BIG DATA IN ENVIRONMENTAL SCIENCE AND TOXICOLOGY

superfund.tamu.edu

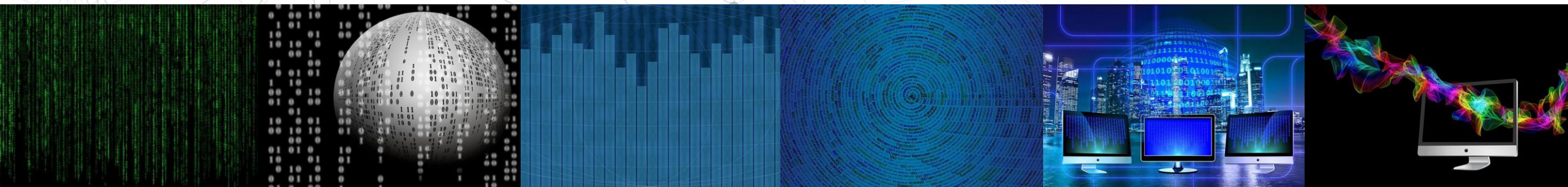
TEXAS A&M UNIVERSITY
Superfund Research Center

Welcome !

This Session will Begin at 2:00 pm Eastern US Time

"Manipulating and Displaying Big(ish) Data in R "

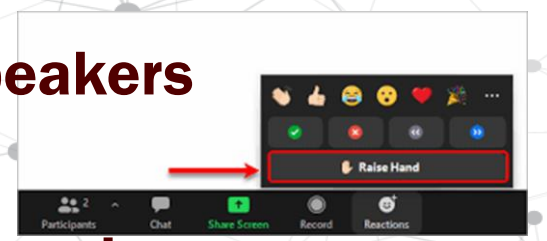
Allison Dickey– North Carolina State University
Fred Wright– North Carolina State University
Burcu Beykal– University of Connecticut



BIG DATA IN ENVIRONMENTAL SCIENCE AND TOXICOLOGY

TEXAS A&M UNIVERSITY
Superfund Research Center

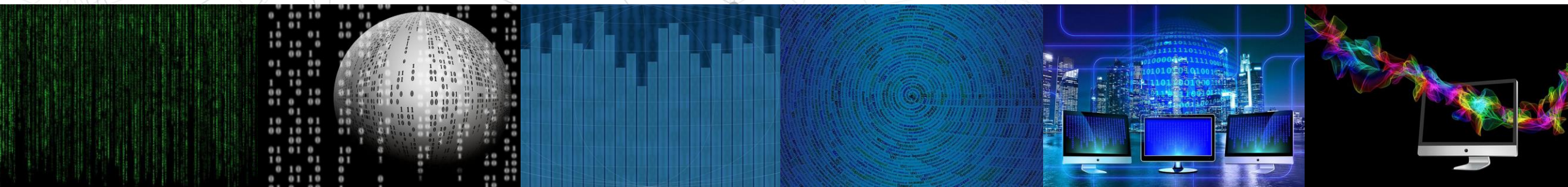
- All participants are muted to enable the speakers to present without interruption.
- Please rename yourself and designate Full Name and Affiliation.
- Last-minute installation issues? Use private chat to any of the speakers when they're not speaking.



- Use the reaction icon at the bottom of your screen to raise your hand.

However, due to time constraints, most questions will be answered in chat window.

- This meeting will be recorded and posted on the @tamusuperfund website https://superfund.tamu.edu/big-data-series-2021/ in the coming weeks.



Allison Dickey

Fred Wright

Burcu Beykal

Part 1: An introduction to R

Part 2: Visualization and analysis

Part 3: More Advanced Visualization

What is this session about?

- This session is NOT a comprehensive introduction to R
- We present a number of topics that have been useful to the speakers for Superfund-relevant data.
- In other words, what do we and collaborators do most often?
- We do not assume familiarity with R, and you might need to review material later!

Do you want to just drive it, or understand it? (a bit of both)


forbes.com

vs.


www.southfloridasigncompany.com

Why are there so many… programs about R?

R

RStudio

WALT DISNEY PICTURES/PHOTOFEST

R Open

RShiny

RMarkdown

Why are there so many… programs about R?

Actual language

R

Enhanced interface

RStudio

WALT DISNEY PICTURES/PHOTOFEST

~~R Open~~

Another version of R, can be faster on some computers for some code

~~RMarkdown~~

Enhanced reproducibility, save as html

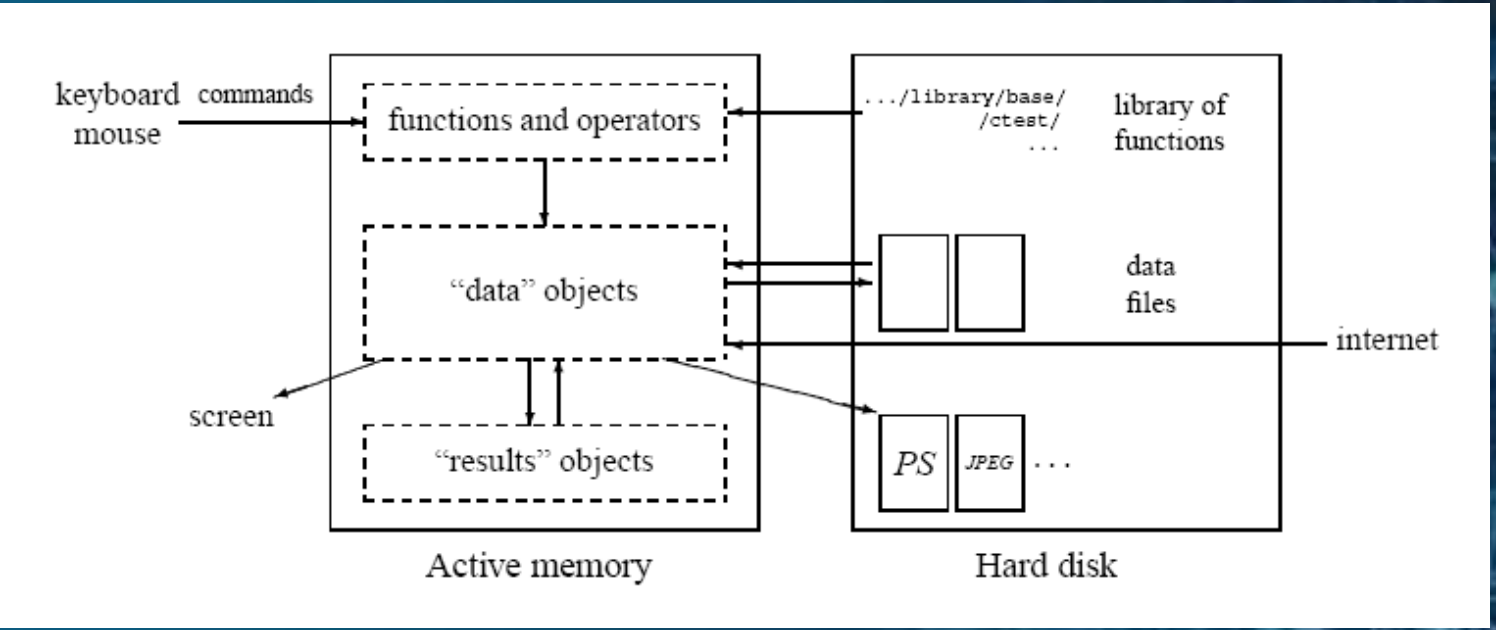
~~RShiny~~

Web tools, sliders, radio buttons, etc.

R: a pre-introduction (Part 0)

Fred Wright, NC State University

A schematic view
This will make more sense after today!



A schematic view of R (from E. Paridis, R for Beginners)

- *R* stores variables, data, functions, results, etc, in the form of *objects* which have a *name*.

- The user can do actions on these objects with *operators* (arithmetic, logical, and comparison) and *functions* (which are themselves objects)

- Let's illustrate by starting with the command line in a new session. You'll see ">" starting each command, to distinguish it from output. Later we may drop this

Simple Arithmetic Operations

```
> 5 + 3
[1] 8
> 5 - 3
[1] 2
> 5 * 3
[1] 15
> 5 / 3
[1] 1.666667
> 5^3
[1] 125
> 5 %% 3
[1] 2
> (5 * 3) - (5 / 3)
[1] 13.33333
```

Addition

Subtraction

Multiplication

Division

Exponent

Remainder

```
> n
Error: object 'n' not found
> n = 10
> n
[1] 10
> n <- 15
> n
[1] 15
```

I typed “n” and hit return

Doesn’t exist yet

Assign (using =) the value 10 to object n

Now it exists

Alternate way of assigning. Seen in older R scripts. Think of this as “n is assigned the value 15”. The <- is meant to look like an arrow. ←

If the object already exists, its previous value is erased (the modification affects only the objects in the active memory, not the data on the disk).

Object names : applies to variables & functions

Object names cannot start with digits or periods, but they can have periods (.), underscores (_) and digits within them.

Names are case sensitive.

```
> _d <- 5
Error: unexpected input in "_"
>
> _d <- 5
Error: unexpected input in "_"
> d_ <- 5
> d_
[1] 5
> 1f <- 5
Error: unexpected symbol in "1f"
> f1 <- 5
> f1
[1] 5
> var.name <- 10
> var.name
[1] 10
```

```
> n == 20
[1] FALSE
> n == 15
[1] TRUE
> n < 10
[1] FALSE
> n > 10
[1] TRUE
> n <= 20
[1] TRUE
> n == n
[1] TRUE
> a <- 15
> a < -15
[1] FALSE
> n == a
[1] TRUE
```

Logical “equals” comparison.

“Less than or equal to”

Assignment statement

A query. (Is ‘a’ less than -15?)

Primitive data types : character, numeric, logical

CHARACTER :

```
> a <- "scooby doo"
> a
[1] "scooby doo"
> is.character(a)
[1] TRUE
> typeof(a)
[1] "character"
```

functions that start with “is” often can answer a question about an object

NUMERIC :

```
> a <- 10
> is.numeric(a)
[1] TRUE
> is.character(a)
[1] FALSE
> a <- 10/3
> a
[1] 3.333333
> is.numeric(a)
[1] TRUE
> a^100
[1] 1.940325e+52
> a^1000
[1] Inf
> a^(-1000)
[1] 0
```

Overflow. Value too large.

Underflow. Value too small.

Functions

Functions need an argument, provided in parentheses

```
> a <- 10
> sqrt(a)
[1] 3.162278
> is.numeric(a)
[1] TRUE
```

R object types (not exhaustive)

- **Vector:**
  - a one-dimensional “array.” All elements of the vector must be of the same data type--numerical, character, etc.
- **Matrix:**
  - a two-dimensional array with an arbitrary number of rows and columns. Again all elements of the matrix must be of the same data type.
- **Data frame:**
  - Also used to store tabular data. Each column can be a different data type. Columns can be thought of as different “variables,” and column names can be used in R built-in functions.

Here is a matrix I made


```
> m
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
 [1,]   -5   -4   -3   -2   -1    0    1    2    3     4     5
 [2,]  -10   -8   -6   -4   -2    0    2    4    6     8    10
 [3,]  -15  -12   -9   -6   -3    0    3    6    9    12    15
 [4,]  -20  -16  -12   -8   -4    0    4    8   12    16    20
 [5,]  -25  -20  -15  -10   -5    0    5   10   15    20    25
 [6,]  -30  -24  -18  -12   -6    0    6   12   18    24    30
 [7,]  -35  -28  -21  -14   -7    0    7   14   21    28    35
 [8,]  -40  -32  -24  -16   -8    0    8   16   24    32    40
 [9,]  -45  -36  -27  -18   -9    0    9   18   27    36    45
[10,]  -50  -40  -30  -20  -10    0   10   20   30    40    50
[11,]  -55  -44  -33  -22  -11    0   11   22   33    44    55
> dim(m)
[1] 11 11
```

Single Element

```
> m[3,2]
[1] -12
```

[row, column]

Note : element selection used square brackets [].

Column Selection

```
> m[,2]
 [1]  -4  -8 -12 -16 -20 -24 -28 -32 -36 -40 -44
```

[all rows, column 2]

```
> smaller.m <- m[2:3,4:8]
> smaller.m
     [,1] [,2] [,3] [,4] [,5]
[1,]   -4   -2    0    2    4
[2,]   -6   -3    0    3    6
> smaller.m^2
     [,1] [,2] [,3] [,4] [,5]
[1,]   16    4    0    4   16
[2,]   36    9    0    9   36
```

Subset of matrix is a matrix

Many operations operate element-wise

A simple data frame

```
> x<-c(1,2,9,6)
> y<-c(1.2,3.4,1.9,4.5)
> mydf<-data.frame(x,y)
> print(mydf)
  x   y
1 1 1.2
2 2 3.4
3 9 1.9
4 6 4.5
> mydf[,2]
[1] 1.2 3.4 1.9 4.5
> mydf$y
[1] 1.2 3.4 1.9 4.5
```

Next up: Part 1 - The actual Intro! (shown as .html and R code)

http://publicdomainaudiovideo.blogspot.com/

Part 2: Data visualization and basic analysis

Fred Wright, NC State University

Review A: Got an R question?

- In RStudio, use the Help menu
- In the console window (whether Rstudio or basic R), `help()`
- Or `?` before a command
- If you're still not sure, try googling the question, which often leads to a keyword
- More advanced code and solutions appear on sites such as stackoverflow (again, google-able)

Review B: Be careful not to lose data!

- R can “hang” if you try to plot something huge, or start a very data intensive procedure, etc.
- Sometimes you can “escape.” Literally!
- R objects are stored in memory
- You might need to kill an R session.
- The best defense against data loss? Scripts that can reproduce everything.

Plotting in “base” R

- Simple code for simple plots
- More advanced plots require some understanding of R syntax, but can be fancy
- Want fancy without lots of coding yourself? Stay tuned for next section (but still pay attention now!)

Reading in the data (review from previous)

```
# this code picks up where Part 1 left off
count_table <- read.csv("GSE62902.csv", row.names = 1) # read data
alignment_subset <- count_table[39181:39185, ] # figure out which are the pesky last rows
count_table <- count_table[-(39181:39185), ] # keep only the non-pesky rows
dosage <- count_table[1, ] # save dosage as its own object
count_table <- count_table[-1, ] # now remove dosage, because it's not a sequencing count
```

Normalizing data (not publication-quality)

```
install.packages("preprocessCore") # need only do once for R version
library(preprocessCore) # need only do once for R session

dosage[dosage==0]<-0.003 # I'll eventually take logs, so can't have zero
log10dose<-log10(as.numeric(dosage)) # I like the log scale for dose
X<-as.matrix(count_table)+0.5 # to make sure it's a matrix, +.5 to zero-protect
Xnorm1<-log10(t(t(X)/colSums(X))) # tricky, divide by column sums and take logs
Xnorm2<-normalize.quantiles(Xnorm1) # quantile normalization, "final" expression data
#Xnorm2<-data.frame(Xnorm2) # data frame has some nice properties
colnames(Xnorm2)<-colnames(X) # but now we should add the column names back
rownames(Xnorm2)<-rownames(X) # and we should add the row names back
```

Histograms

```
hist(Xnorm2[,1]) # histogram of the first column
hist(Xnorm2$GSM1535917) # won't work becuase Xnorm2 is not a data frame
hist(Xnorm2,col="yellow") # color!
par(mfrow=c(2,2)) # sets "parameters" for plotting, in this case 2*2 rows/columns plotting
hist(Xnorm2[1,],col=1) # first row=first gene
hist(Xnorm2[2,],col=2) # why did it do that?
hist(Xnorm2[3,],col=3)
hist(Xnorm2[4,],col=4)
```

Boxplots

```
boxplot(Xnorm1) # a boxplot
boxplot(Xnorm2) # reflects alignment
dev.off() # turn off graphics device
```

Summary stats

```
summary(Xnorm2[,1]) # summarizes basic stats
summary(Xnorm2[1,])
print(mean(Xnorm2)) # the mean. try sd,, median, etc.
```

Scatterplots

```
plot(Xnorm2[,2],Xnorm2[,30]) # mouse 2 vs. mouse 30
abline(0,1,col=2) # col=2 means "red"
identify(Xnorm2[,2],Xnorm2[,30]) # click near points
and then to console, hit escape
```

Scatterplots cont.

```
plot(log10dose,Xnorm2[4305,]) # gene 4305 vs. log10dose
plot(log10dose,Xnorm2[4305,],pch=16,col="blue")
plot(log10dose,Xnorm2[4305,],pch=16,col="blue",cex=1.5)
plot(log10dose,Xnorm2[4305,],pch=16,col="blue",cex=1.5,
ylab="normalized expression",main="My plot!")
```

Regression

```
mylm<-lm(Xnorm2[4305,]~log10dose) # fit linear model
summary(mylm) # shows what's in mylm
abline(mylm) # add overlay line

log10dose.2<-log10dose^2 # squared term
mylm<-lm(Xnorm2[4305,]~log10dose+log10dose.2) # fit again, now with quadratic term
points(log10dose,mylm$fitted.values,col=2,pch=3) # overlay fitted values
points(log10dose,mylm$fitted.values,col=2,pch=3,lwd=3) # overlay fittted values, thicker
```

t-tests

```
t.test(Xnorm2[4305,dosage==0.003],Xnorm2[4305,dosage==30]) # t-test of two groups at extremes
?t.test
```

Analysis of variance

```
myaov<-aov(Xnorm2[4305,]~dosage) # analysis of variance. what's wrong?
myaov<-aov(Xnorm2[4305,]~as.factor(dosage)) # okay now it works
summary(myaov) # results
```

Principal components

```
mypca<-prcomp(t(Xnorm2)) #PCA, we use t() to turn data on side
summary(mypca)
str(mypca)
#https://www.datacamp.com/community/tutorials/pca-analysis-r
pairs(mypca$x[,1:3]) # pairs command does multiple pairwise scatterplots
pairs(mypca$x[,1:3],col=as.factor(dosage),pch=16) # color by dose
plot(log10dose,mypca$x[,1],pch=1) # wow! The first PC is largely driven by dose
```

Who says base R can't make fancy plots? (this is utterly gratuitous)

```
#gratuitous plot
a <- (-440:200) / 200
b <- c(-300:300) / 200
b <- complex(length(b), 0, b)
m <- outer(a, b, FUN = "+")
magnitude.m <- sqrt(Re(m)^2 + Im(m)^2)
z <- matrix(0,length(a), length(b))
for (i in (1:20)) {
  z <- z^2 + m
}
magnitude.z <- sqrt(Re(z)^2 + Im(z)^2)
magnitude.z[is.na(magnitude.z)] <-magnitude.m[is.na(magnitude.z)]^500
image(a, Im(b), -log(magnitude.z) * (magnitude.z > 10), col =
c(heat.colors(100),1))
```

Next up: More advanced visualization!

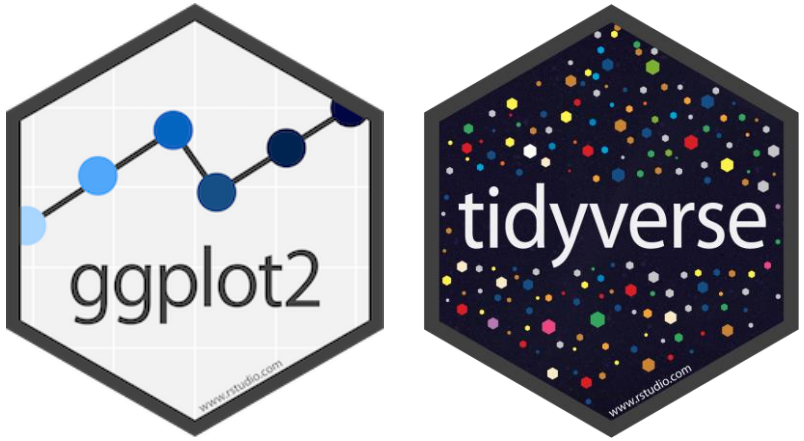
http://publicdomainaudiovideo.blogspot.com/

MORE ADVANCED VISUALIZATION

Burcu Beykal, PhD

Advanced Visualization with the ggplot2 Library

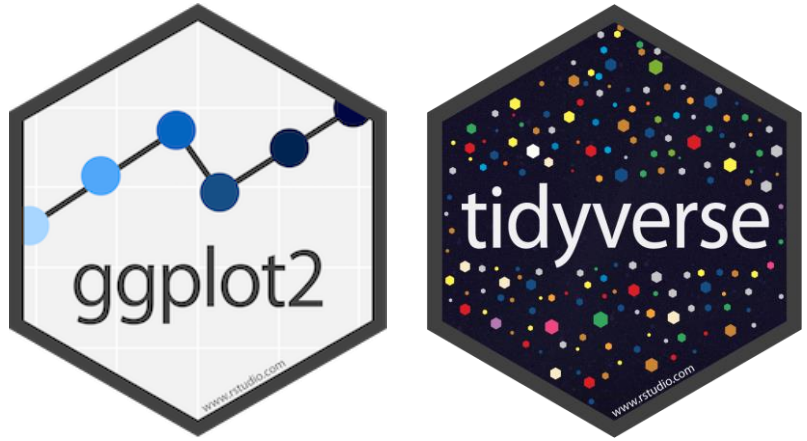
- ggplot2 is based on the concept called **grammar of graphics**
- **Main idea:** any graph can be made from the same 3 components:
    1. **Dataset**
    2. **Coordinate system**
    3. **Geoms** – visual marks to represent data points



TEXAS A&M UNIVERSITY SUPERFUND RESEARCH CENTER

Advanced Visualization with the ggplot2 Library

- ggplot2 is based on the concept called **grammar of graphics**
- **Main idea:** any graph can be made from the same 3 components:
  1. **Dataset**
  2. **Coordinate system**
  3. **Geoms** – visual marks to represent data points



Data

Feature 1	Feature 2
0.0924	5.3452
1.5224	0.9036
⋮	⋮
0.3422	3.2532

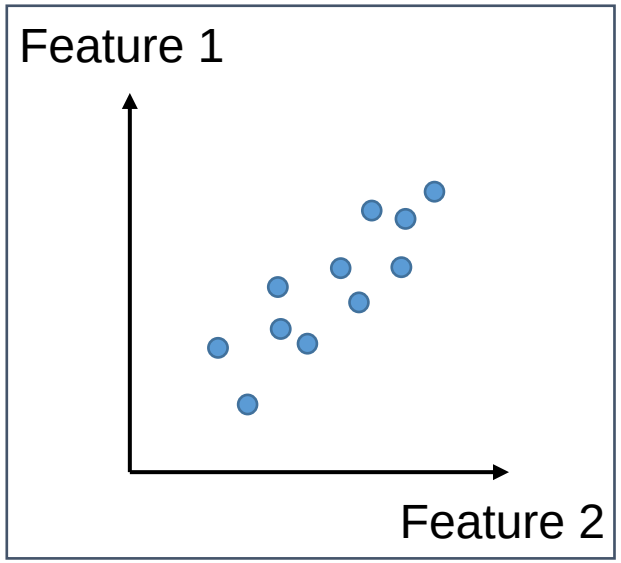
Geoms

Coordinate System

Feature 1

Feature 2

Final Plot



TEXAS A&M UNIVERSITY SUPERFUND RESEARCH CENTER

Syntax of ggplot2

- **Getting started with ggplot2**

```
library(ggplot2)        → load ggplot2 library
mydata <- read.csv(...) → load your dataset
```

- **Generic syntax (help you get things plotted):**

```
ggplot(data=<DATA>) + <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

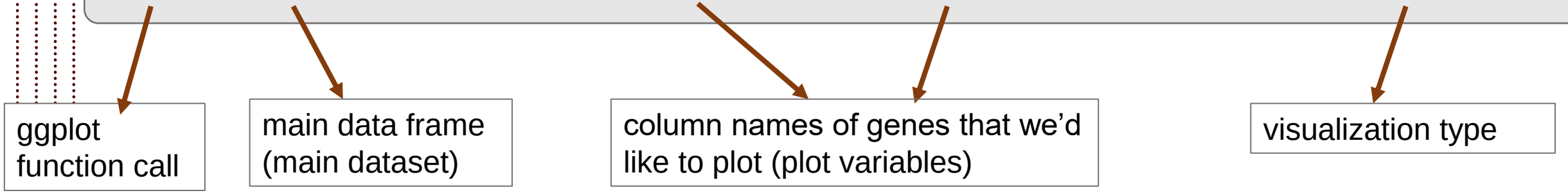
- **Optional layers (help you beautify plots and improve visualization):**

```
+ optional layers (<COORDINATE FUNCTION>, <FACET_FUNCTION>, <SCALE_FUNCTION>,
<THEME_FUNCTION>)
```

Scatter Plot Example

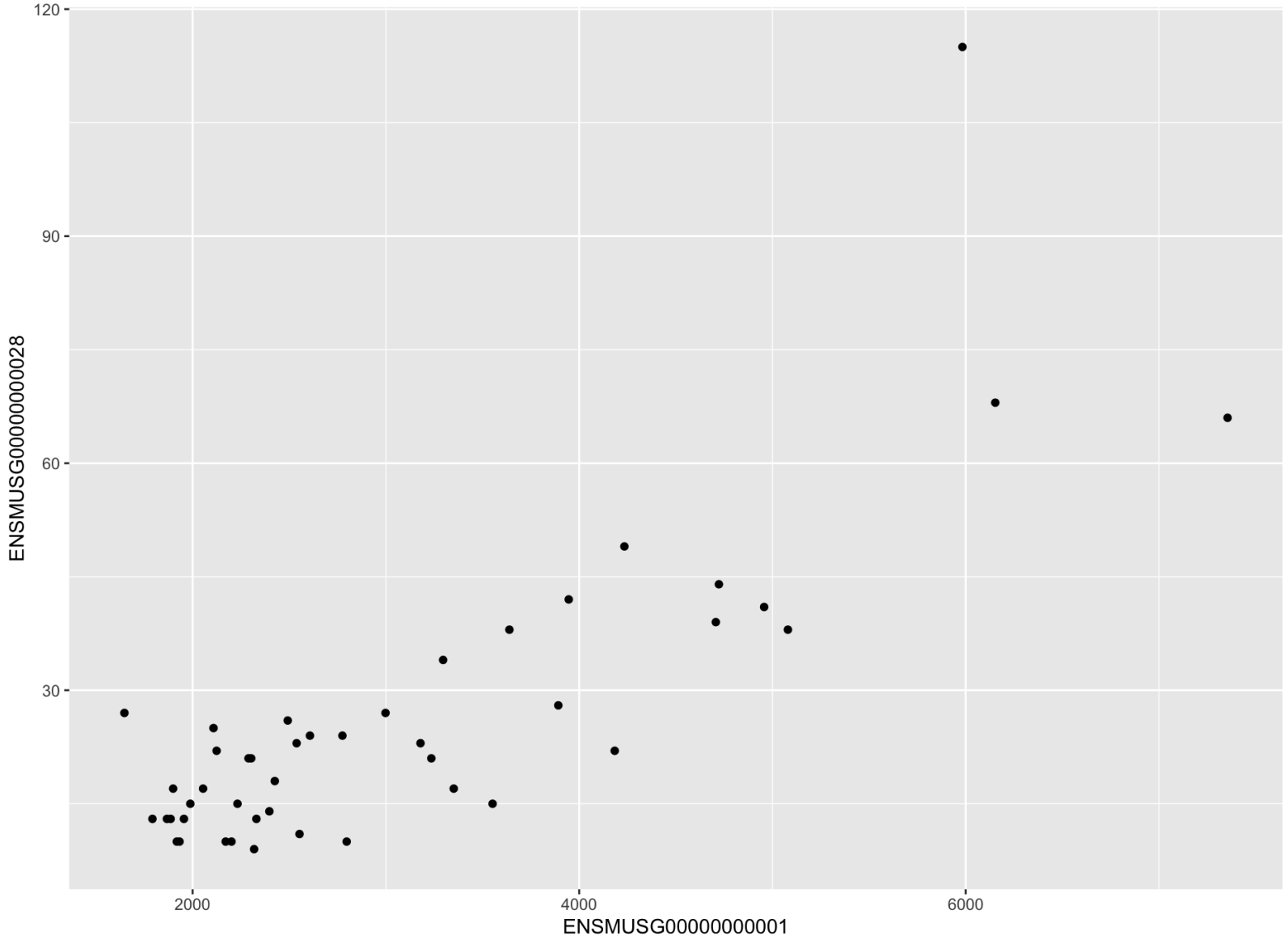
- **Once we have a clean dataset:**

```
ggplot(data=df, aes(x=ENSMUSG00000000001, y=ENSMUSG00000000028)) + geom_point()
```



TEXAS A&M UNIVERSITY
SUPERFUND
RESEARCH CENTER

Scatter Plot Example



Scatter Plot Example

- **Adding elements like the dose information:**

```
ggplot(df, aes(ENSMUSG00000000001, ENSMUSG00000000028),
               colour = CLASS.DOSE)) + geom_point()
```

column name that has the dose information in df

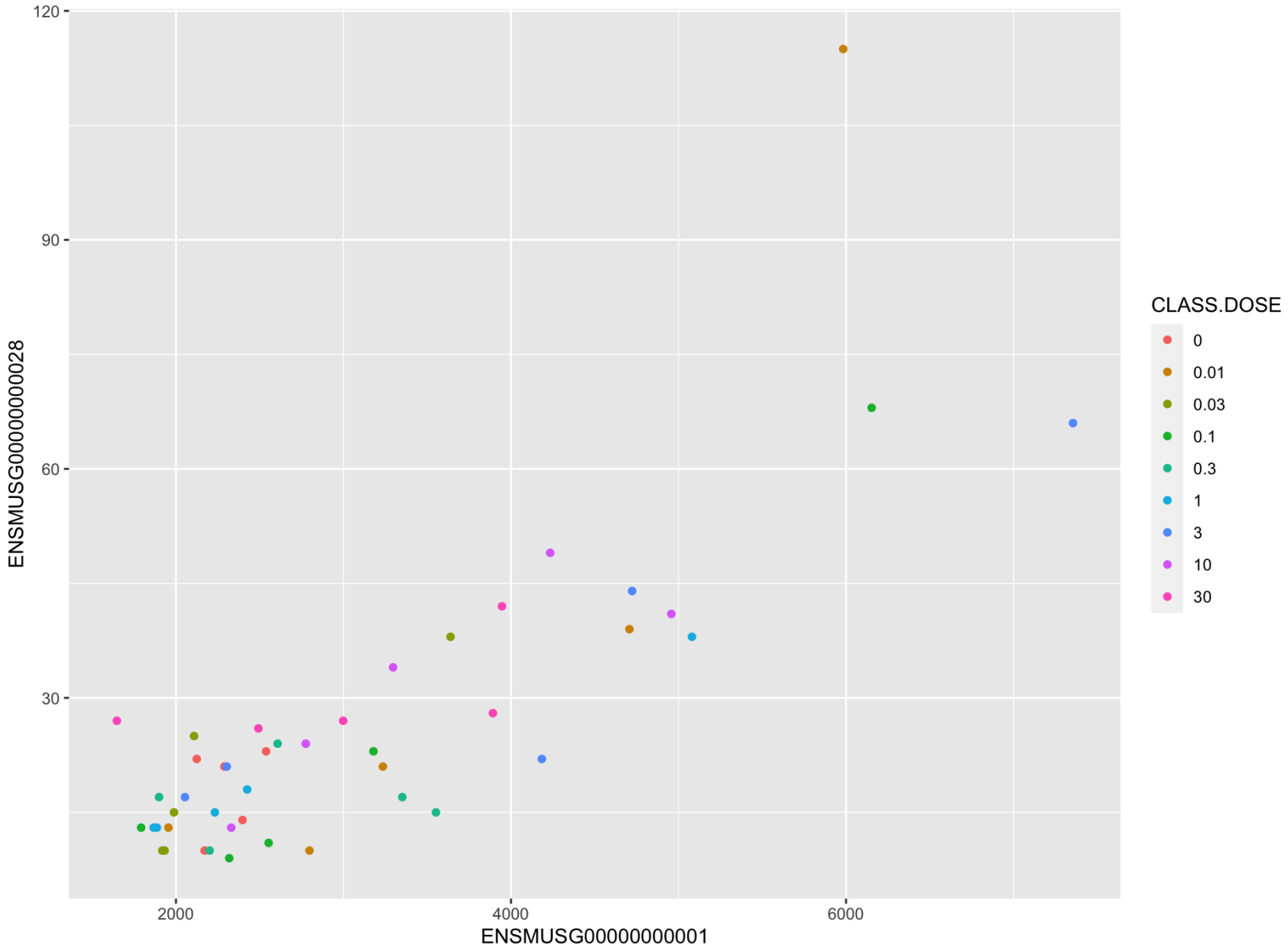
you don't need to type "data = ..." or "x =..." "y=..." in aes.

plot values with respect to their dose information and present this in different colors

if you type this as "color" it will still plot

TEXAS A&M UNIVERSITY
SUPERFUND
RESEARCH CENTER

Scatter Plot Example

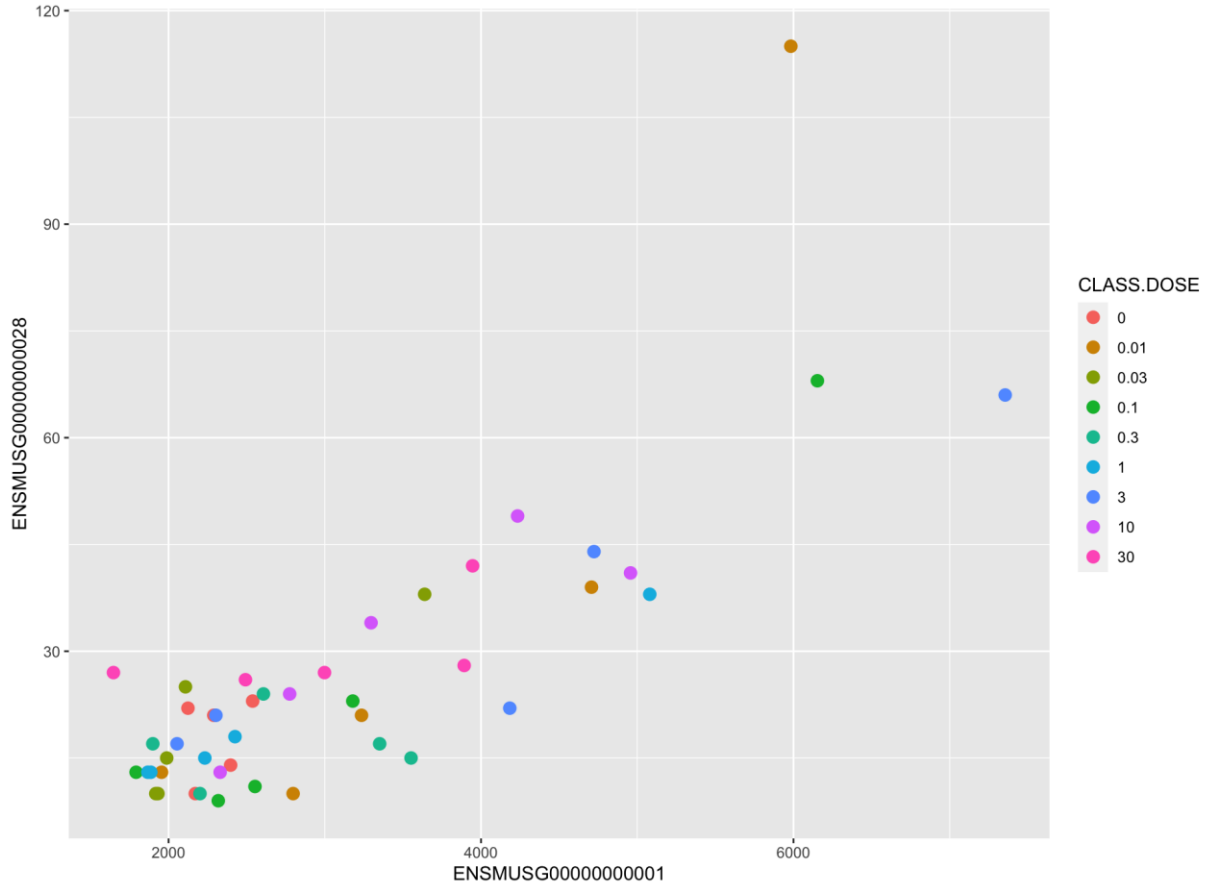


Scatter Plot Example

- **Improving visualization - increase point size**

add size element in geom_point()

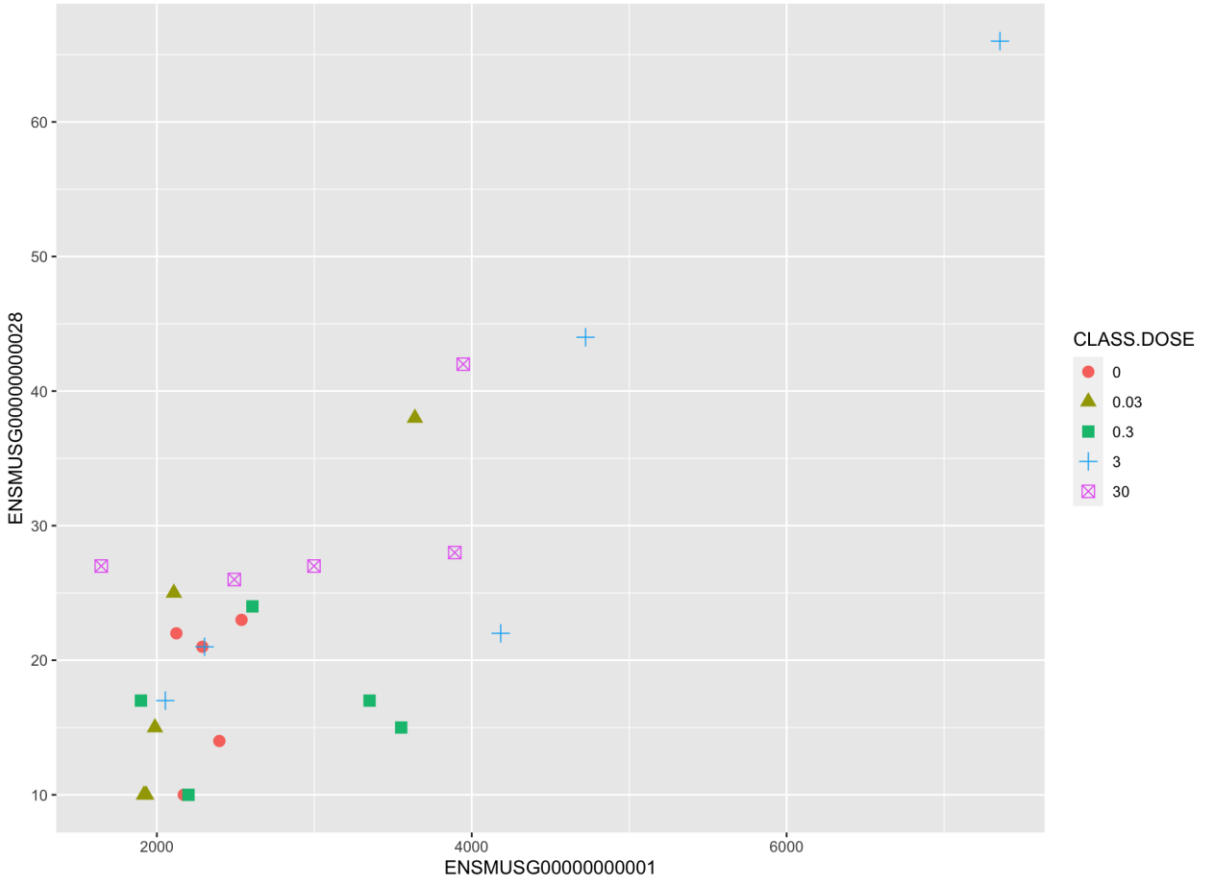
```
ggplot(df, aes(ENSMUSG00000000001, ENSMUSG00000000028),
                    colour = CLASS.DOSE)) + geom_point(size = 3)
```



Scatter Plot Example

- **Improving visualization – changing shape**

```
ggplot(df_shape, aes(ENSMUSG00000000001, ENSMUSG00000000028),
        colour = CLASS.DOSE)) + geom_point(size = 3, aes(shape=CLASS.DOSE))
```



shape is an aesthetic element. In this example, shape changes based on the class dose.

color is also an aesthetic element and can be put in geom_point() command.

TEXAS A&M UNIVERSITY SUPERFUND RESEARCH CENTER

Boxplot Example

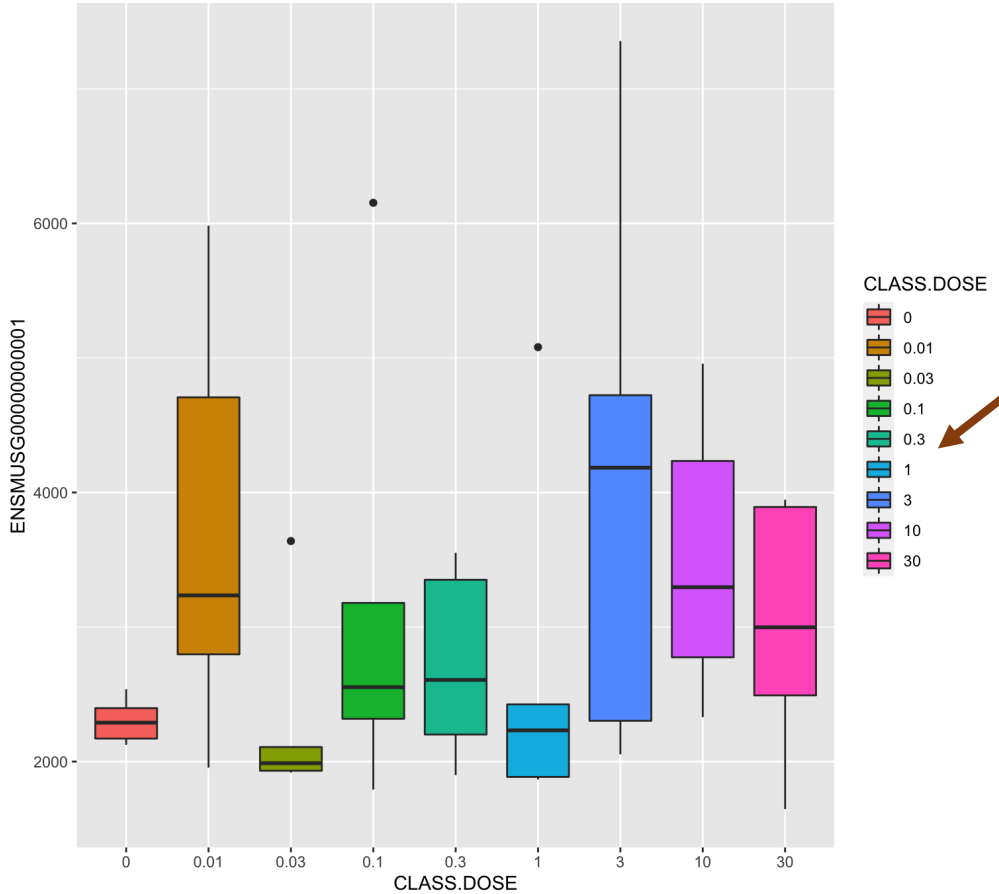
first gene is selected for the plot

- **Getting started with boxplot: Analyzing 1 Gene across doses**

```
ggplot(df, aes(x=CLASS.DOSE, y=ENSMUSG00000000001, fill=CLASS.DOSE)) +
```

geom_boxplot()

now, our geom is boxplot.

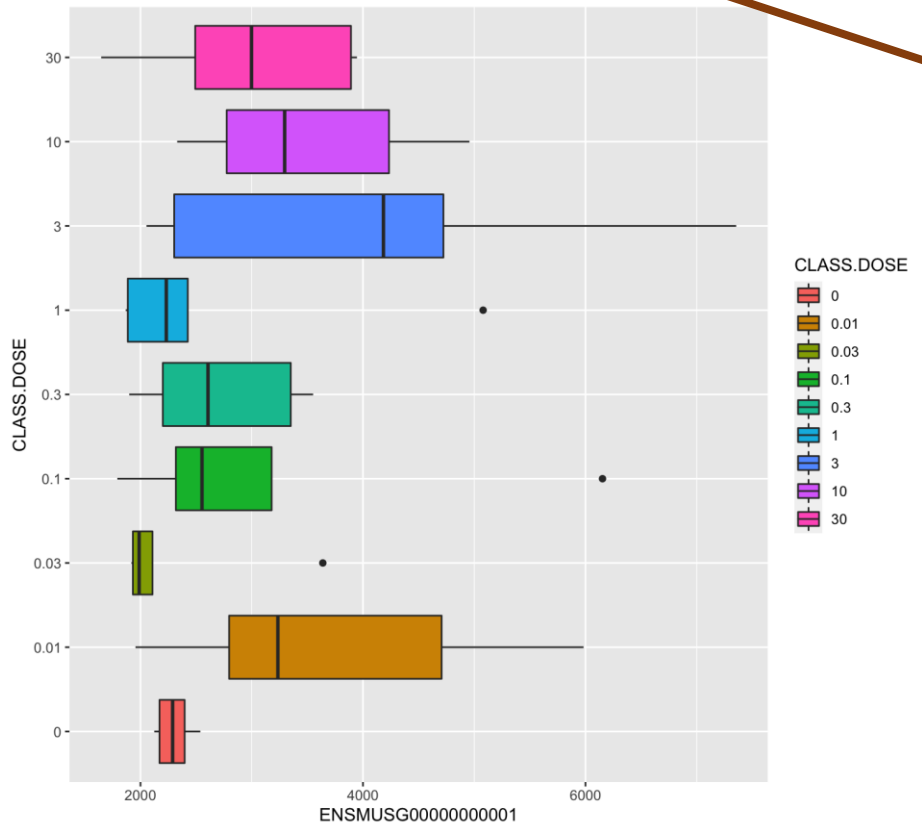


Eliminating redundant elements will improve visualization such as the legend in this example

Boxplot Example

- **Improve boxplot visualization: flip axis**

```
ggplot(df, aes(x=CLASS.DOSE, y=ENSMUSG00000000001, fill=CLASS.DOSE)) +                    geom_boxplot() + coo
```

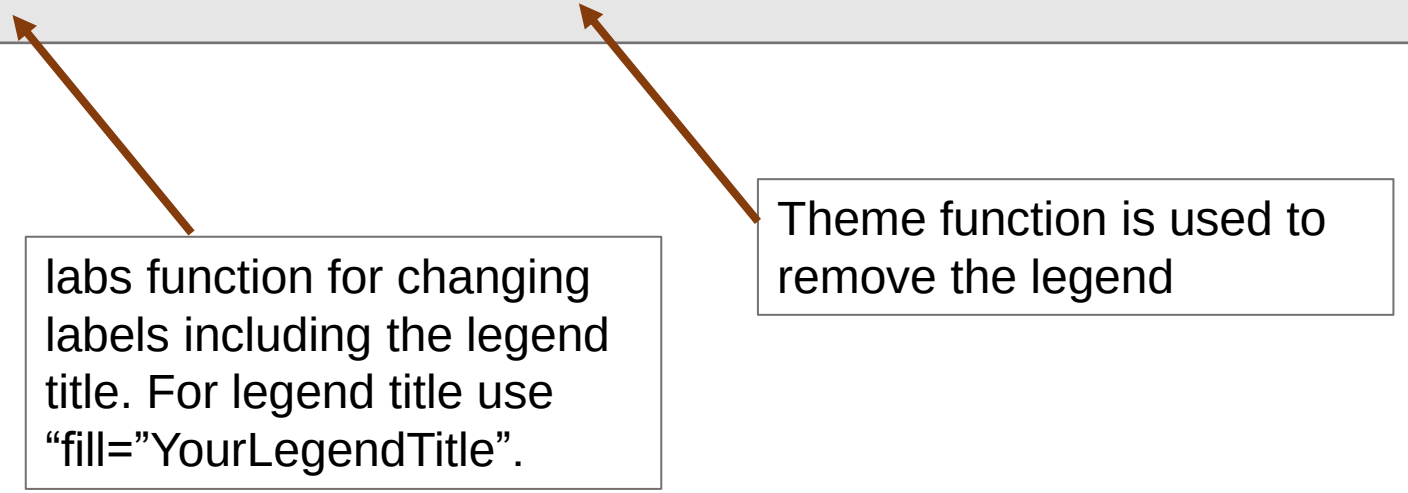


flips the coordinates. This is very useful in visualizing boxplots

Boxplot Example

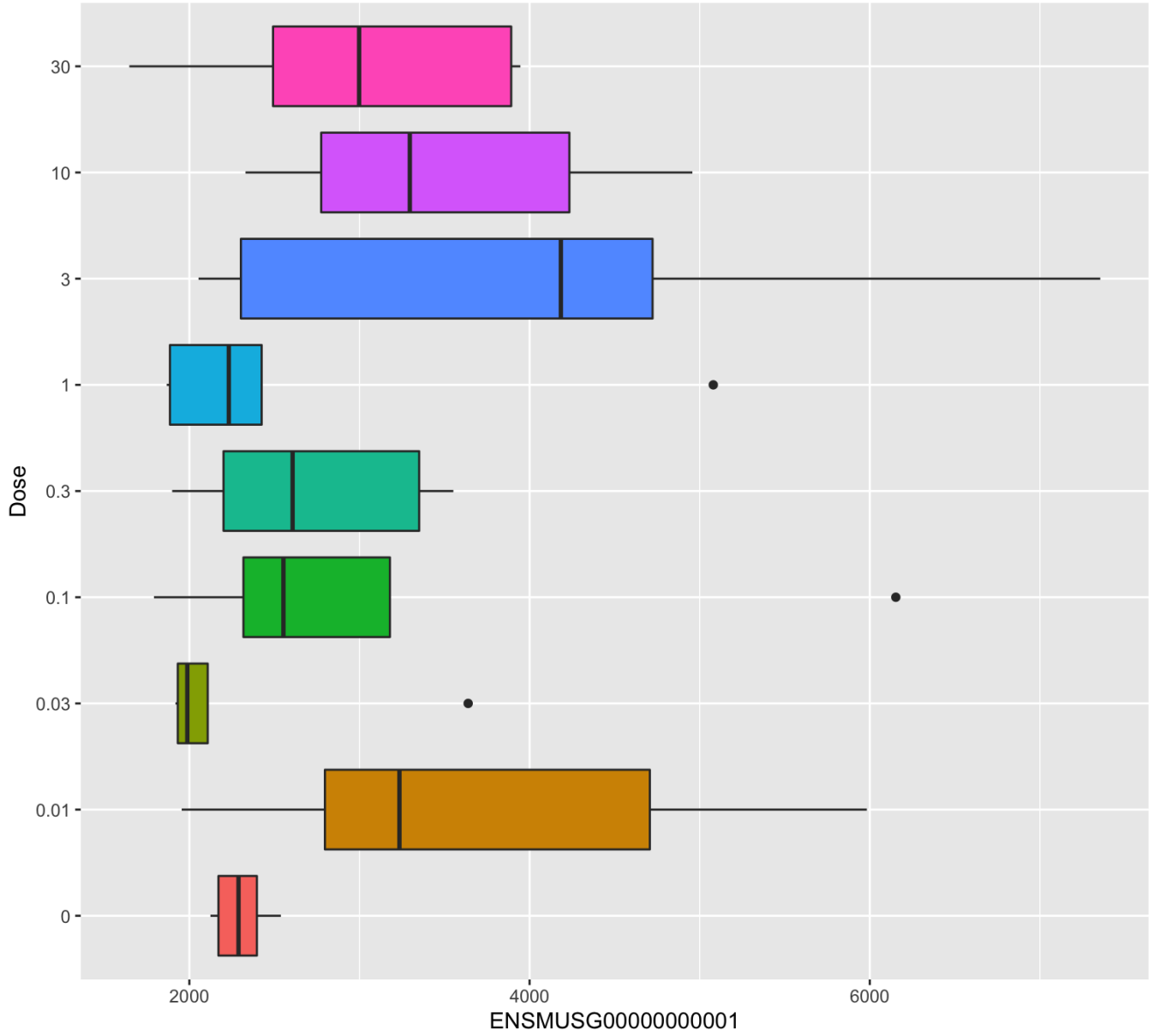
- **Improve boxplot visualization: Change x-axis label & remove legend**

```
ggplot(df, aes(x=CLASS.DOSE, y=ENSMUSG00000000001, fill=CLASS.DOSE)) +                    geom_boxplot() + coo
        labs(x = "Dose") + theme(legend.position="none")
```



TEXAS A&M UNIVERSITY
SUPERFUND
RESEARCH CENTER

Boxplot Example



There are still many ways improving this plot. Changing the background color, font styles, and sizes are some examples.

Boxplot Example

- **Improve boxplot visualization: Customizing labels and other texts in ggplot**

```
ggplot(df, aes(x=CLASS.DOSE, y=ENSMUSG00000000001, fill=CLASS.DOSE)) +                geom_boxplot() +
coord_flip() +
        labs(x = "Dose") + theme(legend.position="none",
                text = element_text(size=20),
              axis.text.x = element_text(angle=45,
                                         hjust = 1))
```

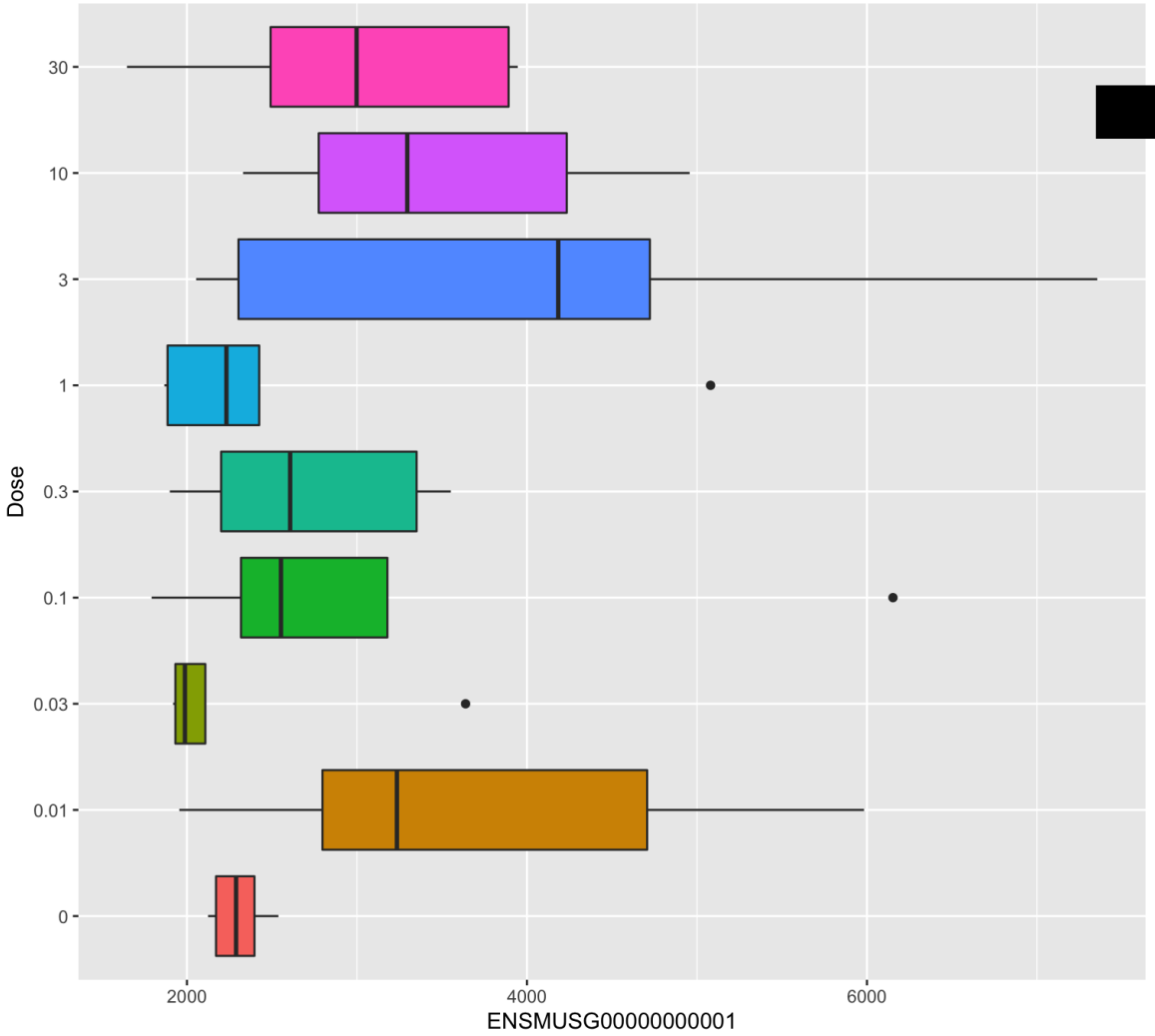
making all text sizes 20

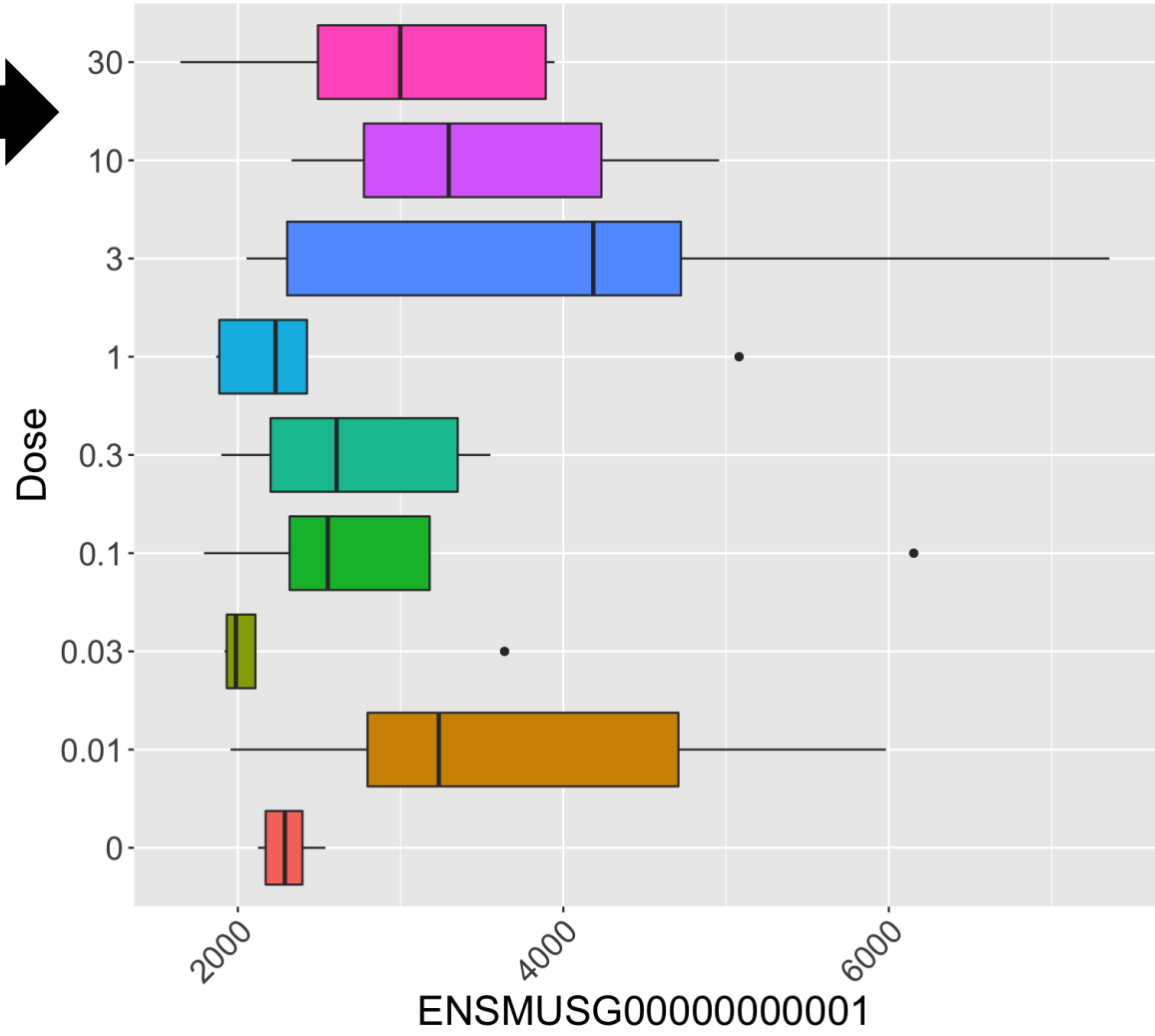
changing axis tick text angle to 45 and nudging it outward by 1 with hjust

Notice that these customizations are added under the “theme” command

TEXAS A&M UNIVERSITY
SUPERFUND
RESEARCH CENTER

Boxplot Example





Plot Customization: Using Themes in ggplot

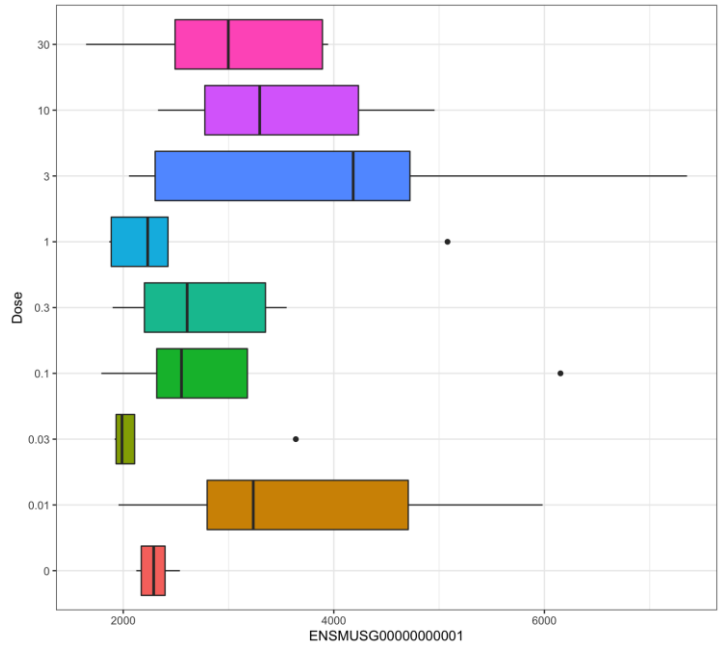
- Using the ”theme” command is a great way to customize plots.
- There are many complete themes to choose from and they are fully customizable

```
ggplot(df,aes(x=CLASS.DOSE, y=ENSMUSG00000000001, fill=CLASS.DOSE)) +
        geom_boxplot() + coord_flip() + labs(x = "Dose") +
        theme_bw() + theme(legend.position="none")
```

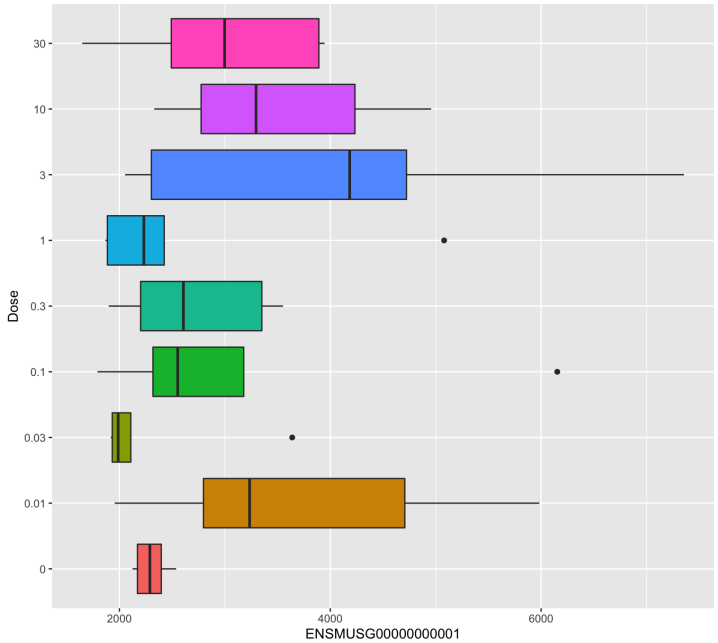
theme black and white is one of the complete themes that can be used. Note how this is placed before the “theme()” command. If this is placed after, your customizations will be overwritten by the theme_bw()

TEXAS A&M UNIVERSITY
SUPERFUND
RESEARCH CENTER

Plot Customization: Using Themes in ggplot

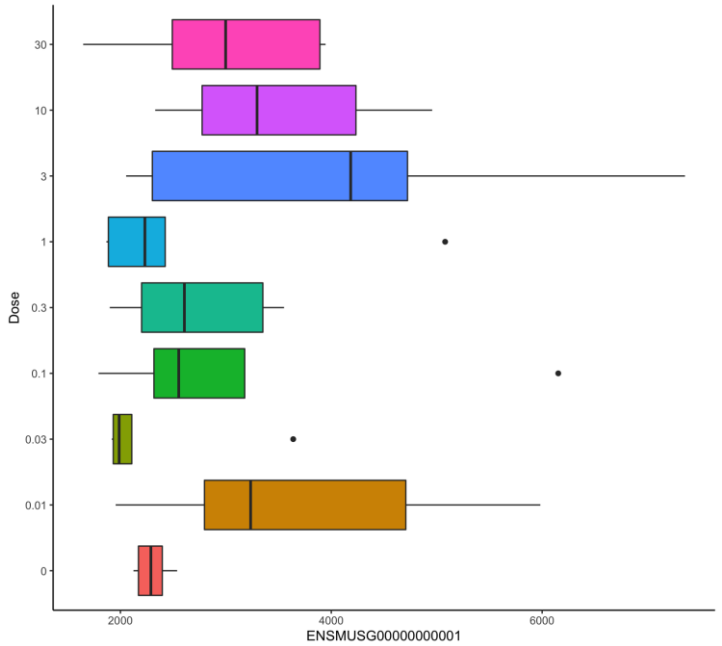


theme_bw()



theme_grey()

theme_classic()



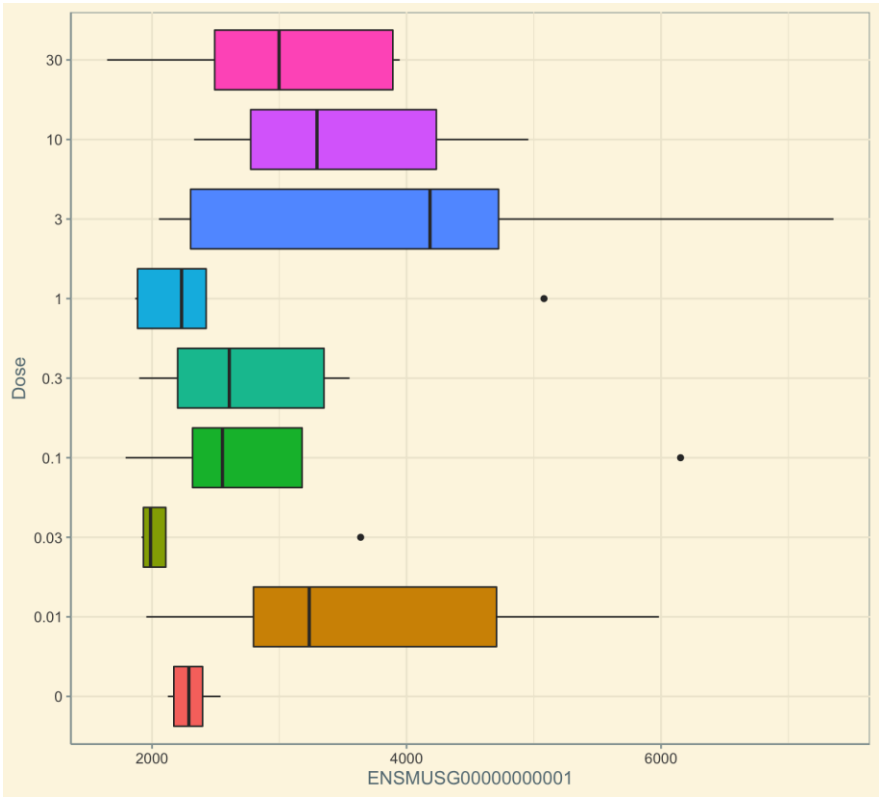
Plot Customization: Using Themes in ggplot

library(ggthemes) → custom themes library

You may get a warning but it's ok → Warning message:
package 'ggthemes' was built under R version 4.0.2

Try new themes from the library

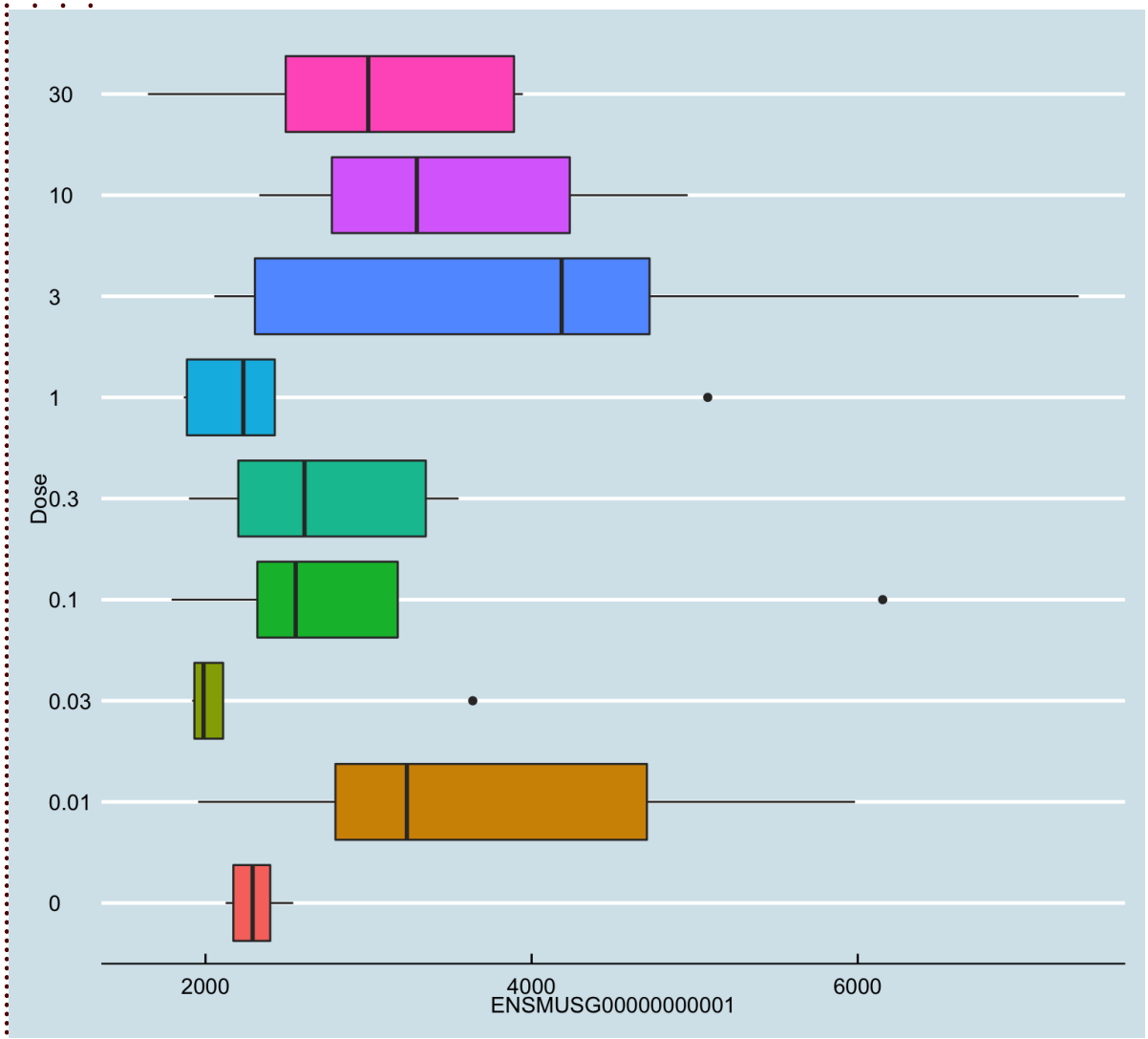
theme_solarized()



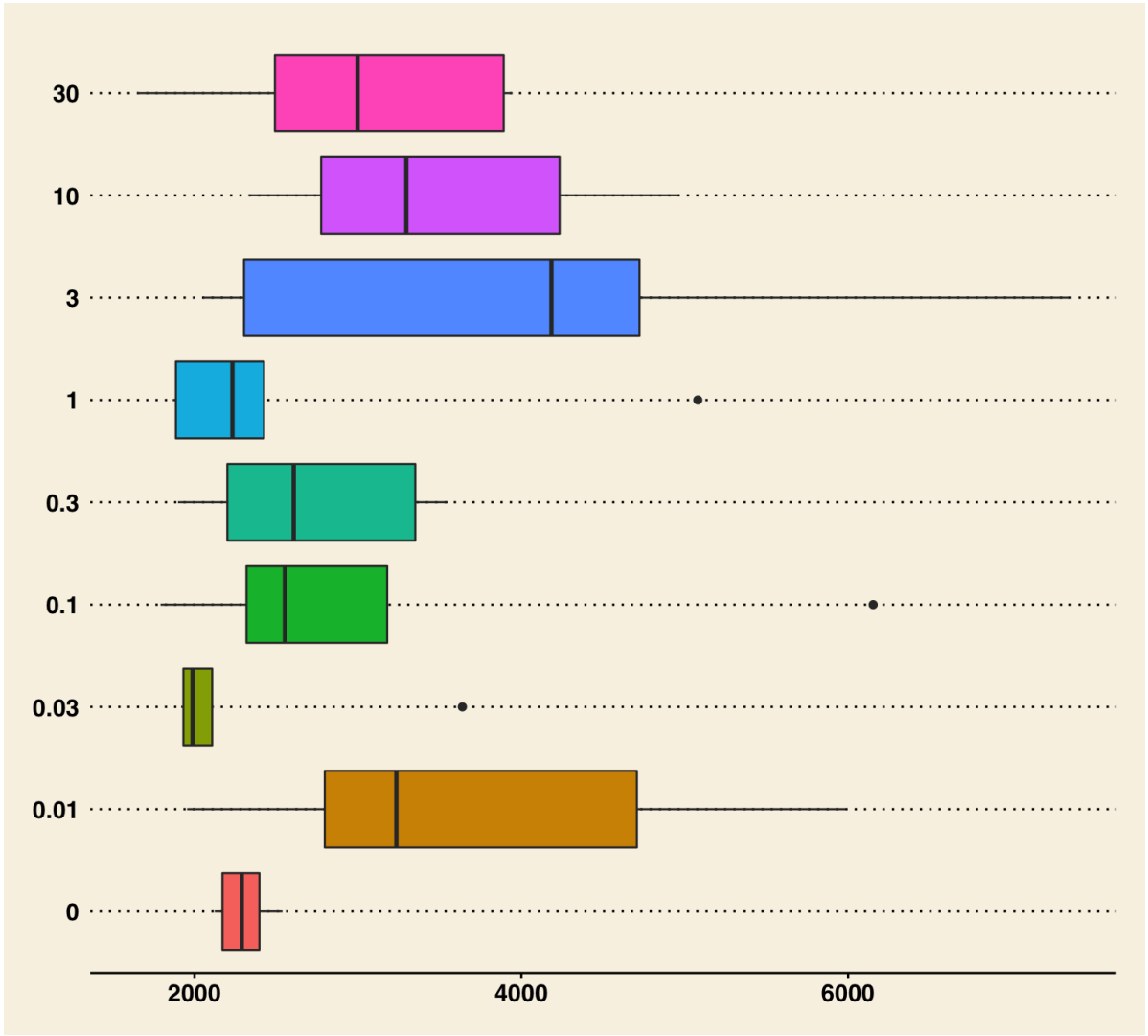
TEXAS A&M UNIVERSITY SUPERFUND RESEARCH CENTER

Plot Customization: Using Themes in ggplot

theme_economist()



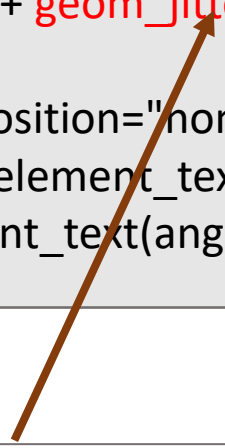
theme_wsj()



Adding Individual Observations on the Boxplot

- **Improve boxplot visualization: Overlaying samples on to the plot**

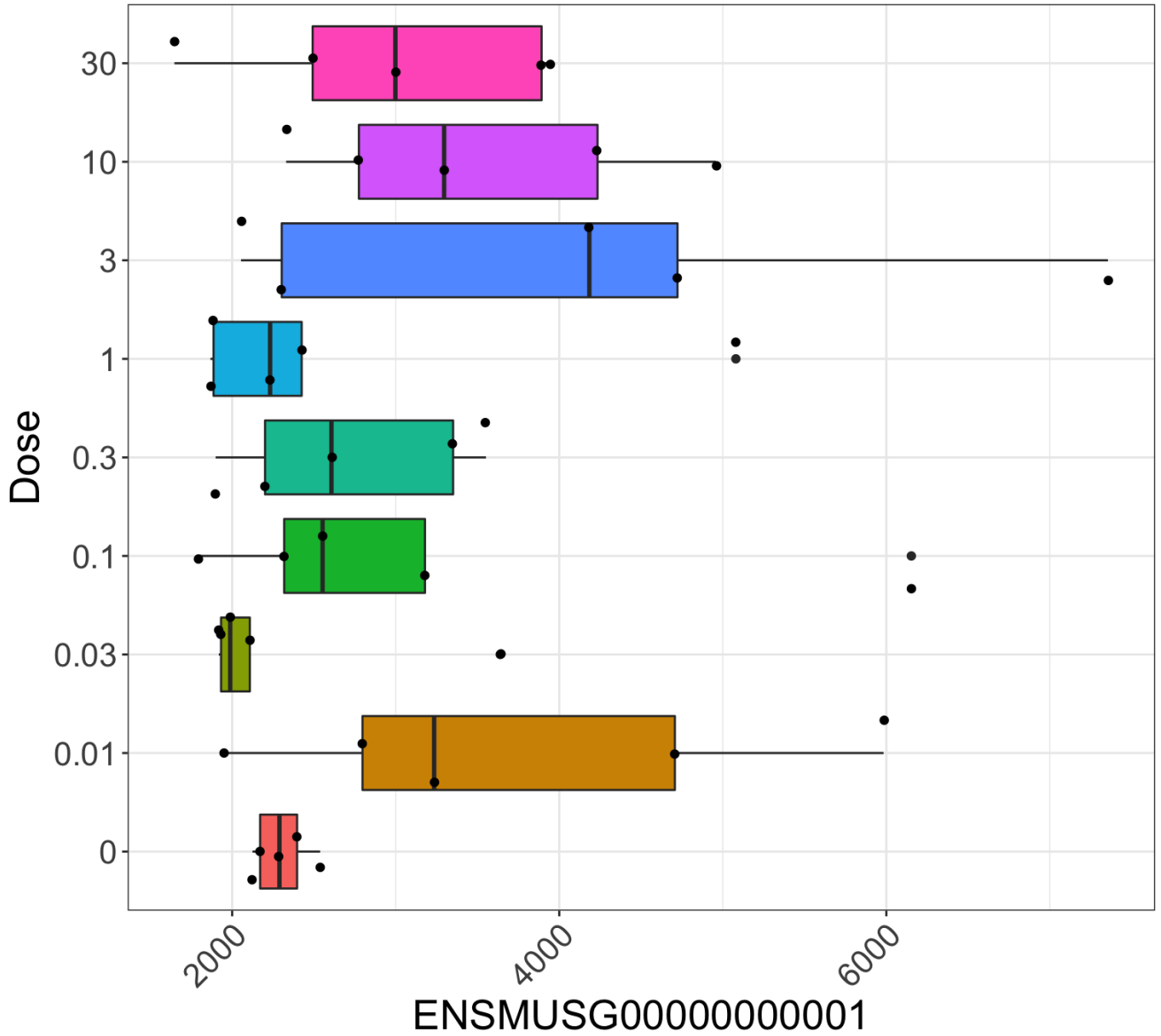
```
ggplot(df,aes(x=CLASS.DOSE, y=ENSMUSG00000000001, fill=CLASS.DOSE)) +
        geom_boxplot() + geom_jitter(color="black") + coord_flip() +        labs(x = "Dose") +
theme_bw() +
        theme(legend.position="none",
                text = element_text(size=20),
      axis.text.x = element_text(angle=45, hjust = 1))
```



geom_jitter will add the samples used to construct the boxplot.
You can specify preferred colors to improve visualization.

TEXAS A&M UNIVERSITY
SUPERFUND
RESEARCH CENTER

Adding Individual Observations on the Boxplot



TEXAS A&M UNIVERSITY SUPERFUND RESEARCH CENTER

Changing Colors in ggplot

- **Improve boxplot visualization: Adding color elements**

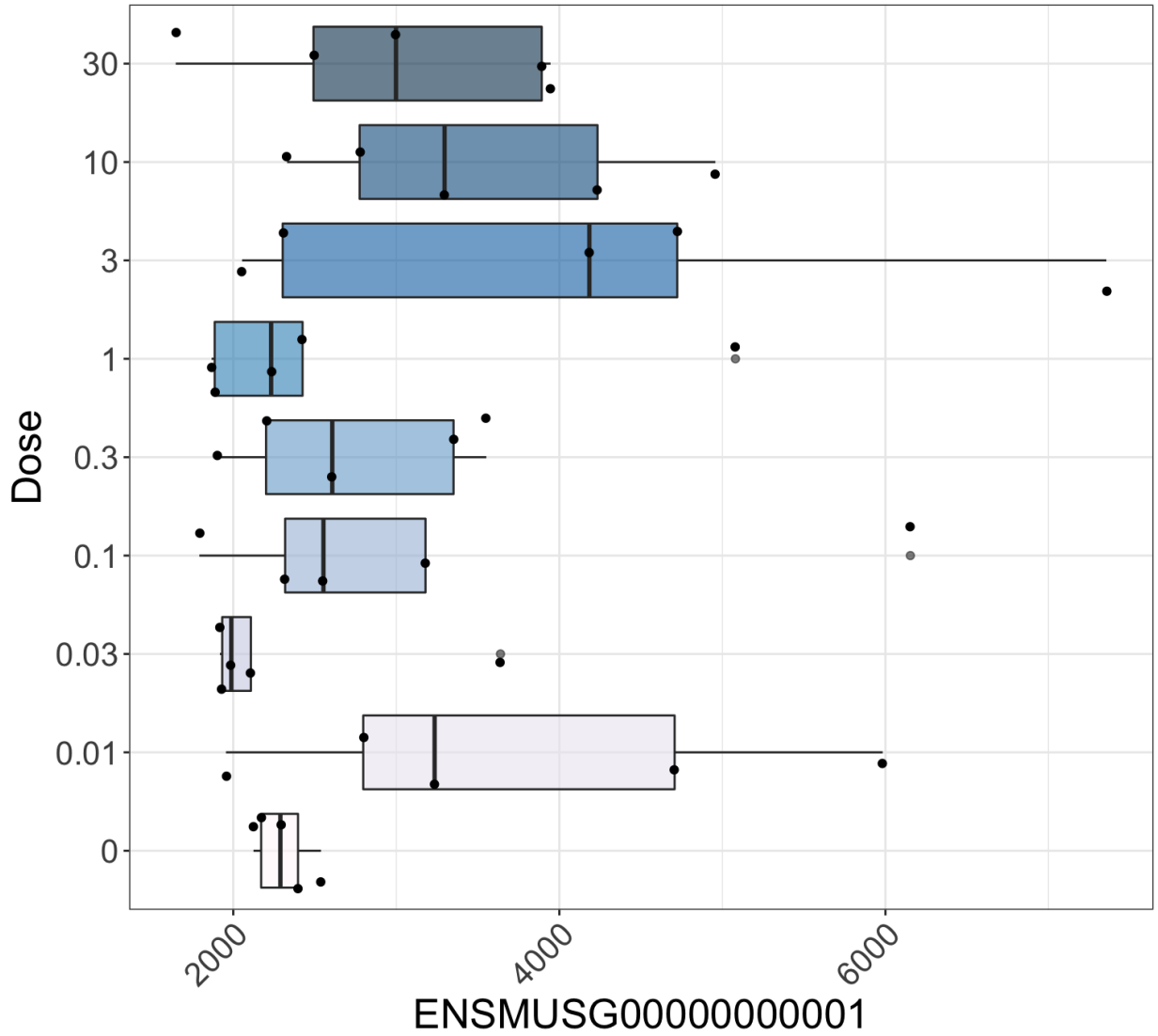
```
ggplot(df,aes(x=CLASS.DOSE, y=ENSMUSG00000000001, fill=CLASS.DOSE)) +
        geom_boxplot(alpha=0.6) + geom_jitter(color="black") +    coord_flip() + labs(x = "Dose") +
theme_bw() +
        theme(legend.position="none",
                text = element_text(size=20),
      axis.text.x = element_text(angle=45, hjust = 1)) +
        scale_fill_brewer(palette="PuBu")
```

you can adjust the transparency in bloxplot fill colors

R Color Brewer library have default sets of colors available

TEXAS A&M UNIVERSITY
SUPERFUND
RESEARCH CENTER

Changing Colors in ggplot



TEXAS A&M UNIVERSITY SUPERFUND RESEARCH CENTER

Choosing Colors in R

- Colors in R are defined using hexadecimals.

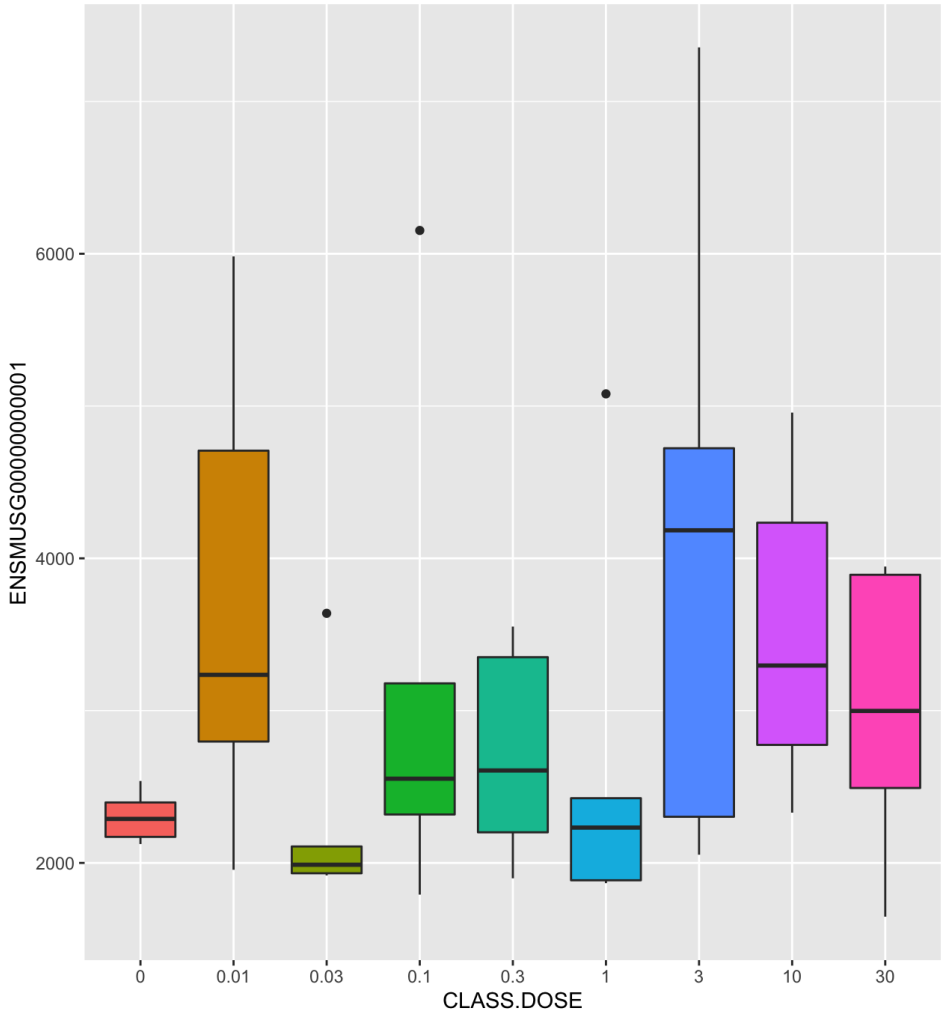
- **Colors in R document**
  - list of color names and options to chose from
  - http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf

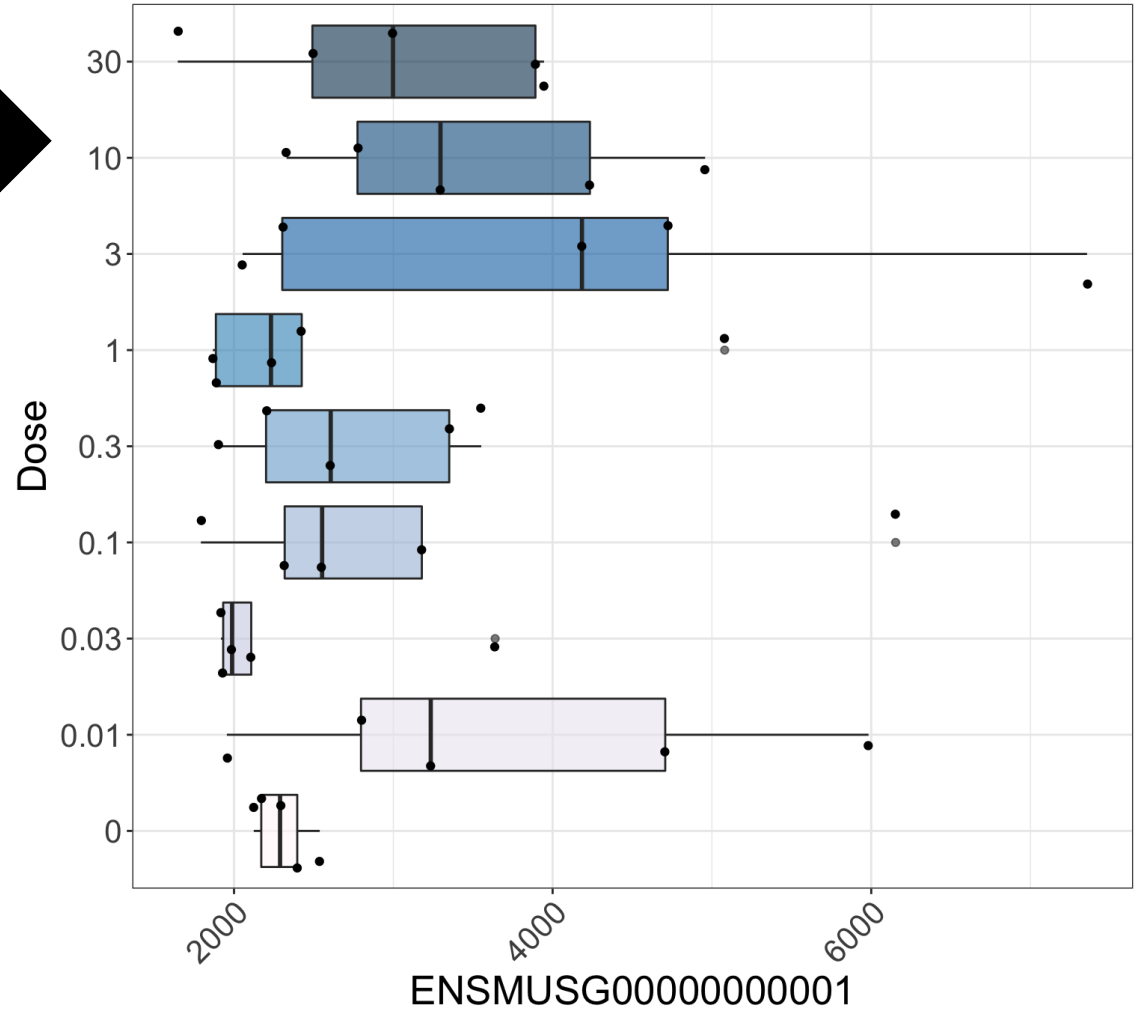
- R Color Cheatsheet document
  - https://www.nceas.ucsb.edu/sites/default/files/2020-04/colorPaletteCheatsheet.pdf
  - Explains how colors are defined in R
  - Calling colors using RColorBrewer library

- For color advice:
  - https://colorbrewer2.org/#type=sequential&scheme=Greens&n=3
  - Great for maps and other plot types

TEXAS A&M UNIVERSITY
SUPERFUND
RESEARCH CENTER

Final Boxplot





TEXAS A&M UNIVERSITY SUPERFUND RESEARCH CENTER

Density Plots

- **Create density plot of measurements across doses:**

use measurements for this gene

```
ggplot(df, aes(x=ENSMUSG00000000049, group=CLASS.DOSE, fill=CLASS.DOSE)) +
        geom_density() +
        labs(fill='Dose') +
        theme_bw() +
        theme(text = element_text(size=20)) +
        scale_fill_brewer(palette="Reds")
```

density geom command

legend title should display ‘Dose’

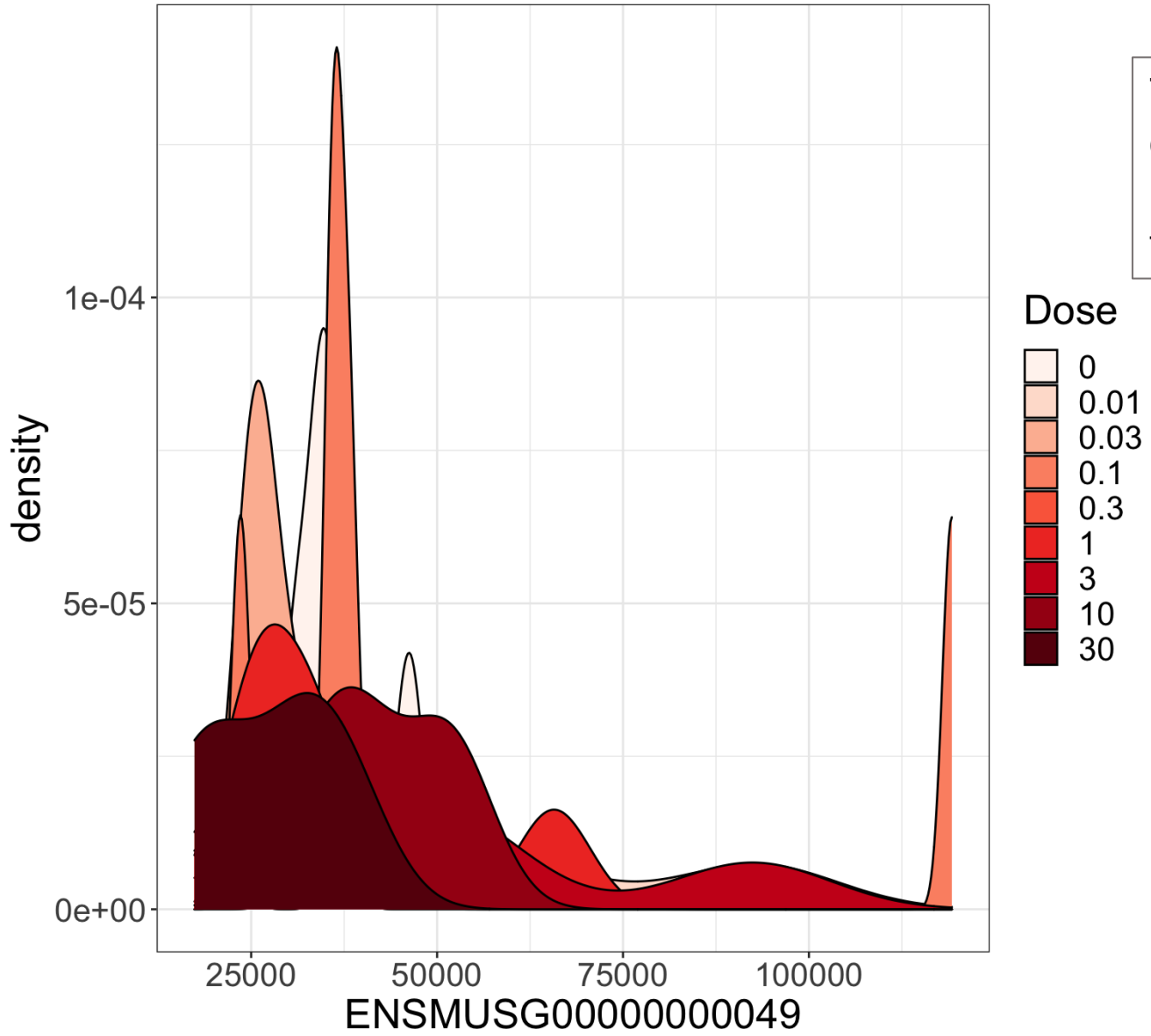
use the ”Reds” color palette for coloring

use measurements from each dose

color with respect to dosage

TEXAS A&M UNIVERSITY SUPERFUND RESEARCH CENTER

Density Plots



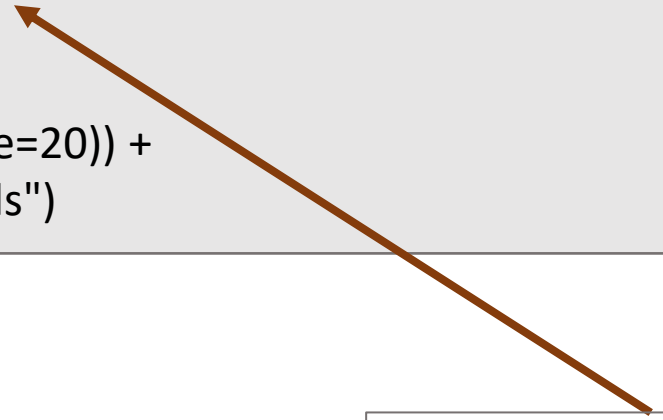
Too much overlap is obstructing view. Let's increase the colors transparency.

TEXAS A&M UNIVERSITY SUPERFUND RESEARCH CENTER

Density Plots

- **Let's change transparency to visualize better:**

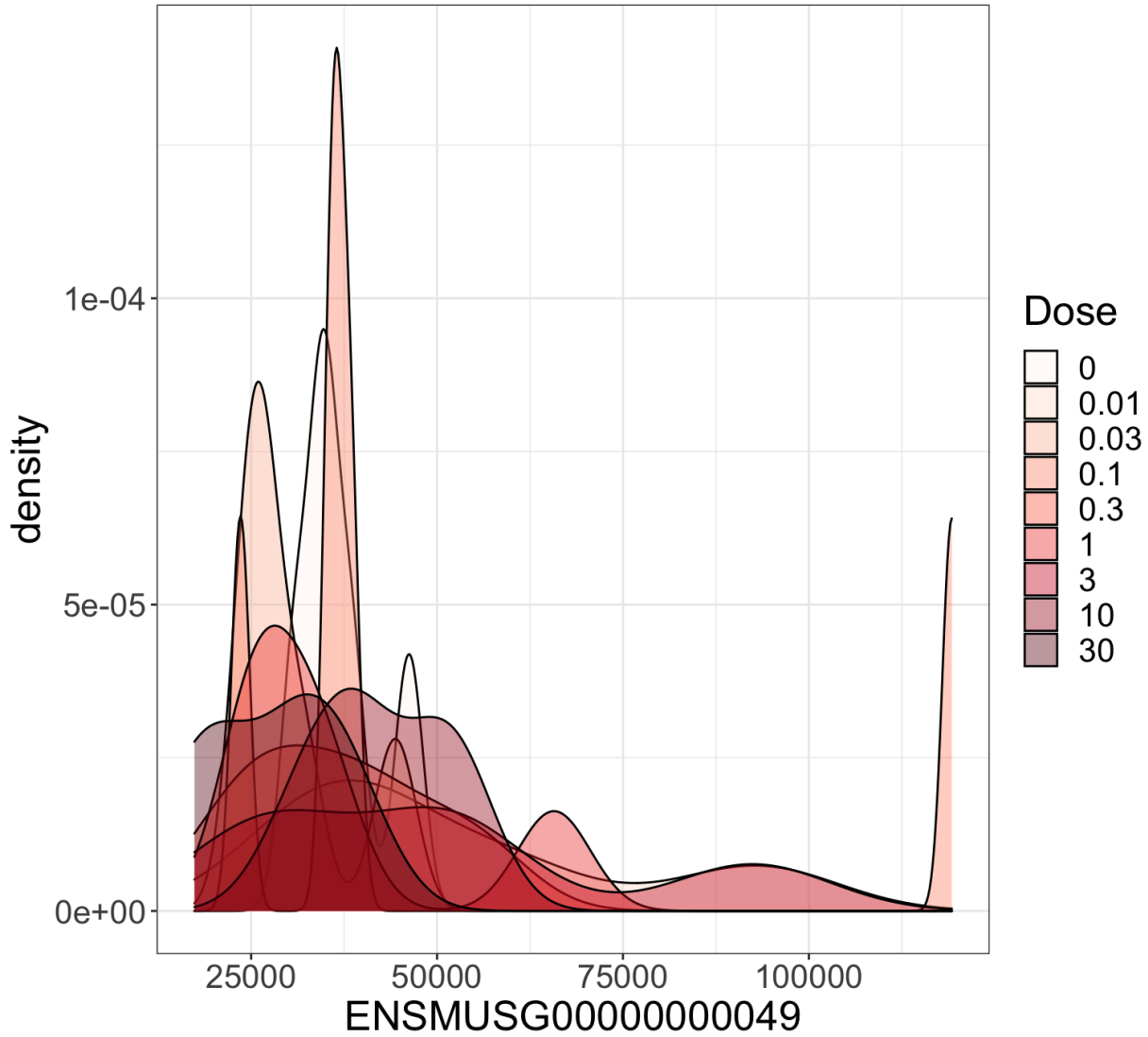
```
ggplot(df, aes(x=ENSMUSG00000000049, group=CLASS.DOSE, fill=CLASS.DOSE)) +
        geom_density(alpha=0.4) +
        labs(fill='Dose') +
        theme_bw() +
        theme(text = element_text(size=20)) +
        scale_fill_brewer(palette="Reds")
```



making fill colors transparent

TEXAS A&M UNIVERSITY
SUPERFUND
RESEARCH CENTER

Density Plots



Better visualization, yet still too much overlap.

TEXAS A&M UNIVERSITY SUPERFUND RESEARCH CENTER

Faceting Plots

- **Faceting plots with respect to dosage:**

```
ggplot(df, aes(x=ENSMUSG00000000049, group=CLASS.DOSE, fill=CLASS.DOSE)) +
        geom_density() +
        theme_bw() +
        facet_wrap(~CLASS.DOSE) +
        theme(text = element_text(size=14),
        legend.position="none",
        panel.spacing = unit(0.1, "lines")) +
        scale_fill_brewer(palette="Reds")
```

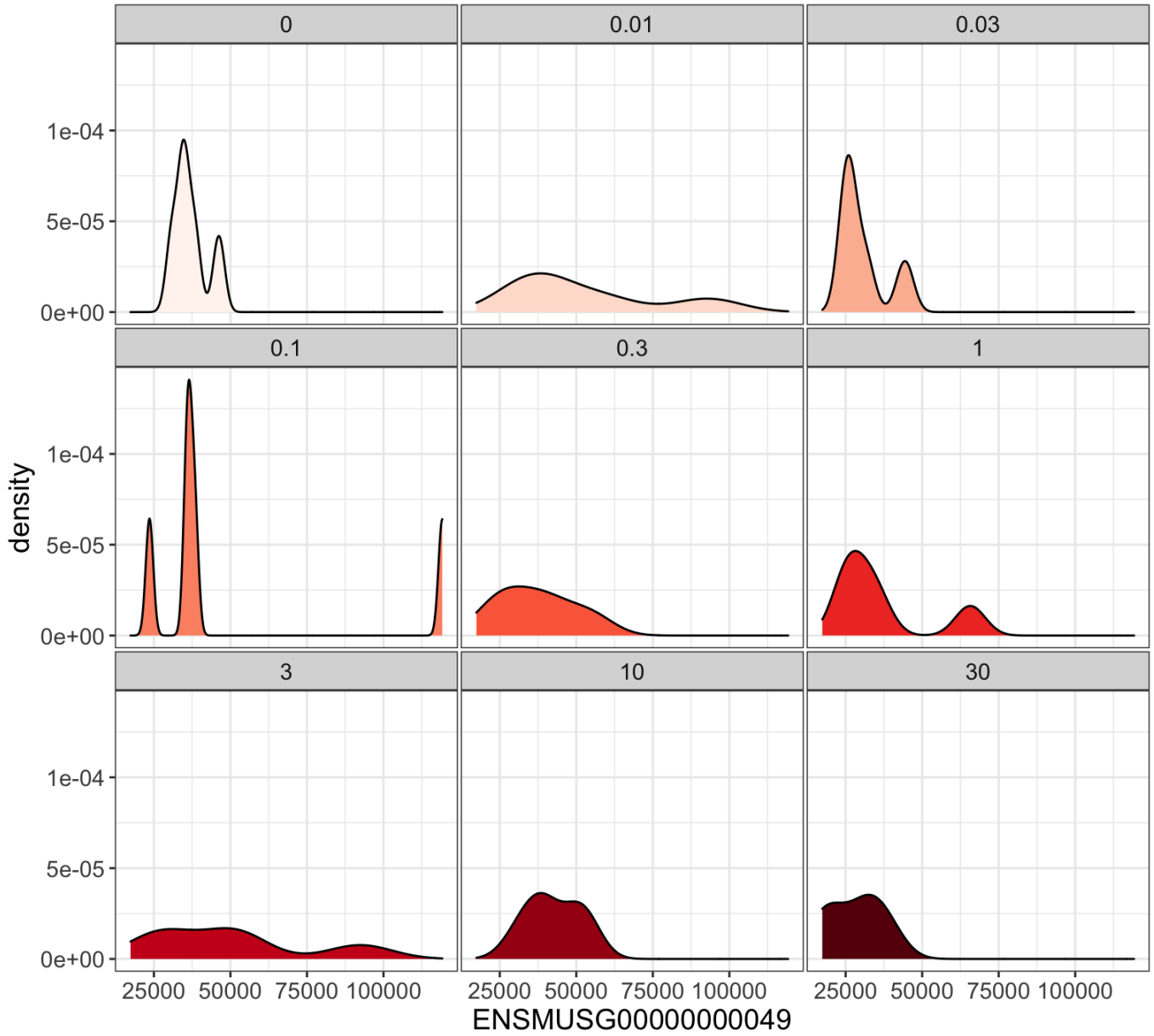
divide up the plot with respect to dosage

remove legend because it is redundant

adjust the panel spacing between faceted graphs

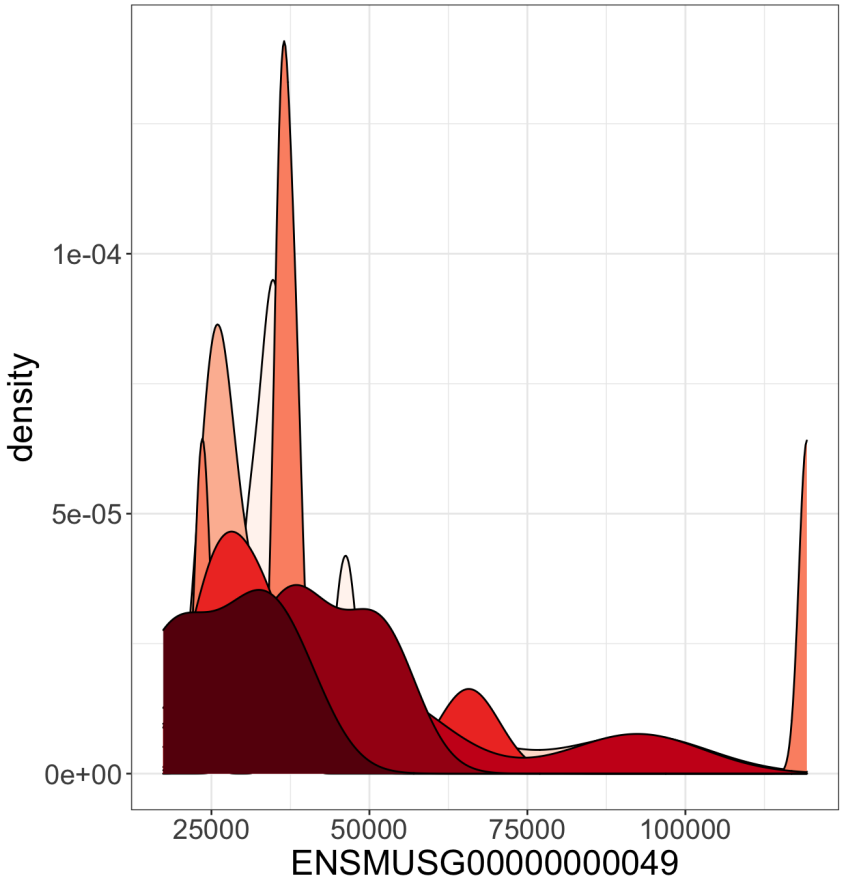
TEXAS A&M UNIVERSITY
SUPERFUND
RESEARCH CENTER

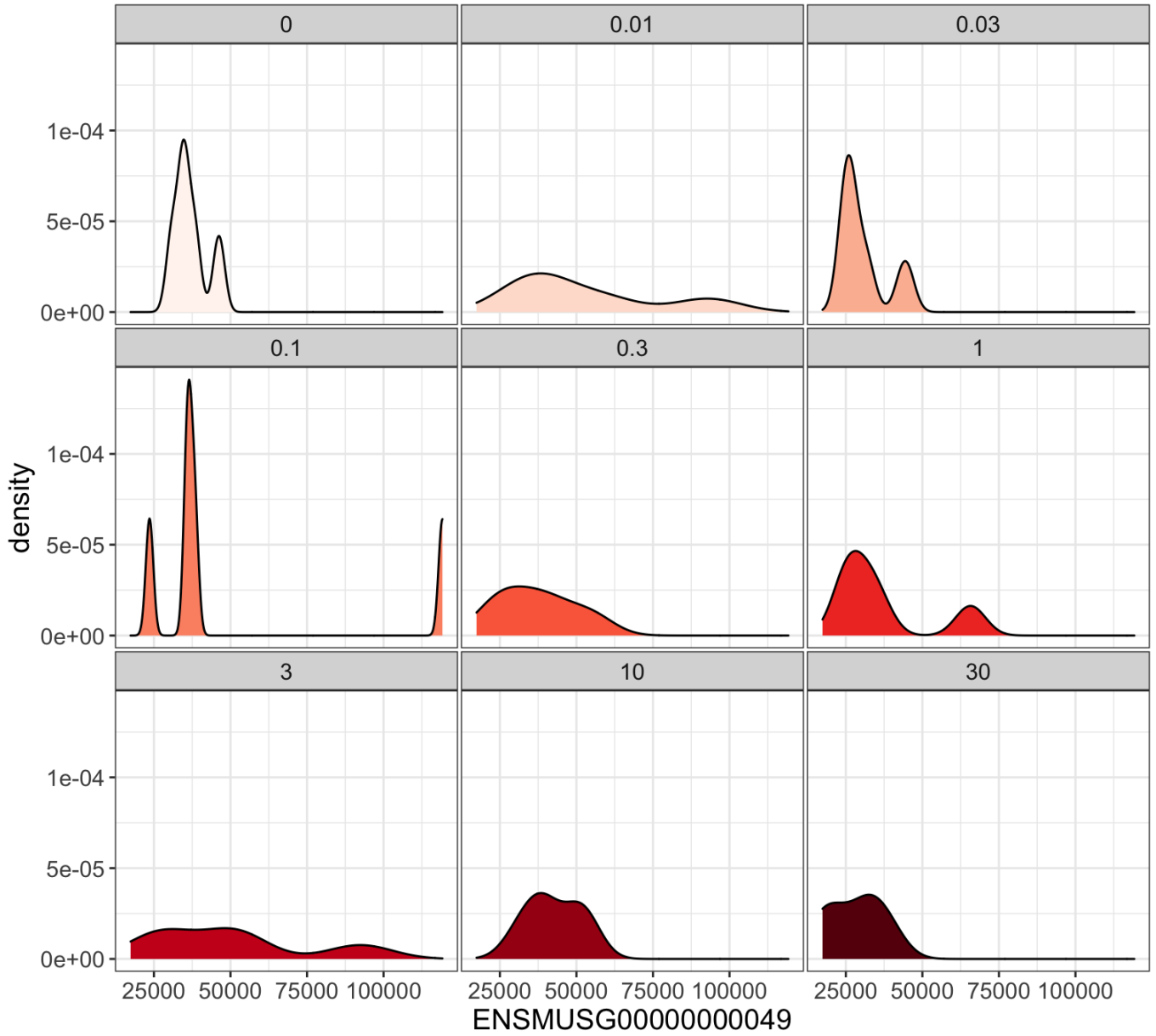
Faceting Plots



TEXAS A&M UNIVERSITY
SUPERFUND
RESEARCH CENTER

Faceting Plots





TEXAS A&M UNIVERSITY SUPERFUND RESEARCH CENTER

Plotting Spatial Data

- **Creating maps in R using ggplot2:**

using the built-in map data. You can also get shape files from government/census websites

```
usa <- map_data("usa")
```

Longitude

Latitude

dataset

```
ggplot(usa, aes(x=long, y = lat, group = group)) + geom_polygon() +
  coord_fixed(1.3)
```

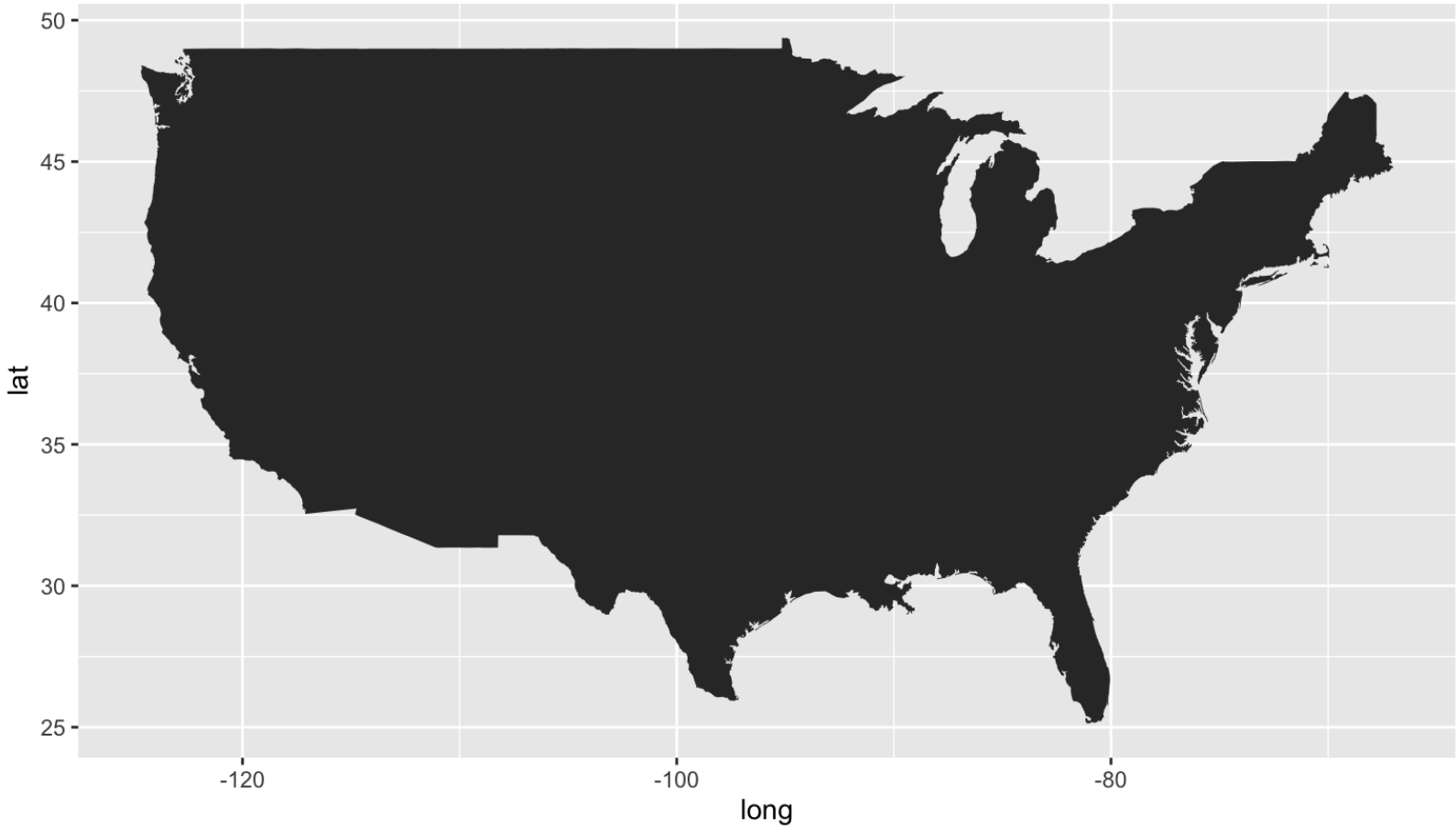
aspect ratio for the map

generating maps using polygons

TEXAS A&M UNIVERSITY
SUPERFUND
RESEARCH CENTER

Plotting Spatial Data

- **Creating maps in R using ggplot2:**



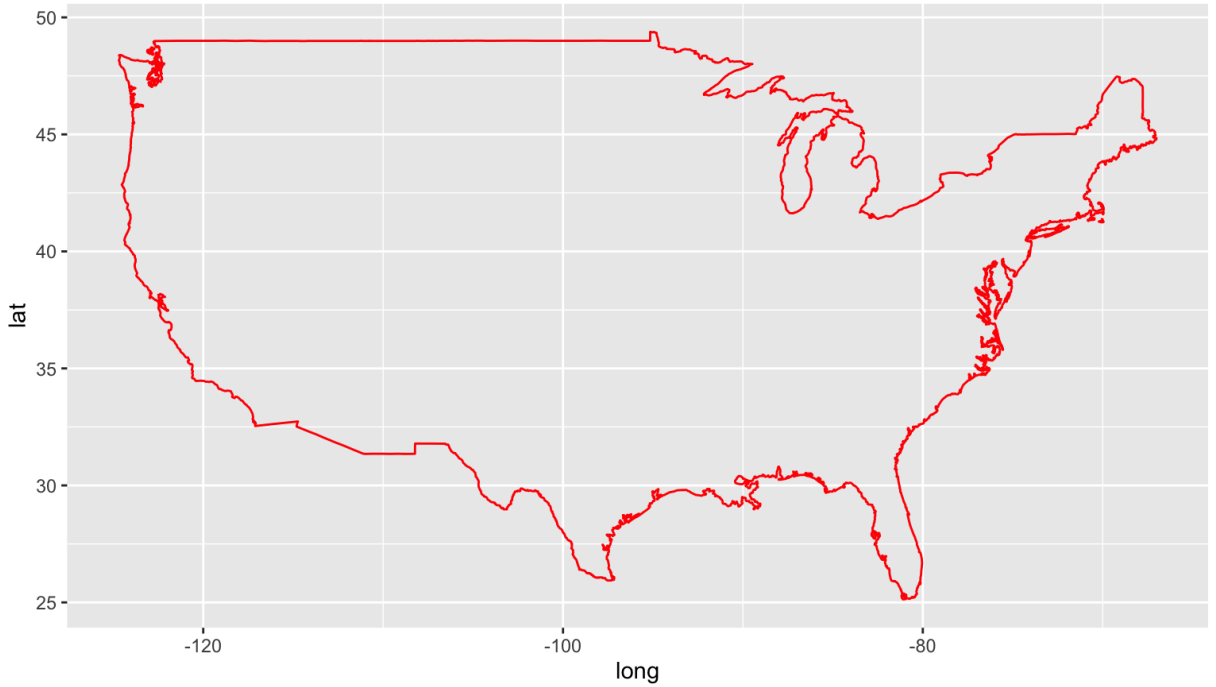
Plotting Spatial Data

- **Remove the fill color and change border color:**

```
ggplot(usa, aes(x=long, y = lat, group = group)) +
        geom_polygon(fill = NA, color = "red") + coord_fixed(1.3)
```

fill is assigned as NA

border color assigned as red



Plotting Spatial Data

- **Getting states on the map:**

```
states <- map_data("state")
head(states)
```

get states information and check the data frame

fill colors with respect to different regions/states

```
ggplot(states, aes(x = long, y = lat, fill = region, group = group)) +
  geom_polygon(color = "white") + coord_fixed(1.3) + labs(title = "US States", subtitle = "This is a blank map of the states.") +
  theme(legend.position="none")
```

polygons are used to draw the white boundaries between states

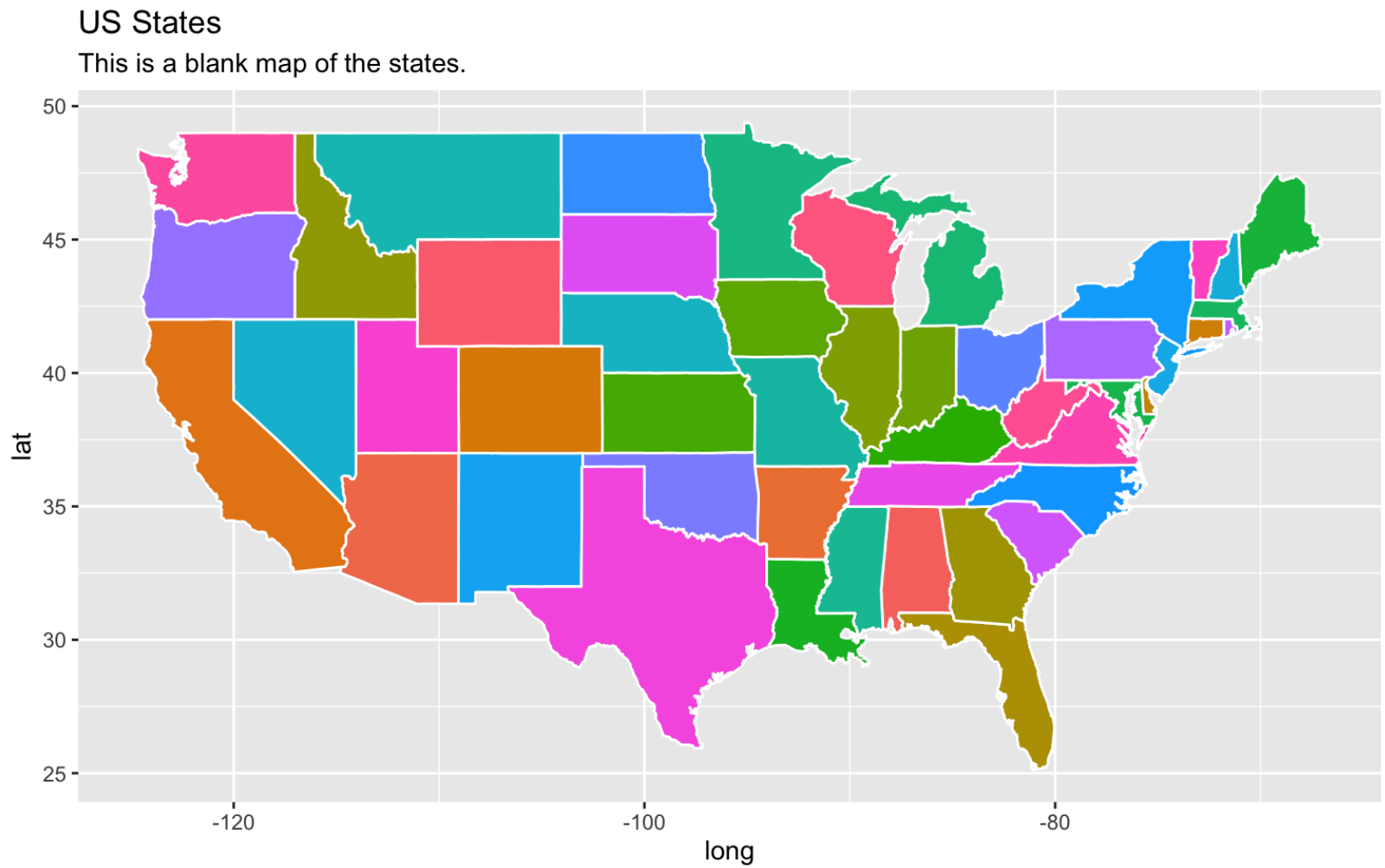
removing the legend

you can add titles and subtitles to your plot

TEXAS A&M UNIVERSITY
SUPERFUND
RESEARCH CENTER

Plotting Spatial Data

- **Getting states on the map:**



Plotting Spatial Data

- **Isolating a state for plotting:**

subset a state based on the region information

```
ca_df <- subset(states, region == "california")
counties <- map_data("county")
ca_county <- subset(counties, region == "california")
head(ca_county)
```

extract county information from the function and subset only for California

California dataset goes to ggplot command

```
ggplot(ca_df, aes(x = long, y = lat, group = group)) +
        geom_polygon(data = ca_county, fill = "gray", color = "black") +        coord_fixed(1.3) + theme_bw()
```

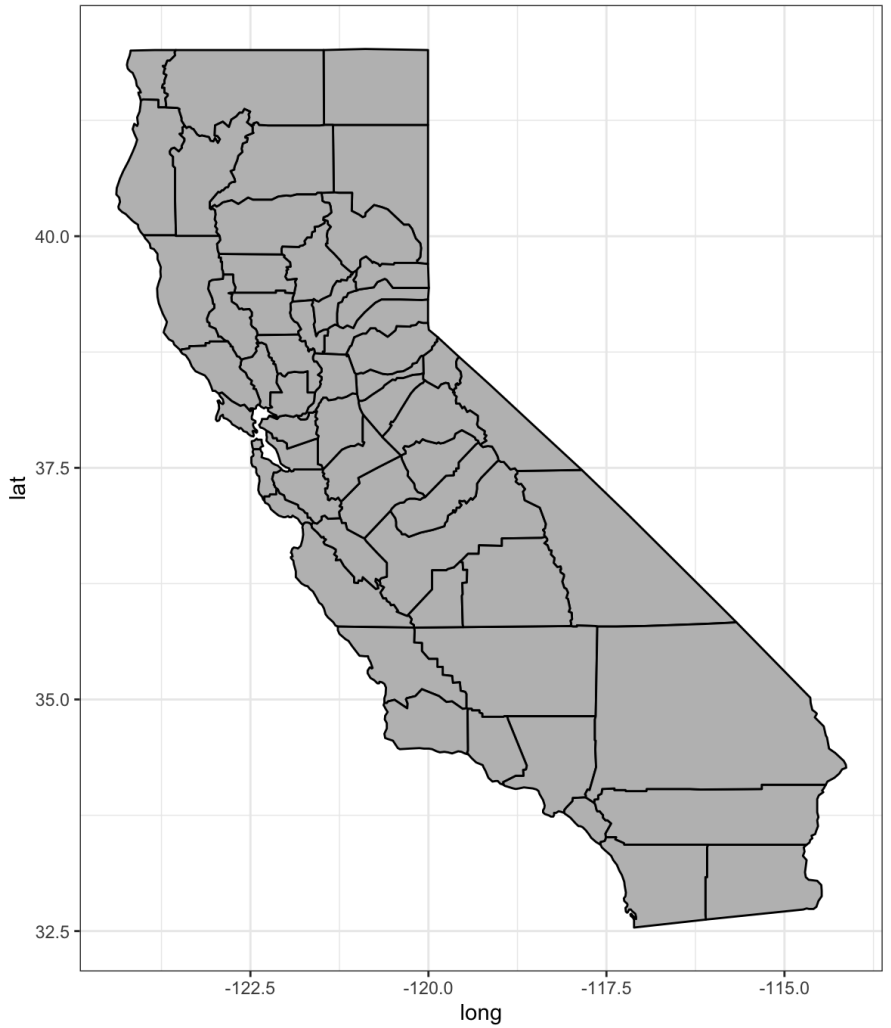
theme element for quick formatting

county information is fed to geom_polygon for drawing county borders

TEXAS A&M UNIVERSITY
SUPERFUND
RESEARCH CENTER

Plotting Spatial Data

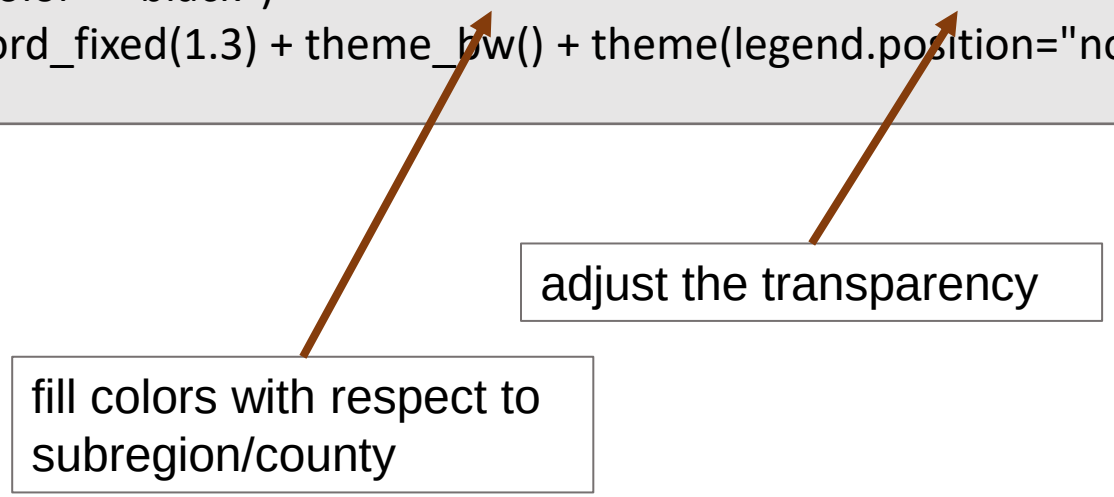
- **Isolating a state for plotting:**



Plotting Spatial Data

- **Changing the colors of counties:**

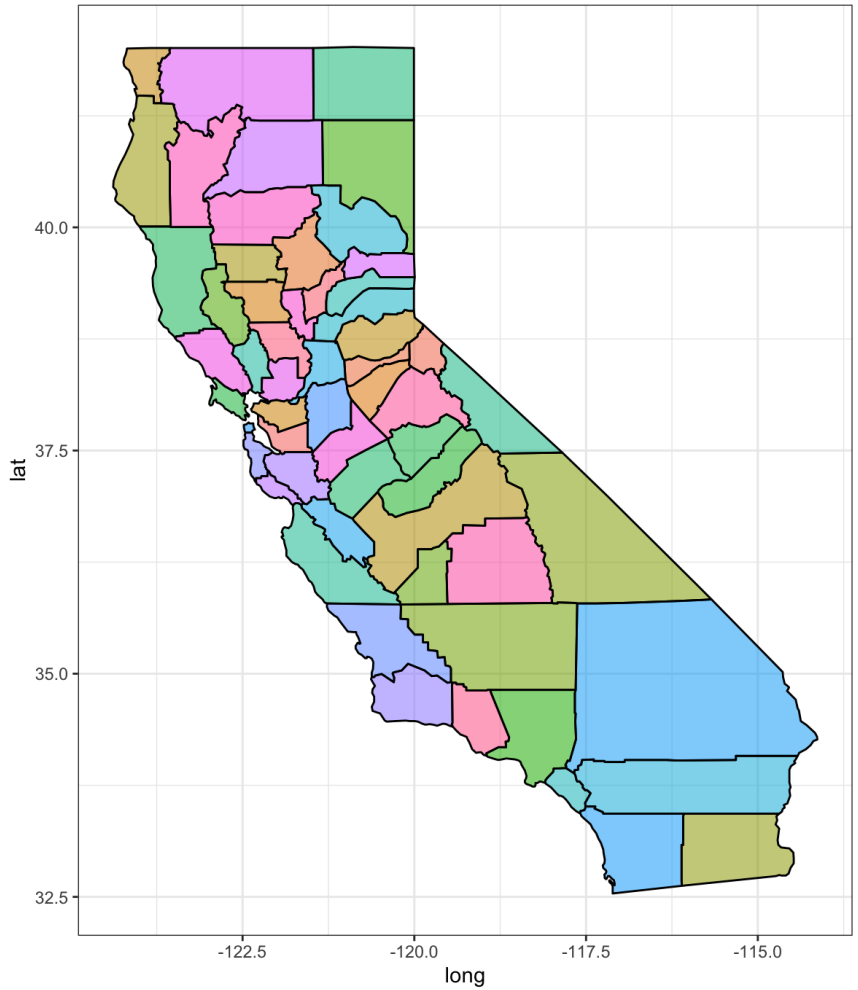
```
ggplot(ca_df, aes(x = long, y = lat, group = group)) + geom_polygon(data =       ca_county, aes(fill = subregion, alpha=0.6), color = "black") +
          coord_fixed(1.3) + theme_bw() + theme(legend.position="none")
```



TEXAS A&M UNIVERSITY
SUPERFUND
RESEARCH CENTER

Plotting Spatial Data

- **Changing the colors of counties:**



Plotting Spatial Data

- **Add text to map:**

find the centroid of the regions to place the text

```
centroid <- aggregate(cbind(long,lat) ~ subregion, data=ca_county, FUN=mean)
head(centroid)
```

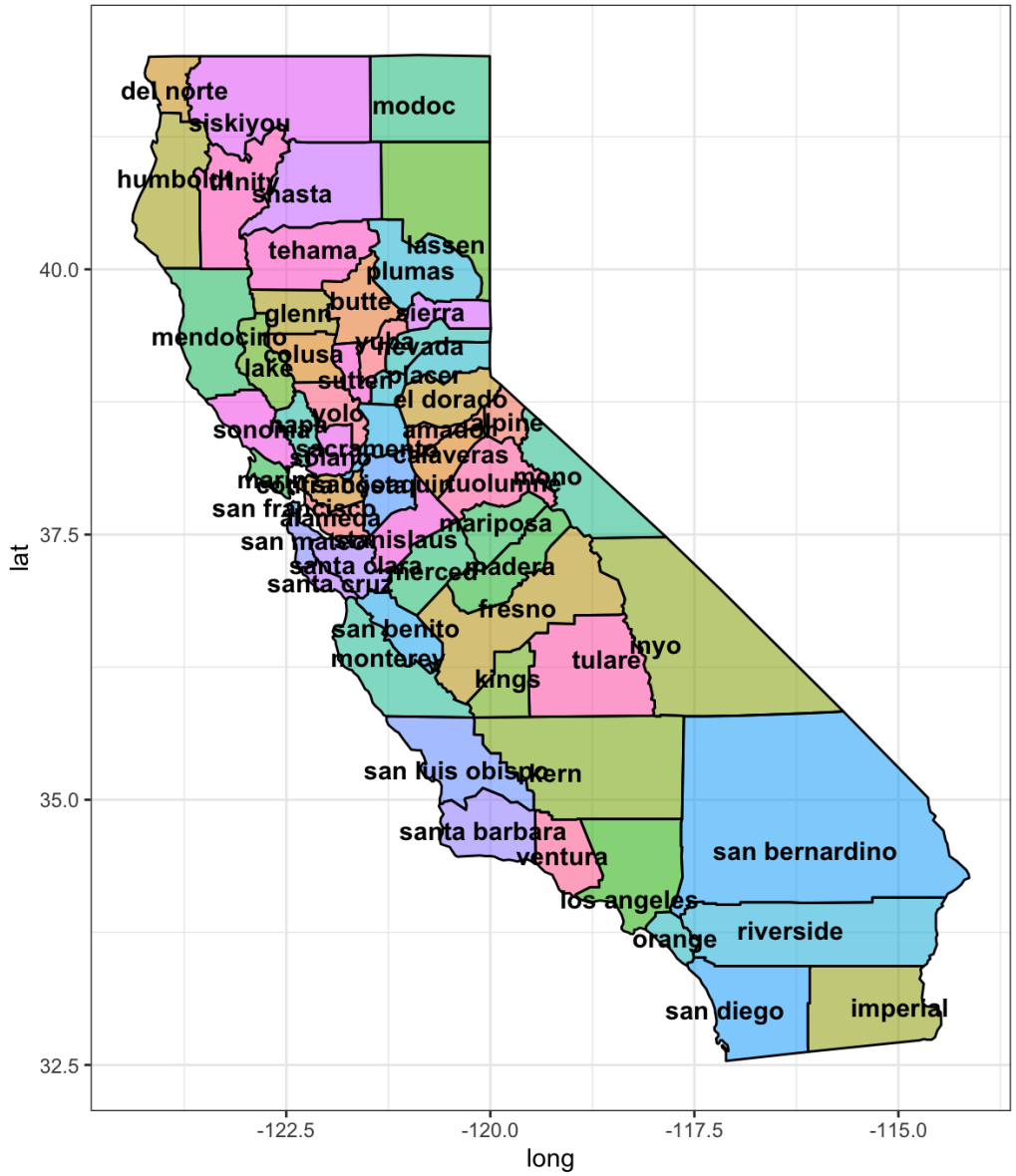
```
ggplot(ca_df, aes(x = long, y = lat, group = group)) +
        geom_polygon(data = ca_county, aes(fill = subregion, alpha=0.6),        color = "black") +
        geom_text(data = centroid, aes(x=long, y=lat, label=subregion),  inherit.aes = FALSE, fontface = "bold
+
        coord_fixed(1.3) + theme_bw() + theme(legend.position="none")
```

place the text at the centroids and add the subregion labels

TEXAS A&M UNIVERSITY
SUPERFUND
RESEARCH CENTER

Plotting Spatial Data

The figure looks busy. Let's highlight only a few counties to improve the visualization



Plotting Spatial Data

- **Select a subset of counties for labeling:**

find the centroid of the regions to place the text

```
select_counties <- c("los angeles", "san diego","monterey","santa barbara")
select_ca <-   centroid[centroid$subregion %in% select_counties,]
```

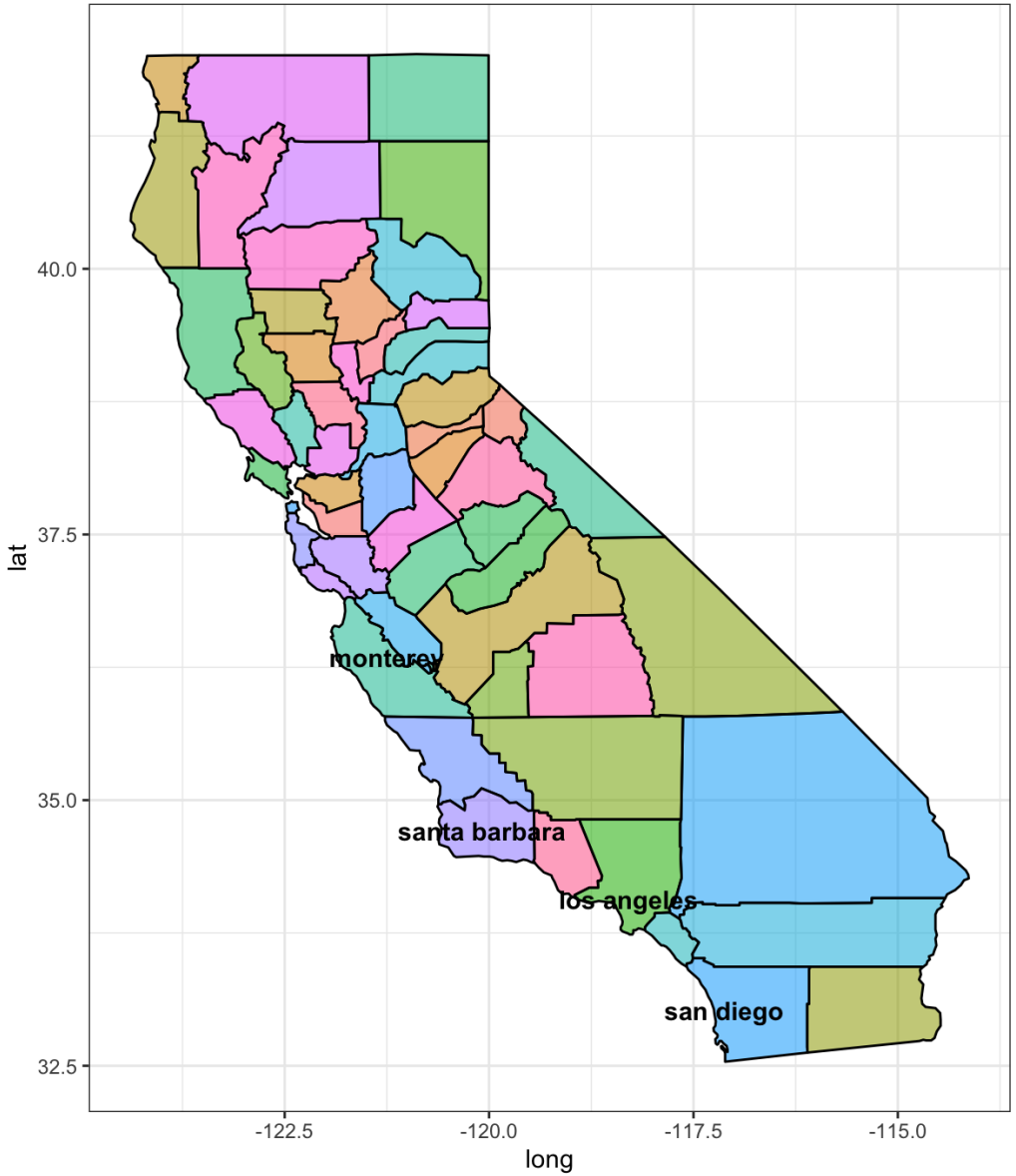
```
ggplot(ca_df, aes(x = long, y = lat, group = group)) + geom_polygon(data =       ca_county, aes(fill = subregion,
alpha=0.6), color = "black") +
          geom_text(data = select_ca, aes(x=long, y=lat,     label=subregion),inherit.aes = FALSE, fontface = "bold") +
          coord_fixed(1.3) + theme_bw() + theme(legend.position="none")
```

only add the selected text to the plot

TEXAS A&M UNIVERSITY
SUPERFUND
RESEARCH CENTER

Plotting Spatial Data

Let's push the labels outward for better visualization



Plotting Spatial Data

- **Select a subset of counties for labeling:**

```
library(ggrepel)
```

load the ggrepel library

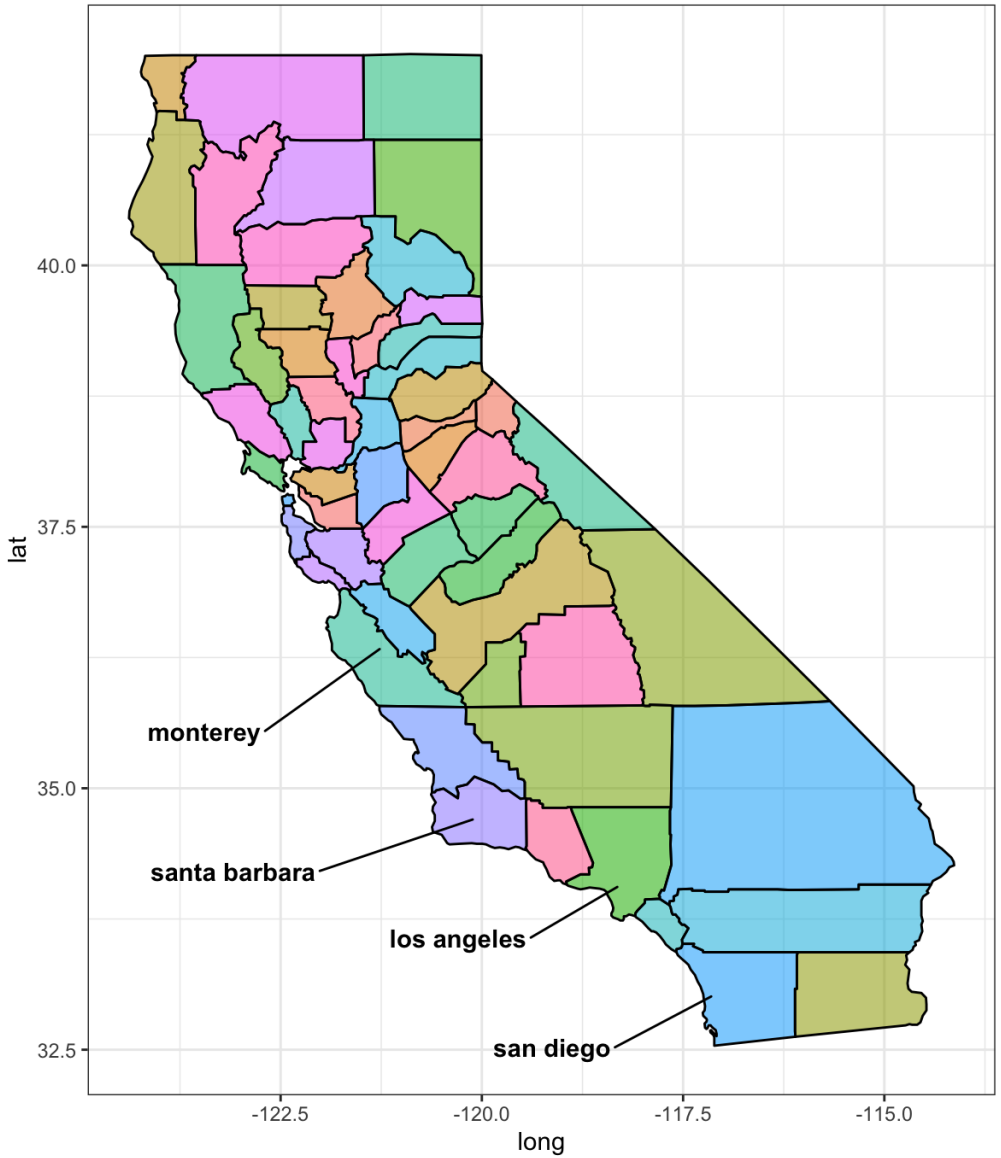
```
ggplot(ca_df, aes(x = long, y = lat, group = group)) + geom_polygon(data =
ca_county, aes(fill = subregion, alpha=0.6), color = "black") +
  geom_text_repel(data = select_ca, aes(x = long, y = lat, label =
subregion), inherit.aes = FALSE, fontface = "bold", nudge_x = c(-2, -
2.2, -2, -3), nudge_y = c(-0.5,-0.8, -0.5, -0.5)) +
  coord_fixed(1.3) + theme_bw() + theme(legend.position="none")
```

call the geom_text_repel function

call the nudge x and y commands to shift the text top, bottom, left and right

TEXAS A&M UNIVERSITY
SUPERFUND
RESEARCH CENTER

Plotting Spatial Data



TEXAS A&M UNIVERSITY SUPERFUND RESEARCH CENTER

Overlaying Environmental Data on Maps

- **Load the Canada region data:**

```
library(autoimage)
data(canada)
library(broom)
canada_df <- tidy(canada)
```

add sampling locations from the environmental data as points to the map. The shape of the point is selected to be diamonds.

- **Load the environmental data:**

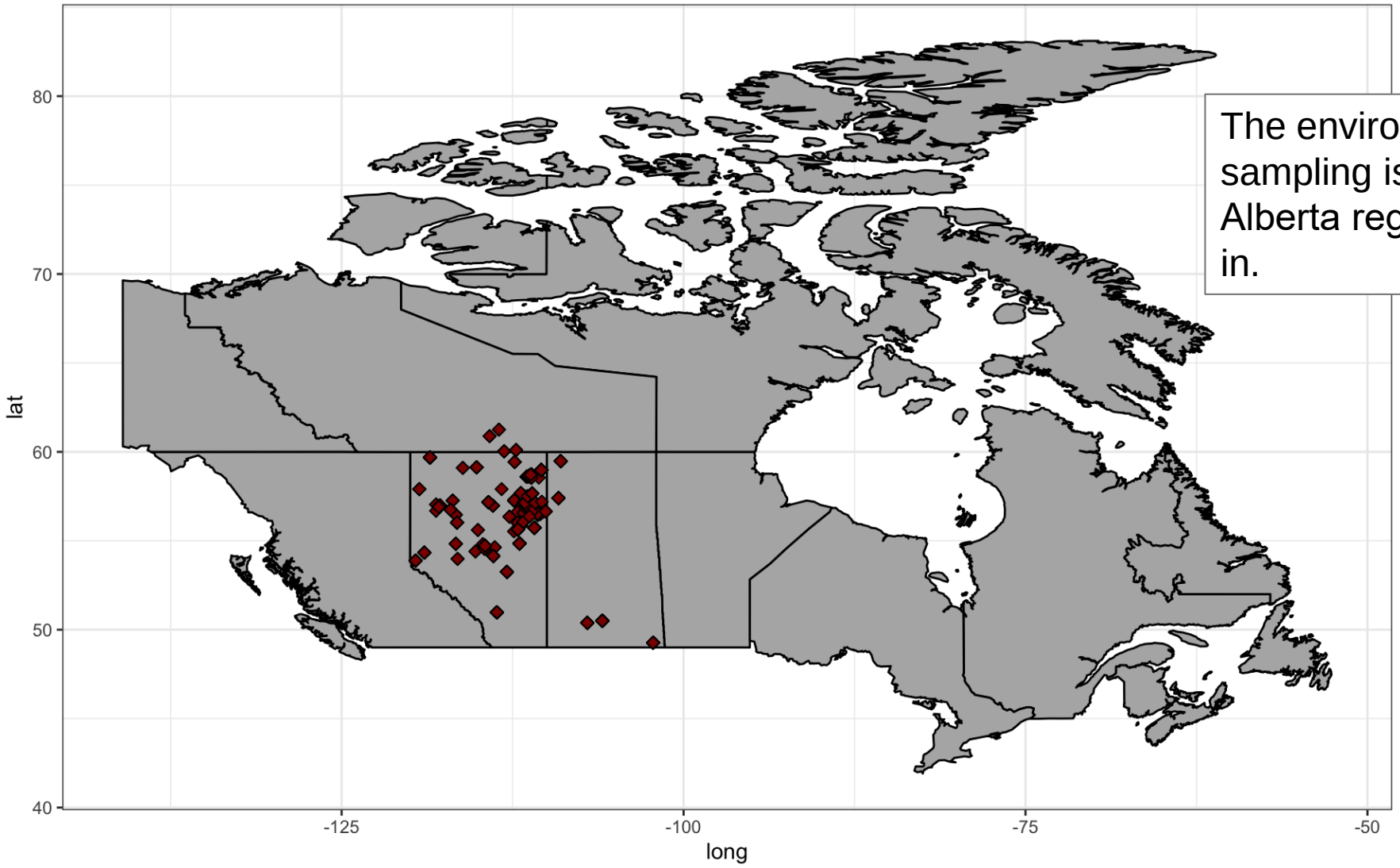
```
env_data <- read.csv("~/Desktop/workshop/complex_exposure_raw_data.csv")
```

- **Plot:**

```
ggplot(canada_df, aes(x=long, y = lat, group = group)) +      geom_polygon(fill="gray70", color="black") +
            geom_point(data = env_data, aes(x = Long, y = Lat), inherit.aes =      FALSE, size = 2, shape = 23,
fill = "darkred") +
            coord_fixed(1.3) + theme_bw()
```

TEXAS A&M UNIVERSITY
SUPERFUND
RESEARCH CENTER

Overlaying Environmental Data on Maps



The environmental sampling is mainly in the Alberta region. Let's zoom in.

Overlaying Environmental Data on Maps

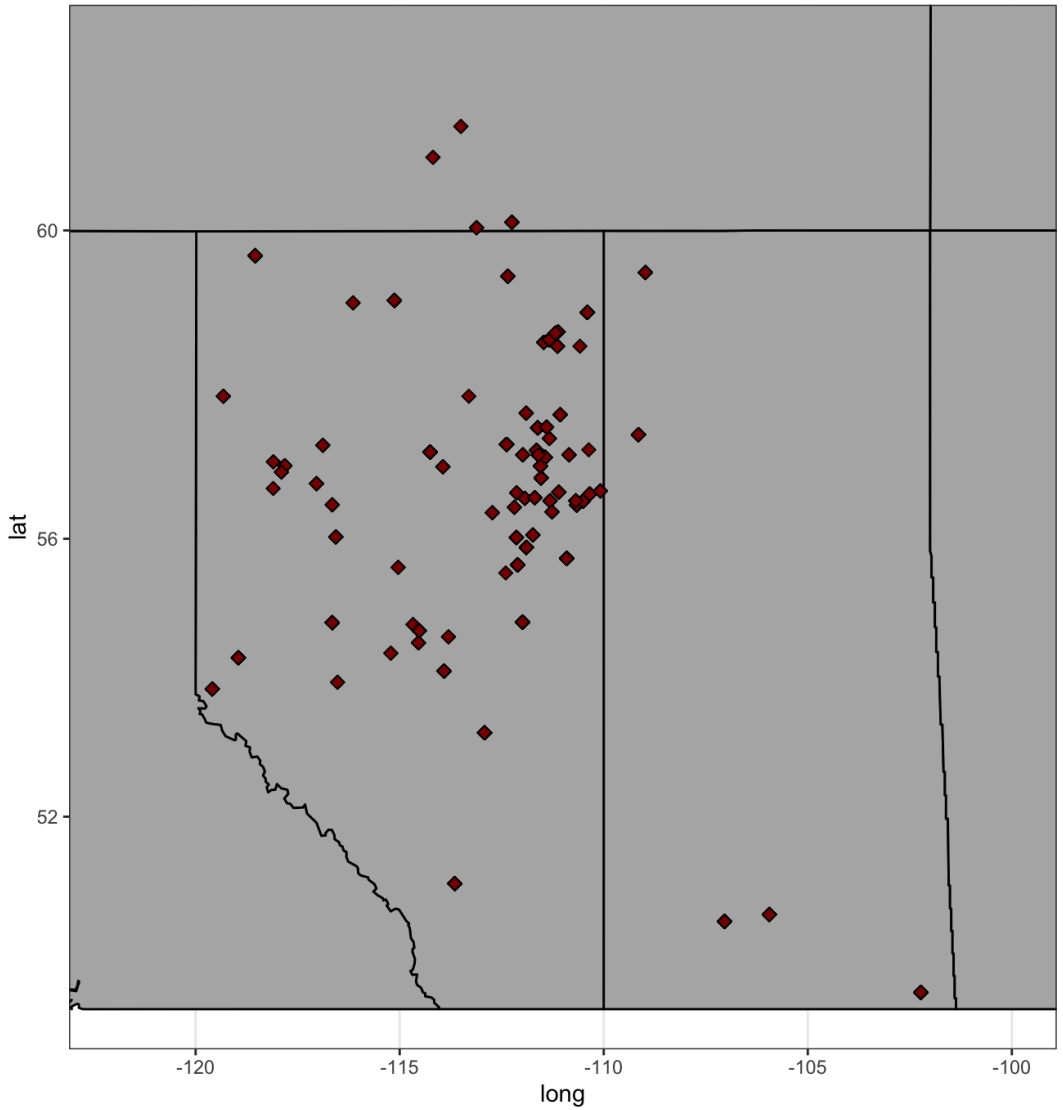
- **Redefining axis limits:**

```
ggplot(canada_df, aes(x=long, y = lat, group = group)) +      geom_polygon(fill="gray70",color="black") +
        geom_point(data = env_data, aes(x = Long, y = Lat), size = 2,          shape = 23, fill = "darkred") +
        theme_bw() +
        coord_map(xlim=c(-122, -100), ylim=c(49, 62))
```

define x and y limits to zoom in

TEXAS A&M UNIVERSITY
SUPERFUND
RESEARCH CENTER

Overlaying Environmental Data on Maps



Let’s remove the map fill color and plot lead levels for each sample.

Overlaying Environmental Data on Maps

- **Redefining axis limits:**

remove the map fill color

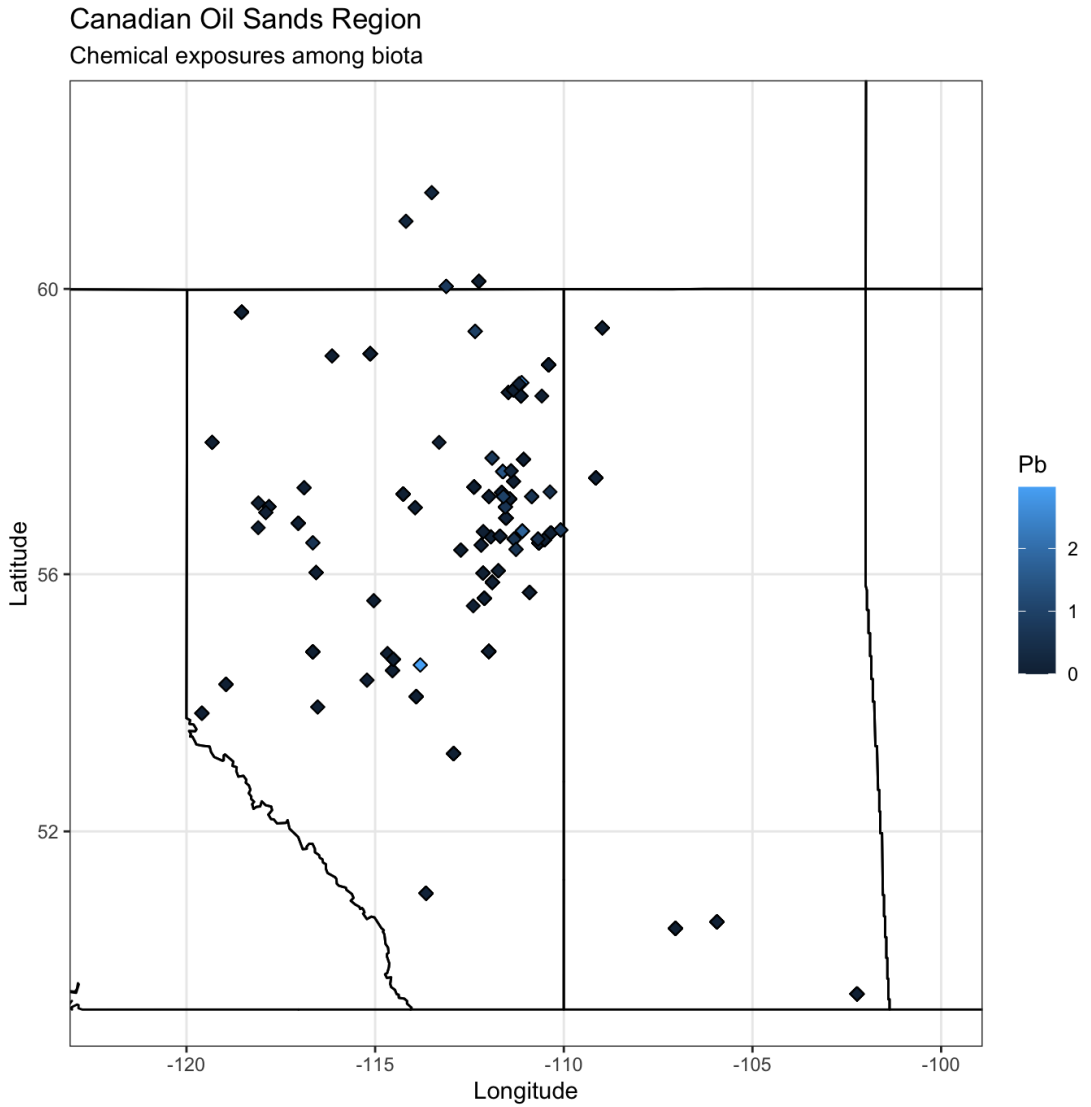
```
ggplot(canada_df, aes(x=long, y = lat, group = group)) + geom_polygon(fill=NA,          color="black") +
geom_point(data = env_data, aes(x = Long, y = Lat,
            fill = Pb), inherit.aes = FALSE, size = 2, shape = 23) +
            theme_bw() +
            coord_map(xlim=c(-122, -100), ylim=c(49, 62)) +
            labs(title="Canadian Oil Sands Region", subtitle = "Chemical exposures          among biota")
+xlab("Longitude") + ylab("Latitude")
```

add titles

adjust the fill color of the samples based on the lead levels in the dataset

TEXAS A&M UNIVERSITY
SUPERFUND
RESEARCH CENTER

Overlaying Environmental Data on Maps



The legend shows the lead levels in the samples

TEXAS A&M UNIVERSITY SUPERFUND RESEARCH CENTER

Overlaying Environmental Data on Maps

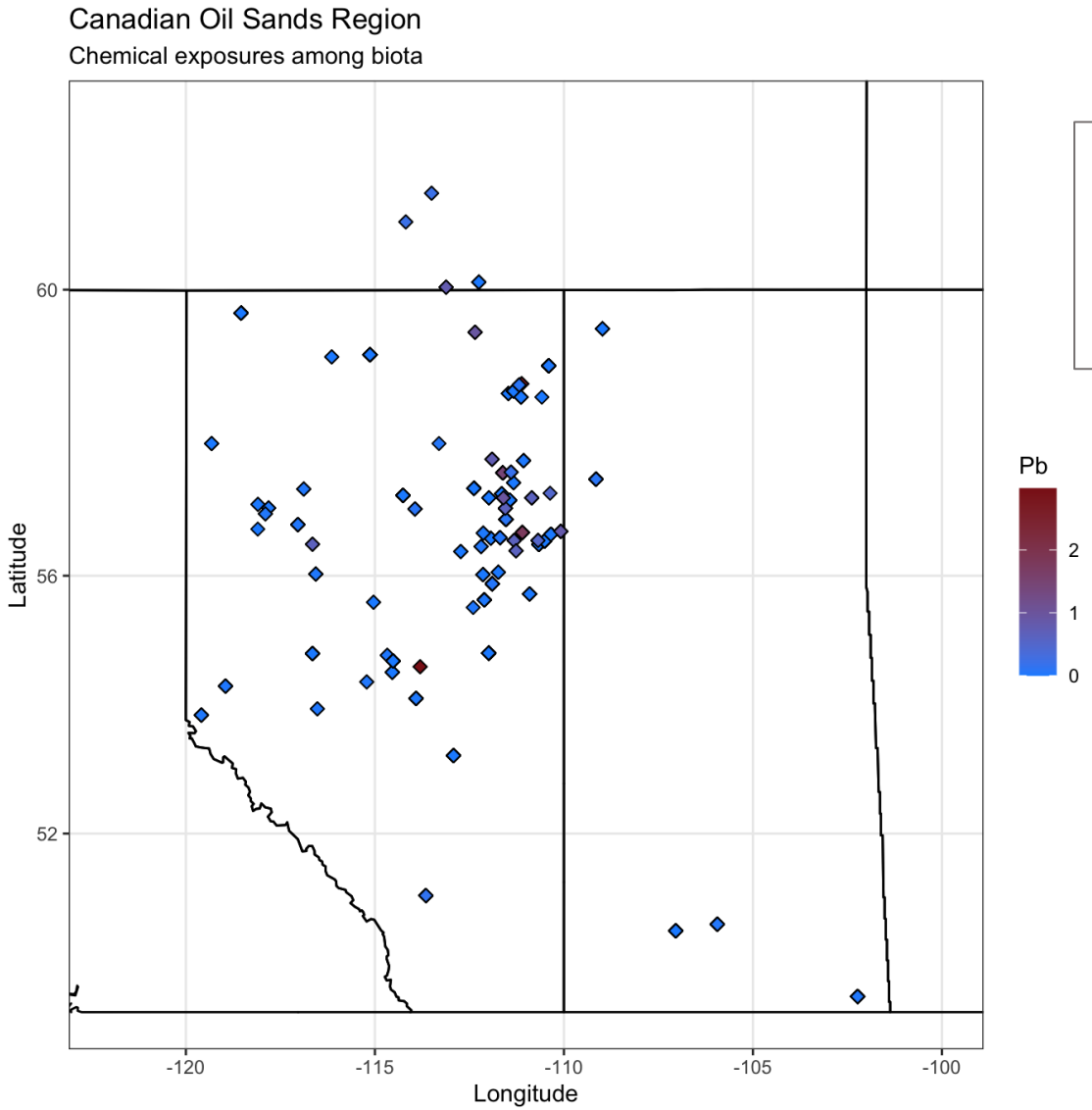
- **Change the gradient colors for improved communication:**

```
ggplot(canada_df, aes(x=long, y = lat, group = group)) + geom_polygon(fill=NA, color="black") +
geom_point(data = env_data, aes(x = Long, y = Lat,
           fill = Pb), inherit.aes = FALSE, size = 2, shape = 23) +
           theme_bw() + coord_map(xlim=c(-122, -100), ylim=c(49, 62)) +
           scale_fill_gradient(low = "dodgerblue", high = "firebrick4") +
           labs(title="Canadian Oil Sands Region", subtitle = "Chemical exposures among biota") +
xlab("Longitude") + ylab("Latitude")
```

adjust the gradient color with low levels shown in blue (cold) and high lead levels with red (hot)

TEXAS A&M UNIVERSITY
SUPERFUND
RESEARCH CENTER

Overlaying Environmental Data on Maps



You can explore different metals and chemicals in the dataset using the same set of codes

Overlaying Environmental Data on Maps

- **Show species on the map:**

```
env_data$Species <- factor(env_data$Species)
```

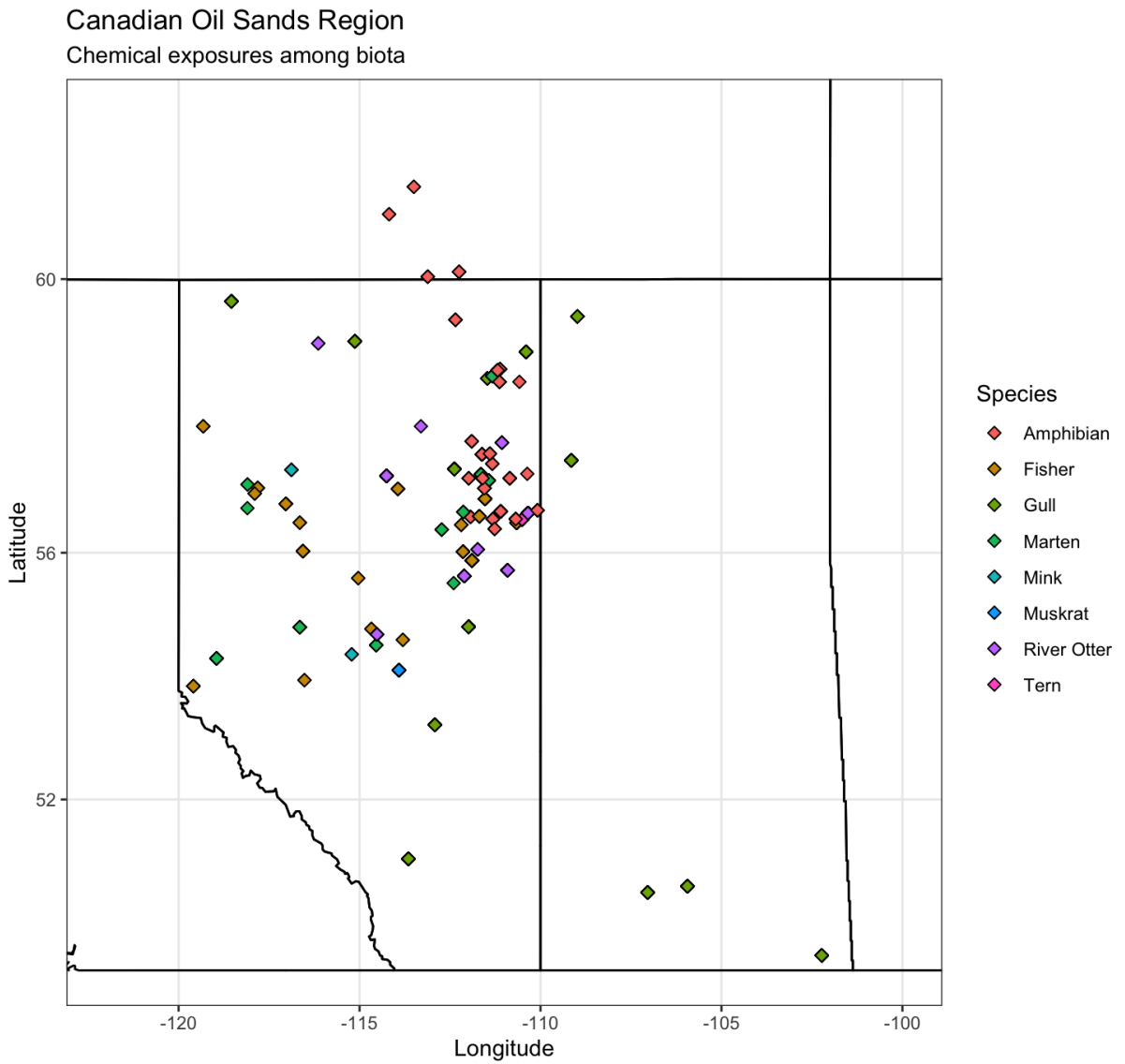
make species factor

```
ggplot(canada_df, aes(x=long, y = lat, group = group)) +      geom_polygon(fill=NA, color="black") +
geom_point(data = env_data,              aes(x = Long, y = Lat, fill = Species), inherit.aes = FALSE, size
            = 2, shape = 23) + theme_bw() +
            coord_map(xlim=c(-122, -100), ylim=c(49, 62)) +
            labs(title="Canadian Oil Sands Region", subtitle = "Chemical            exposures among biota") +
xlab("Longitude") + ylab("Latitude")
```

color samples with respect to the studies species

TEXAS A&M UNIVERSITY
SUPERFUND
RESEARCH CENTER

Overlaying Environmental Data on Maps



Last Step: Saving Plots

- **Saving as png, pdf, and other formats:**

```
png("Species.png", width = 8, height = 8, units="in",res = 500)

ggplot(canada_df, aes(x=long, y = lat, group = group)) +      geom_polygon(fill=NA, color="black") +
            geom_point(data = env_data, aes(x = Long, y = Lat, fill =      Species), inherit.aes = FALSE, size = 2,
shape = 23) + theme_bw() +
            coord_map(xlim=c(-122, -100), ylim=c(49, 62)) +
            labs(title="Canadian Oil Sands Region", subtitle = "Chemical            exposures among biota") +
xlab("Longitude") + ylab("Latitude")

dev.off()
```

size parameters

resolution

file type for saving

close the plotting device

TEXAS A&M UNIVERSITY SUPERFUND RESEARCH CENTER

Last Step: Saving Plots

- **Using ggsave command to save plots:**

```
ggplot(df, aes(x=ENSMUSG00000000049, group=CLASS.DOSE, fill=CLASS.DOSE)) +
  geom_density() +
  theme_bw() +
  facet_wrap(~CLASS.DOSE) +
  theme(text = element_text(size=14),
        legend.position="none",
        panel.spacing = unit(0.1, "lines")) +
  scale_fill_brewer(palette="Reds")

ggsave("DensityFacet.pdf", width = 6.5, height = 6, dpi = 600)
```

Notice that ggsave is executed after the ggplot command

file name and type

size parameters

resolution

Last Step: Saving Plots

- **Where does R save the plots? Check and change the working directory**

```
getwd()

setwd("~/Desktop/workshop/")
```

get working directory

set working directory to "workshop"

If you don't change the working directory, R will save your plots to the main directory.

TEXAS A&M UNIVERSITY SUPERFUND RESEARCH CENTER

Interactive Plots - Plotly

```
library(plotly)
```

→ load plotly library

- **Interactive density plot:**

```
myplot <- ggplot(df, aes(x=ENSMUSG00000000049, group=CLASS.DOSE,
          fill=CLASS.DOSE)) +
                geom_density() +
                theme_bw() +
                facet_wrap(~CLASS.DOSE) +
                theme(text = element_text(size=14),
                legend.position="none",
                panel.spacing = unit(0.1, "lines")) +
                scale_fill_brewer(palette="Reds")
```

save the ggplot command to a variable

```
ggplotly(myplot)
```

← call the variable in ggplotly to view the interactive plot

Interactive Plots - Plotly

- **Interactive scatter plot:**

```
my_scatter <- ggplot(df, aes(ENSMUSG00000000001, ENSMUSG00000000028,
                             colour = CLASS.DOSE)) + geom_point(size=3) +
        labs(col='Dose') +
                theme_bw() +
        theme(text = element_text(size=14))
```

```
ggplotly(my_scatter)
```

call the variable in ggplotly to view the interactive plot

TEXAS A&M UNIVERSITY SUPERFUND RESEARCH CENTER

Interactive Plots - Plotly

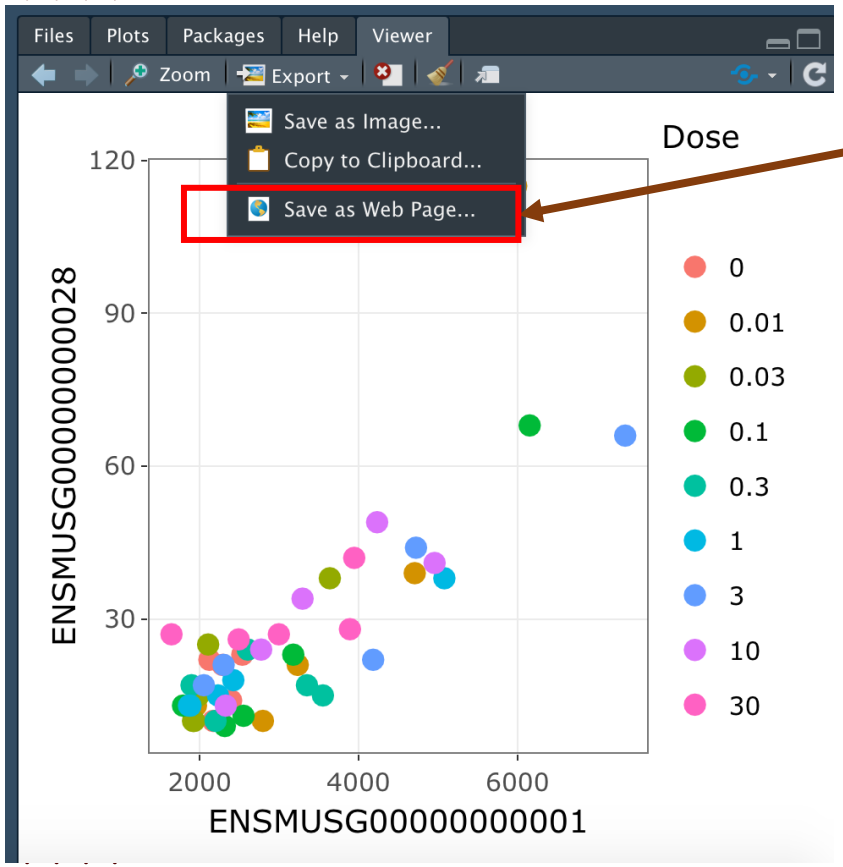
- **Interactive boxplot plot:**

```
myboxplot <- ggplot(df,aes(x=CLASS.DOSE, y=ENSMUSG00000000001,
                           fill=CLASS.DOSE)) +  geom_boxplot(alpha=0.6) +
                                  geom_jitter(color="black") + coord_flip() +
                                  labs(x = "Dose") + theme_bw() +
                                  theme(legend.position="none",
                    text = element_text(size=20),
                    axis.text.x = element_text(angle=45, hjust = 1)) +
                                  scale_fill_brewer(palette="PuBu")
```

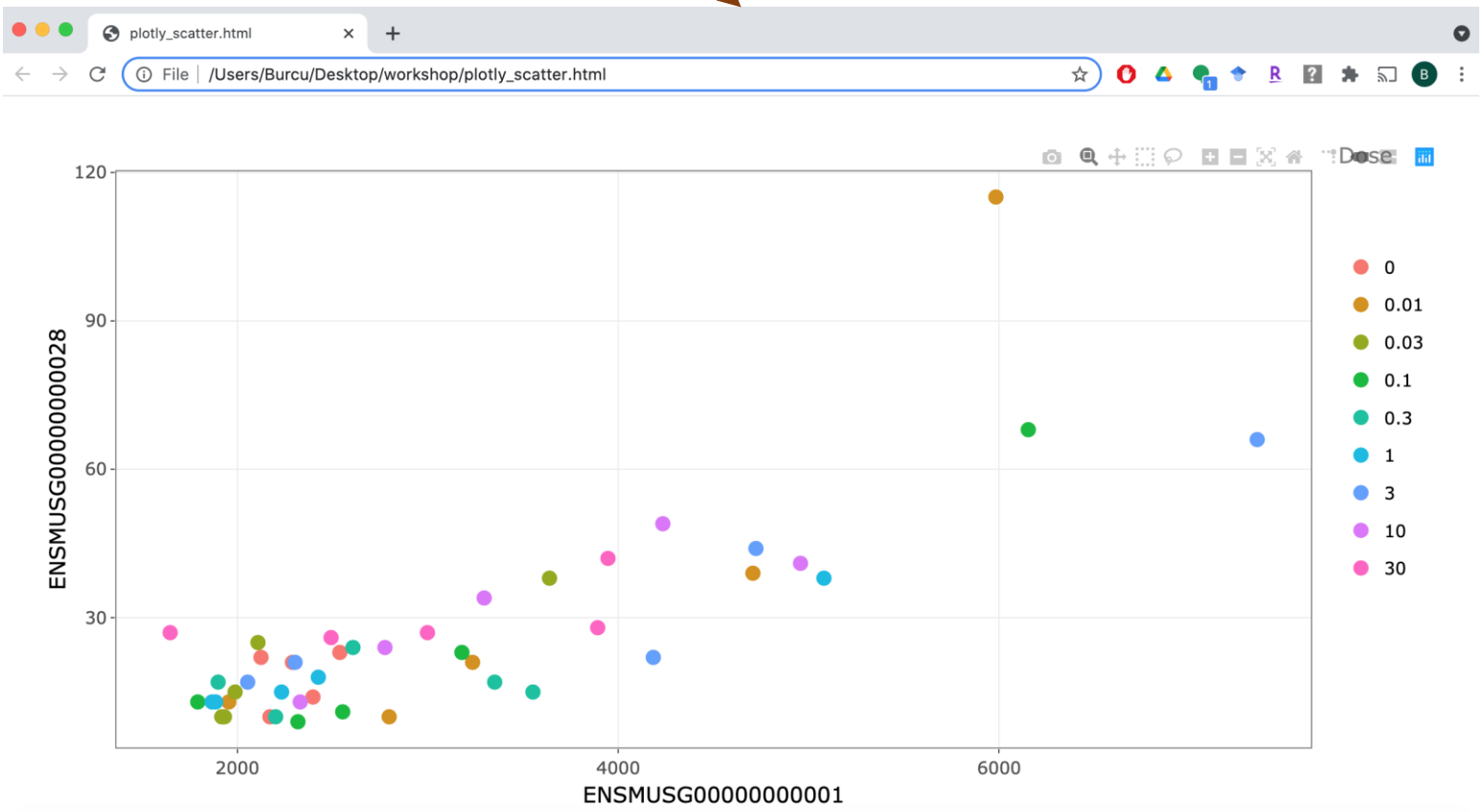
```
ggplotly(myboxplot)
```

call the variable in ggplotly to view the interactive plot

Saving an Interactive Plot as an HTML File



click save as web page to save this interactive plot as an html file

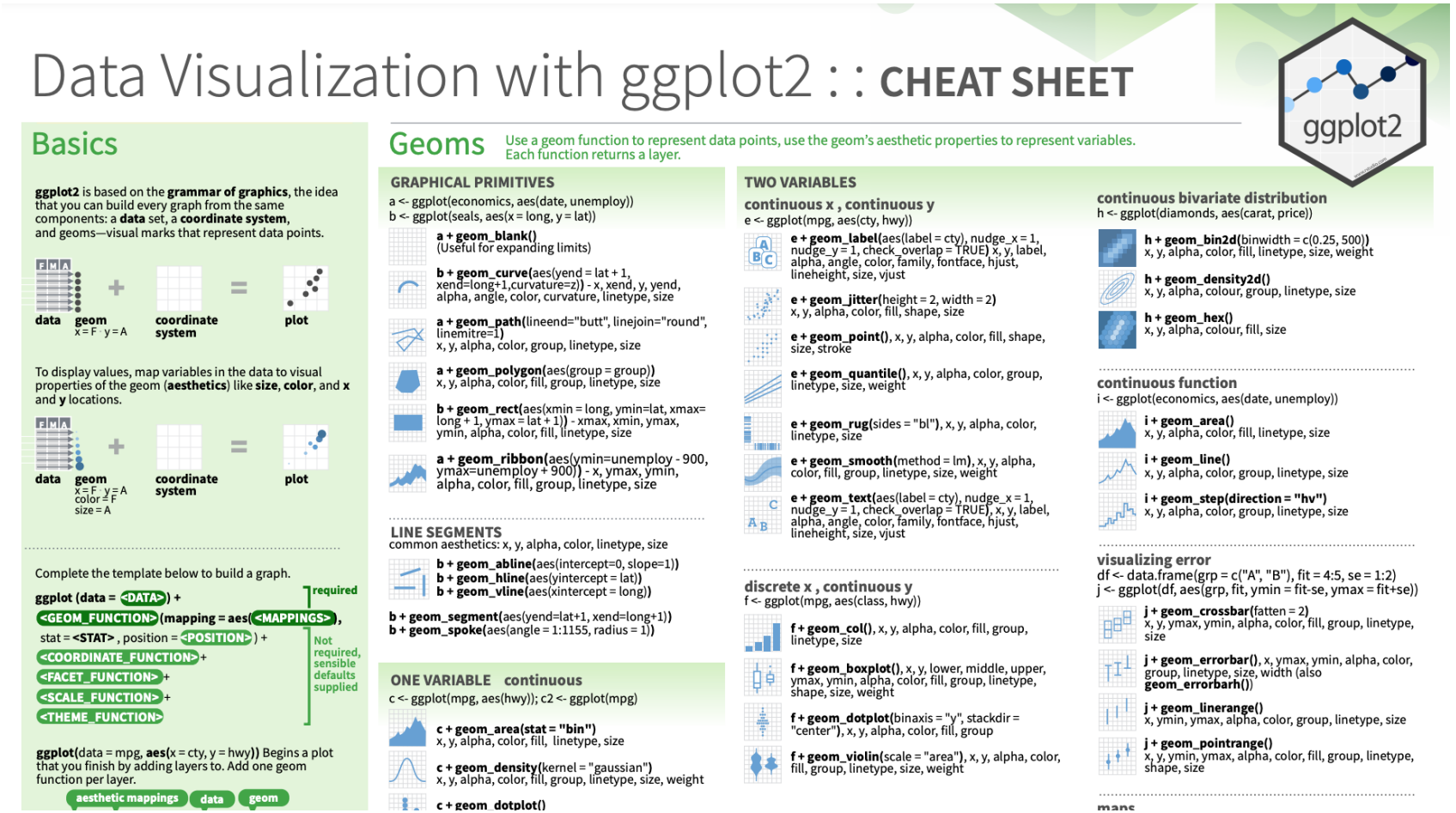


TEXAS A&M UNIVERSITY
SUPERFUND
RESEARCH CENTER

Cheatsheet for ggplot2

- **Link to the cheatsheet**

  https://www.maths.usyd.edu.au/u/UG/SM/STAT3022/r/current/Misc/data-visualization-2.1.pdf



Data Visualization with ggplot2 : : CHEAT SHEET

ggplot2

Basics

ggplot2 is based on the grammar of graphics, the idea that you can build every graph from the same components: a data set, a coordinate system, and geoms—visual marks that represent data points.

data + geom (x = F, y = A) = coordinate system = plot

To display values, map variables in the data to visual properties of the geom (aesthetics) like size, color, and x and y locations.

data + geom (x = F, y = A, color = F, size = A) = coordinate system = plot

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +                                  required
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
 stat = <STAT> , position = <POSITION> ) +                Not required, sensible defaults supplied
<COORDINATE_FUNCTION>+
<FACET_FUNCTION>+
<SCALE_FUNCTION>+
<THEME_FUNCTION>
```

ggplot(data = mpg, aes(x = cty, y = hwy)) Begins a plot that you finish by adding layers to. Add one geom function per layer.

aesthetic mappings | data | geom

Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))

- **a + geom_blank()** (Useful for expanding limits)
- **b + geom_curve(**aes(yend = lat + 1, xend=long+1,curvature=z)) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size
- **a + geom_path(**lineend="butt", linejoin="round", linemitre=1) x, y, alpha, color, group, linetype, size
- **a + geom_polygon(**aes(group = group)) x, y, alpha, color, fill, group, linetype, size
- **b + geom_rect(**aes(xmin = long, ymin=lat, xmax= long + 1, ymax = lat + 1)) - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size
- **a + geom_ribbon(**aes(ymin=unemploy - 900, ymax=unemploy + 900)) - x, ymax, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

- **b + geom_abline(**aes(intercept=0, slope=1))
- **b + geom_hline(**aes(yintercept = lat))
- **b + geom_vline(**aes(xintercept = long))
- **b + geom_segment(**aes(yend=lat+1, xend=long+1))
- **b + geom_spoke(**aes(angle = 1:1155, radius = 1))

ONE VARIABLE continuous

c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)

- **c + geom_area(stat = "bin")** x, y, alpha, color, fill, linetype, size
- **c + geom_density(**kernel = "gaussian") x, y, alpha, color, fill, group, linetype, size, weight
- **c + geom_dotplot()**

TWO VARIABLES

continuous x , continuous y

e <- ggplot(mpg, aes(cty, hwy))

- **e + geom_label(**aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE) x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust
- **e + geom_jitter(**height = 2, width = 2) x, y, alpha, color, fill, shape, size
- **e + geom_point()**, x, y, alpha, color, fill, shape, size, stroke
- **e + geom_quantile()**, x, y, alpha, color, group, linetype, size, weight
- **e + geom_rug(**sides = "bl"), x, y, alpha, color, linetype, size
- **e + geom_smooth(**method = lm), x, y, alpha, color, fill, group, linetype, size, weight
- **e + geom_text(**aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE), x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

discrete x , continuous y

f <- ggplot(mpg, aes(class, hwy))

- **f + geom_col()**, x, y, alpha, color, fill, group, linetype, size
- **f + geom_boxplot()**, x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight
- **f + geom_dotplot(**binaxis = "y", stackdir = "center"), x, y, alpha, color, fill, group
- **f + geom_violin(**scale = "area"), x, y, alpha, color, fill, group, linetype, size, weight

continuous bivariate distribution

h <- ggplot(diamonds, aes(carat, price))

- **h + geom_bin2d(**binwidth = c(0.25, 500)) x, y, alpha, color, fill, linetype, size, weight
- **h + geom_density2d()** x, y, alpha, colour, group, linetype, size
- **h + geom_hex()** x, y, alpha, colour, fill, size

continuous function

i <- ggplot(economics, aes(date, unemploy))

- **i + geom_area()** x, y, alpha, color, fill, linetype, size
- **i + geom_line()** x, y, alpha, color, group, linetype, size
- **i + geom_step(direction = "hv")** x, y, alpha, color, group, linetype, size

visualizing error

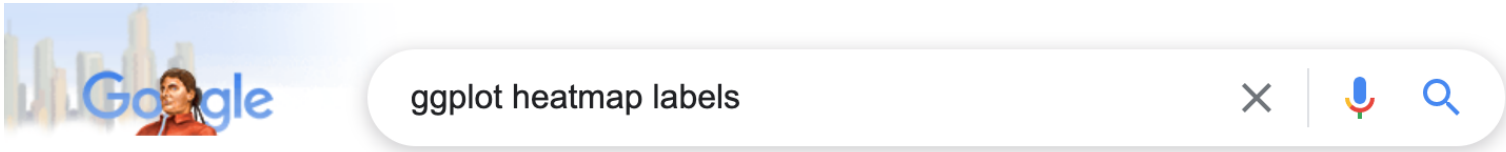
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))

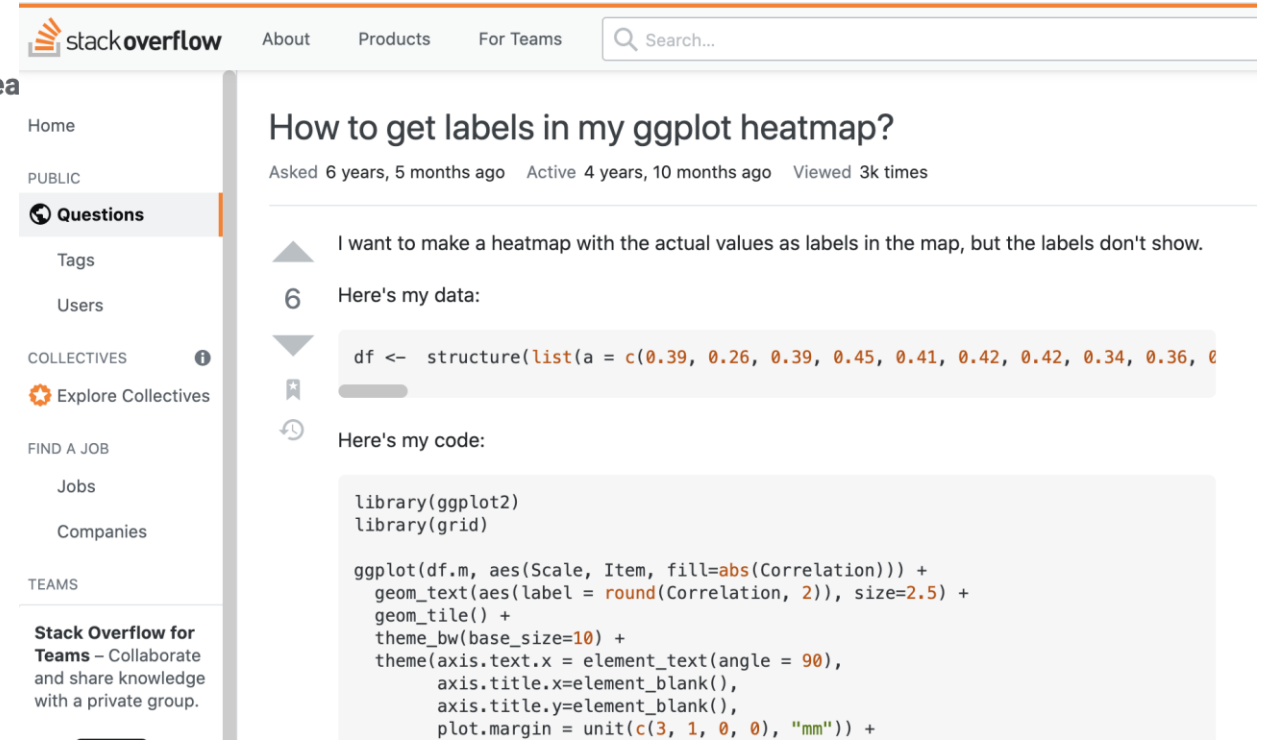
- **j + geom_crossbar(**fatten = 2) x, y, ymax, ymin, alpha, color, fill, group, linetype, size
- **j + geom_errorbar()**, x, ymax, ymin, alpha, color, group, linetype, size, width (also **geom_errorbarh()**)
- **j + geom_linerange()** x, ymin, ymax, alpha, color, group, linetype, size
- **j + geom_pointrange()** x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps

TEXAS A&M UNIVERSITY SUPERFUND RESEARCH CENTER

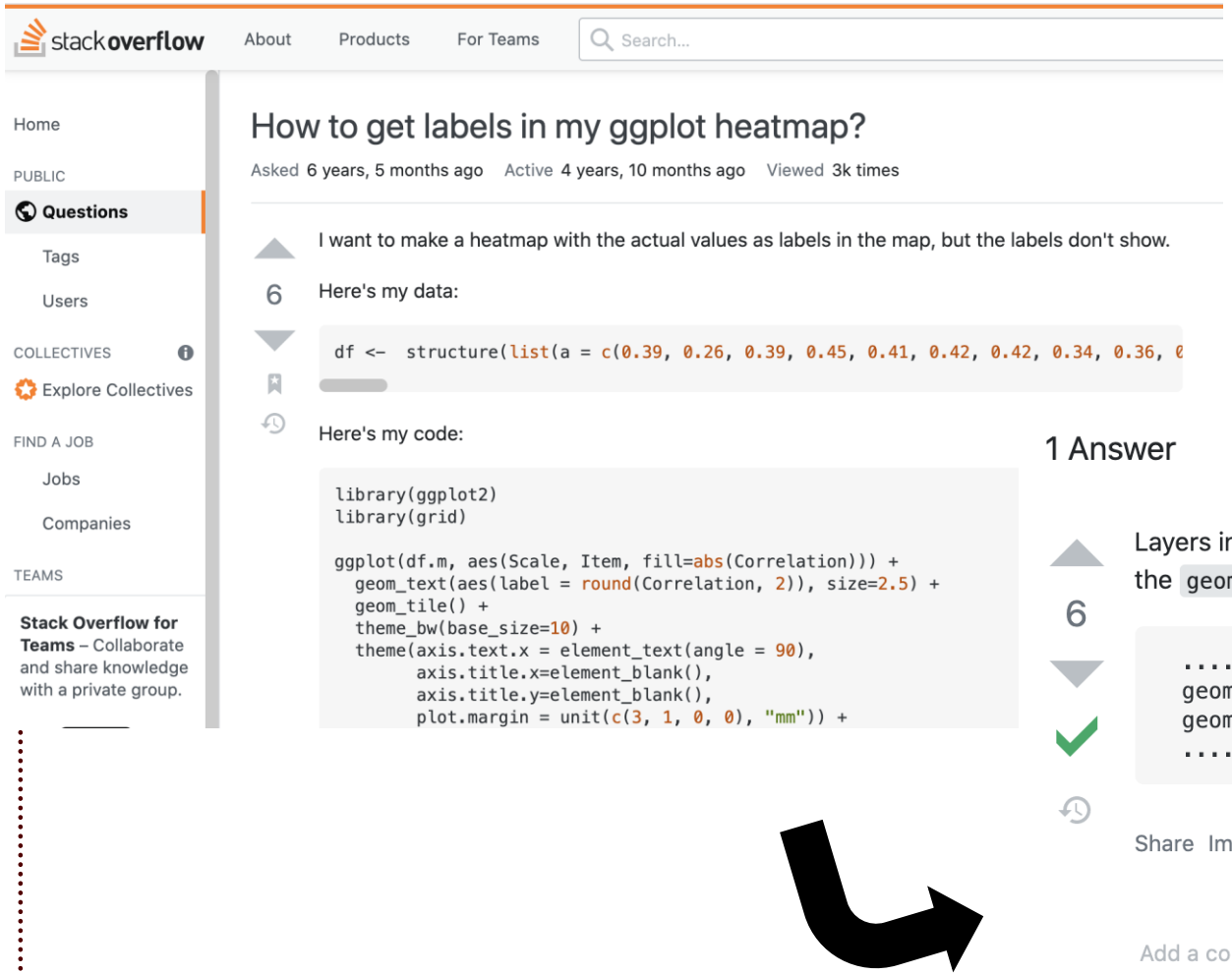
Stackoverflow





TEXAS A&M UNIVERSITY SUPERFUND RESEARCH CENTER

Stackoverflow



Scroll down to see the answer

1 Answer

Active | Oldest | Votes

6

Layers in the `ggplot` are plotted in order as they are written. To display text above the heatmap the `geom_text()` call should be place after `geom_tile()` call.

```
.....
geom_tile() +
geom_text(aes(label = round(Correlation, 2)), size=2.5) +
.....
```

Share Improve this answer Follow

answered Apr 22 '15 at 8:09

Didzis Elferts

86.6k ● 12 ● 244 ● 192

Add a comment

Questions??

TEXAS A&M UNIVERSITY
Superfund Research Center

Thank you for joining us !

The next session is on November 3, 2021
2:00 – 4:00 pm Eastern US Time

"PLACING TOXICOLOGY DATA IN THE CONTEXT OF EXPOSURE"

Caroline Ring—US Environmental Protection Agency