex-specific
changes in hippocampal gene expression after subchronic immune
challenge" "Expression data from the hippocampus of LPS induced
depression  mice model treated with Luteolin" "Modulation of
behavioral and hippocampal transcriptomic responses in rat prolonged
chronic unpredictable stress"| __truncated__
# $ experiment.ID             : int  15127 21341 24923
# $ experiment.description    : chr  " The goals of this study
include examining long-lasting changes in hippocampal gene expression
in males and in "| __truncated__ " Gene expression profiling reveals
a potential role of Luteolin in LPS induced depression model LPS
depression "| __truncated__ " Here, we examine behavioral and brain
transcriptomic (RNA-seq) responses in rat prolonged chronic
unpredictabl"| __truncated__
# $ experiment.troubled       : logi  FALSE FALSE FALSE
# $ experiment.accession      : chr  "GSE126678" "GSE181285"
"GSE205325"
# $ experiment.database       : chr  "GEO" "GEO" "GEO"
# $ experiment.URI            : chr
"https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE126678"
"https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE181285"
"https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE205325"
# $ experiment.sampleCount    : int  24 16 21
# $ experiment.lastUpdated    : chr  "2023-12-16 09:57:45.253" "2023-
12-17 13:14:26.142" "2023-12-18 00:16:39.838"
# $ experiment.batchEffectText: chr  "NO_BATCH_INFO" "NO_BATCH_INFO"
"NO_BATCH_INFO"
# $ experiment.batchCorrected : logi  FALSE FALSE FALSE
# $ experiment.batchConfound  : int  0 0 0
```

```
# $ experiment.batchConfound    : int  0 0 0
# $ experiment.batchEffect      : int  0 0 0
# $ experiment.rawData          : int  1 1 1
# $ geeq.qScore                 : num  0.282 0.283 0.28
# $ geeq.sScore                 : num  1 0.812 0.75
# $ taxon.name                  : chr  "mouse" "mouse" "rat"
# $ taxon.scientific            : chr  "Mus musculus" "Mus musculus"
"Rattus norvegicus"
# $ taxon.ID                    : int  2 2 3
# $ taxon.NCBI                  : int  10090 10090 10116
# $ taxon.database.name         : chr  "mm10" "mm10" "rn6"
# $ taxon.database.ID           : int  81 81 86


#Pull out the dataset ids (GSE#s) for your datasets:

ExperimentIDs<-MyDatasets_Screened$experiment.shortName

#Format of new ExperimentIDs object:

ExperimentIDs
#[1] "GSE126678" "GSE181285" "GSE205325"

#Alternatively, you can just enter them by hand as a vector:
#ExperimentIDs<-c("GSE126678", "GSE181285", "GSE205325")
```

8.2 Use the dataset ids (GSE#s) to access the result ids and contrast ids within the Gemma database for each dataset, and their accompanying metadata. Format this information into a more easily reviewable data frame.

Command

R Function: Getting result set ids and contrast ids for each dataset

```
GettingResultSetInfoForDatasets<-function(ExperimentIDs){

  #Making an empty data.frame to store results:

  ResultSets_toScreen<-
data.frame(ExperimentID="NA",ResultSetIDs="NA", ContrastIDs="NA",
ExperimentIDs="NA", FactorCategory="NA", ExperimentalFactors="NA",
BaselineFactors="NA", Subsetted=FALSE, SubsetBy="NA")

  str(ResultSets_toScreen)
  # 'data.frame':       1 obs. of  9 variables:
  # $ ExperimentID       : chr "NA"
  # $ ResultSetIDs       : chr "NA"
  # $ ContrastIDs        : chr "NA"
  # $ ExperimentIDs      : chr "NA"
  # $ FactorCategory     : chr "NA"
  # $ ExperimentalFactors: chr "NA"
  # $ BaselineFactors    : chr "NA"
  # $ Subsetted          : logi FALSE
  # $ SubsetBy           : chr "NA"

  #We will then loop over each of the datasets:

  for(i in c(1:length(ExperimentIDs))){

    #For each dataset, we will use Gemma's API to access the
experimental design info:
    Design<-
gemma.R::get_dataset_differential_expression_analyses(ExperimentIDs[i]
)

    if(nrow(Design)>0){
      #Next, we'll make some empty vectors to store the experimental
factor and baseline factor information for each result id for the
dataset:
      ExperimentalFactors<-vector(mode="character",
length(Design$result.ID))
      BaselineFactors<-vector(mode="character",
length(Design$result.ID))
```

```
      #We will then loop over each of the result ids for the dataset:
      for(j in c(1:length(Design$result.ID))){

        #And grab the vector of experimental factors associated with
that result id
        ExperimentalFactorVector<-
Design$experimental.factors[[j]]$summary
        #And collapse that info down to a single entry that will fit
in our data.frame
        ExperimentalFactors[j]<-paste(ExperimentalFactorVector,
collapse="; ")

        #And then grab the vector of baseline/control/reference
values associated with that result id
        BaselineFactorVector<-Design$baseline.factors[[j]]$summary
        #And collapse that info down to a single entry that will fit
in our data.frame
        BaselineFactors[j]<-paste(BaselineFactorVector, collapse="; ")
      }

      #Some of the datasets are subsetted for the differential
expression analyses
      #We will make an empty vector to store subset information for
each result id
      SubsetBy<-vector(mode="character", length(Design$result.ID))

      #Then we will determine whether the dataset is subsetted:
      if(Design$isSubset[1]==TRUE){

        #If it is subsetted, we will loop over each result id for the
dataset
        for (j in c(1:length(Design$result.ID))){

          #And grab the vector of subsetting information
          SubsetByVector<-Design$subsetFactor[[j]]$summary

          #And then collapse that information down to a single entry
that will fit in our dataframe
          SubsetBy[j]<-paste(SubsetByVector, collapse="; ")
        }

        #if the dataset wasn't subsetted for the differential
expression analysis:
      }else{
        #We'll just make a vector of NA values to put in the
"Subsetted by" column
```

```
        SubsetBy<-rep(NA, length((Design$result.ID)))
      }

      #Then we combine all of the information for all of the result
sets for the dataset into a dataframe
      ResultSets_ForExperiment<-
cbind.data.frame(ExperimentID=rep(ExperimentIDs[i],length(Design$resul
t.ID)),ResultSetIDs=Design$result.ID, ContrastIDs=Design$contrast.ID,
ExperimentIDs=Design$experiment.ID,
FactorCategory=Design$factor.category, ExperimentalFactors,
BaselineFactors, Subsetted=Design$isSubset, SubsetBy)

      #And add that information as rows to our data frame including
the result set information for all datasets:
      ResultSets_toScreen<-rbind.data.frame(ResultSets_toScreen,
ResultSets_ForExperiment)

      #Then clean up our space before looping to the next dataset:
      rm(ResultSets_ForExperiment, Design, ExperimentalFactors,
BaselineFactors, SubsetBy)

    }else{
      rm(Design)
    }

  }

  #When we're done, we'll want to remove the initial (empty) row in
our data.frame:
  ResultSets_toScreen<-ResultSets_toScreen[-1,]

  #We can make some empty vectors that we can use to store screening
notes:
  Include<-vector(mode="character", length=nrow(ResultSets_toScreen))
  WrongBaseline<-vector(mode="character",
length=nrow(ResultSets_toScreen))
  ResultsNotRegionSpecific<-vector(mode="character",
length=nrow(ResultSets_toScreen))
  ReAnalyze<-vector(mode="character",
length=nrow(ResultSets_toScreen))

  #And add them as columns to our dataframe:
  ResultSets_toScreen<-cbind.data.frame(ResultSets_toScreen, Include,
WrongBaseline, ResultsNotRegionSpecific, ReAnalyze)

  #And then write everything out as a .csv file that we can easily
mark up in a spreadsheet program:
```

```
mark up in a spreadsheet program.
  write.csv(ResultSets_toScreen, "ResultSets_toScreen.csv")

  print("The Result Sets for your Datasets have been outputted into
ResultSets_toScreen.csv")
  print(str(ResultSets_toScreen))

  #And clean up our environment:
  rm(Include, WrongBaseline, ResultsNotRegionSpecific, ReAnalyze)
}
```

Command

Example R Function Usage: Getting result ids and contrast ids for selected datasets

```
#This function makes a data.frame that includes all of the result ids
and contrast ids for each dataset with their basic metadata in a
format (.csv) that is easily readable in a spreadsheet program.

#You can input this function by running the code discussed above to
create the function in your R environment.

#Alternatively, you can download the script for the function from our
Github site and save the file in your working directory:
#https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_G
emmaDEResults_2024/Function_GettingResultSetInfoForDatasets.R

#And then source it from your working directory:
source("Function_GettingResultSetInfoForDatasets.R")

#Example function usage:

GettingResultSetInfoForDatasets(ExperimentIDs)

#The results from this functions will be outputted as a .csv file in
the working directory named "ResultSets_toScreen.csv"
```

Note

The results from this function will be outputted as a .csv file in the working directory named "ResultSets_toScreen.csv"

8.3 Open the file containing the result set and statistical contrast metadata ("ResultSets_toScreen.csv") using a standard spreadsheet software program, like Excel or Google Sheets. Immediately re-save the file in a spreadsheet format (e.g., .xlsx) - otherwise any changes that you make to the formatting (like color) will be lost when you close the file (.csv files do not include formatting).

Note

The file "ResultSets_toScreen.csv" will be located in your current working directory.
We are switching over to looking at the result set and contrast metadata in a spreadsheet software program instead of R because it is makes it easier to explore and annotate the metadata records. That said, because we are switching over to making changes to the dataset metadata by hand (instead of using coding), make sure that you take good notes about what changes you make (and why!).

8.4 When reviewing the available result sets and statistical contrast statistical contrasts should be excluded using pre-specified criteria.

Note

Pre-specified exclusion criteria:

Result set exclusion criteria:

1) Differential expression results for the dataset in Gemma are not specific to the targeted tissue/ ROI.

For datasets that include samples from more than one tissue or brain region, the result sets should be subsetted so that there is a result set for each brain region or tissue (instead of including brain region ("OrganismPart") as a factor in the model). If the result sets haven't been subsetted by region, we will need to either exclude the dataset or re-run the differential expression analysis.

(Depending on the goals of the meta-analysis, we may also need to re-run the differential expression analysis to remove other unwanted subjects - e.g., removing subjects with genotypes that might interfere with our results)

Statistical contrast exclusion criteria:

1) Statistical contrast (differential expression results) do not reflect the experimental manipulation of interest.

For datasets that include multiple variables (e.g., sex, treatment, stress intervention, genotype), there will be a statistical contrast (differential expression output) for each variable that was included in the statistical model, as well as occasionally differential expression output for their interacting effects.
We are only interested in the statistical contrasts (differential expression results) that represent the effect of our experimental manipulation or variable of interest.

8.5 For the statistical contrasts that remain, we need to double-check that the contrasts include an appropriate reference group (baseline) for our research question. If not, make a note that we will need to later invert the direction of effect in the differential expression results.

Note

Sometimes the reference (baseline) and experimental groups are reversed in Gemma (e.g., having the drug treatment set as the reference/baseline, with vehicle as the manipulation). If this is the case, we will need to invert the effects (Log2 Fold Changes) when we input them into our meta-analysis (multiply the effects by -1).

8.6 Copy the result set and contrast id metadata worksheet and use it to make a new (second) worksheet in the same file. For this new worksheet, delete all metadata records for result sets and contrast ids that were excluded so that it is easier to easily view/read the remaining potential datasets.

8.7 Copy the result set and contrast id metadata for this final set of selected contrasts into a new .csv file and save it in your working directory with the name "ResultSets_Screened.csv".

8.8 Following this tertiary filtering of result sets and contrasts by the individual experimenter, the 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 2024 cohort of the Brain Data Alchemy Project, these inclusion/exclusion decisions were reviewed by the mentor (Dr. Hagenauer).

8.9 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.

8.10 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/1KptuleWQH7B6SDk5z7K-Dub2beMSnmSa/edit?usp=sharing&ouid=106595687423493776462&rtpof=true&sd=true

Result Extraction: Gemma's Differential Expression Results

9 Result Extraction: Gemma's Differential Expression Results. This coding is all performed within the R environment using Rstudio.

Note

A detailed version of example code can be found here: https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_GemmaDEResults_2024/2024_Example_GemmaDEResults_ImportPipeline.R

9.1 Import the file containing the final selected result sets and contrast ids ("ResultSets_Screened.csv") into R.

Command

Example R Code: Import the final selected result sets and contrast ids into R

```
ResultSet_contrasts<-read.csv("ResultSets_Screened.csv", header=TRUE,
stringsAsFactors = FALSE )
```

9.2 Download the differential expression results from Gemma for the selected result sets and statistical contrasts.

Command

Example R Function: Downloading differential expression results from Gemma for selected result sets

```
DownloadingDEResults<-function(ResultSet_contrasts){

  #Some ResultSets have more than one statistical contrast, so they
are present more than once in our data frame, e.g.:
  #ResultSet_contrasts$ResultSetIDs
  #[1] 553805 553805 553805 570552 556647

  #To pull down the statistical results, we'll only want the unique
result set ids:
  UniqueResultSetIDs<-unique(ResultSet_contrasts$ResultSetIDs)

  print("These are the result sets that you identified as being of
interest")
  print(UniqueResultSetIDs)
  #553805 570552 556647

  differentials <- UniqueResultSetIDs %>% lapply(function(x){
    #   # take the first and only element of the output. the function
returns a list
    #   # because single experiments may have multiple resultSets.
Here we use the
    #   # resultSet argument to directly access the results we need
    get_differential_expression_values(resultSet = x)[[1]]
  })

  str(differentials)
  #That code worked. Excellent!

  # # some datasets might not have all the advertised differential
expression results
  # # calculated due to a variety of factors. here we remove the
empty differentials
  missing_contrasts <- differentials %>% sapply(nrow) %>% {.==0}
  #[1] FALSE FALSE FALSE
  differentials <<- differentials[!missing_contrasts]
  UniqueResultSetIDs<<-UniqueResultSetIDs[!missing_contrasts]

  print("These are the result sets that had differential expression
results:")
```

```
results. ")
  print(UniqueResultSetIDs)

  print("Your differential expression results for each of your result
sets are stored in the object named differentials. This object is
structured as a list of data frames. Each element in the list
represetns a result set, with the data frame containing the
differential expression results")

  #Within any particular Result Set, there are likely to be some
contrasts that we want and others that we don't want
  #For example, a result set might contain a variety of stress
interventions
  #And maybe we only want the acute stress contrast results

  #We already identified which statistical contrasts we wanted during
our screening:
  #This is the object with the specific contrast ids that we want:
  #ResultSet_contrasts$ContrastIDs

  #Which will be these columns within the listed dataframes of
differential expression results:

  print("These are the columns for the effect sizes for our
statistical contrasts of interest (Log(2) Fold Changes")
  Contrasts_Log2FC<<-paste("contrast_",
ResultSet_contrasts$ContrastIDs, "_log2fc", sep="")

  print(Contrasts_Log2FC)
  #[1] "contrast_151618_log2fc" "contrast_151617_log2fc"
"contrast_151619_log2fc"
  #[4] "contrast_186753_log2fc" "contrast_204289_log2fc"

  print("these are the columns for the T-statistics for our
statistical contrasts of interest - we will use that information to
derive the sampling variances")

  Contrasts_Tstat<<-paste("contrast_",
ResultSet_contrasts$ContrastIDs, "_tstat", sep="")

  print(Contrasts_Tstat)
  # [1] "contrast_151618_tstat" "contrast_151617_tstat"
"contrast_151619_tstat"
  # [4] "contrast_186753_tstat" "contrast_204289_tstat"

}
```

Command

Example R function Usage: Downloading differential expression results for selected result sets and statistical contrasts

```
#You can input this function by running the code discussed above to
create the function in your R environment.

#Alternatively, you can download the script for the function from our
Github site and save the file in your working directory:
#https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_G
emmaDEResults_2024/Function_DownloadingDEResults.R

#And then source it from your working directory:
source("Function_DownloadingDEResults.R")

#Example usage for the function:

DownloadingDEResults(ResultSet_contrasts)

#Example output:

# [1] "These are the result sets that you identified as being of
interest"
# [1] 553805 570552 556647
# List of 3
# $ :Classes ‘data.table’ and 'data.frame':     21693 obs. of  19
variables:
#   ..$ Probe                      : int [1:21693] 20344 20558 110454
20339 17067 68774 21946 21366 76905 16190 ...
# ..$ NCBIid                     : int [1:21693] 20344 20558 110454
20339 17067 68774 21946 21366 76905 16190 ...
# ..$ GeneSymbol                 : chr [1:21693] "Selp" "Slfn4"
"Ly6a" "Sele" ...
# ..$ GeneName                   : chr [1:21693] "selectin, platelet"
"schlafen 4" "lymphocyte antigen 6 family member A" "selectin,
endothelial cell" ...
# ..$ pvalue                     : num [1:21693] 1.70e-10 9.11e-09
5.38e-08 7.73e-08 1.00e-07 ...
# ..$ corrected_pvalue           : num [1:21693] 3.69e-06 9.88e-05
4.00e-04 4.00e-04 4.00e-04 ...
# ..$ rank                       : num [1:21693] 4.61e-05 9.22e-05
1.00e-04 2.00e-04 2.00e-04 ...
#   $ contrast_151617_coefficient: num [1:21693] 3.102 3.27 0.763
```

```
# ..$ contrast_151617_coefficient: num [1:21693] 3.102 3.27 0.763
2.868 0.513 ...
# ..$ contrast_151617_log2fc     : num [1:21693] 3.102 3.27 0.763
2.868 0.513 ...
# ..$ contrast_151617_tstat      : num [1:21693] 2.97 3.62 3.86 2.31
4.17 ...
# ..$ contrast_151617_pvalue     : num [1:21693] 0.0077 0.0017 0.001
0.0318 0.0005 ...
# ..$ contrast_151618_coefficient: num [1:21693] 7.01 5.21 1.35 5.86
0.82 ...
# ..$ contrast_151618_log2fc     : num [1:21693] 7.01 5.21 1.35 5.86
0.82 ...
# ..$ contrast_151618_tstat      : num [1:21693] 7.81 6.14 7.02 5.28
6.78 ...
# ..$ contrast_151618_pvalue     : num [1:21693] 1.86e-07 5.64e-06
9.01e-07 3.78e-05 1.47e-06 ...
# ..$ contrast_151619_coefficient: num [1:21693] 1.223 -0.303 -0.243
0.118 -0.214 ...
# ..$ contrast_151619_log2fc     : num [1:21693] 1.223 -0.303 -0.243
0.118 -0.214 ...
# ..$ contrast_151619_tstat      : num [1:21693] 1.0192 -0.2593
-1.1221 0.0785 -1.6535 ...
# ..$ contrast_151619_pvalue     : num [1:21693] 0.32 0.798 0.275
0.938 0.114 ...
# ..- attr(*, ".internal.selfref")=<externalptr>
#   ..- attr(*, "call")= chr
"https://gemma.msl.ubc.ca/rest/v2/resultSets/553805"
# ..- attr(*, "env")=<environment: 0x7f897aec2838>
#   $ :Classes ‘data.table’ and 'data.frame':   45100 obs. of  11
variables:
#   ..$ Probe                      : chr [1:45100] "1426114_PM_at"
"1452267_PM_at" "1441388_PM_at" "1436221_PM_at" ...
# ..$ NCBIid                     : chr [1:45100] "" "224613" ""
"100039795" ...
# ..$ GeneSymbol                 : chr [1:45100] "" "Flywch1" ""
"Ildr2" ...
# ..$ GeneName                   : chr [1:45100] "" "FLYWCH-type zinc
finger 1" "" "immunoglobulin-like domain containing receptor 2" ...
# ..$ pvalue                     : num [1:45100] 3.74e-09 1.57e-08
1.32e-08 8.87e-09 2.37e-08 ...
# ..$ corrected_pvalue           : num [1:45100] 2e-04 2e-04 2e-04 2e-
04 2e-04 3e-04 3e-04 3e-04 4e-04 5e-04 ...
# ..$ rank                       : num [1:45100] 2.22e-05 8.87e-05
6.65e-05 4.44e-05 1.00e-04 ...
# ..$ contrast_186753_coefficient: num [1:45100] 0.1587 0.0967
-0.5049 0.7694 0.1039 ...
# ..$ contrast_186753_log2fc     : num [1:45100] 0.1587 0.0967
```

```
-0.5049 0.7694 0.1039 ...
# ..$ contrast 186753 tstat      : num [1:45100] 1.65 2.28 -5.4 10.4
8.18e-05 3.93e-08 5.26e-02
# - attr(*, ".internal.selfref")=<externalptr>
# - attr(*, "env")=<environment: 0x7f897ab59f78>
#   $ :Classes 'data.table' and 'data.frame':   18964 obs. of  11
50 110 498386 299052 25636 65166 102548697 100364561 24183 ...
# ..$ NCBIid                     : int [1:18964] 497918 303604 501110
498386 299052 25636 65166 102548697 100364561 24183 ...
```

Note

This function may take a while to run - especially if you are downloading a large number of result sets from Gemma.
The output from the function is a single object (*differentials*) that stores the differential expression results for all of the result sets, listed numerically.
E.g, to access the first result set: differentials[[1]], to access the second result set: differentials[[2]], etc.
Note that some datasets might not have all the advertised differential expression results calculated due to a variety of factors. The function will remove these empty differential expression results and then provide a full list of which differential expression results were available.

9.3 For record-keeping purposes, output and save the differential expression results for each result set.

Note

The reason to output and save the differential expression results for each result set is to make the meta-analysis more reproducible. Gemma updates the annotation and probe/read alignment for the transcriptional profiling data every time a new reference genome is released.
Therefore, our meta-analysis results would not be able to be fully reproduced in the future with just our code.

Command

Example R Function: Saving the differential expression results for each result set

```
SavingGemmaDEResults_forEachResultSet<-function(differentials,
UniqueResultSetIDs, ResultSet_contrasts){

  for (i in c(1:length(differentials))){

    ThisResultSet<-UniqueResultSetIDs[i]

    #Pulling out the dataset name from our other data frame
    #For some reason I can find this in the Gemma ResultSet
differential expression output
    #Since some datasets have multiple result sets, we just grab the
dataset name from the first entry
    ThisDataSet<-
ResultSet_contrasts$ExperimentID[ResultSet_contrasts$ResultSetIDs==Thi
sResultSet][1]

    #Write out a data frame containing the differential expression
output for the result set
    #And name it with the dataset id and result set id:
    write.csv(differentials[[i]], paste("DEResults", ThisDataSet,
ThisResultSet, ".csv", sep="_"))

    rm(ThisDataSet, ThisResultSet)
  }
}
```

Command

Example R Function Usage: Saving the differential expression results for each result set

```
#You can input this function by running the code discussed above to
create the function in your R environment.

#Alternatively, you can download the script for the function from our
Github site and save the file in your working directory:
#https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_G
emmaDEResults_2024/Function_SavingGemmaDEResults_forEachResultSet.R

#And then source it from your working directory:
source("Function_SavingGemmaDEResults_forEachResultSet.R")

#Example usage:

SavingGemmaDEResults_forEachResultSet(differentials,
UniqueResultSetIDs, ResultSet_contrasts)
```

9.4 The next few processing steps (steps #9.4-9.10) need to be applied to a single result set at a time.

To begin applying these processing steps, we will need to begin by pulling the result set out from the differentials object (which contains all of the result sets, listed numerically) and temporarily making it a new object (*DE_Results*).

Command

Example R Code: Making the differential expression results for a result set its own temporary object

```
#Here is an example of pulling out the results of the first result
set in the differentials object:

DE_Results<-differentials[[1]]

#The structure of the new DE_Results object:
str(DE_Results)
#Classes ‘data.table’ and 'data.frame': 21693 obs. of  19 variables:


#After finishing processing that first result set (steps #9.4-#9.8),
we would move on to the second result set.
#To pull out the results for the second result set, we would use this
code:

DE_Results<-differentials[[2]]
```

9.5 Within the object containing the differential expression results for a dataset, rows that lack unambiguous EntrezID annotation (including “”, “null”, '\\|' in the "NCBIid" column) need to be counted and excluded.

Command

Example R Function: Remove rows of data that have missing or unambiguous gene annotation

```
FilteringDEResults_GoodAnnotation<-function(DE_Results){

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

  print("# of rows with missing NCBI annotation:")
  print(sum(DE_Results$NCBIid==""|DE_Results$NCBIid=="null"))

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

  print("# of rows with missing Gene Symbol annotation:")
  print(sum(DE_Results$GeneSymbol==""|DE_Results$GeneSymbol=="null"))

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

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

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

  #I also only want the subset of data that is annotated with a
single gene (not ambiguously mapped to more than one gene)
  if(length(grep('\\|', DE_Results_NoNA$NCBIid))==0){
    DE_Results_GoodAnnotation<<-DE_Results_NoNA
  }else{
    #I only want rows annotated with a single Gene Symbol (no pipe):
    DE_Results_GoodAnnotation<<-DE_Results_NoNA[-(grep('\\|',
DE_Results_NoNA$NCBIid)),]
  }
  #I used a double arrow in that conditional to place
DE_Results_GoodAnnotation back out in the environment outside the
function
```

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

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

  rm(DE_Results_NoNA, DE_Results)

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

Command

Example R Function Usage: Remove rows of data that have missing or unambiguous gene annotation

```
#Cleaning up the results for a single result set

#Reading in the function

#You can input this function by running the code discussed above to
create the function in your R environment.

#Alternatively, you can download the script for the function from our
Github site and save the file in your working directory:
#https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_G
emmaDEResults_2024/Function_FilteringDEResults_GoodAnnotation.R

#And then source it from your working directory:
source("Function_FilteringDEResults_GoodAnnotation.R")

#Example of using the function for a dataset:

FilteringDEResults_GoodAnnotation(DE_Results)

# [1] "# of rows in results"
# [1] 21693
# [1] "# of rows with missing NCBI annotation:"
# [1] NA
# [1] "# of rows with NA NCBI annotation:"
# [1] 79
# [1] "# of rows with missing Gene Symbol annotation:"
# [1] 79
# [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] 21614
# [1] "Outputted object: DE_Results_GoodAnnotation"

str(DE_Results_GoodAnnotation)
# Classes ‘data.table’ and 'data.frame':        21614 obs. of  19
variables:
#  $ Probe                       : int  20344 20558 110454 20339
```

```
# $ ProbeId                    : int  20344 20558 110454 20339
17067 68774 21946 21366 76905 16190 ...
# $ NCBIid                     : int  20344 20558 110454 20339 17067
68774 21946 21366 76905 16190 ...
# $ GeneSymbol                 : chr  "Selp" "Slfn4" "Ly6a" "Sele" ...
# $ GeneName                   : chr  "selectin, platelet" "schlafen
4" "lymphocyte antigen 6 family member A" "selectin, endothelial
cell" ...
# $ pvalue                     : num  1.70e-10 9.11e-09 5.38e-08
7.73e-08 1.00e-07 ...
# $ corrected_pvalue           : num  3.69e-06 9.88e-05 4.00e-04
4.00e-04 4.00e-04 ...
# $ rank                       : num  4.61e-05 9.22e-05 1.00e-04
2.00e-04 2.00e-04 ...
# $ contrast_151617_coefficient: num  3.102 3.27 0.763 2.868 0.513 ...
# $ contrast_151617_log2fc     : num  3.102 3.27 0.763 2.868 0.513 ...
# $ contrast_151617_tstat      : num  2.97 3.62 3.86 2.31 4.17 ...
# $ contrast_151617_pvalue     : num  0.0077 0.0017 0.001 0.0318
0.0005 ...
# $ contrast_151618_coefficient: num  7.01 5.21 1.35 5.86 0.82 ...
# $ contrast_151618_log2fc     : num  7.01 5.21 1.35 5.86 0.82 ...
# $ contrast_151618_tstat      : num  7.81 6.14 7.02 5.28 6.78 ...
# $ contrast_151618_pvalue     : num  1.86e-07 5.64e-06 9.01e-07
3.78e-05 1.47e-06 ...
# $ contrast_151619_coefficient: num  1.223 -0.303 -0.243 0.118
-0.214 ...
# $ contrast_151619_log2fc     : num  1.223 -0.303 -0.243 0.118
-0.214 ...
# $ contrast_151619_tstat      : num  1.0192 -0.2593 -1.1221 0.0785
-1.6535 ...
# $ contrast_151619_pvalue     : num  0.32 0.798 0.275 0.938 0.114 ...
# - attr(*, ".internal.selfref")=<externalptr>
#   - attr(*, "call")= chr
"https://gemma.msl.ubc.ca/rest/v2/resultSets/553805"
# - attr(*, "env")=<environment: 0x7f89998ffea8>
```

Note

Note: If gene annotation (especially Entrez ID) is missing for the entire result set, additional annotation information for the transcriptional profiling platform can sometimes be downloaded from Gene Expression Omnibus. This sometimes occurs for microarray datasets, which may only have ProbeID annotation for each row of data.

Gene Expression Omnibus transcriptional profiling platform information: https://www.ncbi.nlm.nih.gov/geo/browse/?view=platforms

This information would then need to be read into R, and joined to the differential expression result file using customized code before running the current processing step to remove rows lacking annotation.

9.6 Extract the specific differential expression results for the statistical contrasts (group comparisons) of interest (*i.e.,* related to your research question).

Command

Example R Function: Extract the specific differential expression results for the statistical contrasts of interest

```
#This code includes a function within a function.

#This is the inner function:
GetContrastIDsforResultSet<-function(NamesOfFoldChangeColumns){
  #I split apart the column names:
  ColumnNames_BrokenUp<-strsplit(NamesOfFoldChangeColumns, "_")
  #Put them in a matrix format
  MatrixOfColumnNames_BrokenUp<-matrix(unlist(ColumnNames_BrokenUp),
ncol=3,byrow=T)
  #And then grab the contrast ids:
  ContrastIDs_inCurrentDF<-MatrixOfColumnNames_BrokenUp[,2]
  rm(ColumnNames_BrokenUp, MatrixOfColumnNames_BrokenUp)
  return(ContrastIDs_inCurrentDF)
}

#This is the outer function:

ExtractingDEResultsForContrasts<-function(DE_Results_GoodAnnotation,
Contrasts_Log2FC, Contrasts_Tstat, ResultSet_contrasts){

  print("These are all of the columns in the differential expression
results for our current result set:")

  print(colnames(DE_Results_GoodAnnotation))
  # [1] "Probe"                       "NCBIid"
  # [3] "GeneSymbol"                  "GeneName"
  # [5] "pvalue"                      "corrected_pvalue"
  # [7] "rank"                        "contrast_151617_coefficient"
  # [9] "contrast_151617_log2fc"      "contrast_151617_tstat"
  # [11] "contrast_151617_pvalue"      "contrast_151618_coefficient"
  # [13] "contrast_151618_log2fc"      "contrast_151618_tstat"
  # [15] "contrast_151618_pvalue"      "contrast_151619_coefficient"
  # [17] "contrast_151619_log2fc"      "contrast_151619_tstat"
  # [19] "contrast_151619_pvalue"

  print("These are the names of the Log(2) Fold Change Columns for
our statistical contrasts of interest within the differential
expression results for this particular result set:")
```

```
  NamesOfFoldChangeColumns<<-colnames(DE_Results_GoodAnnotation)
[colnames(DE_Results_GoodAnnotation)%in%Contrasts_Log2FC]

  print(NamesOfFoldChangeColumns)
  #[1] "contrast_151617_log2fc" "contrast_151618_log2fc"
"contrast_151619_log2fc"

  print("These are the names of the T-statistic Columns for our
statistical contrasts of interest within the differential expression
results for this particular result set:")

  NamesOfTstatColumns<<-colnames(DE_Results_GoodAnnotation)
[colnames(DE_Results_GoodAnnotation)%in%Contrasts_Tstat]

  print(NamesOfTstatColumns)
  #[1] "contrast_151617_tstat" "contrast_151618_tstat"
"contrast_151619_tstat"

  #Next we're going to pull out the contrast IDs associated with each
result:

  ContrastIDs_inCurrentDF<-
GetContrastIDsforResultSet(NamesOfFoldChangeColumns)

  print("These are the contrast ids for the statistical contrasts of
interest within your current result set:")
  print(ContrastIDs_inCurrentDF)
  #[1] "151617" "151618" "151619"

  #Next we're going to grab some metadata to go with those
statistical contrasts

  print("This is the dataset id for the result set and statistical
contrasts:")
  Datasets_inCurrentDF<-
ResultSet_contrasts$ExperimentID[ResultSet_contrasts$ContrastIDs%in%Co
ntrastIDs_inCurrentDF]

  GSE_ID<<-Datasets_inCurrentDF[1]

  print(GSE_ID)
  #[1] "GSE126678"

  #And I would like the experimental factor information for our
statistical contrast:
  Factors_inCurrentDF<-
```

```
ResultSet_contrasts$ExperimentalFactors[ResultSet_contrasts$ContrastID
s%in%ContrastIDs_inCurrentDF]

  #We can combine those to make an interpretable unique identifier
for each statistical comparison:
  ComparisonsOfInterest<<-paste(Datasets_inCurrentDF,
Factors_inCurrentDF, sep="_" )

  print("These are the current names for your statistical contrasts
of interest - if they are unwieldy, you may want to change them")
  print(ComparisonsOfInterest)

  #cleaning up the workspace:
  rm(Datasets_inCurrentDF, Factors_inCurrentDF,
ContrastIDs_inCurrentDF)

}
```

Command

Example R Function Usage: Extracting the specific differential expression results for the statistical contrasts of interest

```
#Read in the function:

#You can input this function by running the code discussed above to
create the function in your R environment.

#Alternatively, you can download the script for the function from our
Github site and save the file in your working directory:
#https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_G
emmaDEResults_2024/Function_ExtractingDEResultsForContrasts.R

#And then source it from your working directory:
source("Function_ExtractingDEResultsForContrasts.R")

#Example function usage:
ExtractingDEResultsForContrasts(DE_Results_GoodAnnotation,
Contrasts_Log2FC, Contrasts_Tstat, ResultSet_contrasts)

# [1] "These are all of the columns in the differential expression
results for our current result set:"
# [1] "Probe"                       "NCBIid"
# [3] "GeneSymbol"                  "GeneName"
# [5] "pvalue"                      "corrected_pvalue"
# [7] "rank"                        "contrast_151617_coefficient"
# [9] "contrast_151617_log2fc"      "contrast_151617_tstat"
# [11] "contrast_151617_pvalue"      "contrast_151618_coefficient"
# [13] "contrast_151618_log2fc"      "contrast_151618_tstat"
# [15] "contrast_151618_pvalue"      "contrast_151619_coefficient"
# [17] "contrast_151619_log2fc"      "contrast_151619_tstat"
# [19] "contrast_151619_pvalue"
# [1] "These are the names of the Log(2) Fold Change Columns for our
statistical contrasts of interest within the differential expression
results for this particular result set:"
# [1] "contrast_151617_log2fc" "contrast_151618_log2fc"
"contrast_151619_log2fc"
# [1] "These are the names of the T-statistic Columns for our
statistical contrasts of interest within the differential expression
results for this particular result set:"
# [1] "contrast_151617_tstat" "contrast_151618_tstat"
"contrast_151619_tstat"
```

```
# [1] "These are the contrast ids for the statistical contrasts of
interest within your current result set:"
# [1] "151617" "151618" "151619"
# [1] "This is the dataset id for the result set and statistical
contrasts:"
# [1] "GSE126678"
# [1] "These are the current names for your statistical contrasts of
interest - if they are unwieldy, you may want to change them"
# [1] "GSE126678_lipopolysaccharide has modifier Acute immune
challenge , vehicle; lipopolysaccharide has modifier Acute immune
challenge , vehicle"
# [2] "GSE126678_lipopolysaccharide has modifier Long-term subchronic
immune challenge + acute immune challenge"

# [3] "GSE126678_lipopolysaccharide has modifier long-term subchronic
immune challenge , vehicle; lipopolysaccharide has modifier long-term
subchronic immune challenge , vehicle"


#Those names are super unwieldy. I'm going to rename them:
#Note: Make sure the name still includes the GEO dataset id number
(GSE...)
ComparisonsOfInterest<-c("GSE126678_LPS_Acute",
"GSE126678_LPS_SubchronicPlusAcute", "GSE126678_Subchronic")
```

9.7 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 9.9)*

Note

Log2FC/Tstat=SE

9.8 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 9.9)*

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.

9.9 If there is more than one row of results representing the same gene (same Entrez ID), the Log2FC and SE's should be averaged for each gene EntrezID using the *tapply()* function.

Command

R Function: Calculating SV and Collapsing DE Results to One Result Per Gene

```
CollapsingDEResults_OneResultPerGene<-function(GSE_ID,
DE_Results_GoodAnnotation, ComparisonsOfInterest,
NamesOfFoldChangeColumns, NamesOfTstatColumns){

  print("Double check that the vectors containing the fold change and
tstat column names contain the same order as the group comparisons of
interest - otherwise this function won't work properly!  If the order
matches, proceed:")

  print("# of rows with unique NCBI IDs:")
  print(length(unique(DE_Results_GoodAnnotation$NCBIid)))

  print("# of rows with unique Gene Symbols:")
  print(length(unique(DE_Results_GoodAnnotation$GeneSymbol)))

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

  #And then sets the working directory to be that folder:
  setwd(paste("./", GSE_ID, sep=""))

  #Calculating the average Log2FC and T-stat for each gene:

  #Creating an empty list to store our results from each of our
statistical contrasts:
  DE_Results_GoodAnnotation_FoldChange_Average<-list()
  DE_Results_GoodAnnotation_Tstat_Average<-list()
  DE_Results_GoodAnnotation_SE_Average<-list()

  #For each of the columns containing fold change and t-stat
information for our statistical contrasts of interest:

  for(i in c(1:length(NamesOfFoldChangeColumns))){

    #Grab our Log2FC column of interest:
    FoldChangeColumn<-select(DE_Results_GoodAnnotation,
NamesOfFoldChangeColumns[i])

    #Grab our Tstat column of interest:
```

```
    TstatColumn<-select(DE_Results_GoodAnnotation,
NamesOfTstatColumns[i])

    #Calculate the SE:
    DE_Results_GoodAnnotation_SE<-
FoldChangeColumn[[1]]/TstatColumn[[1]]

    #Calculate the average Log2FC per gene:
    DE_Results_GoodAnnotation_FoldChange_Average[[i]]<-
tapply(FoldChangeColumn[[1]], DE_Results_GoodAnnotation$NCBIid, mean)

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

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

  }

  #Currently we have all of this averaged Log2FC information stored
in a list, with one entry for each statistical contrast
  #Let's convert this to a data frame
  DE_Results_GoodAnnotation_FoldChange_AveragedByGene<-do.call(cbind,
DE_Results_GoodAnnotation_FoldChange_Average)

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

  #Name the columns in the dataframe in a manner that describes the
dataset and factors for each statistic contrast
  colnames(DE_Results_GoodAnnotation_FoldChange_AveragedByGene)<-
ComparisonsOfInterest

  #and then write it out to save it:
  write.csv(DE_Results_GoodAnnotation_FoldChange_AveragedByGene,
"DE_Results_GoodAnnotation_FoldChange_AveragedByGene.csv")

  #And do the same for the T-stats:
  DE_Results_GoodAnnotation_Tstat_AveragedByGene<-do.call(cbind,
DE_Results_GoodAnnotation_Tstat_Average)

  colnames(DE_Results_GoodAnnotation_Tstat_AveragedByGene)<-
ComparisonsOfInterest
```

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

  #And the same for the SE:
  DE_Results_GoodAnnotation_SE_AveragedByGene<-do.call(cbind,
DE_Results_GoodAnnotation_SE_Average)

  colnames(DE_Results_GoodAnnotation_SE_AveragedByGene)<-
ComparisonsOfInterest

  write.csv(DE_Results_GoodAnnotation_SE_AveragedByGene,
"DE_Results_GoodAnnotation_SE_AveragedByGene.csv")

  #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.

  DE_Results_GoodAnnotation_SV<-
(DE_Results_GoodAnnotation_SE_AveragedByGene)^2

  #Writing that information out to store it:
  write.csv(DE_Results_GoodAnnotation_SV,
"DE_Results_GoodAnnotation_SV.csv")

  #Compiling all of our results into a single, big object
  TempMasterResults<-
list(Log2FC=DE_Results_GoodAnnotation_FoldChange_AveragedByGene,
Tstat=DE_Results_GoodAnnotation_Tstat_AveragedByGene,
SE=DE_Results_GoodAnnotation_SE_AveragedByGene,
SV=DE_Results_GoodAnnotation_SV)

  #And then sending that object out into our environment outside the
function
  #With a name that we can read and understand (DEResults for a
particular GSEID)
  assign(paste("DEResults", GSE_ID, sep="_"), TempMasterResults,
envir = as.environment(1))

  #Letting us know what that name is:
  print(paste("Output: Named DEResults", GSE_ID, sep="_"))

  #And then cleaning up our environment of temporary results:
  rm(TempMasterResults, DE_Results_GoodAnnotation,
DE_Results_GoodAnnotation_SV, DE_Results_GoodAnnotation_SE,
DE_Results_GoodAnnotation_FoldChange_AveragedByGene,
DE_Results_GoodAnnotation_FoldChange_Average
```

```
DE_Results_GoodAnnotation_FoldChange_Average,
DE_Results_GoodAnnotation_Tstat_AveragedByGene,
DE_Results_GoodAnnotation_Tstat_Average,
DE_Results_GoodAnnotation_SE_Average, FoldChangeColumn, TstatColumn,
GSE_ID, ComparisonsOfInterest, NamesOfFoldChangeColumns,
NamesOfTstatColumns)

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

}
```

Command

Example R Function Usage: Calculating SV and collapsing DE results to one result per gene

```
#Reading in the function:

#You can input this function by running the code discussed above to
create the function in your R environment.

#Alternatively, you can download the script for the function from our
Github site and save the file in your working directory:
#https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_G
emmaDEResults_2024/Function_CollapsingDEResults_OneResultPerGene.R

#And then source it from your working directory:
source("Function_CollapsingDEResults_OneResultPerGene.R")

#Notes about parameters for function
CollapsingDEResults_OneResultPerGene()
#GSE_ID is a string indicating the name of the Gemma dataset
#DE_Results_GoodAnnotation is the data frame outputted by our
previous function
#ComparisonsOfInterest is a character vector containing the names of
the group comparisons of interest within this dataset. Important:
These group comparisons should be listed in exactly the same order as
the order that you provide the column names for their associated Fold
Change and Tstat output.
#NamesOfFoldChangeColumns is a vector containing the names of the
columns of DE_Results_GoodAnnotation containing the FoldChange
results for your comparisons of interes, in the same order as the
ComparisonsOfInterest vector
#NamesOfTstatColumns is a vector containing the names of the columns
of DE_Results_GoodAnnotation containing the Tstat results for your
comparisons of interes, in the same order as the
ComparisonsOfInterest vector

#Example usage:
CollapsingDEResults_OneResultPerGene(GSE_ID,
DE_Results_GoodAnnotation, ComparisonsOfInterest,
NamesOfFoldChangeColumns, NamesOfTstatColumns)

# [1] "Double check that the vectors containing the fold change and
tstat column names contain the same order as the group comparisons of
```

```
tstat column names contain the same order as the group comparisons of
interest - otherwise this function won't work properly!  If the order
matches, proceed:"
# [1] "# of rows with unique NCBI IDs:"
# [1] 21614
# [1] "# of rows with unique Gene Symbols:"
# [1] 21614
# [1] "Dimensions of Fold Change matrix, averaged by gene symbol:"
# [1] 21614     3
# [1] "Output: Named DEResults_GSE126678"
```

9.10 The reference group for the statistical comparison should be confirmed to fit expectations (e.g., represent the appropriate control or baseline group).

Note

If the expected reference and treatment groups are reversed, the effect sizes for the differential expression output for the contrast (Log2 Fold Change or Log2FC) can be multiplied by -1.

9.11 ... and then move on to repeat steps #9.4-#9.10 for each of the other result sets.

9.12 Make sure you save the R code and R workspace!

Note

Based on your project settings, this may happen automatically.

In the GUI (drop down menus), the code file is saved using:
"File"->"Save"
The code file will have a file extension ".R"

Whereas the workspace is saved using:
"Session"->"Save workspace as"
The workspace file will have a file extension ".Rdata"

An aside:

Many times you will hear advice *not* to save the workspace because it should be able to be completely recreated using your code.

Our situation is an exception: We *definitely* want to save the workspace because it can take a lot of time to import all of the results from Gemma's API and process them.
Also, saving the workspace makes it easier to go back and recreate figures and results when needed for publications, etc, because if we import the results from Gemma again, they may be slightly different due to updates to the genome build and annotation.

Aligning Differential Expression Results Across Datasets and Species

10 Align the differential expression results (Log2FC, SV) from the different datasets and result sets to make a single data frame, with each row representing one gene and each column representing the differential expression results from one statistical contrast.

Note

Each dataset has differential expression results from a slightly different list of genes. The list of genes with measurements in any particular dataset can vary with the exact tissue dissected, the sensitivity of the transcriptional profiling platform, the representation of transcripts targeted by the transcriptional profiling platform (for microarray), and the experimental conditions.

The differential expression results from different datasets will also be in a slightly different order.
We want to align these results so that the differential expression results from each dataset are columns, with each row representing a different gene.

10.1 If there are datasets derived from mice, align them using mouse Entrez ID (“NCBI_ID”).

Note

The output of this function will be a single data frame for Log2FCs and a single data frame for the sampling variances (SVs) from all of the rat differential expression results, with each row as a mouse Entrez ID and each column representing a statistical contrast.
It is important to run this step even if there is only a single dataset from mice.

Command

Example R Function: Aligning mouse differential expression results from different datasets by EntrezID

```
#A function for aligning all of our mouse differential expression
results from different datasets into a single data frame for Log2FCs
and sampling variances (SVs):

#This function works the same way as the rat alignment function:

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))

  rm(Mouse_MetaAnalysis_SV_Dfs, Mouse_MetaAnalysis_FoldChange_Dfs)
}
```

Command

Example R Function Usage: Align mouse differential expression results from different datasets by EntrezID

```
#Reading in the function:

#You can input this function by running the code discussed above to
create the function in your R environment.

#Alternatively, you can download the script for the function from our
Github site and save the file in your working directory:
#https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_G
emmaDEResults_2024/Function_AligningDEResults.R

#And then source it from your working directory:
source("Function_AligningDEResults.R")

#Aligning the mouse datasets with each other:

#Example Usage:
#In this example, there are differential expression results from two
mouse datasets (GSE126678 and GSE181285):

ListOfMouseDEResults<-list(DEResults_GSE126678, DEResults_GSE181285)


AligningMouseDatasets(ListOfMouseDEResults)

# [1] "Mouse_MetaAnalysis_FoldChange_Dfs:"
# List of 2
# $ :'data.frame':      21614 obs. of  4 variables:
#   ..$ Mouse_EntrezGene.ID              : chr [1:21614] "11287"
"11298" "11302" "11303" ...
# ..$ GSE126678_LPS_Acute              : num [1:21614] 1.9397 0.0805
0.0595 0.0306 0.276 ...
# ..$ GSE126678_LPS_SubchronicPlusAcute: num [1:21614] 1.2967 -0.0472
-0.1459 0.1367 1.5651 ...
# ..$ GSE126678_LPS_Subchronic         : num [1:21614] 0.0582 0.203
-0.1144 0.1361 -0.0051 ...
# $ :'data.frame':      18563 obs. of  2 variables:
#   ..$ Mouse_EntrezGene.ID: chr [1:18563] "100008567" "100009600"
"100012" "100017" ...
#   $ GSE181285_LPS_Acute: num [1:18563] 0.0198 0.0225 0.0641 -0.0049
```

```
# ..$ GSE181285_LPS_Acute: num [1:18563] 0.0198 0.0225 0.0041 0.0049
-0.0588 ...
# NULL
# [1] "Mouse_MetaAnalysis_FoldChanges:"
# 'data.frame': 24287 obs. of  5 variables:
#   $ Mouse_EntrezGene.ID             : chr  "11287" "11298" "11302"
"11303" ...
# $ GSE126678_LPS_Acute             : num  1.9397 0.0805 0.0595
0.0306 0.276 ...
# $ GSE126678_LPS_SubchronicPlusAcute: num  1.2967 -0.0472 -0.1459
0.1367 1.5651 ...
# $ GSE126678_LPS_Subchronic        : num  0.0582 0.203 -0.1144
0.1361 -0.0051 ...
# $ GSE181285_LPS_Acute             : num  -0.042 -0.0368 -0.0534
0.1067 -0.5258 ...
# NULL
# [1] "Mouse_MetaAnalysis_SV_Dfs:"
# List of 2
# $ :'data.frame':      21614 obs. of  4 variables:
#   ..$ Mouse_EntrezGene.ID             : chr [1:21614] "11287"
"11298" "11302" "11303" ...
# ..$ GSE126678_LPS_Acute             : num [1:21614] 0.62127
0.14737 0.00437 0.01624 1.34369 ...
# ..$ GSE126678_LPS_SubchronicPlusAcute: num [1:21614] 0.66434
0.14559 0.00438 0.01567 0.98359 ...
# ..$ GSE126678_LPS_Subchronic        : num [1:21614] 0.84004
0.14211 0.00439 0.01592 1.40671 ...
# $ :'data.frame':      18563 obs. of  2 variables:
#   ..$ Mouse_EntrezGene.ID: chr [1:18563] "100008567" "100009600"
"100012" "100017" ...
# ..$ GSE181285_LPS_Acute: num [1:18563] 0.11456 0.01612 0.00329
0.00487 0.00719 ...
# NULL
# [1] "Mouse_MetaAnalysis_SV:"
# 'data.frame': 24287 obs. of  5 variables:
#   $ Mouse_EntrezGene.ID             : chr  "11287" "11298" "11302"
"11303" ...
# $ GSE126678_LPS_Acute             : num  0.62127 0.14737 0.00437
0.01624 1.34369 ...
# $ GSE126678_LPS_SubchronicPlusAcute: num  0.66434 0.14559 0.00438
0.01567 0.98359 ...
# $ GSE126678_LPS_Subchronic        : num  0.84004 0.14211 0.00439
0.01592 1.40671 ...
# $ GSE181285_LPS_Acute             : num  0.00391 0.02738 0.00601
0.0101 0.03332 ...
# NULL
```

10.2 If there are datasets derived from rats, align them by rat Entrez ID (“NCBI_ID”).

Note

The output of this function will be a single data frame for Log2FCs and a single data frame for the sampling variances (SVs) from all of the rat differential expression results, with each row as a rat Entrez ID and each column representing a statistical contrast.
It is important to run this step even if there is only a single dataset from rats.

Command

Example R Function: Aligning rat differential expression results from different datasets by EntrezID

```
#A function for aligning all of our rat differential expression
results from different datasets into a single data frame for Log2FCs
and sampling variances (SVs):

AligningRatDatasets<-function(ListOfRatDEResults){

  #Making an empty list to hold our results:
  Rat_MetaAnalysis_FoldChange_Dfs<-list()

  #Looping over all of the rat differential expression results:
  for(i in c(1:length(ListOfRatDEResults))){

    #Placing each of the log2FC results for each dataset into a
single list
    #Each element in the list is formatted so that the rownames are
Rat Entrez Gene ID and then there are columns containing the Log2FC
for the differential expression results:
    Rat_MetaAnalysis_FoldChange_Dfs[[i]]<-
data.frame(Rat_EntrezGene.ID=row.names(ListOfRatDEResults[[i]]
[[1]]),ListOfRatDEResults[[i]][[1]], stringsAsFactors=FALSE)
  }

  #Letting the user know the structure of the set of Log2FC
dataframes that we are starting out with:
  print("Rat_MetaAnalysis_FoldChange_Dfs:")
  print(str(Rat_MetaAnalysis_FoldChange_Dfs))

  #Running "join all" on the list to align all of the results by
Entrez Gene ID and make them a single data frame:
  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)

  #Letting the user know the structure of the dataframe that we just
created:
  print("Rat_MetaAnalysis_FoldChanges:")
  print(str(Rat_MetaAnalysis_FoldChanges))
```

```
  print(str(Rat_MetaAnalysis_FoldChanges))

  #Doing the same steps for the sampling variances:

  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))

  #Cleaning up our environment to remove unneeded objects:
  rm(Rat_MetaAnalysis_SV_Dfs, Rat_MetaAnalysis_FoldChange_Dfs)
}
```

Command

Example R Function Usage: Aligning rat differential expression results from different datasets by EntrezID

```
#Reading in the function:

#You can input this function by running the code discussed above to
create the function in your R environment.

#Alternatively, you can download the script for the function from our
Github site and save the file in your working directory:
#https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_G
emmaDEResults_2024/Function_AligningDEResults.R

#And then source it from your working directory:
source("Function_AligningDEResults.R")

#Example Usage:

#This is an example in which there are only differential expression
results from one rat dataset (GSE205325):

ListOfRatDEResults<-list(DEResults_GSE205325)

AligningRatDatasets(ListOfRatDEResults)
# [1] "Rat_MetaAnalysis_FoldChange_Dfs:"
# List of 1
# $ :'data.frame':      17196 obs. of  2 variables:
#   ..$ Rat_EntrezGene.ID    : chr [1:17196] "24153" "24157" "24158"
"24159" ...
# ..$ GSE205325_LPS_Chronic: num [1:17196] 0.3485 0.0288 -0.1887
-0.2126 0.1235 ...
# NULL
# [1] "Rat_MetaAnalysis_FoldChanges:"
# 'data.frame': 17196 obs. of  2 variables:
#   $ Rat_EntrezGene.ID    : chr  "24153" "24157" "24158" "24159" ...
# $ GSE205325_LPS_Chronic: num  0.3485 0.0288 -0.1887 -0.2126 0.1235
...
# NULL
# [1] "Rat_MetaAnalysis_SV_Dfs:"
# List of 1
# $ :'data.frame':      17196 obs. of  2 variables:
#   $ Rat_EntrezGene.ID    : chr [1:17196] "24153" "24157" "24158"
```

```
# ..$ Rat_EntrezGene.ID    : chr [1:17196] "24153" "24157" "24158"
"24159" ...
# ..$ GSE205325_LPS_Chronic: num [1:17196] 0.0745 0.0229 0.0383
0.0257 0.0135 ...
# NULL
# [1] "Rat_MetaAnalysis_SV:"
# 'data.frame': 17196 obs. of  2 variables:
#   $ Rat_EntrezGene.ID    : chr  "24153" "24157" "24158" "24159" ...
# $ GSE205325_LPS_Chronic: num  0.0745 0.0229 0.0383 0.0257 0.0135 ...
# NULL
```

10.3 Next differential expression results derived from different species (rats vs. mice) need to be aligned (joined) using a gene ortholog database.
We need to read this ortholog information into R.

Note

What are gene orthologs?

- Homology refers to biological features, including genes and their products, that are descended from a feature present in a common ancestor.
- Homologous genes become separated in evolution in two different ways: separation of two populations with the ancestral gene into two species or gene duplication of the ancestral gene within a lineage:
  - Genes separated by speciation are called orthologs.
  - Genes separated by gene duplication events are called paralogs.

This definition came from NCBI:
https://www.nlm.nih.gov/ncbi/workshops/2023-08_BLAST_evol/ortho_para.html

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 April 25, 2024). In preparation for the analysis, the instructor (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 ortholog database can be downloaded here: https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_GemmaDEResults_2024/MouseVsRat_NCBI_Entrez_JacksonLab_20240425.csv

Make sure the file ("MouseVsRat_NCBI_Entrez_JacksonLab_20240425.csv") is placed in your working directory.

The code for preparing the ortholog database for our analyses can be found here: https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_GemmaDEResults_2024/2024_FormattingRatMouseOrthologDatabase.R

Command

Example R Code: Input mouse/rat gene ortholog information into R

```
#We have a ortholog database that we downloaded from Jackson Labs on
April 25, 2024
#The reformatted version can be downloaded from our github site:
#https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_G
emmaDEResults_2024/MouseVsRat_NCBI_Entrez_JacksonLab_20240425.csv
#Make sure it is saved in your working directory


MouseVsRat_NCBI_Entrez<-
read.csv("MouseVsRat_NCBI_Entrez_JacksonLab_20240425.csv",
header=TRUE, stringsAsFactors = FALSE, row.names=1,
colClasses=c("character", "character", "character"))
```

10.4 Join the gene ortholog database to the two data frames containing all of the aligned mouse differential expression results (the Log2FC dataframe and the SV dataframe).

Command

Example R Code: Join gene ortholog information to the mouse DE Results data-frame

```
#We want to join this ortholog database to our mouse results (Log2FC
and SV):

Mouse_MetaAnalysis_FoldChanges_wOrthologs<-
join(MouseVsRat_NCBI_Entrez, Mouse_MetaAnalysis_FoldChanges,
by="Mouse_EntrezGene.ID", type="full")

#Structure of the new object:
str(Mouse_MetaAnalysis_FoldChanges_wOrthologs)
#'data.frame':  25288 obs. of  6 variables:

Mouse_MetaAnalysis_SV_wOrthologs<-join(MouseVsRat_NCBI_Entrez,
Mouse_MetaAnalysis_SV, by="Mouse_EntrezGene.ID", type="full")

#Structure of the new object:
str(Mouse_MetaAnalysis_SV_wOrthologs)
#'data.frame':  25288 obs. of  6 variables:
```

10.5 If there are rat datasets, we then want to join our mouse Log2FC and SV data frames to the rat Log2FC and SV dataframes using the rat/mouse gene ortholog information

Command

Example R code: Join the mouse Log2FC and SV dataframes to the rat Log2FC and SV dataframes

```
#*If there are rat datasets*, we then want to join our mouse Log2FC
and SV results to the rat results using the ortholog information:

MetaAnalysis_FoldChanges<-
join(Mouse_MetaAnalysis_FoldChanges_wOrthologs,
Rat_MetaAnalysis_FoldChanges, by="Rat_EntrezGene.ID", type="full")

#Structure of the new object:
str(MetaAnalysis_FoldChanges)
#'data.frame':  28101 obs. of  7 variables:

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

#Structure of the new object:
str(MetaAnalysis_SV)
#'data.frame':  28101 obs. of  7 variables:
```

10.6 If there aren't any rat datasets, we just rename the mouse Log2FC and SV dataframes so that our downstream code works.

Command

Example R code: Renaming the mouse Log2FC and SV data frames

```
#*If there aren't any rat datasets*, we just rename the dataframes so
that our downstream code works:
MetaAnalysis_FoldChanges<-Mouse_MetaAnalysis_FoldChanges_wOrthologs
str(MetaAnalysis_FoldChanges)

MetaAnalysis_SV<-Mouse_MetaAnalysis_SV_wOrthologs
str(MetaAnalysis_SV)
```

10.7 The current Mouse-Rat Entrez annotation is missing entries for any genes in the datasets that don't have orthologs. Replace it with a fixed version.

Command

Example R Code: Fixing the Mouse-Rat Entrez ID annotation information

```
#For simplicity's sake, I'm going to replace that Mouse-Rat Entrez
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="_")
```

10.8 *Optional:* Compare the effects (Log2FCs) across the different datasets and statistical contrasts using the data from all represented genes.

Three main ways to do this are:
1) *Pairwise*: Compare the Log2FCs from two statistical contrasts. Here, I provide example code for making a scatterplot, but rank-rank hypergeometric overlap (RRHO) plots are also popular.
2) *Correlation Matrix*: Compare the Log2FCs from all statistical contrasts by outputting a matrix containing the correlation coefficients (in this case, Spearman) quantifying the relationship between each possible pair of statistical contrasts.
3) *Clustered Heatmap*: Make a hierarchically clustered heatmap to illustrate which contrasts/datasets have the most similar Log2FCs for all represented genes.

Command

Example R Code: Comparing the effects (Log2FCs) across the different datasets and statistical contrasts using the data from all genes

```
#Example scatter plot comparing the effects (Log2FCs) for all of the
genes represented in two datasets:
plot(MetaAnalysis_FoldChanges$GSE126678_LPS_Acute~MetaAnalysis_FoldCha
nges$GSE181285_LPS_Acute)

#Note - many people prefer to plot these relationships using RRHOs
(Rank rank hypergeometric overlap plots)
#I like using both.
#The code for the RRHOs is a little complicated, but I'm happy to
share if folks are interested.

#Here's code for looking at the correlation of all of our log2FC
results with all of our other log2FC results
#This is called a correlation matrix
#This one uses non-parametric (Spearman) correlation coefficients

write.csv(cor(as.matrix(MetaAnalysis_FoldChanges[,-c(1:3)]),
use="pairwise.complete.obs", method="spearman"),
"CorMatrix_AllLog2FCs.csv")

#An illustration of the correlation matrix using a hierarchically
clustered heatmap:
heatmap(cor(as.matrix(MetaAnalysis_FoldChanges[,-c(1:3)]),
use="pairwise.complete.obs", method="spearman"))
```

Meta-Analysis of Differential Expression Results

11 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_GemmaDEResults_2024/2024_Example_Pipeline_for_MetaAnalysis.R

11.1 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.

11.2 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.4.6-0) available on 06/01/2024. 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

11.3 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 11.5)

11.4 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 11.5)

11.5 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

Example 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){

  #The function first provides information about how many of the
statistical contrasts have NA values as their differential expression
results for each gene:
  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))

  #Then any row (gene) that has too many NAs is removed from the
analysis:
  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 (i.e., the columns that aren't annotation) for
each gene (row):
  for(i in c(1:nrow(MetaAnalysis_FoldChanges_ForMeta))){

    #When pulling out the log2FC values and sampling variances (SV)
for each gene, we use the function as.numeric to make sure they are
in numeric matrix format because this is the required input format
for the meta-analysis function that we will use:
    effect<-as.numeric(MetaAnalysis_FoldChanges_ForMeta[i,-c(1:3)])
    var<-as.numeric(MetaAnalysis_SV_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 everything looks good, we move on to running the meta-
analysis using a model that treats the variation in Log2FC across
studies as random effects:
    if(skip_to_next){}else{
      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)
  }

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

  #The row names for our output are the combined mouse-rat entrez
ids:
  row.names(metaOutput)<-MetaAnalysis_FoldChanges_ForMeta[,3]

  #We return this output back into our global environment
  metaOutput<<-metaOutput
  MetaAnalysis_Annotation<<-MetaAnalysis_FoldChanges_ForMeta[,c(1:3)]
  return(metaOutput)
  return(MetaAnalysis_Annotation)

  #... and provide the user with an update about the newly created
object:

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

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

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

}
```

Command

Example R Function 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 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     4     5
# 13355  3200  5059   277  5293   917

#For this dataset, I'm going to try running a meta-analysis using genes that were found in at least 4 sets of differential expression results
#Since there are 5 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=5
CutOffForNAs=2
#I have 5 statistical contrasts total (comparisons)
```

```
#I have 5 statistical contrasts total (comparisons)
#and 2 NA is too many


#Running a basic meta-analysis:

#This function  is designed to run a basic meta-analysis of Log2FC
and sampling variance values using our previously generated objects
MetaAnalysis_FoldChanges & MetaAnalysis_SV

#Reading in the function:

#You can input this function by running the code discussed above to
create the function in your R environment.

#Alternatively, you can download the script for the function from our
Github site and save the file in your working directory:
#https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_G
emmaDEResults_2024/Function_RunBasicMetaAnalysis.R

#And then source it from your working directory:
source("Function_RunBasicMetaAnalysis.R")

#Example usage:

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...


#Example output:

# [1] "Table of # of NAs per Row (Gene):"
# MetaAnalysis_FoldChanges_NAsPerRow
# 0     1     2     3     4     5
# 13355  3200  5059   277  5293   917
# [1] "MetaAnalysis_FoldChanges_ForMeta:"
# 'data.frame': 16555 obs. of  8 variables:
#   $ Rat_EntrezGene.ID                : chr  "114087" "191569"
"246307" "65041" ...
# $ Mouse_EntrezGene.ID              : chr  "23825" "18585" "66514"
"20480" ...
# $ MouseVsRat_EntrezGene.ID         : chr  "23825_114087"
"18585_191569" "66514_246307" "20480_65041" ...
# $ LPS_SubchronicAndAcute_vs_Vehicle : num  -0.0239 -0.0858 -0.0686
```

```
0.0891 0.0376 ...
```

11.6 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

12 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.

12.1 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 06/01/2024. 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)
```

12.2 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_GemmaDEResults_2024/HOM_MouseVsRat_20240425.csv

Make sure it is saved in your working directory.

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 April 25, 2024) and reformatted for easy use while preparing the rat-mouse ortholog gene database used in protocol step 10.3 above.

The code used to prepare the database for our analyses can be found here: https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_GemmaDEResults_2024/2024_FormattingRatMouseOrthologDatabase.R

Command

Example R code: Inputting additional mouse/rat gene annotation

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

#Reading in a database containing more detailed gene annotation:
HOM_MouseVsRat <- read.csv("HOM_MouseVsRat_20240425.csv", header =
TRUE, row.names = 1)

colnames(HOM_MouseVsRat)
# [1] "DB.Class.Key"
# [2] "Mouse_Common.Organism.Name"
# [3] "Mouse_NCBI.Taxon.ID"
# [4] "Mouse_Symbol"
# [5] "Mouse_EntrezGene.ID"
# [6] "Mouse_Mouse.MGI.ID"
# [7] "Mouse_HGNC.ID"
# [8] "Mouse_OMIM.Gene.ID"
# [9] "Mouse_Genetic.Location"
# [10] "Mouse_Genome.Coordinates..mouse..GRCm39.human..GRCh38."
# [11] "Mouse_Name"
# [12] "Mouse_Synonyms"
# [13] "Rat_Common.Organism.Name"
# [14] "Rat_NCBI.Taxon.ID"
# [15] "Rat_Symbol"
# [16] "Rat_EntrezGene.ID"
# [17] "Rat_Mouse.MGI.ID"
# [18] "Rat_HGNC.ID"
# [19] "Rat_OMIM.Gene.ID"
# [20] "Rat_Genetic.Location"
# [21] "Rat_Genome.Coordinates..mouse..GRCm39.human..GRCh38."
# [22] "Rat_Name"
# [23] "Rat_Synonyms"

#Renaming the columns so that we can easily join the annotation to
our meta-analysis results:
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).

12.3 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

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

```
FalseDiscoveryCorrection<-function(metaOutput, HOM_MouseVsRat,
MetaAnalysis_Annotation){

  #This calculates the false discovery rate, or q-value, for each of
our p-values using the Benjamini-Hochberg procedure:
  tempPvalAdjMeta<-mt.rawp2adjp(metaOutput[,3], proc=c("BH"))

  #Then we put those results back into the order of our orginal
output:
  metaPvalAdj<-tempPvalAdjMeta$adjp[order(tempPvalAdjMeta$index),]

  #And bind the false discovery rate (FDR) to the rest of the meta-
analysis output:
  metaOutputFDR<-cbind(metaOutput, metaPvalAdj[,2])

  #And name that column FDR:
  colnames(metaOutputFDR)[7]<-"FDR"

  #These results are returned to our global environment:
  metaOutputFDR<<-metaOutputFDR

  #We let the user know the basic structure of the meta-analysis
output with FDR added to it (just to make sure everything still looks
good)
  print("metaOutputFDR:")
  print(str(metaOutputFDR))

  #Then we make a dataframe that adds the annotation to that output:
  TempDF<-cbind.data.frame(metaOutputFDR, MetaAnalysis_Annotation)
  #And then adds even more detailed gene annotation:

  #First the detailed annotation for the mouse genes:
  TempDF2<-join(TempDF, HOM_MouseVsRat[,c(4:5,9:11)],
by="Mouse_EntrezGene.ID", type="left", match="first")

  #Next the annnotation for the rat genes:
  TempDF3<-join(TempDF2, HOM_MouseVsRat[,c(15:16,20:22)],
by="Rat_EntrezGene.ID", type="left", match="first")

  #This is renamed and returned to our global environment:
```

```
  metaOutputFDR_annotated<-TempDF3
  metaOutputFDR_annotated<<-metaOutputFDR_annotated

  #And written out into our working directory:
  write.csv(metaOutputFDR_annotated, "metaOutputFDR_annotated.csv")

  #Then we make a version of the output in order by p-value:
  metaOutputFDR_OrderbyPval<<-
metaOutputFDR_annotated[order(metaOutputFDR_annotated[,5]),]

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

  #And give the user some information about their results:

  print("Do we have any genes that are statistically significant
following loose false discovery rate correction (FDR<0.10)?")
  print(sum(metaOutputFDR_annotated[,9]<0.10, na.rm=TRUE))

  print("Do we have any genes that are statistically significant
following traditional false discovery rate correction (FDR<0.05)?")
  print(sum(metaOutputFDR_annotated[,9]<0.05, na.rm=TRUE))

  print("What are the top results?")

print(head(metaOutputFDR_annotated[order(metaOutputFDR_annotated[,5]),
]))

  rm(tempPvalAdjMeta, metaPvalAdj)

}
```

Command

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

```
#This code runs a function that corrects the meta-analysis output to
take into account the fact that we are running the statistical
calculations thousands of times and therefore have a heightened risk
of false discovery (false discovery rate correction)

#Reading in the function:

#You can input this function by running the code discussed above to
create the function in your R environment.

#Alternatively, you can download the script for the function from our
Github site and save the file in your working directory:
#https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_G
emmaDEResults_2024/Function_FalseDiscoveryCorrection.R

#And then source it from your working directory:
source("Function_FalseDiscoveryCorrection.R")

#Example usage:

FalseDiscoveryCorrection(metaOutput, HOM_MouseVsRat,
MetaAnalysis_Annotation)

#Example output:

# [1] "metaOutputFDR:"
# num [1:16555, 1:7] 0.0234 0.1315 -0.023 0.0437 0.0769 ...
# - attr(*, "dimnames")=List of 2
# ..$ : chr [1:16555] "23825_114087" "18585_191569" "66514_246307"
"20480_65041" ...
# ..$ : chr [1:7] "Log2FC_estimate" "SE" "pval" "CI_lb" ...
# NULL
# [1] "Do we have any genes that are statistically significant
following loose false discovery rate correction (FDR<0.10)?"
# [1] 253
# [1] "Do we have any genes that are statistically significant
following traditional false discovery rate correction (FDR<0.05)?"
# [1] 136
# [1] "What are the top results?"
```

```
# [1] "What are the top results?"
# Rat_EntrezGene.ID Mouse_EntrezGene.ID Log2FC_estimate         SE
     pval      CI_lb       CI_ub
# 13170_24309               24309               13170      -0.4293013
0.07014204 9.330764e-10 -0.5667771 -0.29182542
# 74772_362645              362645               74772      -0.1313814
0.02330647 1.729185e-08 -0.1770612 -0.08570151
# 192188_282580             282580              192188       0.3562365
0.06392332 2.505735e-08  0.2309491  0.48152389
# 54006_83632                83632               54006      -0.1635032
0.02996583 4.861038e-08 -0.2222351 -0.10477121
# 216565_305556             305556              216565       0.1764604
0.03363489 1.551430e-07  0.1105372  0.24238352
# 268445_360575             360575              268445      -0.1584491
0.03081248 2.712959e-07 -0.2188405 -0.09805779
# Number_Of_Comparisons          FDR MouseVsRat_EntrezGene.ID
Mouse_Symbol Mouse_Genetic.Location
# 13170_24309                       5 1.544708e-05
13170_24309          Dbp               Chr7  cM
# 74772_362645                      5 1.382748e-04
74772_362645      Atp13a2               Chr4  cM
# 192188_282580                     5 1.382748e-04
192188_282580        Stab2              Chr10  cM
# 54006_83632                       5 2.011862e-04
54006_83632         Deaf1               Chr7  cM
# 216565_305556                     5 5.136784e-04
216565_305556        Ehbp1              Chr11  cM
# 268445_360575                     5 7.485505e-04
268445_360575     Ankrd13b              Chr11  cM
# Mouse_Genome.Coordinates..mouse..GRCm39.human..GRCh38.
               Mouse_Name Rat_Symbol
# 13170_24309                                  Chr7:45354658-
45359579(+) D site albumin promoter binding protein        Dbp
# 74772_362645                               Chr4:140714184-
140734641(+)                        ATPase type 13A2    Atp13a2
# 192188_282580                               Chr10:86677062-
86843889(-)                              stabilin 2      Stab2
# 54006_83632                                Chr7:140877093-
140907603(-)             DEAF1, transcription factor      Deaf1
# 216565_305556                              Chr11:21955825-
22237086(-)             EH domain binding protein 1      Ehbp1
# 268445_360575                              Chr11:77361311-
77380504(-)               ankyrin repeat domain 13b   Ankrd13b
# Rat_Genetic.Location
Rat_Genome.Coordinates..mouse..GRCm39.human..GRCh38.
                 Rat_Name
# 13170_24309               Chr1 q22
```

```
                    NA D-box binding PAR bZIP transcription factor
# 74772_362645                Chr5_g36
```

12.4 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%).

```
                    NA                  DEAF1 transcription factor
```

Exploring Meta-Analysis Results

13 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".

13.1 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.

13.2 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

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

```
MakeForestPlots<-function(metaOutputFDR_annotated,
EntrezIDAsCharacter, species){

  #I originally wrote this function using only mouse Entrez IDs as
input
  #but then I realized that the function didn't work for genes that
were only found in rats
  #so now the function allows either rat or mouse Entrez ids as
input, and includes a conditional (if/else) statement

  if(species=="Mouse"){

    #This grabs the mouse gene symbol for the EntrezID from our meta-
analysis annotation:
    MouseGeneSymbol<-
metaOutputFDR_annotated$Mouse_Symbol[which(metaOutputFDR_annotated$Mou
se_EntrezGene.ID==EntrezIDAsCharacter)]
    #This grabs the rat gene symbol for the EntrezID from our meta-
analysis annotation:
    RatGeneSymbol<-
metaOutputFDR_annotated$Rat_Symbol[which(metaOutputFDR_annotated$Mouse
_EntrezGene.ID==EntrezIDAsCharacter)]

    #This grabs the Log2FC values for the EntrezID
    effect<-
as.numeric(MetaAnalysis_FoldChanges_ForMeta[which(MetaAnalysis_FoldCha
nges_ForMeta$Mouse_EntrezGene.ID==EntrezIDAsCharacter),-c(1:3)])

    #This grabs the sampling variance (SV) values for the EntrezID
    var<-
as.numeric(MetaAnalysis_SV_ForMeta[which(MetaAnalysis_FoldChanges_ForM
eta$Mouse_EntrezGene.ID==EntrezIDAsCharacter),-c(1:3)])

  }else if(species=="Rat"){

    #This set of code does all of the same processes as above, but
interpreting the EntrezID as a Rat Entrez ID:

    RatGeneSymbol<-
```

```
    RatGeneSymbol<-
metaOutputFDR_annotated$Rat_Symbol[which(metaOutputFDR_annotated$Rat_E
ntrezGene.ID==EntrezIDAsCharacter)]

    MouseGeneSymbol<-
metaOutputFDR_annotated$Mouse_Symbol[which(metaOutputFDR_annotated$Rat
_EntrezGene.ID==EntrezIDAsCharacter)]

    effect<-
as.numeric(MetaAnalysis_FoldChanges_ForMeta[which(MetaAnalysis_FoldCha
nges_ForMeta$Rat_EntrezGene.ID==EntrezIDAsCharacter),-c(1:3)])

    var<-
as.numeric(MetaAnalysis_SV_ForMeta[which(MetaAnalysis_FoldChanges_ForM
eta$Rat_EntrezGene.ID==EntrezIDAsCharacter),-c(1:3)])

  }else{

    print("Please use either 'Mouse' or 'Rat' to indicate whether you
are using annotation for mouse or rat genes")

  }

#This code makes the Forest Plot

  #First it opens up a .pdf file to output the plot into:
  #It automatically names that file with the mouse and rat ene symbols
  pdf(paste("ForestPlot_Mouse", MouseGeneSymbol,"Rat",
RatGeneSymbol,".pdf", sep="_"), height=5, width=8)

  #This code makes the forest plot:
  #Note that the x-axis limits are currently set to -3 to 3
  #This may be too big or too small for visualizing the results for
some genes.
  forest.rma(rma(effect, var),
slab=colnames(MetaAnalysis_FoldChanges_ForMeta)[-c(1:3)],  xlim=c(-3,
3))

  #This code labels the forest plot with the mouse and rat gene
symbols:
  mtext(paste("Mouse", MouseGeneSymbol, "Rat", RatGeneSymbol,
sep="_"), line=-1.5, cex=2)

  #This closes the connection to the .pdf file, finishing the plot
  dev.off()

}
```

Command

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

```
#Reading in the function:

#You can input this function by running the code discussed above to
create the function in your R environment.

#Alternatively, you can download the script for the function from our
Github site and save the file in your working directory:
#https://github.com/hagenaue/BrainDataAlchemy/blob/main/MetaAnalysis_G
emmaDEResults_2024/Function_MakeForestPlots.R

#And then source it from your working directory:
source("Function_MakeForestPlots.R")

#Example Usage:

#Note - this function currently uses Entrez ID (NCBI ID) as it's input
#It needs to be formatted as a character (not an integer) to work
#It can accept either mouse annotation (Entrez ID) or rat annotation
(Entrez ID) as its input

#Making a forest plot for gene Dbp (Mouse Entrez ID 13170):

MakeForestPlots(metaOutputFDR_annotated, EntrezIDAsCharacter="13170",
species="Mouse")

#Making a forest plot for gene Atp13a2 (Mouse Entrez ID 74772):

MakeForestPlots(metaOutputFDR_annotated, EntrezIDAsCharacter="74772",
species="Mouse")

#Making a forest plot for gene Stab2 (Rat Entrez ID 282580):

MakeForestPlots(metaOutputFDR_annotated,
EntrezIDAsCharacter="282580", species="Rat")
```

Note

Note: The way that this function is currently written, I suspect it might potentially throw up an error message if there is more than one set of Log2FC associated with an Entrez ID. This would happen if an Entrez ID in one species (e.g., mouse) mapped to more than one Entrez ID in the other species (e.g., rat) at the point that the results from the two species was joined.
It should be solvable by just using the EntrezID for the other species (e.g., rat) as the input to the function.

13.3 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

13.4 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/

13.5 *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

14 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.

14.1 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".

14.2 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)

Note

These files should include:
- Your Gemma Search Code
- Any code used to download experimental design information about your potential datasets
- Any code used to examine the distribution of Log2 expression data for your datasets
- Your code for importing in the differential expression results for your datasets from Gemma,
- Your code for aligning your different datasets by gene id
- Your code for running the meta-analysis and outputting the results
- All final R workspaces

14.3 To make your analyses reproducible, make sure that you preserve 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.

Note

- These files were outputted by: Function_SavingGemmaDEResults_forEachResultSet.R
- Each of these files is named DEResults followed by the GSE# and ResultSet ID #

14.4 In addition to your PRISMA search diagram and summary table of final selected datasets, make sure that you preserve the files containing your intermediate notes documenting the process of triaging datasets and result sets.

Note

These files should include:
- MyDatasets_Screened.csv
- ResultSets_toScreen.csv
- ResultSets_Screened.csv

14.5 In addition to your results summary and example forest plots, 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.

Note

The files containing the full meta-analysis results were outputted as:
- metaOutputFDR_annotated.csv
- metaOutputFDR_orderedByPval.csv

14.6 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 11.2

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

Step 12.1

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 13.4

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 13.4

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 13.4

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 13.4

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 13.5

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 13.5

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 4.3

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

Step 7

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