Week 2 - Introduction to RNAseq

Sidharth Jain

5/26/2020

Package list: (Use the library function to load in additional libraries)

If you're missing packages, run install.packages("packagename").

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.1     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Taking a look at CCLE RNAseq data

For this week's exercise, we're going to start digging into RNAseq expression data. The data we'll be looking at has gone through several stages of processing. Briefly, RNA is collected from each sample, usually with technical replicates, and fragmented (read more about RNAseq experiment preparation here). A sequencing machine will produce fixed-length reads from the prepared RNA library, and identify the sequences. This information is stored in a file format called `.fastq`, which captures both the actual nucleotide sequence, as well as the quality of the read from each nucleotide. Sequences are then aligned to the genome to identify which gene they originated from, and the number of alignments to each gene creates a "gene count", which represents how expressed each gene was in the cell.

The counts are then normalized by the length of the gene/region of the genome they emerged from. Currently, we used either RPKM, FPKM, or TPM to present normalized RNAseq readouts. You can read more about the similarities/differences here, but just remember that we now have a standard way of comparing one gene's expression to another. Using this data, we can now compare expression of genes within a single sample. To compare between different samples, we'll also need to normalize between samples using more complicated normalization techniques.

We'll be diving into RNAseq experiments in much greater depth later on, but for now, we just need to understand how the data we're working with was acquired.

Let's load in the data. This week, we'll be using slightly different CCLE metadata compared to last week. We'll also be using the mutation and expression data (which has been reported in TPM).

Loading the data

```
# Unlike last time where we used the read.delim() function of base R,
# we'll be using the readr package's read_csv() function.
CCLE_metadata <- read_csv("sample_info.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   COSMICID = col_double(),
##   Achilles_n_replicates = col_double(),
##   cell_line_NNMD = col_double(),
##   age = col_double(),
##   depmap_public_comments = col_logical()
## )
```

```
## See spec(...) for full column specifications.
```

```
## Warning: 11 parsing failures.
##  row                    col           expected                                      actual
## 1105 depmap_public_comments 1/0/T/F/TRUE/FALSE Adherent version of CCLFPEDS0001T          'sample_in
## 1166 depmap_public_comments 1/0/T/F/TRUE/FALSE SV40+TERT immortalized kidney cell line    'sample_in
## 1454 depmap_public_comments 1/0/T/F/TRUE/FALSE Drug resistance: Dabrafenib and Trametinib 'sample_in
## 1455 depmap_public_comments 1/0/T/F/TRUE/FALSE Drug resistance: SCH772984                 'sample_in
## 1456 depmap_public_comments 1/0/T/F/TRUE/FALSE Drug resistance: Dabrafenib and Trametinib 'sample_in
## .... ..................... .................. ..................................... .........
## See problems(...) for more details.
```

```
# These next lines might take a little while to run - try not to run them too many times!
CCLE_expression <- read_csv("CCLE_expression.csv")
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   X1 = col_character()
## )
## See spec(...) for full column specifications.
```

```
CCLE_mutations <- read_delim("CCLE_mutations.csv",
    "\t", escape_double = FALSE, trim_ws = TRUE)
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   Entrez_Gene_Id = col_double(),
##   NCBI_Build = col_double(),
##   Chromosome = col_double(),
##   Start_position = col_double(),
```

```
##   End_position = col_double(),
##   isDeleterious = col_logical(),
##   isTCGAhotspot = col_logical(),
##   TCGAhsCnt = col_double(),
##   isCOSMIChotspot = col_logical(),
##   COSMIChsCnt = col_double(),
##   ExAC_AF = col_double(),
##   HC_AC = col_time(format = ""),
##   RD_AC = col_logical()
## )
## See spec(...) for full column specifications.

## Warning: 121087 parsing failures.
##  row   col   expected actual                file
## 1007 HC_AC valid date 69:116 'CCLE_mutations.csv'
## 1009 HC_AC valid date 30:113 'CCLE_mutations.csv'
## 1011 HC_AC time like  3:5    'CCLE_mutations.csv'
## 1018 HC_AC valid date 40:170 'CCLE_mutations.csv'
## 1019 HC_AC valid date 85:24  'CCLE_mutations.csv'
## .... ..... .......... ...... ....................
## See problems(...) for more details.
```

The expression data has cell lines in the first column (with an ACH ID), and gene expression in the other columns.

The mutation data is complicated, and we won't really worry too much about it right now, except to isolate mutations in the gene KRAS, as done below.

```
# Our goal here is to extract the relevant parts from the mutation data to merge it with the CCLE_metad

kras_mutations <- CCLE_mutations %>%
  # Filter for only KRAS mutations
  filter(Hugo_Symbol=="KRAS") %>%
  # Select for the columns we need to merge our data
  select(Hugo_Symbol, DepMap_ID, Protein_Change, Codon_Change)

# To make our lives easier, we're going to merge the CCLE_metadata with the kras_mutations data
CCLE_metadata <- merge(CCLE_metadata, kras_mutations, all.x = T)
```

Exploring the data

First, let's look at how many rows and columns are in the RNAseq data.

```
# Get the number of rows and number of columns separately
nrow(CCLE_expression)
```

```
## [1] 1305
```

```
ncol(CCLE_expression)
```

```
## [1] 19145
```

```
# Alternatively, use the dim() function to get a printout of 2 numbers
# Rows, columns
dim(CCLE_expression)
```

```
## [1]  1305 19145
```

So from this, it looks like we have 1,305 columns, and 19,145 rows. If you had guess, which one represents the number of cell lines, and which represents the number of genes?

As you might have guessed, there are under ~1400 cell lines, so each column represents a cell line with RNAseq data (not all cell lines have data!).And there are 19,145 genes being reported here. That would mean that we would have 24984225 different points of data.

Let's take a peek at the data to see what we're actually working with.

```
CCLE_expression[1:3,1:3]
```

X1	TSPAN6 (7105)	TNMD (64102)
ACH-001097	0.000000	0
ACH-001636	0.000000	0
ACH-001804	4.934988	0

Why tidy data?

When we encounter a new dataset, we want to first try to make our data *tidy*. What is tidy data? Generally, for tidy data, we want each row to consist of an observation, a column to describe a type of variable (or key), and value that matches with that variable (or value). We can use `gather` and `spread` to make our data a more tidy format. Tidy data is useful in that it becomes easier to take pieces of data that would be more useful. We can also spread our data so that we can view all of the observations for a single sample, which might be useful to better understand how each cell line is behaving, and compare different features of each cell line. For the purposes of the analysis we do here, we want a dataframe where each row is a cell line, and each column describes some feature about that cell line – which is a similar format to our CCLE_metadata dataframe. This format isn't EXACTLY tidy, but we can achieve this format from a tidy dataset.

Tidying the RNAseq data

So the first thing to note is that the first column doesn't contain any RNA expression data! Instead, it contains an ID that looks like ACH-XXXXXX, which is an identifier for the cell line. It should match up with the `DepMap_ID` column in the CCLE_metadata.

The second thing to note are the other columns that each represent a gene. The format of these is SYMBOL (EntrezID). SYMBOLS are the gene names we're most familiar with, while the Entrez ID is a format that's used to maintain databases of genes, and can help identify homologs in other species, for example.

One thing we definitely want to do is map the DepMap_ID to the CCLE_Name in the `CCLE_metadata` dataframe, because those names are more human-readable. Here, I add a new first column of DepMap IDs with CCLE names using the `match` function to find the indices of the DepMap_IDs in the CCLE_expression data with the DepMap_IDs in CCLE_metadata. Using those matched indices, I can add data from the metadata to the expression data.

Here, I also use the `mutate` function to add new columns to the `CCLE_expression` dataframe. Mutate simply allows us to change the columns in the dataframe, either by adding new columns, or modifying

existing columns. One cool thing is that in line #104 (marked by the ***), I'm using mutate to refer to an existing column in the dataframe to create a new column.

```
# Create a new CCLE_combined dataframe that brings together the metadata and expression data.
CCLE_combined <- CCLE_expression %>%
  # Rename the 1st column to DepMap_ID (because that's what it actually represents)
  rename(DepMap_ID = X1) %>%
  # Use the merge function to combine all of the expression data with the metadata
  merge(CCLE_metadata) %>%
  # Add a column of TRUE or FALSE, indicating if KRAS is mutated or not
  # If codon change is NOT NA (the ! means NOT), then set the value to TRUE.
  mutate(KRAS_mutated = !is.na(Codon_Change)) # *** #
```

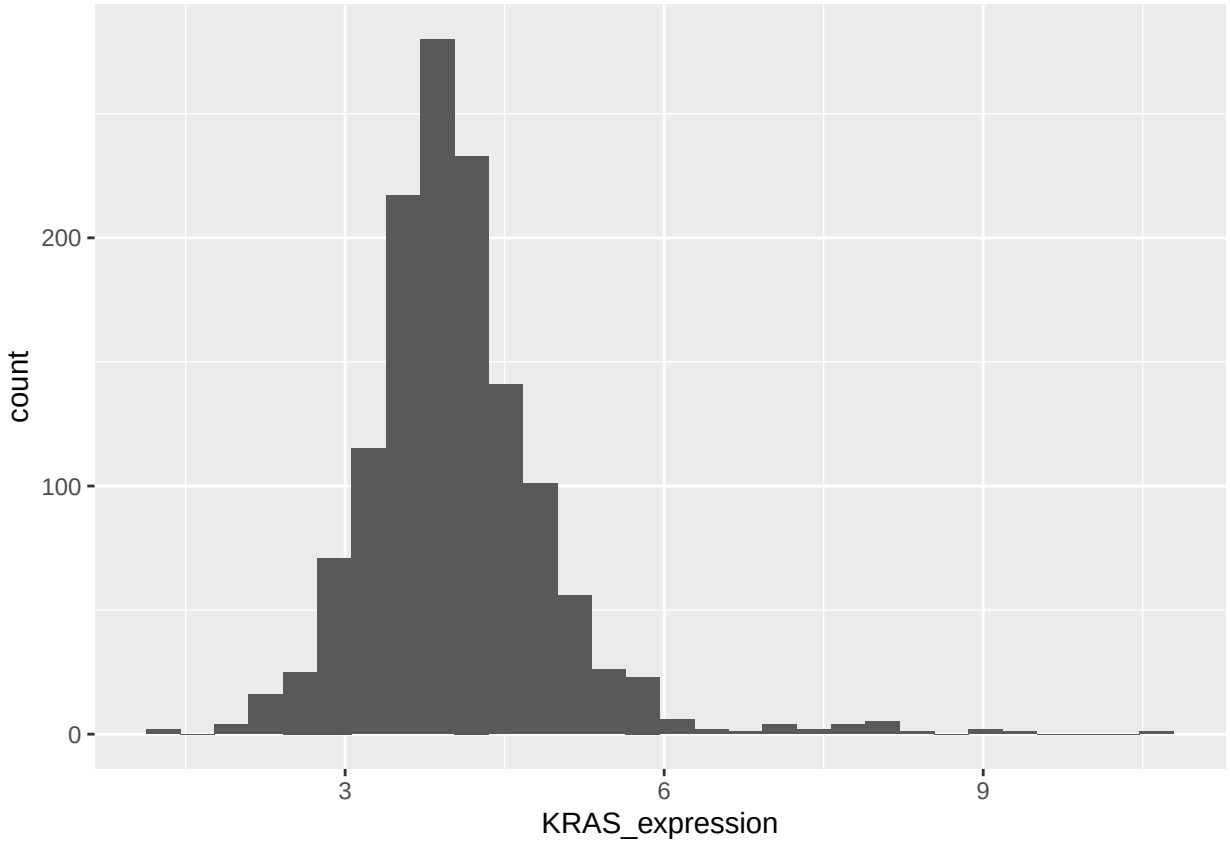
Understanding RNA expression patterns

Now that we've loaded in and cleaned up our data a little, let's start looking at some of the patterns in the data. One way we can easily understand the full scope of the data is by visualizing it with a histogram. Histograms are useful because they let us see the full range of data, and help us identify any outliers or what the overall distribution of the data is.

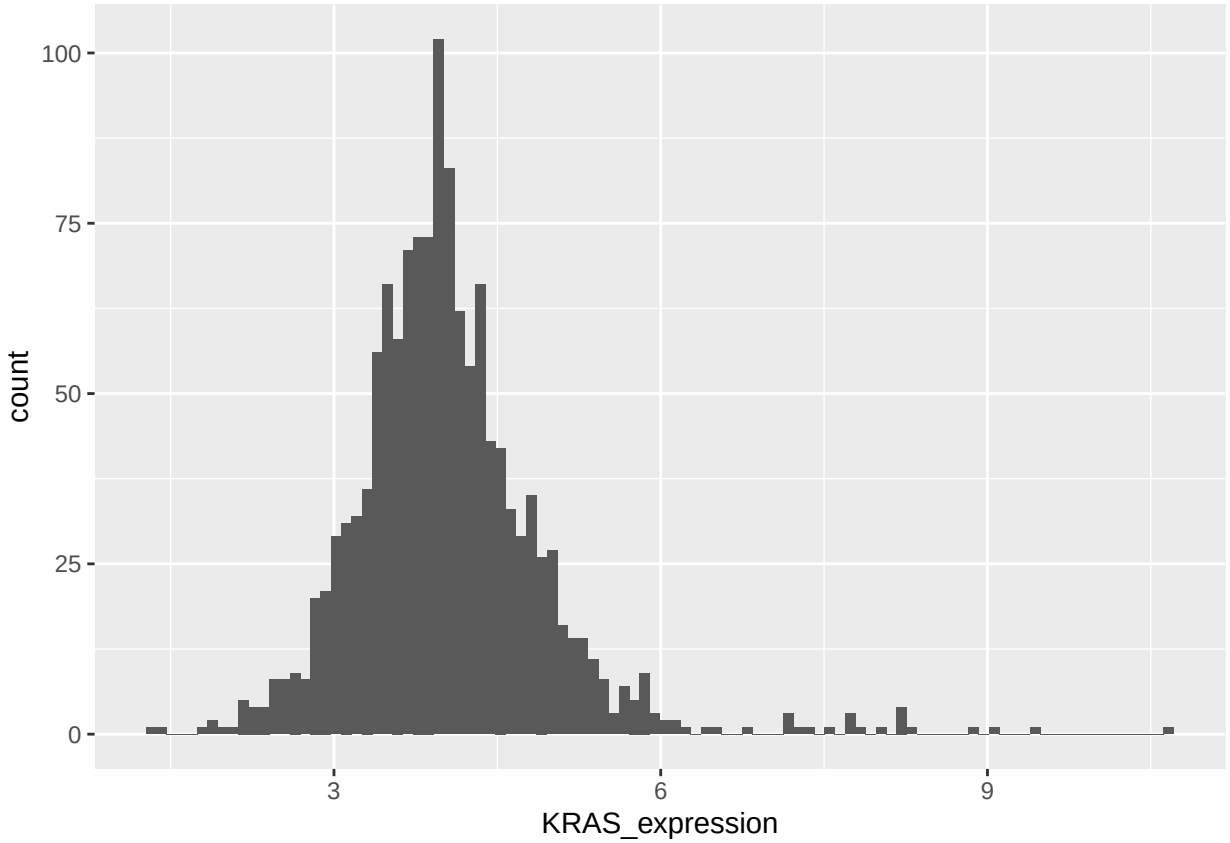
Let's visualize what the expression of one gene across many cell lines looks like. We can narrow down the dataframe to select columns that we want to use with the `select` function.

```
# First, pick a gene of interest. Here, we'll look at KRAS expression,
KRAS_data <- CCLE_combined %>%
  select(KRAS_expression = `KRAS (3845)`, CCLE_Name, KRAS_mutated) # Pick the columns of interest (rena

# Now let's visualize with ggplot - note that with ggplot, we don't need to quote anything!
ggplot(data = KRAS_data, mapping = aes(x = KRAS_expression)) +
  geom_histogram() # Let's use a histogram to visualize the distribution of expression across cell line
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

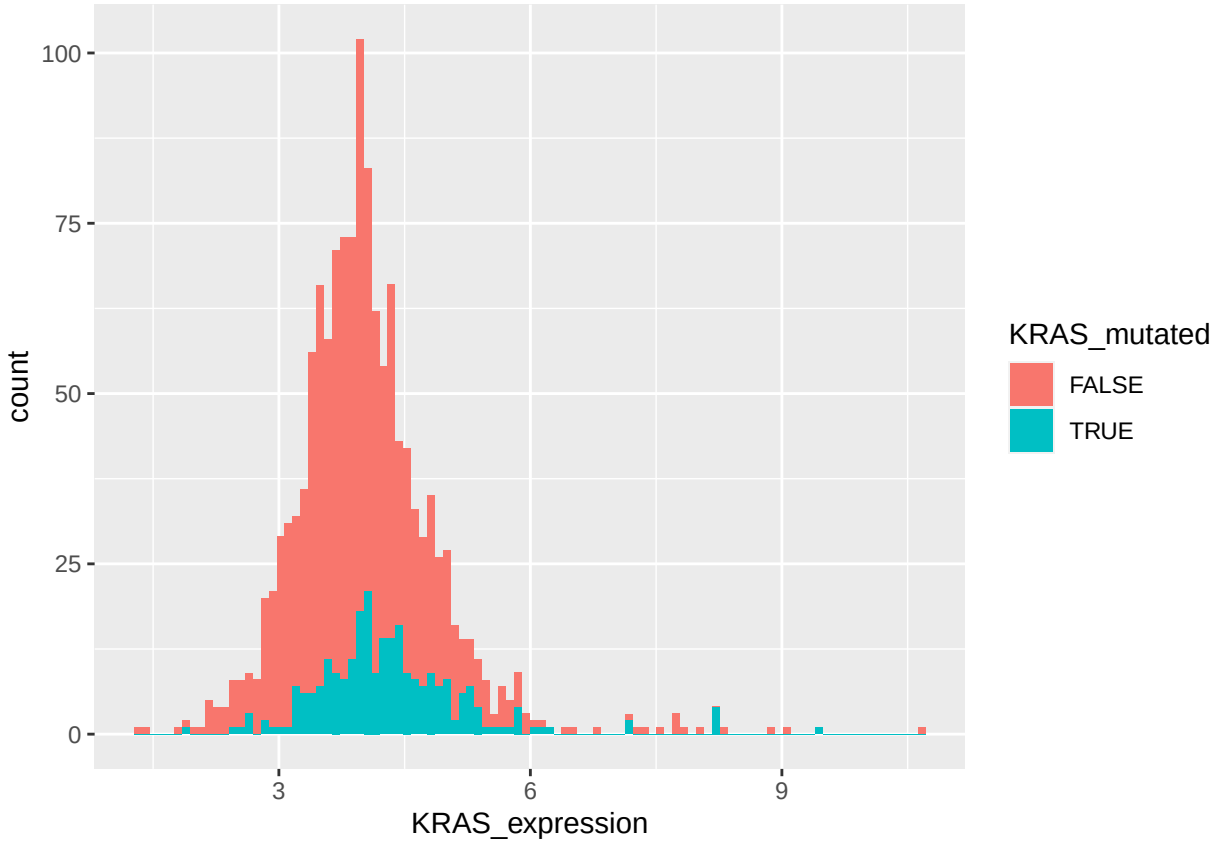



```
# If you notice, we got a warning saying that we should pick a better binwidth value.
# That's why we can't see as much of the granularity of the data as we want to.
# So let's try making another histogram, this time with more bins.
ggplot(data = KRAS_data, mapping = aes(x = KRAS_expression)) +
  geom_histogram(bins = 100)
```



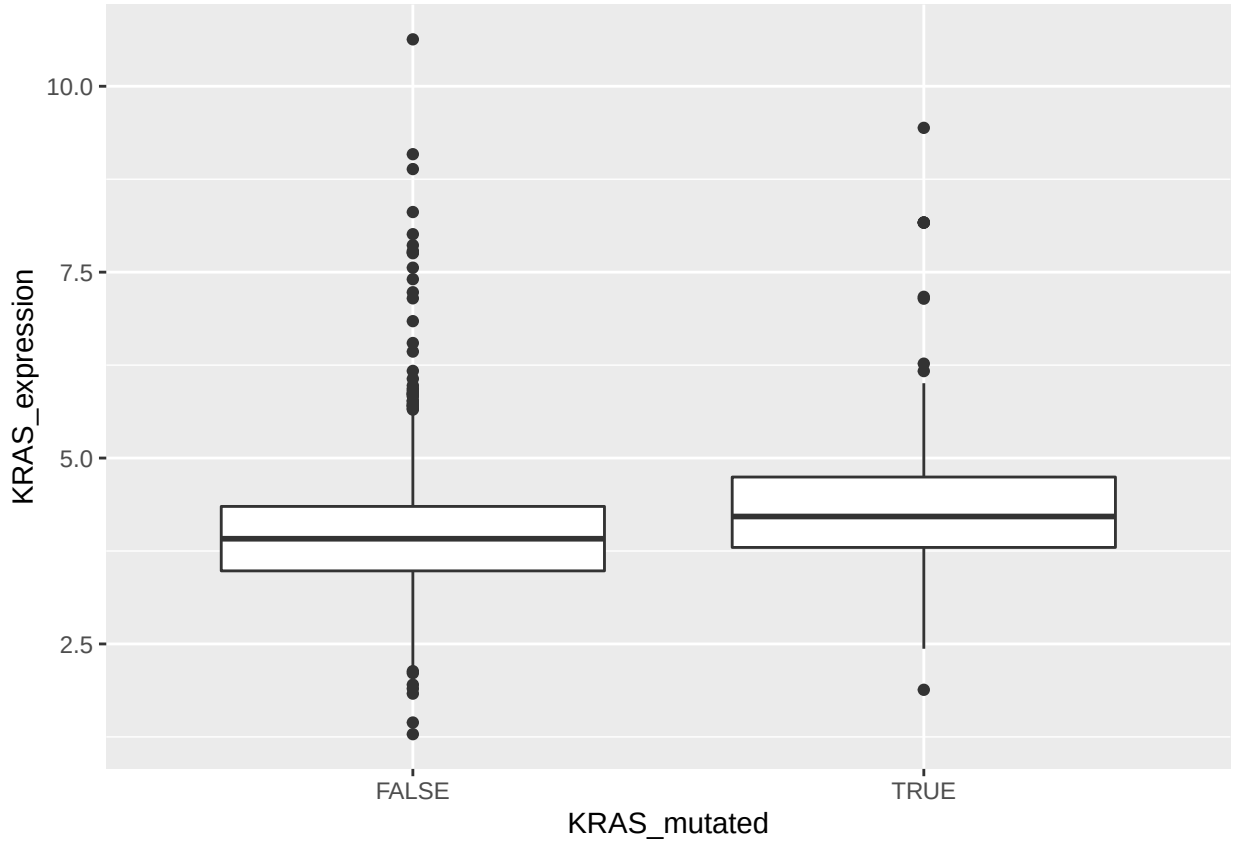
So it looks like there is a wide spread of cell lines expressing KRAS. There are also some outlier cell lines that seem to be expressing very high levels of KRAS. In the Stephens et al. 2017 paper you were assigned to read, the authors explored the relationship between KRAS expression and KRAS mutation. Let's try to separate KRAS mutant cell lines from KRAS wild-type cell lines to see if there's a difference between the distributions in KRAS expression. To do this, we can add another dimension to our histogram by adding a fill color, which we'll link to the variable `KRAS_mutated`.

```
# Use KRAS_mutated as a fill, to color by KRAS mutant status
ggplot(data = KRAS_data, mapping = aes(x = KRAS_expression, fill = KRAS_mutated)) +
  geom_histogram(bins = 100)
```



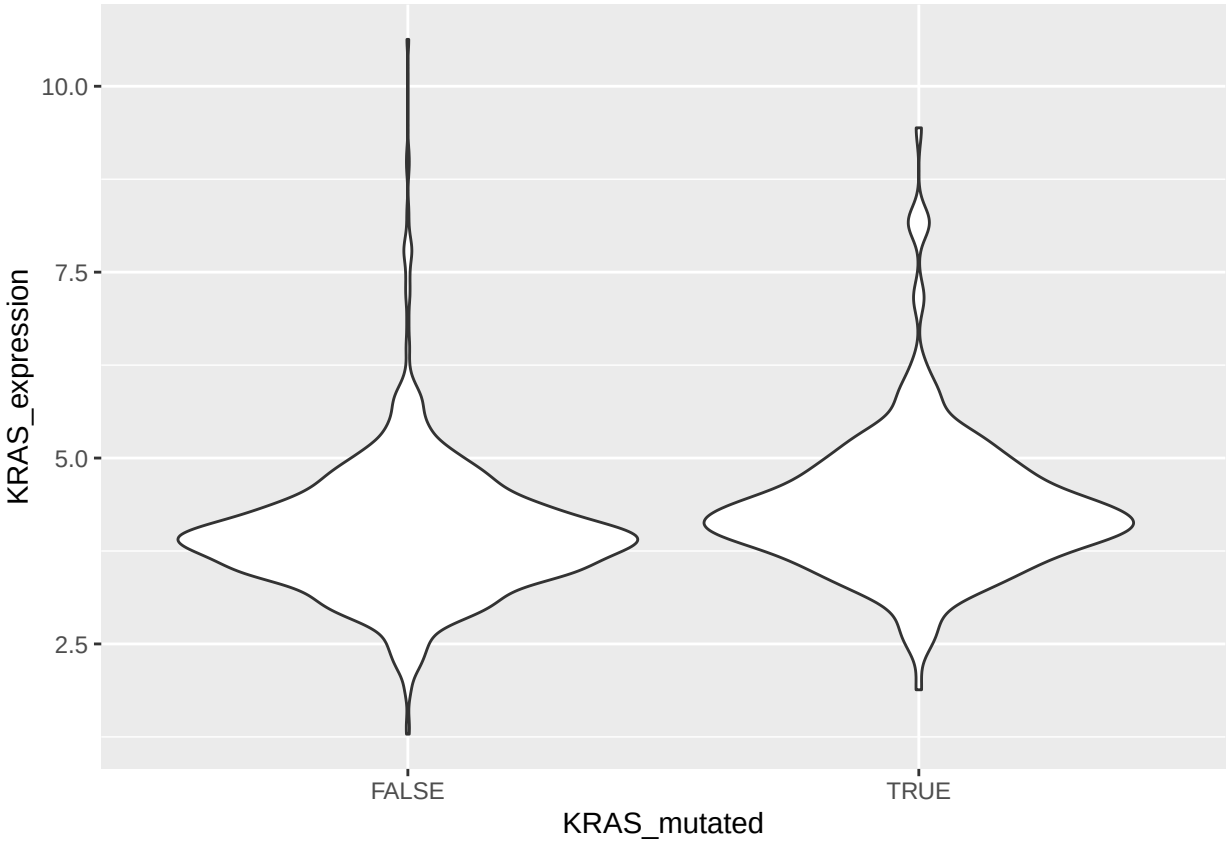
Another way to visualize this data might be with a boxplot. Boxplots are useful in that they show us where the median/quartiles of the data are, and allow us to more direclty compare two groups. Here, we make a boxplot comparing the same two groups.

```
ggplot(data = KRAS_data, aes(x = KRAS_mutated, y = KRAS_expression)) +
  geom_boxplot()
```



And a third way to combine the nuances of a histogram with the delineated nature of a boxplot would be a violin plot, seen here:

```
ggplot(data = KRAS_data, aes(x = KRAS_mutated, y = KRAS_expression)) +
  geom_violin()
```



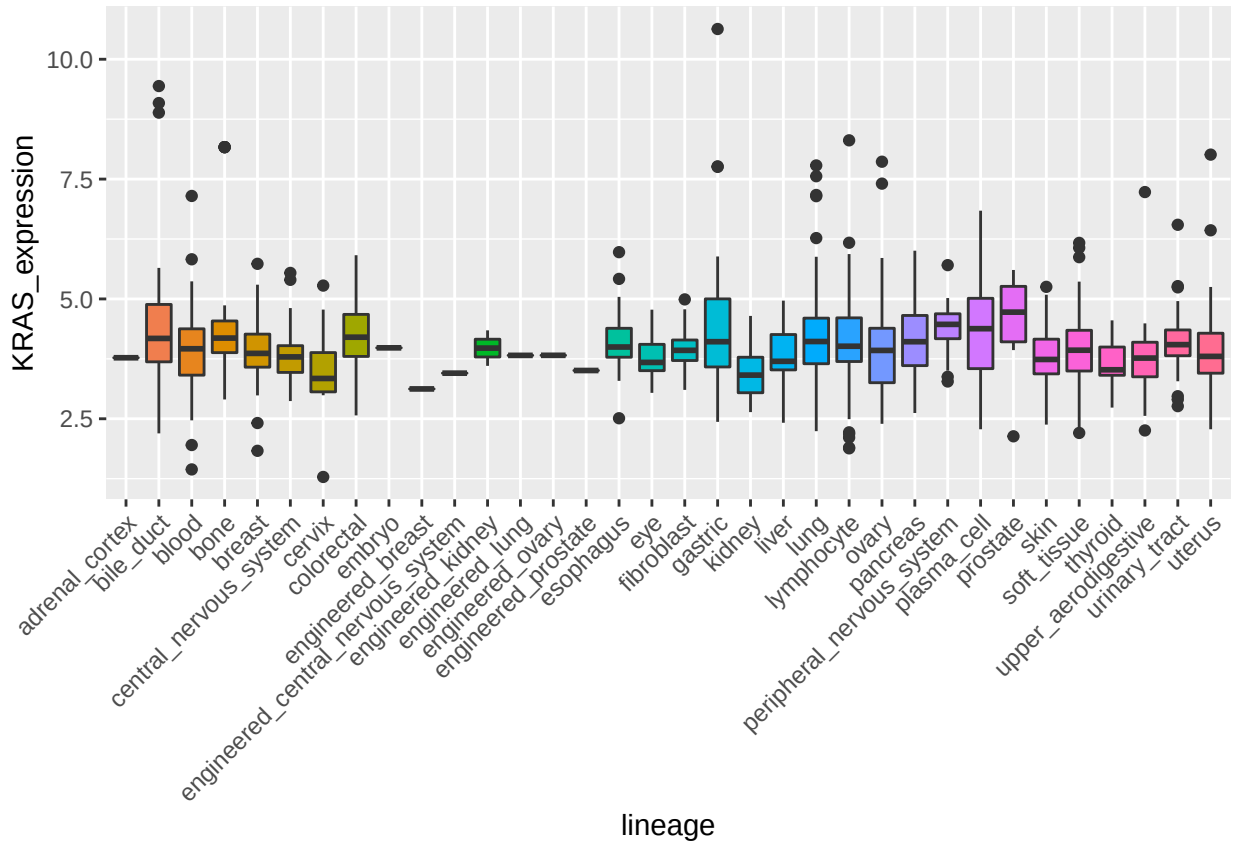
What conclusions can you draw from these graph about the difference in KRAS expression between KRAS mutant and KRAS wildtype cells? Does it look like there is a meaningful difference between the two groups? How might you determine if there is a statistical difference between the two groups? No need to answer, just food for thought?

Looking at differences between tissue types

Another difference we may want to take a closer look at is how different tissue types may impact KRAS expression. Biologically, different tissues express different transcription factors, so this may in fact explain a large proportion of variability in expression.

The first thing we want to do is understand which tissue types tend to have higher or lower expression.

```
# Again, let's create a new dataframe with the data we want to visualize.
KRAS_data <- CCLE_combined %>%
  select(KRAS_expression = `KRAS (3845)`, lineage) # Pick the columns of interest (renaming the KRAS da

# Make a boxplot, colored by tissue (lineage), for each tissue's expression of KRAS.
ggplot(data = KRAS_data,
       aes(x = lineage, y = KRAS_expression, fill = lineage)) +
  geom_boxplot() +
  theme(legend.position = 'none', #Remove the legend (it's the same as the x-axis)
        axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1)) # Rotate the x-axis labels by 45
```



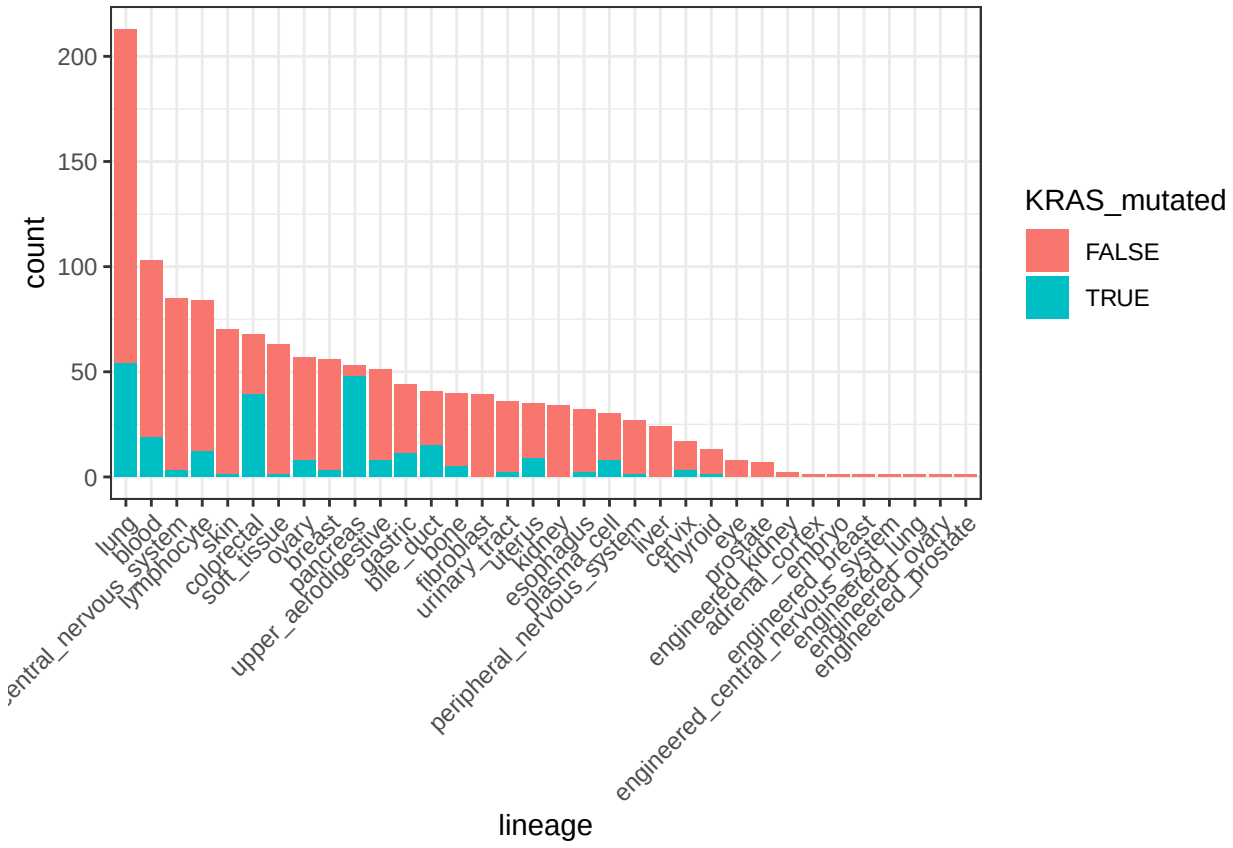
So we can see the ranges of KRAS expression in different lineages. In the Stephens et al. 2017 paper you were assigned to read, the authors explored the relationship between KRAS expression and KRAS mutation.

Questions

> Mutations in the KRAS gene occur in most pancreas and many lung, colorectal and uterine corpus endometrial cancers. Less frequent KRAS mutations have been observed in cancers of the breast, cervix, hepatobiliary duct and testicular germ cell.

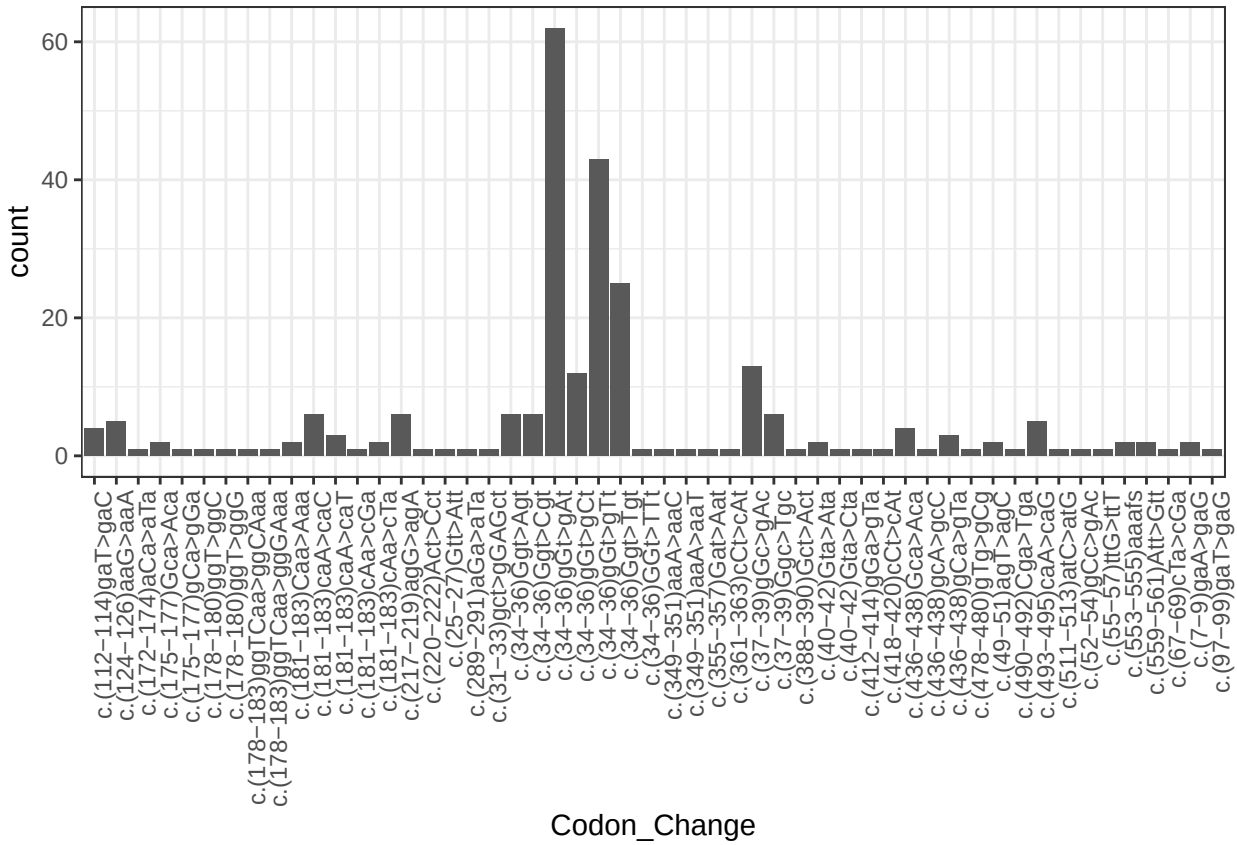
- Q1. Load in CCLE's cell line annotations. For each type of cell line ("lineage" in annotation), what is the number of cell lines that has a KRAS mutation? Use a barplot to display it: x-axis is the cell line type, y-axis is the count. Then, sort the barplot with the most frequent disease type to come first, least frequent disease type to come last (hint: try making lineage an ordered factor using the `factor()` function, and be sure to specify levels!).

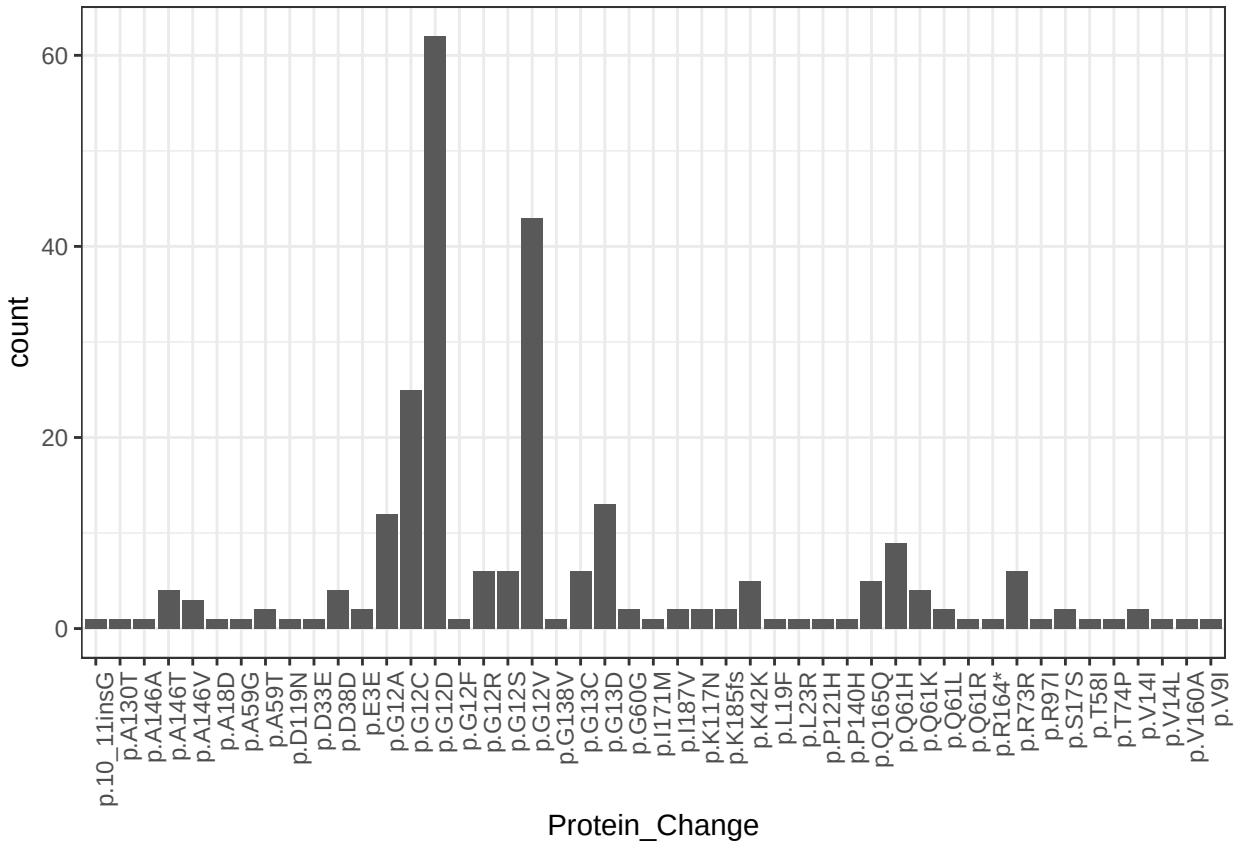
```
theme_set(theme_bw()) # Set a nice theme

# Re-order the lineage by converting it to an ordered factor
CCLE_combined$lineage <- factor(CCLE_combined$lineage, levels = names(sort(table(CCLE_combined$lineage)

# Create a plot
ggplot(data = CCLE_combined, aes(x= lineage, fill = KRAS_mutated)) +
  geom_bar() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1))
```



In cancer, mutations (primarily in codons 12, 13, and 61) render Ras proteins resistant to GAP-assisted GTP hydrolysis resulting in a constitutively active RAS molecule. Less frequently, mutations occur (eg, at alanine 146) that lower the affinity of Ras proteins for GDP, allowing exchange of GDP for GTP without upstream signaling through GEFs. Importantly, these activating mutations in KRAS affect both the KRAS4A and KRAS4B isoforms.

- Q2. Take a look at the columns "Codon_Change" and "Protein_Change". This gives information about which codon is mutated and the context of the mutation. Again, make a barplot documenting the count of the different types of codon/protein changes in KRAS. Where do the majority of codon/protein changes occur? Can you infer anything about that site?

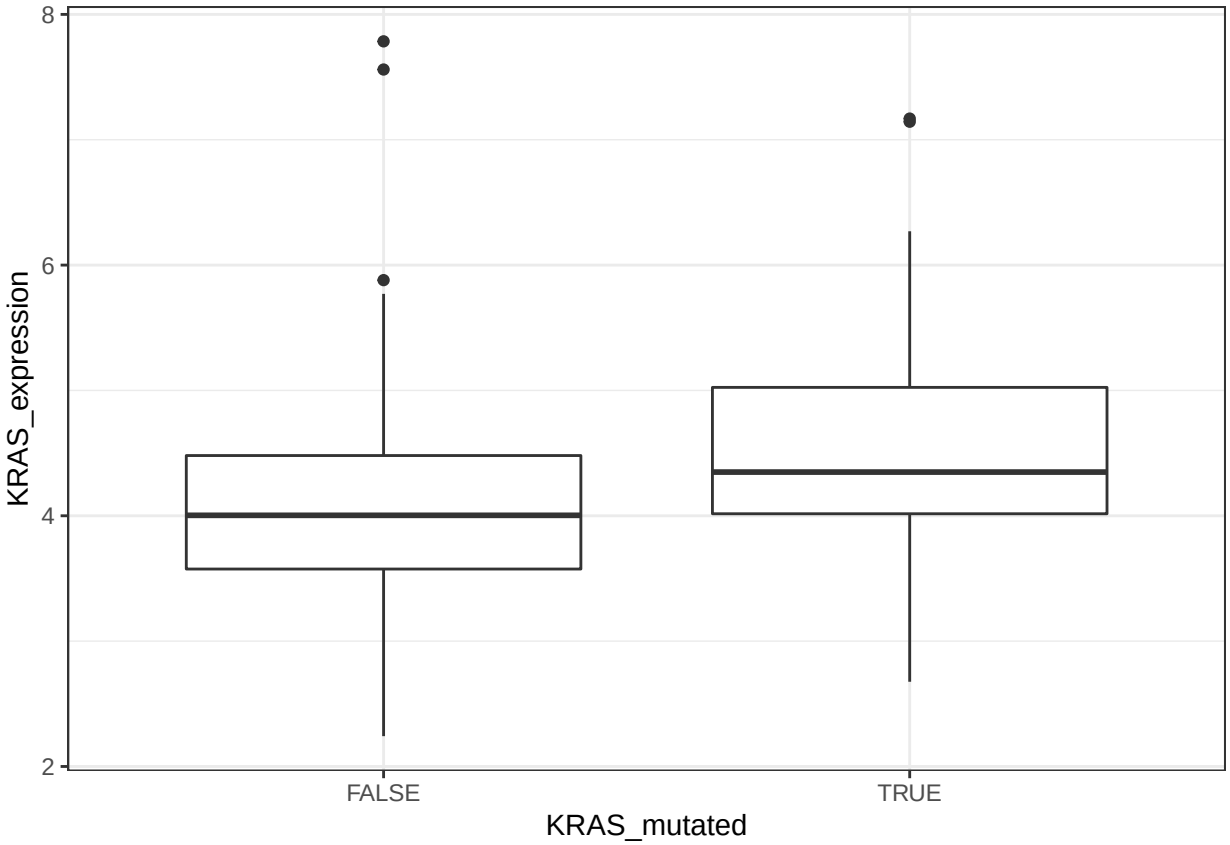

count
60
40
20
0
c.(112−114)gaT>gaC
c.(124−126)aaG>aaA
c.(172−174)aCa>aTa
c.(175−177)Gca>Aca
c.(175−177)gCa>gGa
c.(178−180)ggT>ggC
c.(178−180)ggT>ggG
c.(178−183)ggTCaa>ggCAaa
c.(178−183)ggTCaa>ggGAaa
c.(181−183)Caa>Aaa
c.(181−183)caA>caC
c.(181−183)caA>caT
c.(181−183)cAa>cGa
c.(181−183)cAa>cTa
c.(217−219)agG>agA
c.(220−222)Act>Cct
c.(25−27)Gtt>Att
c.(289−291)aGa>aTa
c.(31−33)gct>gGAGct
c.(34−36)Ggt>Agt
c.(34−36)Ggt>Cgt
c.(34−36)gGt>gAt
c.(34−36)gGt>gCt
c.(34−36)gGt>gTt
c.(34−36)Ggt>Tgt
c.(34−36)GGt>TTt
c.(349−351)aaA>aaC
c.(349−351)aaA>aaT
c.(355−357)Gat>Aat
c.(361−363)cCt>cAt
c.(37−39)gGc>gAc
c.(37−39)Ggc>Tgc
c.(388−390)Gct>Act
c.(40−42)Gta>Ata
c.(40−42)Gta>Cta
c.(412−414)gGa>gTa
c.(418−420)cCt>cAt
c.(436−438)Gca>Aca
c.(436−438)gcA>gcC
c.(436−438)gCa>gTa
c.(478−480)gTg>gCg
c.(49−51)agT>agC
c.(490−492)Cga>Tga
c.(493−495)caA>caG
c.(511−513)atC>atG
c.(52−54)gCc>gAc
c.(55−57)ttG>ttT
c.(553−555)aaafs
c.(559−561)Att>Gtt
c.(67−69)cTa>cGa
c.(7−9)gaA>gaG
c.(97−99)gaT>gaG
Codon_Change



In addition to point mutations that increase RAS activity, RAS genes are often amplified in human cancers. For example, KRAS is frequently amplified in ovarian, lung squamous cell, uterine, adrenocortical, and esophageal tumors. This suggests that both mutation and expression levels contribute to the roles that RAS genes play in oncogenesis. We first focused on KRAS and its KRAS4A and KRAS4B isoforms within the TCGA tumor types in which KRAS is frequently mutated. Figure 1A shows expression levels for KRAS in lung adenocarcinoma (LUAD), pancreatic ductal adenocarcinoma (PAAD), and colon adenocarcinoma (COAD) samples. In each tumor type, we observed statistically significant increases in expression of mutant KRAS using the t test. The values are shown on the figure along with the numbers of samples for each condition.

- Q3. Using the RNAseq expression data AND mutation data, we want to identify if there is a significant relationship between KRAS mutation and expression. Subset to lung cancer cell lines only and create a boxplot of KRAS expression dependent on KRAS WT/MUT. Does your figure resemble Fig. 1? Why or why not? Do the same for pancreatic and colorectal lines. BONUS: Perform a t-test to see if there is a difference in KRAS expression between KRAS wild-type and KRAS mutant cell lines.

```
subsetted_data <- CCLE_combined %>%
  filter(lineage == "lung") %>%
  select(KRAS_expression = `KRAS (3845)`, KRAS_mutated = KRAS_mutated)

ggplot(data = subsetted_data, aes(x = KRAS_mutated, y = KRAS_expression)) +
  geom_boxplot()
```




```
# Test if these two are significantly different
t.test(KRAS_expression ~ KRAS_mutated, data = subsetted_data) # YES!
```

```
##
##  Welch Two Sample t-test
##
## data:  KRAS_expression by KRAS_mutated
## t = -3.2231, df = 81.346, p-value = 0.001826
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.7381980 -0.1746863
## sample estimates:
## mean in group FALSE  mean in group TRUE
##            4.059474            4.515917
```

- Q4. Upstream of KRAS is the receptor tyrosine kinase EGFR. EGFR is often the driver mutation in non-KRAS driven cancers, and, interestingly, EGFR mutations tend to be mutually exclusive with KRAS mutations. In fact, this is addressed in the paper:

  In the bars at the bottom of the plot, the positions of samples with mutations in TP53, epidermal growth factor receptor (EGFR), KEAP1, and NF1 are indicated. Of these, the EGFR mutations do seem to be enriched toward the left side of the panel, indicating lowered KRAS expression. Although the TP53 mutations are enriched in the non-KRAS–mutated samples (data not shown), that relationship is not evident on the plot. We did not see any enrichment of either KEAP1 or NF1 in terms of KRAS expression levels. The EGFR relationship is expected as KRAS mutations and EGFR mutations are mutually exclusive.

We can follow this up by looking at EGFR mutations with EGFR expression, in the same way we explored KRAS above. Do EGFR-mutated cell lines also express EGFR higher than EGFR-wildtype cell lines?

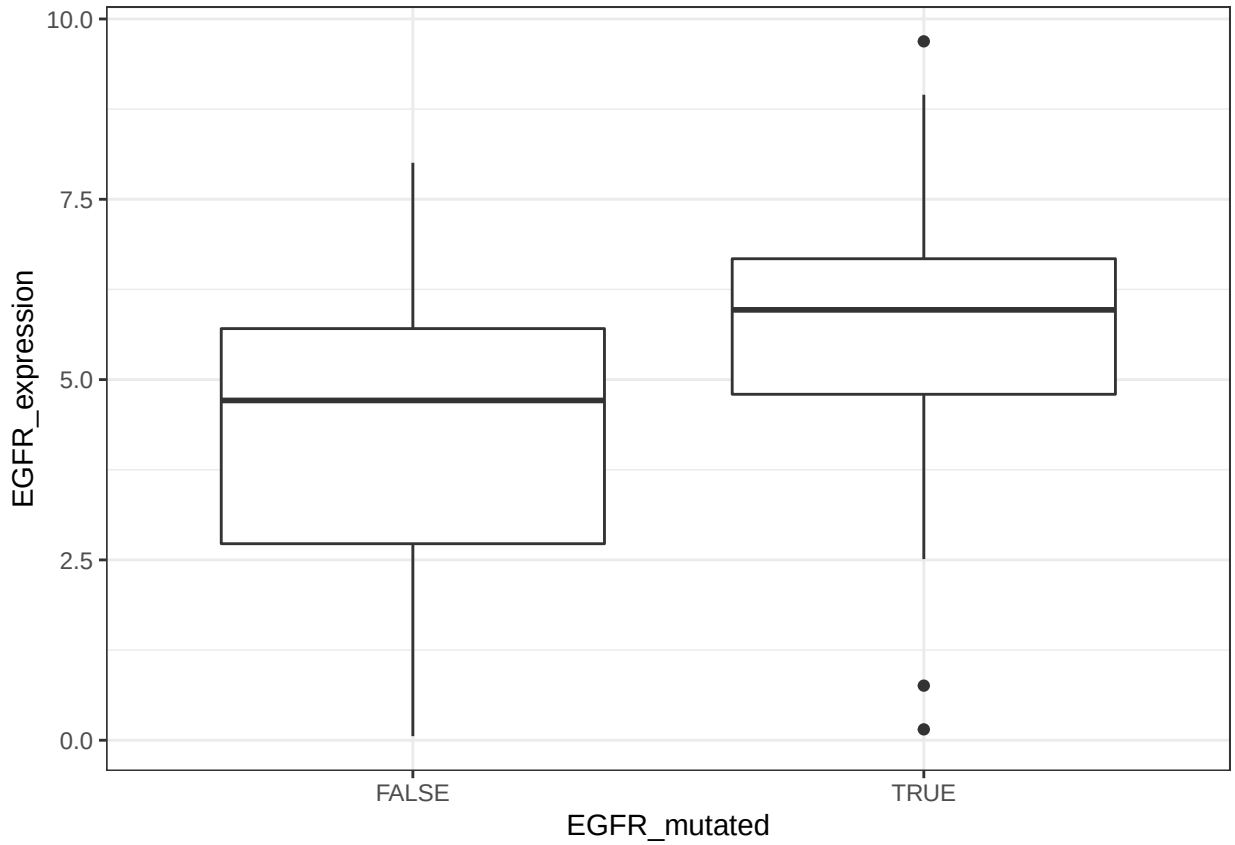
```
egfr_mutations <- CCLE_mutations %>%
  filter(Hugo_Symbol=="EGFR") %>%
  select(Hugo_Symbol, DepMap_ID, Protein_Change, Codon_Change)

CCLE_metadata <- read_csv("sample_info.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   COSMICID = col_double(),
##   Achilles_n_replicates = col_double(),
##   cell_line_NNMD = col_double(),
##   age = col_double(),
##   depmap_public_comments = col_logical()
## )
```

```
## See spec(...) for full column specifications.
```

```
## Warning: 11 parsing failures.
##  row                    col           expected                                        actual
## 1105 depmap_public_comments 1/0/T/F/TRUE/FALSE Adherent version of CCLFPEDS0001T          'sample_in
## 1166 depmap_public_comments 1/0/T/F/TRUE/FALSE SV40+TERT immortalized kidney cell line    'sample_in
## 1454 depmap_public_comments 1/0/T/F/TRUE/FALSE Drug resistance: Dabrafenib and Trametinib 'sample_in
## 1455 depmap_public_comments 1/0/T/F/TRUE/FALSE Drug resistance: SCH772984                 'sample_in
## 1456 depmap_public_comments 1/0/T/F/TRUE/FALSE Drug resistance: Dabrafenib and Trametinib 'sample_in
## .... ..................... .................. ........................................ .........
## See problems(...) for more details.
```

```
CCLE_metadata <- merge(CCLE_metadata, egfr_mutations, all.x = T)

CCLE_combined <- CCLE_expression %>%
  rename(DepMap_ID = X1) %>%
  merge(CCLE_metadata) %>%
  mutate(EGFR_mutated = !is.na(Codon_Change))


# Subset the data for only cell lines where lineage == "lung"
EGFR_data <- CCLE_combined %>%
  filter(lineage == "lung") %>%
  select(EGFR_expression = `EGFR (1956)`, CCLE_Name, EGFR_mutated)

# Create a boxplot
ggplot(data = EGFR_data, aes(x = EGFR_mutated, y = EGFR_expression)) +
  geom_boxplot()
```




```
# Test if these two are significantly different
t.test(EGFR_expression ~ EGFR_mutated, data = EGFR_data) # YES!
```

```
##
##  Welch Two Sample t-test
##
## data:  EGFR_expression by EGFR_mutated
## t = -3.9906, df = 43.657, p-value = 0.0002481
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2.3856176 -0.7843476
## sample estimates:
## mean in group FALSE  mean in group TRUE
##            4.078850            5.663832
```

*Q5. BONUS: Are there any other genes or pathways that you're interested in? Feel free to think beyond the scope of this paper or your readings. What kinds of questions can you ask with a combination of mutation and expression data for all of these different cell lines/tissue types?