Differential Expression and the Mouse Gut-Brain Axis

## Introduction

This module provides data for an independent project exploring differential expression in RNA-seq data.

This activity will walk through differential expression analysis and use bioinformatics tools (R) to understand how gut bacteria (the *gut microbiome*) can influence the expression of genes in the brain (the gut-brain axis). You will work with real data from a mouse RNA-seq experiment.

The basic steps of your analysis are:

1. Identify mouse genes of interest for your project.
2. Characterize the gene expression of your genes of interest in the dataset.
3. Explore differential gene expression in the control vs experimental group.

### The original study

The original study, *Human Gut Microbiota from Autism Spectrum Disorder Induces Behavioral Deficits in Mice*, was published in 2019 [1]. Gut microbiota are known to be different between individuals with ASD and individuals who are considered typically-developing. Additionally, some individuals with ASD also experience gastrointestinal symptoms, and their gut microbiota show the greatest difference when compared to the gut microbiota of typically-developing individuals. Some researchers have proposed that gut bacteria can influence some of the symptoms of ASD. The relationship between the intestinal microbiome and the development and function of the human brain is known as the *gut-brain axis.*

In this study, researchers explored whether they could induce ASD-like behaviors in mice by changing their gut microbiome. Mice in this experiment received fecal transplants. Some mice received transplants from humans who have been diagnosed with ASD, while other mice received transplants from humans who did not have any diagnosis (control). This allowed researchers to control the composition of the gut microbiome in each mouse. They discovered that colonization with gut microbiota was enough to induce ASD-like behaviors in the mice. They also let the mice breed and collected gene expression data from the brains of their offspring to explore whether changing the gut microbiota could result in changed gene expression. In particular, they discovered that the offspring of mice who received stool from ASD donors showed different gene splicing and expression profiles of certain ASD-relevant genes.

|  |
| --- |
| Note |
| It is important to note that researchers are **not** suggesting that ASD is entirely induced by gut bacteria. There is a strong genetic component to ASD. Scientists have known for years that there are **both genetic and environmental components** to the development or severity of **some** ASD symptoms. This research explores one possible environmental component. |

You can read the original research paper [here](https://www.cell.com/cell/fulltext/S0092-8674(19)30502-1).

### The Gut-Brain Axis

The *gut-brain axis* is the term used for the proposed connection between the gut microbiome and the brain. In recent years, studies have identified links between neurological symptoms and the presence of intestinal inflammation. There are around 100 million neurons in the human gut, and around 70% of the serotonin (an important neurotransmitter) in the human body is made in the gut. Communication between the microbes in the gut and the brain happens through multiple routes, including immune signaling (signals sent to the brain by immune cells), stimulation of the vagus nerve, and activation of the enteric nervous system (a system of neurons in the intestinal that are often called the “second brain”) through formation of microbial-derived metabolites like tryptophan, short-chain fatty acids, or secondary bile acids.



In the figure above, the “up” arrow indicates an increase; this could be increased production of a protein, increased activity of a biological pathway, or a general increase in the presence of a structure. The “down” arrow indicates a decrease.

Researchers have identified multiple ways the intestinal microbiota in the gut might remodel the structure of the nervous system. This includes changes in myelination, neurogenesis, neurotransmitters and synaptic plasticity, growth of dendrites, activation of microglia, and changes in the permeability of the blood-brain-barrier. The exact mechanism for these changes is still unknown.

|  |
| --- |
| Note |
| Why are the neural processes mentioned above important?  **myelination**: the process of forming protein sheaths (called myelin) around the long axons of nerves. This insulates the axon and improves the speed and efficiency of transmission of neural signals.  **neurogenesis**: the formation of new neurons, particularly in the adult central nervous system. This is important for memory and learning.`  **synaptic plasticity**: the process of strengthening or weakening connections at the synapse of nerves (where the signal is transferred from one nerve to another). This sort of adaptive change is fundamental for memory and learning, but also helps with recovery after injury.  **dendritic growth**: the process by which neurons form new branches (dendrites) and spines (small protrusions on the dendrites). This is a key aspect of synaptic plasticity and is essential for brain development, learning, and memory formation.  **microglia**: immune cells found in the central nervous system. They support central nervous system development and synaptogenesis (formation of new synapses), maintain homeostasis, contribute to an immune response against infectious agents, and are associated with adult neurogenesis, neuroinflammation, degenerative diseases, stroke, trauma and regeneration.  **blood-brain-barrier**: a highly selective semi-permeable barrier located in the central nervous system that separates the bloodstream from the fluid that surrounds the brain. It maintains a stable environment for neuronal function and regulates passage of molecules between the brain and the rest of the body. |

Remodeling of the nervous system via the gut microbiome is suspected to be responsible for the severity of some symptoms associated with neurodevelopmental and neurodegenerative disorders. A possible link between the gut microbiome and increased symptom severity is actively being studied for Major Depressive Disorder, Autism Spectrum Disorder, fibromyalgia, migraines, Parkinson’s disease, Alzheimer’s disease, and more.

The samples in this dataset are from mice that *do not* carry genes associated with either neurodevelopmental or neurodegenerative diseases. Instead, any symptoms or changes in the brain they show are the result of microbes introduced to their gut via fecal transplant. **This dataset gives you the opportunity to study possible effects of the gut-brain axis firsthand.**

### Deciding on your research question

The data for this activity includes gene expression data from two different brain regions (striatum and prefrontal cortex) in mice. Additionally, there are two different types of mice. Some of the mice are an Autism Spectrum Disorder-type (see below for details), while others are control-type.

Before you start working with the dataset, you will need to make some decisions about the questions you want to answer.

**Are you interested in exploring gene expression differences between two regions of the mouse brain, without focusing on the gut-brain axis? Expand this section to read more.**

|  |
| --- |
| Brain Structure: The Striatum and the Prefrontal Cortex |
| This dataset includes gene expression data from two different regions in the brain: the striatum and the prefrontal cortex.    The *striatum* is part of the brain involved in motor control and cognitive tasks like reward processing, decision-making, and social interactions (often called executive functions). It lies deep within the center of the brain and is composed of both gray matter (which can be thought of as the “processing” part of brain tissue) and white matter (which is the brain structure involved in transporting messages); the combination of the gray and white matter give this region of the brain a striped appearance, resulting in the name “striatum”. The striatum is involved in both reflexive movement - that is, involuntary movement that happens as an immediate response to a stimulus - and slower, planned movement like walking. In Parkinson’s disease, some patients experience degeneration of parts of the striatum, resulting in spastic, uncontrollable movement.    The *prefrontal cortex* is the part of the brain that is primarily in charge of decision making, reasoning, personality, maintaining social appropriateness, and other complex behaviors that fall under the umbrella of executive functions. This can include planning, self-control, and working towards long-term goals. The prefrontal cortex is located in the very front of the brain, just behind your forehead. One of the most famous brain injury patients was Phineas Gage, a railroad worker who survived an iron rod through his forehead. His prefrontal cortex was destroyed in this accident, and doctors noted huge behavioral and personality changes. You can read more about his case [here](https://pmc.ncbi.nlm.nih.gov/articles/PMC1114479/). |

**Are you interested in exploring neurodevelopmental disorders?** These are disorders that are the result of altered brain development. Although this dataset was created using “ASD-type” mice, many neurological disorders have shared symptoms and probably share a biological basis. **Expand this section to read more.**

|  |
| --- |
| Brain Disorders: Neurodevelopmental Disorders |
| Although the following disorders can be diagnosed at any age, they are considered neurodevelopmental disorders because symptoms usually show up within the first two years of life. Many of these disorders have overlapping symptoms, including difficulty managing emotions, experiencing intense emotional outbursts, a dysregulated stress response system, difficulty with executive functions (like attention regulation, organization, and planning), and issues with social communication and unspoken social norms. Many of these disorders have similar biological roots, such as neuroinflammation, genetic variation, and disruptions in systems that rely on dopamine, norepinephrine, and serotonin (which can cause impaired connections between regions of the brain, called white matter connectivity). People with these disorders also show differences in brain regions like the prefrontal cortex and the amygdala compared to people without neurodevelopmental disorders. For example, malfunctioning of the hypothalamic-pituitary-adrenal (HPA) axis is common in neurodevelopmental disorders. The HPA axis regulates the brain and body’s response to stress. Many people with neurodevelopmental disorders show an unusually strong reaction to stress and can have difficulty recovering from it.    This is not a comprehensive list of all neurodevelopmental disorders - you may know of others that are not included! Any of these could be a good starting point for your research question.  *Autism spectrum disorder (ASD)* is a neurological disorder that affects behavioral and social interactions, among other things. Individuals diagnosed with ASD can experience a wide range of symptoms, including differences in social behaviors and communication styles, as well as intellectual disabilities and physical issues like sensory sensitivities or gastrointestinal problems. Some genes that have been suggested as ASD risk genes include SHANK3, NRXN1, CHD8, ARIH2, ZDHHC15, and TRPC6. Individuals with ASD also show changes in both white and gray matter volume, as well as an imbalance in connectivity of neural circuits (there are more connections and stronger connections than typical within local regions of the brain, and both fewer and weaker connections between distant regions of the brain). The neural circuits that are imbalanced connect brain regions like the prefrontal cortex and the amygdala.    **Attention-Deficit/Hyperactivity Disorder (ADHD)** is characterized by difficulties with inattention, hyperactivity, and impulsivity. Individuals may show predominantly inattention symptoms (predominantly inattentive ADHD), predominantly hyperactivity and impulsivity symptoms (predominantly hyperactive ADHD), or a combination of all (combined type ADHD). Some proposed ADHD risk genes are SKI, ZNF544, ST3GAL3, and PEX2. Additionally, researchers believe alterations to the prefrontal cortex and the dopaminergic system are major contributing factors for ADHD.    **Learning Disorders (LDs)** include disorders such as dyslexia, dyscalculia, and dysgraphia. Each of these is characterized by difficulty developing a fundamental academic skill, including reading (dyslexia), math and working with numbers (dyscalculia), writing (dysgraphia), listening (auditory processing disorder), speaking (language processing disorder), or non-verbal reasoning (nonverbal learning disorder). Individuals with LDs frequently show differences neural connectivity within brain regions like the prefrontal cortex, hippocampus, and parietal lobes, which may contribute to decreased brain processing speed and executive functioning. Research also suggests that altered tryptophan metabolism is associated with LDs.    **Mood Disorders** are a group of disorders in which affected individuals exhibit emotional state changes, which can include episodes of deep sadness (dysphoria), lack of feeling anything (anhedonia), and extreme happiness or euphoria (hypomania or mania). The most well-known of these disorders are Major Depressive Disorder (MDD) and Bipolar Disorder (BD). Some individuals with Schizoaffective Disorder or Schizophrenia (SCZ) also experience hallucinations and psychosis. Variation in serotonin transporter genes and dopamine receptor genes have been associated with Mood Disorders, as well as neurotransmitter alterations and brain circuit dysfunction, particularly within the prefrontal cortex and the limbic system (where the amygdala is).  Mood disorders like depression, bipolar, and schizophrenia are not currently listed as neurodevelopmental disorders in the American Psychiatric Association’s Diagnostic and Statistical Manual of Mental Disorders, Fifth Edition (DSM-5). This is the official handbook of psychiatric disorders used for diagnosis in the field. However, researchers have suggested that mood disorders should be considered neurodevelopmental disorders due to growing evidence that neurodevelopmental pathways play a role in their development and manifestation. For the purposes of this module, we have included them in the “Neurodevelopmental Disorder” section.    If you’d like to read more about the biology of neurodevelopmental disorders, (Bertollo et al. 2025)[https://www.mdpi.com/2076-3425/15/3/307] is a great starting point! |

**Are you interested in exploring neurodegenerative disorders?** These are disorders that that occur when brain structures and systems experience degradation and may be the result of similar biological processes that cause neurodevelopmental disorders. **Expand this section to read more.**

|  |
| --- |
| Brain Disorders: Neurodegenerative Disorders |
| Neurodegenerative disorders are usually considered diseases of aging because a person’s risk of developing one of these disorders increases as they age. These disorders are the result of the breakdown and eventual loss of neurons and synaptic connections. An individual’s specific symptoms and diagnosis are the result of where in the brain or central nervous system the breakdown of neurons and synapses occur. Although the symptoms and progression may differ between diseases, the biological basis is generally one of seven things: a genetic mutation, accumulation of misfolded proteins, excitotoxicity (overstimulation of glutamate receptors, ultimately causing cell death), loss of regulatory function of noncoding RNAs (causes biological processes to break down), oxidative stress, mitochondrial dysfunction, or neuroinflammation (when the immune system attacks and damages the neurons).    This is not a comprehensive list of all neurodegenerative disorders - you may know of others that are not included! Any of these could be a good starting point for your research question. Remember that this is a *gut-brain axis* dataset, so you should verify that disorders that are not on this list do have an association with changes in the gut microbiome first.  **Alzheimer’s Disease (AD)** is the leading cause of dementia in seniors and is marked by progressive decline in memory, cognition, and behavior. It generally begins as a result of neuron loss in the hippocampus, or the region of the brain responsible for episodic memory. Beta-amyloid plaques and clumps of tau proteins are deposited on nerve cells, which prevents normal nerve impulses from traveling between neurons. This eventually leads to cell death and deterioration of brain tissue. Tau proteins are also present in normal brain tissue to stabilize microtubules and aid transport within neurons. They cause problems when they become damaged and detach from microtubules to form tangles, or clumps.    **Parkinson’s Disease (PD)** is a movement disorder of the nervous system. Over time, individuals with PD lose their ability to regulate and control fine motor movement, resulting in tremors, rigid muscles, slow movement (bradykinesia), and difficulties with balance and coordination. They can also show cognitive decline. These symptoms are the result of gradual deterioration of dopamine-producing neurons, specifically in an area of the brain called the substantia nigra. The substantia nigra is responsible for motor movement and has two parts - the pars compacta, which produces dopamine, and the pars reticulata, which uses GABA (another type of neurotransmitter).    **Multiple Sclerosis (MS)** is an immune-mediated neurodegenerative disorder. In MS, the immune system attacks the protective myelin covering around neurons in both the brain and the spinal cord, causing inflammation, demyelination (of loss of the protective myelin sheath), and damage to the underlying neuron. Eventually, these neurons become less efficient at passing electrical impulses to connecting neurons. Individuals with MS experience fatigue, numbness or tingling, difficulties with coordination, blurred or double vision, and memory and concentration issues.    **Amyotrophic Lateral Sclerosis (ALS)**, also known as Lou Gehrig’s Disease, is caused by progressive degeneration of motor neurons in the brain and spinal cord. The exact mechanisms that trigger this degeneration are not known, but research has suggested that oxidative stress and mitochondrial dysfunction might contribute to the development of ALS. Additionally, the genes C9orf72, SOD1, TARDBP, and FUS have been identified as risk factors for developing an inherited form of the disease.    If you’re interested in reading more about the biology of neurodegenerative disorders, [Gadhave et al. 2021](https://www.sciencedirect.com/science/article/pii/S1568163724001752) is a good place to start! |

### Refining your research question

Once you’ve picked brain structure, a neurodevelopmental disorder, or a neurodegenerative disorder to focus on, you’ll want to start identifying specific genes for your research project. The following questions could be a good starting point.

1. Is there a gene or set of genes linked to a biological process you’re particularly interested in? This could include things like “cytokines”, “anti-inflammatory genes”, “genes involved in building neurons”, or any biological process you find interesting!
2. Are there a known risk genes for your disease or disorder of interest (or genes that are protective against this disorder)? Remember, you don’t have to limit yourself to Autism Spectrum Disorder, either; there are many disorders that have a shared genetic and biological basis! If you decide to start here, check that researchers have linked the symptoms of your disease or disorder of interest to the gut microbiome.
3. Are you curious about the standard “housekeeping” genes that are broadly expressed across cell types and involved in common biological processes, like DNA replication, metabolism, and cell cycle regulation? These processes are vitally important for proper cellular function, so disruption in them can cause problems for the organism.

Once you’ve picked a disorder or cellular process that interests you, start looking up research papers to find genes! You can use sites like PubMed and Google Scholar to look up studies. Skim through at least 4-5 different papers that contain some type of gene name or a biological process. **DO NOT GET TOO LOST INTO THE PAPERS!** Make sure you skim through them for important information, but be willing to set a paper aside if you’re struggling to find what you need.

## Identifying genes of interest with the Mouse Genome Informatics database

When you started developing your research question, you likely identified a human diseases or human genes that you’re interested in. However, the dataset you will be using contains gene expression data for *mouse* samples. Before you can actually begin the data analysis, you’ll need to figure out the mouse gene orthologs for the human genes you identified.

(If you haven’t yet identified human genes of interest, don’t worry - you can also start with a human disease for this part.)

The [Mouse Genome Informatics](https://www.informatics.jax.org/genes.shtml) database that tracks mouse genes and expression data. A full introduction to everything available through the MGI can be found [here](https://www.informatics.jax.org/mgihome/other/homepage_usingMGI.shtml). We’ll reproduce some of it below.

### The mouse as a model organism

The mouse is the most commonly-used model organism in laboratory work. In fact, mice and rats make up 95% of the lab animal population, and more than 80% of the research that has been awarded the Nobel Prize for Medicine was done at least in part with mouse models [2], [3].



So what makes mice such good model organisms for biomedical research? Well, first, they’re economical and relatively easy to keep. Since mice are small, they don’t require a huge amount of space or food. They also have fast reproductive cycles, so researchers can study multiple generations within only a few years. Most importantly, though, mice and humans are both mammals and have about 85% of their protein-coding genome in common. As a result, mouse physiology is quite similar to human physiology. The mouse circulatory, reproductive, digestive, hormonal, and nervous systems are frequently used as models to study how humans grow, age, and develop chronic diseases. They are particularly important model organisms for cancer research and neuroscience.

You can find additional information about how the mouse is used in research [here!](https://www.google.com/url?q=https://www.informatics.jax.org/mgihome/other/homepage_IntroMouse.shtml)

### Mouse gene IDs

In this dataset, genes are identified using their Ensembl Gene ID code. Every gene has an ID that looks something like this:

ENSMUSG00000000001

**ENS** stands for “Ensembl”. Ensembl is a [genome database project](https://ensemblgenomes.org/) managed by the European Bioinformatics Institute. It’s one of several databases like this. Others include NIH’s National Center for Biotechnology Information (NCBI; the organization that manages GenBank and PubMed) and the University of California, Santa Cruz (UCSC) Genome Browser. When a gene code starts with “ENS”, it means you should look up the gene code in the Ensembl database.

**MUS** stands for “Mus”, which is the genus for the mouse.

**G** stands for “Gene”. When you see “G” in the ID name, you know you are working with a gene. There are also codes for transcripts (“T”), exons (“E”), and proteins (“P”).

**00000000001** is the numerical code associated with the gene.

### Picking your genes

The MGI allows you to look up information about genes you’re interested in but don’t know much about. This is particularly helpful if you are interested in genes that were identified in other research organisms or in previous studies. You could also start by looking up a human disease and associated human risk genes, then identify the homologous mouse genes.

Make sure you are recording the name and gene ID for the mouse! Gene names are often similar between mice and humans, but the gene IDs will be very different. If you don’t use the correct gene ID when exploring this dataset, you won’t get very far.

There are multiple ways to use MGI to find information about your mouse genes. **Expand each box for detailed instructions.**

|  |
| --- |
| Looking up genes based on human gene name |
| You may have identified a human gene that you’re interested in for your project, so you need to figure out what the mouse ortholog is and (more importantly) what the mouse Ensembl ID is. Luckily, the MGI allows you to search based on the human gene name.  Let’s say you’re interested in the human gene REG3A, pancreatic secretory protein that may be involved in cell proliferation or differentiation. We know from research papers that it also is upregulated as a result of pancreatic inflammation.  Open the [Mouse Genome Informatics](https://www.informatics.jax.org/genes.shtml) website. To look up information on this human gene, type “REG3A” into the “Quick Search” bar and press enter.    You’ll notice that MGI returns 3 possible mouse genes that are orthologs to the human gene REG3A. This is totally normal! Remember, orthologs are gene that shares the same ancestry. In this case, there was likely a gene duplication event in the mouse ancestor after it split off from the human ancestor.  You can pick whichever ortholog you like (or even pick all of them). For this example, we’ll choose the mouse gene “Reg3a”. Click on the gene symbol (in this example, Reg3a) to get more detailed information.      On the new page that you open, details about the gene are organized into familiar categories. But first, let’s take a look at the IDs on the right-hand side of the “summary” section. In particular, we’re interested in the NCBI link. Click this link.    We’re now on the NCBI page for the mouse gene Reg3a, which has the Ensembl ID for this gene. Copy this so you can later use it to look through the dataset!    Return back to the MGI page on the Reg3a gene. Down the left-hand side of the page, you will see sections about the chromosomal location, homology, gene ontology, expression data, and more. Most sections are expanded by default, but you’ll need to expand the “homology” section yourself.    Once this section is expanded, you can find information about possible human homologs to the mouse gene, including alternate names and where the human homolog is located in the human genome. Check this section to make sure the human gene symbol (name) you originally looked up is listed - it’s always good to verify you haven’t ended up on the page for the wrong mouse gene!    If you continue scrolling down the page, you can also examine the pathways and processes the gene product is involved in under the “gene ontology” section. Clicking on the blue squares takes you to a page with more information about how that particular gene was assigned to a pathway or molecular process.    In the gene ontology section, check whether the associated GO terms make sense for the question you want to explore. For example, maybe we are looking at the REG3A gene because we’re interested in neuroinflammation. We can see that the term “immune system process” (under Biological Process) is associated with the REG3A gene. That tells us that we can indeed use this gene to explore neuroinflammation.  Directly underneath the “ontology” section is information about when the gene is expressed during development. You can learn more information by clicking on the blue squares, or by clicking on the links in the upper right-hand corner. These links will take you to other websites.    Remember, the dataset you’re working with includes samples from the mouse brain. It’s important that you choose genes that are actually expressed in the nervous system. It looks like REG3A isn’t known for its expression in the nervous system, which is a bit troubling. **However, this doesn’t mean it *isn’t* expressed in the brain.** We’ll keep this gene on our gene list for now and check that it’s in our actual dataset. Just be aware that we may have to drop this gene from our analysis later. |

|  |
| --- |
| Looking up genes associated with a human disease |
| You may not know a particular gene that you want to explore in this dataset, but maybe you do have a human disease or condition that you’re interested in. You can look up mouse genes that might be associated with this disease in the MGI as your starting point.  Open the [Mouse Genome Informatics](https://www.informatics.jax.org/genes.shtml) website. Choose the tab that says “Human-Mouse: Disease Connection”.    You’ll then want to choose “Disease or Phenotype Name” from the drop-down menu on the left and type the disease of interest into the search bar on the right. For this example, let’s look up “pancreatic inflammation”.    After a brief pause (in which you may see graphics of a human and mouse running on wheels), a new page will open that lists human genes associated with pancreatic inflammation, as well as the mouse orthologs for those genes. This graphic will also indicate the molecular function these genes are associated with. You can click on any of the mouse genes to explore further. Let’s click on the top hit, Abcb1a.    You should now be on a page that contains all sorts of information about the mouse gene Abcb1a. In the “summary” section up top, you can click on the NCBI ID to get the Ensembl mouse ID for this gene.    Specifically, you will find the Ensemble ID for the mouse gene Abcb1a under the “See Related” section. Copy this so you can later use it to look through the dataset!    Return back to the MGI page on the Abcb1a gene. Down the left-hand side of the page, you will see sections about the chromosomal location, homology, gene ontology, expression data, and more. Most sections are expanded by default, but you’ll need to expand the “homology” section yourself.    Once this section is expanded, you can find information about possible human homologs to the mouse gene, including alternate names and where the human homolog is located in the human genome. Keep a note of what the possible human homolog gene symbols (names) are - this will come in handy for additional research!    If you continue scrolling down the page, you can also examine the pathways and processes the gene product is involved in under the “gene ontology” section. Clicking on the blue squares takes you to a page with more information about how that particular gene was assigned to a pathway or molecular process.    Interestingly, this particular gene doesn’t seem to be associate with the Biological Process GO term “immune system process.” This might seem kind of strange since we found it by looking for genes associated with inflammation. However, all is not lost. This gene *is* associated with GO terms like “signaling” and “stimulus response”. We know that one possible mechanism for how the gut-brain axis works is through altered cellular signaling. So, this gene still seems promising.  Directly underneath the “ontology” section is information about when the gene is expressed during development. You can learn more information by clicking on the blue squares, or by clicking on the links in the upper right-hand corner. These links will take you to other websites.    Remember, the dataset you’re working with includes samples from the mouse brain. It’s important that you choose genes that are actually expressed in the nervous system. Luckily, the Abcb1a gene fits.  Don’t panic if one of your genes isn’t known to be expressed in the brain according to MGI! **This doesn’t mean it *isn’t* expressed in the brain.** Keep the gene on your gene list for now. You can check that it’s in the actual dataset during the first steps of your analysis. Just be aware that you may have to drop this gene from your analysis later. |

Now that you have your genes of interest, you’re ready to start working with the dataset.

## Exploring the gene expression count data

Let’s start by looking at a dataset that includes information on how many times each gene was expressed in a sample. This is the gene expression count dataset.

This dataset gives us an idea of the scale of gene expression - for example, which genes are expressed the most? Which genes are expressed the least? This is particularly useful when you want to know if a gene is usually turned on or off in a sample.

### Package Install and Load

We will need to install the tidyverse package for this activity.

Type the following into the SciServer console and press return to run the code.

install.packages("tidyverse")

Next, we will load the package so it’s ready to use:

library(tidyverse)

|  |
| --- |
| Note |
| Packages are collections of R code, data, and documentation that extend the base functionality of R. Think of them like “expansion packs” on top of your basic R software.  Packages are developed by the R community and made available through repositories like CRAN (Comprehensive R Archive Network), Bioconductor, and GitHub. They are especially useful if you want to do a specialized kind of analysis, such as genomic analysis!  We use the library command to load and attach packages to the R environment. This means links the package you downloaded to your current session of R.  The “tidyverse” package that you loaded is useful for loading, wrangling, and exploring data. |

### Loading the gene expression count data

The gene expression count data for this project is stored online. R has the ability to load datasets from URLs with the tidyverse’s read\_csv command. Let’s load the gene expression count data for the samples that come from just the control-type mice. This contains all the prefrontal cortex and striatum samples taken from control-type mice (so, two brain regions, only one mouse type).

Remember, R code always follows the same basic format! In this case, you are telling R to open the csv file that exists at a URL and save it as an object called “gutbrain\_genes”.



You can load this dataset into R using this code.

gutbrain\_genes <- read\_csv("https://genomicseducation.org/data/mouse\_gutbrain\_de\_counts\_controls.csv")

Rows: 55421 Columns: 59  
── Column specification ────────────────────────────────────────────────────────  
Delimiter: ","  
chr (1): GeneID  
dbl (58): SRR6652471, SRR6652507, SRR6652508, SRR6652511, SRR6652512, SRR665...  
  
ℹ Use `spec()` to retrieve the full column specification for this data.  
ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

When we load this dataset, we see it has 59 columns and 55,421 rows. Each row should represent a gene, and each column should be the counts for an individual mouse (sample), except for the first column (“GeneID”, which is the Ensembl gene ID).

Here are the URLs for the other gene expression count datasets for this project:

*Only ASD-type mouse samples* (contains both prefrontal cortex and striatum; two brain regions, only one mouse type): <https://genomicseducation.org/data/mouse_gutbrain_de_counts_asd.csv>

*Only prefrontal cortex samples* (contains both ASD-type and control-type mice; two mice types, only one brain region): <https://genomicseducation.org/data/mouse_gutbrain_de_counts_prefrontalcortex.csv>

*Only striatum samples* (contains both ASD-type and control-type mice; two mice types, only one brain region): <https://genomicseducation.org/data/mouse_gutbrain_de_counts_striatum.csv>

If you load multiple datasets, **make sure to assign them to different object names**. If you load one dataset and save it as gutbrain\_genes, loading a second dataset and assigning it to gutbrain\_genes will overwrite your first dataset. You need to manually change the object name you’re saving each dataset.

### Exploring and summarizing count data

When you first open the gene expression data, you want to verify that the data has loaded correctly. You can do this with the head and tail commands. Remember, head prints the first 6 rows of a dataset, while tail prints the last 6 rows.

head(gutbrain\_genes)

# A tibble: 6 × 59  
 GeneID SRR6652471 SRR6652507 SRR6652508 SRR6652511 SRR6652512 SRR6652513  
 <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 ENSMUSG0000… 0 0 0 0 0 138  
2 ENSMUSG0000… 0 0 0 0 0 0  
3 ENSMUSG0000… 0 0 0 0 0 0  
4 ENSMUSG0000… 0 0 0 0 0 0  
5 ENSMUSG0000… 0 0 0 0 0 0  
6 ENSMUSG0000… 0 0 0 0 0 0  
# ℹ 52 more variables: SRR6652514 <dbl>, SRR6652457 <dbl>, SRR6652458 <dbl>,  
# SRR6652459 <dbl>, SRR6652460 <dbl>, SRR6652465 <dbl>, SRR6652466 <dbl>,  
# SRR6652472 <dbl>, SRR6652475 <dbl>, SRR6652476 <dbl>, SRR6652477 <dbl>,  
# SRR6652478 <dbl>, SRR6652483 <dbl>, SRR6652484 <dbl>, SRR6652489 <dbl>,  
# SRR6652490 <dbl>, SRR6652493 <dbl>, SRR6652494 <dbl>, SRR6652495 <dbl>,  
# SRR6652496 <dbl>, SRR6652501 <dbl>, SRR6652502 <dbl>, SRR6652519 <dbl>,  
# SRR6652520 <dbl>, SRR6652443 <dbl>, SRR6652389 <dbl>, SRR6652390 <dbl>, …

tail(gutbrain\_genes)

# A tibble: 6 × 59  
 GeneID SRR6652471 SRR6652507 SRR6652508 SRR6652511 SRR6652512 SRR6652513  
 <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 ENSMUSG0000… 0 0 0 0 0 0  
2 ENSMUSG0000… 17809 4859 10099 9495 10105 11322  
3 ENSMUSG0000… 5384 1952 4737 4702 4385 3985  
4 ENSMUSG0000… 62645 30726 49800 57108 58010 35727  
5 ENSMUSG0000… 100 239 34 131 0 454  
6 ENSMUSG0000… 1510 537 2655 1769 2278 1810  
# ℹ 52 more variables: SRR6652514 <dbl>, SRR6652457 <dbl>, SRR6652458 <dbl>,  
# SRR6652459 <dbl>, SRR6652460 <dbl>, SRR6652465 <dbl>, SRR6652466 <dbl>,  
# SRR6652472 <dbl>, SRR6652475 <dbl>, SRR6652476 <dbl>, SRR6652477 <dbl>,  
# SRR6652478 <dbl>, SRR6652483 <dbl>, SRR6652484 <dbl>, SRR6652489 <dbl>,  
# SRR6652490 <dbl>, SRR6652493 <dbl>, SRR6652494 <dbl>, SRR6652495 <dbl>,  
# SRR6652496 <dbl>, SRR6652501 <dbl>, SRR6652502 <dbl>, SRR6652519 <dbl>,  
# SRR6652520 <dbl>, SRR6652443 <dbl>, SRR6652389 <dbl>, SRR6652390 <dbl>, …

You should see rows with data in your R environment. Previously you found out that:

Each row should represent a gene, and each column should be the counts for an individual mouse (sample), except for the first column (“GeneID”, which is the Ensembl gene ID).

Can you verify that your first column is called “GeneID” and each following column is named something that starts with “SRR”? (This is how each of the samples was named.) If so, great! The dataset loaded correctly. If not, check that you haven’t mistyped or incorrectly copied the code.

You always want to check that your columns and rows are loaded and displaying as expected.

### Plotting histograms and dealing with zero counts

A histogram is a good way to visually explore the distribution of the read counts, as well as get an idea of how many genes have few or no reads.

Before we make the histogram, we actually need to sum up the total reads for each gene (row). We can do this in R using the command rowSums. Remember, our dataset has 59 columns, but only columns 2 - 59 contain count data for each gene. The first column is the “GeneID”. So we need to tell R to do the row sum for columns 2-59, then save it as a new column called “total\_counts”.



Because we’re saving the results of our calculation as a new column, we have to use a $ to tell R to make a new column. In this case, R interprets gutbrain\_genes$total\_counts as the total\_counts column in the gutbrain\_genes dataset (object).

gutbrain\_genes$total\_counts <- rowSums(gutbrain\_genes[2:59])

Let’s look at the mean, median, maximum, and minimum values (called descriptive statistics) of the new total\_counts column we made. This lets us know what sort of scale we might need to use when making a graph of the data. We can do this using the summary command.



summary(gutbrain\_genes$total\_counts)

Min. 1st Qu. Median Mean 3rd Qu. Max.   
0.000e+00 0.000e+00 2.958e+03 1.798e+06 3.478e+05 3.321e+09

Notice that we didn’t use the <- to save the summary data to an object! You can, but we’re just curious to check out the descriptive statistics and won’t be using them again.



Let’s try saving the output of the summary statement with the <-.

summary\_stats <- summary(gutbrain\_genes$total\_counts)

When you save the output as summary\_stats, R doesn’t automatically print the output into your console. You’ll have to use a second command to tell R to print.

summary\_stats

Min. 1st Qu. Median Mean 3rd Qu. Max.   
0.000e+00 0.000e+00 2.958e+03 1.798e+06 3.478e+05 3.321e+09

No matter whether you print the summary statistics directly to the console, or save it first as a new object, the results are the same. We can see that some genes are not expressed at all in this dataset (“total\_genes” is 0), while other genes are expressed a lot. We’ll want to take this into account when we make a histogram to explore the distribution of counts visually.

We can make the histogram using the histogram command. Use a log10 scale on the x-axis to plot “total\_counts”. We’re nesting a log transform within the histogram command, as well as adding an argument that lets us add a title.



hist(log10(gutbrain\_genes$total\_counts), main="Log10 Read Counts for Mouse RNA-seq sample")



You may choose to deal with zero-count genes by adding 0.1 to each count before the log10 transformation. We can do this by creating a new column in the gutbrain\_genes dataset.



Remember you did this a couple steps ago when you make the “total\_counts” column! You’ll do essentially the same thing, except you’ll tell R to add 0.1 to the “total\_counts” column and save that value in a new column called “total\_counts\_01”.

gutbrain\_genes$total\_counts\_01 <- gutbrain\_genes$total\_counts + 0.1

Now just recreate your histogram using “total\_count\_01” instead of “total\_counts”.



hist(log10(gutbrain\_genes$total\_counts\_01), main="Transformed Log10 Read Counts for Mouse RNA-seq sample")



There’s obviously a lot of variation in expression among genes!

### Looking at read count for a single gene

You may want to look at the read count for a specific gene in your dataset. This can be done with the filter command. Let’s look at the read counts for the Reg3a gene, across all mice. Remember, this is the gene we looked up earlier in the MGI database. The Ensembl gene ID is “ENSMUSG00000079516”.

For filter, we need to tell R which dataset to use, and which column to search in.

filter(gutbrain\_genes, GeneID == "ENSMUSG00000079516")

Luckily, we chose a gene that is indeed expressed in the dataset! Making sure yours chosen genes are expressed will be important for our next step.

### Visualize gene counts of multiple genes across groups

We can also look at the differential gene expression of a set of genes at once! This is usually done by creating a **heatmap**. In RNA-seq analyses, we usually put individual samples on the x-axis and normalized gene counts on the y-axis.



When creating heatmaps, we want to work with a dataset that includes normalized gene expression data for each gene. We loaded a dataset like this above that we called gutbrain\_genes. If you don’t have an object in your environment called this, scroll back up to the “**Loading the gene expression count data**” section and run the code again.

The dataset you loaded has expression data for 55,421 different genes, which is too many to visualize. (You can try, but R will usually throw an error if you do.) In general, though, we are mostly interested in looking at the visualization of a particular group of genes, or a gene set.

Let’s say we have identified 6 mouse genes that have been linked to the development of motor neurons. We’re interested in whether these genes show differential gene expression between the striatum and the prefrontal cortex. We can use the filter command to pull out normalized gene expression data for just these six genes.

These are the genes (with gene IDs) that we’re including in our gene set:

* Sema3a: ENSMUSG00000028883
* Mapk8: ENSMUSG00000021936
* Nrcam: ENSMUSG00000020598
* Dlg4: ENSMUSG00000020886
* Slit1: ENSMUSG00000025020
* Creb1: ENSMUSG00000025958

First, create a vector that contains these gene IDs. (You can replace the gene IDs with the ones for your genes of interest, but make sure you use Ensembl gene IDs! They should all start with “ENSMUSG”.)



Each gene ID should have quotation marks around it, and the different IDs should be separated using a comma.

gene\_set <- c("ENSMUSG00000028883", "ENSMUSG00000021936", "ENSMUSG00000020598", "ENSMUSG00000020886", "ENSMUSG00000025020", "ENSMUSG00000025958")

Now we simply filter the gutbrain\_genes dataset so that we only keep the rows for the gene IDs in our gene set. Remember, we want to use the filter command. The combination of filter and %in% tells R to only keep those rows in the gutbrain\_genes dataset if the gene ID is also found in our gene list we made earlier.



gutbrain\_motor <- filter(gutbrain\_genes, gutbrain\_genes$GeneID %in% gene\_set)

Now we have our data subsetted the way we want, but we aren’t quite ready for the heatmap yet. In order for R to make a heatmap from our data, we do have to reformat things a little bit. Right now our data is in “tibble” format, but we want it in a numerical “matrix” format. We can’t switch directly from “tibble” to “matrix” (because the GeneID column is character data, not numerical data). Instead, let’s first convert from “tibble” to a format called “dataframe”.



gutbrain\_motor <- as.data.frame(gutbrain\_motor)

To our eyes, not much has changed in the format, but it’s very different to R. Now we’re going to tell R to name the rows based on the “GeneID” column.



We are also going to delete the current “GeneID” column because we no longer need it, as well as the two columns we created earlier, “total\_counts” and “total\_counts\_01”.



rownames(gutbrain\_motor) <- gutbrain\_motor$GeneID  
  
gutbrain\_motor$GeneID <- NULL  
  
gutbrain\_motor$total\_counts <- NULL  
  
gutbrain\_motor$total\_counts\_01 <- NULL

Great! Now we can change the format of gutbrain\_motor\_controls dataset to “matrix” format in order to make our heatmap using the heatmap function.



gutbrain\_motor <- as.matrix(gutbrain\_motor)  
  
heatmap(gutbrain\_motor)



Congratulations, you have just created a heatmap! The map itself is a series of stacked boxes. Each square within the heatmap represents the gene count for a particular gene in a particular sample. By default squares with the high gene counts are colored dark red. Squares with the lowest gene counts are colored pale yellow.

By default, R will order the samples (what we have on the x-axis across the bottom) in a way that clusters similar samples together. This is why we see columns of lighter yellow in the center of the heatmap, with darker orange and red columns on the edges. This suggests that there are groups of samples that have higher gene expression, but it’s hard to tell right now if these groups correspond to the prefrontal cortex vs striatum (which is what we’re interested in). It would be helpful if we could tell R to group based on prefrontal cortex vs striatum instead.

Luckily, this is possible! Our dataset is already in the order we want (with all the prefrontal cortex samples first). We can just tell R to use the default ordering of the dataset instead of clustering and rearranging the x-axis.



|  |
| --- |
| How are the samples in my dataset ordered? |
| When these datasets were made, the samples were ordered in a specific way so that you could easily make your heatmaps.  *Only control-type mouse samples* (contains both prefrontal cortex and striatum; two brain regions, only one mouse type): prefrontal cortex samples first, then the striatum samples  *Only ASD-type mouse samples* (contains both prefrontal cortex and striatum; two brain regions, only one mouse type): prefrontal cortex samples first, then the striatum samples  *Only prefrontal cortex samples* (contains both ASD-type and control-type mice; two mice types, only one brain region): ASD-type samples first, then control-type samples  *Only striatum samples* (contains both ASD-type and control-type mice; two mice types, only one brain region): ASD-type samples first, then control-type samples  But if you ever want to verify the details for a particular sample, you can check out the metadata! This is the data (or information) *about* each of the samples. The metadata is stored at <mouse\_gutbrain\_metadata.csv>. You can load it into R using read\_csv. |

heatmap(gutbrain\_motor, Colv = NA, Rowv = NA)



Perfect! Now we can easily see that four of our six genes seem to be expressed in lower levels by the samples on the right side of the heatmap (the striatum samples), while one gene is expressed in higher levels by the striatum. This is definitely intriguing!

Notice that not all the striatum samples have the same expression for each gene. This is normal variation and could be due to actual difference in gene expression levels or be the result of laboratory methods.

Finally, we can customize the heatmap by changing the color scheme and adding a title. As as example, we’re going to tell R to make a spectrum of colors between white and blue, with white indicating the lowest expression and blue indicating the highest expression, and 16 possible color categories. (We are also adding a title to the heatmap.)



You can find an extensive list of all the colors available in R at <https://www.datanovia.com/en/blog/awesome-list-of-657-r-color-names/>. **The general convention in biology is that the lightest colors on a heatmap indicate the lowest values.**

heatmap(gutbrain\_motor, Colv = NA, Rowv = NA, col = colorRampPalette(c("white", "darkblue"))(16), main = " Motor Neuron Gene List Expression")



Right now we have all the genes identified by the Ensembl gene IDs. Most people have trouble looking at these gene IDs and knowing exactly what gene it represents. We can change the in our dataset using the case\_when command in R.



rownames(gutbrain\_motor) <- case\_when(  
 rownames(gutbrain\_motor) == "ENSMUSG00000028883" ~ "Sema3a",  
 rownames(gutbrain\_motor) == "ENSMUSG00000021936" ~ "Mapk8",  
 rownames(gutbrain\_motor) == "ENSMUSG00000020598" ~ "Nrcam",  
 rownames(gutbrain\_motor) == "ENSMUSG00000020886" ~ "Dlg4",  
 rownames(gutbrain\_motor) == "ENSMUSG00000025020" ~ "Slit1",  
 rownames(gutbrain\_motor) == "ENSMUSG00000025958" ~ "Creb1"  
)

Now we can run our heatmap code again.

heatmap(gutbrain\_motor, Colv = NA, Rowv = NA, col = colorRampPalette(c("white", "darkblue"))(16), main = " Motor Neuron Gene List Expression")



Finally, here’s all the code you just ran to create your heatmap in one place, in case you want to run it again.

#define your geneset and make a vector  
gene\_set <- c("ENSMUSG00000028883", "ENSMUSG00000021936", "ENSMUSG00000020598", "ENSMUSG00000020886", "ENSMUSG00000025020", "ENSMUSG00000025958")  
  
#create new dataset that only include expression data for your geneset  
gutbrain\_motor <- filter(gutbrain\_genes, gutbrain\_genes$GeneID %in% gene\_set)  
  
#change format of dataset from "tibble" to "matrix"  
gutbrain\_motor <- as.data.frame(gutbrain\_motor)  
rownames(gutbrain\_motor) <- gutbrain\_motor$GeneID  
gutbrain\_motor$GeneID <- NULL  
gutbrain\_motor$total\_counts <- NULL  
gutbrain\_motor <- as.matrix(gutbrain\_motor)  
  
#create heatmap with custom colors and title  
heatmap(gutbrain\_motor, Colv = NA, Rowv = NA, col = colorRampPalette(c("white", "darkblue"))(16), main = " Motor Neuron Gene List Expression")

## Exploring the differential expression data

Now let’s look a differential expression dataset, which compares gene expression counts between two groups. This dataset gives us an idea how the change in a gene’s expression might be associated with a disorder or a cell type. This is particularly useful for beginning to map the genetic drivers of a particular phenotype.

### Package Install and Load

Before you start, make sure to load the tidyverse package! You should still have it installed from last time.

library(tidyverse)

### Loading the differential expression data

Let’s say you want to open the dataset that compares gene expression between ASD and control mice in both brain regions and call it asd\_vs\_c. We will do this using read\_csv. Copy and paste this command into your console:

asd\_vs\_c <- read\_csv("https://genomicseducation.org/data/mouse\_gutbrain\_de\_autismVcontrol.csv")

Rows: 55421 Columns: 7  
── Column specification ────────────────────────────────────────────────────────  
Delimiter: ","  
chr (1): GeneID  
dbl (6): baseMean, log2FoldChange, lfcSE, stat, pvalue, padj  
  
ℹ Use `spec()` to retrieve the full column specification for this data.  
ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

Here are the URLs for all the possible comparisons you can examine with this dataset:

**Comparing gene expression between ASD and control mice**

Both brain regions: <https://genomicseducation.org/data/mouse_gutbrain_de_autismVcontrol.csv>

Prefrontal cortex only: <https://genomicseducation.org/data/mouse_gutbrain_de_autismVcontrol_in_prefrontalcortex.csv>

Striatum only: <https://genomicseducation.org/data/mouse_gutbrain_de_autismVcontrol_in_striatum.csv>

**Comparing gene expression between prefrontal cortex and striatum**

All mice: <https://genomicseducation.org/data/mouse_gutbrain_de_tissuetype.csv>

Only ASD mice: <https://genomicseducation.org/data/mouse_gutbrain_de_tissuetype_in_ASDmice.csv>

Only control mice: <https://genomicseducation.org/data/mouse_gutbrain_de_tissuetype_in_controlmice.csv>

|  |
| --- |
| Which differential expression dataset should I use? |
| Are you mostly interested in how gene expression differs between the two brain regions? Are you interested in how neurons develop, or how groups of genes involved in movement might be expressed differently in two different regions? These sorts of questions are best to explore using **striatum and prefrontal cortex** comparisons.  Are you really curious about differential expression in Autism Spectrum Disorder-type mice compared to control-type mice? Are you curious about potential associations between gene expression differences in known neurodevelopmental or neurodegenerative genes? These questions and others can be explored using the **ASD-type mice and control-type mice** comparisons. |

When you open the differential expression dataset, you’ll notice some new column names. The ones you want to pay attention to are **gene**, **log2FoldChange**, and **padj**.

head(asd\_vs\_c)

# A tibble: 6 × 7  
 GeneID baseMean log2FoldChange lfcSE stat pvalue padj  
 <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 ENSMUSG00000089657 16.9 -25.5 1.81 -14.1 2.18e-45 7.29e-41  
2 ENSMUSG00000083812 14.6 -25.4 1.80 -14.1 3.82e-45 7.29e-41  
3 ENSMUSG00000094151 11.2 -25.0 1.79 -14.0 2.97e-44 3.77e-40  
4 ENSMUSG00000102414 14.9 -25.4 1.89 -13.4 4.00e-41 3.81e-37  
5 ENSMUSG00000074445 11.5 -25.0 1.89 -13.3 3.44e-40 2.62e-36  
6 ENSMUSG00000053773 10.6 -24.9 1.99 -12.5 6.66e-36 4.23e-32

**gene** is the Ensembl gene ID. Each row contains differential expression data for a single gene.

**log2FoldChange** is a measure that tells us whether a gene is upregulated or downregulated in experimental mice versus the control mice. We’ll discuss this in more depth in the next section.

**padj** is a measure of whether the difference in gene expression between the experimental and control mice is statistically significant. It stands for “p-adjusted”. A p-value was calculated for each gene and then a correction was added to account for the fact that many statistical tests were being performed.

### Ranking the genes by log2FoldChange

The log2FoldChange value gives us an idea of which genes show the greatest differential expression between the ASD mice and the control mice. It is a log-transformed ratio of how many gene transcripts were found in ASD-type mice compared to control-type mice. We take the log2 of this ratio (the fold change) because it makes interpretation easier.



Genes with no difference in expression between groups have log2FoldChange values close to zero. Negative log2FoldChange values means a **decreased expression** (or, a downregulation in gene expression) in the experimental group versus the control group. Positive log2FoldChange values indicate genes that are upregulated, or have **increased expression** in the experimental group versus the control group.

We can arrange the table based on these values (going from smallest to large using the arrange command.



asd\_vs\_c\_ascending <- arrange(asd\_vs\_c, log2FoldChange)  
head(asd\_vs\_c\_ascending)

# A tibble: 6 × 7  
 GeneID baseMean log2FoldChange lfcSE stat pvalue padj  
 <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 ENSMUSG00000089657 16.9 -25.5 1.81 -14.1 2.18e-45 7.29e-41  
2 ENSMUSG00000102414 14.9 -25.4 1.89 -13.4 4.00e-41 3.81e-37  
3 ENSMUSG00000083812 14.6 -25.4 1.80 -14.1 3.82e-45 7.29e-41  
4 ENSMUSG00000074445 11.5 -25.0 1.89 -13.3 3.44e-40 2.62e-36  
5 ENSMUSG00000094151 11.2 -25.0 1.79 -14.0 2.97e-44 3.77e-40  
6 ENSMUSG00000053773 10.6 -24.9 1.99 -12.5 6.66e-36 4.23e-32

The genes at the top of this table have the most negative log2FoldChange values and are downregulated in the experimental mice.

We can use a similar command to look at the genes with the largest positive log2FoldChange. These are genes that are upregulated in experimental mice.

asd\_vs\_c\_descending <- arrange(asd\_vs\_c, desc(log2FoldChange))  
head(asd\_vs\_c\_descending)

# A tibble: 6 × 7  
 GeneID baseMean log2FoldChange lfcSE stat pvalue padj  
 <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 ENSMUSG00000102375 7.19 24.1 2.29 10.5 6.88e-26 2.91e-22  
2 ENSMUSG00000104583 5.77 24.0 2.31 10.4 2.48e-25 9.47e-22  
3 ENSMUSG00000042414 6.81 6.42 2.93 2.19 2.85e- 2 1.00e+ 0  
4 ENSMUSG00000048763 6.33 6.32 2.30 2.75 6.04e- 3 1.00e+ 0  
5 ENSMUSG00000117911 5.18 6.03 2.77 2.18 2.93e- 2 1.00e+ 0  
6 ENSMUSG00000085421 4.74 5.90 2.93 2.01 4.42e- 2 1.00e+ 0

When we’re interpreting the log2FoldChange, remember that you’ll need to do a little bit of math. A log2FoldChange value of 1 is a fold change value of 2 (2^1 = 2). Similarly, a log2FoldChange value of 4 is a fold change value of 8 (2^4 = 8)

If there is a two fold **increase** (fold change = 2, log2FoldChange = 1) between groups A and B, then the expression in group A is twice as big as the expression in group B.

If there is a two fold **decrease** (fold change = 0.5, log2FoldChange = -1) between groups A and B, then the expression in group A is half as big as the expression in group B (or B is twice as big as A).

### Arranging dataset based on padj

Maybe you also want to arrange the genes by whether the estimated differential expression is statistically significant. Instead of log2FoldChange, you will want to look at padj. You can use the arrange command to sort the results to put the smallest padj values first.

asd\_vs\_c\_sorted <- arrange(asd\_vs\_c, padj)  
head(asd\_vs\_c\_sorted)

# A tibble: 6 × 7  
 GeneID baseMean log2FoldChange lfcSE stat pvalue padj  
 <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 ENSMUSG00000089657 16.9 -25.5 1.81 -14.1 2.18e-45 7.29e-41  
2 ENSMUSG00000083812 14.6 -25.4 1.80 -14.1 3.82e-45 7.29e-41  
3 ENSMUSG00000094151 11.2 -25.0 1.79 -14.0 2.97e-44 3.77e-40  
4 ENSMUSG00000102414 14.9 -25.4 1.89 -13.4 4.00e-41 3.81e-37  
5 ENSMUSG00000074445 11.5 -25.0 1.89 -13.3 3.44e-40 2.62e-36  
6 ENSMUSG00000053773 10.6 -24.9 1.99 -12.5 6.66e-36 4.23e-32

Doing this is a useful way to check how many (or if any!) genes have statistically significant expression changes between groups.

### Creating a volcano plot to visualize differential expression of multiple genes

A volcano plot is a great way to visualize differential gene expression between two groups. These types of plots have log2FoldChange on the x-axis and a log transformed padj on the y-axis.



We will first need to do a log transformation of the padj variable. We do this in order to spread our values out along the y-axis - otherwise, all the dots that represent genes will be too scrunched together. Remember, we did this earlier when we made our histograms!

asd\_vs\_c$padj\_trans <- -log10(asd\_vs\_c$padj)

Now, we just create a scatterplot. The code for this is kind of complicated, but we’ll walk through it.



Here’s the code for your basic volcano plot. Notice that we can just use the column names for x and y! There’s no need to use the dataset$column\_name syntax. The ggplot command already knows to look for these column names in the asd\_vs\_c dataset.

ggplot(data = asd\_vs\_c, aes(x = log2FoldChange, y = padj\_trans)) +  
 geom\_point()

Warning: Removed 17318 rows containing missing values or values outside the scale range  
(`geom\_point()`).



We can also color code the genes (points) which are significantly upregulated or downregulated. In order to do this, we will need to create a new column in our dataset that identifies whether a gene is upregulated in ASD-type mice, downregulated in ASD-type mice, or has the same regulation in ASD-type mice compared to control-type mice.

In this new column, we can decide the thresholds that a log2FoldChange value must meet in order to be considered upregulated or downregulated. We will consider genes to be upregulated if the log2fold change is greater than 0.6 and downregulated if it is less than -0.6. **You can change these thresholds to whatever makes sense for you.**

log2foldchange is just a representation of a ratio, so the sign indicates the direction of regulation. It’s important to remember which group is your comparison group!

For example, if you are look at regulation in group A compared to group B, you might have a log2foldchange value of 1 (indicating the gene is expressed twice as much in A as it is in B).

However, if you are looking at regulation in group B compared to group A, the log2foldchange value will be -1 (because the gene is expressed half as much in B as it is in A).

The comparison group matters with ratios!

We are also setting a significant p-value (padj) as less than 0.05 (but you can also change this based on your best judgment).

We start by creating a new column called diffexpressed and setting everything to “NO” (as in “no expression differences between ASD-type and control-type mice”). We then specify the genes for which this column should be changed to “UP” or “DOWN” (based on log2FoldChange and padj).

To recap: A gene is labeled “UP” if its log2FoldChange value is greater than 0.6 *and* its adjusted p-value is less than 0.05.

A gene is labeled “DOWN” if its log2FoldChange value is less than -0.6 *and* its adjusted p-value is less than 0.05.

asd\_vs\_c$diffexpressed <- "NO"  
  
asd\_vs\_c$diffexpressed[asd\_vs\_c$log2FoldChange > 0.6 & asd\_vs\_c$padj < 0.05] <- "UP"  
  
asd\_vs\_c$diffexpressed[asd\_vs\_c$log2FoldChange < -0.6 & asd\_vs\_c$padj < 0.05] <- "DOWN"  
  
head(asd\_vs\_c)

# A tibble: 6 × 9  
 GeneID baseMean log2FoldChange lfcSE stat pvalue padj padj\_trans  
 <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 ENSMUSG00000… 16.9 -25.5 1.81 -14.1 2.18e-45 7.29e-41 40.1  
2 ENSMUSG00000… 14.6 -25.4 1.80 -14.1 3.82e-45 7.29e-41 40.1  
3 ENSMUSG00000… 11.2 -25.0 1.79 -14.0 2.97e-44 3.77e-40 39.4  
4 ENSMUSG00000… 14.9 -25.4 1.89 -13.4 4.00e-41 3.81e-37 36.4  
5 ENSMUSG00000… 11.5 -25.0 1.89 -13.3 3.44e-40 2.62e-36 35.6  
6 ENSMUSG00000… 10.6 -24.9 1.99 -12.5 6.66e-36 4.23e-32 31.4  
# ℹ 1 more variable: diffexpressed <chr>

Okay! We can now add some color to the volcano plot we made earlier. We do this by adding a line to our code that tells R to color the data points based on the diffexpressed column (col = diffexpressed). This will be added in the aes section of our code.

We also add a line about the colors we want to use (scale\_color\_manual). In this case, we have chosen to make the points for downregulated genes turquoise and the points for upregulated genes gold. The additional command labels just creates the labels on the legend.

You can find an extensive list of all the colors available in R at <https://www.datanovia.com/en/blog/awesome-list-of-657-r-color-names/>.

ggplot(data = asd\_vs\_c, aes(x = log2FoldChange, y = padj\_trans, col = diffexpressed)) +  
 geom\_point() +  
 scale\_color\_manual(values = c("turquoise", "grey", "gold"),   
 labels = c("Downregulated", "Not significant", "Upregulated"))

Warning: Removed 17318 rows containing missing values or values outside the scale range  
(`geom\_point()`).



This is all well and good, but maybe we want to see where the genes from our gene list fall on this plot. Luckily, we can also label genes of interest in our plot.

As an example, let’s label the Reg3a gene (the same one we looked up in the MGI database), as well as Pcdh12 (or protocadherin 12). The two gene IDs we need to know are ENSMUSG00000079516 and ENSMUSG00000024440.

We will create another new column in our dataset. We will then tell R In this new column, the genes in our list (Reg3a and Pcdh12) are named, which everything else will have a missing name. When we remake the volcano plot, R will *only* label those points that do not have missing names in the gene\_label column.

This is going to require a little bit of R trickery and something called a “case\_when” statement.

In R, an “case\_when” statement tells R to do something only when a condition is true. In our command, R is going through the asd\_vs\_c row by row. For each row, we are telling R to first check whether the geneID of that row is “ENSMUSG00000079516”. If it is, then R will fill in the gene\_label column with the gene symbol “Reg3a”. If it is not, R moves onto the next part of our command.

When R moves onto this second command, it’s now checking to see if the geneID for the row is “ENSMUSG00000024440”. If it is, then R fills in the gene\_label column with the gene symbol “Pcdh12”. If it is not, R moves to the third command in the case\_when statement. The final line tells R to fill in the gene\_label column with “NA” for any row that didn’t match the previous commands. This will tell R in the future that the information for this column is missing.

asd\_vs\_c$gene\_label <- case\_when(  
 asd\_vs\_c$GeneID == "ENSMUSG00000079516" ~ "Reg3a",  
 asd\_vs\_c$GeneID == "ENSMUSG00000024440" ~ "Pcdh12",  
 TRUE ~ NA\_character\_  
)

Great! You can check to see what the new column gene\_label looks like by using the head command.

head(asd\_vs\_c)

# A tibble: 6 × 10  
 GeneID baseMean log2FoldChange lfcSE stat pvalue padj padj\_trans  
 <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
1 ENSMUSG00000… 16.9 -25.5 1.81 -14.1 2.18e-45 7.29e-41 40.1  
2 ENSMUSG00000… 14.6 -25.4 1.80 -14.1 3.82e-45 7.29e-41 40.1  
3 ENSMUSG00000… 11.2 -25.0 1.79 -14.0 2.97e-44 3.77e-40 39.4  
4 ENSMUSG00000… 14.9 -25.4 1.89 -13.4 4.00e-41 3.81e-37 36.4  
5 ENSMUSG00000… 11.5 -25.0 1.89 -13.3 3.44e-40 2.62e-36 35.6  
6 ENSMUSG00000… 10.6 -24.9 1.99 -12.5 6.66e-36 4.23e-32 31.4  
# ℹ 2 more variables: diffexpressed <chr>, gene\_label <chr>

Now we just need to remake our volcano plot and add some to code that tells R to label the points we’re interested in. The command label = gene\_label goes right after the code that tells R how to color the points. We also need to add something at the end of our code that tells R to put the labels on top of the plot that we’ve made. That’s the + geom\_text\_repel(max.overlaps = Inf) command.

In order for this command to work, we do need to install and load a new library called ggrepel.

install.packages('ggrepel')

Installing package into '/usr/local/lib/R/site-library'  
(as 'lib' is unspecified)

library(ggrepel)

Excellent. Now that we have installed and loaded this package (which you only need to do once for each R session!), we can run the code to create the volcano plot. Remember, we added label = gene\_label to tell R to add labels to the points, and geom\_text\_repel(max.overlaps = Inf) to tell R to put the labels on top of the plot so we can see them.

ggplot(data = asd\_vs\_c, aes(x = log2FoldChange, y = padj\_trans, col = diffexpressed, label = gene\_label)) +  
 geom\_point() +  
 scale\_color\_manual(values = c("turquoise", "grey", "gold"),   
 labels = c("Downregulated", "Not significant", "Upregulated")) +  
 geom\_text\_repel(max.overlaps = Inf)

Warning: Removed 17318 rows containing missing values or values outside the scale range  
(`geom\_point()`).

Warning: Removed 55419 rows containing missing values or values outside the scale range  
(`geom\_text\_repel()`).



You may want to add a title to your plot. This can be done with the ggtitle command.

ggplot(data = asd\_vs\_c, aes(x = log2FoldChange, y = padj\_trans, col = diffexpressed, label = gene\_label)) +  
 geom\_point() +  
 scale\_color\_manual(values = c("turquoise", "grey", "gold"),   
 labels = c("Downregulated", "Not significant", "Upregulated")) +  
 geom\_text\_repel(max.overlaps = Inf) +  
 ggtitle("Differential Expression in Mouse Prefrontal Cortex")

Warning: Removed 17318 rows containing missing values or values outside the scale range  
(`geom\_point()`).

Warning: Removed 55419 rows containing missing values or values outside the scale range  
(`geom\_text\_repel()`).



Here’s all the code we used to build a volcano plot in one place so that you can easily run it again (and make modifications).

#log transform the padj values so the y-axis is formatted for plotting  
asd\_vs\_c$padj\_trans <- -log10(asd\_vs\_c$padj)  
  
#Create the diffexpressed column so that we can color the upregulated and downregulated genes on the plot  
asd\_vs\_c$diffexpressed <- "NO"  
  
asd\_vs\_c$diffexpressed[asd\_vs\_c$log2FoldChange > 0.6 & asd\_vs\_c$padj < 0.05] <- "UP"  
  
asd\_vs\_c$diffexpressed[asd\_vs\_c$log2FoldChange < -0.6 & asd\_vs\_c$padj < 0.05] <- "DOWN"  
  
#Create the gene\_label column so that we can label individual points  
asd\_vs\_c$gene\_label <- case\_when(  
 asd\_vs\_c$gene == "ENSMUSG00000079516" ~ "Reg3a",  
 asd\_vs\_c$gene == "ENSMUSG00000024440" ~ "Pcdh12",  
 TRUE ~ NA\_character\_  
)  
  
#Make the final plot  
ggplot(data = asd\_vs\_c, aes(x = log2FoldChange, y = padj\_trans, col = diffexpressed, label = gene\_label)) +  
 geom\_point() +  
 scale\_color\_manual(values = c("turquoise", "grey", "gold"),   
 labels = c("Downregulated", "Not significant", "Upregulated")) +  
 geom\_text\_repel(max.overlaps = Inf) +  
 ggtitle("Differential Expression in Mouse Prefrontal Cortex")

## Running a gene set enrichment analysis

We can use a gene set enrichment analysis to explore biological processes might be affected by the genes that are differentially expressed in our comparison. We can also use this to figure out the types of processes genes on our gene list are involved in.

If you are using the SciServer environment, you can skip the **Creating the analysis functions** section below and go straight to the **Loading the data** section. The functions you need have already been created and are saved for your use on SciServer.

### Creating the analysis functions

Before we get started with the actual analysis, we’ll need to run a couple of commands to create a special function to do the gene set analysis. This function will allow us to do a complex set of analysis steps in a single line of code.

Before we can create our function, we need to install a couple of packages. Run the code below.

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

Bioconductor version '3.18' is out-of-date; the current release version '3.21'  
 is available with R version '4.5'; see https://bioconductor.org/install

BiocManager::install("clusterProfiler")

'getOption("repos")' replaces Bioconductor standard repositories, see  
'help("repositories", package = "BiocManager")' for details.  
Replacement repositories:  
 CRAN: https://p3m.dev/cran/\_\_linux\_\_/jammy/2024-02-28

Bioconductor version 3.18 (BiocManager 1.30.22), R 4.3.2 (2023-10-31)

Warning: package(s) not installed when version(s) same as or greater than current; use  
 `force = TRUE` to re-install: 'clusterProfiler'

Old packages: 'boot', 'cluster', 'foreign', 'lattice', 'MASS', 'Matrix',  
 'mgcv', 'nlme', 'rpart', 'survival'

BiocManager::install("org.Mm.eg.db")

'getOption("repos")' replaces Bioconductor standard repositories, see  
'help("repositories", package = "BiocManager")' for details.  
Replacement repositories:  
 CRAN: https://p3m.dev/cran/\_\_linux\_\_/jammy/2024-02-28  
Bioconductor version 3.18 (BiocManager 1.30.22), R 4.3.2 (2023-10-31)

Warning: package(s) not installed when version(s) same as or greater than current; use  
 `force = TRUE` to re-install: 'org.Mm.eg.db'

Old packages: 'boot', 'cluster', 'foreign', 'lattice', 'MASS', 'Matrix',  
 'mgcv', 'nlme', 'rpart', 'survival'

You may have noticed that the installation commands for these two packages are different than the ones you used previously with the tidyverse and ggrepel packages. That’s because the clusterProfiler and org.Mm.eg.db packages are saved in a different location (called a repository) than tidyverse and ggrepel. Both clusterprofiler and org.Mm.eg.db are part of the Bioconductor Project, which focuses on developing, maintaining, and distributing software specifically for biological and bioinformatics research.

Once these two packages have been installed, make sure you load them.

library("clusterProfiler")

clusterProfiler v4.10.1 For help: https://yulab-smu.top/biomedical-knowledge-mining-book/  
  
If you use clusterProfiler in published research, please cite:  
T Wu, E Hu, S Xu, M Chen, P Guo, Z Dai, T Feng, L Zhou, W Tang, L Zhan, X Fu, S Liu, X Bo, and G Yu. clusterProfiler 4.0: A universal enrichment tool for interpreting omics data. The Innovation. 2021, 2(3):100141

Attaching package: 'clusterProfiler'

The following object is masked from 'package:purrr':  
  
 simplify

The following object is masked from 'package:stats':  
  
 filter

library("org.Mm.eg.db")

Loading required package: AnnotationDbi

Loading required package: stats4

Loading required package: BiocGenerics

Attaching package: 'BiocGenerics'

The following objects are masked from 'package:lubridate':  
  
 intersect, setdiff, union

The following objects are masked from 'package:dplyr':  
  
 combine, intersect, setdiff, union

The following objects are masked from 'package:stats':  
  
 IQR, mad, sd, var, xtabs

The following objects are masked from 'package:base':  
  
 anyDuplicated, aperm, append, as.data.frame, basename, cbind,  
 colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,  
 get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,  
 match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,  
 Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,  
 table, tapply, union, unique, unsplit, which.max, which.min

Loading required package: Biobase

Welcome to Bioconductor  
  
 Vignettes contain introductory material; view with  
 'browseVignettes()'. To cite Bioconductor, see  
 'citation("Biobase")', and for packages 'citation("pkgname")'.

Loading required package: IRanges

Loading required package: S4Vectors

Attaching package: 'S4Vectors'

The following object is masked from 'package:clusterProfiler':  
  
 rename

The following objects are masked from 'package:lubridate':  
  
 second, second<-

The following objects are masked from 'package:dplyr':  
  
 first, rename

The following object is masked from 'package:tidyr':  
  
 expand

The following object is masked from 'package:utils':  
  
 findMatches

The following objects are masked from 'package:base':  
  
 expand.grid, I, unname

Attaching package: 'IRanges'

The following object is masked from 'package:clusterProfiler':  
  
 slice

The following object is masked from 'package:lubridate':  
  
 %within%

The following objects are masked from 'package:dplyr':  
  
 collapse, desc, slice

The following object is masked from 'package:purrr':  
  
 reduce

Attaching package: 'AnnotationDbi'

The following object is masked from 'package:clusterProfiler':  
  
 select

The following object is masked from 'package:dplyr':  
  
 select

Now we can make our function! The code below makes two functions. A *function* in R is just a block of organized, reusable code that performs a specific task. The commands you have been using in your analysis (like library, filter, and heatmap, to name a few examples) are actually functions that someone else wrote and saved in packages for everyone to use.

The functions you’re creating also perform special tasks. runClusterProfiler is going to pull information about the GO terms associated with each gene in your list. (Remember, GO terms are standardized names for molecular processes, biological processes, and cellular locations.) The function getClusterProfilerGenes will let you see which genes in your gene list belong to each GO term.

If this sounds confusing, be patient - it should become more clear as you use the functions.

Just run the code below.

runClusterProfiler <- function (x) {  
 ids <- bitr( x$GeneID, "ENSEMBL", "ENTREZID", "org.Mm.eg.db" )  
 kegg <- enrichKEGG(ids$ENTREZID, "mmu", keyType="ncbi-geneid")  
 kegg@result$Description <- sub( " - Mus musculus \\(house mouse\\)", "", kegg@result$Description )  
 return(kegg)  
}  
  
getClusterProfilerGenes <- function (x, i) {  
 data.frame( x ) %>%  
 filter( str\_detect( Description, i ) ) %>%  
 pull( geneID ) %>%  
 strsplit( "/" ) %>%  
 unlist() %>%  
 bitr( fromType="ENTREZID", toType="SYMBOL", OrgDb="org.Mm.eg.db") %>%  
 pull( SYMBOL )  
}

Okay, that’s all the prep work we need to do! Now let’s move onto the actual analysis.

### Loading the data

For this analysis, we’re using the differential expression dataset that you used to make your volcano plot. If it’s loaded into your session, great! You don’t need to do anything. If you need to load it again, here’s the code and links from before. We’re loading the data that compares gene expression between the prefrontal cortex and striatum in the control mice.

prefront\_vs\_striatum\_control <- read\_csv("https://genomicseducation.org/data/mouse\_gutbrain\_de\_tissuetype\_in\_controlmice.csv")

Rows: 55421 Columns: 7  
── Column specification ────────────────────────────────────────────────────────  
Delimiter: ","  
chr (1): GeneID  
dbl (6): baseMean, log2FoldChange, lfcSE, stat, pvalue, padj  
  
ℹ Use `spec()` to retrieve the full column specification for this data.  
ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

Here are the URLs for all the possible comparisons you can examine with this dataset:

**Comparing gene expression between ASD and control mice**

Both brain regions: <https://genomicseducation.org/data/mouse_gutbrain_de_autismVcontrol.csv>

Prefrontal cortex only: <https://genomicseducation.org/data/mouse_gutbrain_de_autismVcontrol_in_prefrontalcortex.csv>

Striatum only: <https://genomicseducation.org/data/mouse_gutbrain_de_autismVcontrol_in_striatum.csv>

**Comparing gene expression between prefrontal cortex and striatum**

All mice: <https://genomicseducation.org/data/mouse_gutbrain_de_tissuetype.csv>

Only ASD mice: <https://genomicseducation.org/data/mouse_gutbrain_de_tissuetype_in_ASDmice.csv>

Only control mice: <https://genomicseducation.org/data/mouse_gutbrain_de_tissuetype_in_controlmice.csv>

### Filtering the data based on padj

In a gene set analysis, we’re interested in identifying the biological pathways that are likely to be different based on the gene expression differences between our two compared groups. We run these kinds of analyses on the genes with the lowest padj values in the dataset (because these are the genes that we feel confident are differentially expressed).

We use filter to create this gene list. You can set “padj” to whatever threshold you like.

sig\_gene\_list <- filter(prefront\_vs\_striatum\_control, padj <= 0.001)

### Visualizing the biological pathway GO terms in the gene set

Now we’re going to use the special functions you created earlier. First we want to use runClusterProfiler. This is the function that gathers all the GO terms associated with the genes in your gene list.

gene\_clusters <- runClusterProfiler(sig\_gene\_list)

'select()' returned 1:many mapping between keys and columns

Warning in bitr(x$GeneID, "ENSEMBL", "ENTREZID", "org.Mm.eg.db"): 14.49% of  
input gene IDs are fail to map...

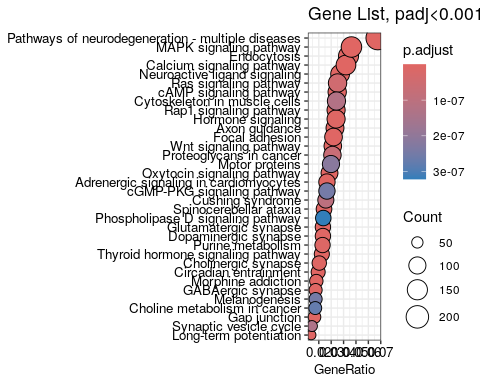
Reading KEGG annotation online: "https://rest.kegg.jp/link/mmu/pathway"...

Reading KEGG annotation online: "https://rest.kegg.jp/list/pathway/mmu"...

Reading KEGG annotation online: "https://rest.kegg.jp/conv/ncbi-geneid/mmu"...

While R can easily read the output from runClusterProfiler, it’s a little harder for us unless we make something called a dot plot.

dotplot(gene\_clusters, showCategory=34, title="Gene List, padj<0.001", font.size=10, label\_format = 50)



In the dot plot, the GO terms (mostly biological processes) are on the y-axis. The x-axis gives us an idea of how many genes in our gene list are found in each GO term category (GeneRatio is proportion of the differentially expressed genes that belong to a specific GO term or pathway). The dot plot also colors the dots based on the “padj” value. In this case, the red dots indicate lower (or more statistically certain) adjusted p-values.

The top GO term on the plot (which is the category with the greatest number of differentially expressed genes in it) is “Pathways of neurodegeneration - multiple diseases.” How interesting! Clearly the expression of neurodegeneration-associated genes differ between the two parts of the brain, even in control mice. Let’s take a closer look at what genes belong to this GO term category using the getClusterProfilerGenes function.

getClusterProfilerGenes(gene\_clusters, "Pathways of neurodegeneration - multiple diseases")

'select()' returned 1:1 mapping between keys and columns

[1] "Ppp3ca" "Wnt7b" "Prnp" "Camk2b" "Tubb2a" "Ptgs2"   
 [7] "Stx1a" "Grm5" "Mapk10" "Hras" "Itpr1" "Mapk8"   
 [13] "Lrrk2" "Pik3r4" "Grm1" "Wnt4" "Tubb4b" "Tuba1b"   
 [19] "Camk2a" "Kras" "Plcb1" "Trpc3" "Atf4" "Tubb5"   
 [25] "Optn" "Dnal4" "Dnah1" "Bdnf" "Tuba1c" "Ryr3"   
 [31] "Ryr1" "Bid" "Wnt9a" "Prkcb" "Tubb3" "Map1lc3a"  
 [37] "Gria3" "Mapk11" "Snca" "Sncaip" "Chrm3" "Mapk1"   
 [43] "Wipi1" "Psen1" "Camk2d" "Atxn1l" "Wnt2" "Apc"   
 [49] "Gsk3b" "Ppp3cc" "Casp3" "Capn2" "Wnt5a" "Ube2l6"   
 [55] "Dnah7c" "Sigmar1" "Camk2g" "Calm3" "Dnah5" "Ube2j1"   
 [61] "Gpr37" "Tuba1a" "Nefl" "Ndufs3" "Tuba8" "Tuba4a"   
 [67] "Sptbn2" "Rab1a" "Lrp6" "Fzd5" "Gpx1" "Ppp3r1"   
 [73] "Fzd1" "Tubb2b" "Zfyve1" "Atxn1" "Plcb4" "Psmc3"   
 [79] "Wnt10a" "Actr10" "Atg14" "Cacna1c" "Map2k7" "Prkcg"   
 [85] "Gpx6" "Wnt3" "Klc1" "Fzd2" "Cyct" "Gria1"   
 [91] "Adrm1" "Gria4" "Atp2a2" "Pdyn" "Gnaq" "Bcl2"   
 [97] "Mapk12" "Ndufa1" "Ndufb10" "Actr1b" "Dnai2" "Rac1"   
[103] "Calm1" "Psmc5" "Ppid" "Map3k5" "Cdk5" "Plcb3"   
[109] "Dnal1" "Ube2g2" "Sdhc" "Cyc1" "Sqstm1" "Fus"   
[115] "Kif5b" "Atg13" "Hip1" "Grin1" "Ube2g1" "Slc25a5"   
[121] "Map1lc3b" "Psen2" "Smcr8" "Nos1" "Psmd12" "Pik3c3"   
[127] "Prkn" "Ndufv1" "Prkca" "Plcb2" "Fzd9" "Psmd4"   
[133] "Fzd8" "Ctnnb1" "Atxn3" "Ift57" "Wnt8b" "Dlg4"   
[139] "Klc2" "Map2k3" "Wnt6" "Atf6" "Fig4" "Grin2a"   
[145] "Dctn2" "Psmc4" "Psmd3" "Klc3" "Rela" "Dnah14"   
[151] "Apc2" "Ndufc2" "Dnah10" "Psmb7" "Ryr2" "Psma3"   
[157] "Grin2c" "Axin1" "Gpx3" "Atp5po" "Ube2l3" "Raf1"   
[163] "Nefh" "Vdac3" "Psmb4" "Dnah6" "Csnk2b" "Csnk1e"   
[169] "Ndufs4" "Fzd3" "Ndufb6" "Dctn4" "Chmp2b" "Bad"   
[175] "Psmd8" "Nfkb1" "Uqcrc2" "Dvl1" "Cox7a2" "Nox1"   
[181] "Braf" "App" "Ubc" "Psmd13" "Vapb" "Mapk9"   
[187] "Ambra1" "Dnali1" "Psmb5" "Atp5f1a" "Slc25a4" "Dnah3"   
[193] "Kif5a" "Actr1a" "Vdac1" "Psma5" "Ndufa10" "Sem1"   
[199] "Cdk5r1" "Uchl1" "Htra2" "Dnah12" "Wnt7a" "Rab39b"   
[205] "Sdha" "Uba1" "Cox8a" "Ulk1" "Atp5mc2" "Sdhb"   
[211] "Dctn1" "Rab5a" "Vdac2" "Wnt1" "Chrm5" "Uqcrfs1"   
[217] "Cybb" "Mfn1" "Ndufa2" "Cox7a1" "Gpx8" "Atg101"   
[223] "Vcp" "Psmd11" "Uqcrh" "Dnah17" "Uqcr10" "Psmc6"   
[229] "Dnah2"

There are a lot of genes in this category! At this point, researchers might pick a some genes from this list and go back to the MGI database to research it for future work.

[1] G. Sharon *et al.*, “Human gut microbiota from autism spectrum disorder promote behavioral symptoms in mice,” *Cell*, vol. 177, no. 6, pp. 1600–1618, 2019, Available: <https://pubmed.ncbi.nlm.nih.gov/31150625/>

[2] C. Hu, “Of mice and model organisms.” 2024. Available: <https://www.cshl.edu/of-mice-and-model-organisms/>

[3] Foundation for Biomedical Research, “Nobel prizes.” 2025. Available: <https://fbresearch.org/medical-advances/nobel-prizes/>