ME.440.825 Quantitative Neurogenomics

Problem Set 1

Part 1: basic bash scripting

First, download this folder of genome sequence files. Then, answer each question using Bash scripting.

You'll need to have your folder of sequence files in your working directory. Print your working directory and its contents.

```
echo Working directory:
pwd
echo $'\n'Contents of working directory:
ls
```

```
## Working directory:
## /Users/acspiegel/Documents/Hopkins/Neuroscience/Courses/Quant_mol_neuro_2022/modules/module_1/pset
##
## Contents of working directory:
## Kcna1_sequence.txt
## Yao2021_ionChannels_expMat_genesByCells.rds
## Yao2021_ionChannels_pData.rds
## findSeqMotif.sh
## ion_channel_sequences
## pset1_introToBashAndR.pdf
## pset1_introToBashAndR.rmd
## utils.R
```

We'll start with Kcna1.fa, a FASTA file that contains the genome sequence of a mouse voltage-gated potassium channel. The first line contains a carat (">"), followed by a unique sequence identifier. The actual sequence starts on the next line. See for yourself by printing the first three lines of the file. Note that FASTA files can contain multiple sequences, but this one has just one.

```
head -3 ion_channel_sequences/Kcna1.fa
```

```
## >ref|NM_010595.3|:1-8970 Mus musculus potassium voltage-gated channel, shaker-related subfamily, meml
## GGGGGCTCCTCAGAGGCTCCGCAGCGGTGGAAGGACTGGAGCTGCTGGCTGCCTCCTCCGGTGCAGCCTG
## TATCCAGGTGCAGCGGCACTGGGGACGCGGTGCATATCCCTTGCTCAGACTGCCACTGTGACCCTTGCGC
```

Print just the sequence into a new file called Kcna1_sequence.txt

```
tail -n +2 ion_channel_sequences/Kcna1.fa > Kcna1_sequence.txt
```

How many lines are in Kcna1_sequence.txt? How many characters are in the file? If you subtract the number of lines from the number of characters, you should get the number of nucleotides in the Kcna1 gene. Why? Answer in a comment.

```
echo Number of characters:
wc -m Kcna1_sequence.txt

echo $'\n'Number of lines:
wc -l Kcna1_sequence.txt

# The wc command counts all the characters in the file, including newline characters.
```

```
## Number of characters:
##     9099 Kcna1_sequence.txt
##
## Number of lines:
```

```
##      129 Kcna1_sequence.txt
```

Count the number of times each nucleotide appears in the Kcna1 gene.

```
echo A count:
grep -o "A" Kcna1_sequence.txt | wc -l
echo G count:
grep -o "G" Kcna1_sequence.txt | wc -l
echo C count:
grep -o "C" Kcna1_sequence.txt | wc -l
echo T count:
grep -o "T" Kcna1_sequence.txt | wc -l
```

```
## A count:
##     2267
## G count:
##     2166
## C count:
##     2207
## T count:
##     2330
```

Many voltage-gated potassium channels have a signature selectivity filter with the amino acid sequence TVGYG. Confirm that the Kcna1 gene has a sequence that would encode these amino acids. Your first step should be to remove newline characters from Kcna1_sequence.txt.

```
cat Kcna1_sequence.txt | tr -d "\n" |
grep -o "AC[TCAG]GT[TCAG]GG[TCAG]TA[TC]GG[TCAG]" | wc -l
```

```
##        1
```

Using *your favorite text editor*, write a bash script that takes a FASTA file (with a single sequence) and a target sequence motif as input and outputs the number of times that the motif appears in the FASTA file. You will use it here to determine which of the FASTA files in the ion_channel_sequences folder are likely to encode a voltage-gated potassium channel.

After writing your script, check its file permissions and make it executable

```
ls -l findSeqMotif.sh
#chmod +x findSeqMotif.sh
```

```
## -rwxr-xr-x@ 1 acspiegel  staff  240 Aug 28 22:46 findSeqMotif.sh
```

Use a loop to check whether each of the FASTA files in the ion_channel_sequences folder contains the TVGYG motif. Print the names of just the files that do contain the motif.

```
tvgyg="AC[TCAG]GT[TCAG]GG[TCAG]TA[TC]GG[TCAG]"

echo voltage-gated potassium channels:

for filename in ion_channel_sequences/*.fa
do
  motifCount=$(./findSeqMotif.sh $filename $tvgyg)
  if [ $motifCount -gt 0 ]
  then
    echo $filename
  fi
done
```

```
## voltage-gated potassium channels:
```

```
## ion_channel_sequences/Kcna1.fa
## ion_channel_sequences/Kcna2.fa
## ion_channel_sequences/Kcnb1.fa
```

Part 1: basic R

Get your dependencies. For this section, you will need the ggplot2 package. If you haven't already, you can get it by installing the whole tidyverse as suggested here: https://ggplot2.tidyverse.org/. You should also download and read the file *utils.R*, which loads your dependencies and contains some supplementary functions. You will want to look at utils.R yourself for guidance on using these functions. Lastly, make sure you download *Yao2021_ionChannels_expMat_genesByCells.rds* and *Yao2021_ionChannels_pData.rds*.

```
# Clear all variables from the environment
rm(list = ls())

# read utils.R here
source("utils.R")
```

Now load Yao2021_ionChannels_expMat_genesByCells.rds, a genes x cells matrix that contains gene expression data from publicly available single-cell RNA-sequencing data performed in motor cortex (Yao et al. 2021, https://doi.org/10.1038/s41586-021-03500-8). Specifically, this matrix includes the expression of the ion channels that you looked at in Part 1 of this problem set.

```
# load the expression data here and save it as a variable
expMat <- readRDS("Yao2021_ionChannels_expMat_genesByCells.rds")
```

The value in each entry of corresponds to the counts of a specific gene detected for a single cell. Therefore, each row has all of the expression values for a particular gene, and each column as all the the expression values for a particular cell. How many genes are included in this matrix? How many cells?

```
print("Number of genes:")
```

```
## [1] "Number of genes:"
```

```
dim(expMat)[1]
```

```
## [1] 14
```

```
print("Number of cells:")
```

```
## [1] "Number of cells:"
```

```
dim(expMat)[2]
```

```
## [1] 71183
```

The row names of the matrix are the short names of the genes. The column names are IDs for each cell. Confirm that this is the case by printing the first 5 row names and the first 5 column names.

```
print("Row names:")
```

```
## [1] "Row names:"
```

```
rownames(expMat)[1:5]
```

```
## [1] "Kcnj10"  "Kcnk2"   "Scn3a"   "Scn9a"   "Slc12a5"
```

```
print("Column names")
```

```
## [1] "Column names"
```

```
colnames(expMat)[1:5]
```

```
## [1] "AAACCCAAGCTTCATG-1L8TX_181211_01_G12"
## [2] "AAACCCAAGTGAGGTC-1L8TX_181211_01_G12"
## [3] "AAACCCACACCAGCCA-1L8TX_181211_01_G12"
## [4] "AAACCCAGTGAACGGT-1L8TX_181211_01_G12"
## [5] "AAACCCAGTGGCATCC-1L8TX_181211_01_G12"
```

What is the most common gene expression value in the matrix? Hint: The built-in R function "mode" will not help you here (feel free to see what it does though!). There's a function in utils.R that you will find useful. Is the answer what you expected?

```
# find the most common gene expression value
getMode(expMat)
```

```
## [1] 0
```

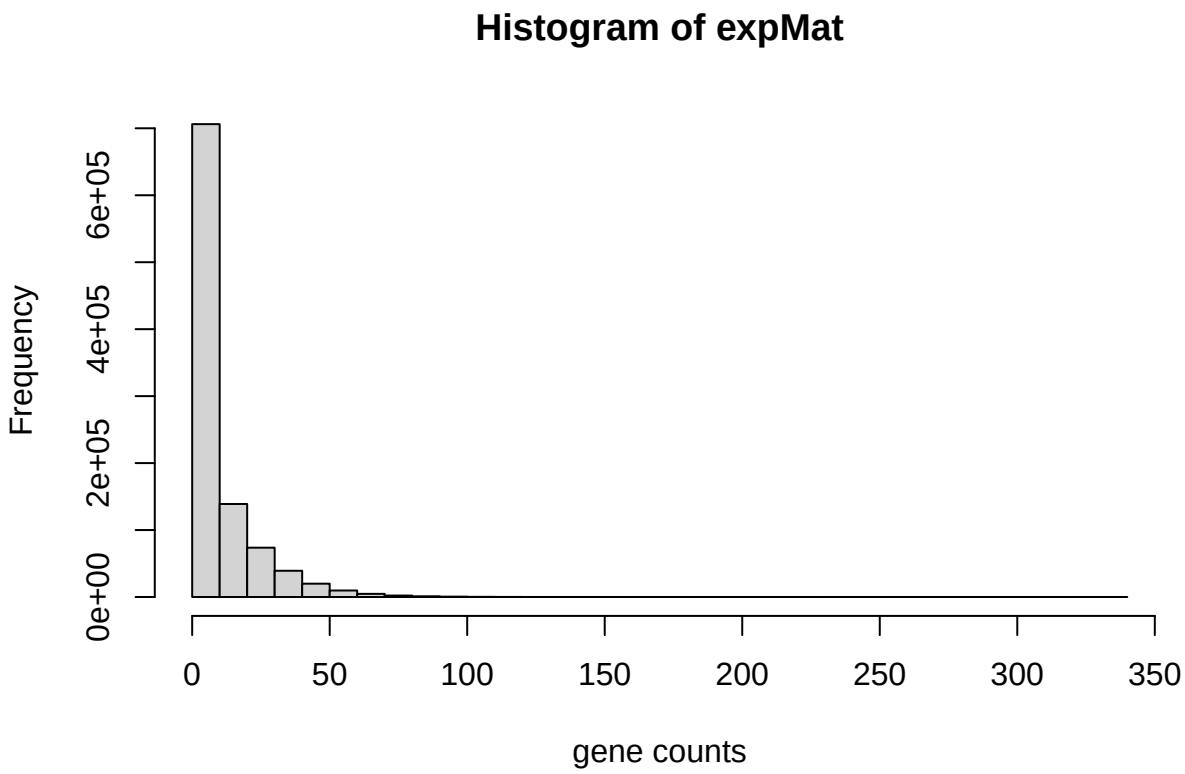
That most common value makes up what percentage of all the expression values in the matrix?

```
# find the most common gene expression value
expMat_mode <- getMode(expMat)
sum(expMat == expMat_mode)/length(expMat) * 100
```

```
## [1] 36.2433
```

To analyze this data, it is important to know whether or not it is normally distributed. Plot a histogram of all the expression values in the matrix. Ensure that the axes make sense. What does this histogram tell us about the single-cell RNA-seq count data?

```
hist(expMat, 40, xlab = "gene counts")
```



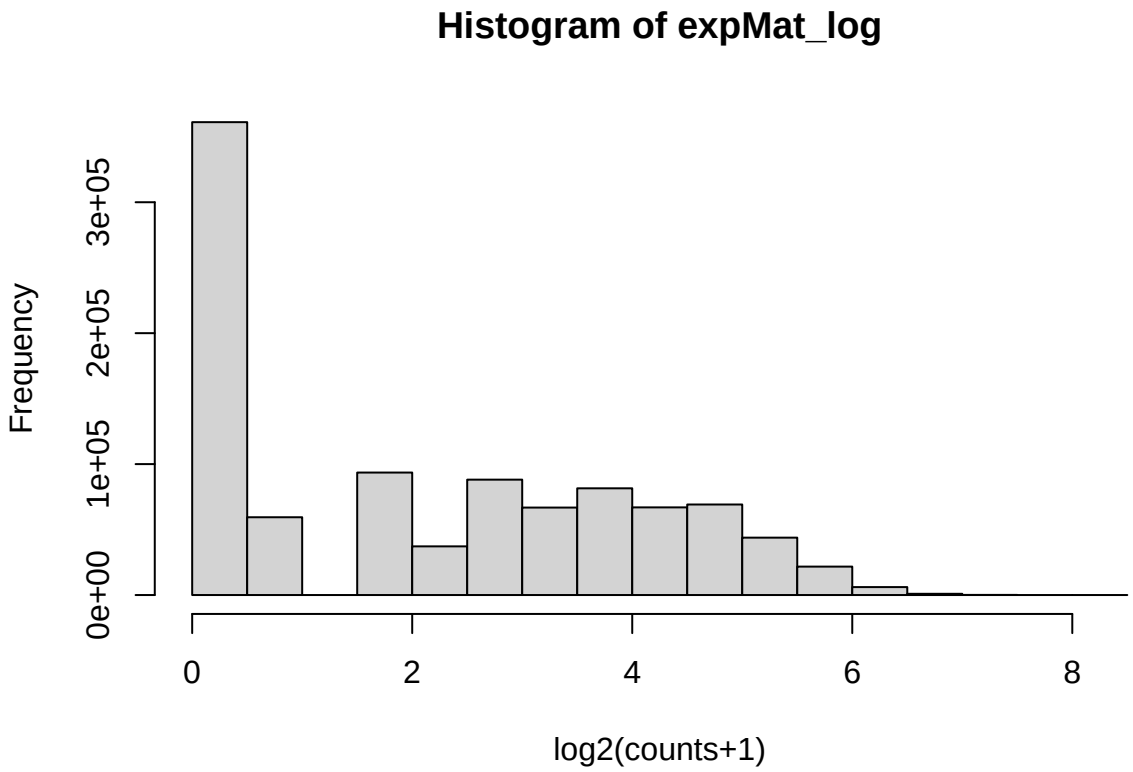

```
# Many genes are very lowly expressed. There is a long
# right tail, corresponding to the few genes with high
```

```
# expression values.
```

Hopefully by now, you've deduced that RNA-seq count data is not normally distributed. It can sometimes be helpful to log transform the data, so that it becomes closer to a normal distribution. Log transform the data using base 2 and a pseudocount of 1. Save the result as a new variable and plot a histogram. Again, check that your axes make sense.

```
# log transform the gene counts
expMat_log <- log2(expMat + 1)

hist(expMat_log, 20, xlab = "log2(counts+1)")
```



Now, we can look at what types of cells express of these ion channel genes. Load Yao2021_ionChannels_pData.rds, which contains a dataframe with information about each of the cells in the counts matrix. Print the columns included in this dataframe.

```
# load the dataframe here and save it as a variable
cellInfo <- readRDS("Yao2021_ionChannels_pData.rds")

print("Dataframe columns:")
```

```
## [1] "Dataframe columns:"
```

```
colnames(cellInfo)
```

```
## [1] "cellID"       "cellClass"    "cellSubclass"
```

What are the possible values in the column "cellClass"? What are the possible values in the column "cellSubclass"?

```
print("Unique values of cellClass")
```

```
## [1] "Unique values of cellClass"
```

```
unique(cellInfo$cellClass)
```

```
## [1] "Glutamatergic" "GABAergic"     "Non-Neuronal"
```

```
print("Unique values of cellSubclass")
```

```
## [1] "Unique values of cellSubclass"
```

```
unique(cellInfo$cellSubclass)
```

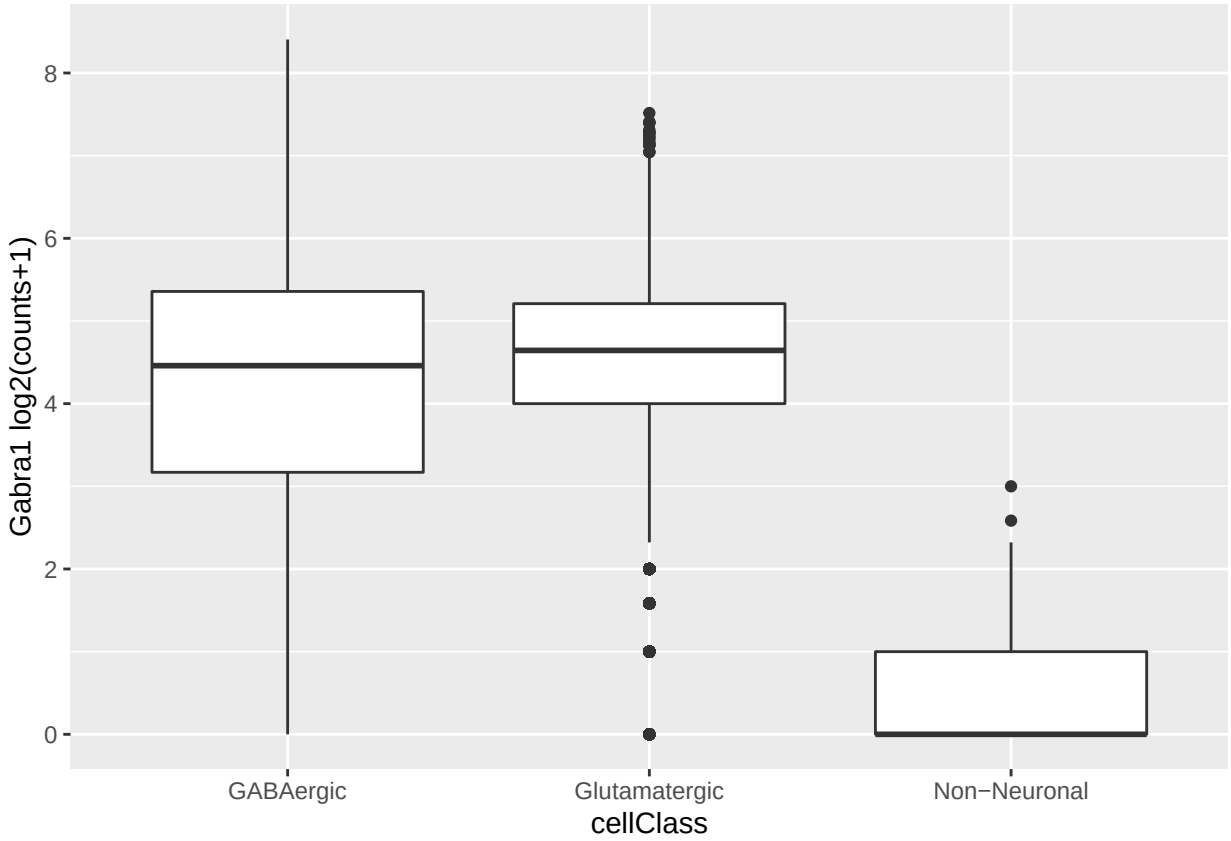
```
##  [1] "L5 IT"      "Vip"        "L6 IT"      "L6 CT"      "L2/3 IT"
##  [6] "Oligo"      "Sst"        "Lamp5"      "L5/6 NP"    "Macrophage"
## [11] "Astro"      "OPC"        "L6b"        "VLMC"       "Endo"
## [16] "Sncg"       "L5 ET"      "L6 IT Car3" "SMC"        "Pvalb"
```

The values in the column "cellID" correspond to the column names of the counts matrix. Merge this dataframe with the transpose of the log-transformed counts matrix to create a new dataframe that includes gene expression information for each cell. This new dataframe should have a new column for each gene.

```
# merge the cell info with the gene expression info
df_ionChannels <- merge(cellInfo, t(expMat_log), by.x = "cellID",
    by.y = "row.names")
```

Now use the function makeBP included in Utils.R to create box plots of gene expression by cell class for your gene of choice. Your new dataframe will be the first argument and your gene of choice will be the second argument.

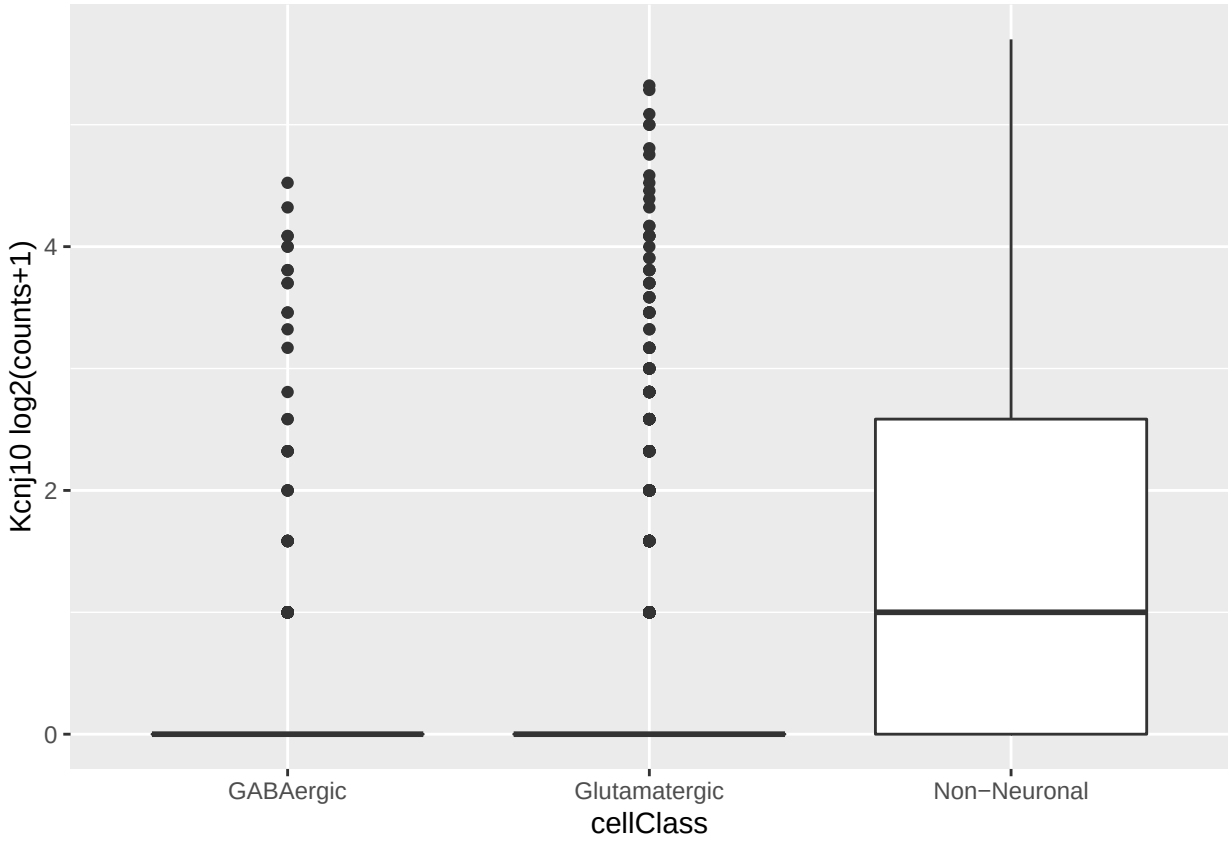
```
makeBP(df_ionChannels, "Gabra1")
```

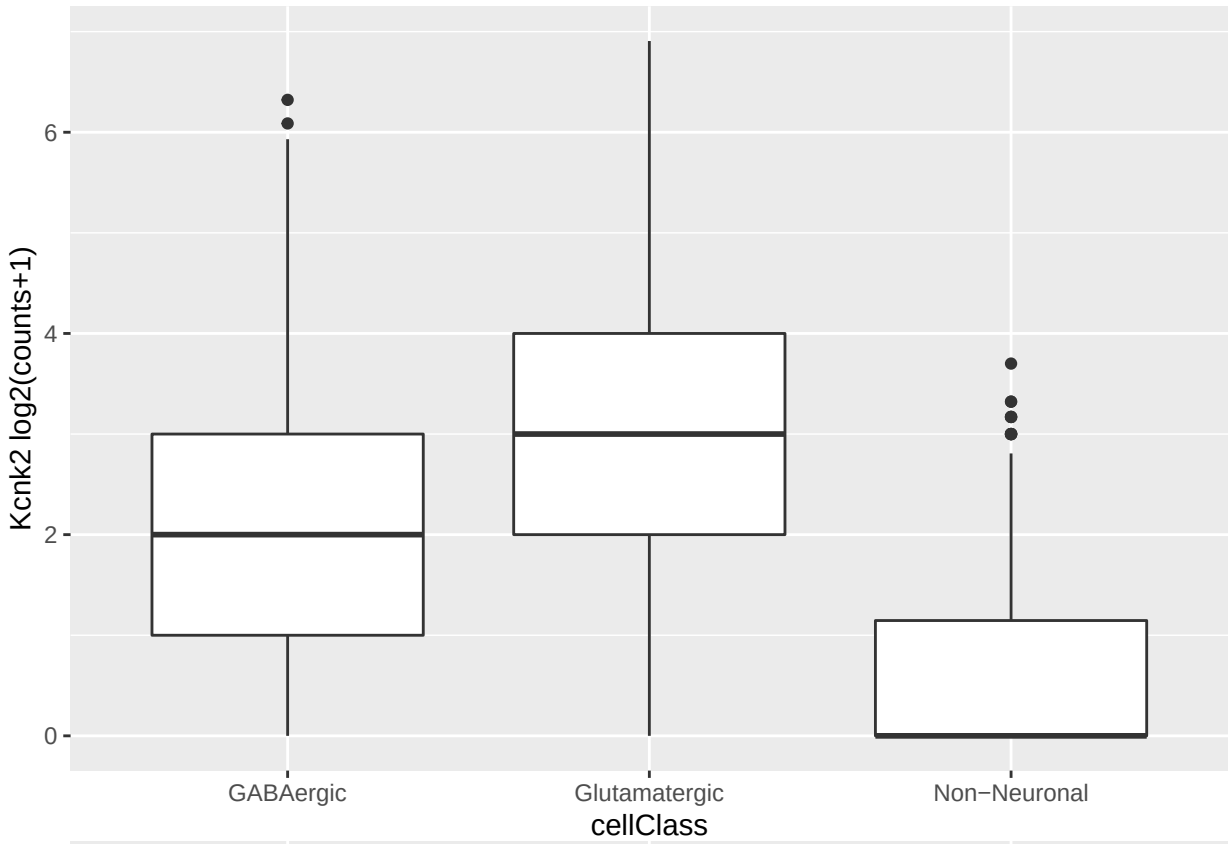


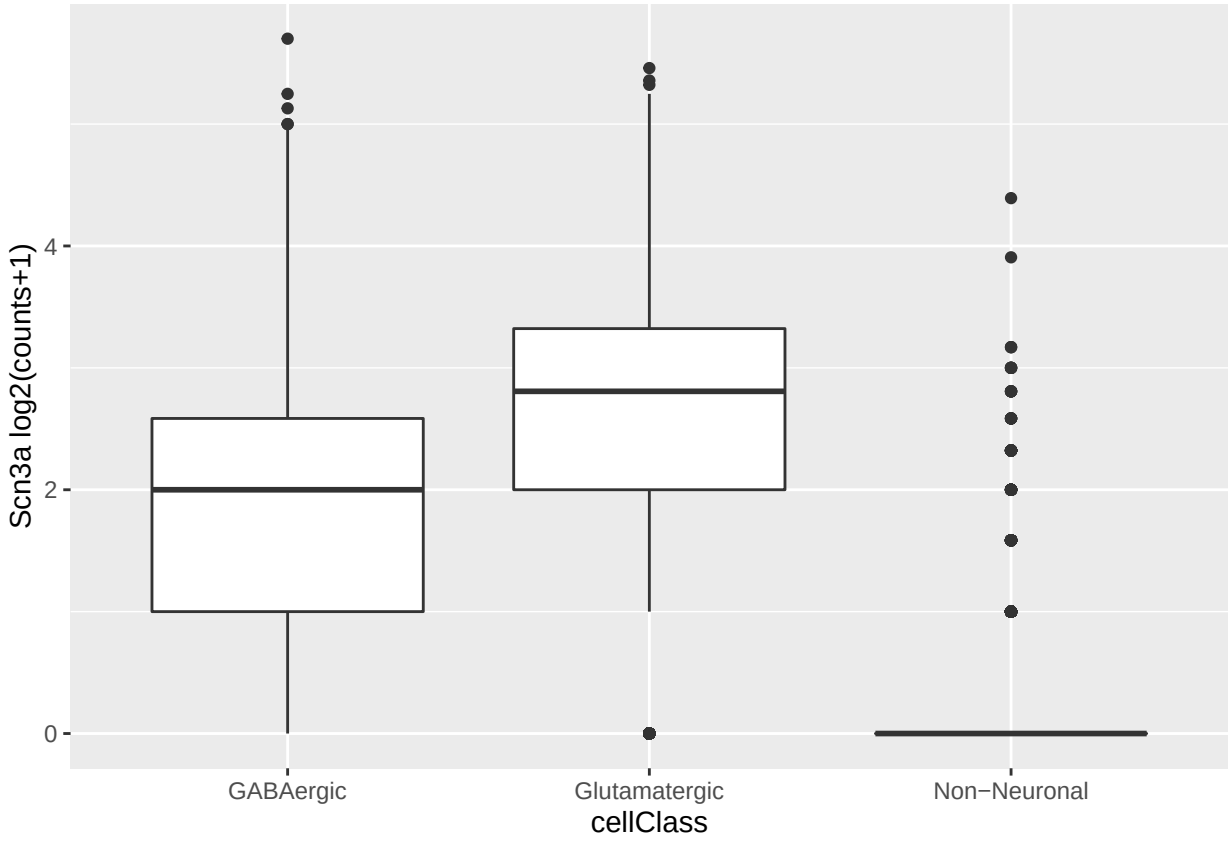
One of the ion channels is expressed more highly in glia than in neurons. To find it, use a loop to create box

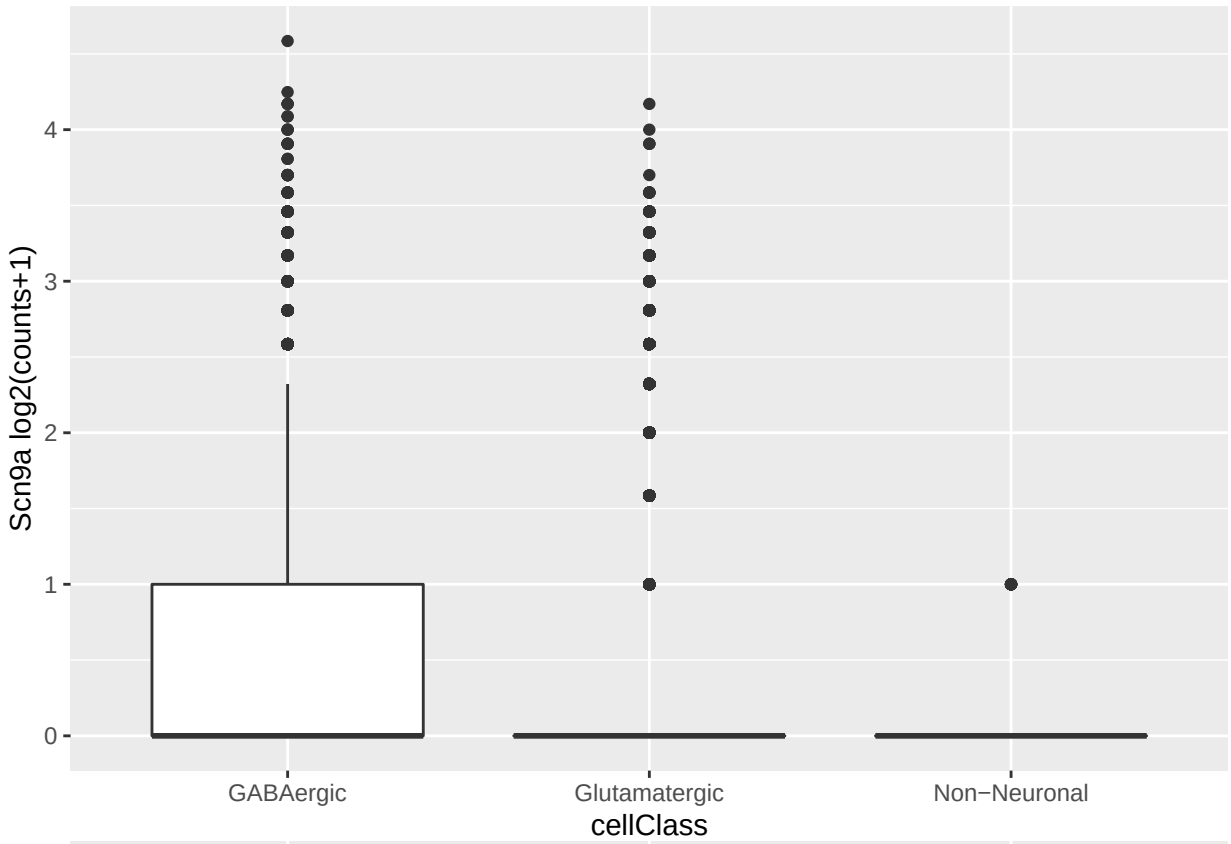
plots for each gene. Note: In this case, it is particularly obvious, but eventually we will want to quantify observations like this.

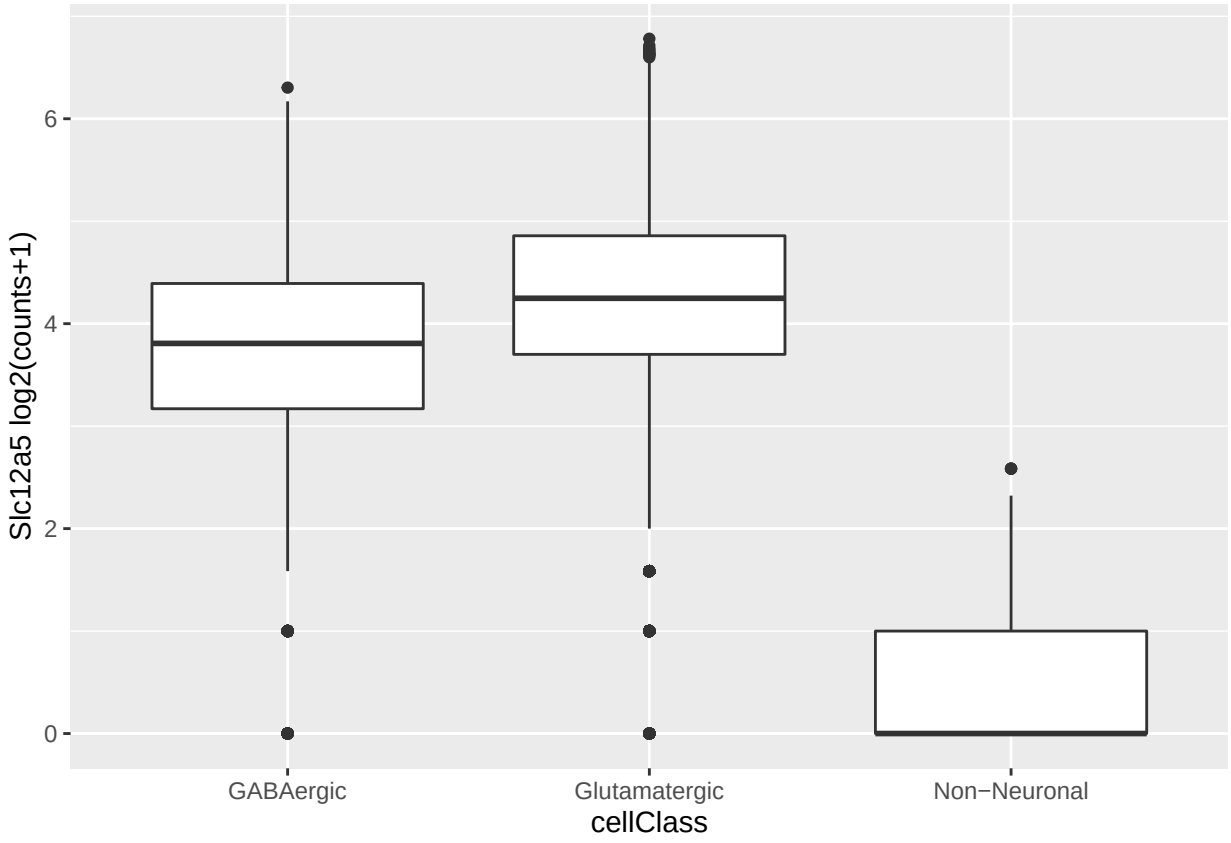
```
for (geneName in rownames(expMat)) {
    makeBP(df_ionChannels, geneName)
}
```

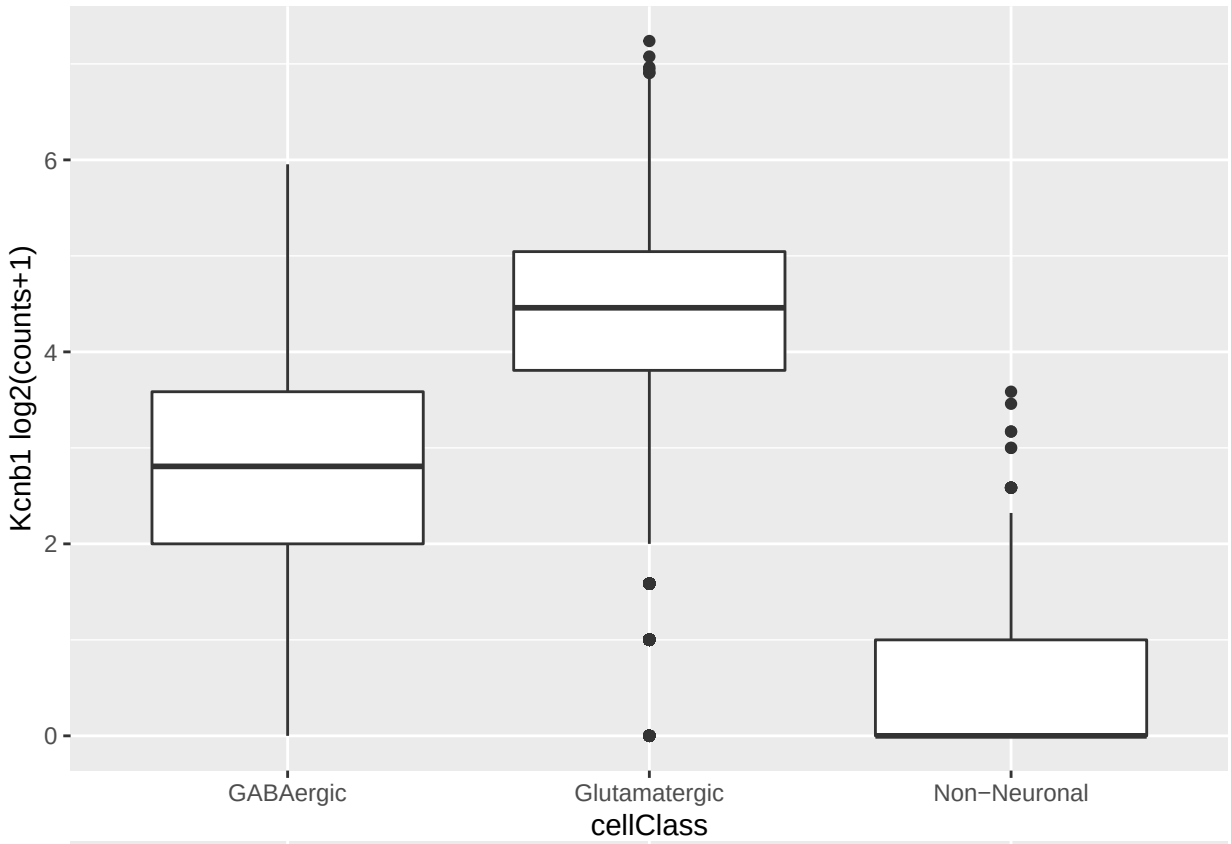


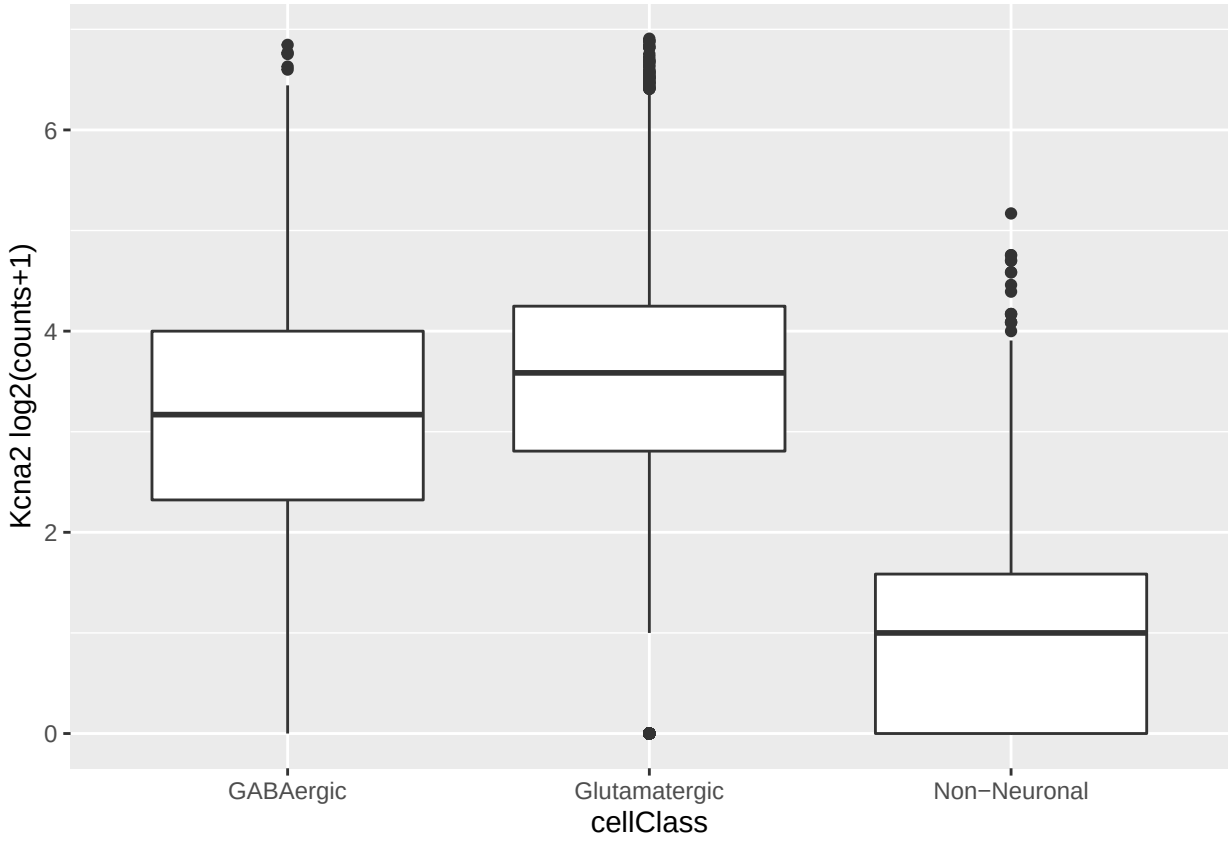

Kcnk2 log2(counts+1)
6
4
2
0
GABAergic
Glutamatergic
Non−Neuronal
cellClass

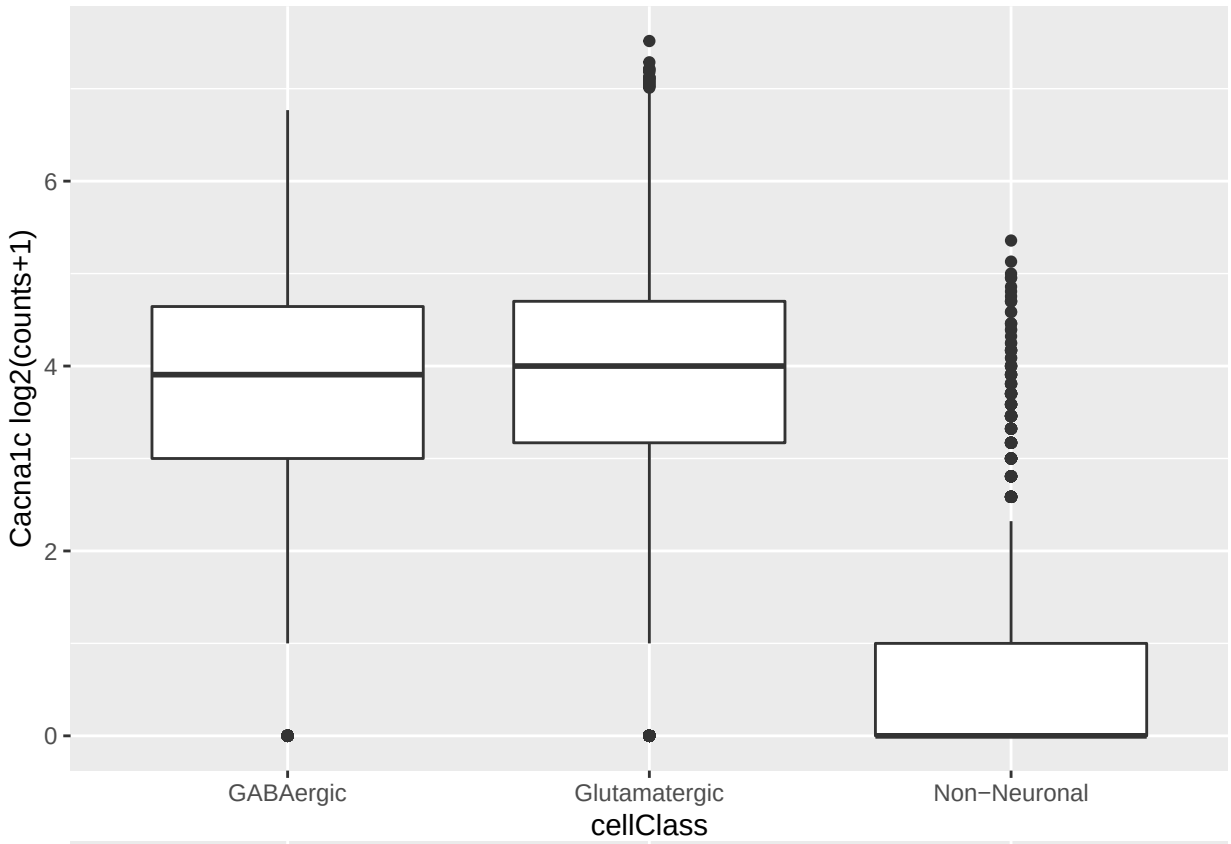

Scn3a log2(counts+1)
4
2
0
GABAergic
Glutamatergic
Non−Neuronal
cellClass

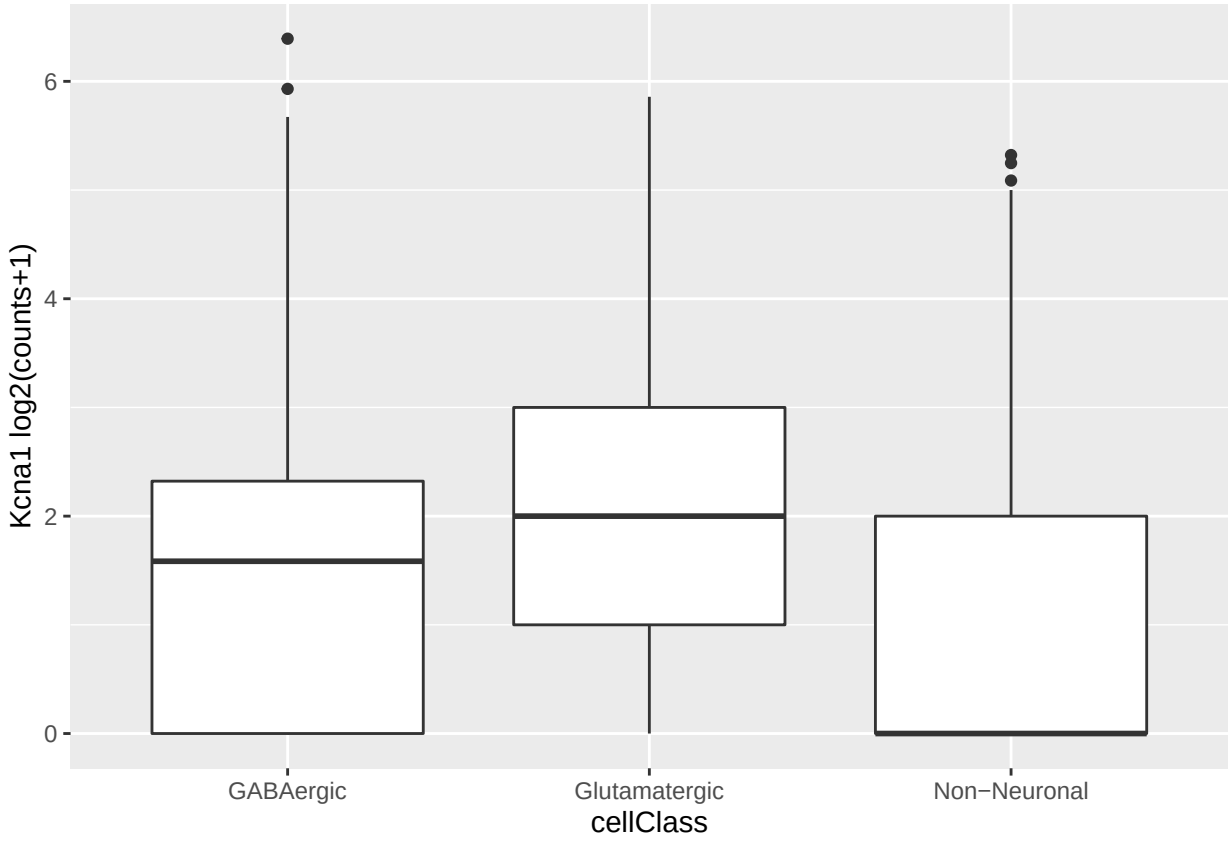

Scn9a log2(counts+1)
0
1
2
3
4
GABAergic
Glutamatergic
Non−Neuronal
cellClass

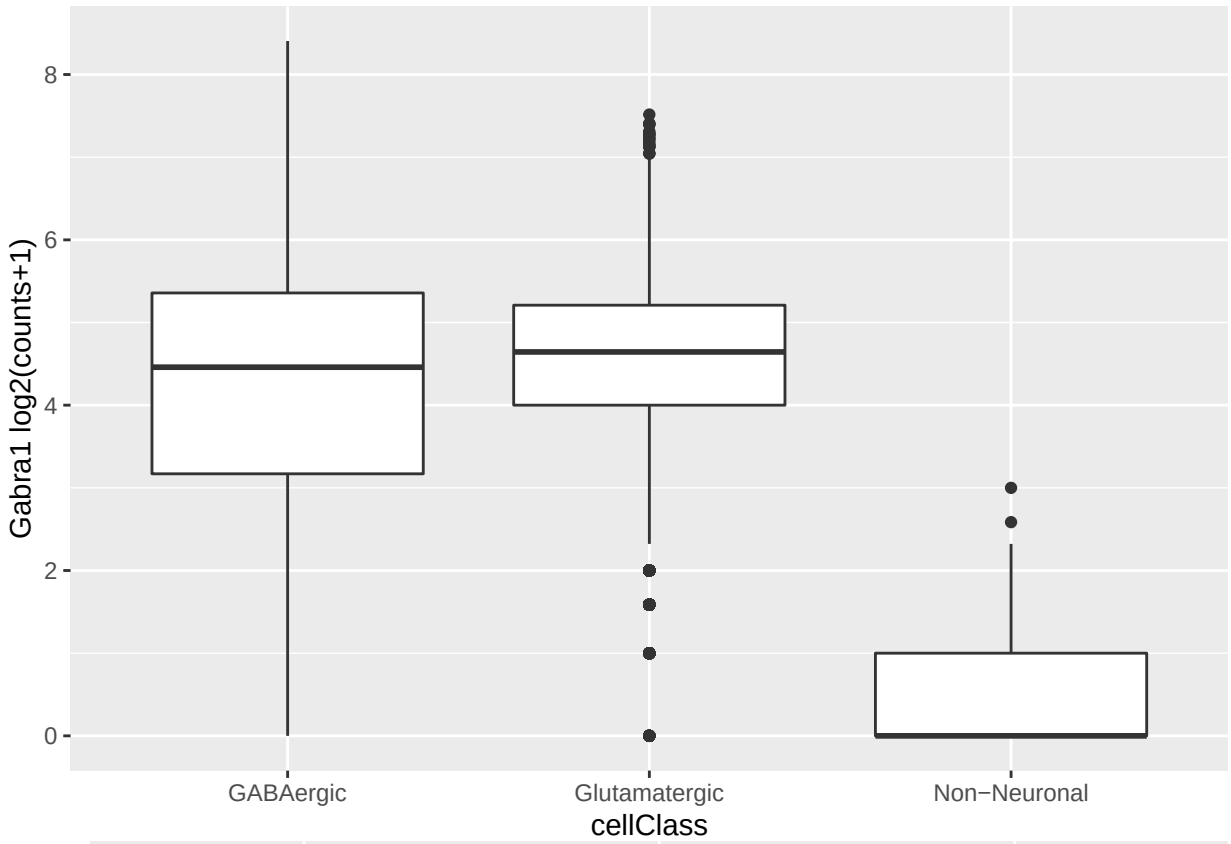

Slc12a5 log2(counts+1)
0
2
4
6
GABAergic
Glutamatergic
Non−Neuronal
cellClass

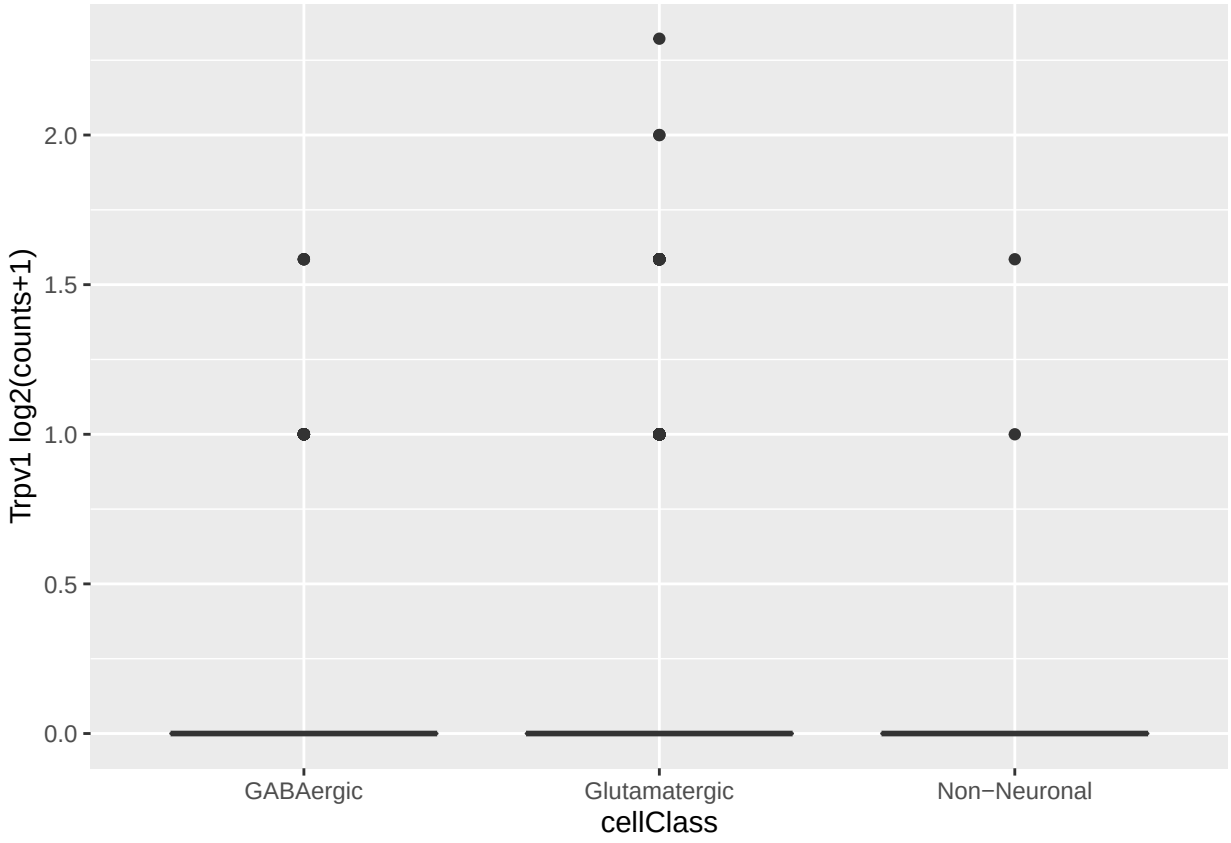

Kcnb1 log2(counts+1)
6
4
2
0
GABAergic
Glutamatergic
Non−Neuronal
cellClass

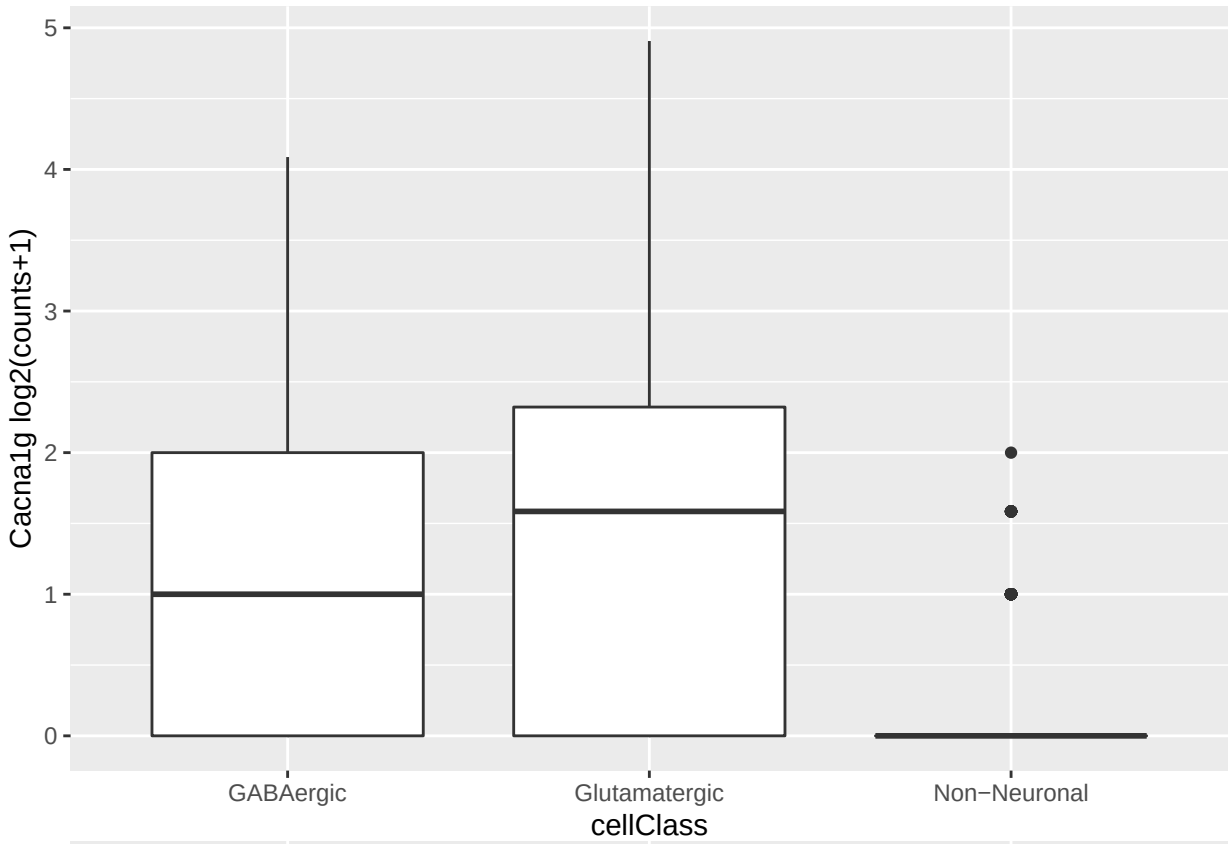

Kcna2 log2(counts+1)
6
4
2
0
GABAergic
Glutamatergic
Non−Neuronal
cellClass

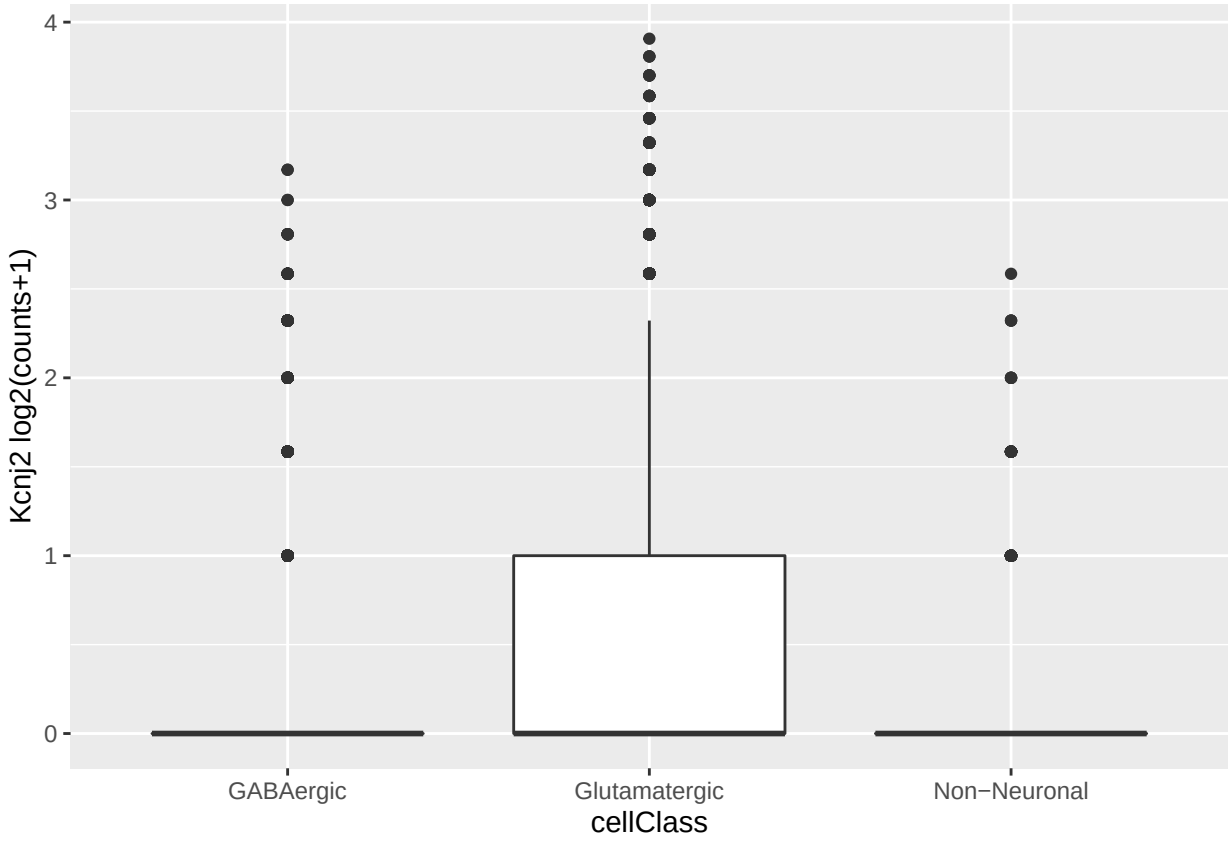

Cacna1c log2(counts+1)
6
4
2
0
GABAergic
Glutamatergic
Non−Neuronal
cellClass

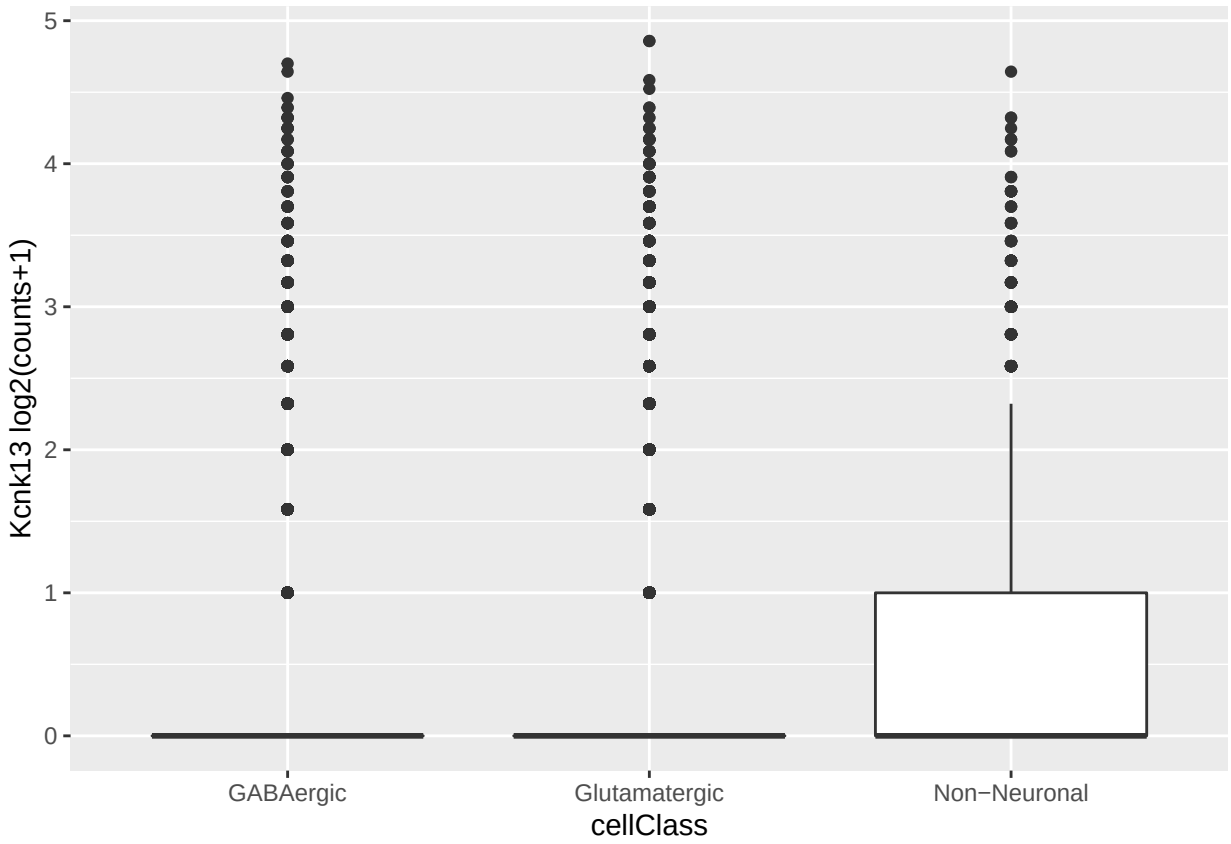

Kcna1 log2(counts+1)
6
4
2
0
GABAergic
Glutamatergic
Non−Neuronal
cellClass


Gabra1 log2(counts+1)
8
6
4
2
0
GABAergic
Glutamatergic
Non−Neuronal
cellClass


Trpv1 log2(counts+1)
2.0
1.5
1.0
0.5
0.0
GABAergic
Glutamatergic
Non−Neuronal
cellClass


Cacna1g log2(counts+1)
5
4
3
2
1
0
GABAergic
Glutamatergic
Non−Neuronal
cellClass


Kcnj2 log2(counts+1)
4
3
2
1
0
GABAergic
Glutamatergic
Non−Neuronal
cellClass




```
# Answer: Kcnj10, which encodes Kir 4.1, an inwardly
# rectifying potassium channel thought to mediate potassium
# buffering by astrocytes.
```

End of Problem Set