More ggplot2

Grayson White

Math 241

Week 2 | Spring 2026

Announcements

- Office Hours Schedule
- P-Set 1 available now.
  - Will discuss how to access the p-sets today!
  - Due at 9am the following Thursday
- My Undergraduate Forestry Data Science summer research program application deadline is **THIS FRIDAY!**

Week 2 Goals

Mon Lecture

- Basics of `ggplot2`
- Explore several `geoms`.
- And a little data wrangling with `dplyr` as needed!

Wed Lecture

- GitHub workflow overview
- Learn how to ask coding questions well.
- Graphing context!
  - Labels
  - Highlighting
  - Useful text
- Look at more `geom`s.
- Explore further customizations.
  - Color
  - Themes

P-set tips

- Start on your problem sets **early**, and **ask questions** (in Slack, office hours, the internet) when you get stuck!
- Problem sets are meant to be engaged with over multiple days. It is a lot less effective for your learning if you try to sprint through it the day before it is due.
- If/when you get stuck, it can be helpful to take a break! Some things to try:
  - Go onto another problem, solving it may give you ideas as to how to solve the one you are stuck on.
  - Think about the problem as you are brushing your teeth or laying in bed.
  - Time where you are “bored”/giving your brain time to wander can be helpful for problem solving.

Now: GitHub workflow demo

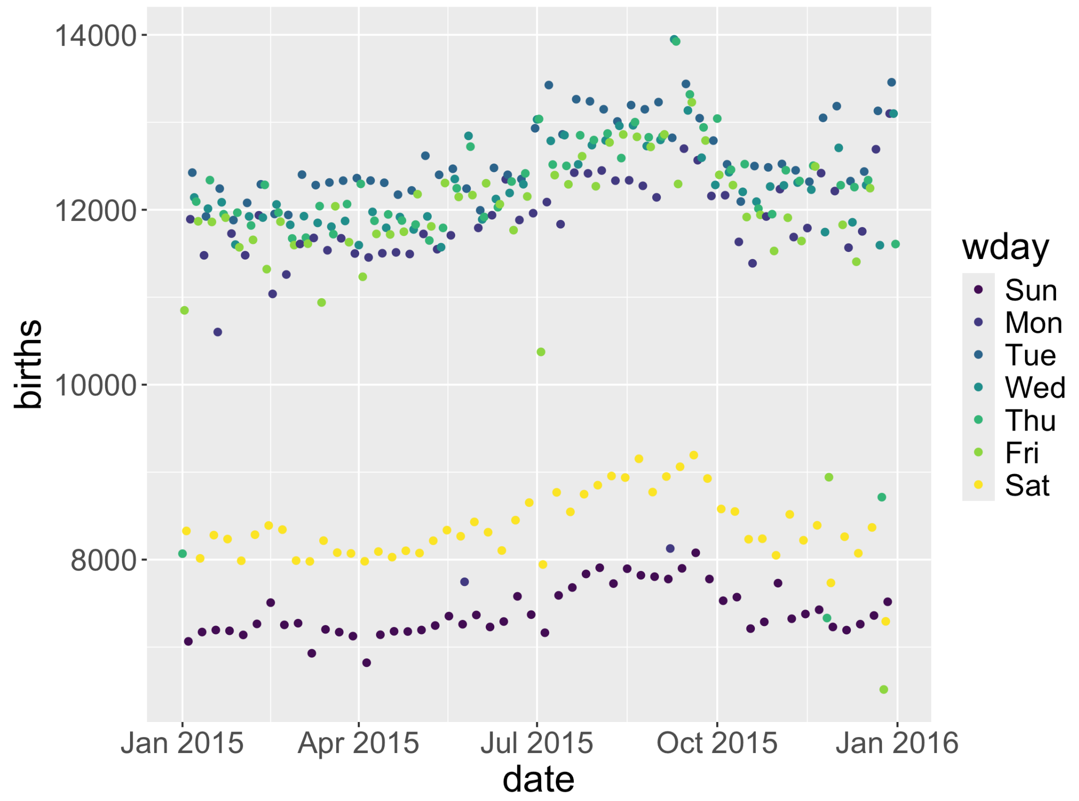
Recap Data: Births2015

```r
# Load libraries
library(mosaicData)
library(tidyverse)


# Grab data
data(Births2015)

# Inspect data
glimpse(Births2015)
```

```
Rows: 365
Columns: 8
$ date         <date> 2015-01-01, 2015-01-02, 2015-01-03, 2015-01-04, 2015-01-…
$ births       <dbl> 8068, 10850, 8328, 7065, 11892, 12425, 12141, 12094, 1186…
$ wday         <ord> Thu, Fri, Sat, Sun, Mon, Tue, Wed, Thu, Fri, Sat, Sun, Mo…
$ year         <dbl> 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 201…
$ month        <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
$ day_of_year  <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17…
$ day_of_month <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17…
$ day_of_week  <dbl> 5, 6, 7, 1, 2, 3, 4, 5, 6, 7, 1, 2, 3, 4, 5, 6, 7, 1, 2, …
```

Recap Data: Births2015

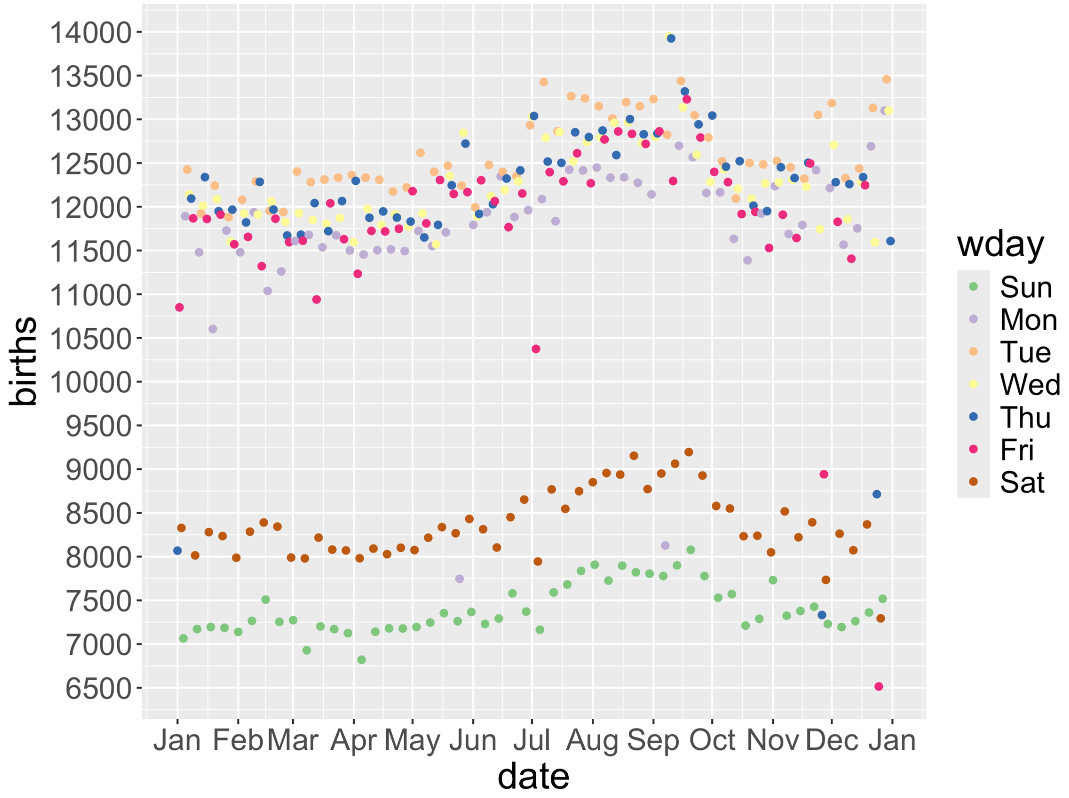
```r
ggplot(data = Births2015,
       mapping = aes(x = date, y = births,
                     color = wday)) +
  geom_point()
```



- Let’s think more about the **scales**.
- Let’s add more **context**!

Scales

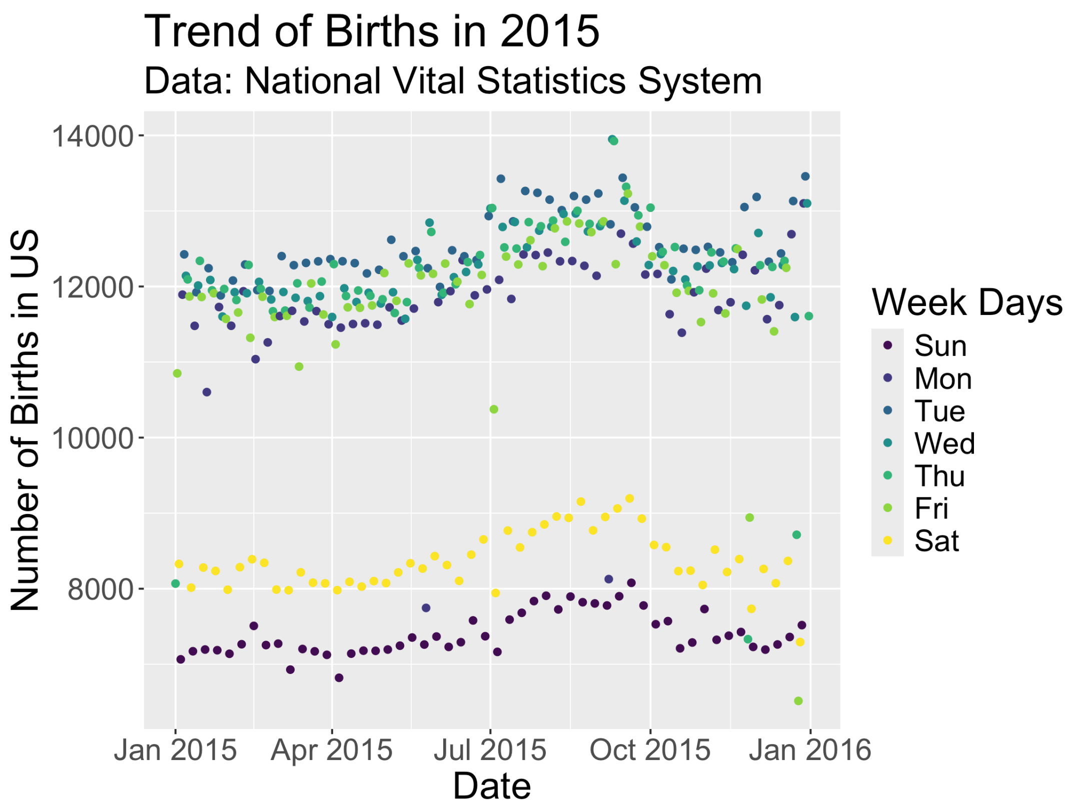
```r
ggplot(data = Births2015,
       mapping = aes(x = date, y = births,
                     color = wday)) +
  geom_point() +
  scale_x_date(date_labels = "%b",
               date_breaks = "1 month") +
  scale_y_continuous(breaks = seq(6000,
                                  14000, by = 500)) +
  scale_color_brewer(type = "qual",
                     palette = 1)
```



- Maybe we want to change the default settings of a scale.
- Maybe we want a different scale than the default.

Context: Labels

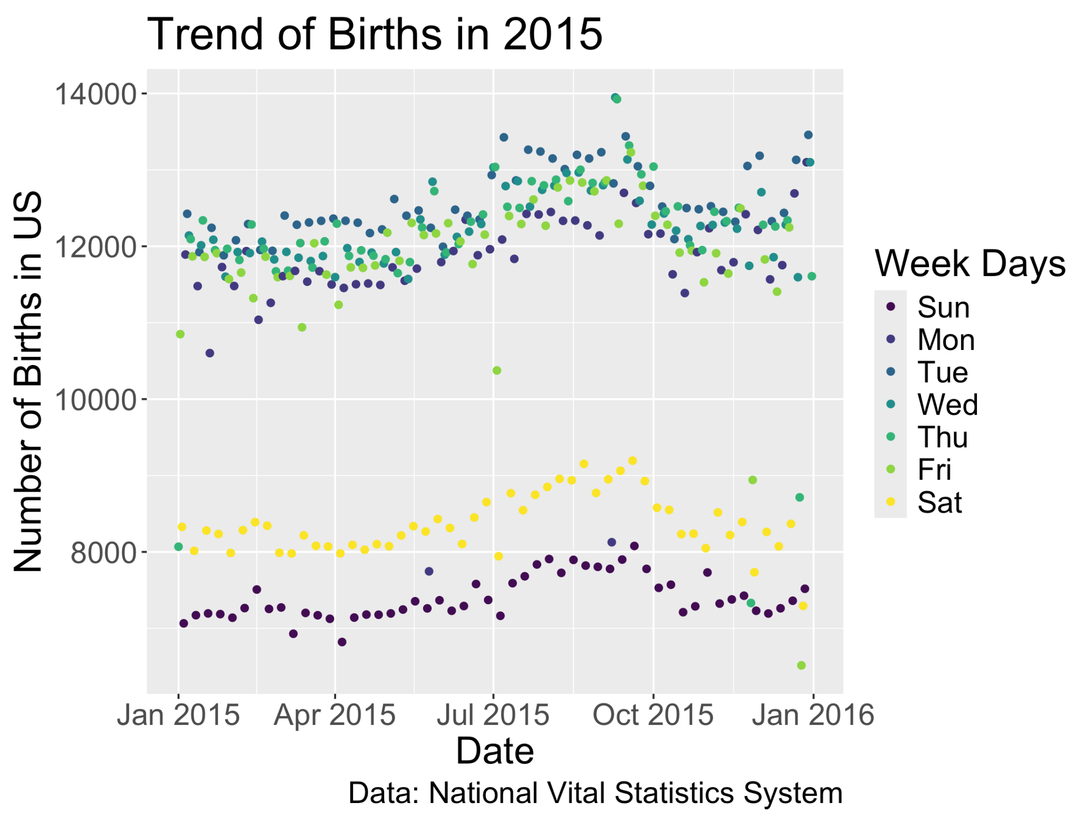
```r
ggplot(data = Births2015,
       mapping = aes(x = date, y = births,
                     color = wday)) +
  geom_point() +
  labs(
    x = "Date",
    y = "Number of Births in US",
    title = "Trend of Births in 2015",
    subtitle = "Data: National Vital Statistics System"
    color = "Week Days"
  )
```



- Prefer citing the data at the bottom?

Context: Labels

```r
ggplot(data = Births2015,
       mapping = aes(x = date, y = births,
                     color = wday)) +
  geom_point() +
  labs(
    x = "Date",
    y = "Number of Births in US",
    title = "Trend of Births in 2015",
    caption = "Data: National Vital Statistics System",
    color = "Week Days"
  )
```



- Prefer citing the data at the bottom?
- For slide space, I will neglect my labeling in the for the rest of the slides.
- Now we want to add even more context to help the reader understand whether or not birth rates on national holidays behave like weekends.

Context: Adding Holidays

```r
library(lubridate)
holidays <-
  data.frame(date = ymd("2015-01-01","2015-05-25", "2015-07-04",
                        "2015-12-25", "2015-11-26", "2015-12-24",
                        "2015-09-07"),
             occasion = c("New Year", "Memorial Day",
                          "Independence Day", "Christmas",
                          "Thanksgiving", "Christmas Eve",
                          "Labor Day"),
             emoji = c("1f389", "1f396", "1f386", "1f384",
                       "1f983", "1f381", "1f477"))

holidays <- left_join(holidays, Births2015)
```

Context: Adding Holidays

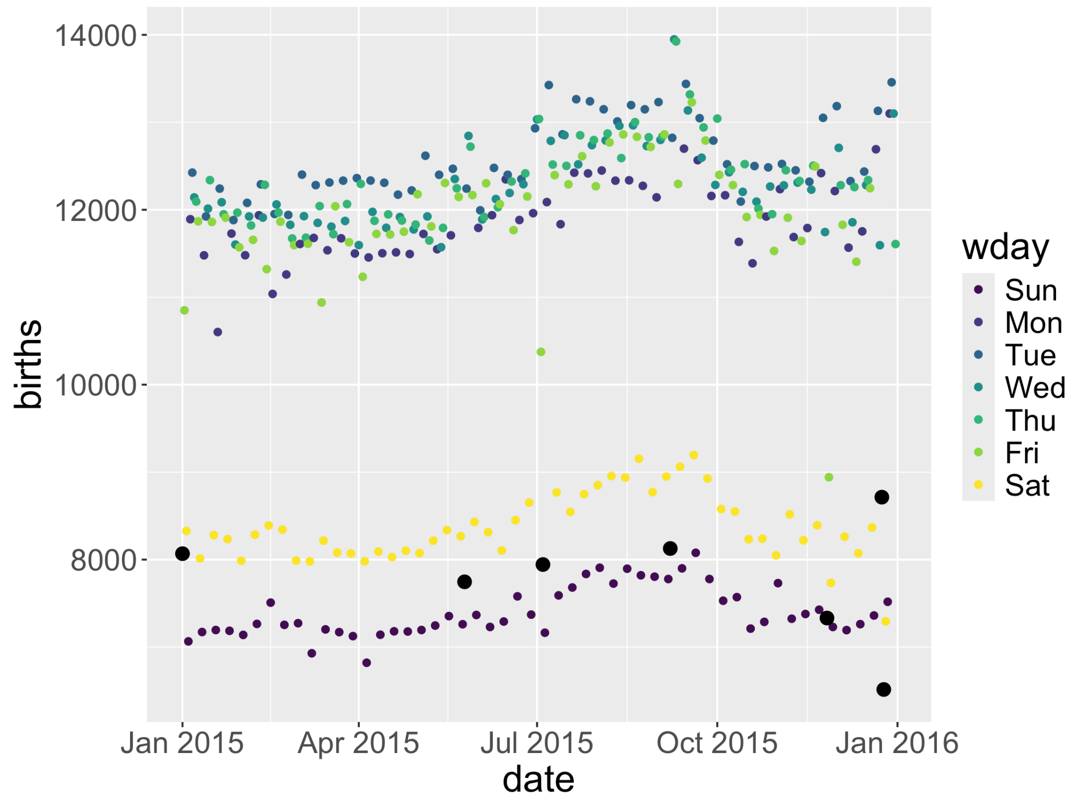
```r
glimpse(holidays)
```

```
Rows: 7
Columns: 10
$ date         <date> 2015-01-01, 2015-05-25, 2015-07-04, 2015-12-25, 2015-11-…
$ occasion     <chr> "New Year", "Memorial Day", "Independence Day", "Christma…
$ emoji        <chr> "1f389", "1f396", "1f386", "1f384", "1f983", "1f381", "1f…
$ births       <dbl> 8068, 7746, 7944, 6515, 7332, 8714, 8127
$ wday         <ord> Thu, Mon, Sat, Fri, Thu, Thu, Mon
$ year         <dbl> 2015, 2015, 2015, 2015, 2015, 2015, 2015
$ month        <dbl> 1, 5, 7, 12, 11, 12, 9
$ day_of_year  <int> 1, 145, 185, 359, 330, 358, 250
$ day_of_month <dbl> 1, 25, 4, 25, 26, 24, 7
$ day_of_week  <dbl> 5, 2, 7, 6, 5, 5, 2
```

- Let’s add some context.

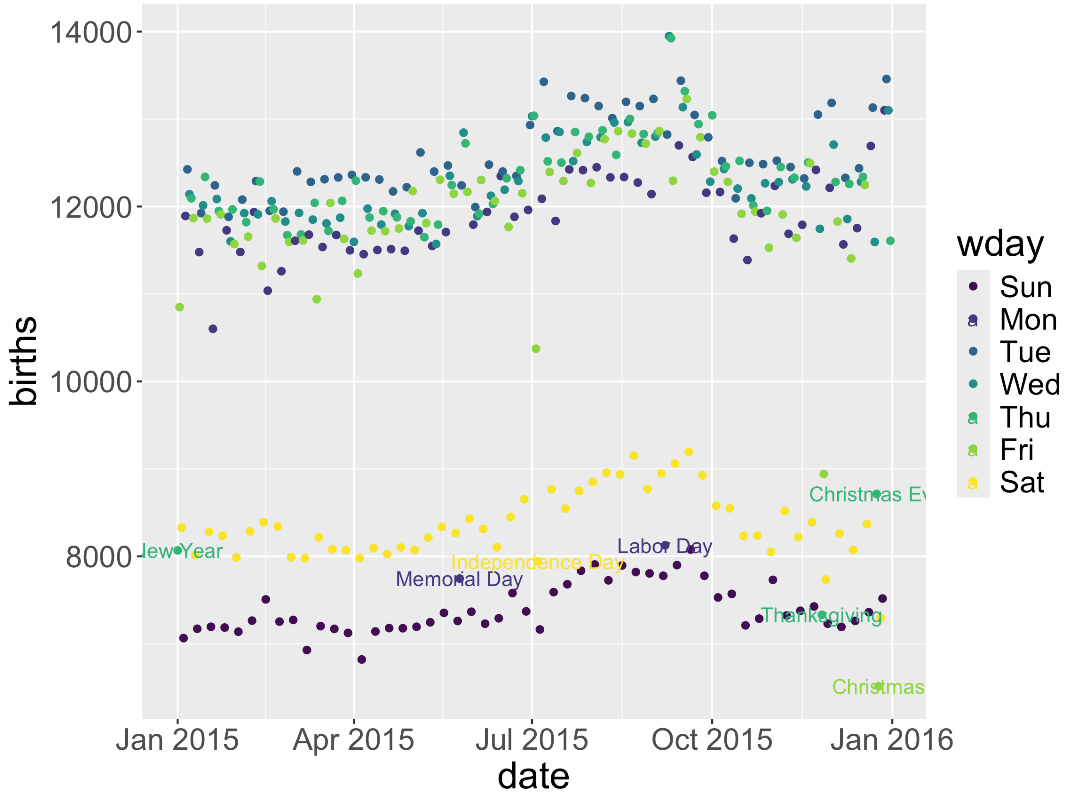
Context: Adding Holidays

```r
ggplot(data = Births2015,
       mapping = aes(x = date, y = births,
                     color = wday)) +
  geom_point() +
  geom_point(data = holidays,
             mapping = aes(x = date, y = births),
             color = "black", size = 3)
```



Context: Adding Holidays

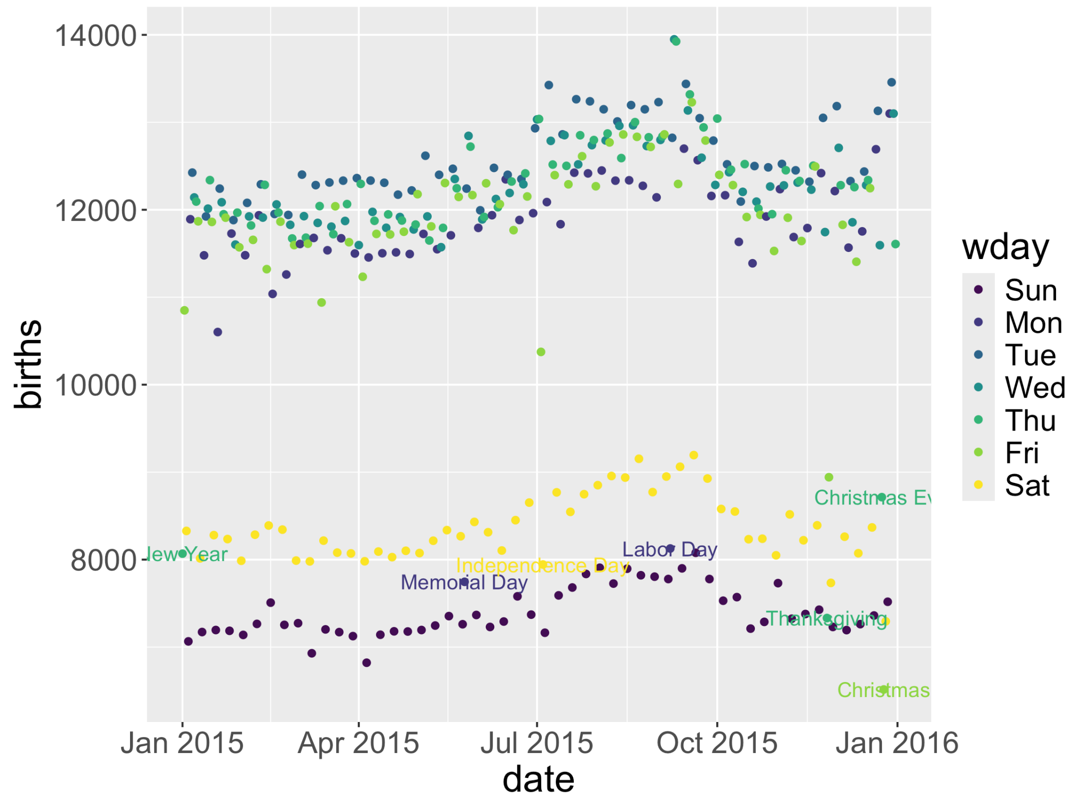
```r
ggplot(data = Births2015,
       mapping = aes(x = date, y = births,
                     color = wday)) +
  geom_point() +
  geom_text(data = holidays,
            mapping = aes(label = occasion))
```



- Problems?

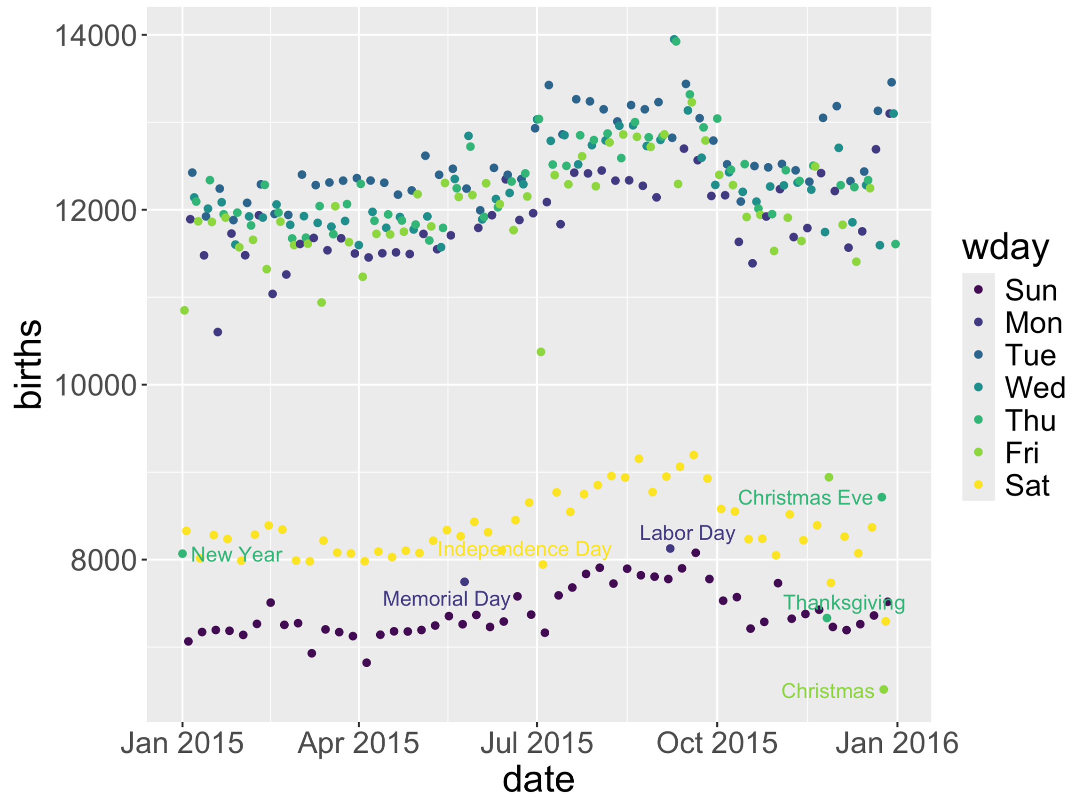
Context: Adding Holidays

```
ggplot(data = Births2015,
       mapping = aes(x = date, y = births,
                     color = wday)) +
  geom_point() +
  geom_text(data = holidays,
            mapping = aes(label = occasion),
            show.legend = FALSE)
```



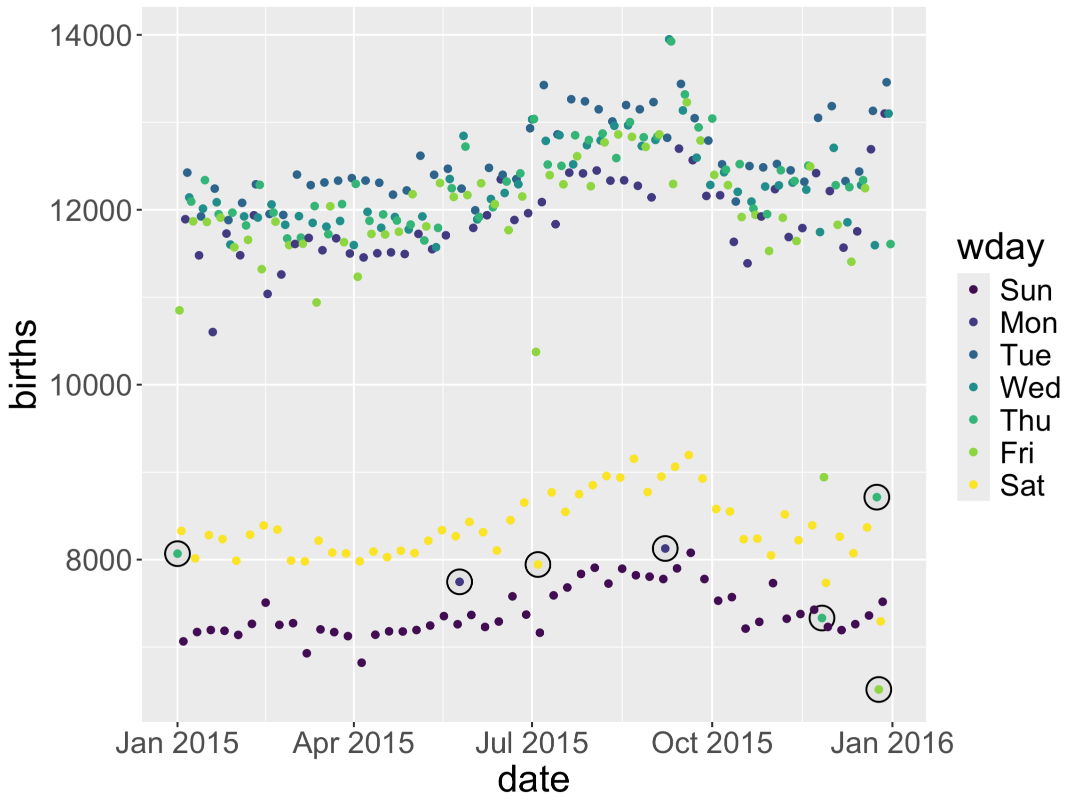
Context: Adding Holidays

```
library(ggrepel)
ggplot(data = Births2015,
       mapping = aes(x = date, y = births,
                     color = wday)) +
  geom_point() +
  geom_text_repel(data = holidays,
          mapping = aes(label = occasion),
          show.legend = FALSE)
```



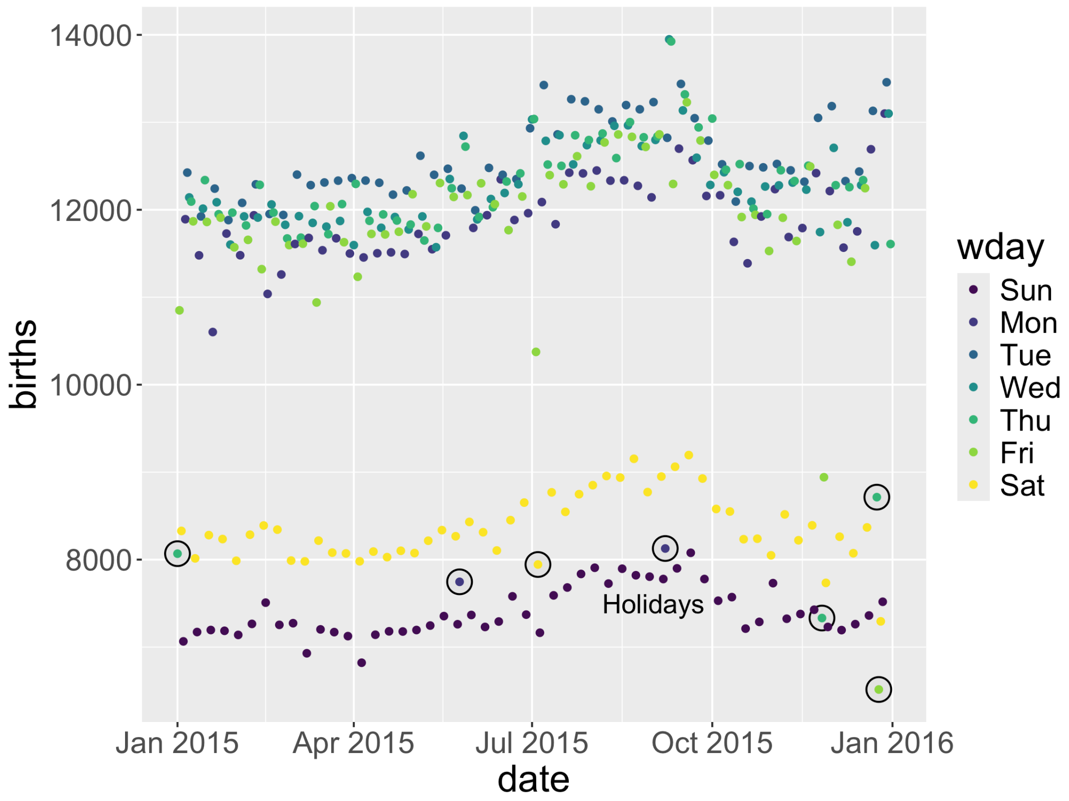
Context: Adding Holidays

```r
ggplot(data = Births2015,
       mapping = aes(x = date, y = births,
                     color = wday)) +
  geom_point(size = 6, color = "black",
             data = holidays) +
  geom_point(size = 5, color = "grey90",
             data = holidays) +
  geom_point()
```



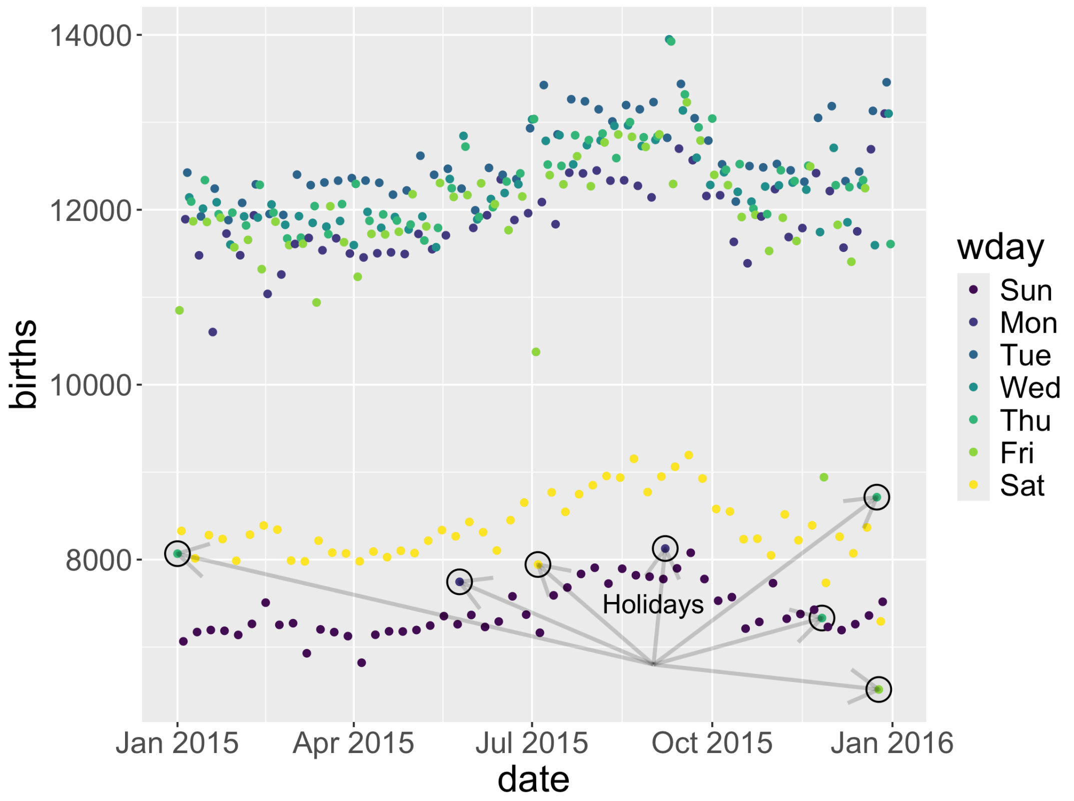
Context: Adding Holidays

```r
ggplot(data = Births2015,
       mapping = aes(x = date, y = births,
                     color = wday)) +
  geom_point(size = 6, color = "black",
             data = holidays) +
  geom_point(size = 5, color = "grey90",
             data = holidays) +
  geom_point() +
  annotate("text", x = as_date("2015-09-01"),
           y = 7500, label = "Holidays",
           color="black", size=5)
```



Context: Adding Holidays

```r
ggplot(data = Births2015,
       mapping = aes(x = date, y = births,
                     color = wday)) +
  geom_point(size = 6, color = "black",
             data = holidays) +
  geom_point(size = 5, color = "grey90",
             data = holidays) +
  geom_point() +
  annotate("segment", colour = "black",
           x = as_date("2015-09-01"),
           xend = holidays$date,
           y = 6800, yend = holidays$births,
           size = 1, alpha = 0.2, arrow = arrow())+
  annotate("text", x = as_date("2015-09-01"),
           y = 7500, label = "Holidays",
           color="black", size=5)
```



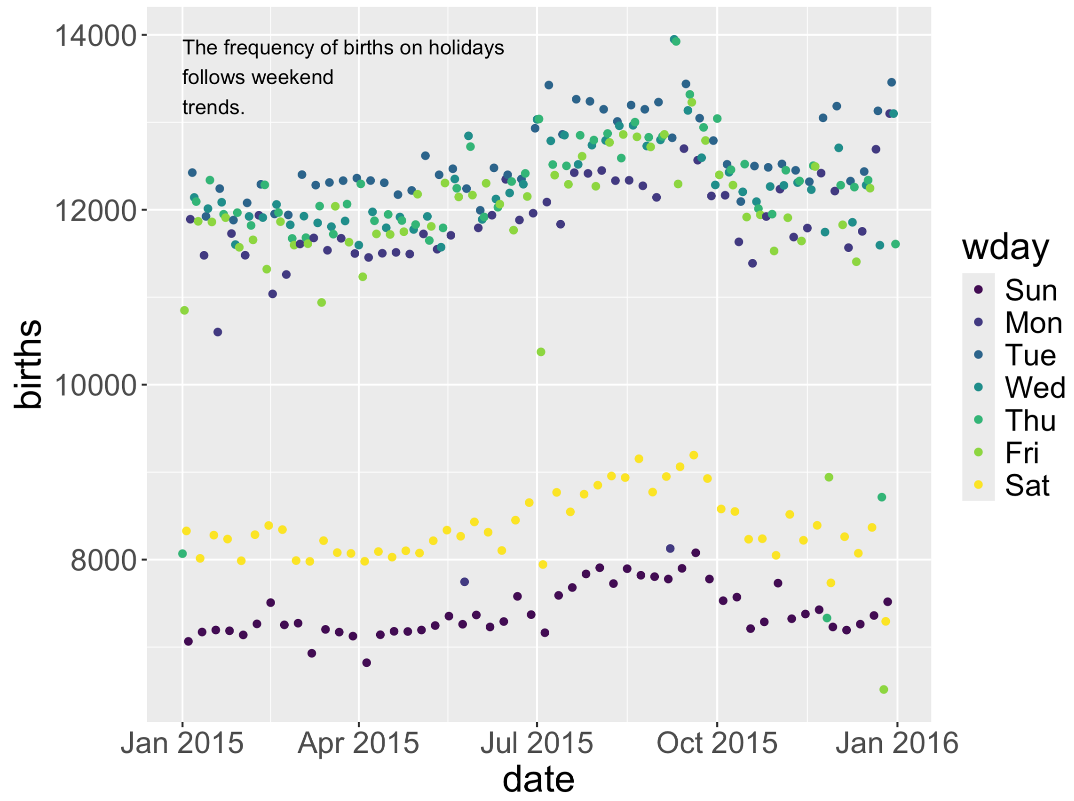
Context: Adding Holidays

```r
# Create a story label
label_data <- data.frame(
  date = ymd("2015-01-01"),
  births = max(Births2015$births),
  label = "The frequency of births on holidays \nfollows weekend \ntrends."
)
```

- What do you think “\n” does?

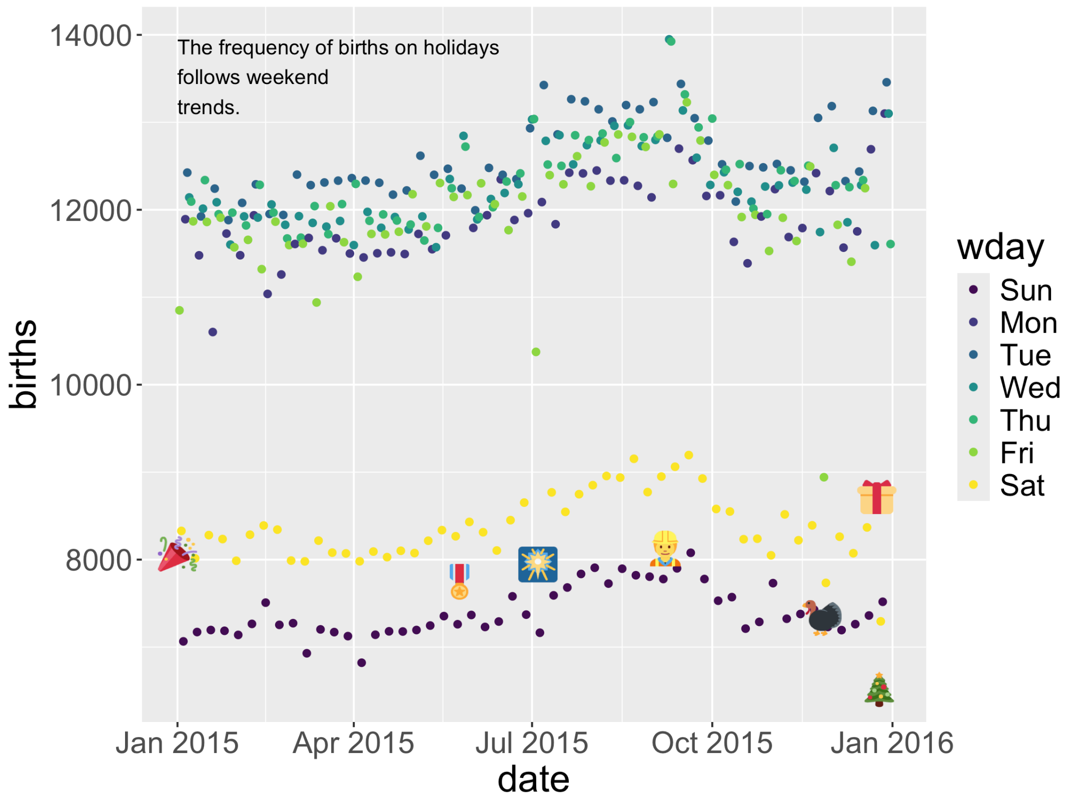
Context: Adding Holidays

```r
ggplot(data = Births2015,
       mapping = aes(x = date, y = births,
                     color = wday)) +
  geom_point() +
  geom_text(mapping = aes(label = label),
            data = label_data,
            color = "black", vjust = "top",
            hjust = "left")
```



Context: Adding Holidays

```r
# devtools::install_github("dill/emoGG")
library(emoGG)
ggplot(data = Births2015,
       mapping = aes(x = date, y = births,
                     color = wday)) +
  geom_point() +
  geom_text(mapping = aes(label = label),
            data = label_data,
            color = "black", vjust = "top",
            hjust = "left") +
  geom_emoji(data = holidays,
             mapping = aes(emoji = emoji,
                           x = date,
                           y = births))
```

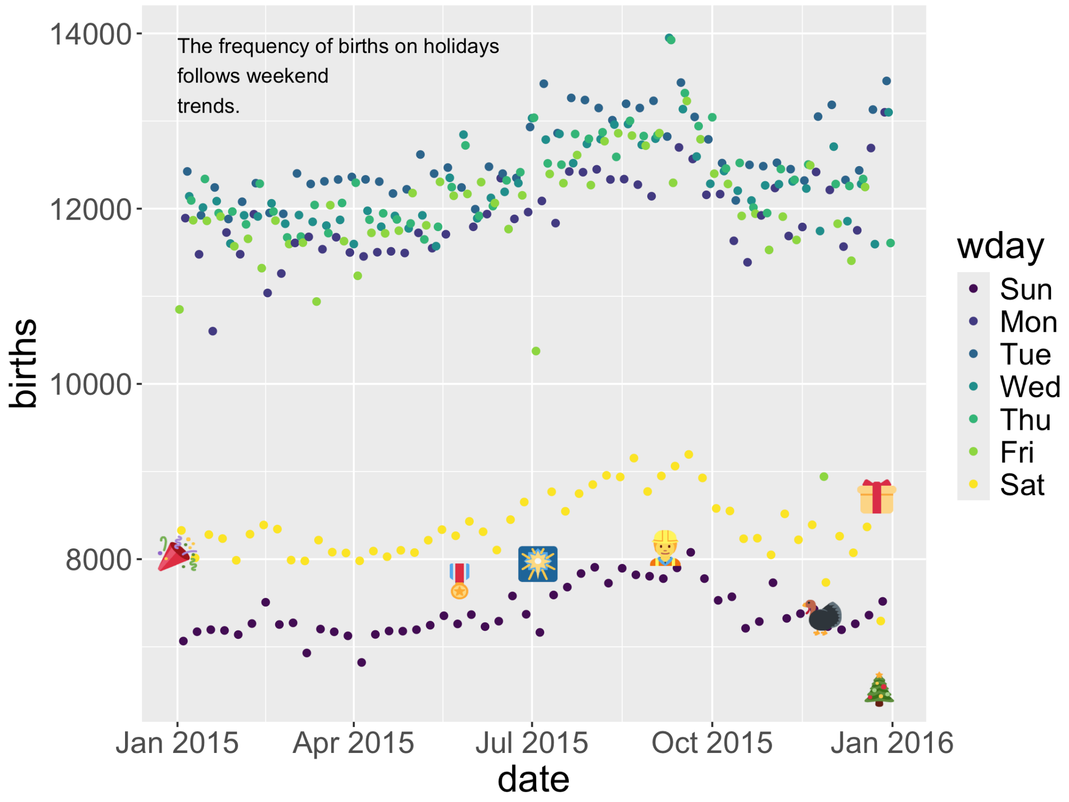


Context

And there are lots more ways to annotate your graph (shaded regions, other context…).

A few notes:

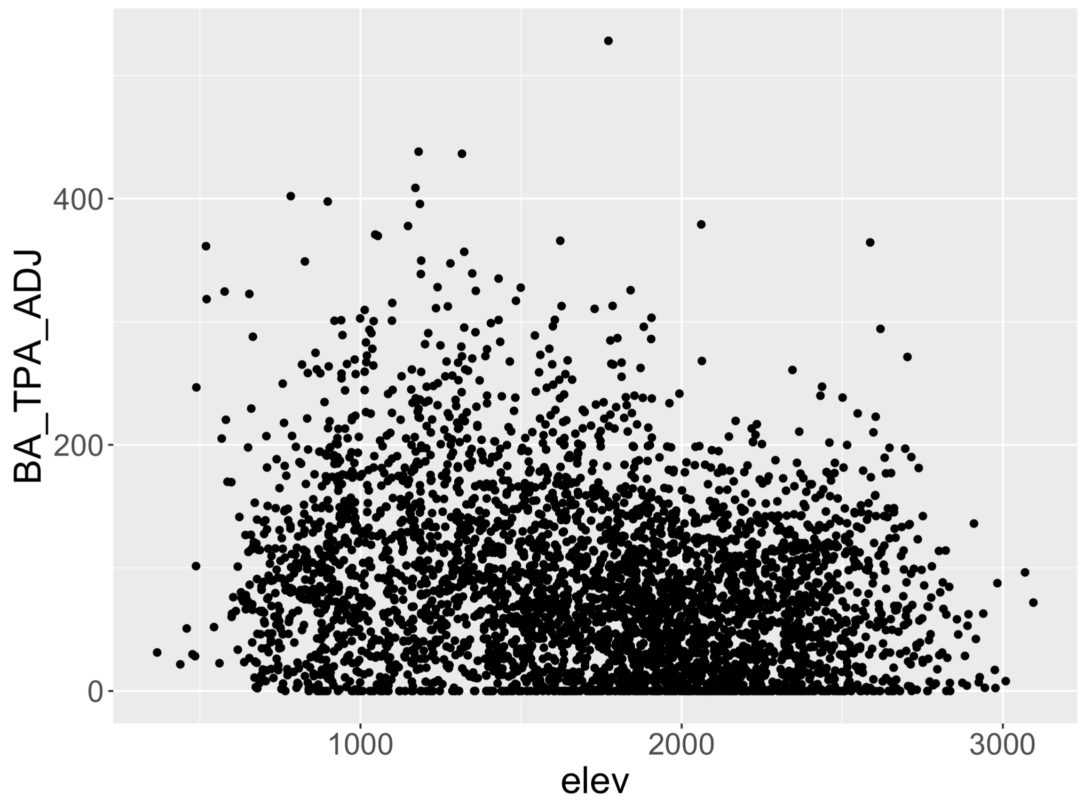
- Don't over do it!
- Like with selecting a `geom` or a `mapping` or a `scale`, try several out first.



Handling Over-plotting

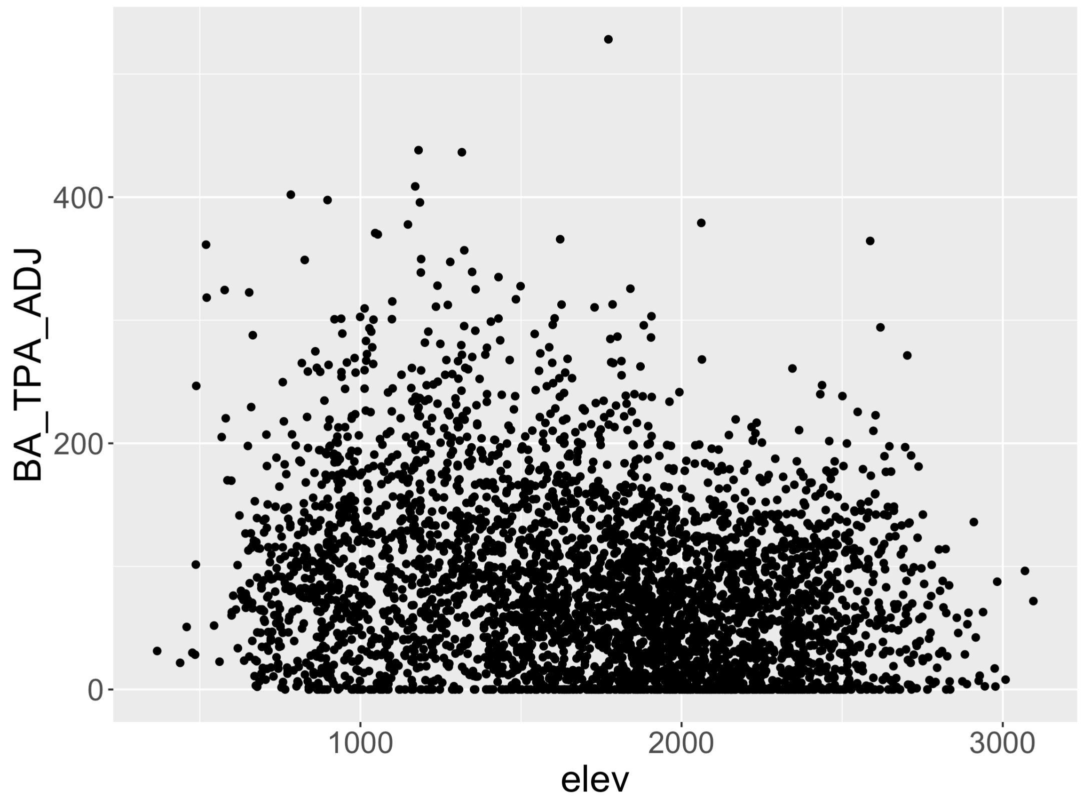
Let’s return to the US Forest Inventory and Analysis Program Idaho data

```r
dat <- readRDS("data/IDdata.rds")
idaho_plots <- dat$pltassgn
ggplot(data = idaho_plots,
       mapping = aes(x = elev,
                     y = BA_TPA_ADJ)) +
  geom_point()
```



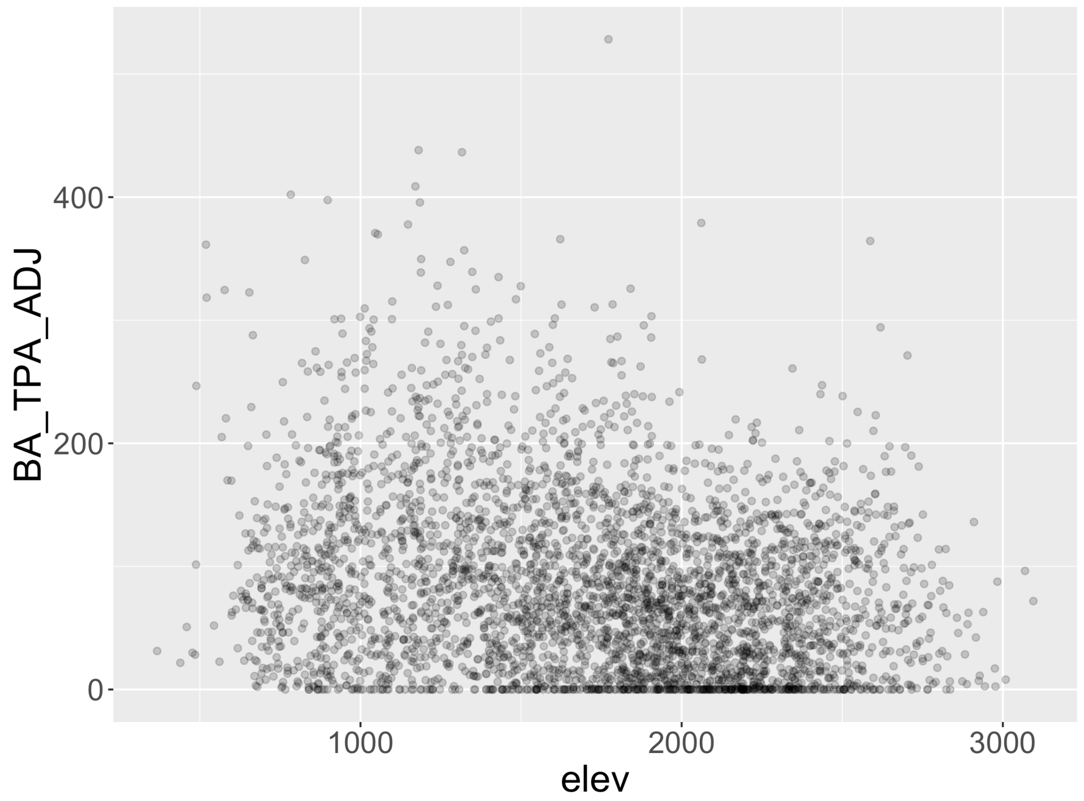
Handling Over-plotting

```r
ggplot(data = idaho_plots,
       mapping = aes(x = elev,
                     y = BA_TPA_ADJ)) +
  geom_jitter()
```



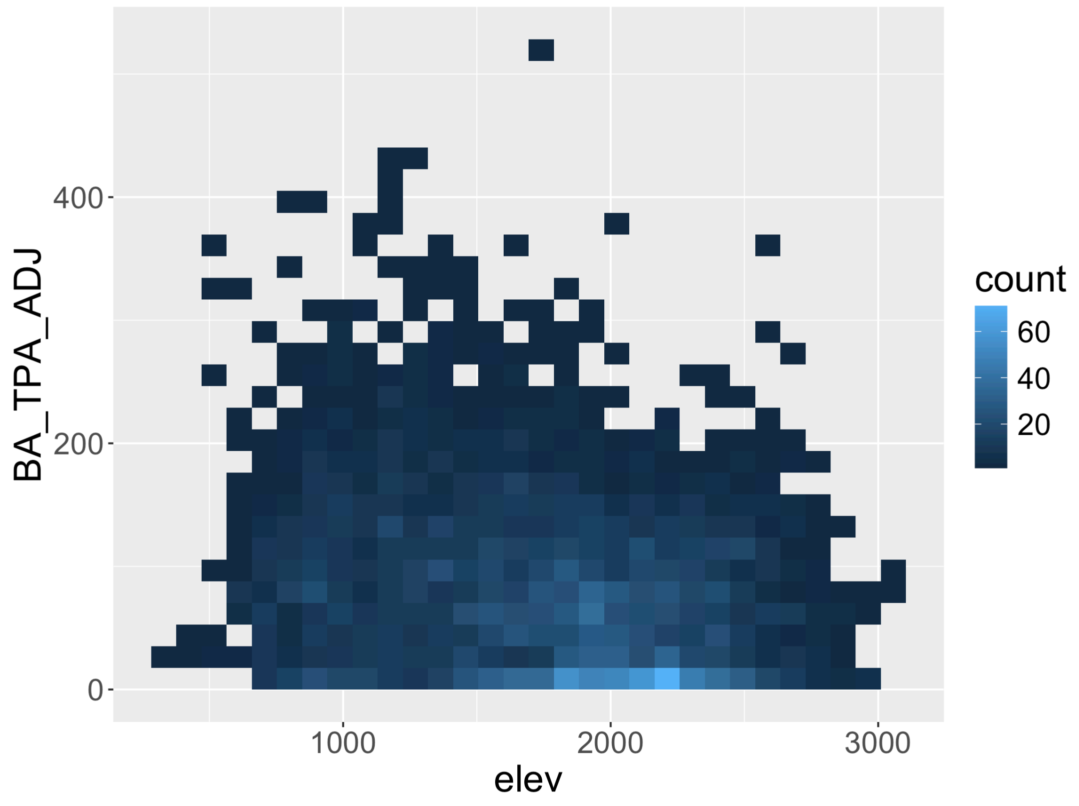
Handling Over-plotting

```
ggplot(data = idaho_plots,
       mapping = aes(x = elev,
                     y = BA_TPA_ADJ)) +
  geom_point(alpha = 0.2)
```



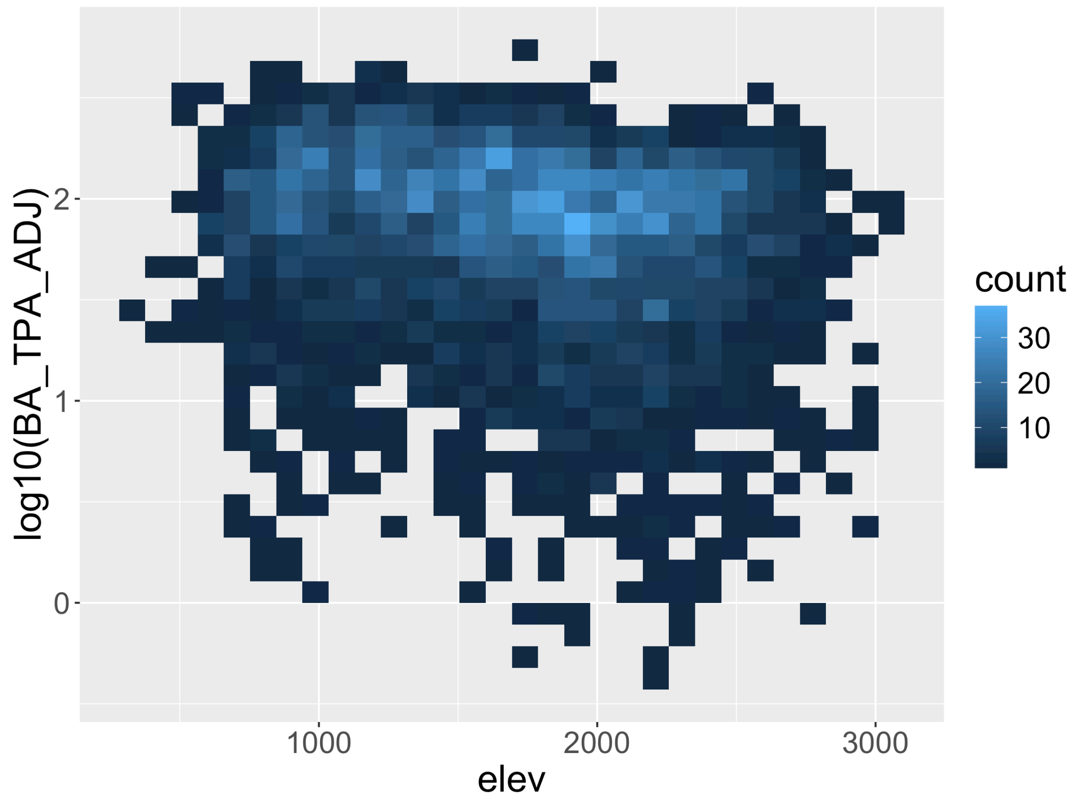
Handling Over-plotting

```r
ggplot(data = idaho_plots,
       mapping = aes(x = elev,
                     y = BA_TPA_ADJ)) +
  geom_bin2d()
```



Handling Transformations

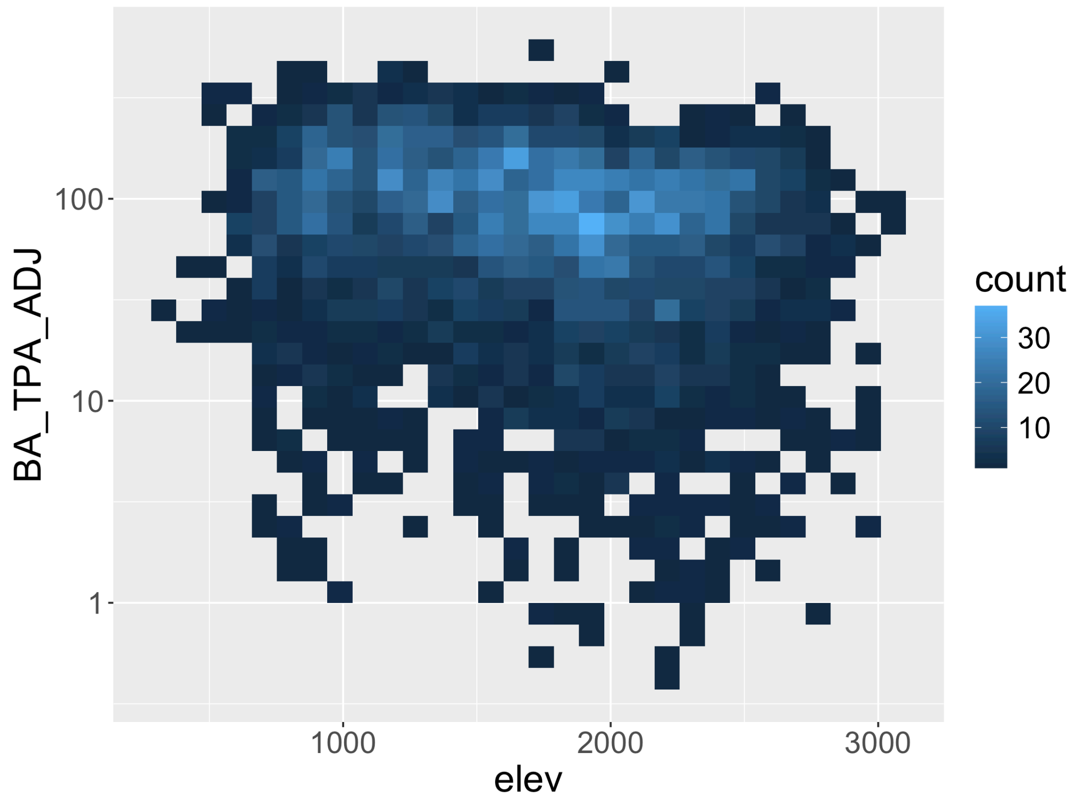
```r
ggplot(data = idaho_plots,
       mapping = aes(x = elev,
                     y = log10(BA_TPA_ADJ))) +
  geom_bin2d()
```



- Transform variables directly.

Handling Transformations

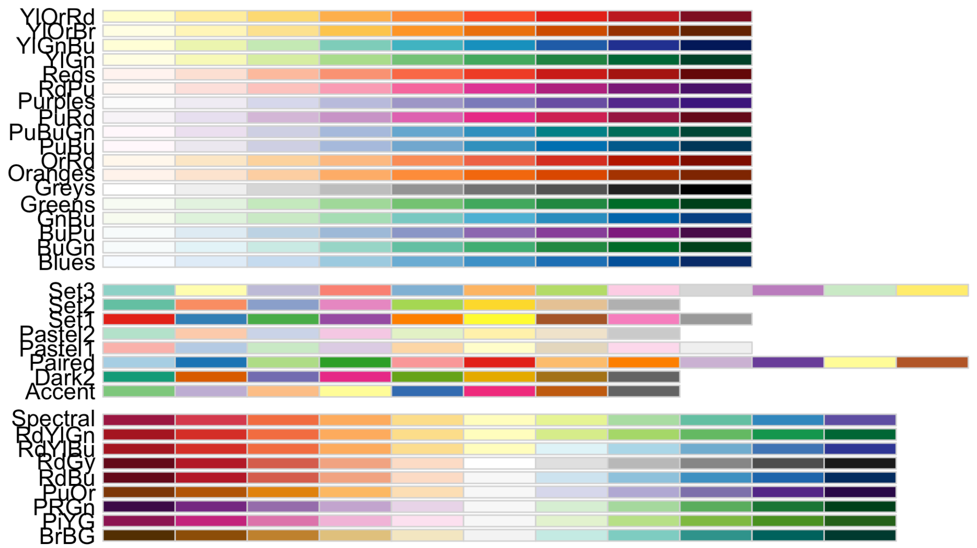
```r
ggplot(data = idaho_plots,
       mapping = aes(x = elev,
                     y = BA_TPA_ADJ)) +
  geom_bin2d() +
  scale_y_log10()
```



- Transform scale.

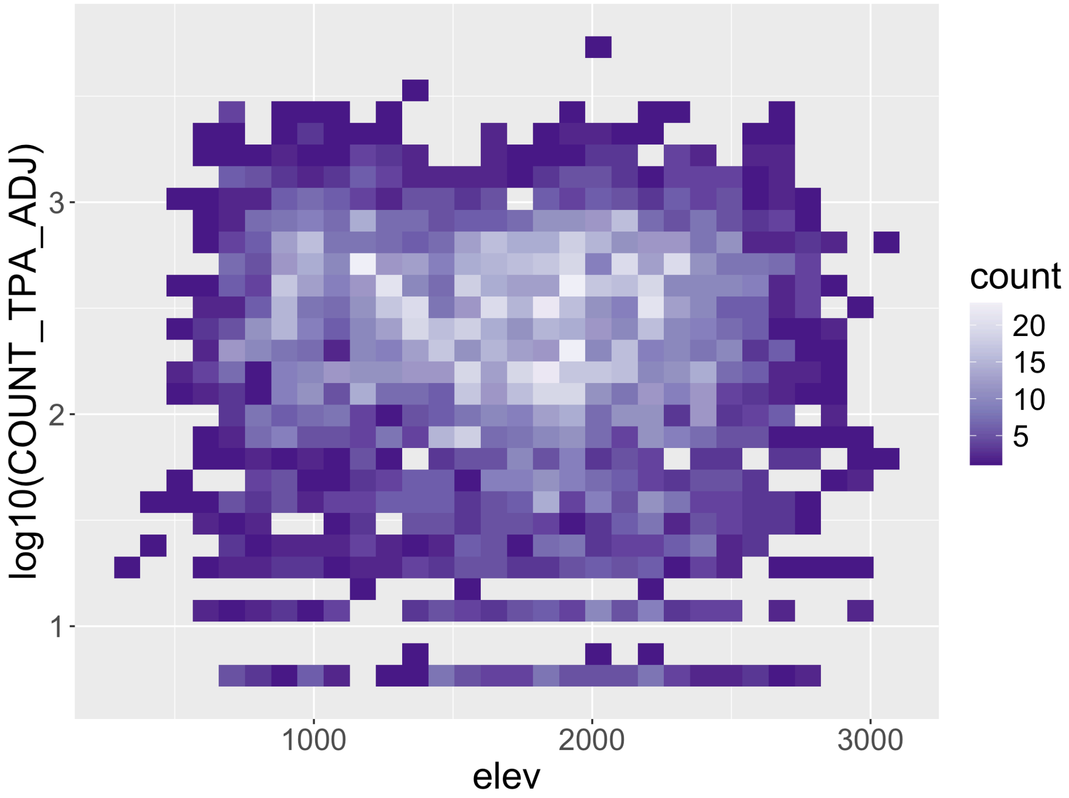
ColorBrewer

```r
RColorBrewer::display.brewer.all()
```



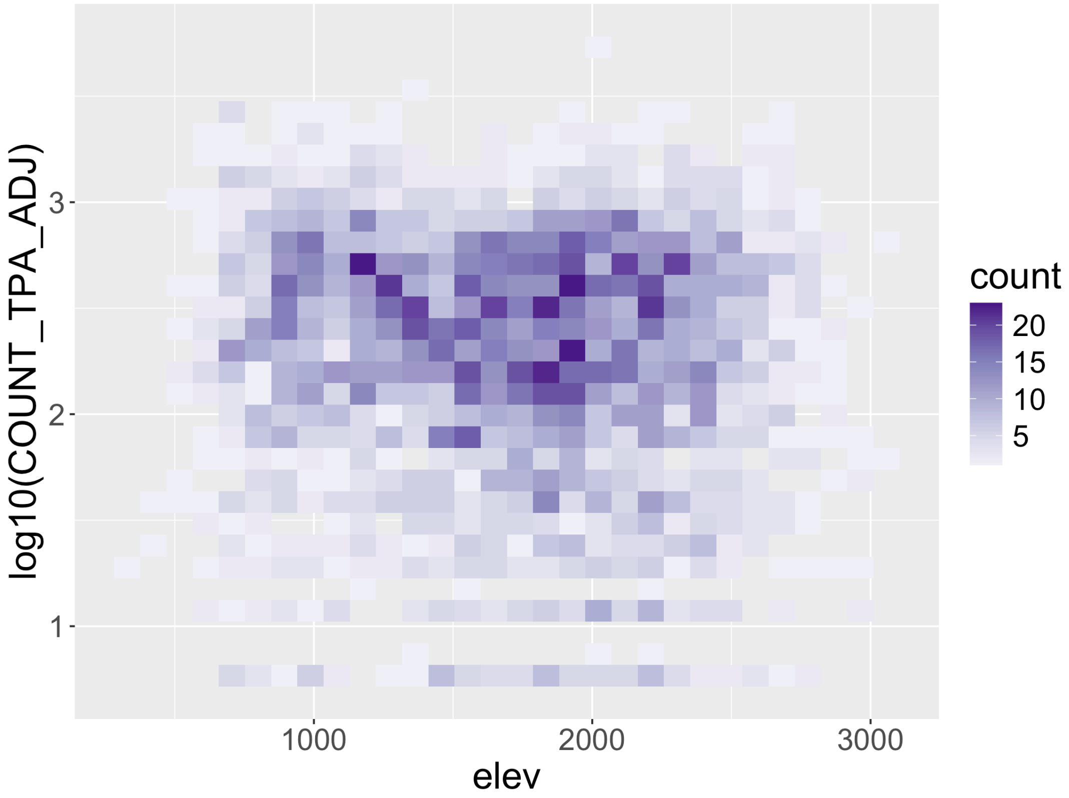
Color Options: Saturation

```r
ggplot(data = idaho_plots,
       mapping = aes(x = elev,
                     y = log10(COUNT_TPA_ADJ))) +
  geom_bin2d() +
  scale_fill_distiller(palette = "Purples")
```



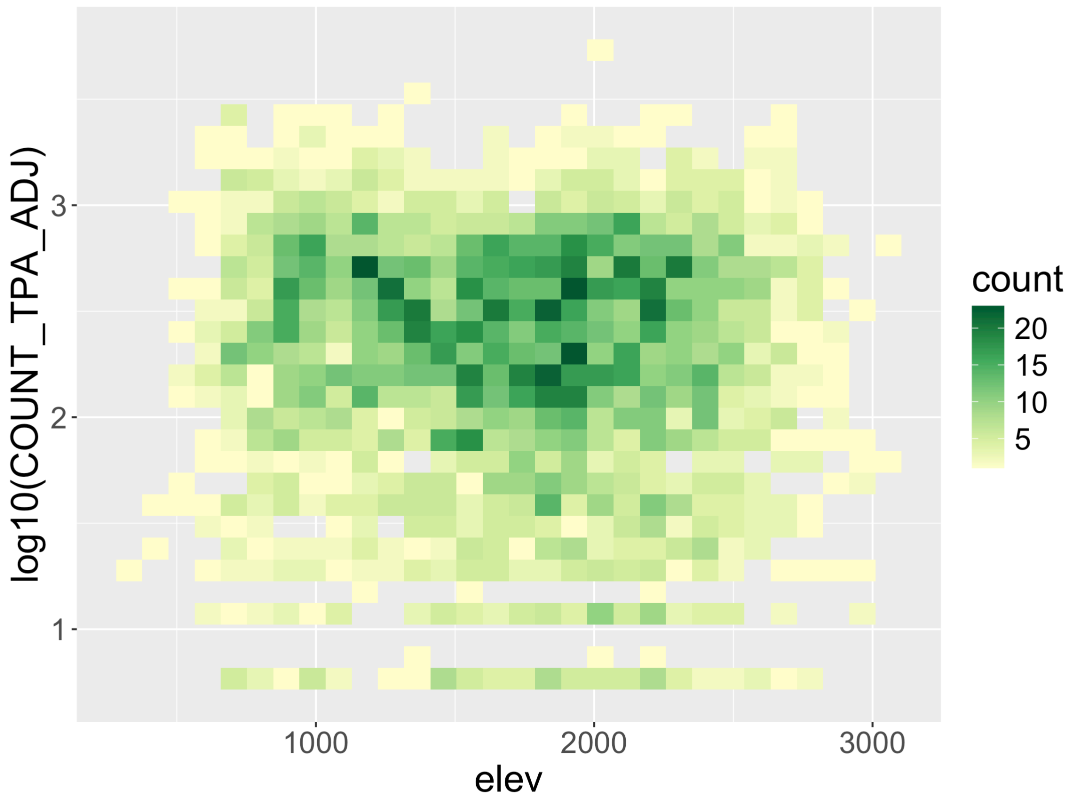
Color Options: Saturation

```r
ggplot(data = idaho_plots,
       mapping = aes(x = elev,
                     y = log10(COUNT_TPA_ADJ))) +
  geom_bin2d() +
  scale_fill_distiller(palette = "Purples",
                       direction = 1)
```



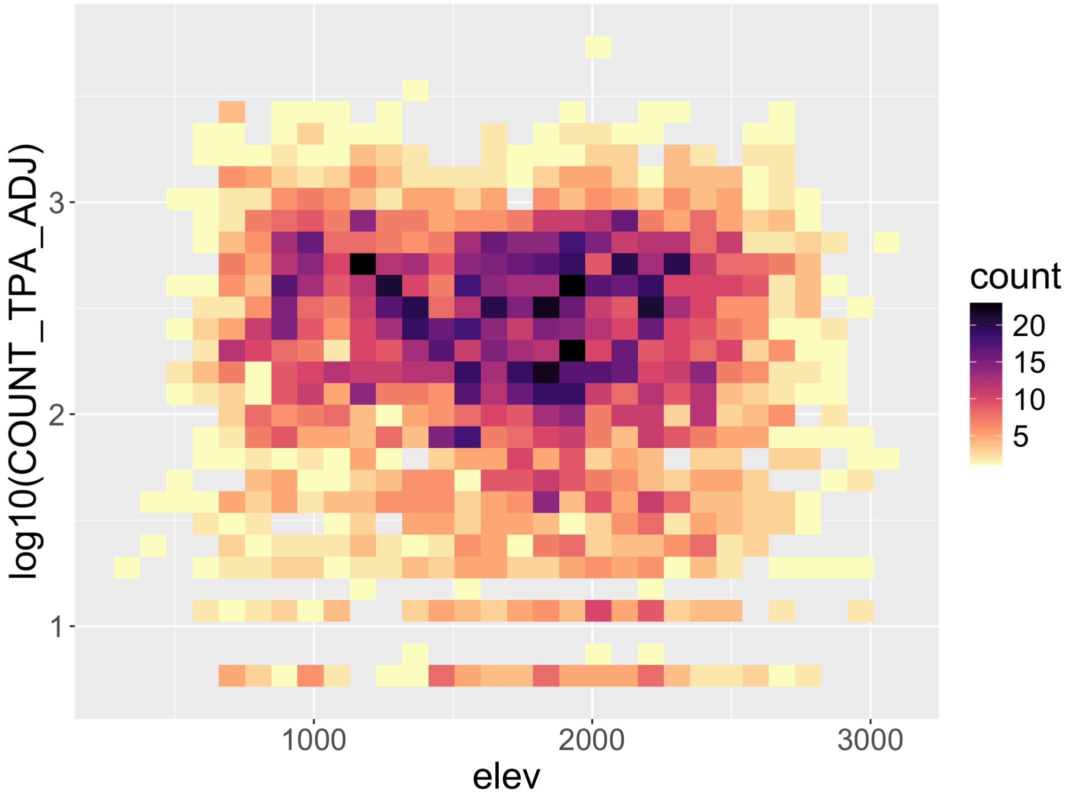
ColorBrewer YlGn palette

```r
ggplot(data = idaho_plots,
       mapping = aes(x = elev,
                     y = log10(COUNT_TPA_ADJ))) +
  geom_bin2d() +
  scale_fill_distiller(palette = "YlGn",
                       direction = 1)
```



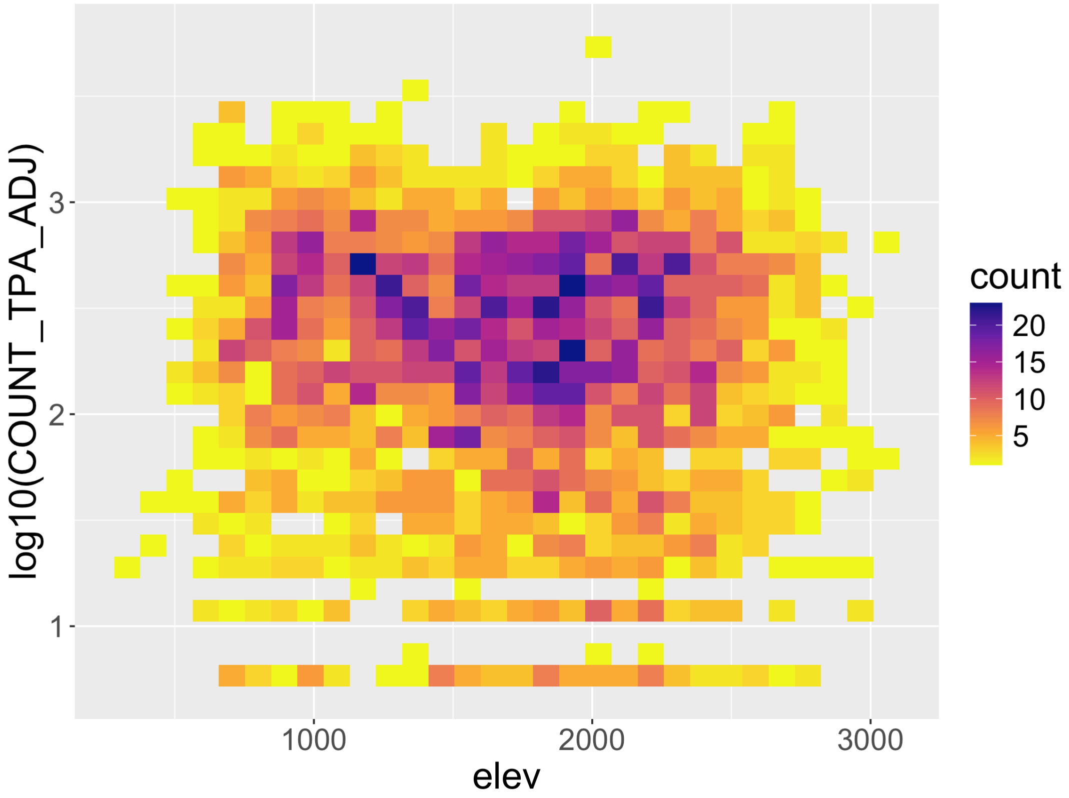
Viridis Palette

```r
library(viridis)
ggplot(data = idaho_plots,
       mapping = aes(x = elev,
                     y = log10(COUNT_TPA_ADJ))) +
  geom_bin2d() +
  scale_fill_viridis_c(direction = -1,
                       option = "A")
```



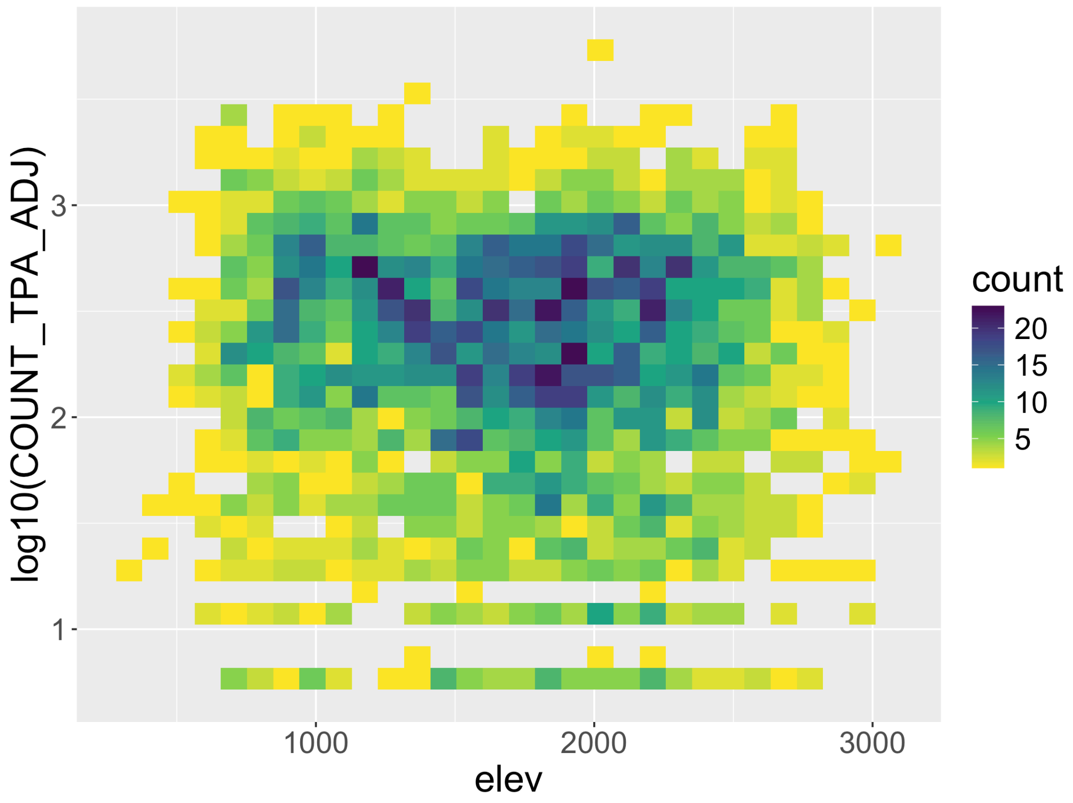
Viridis Palette

```r
library(viridis)
ggplot(data = idaho_plots,
       mapping = aes(x = elev,
                     y = log10(COUNT_TPA_ADJ))) +
  geom_bin2d() +
  scale_fill_viridis_c(direction = -1,
                       option = "C")
```



Viridis Palette

```r
library(viridis)
ggplot(data = idaho_plots,
       mapping = aes(x = elev,
                     y = log10(COUNT_TPA_ADJ))) +
  geom_bin2d() +
  scale_fill_viridis_c(direction = -1,
                       option = "D")
```



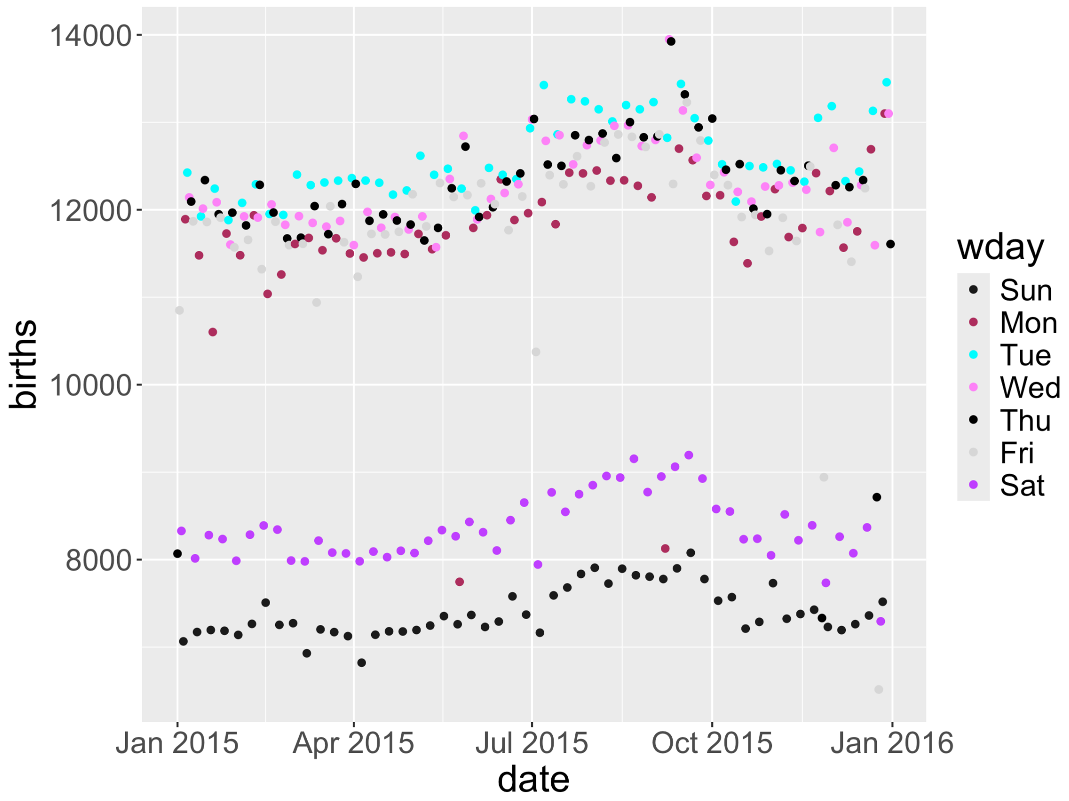
Color Options: Hue

```r
colors()
```

```
 [1] "white"                 "aliceblue"             "antiquewhite"         
 [4] "antiquewhite1"         "antiquewhite2"         "antiquewhite3"        
 [7] "antiquewhite4"         "aquamarine"            "aquamarine1"          
[10] "aquamarine2"           "aquamarine3"           "aquamarine4"          
[13] "azure"                 "azure1"                "azure2"               
[16] "azure3"                "azure4"                "beige"                
[19] "bisque"                "bisque1"               "bisque2"              
[22] "bisque3"               "bisque4"               "black"                
[25] "blanchedalmond"        "blue"                  "blue1"                
[28] "blue2"                 "blue3"                 "blue4"                
[31] "blueviolet"            "brown"                 "brown1"               
[34] "brown2"                "brown3"                "brown4"               
[37] "burlywood"             "burlywood1"            "burlywood2"           
[40] "burlywood3"            "burlywood4"            "cadetblue"            
```

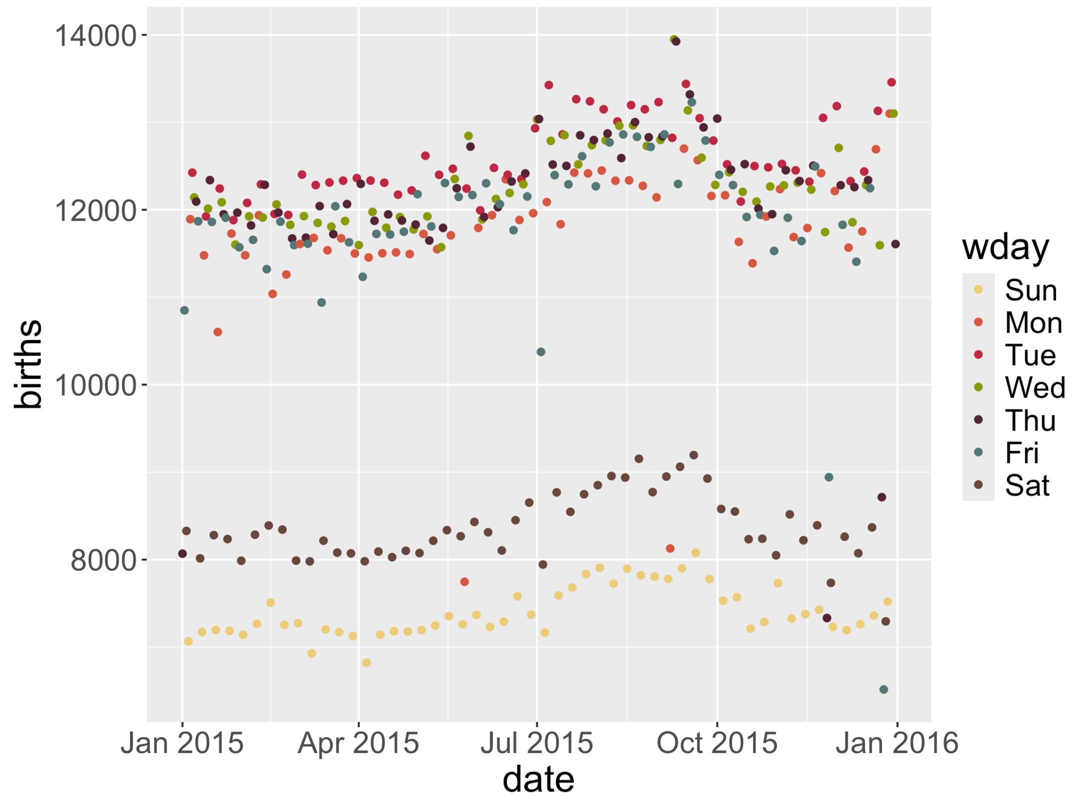
Color Options: Hue

```r
ggplot(data = Births2015,
       mapping = aes(x = date, y = births,
                     color = wday)) +
  geom_point() +
  scale_color_manual(values =
                       sample(colors(), 7))
```



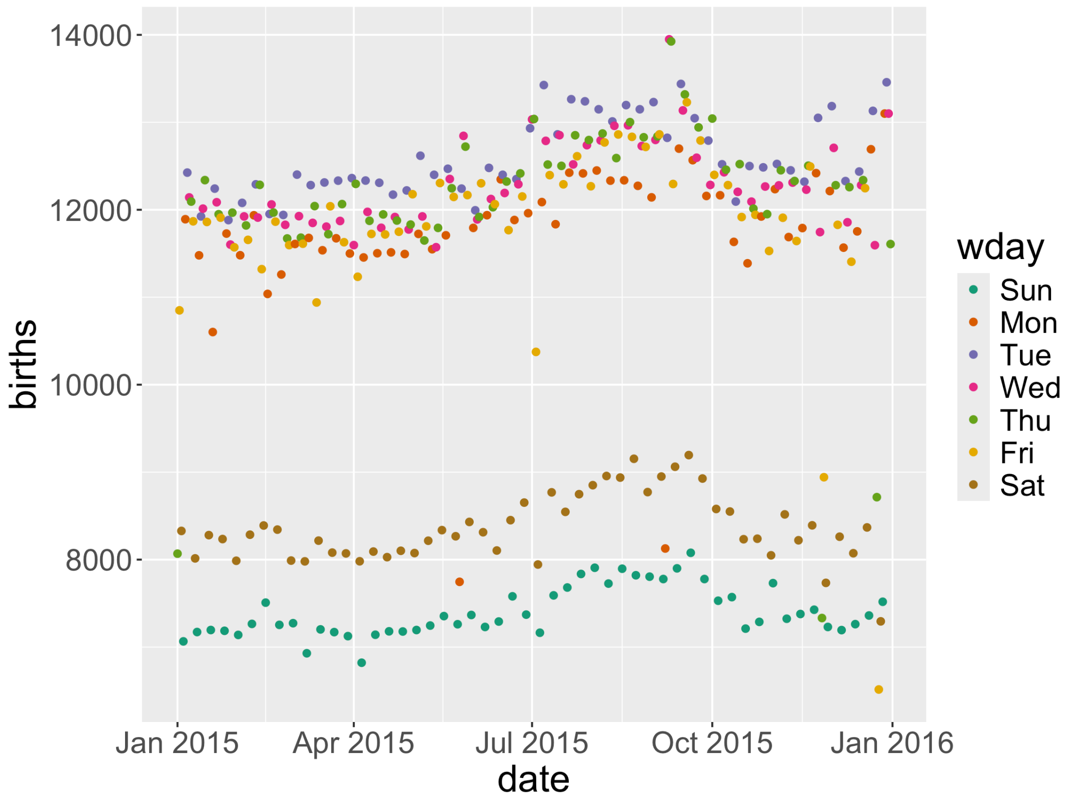
Color Options: Hue

```
ggplot(data = Births2015,
       mapping = aes(x = date, y = births,
                     color = wday)) +
  geom_point() +
  scale_color_manual(values=
                       c("#ECD078", "#D95B43",
                         "#C02942", "#8A9B0F",
                         "#542437", "#53777A",
                         "#6A4A3C"))
```



ColorBrewer Hue

```r
ggplot(data = Births2015,
       mapping = aes(x = date, y = births,
                     color = wday)) +
  geom_point() +
  scale_color_brewer(palette = "Dark2")
```



Themes

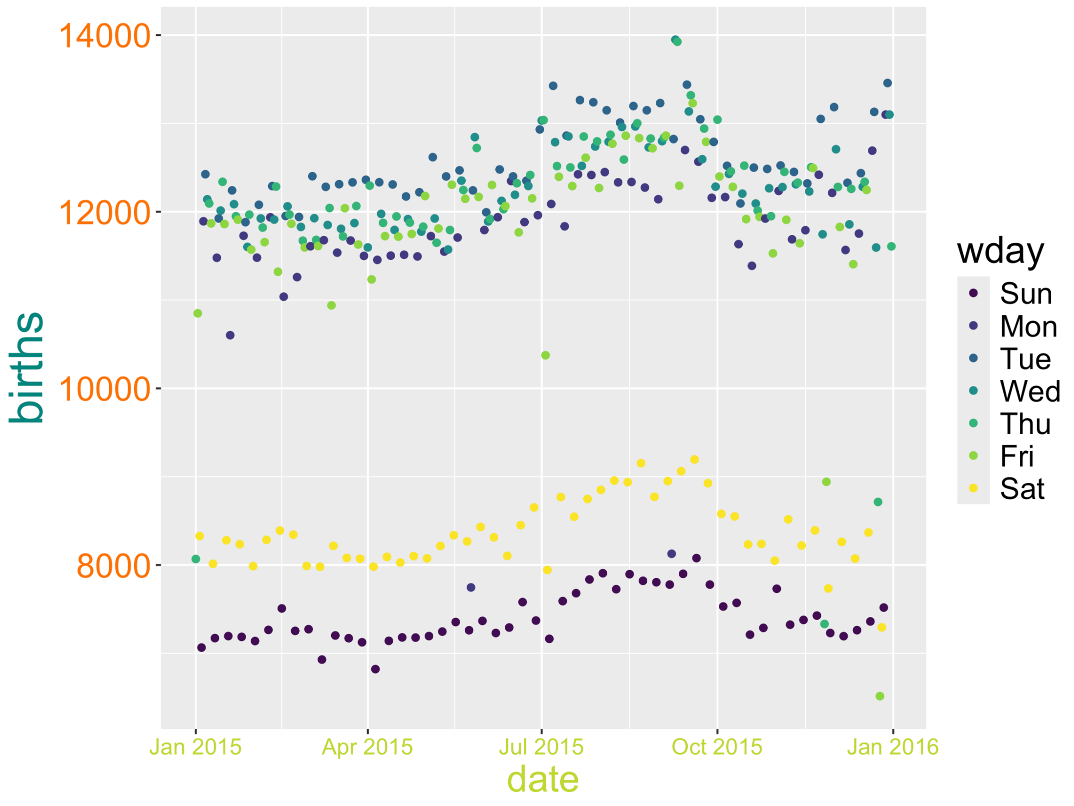
```r
# ?theme
```

- Can override specific aspects of the theme
  - EX: `+ theme(legend.position = "bottom")`
- Can globally set theme options:

```r
theme_update(text = element_text(size = 20))
```

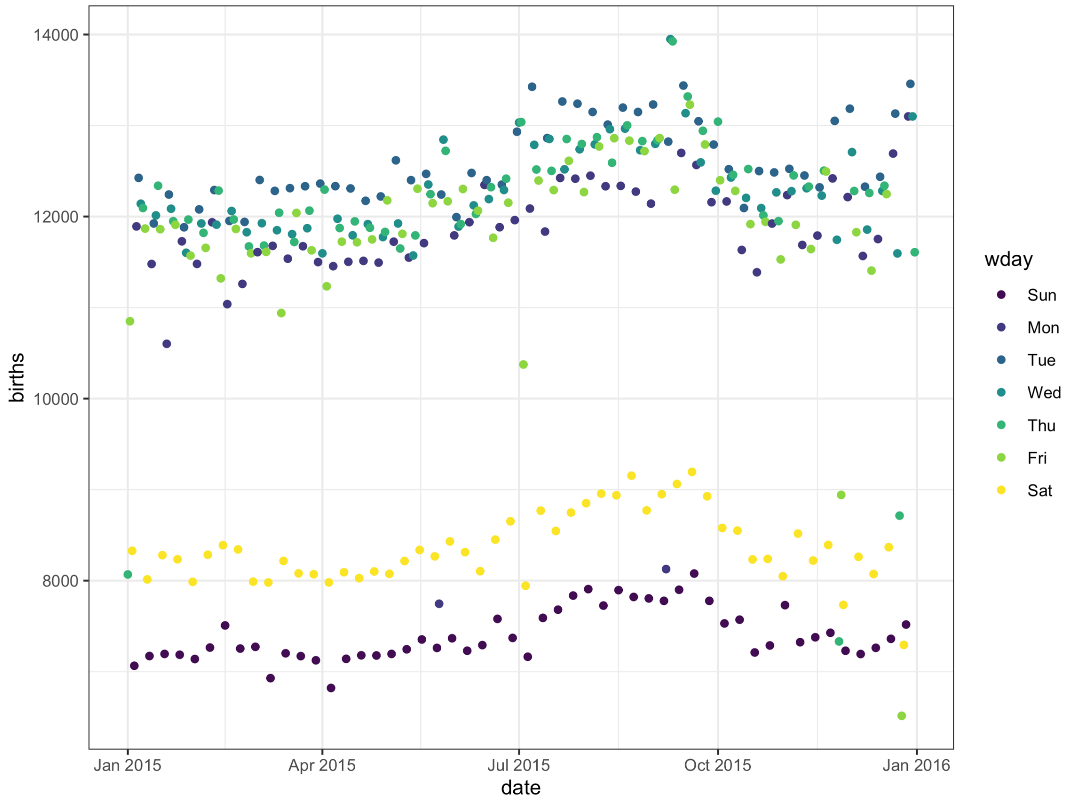
Themes

```r
ggplot(data = Births2015,
       mapping = aes(x = date, y = births,
                     color = wday)) +
  geom_point()  +
  theme(axis.title.x =
        element_text(color = "#C1D82F", size = 20),
        axis.title.y =
        element_text(color = "#00857D", size = 25),
        axis.text.x =
        element_text(color = "#C1D82F", size = 12),
        axis.text.y =
        element_text(color = "#FF7401", size = 18))
```



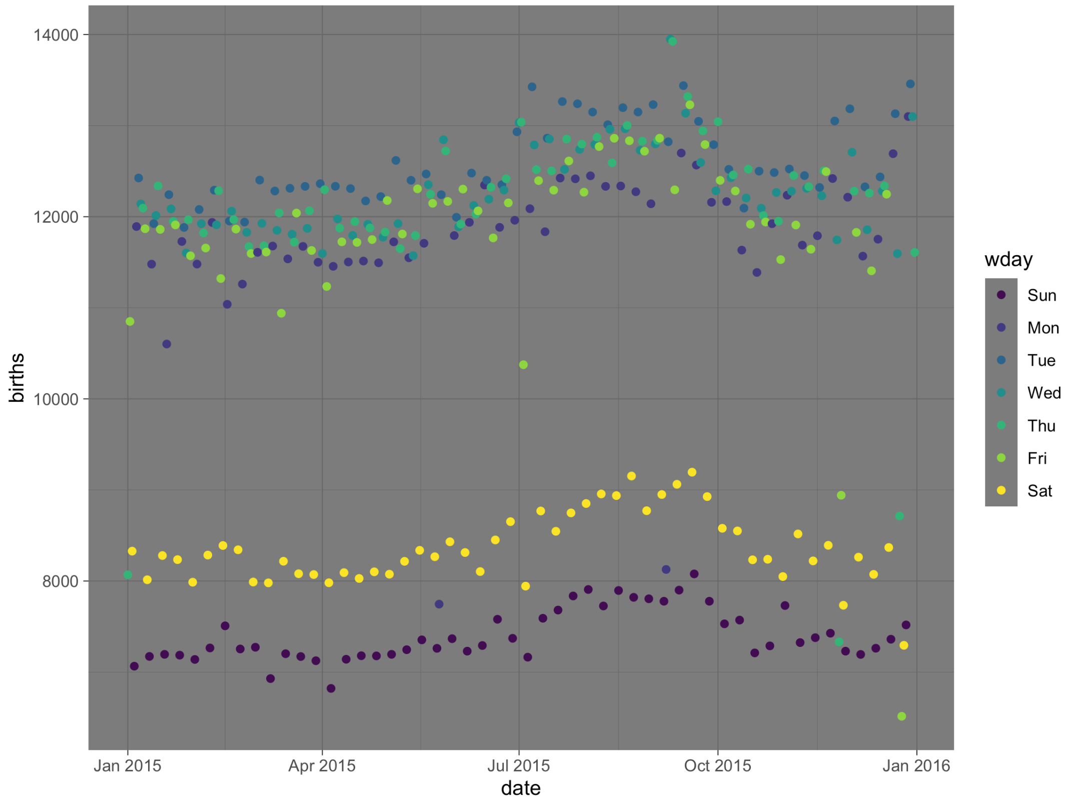
Built-in Themes

```r
ggplot(data = Births2015,
       mapping = aes(x = date, y = births,
                     color = wday)) +
  geom_point()  +
  theme_bw()
```



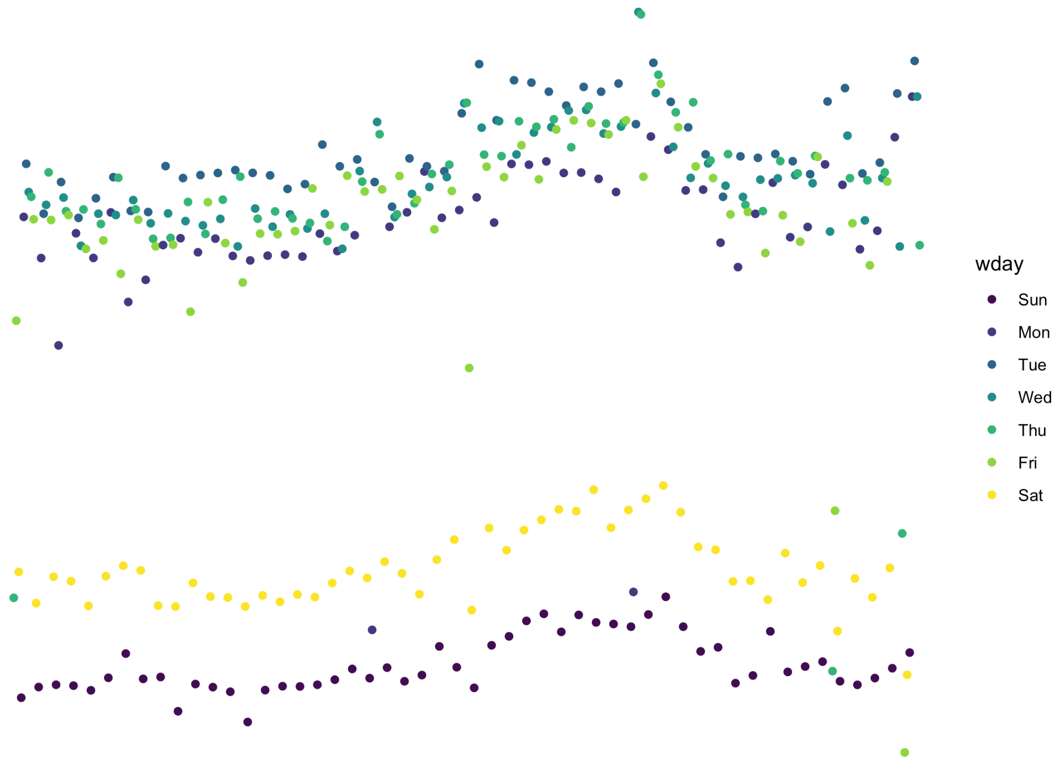
Built-in Themes

```r
ggplot(data = Births2015,
       mapping = aes(x = date, y = births,
                     color = wday)) +
  geom_point()  +
  theme_dark()
```



Built-in Themes

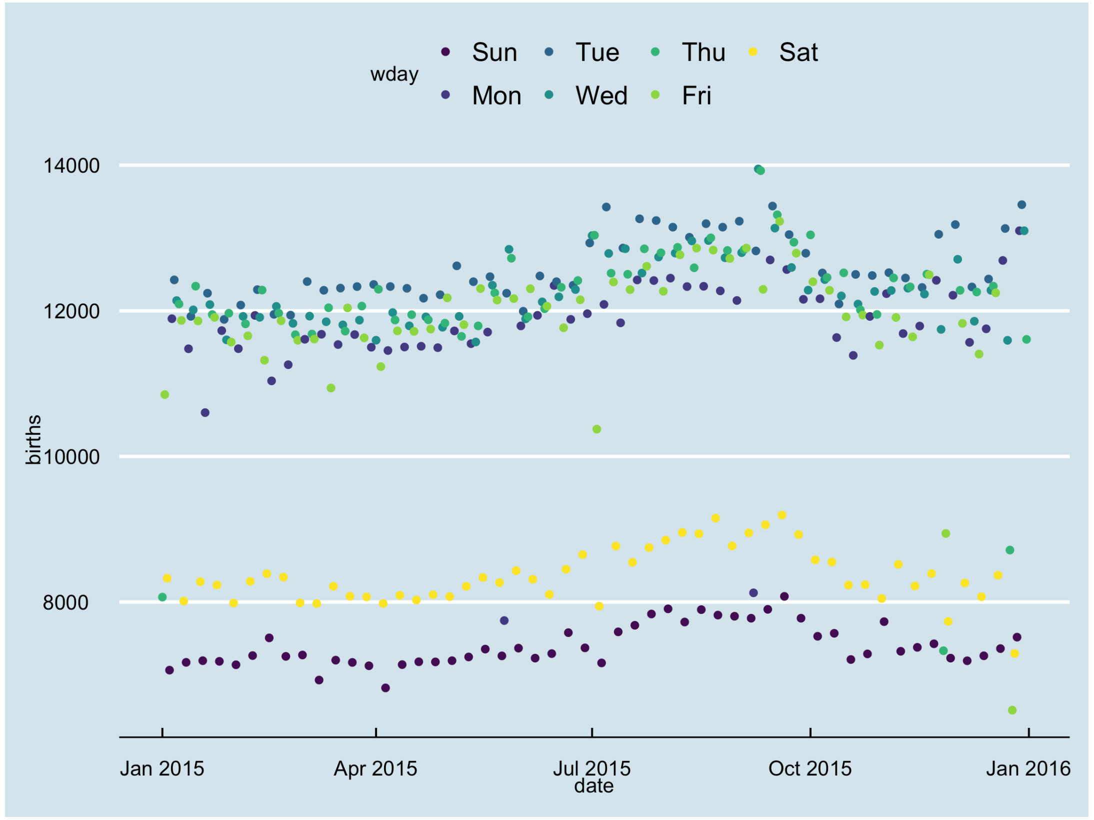
```r
ggplot(data = Births2015,
       mapping = aes(x = date, y = births,
                     color = wday)) +
  geom_point()  +
  theme_void()
```



- Useful for maps and pie charts!

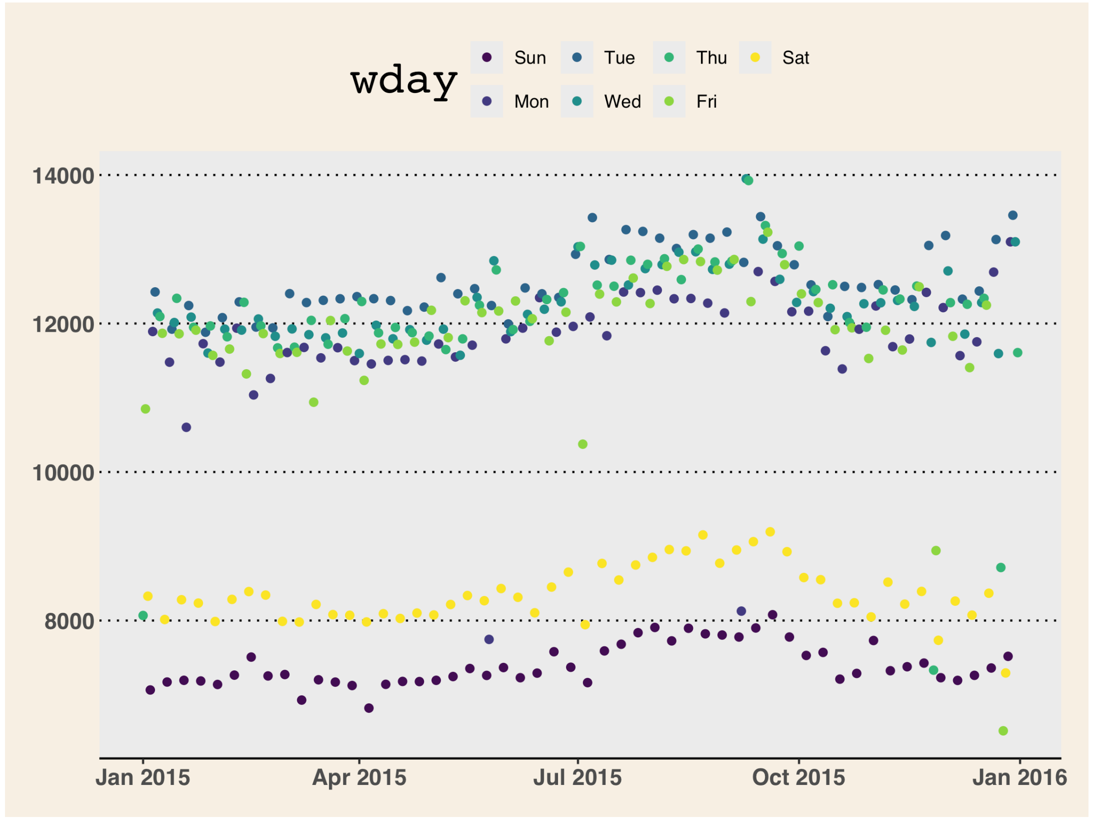
Additional Themes Package: ggthemes

```r
library(ggthemes)
ggplot(data = Births2015,
       mapping = aes(x = date, y = births,
                     color = wday)) +
  geom_point()  +
  theme_economist()
```



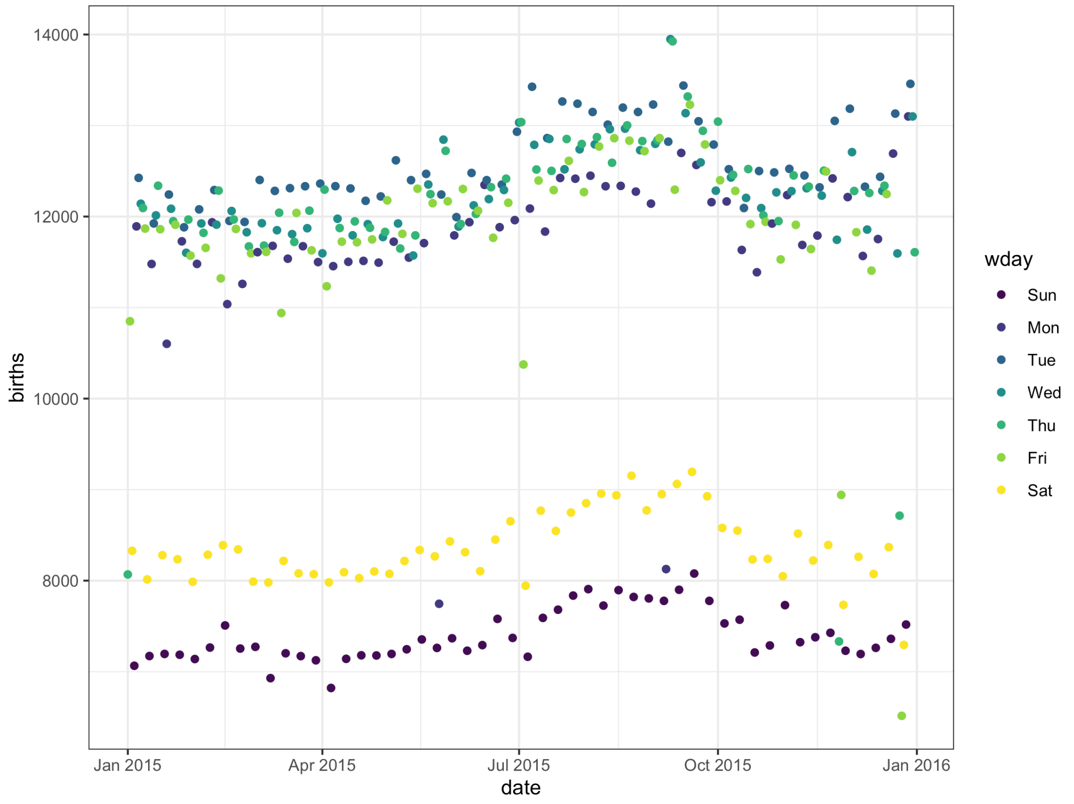
Additional Themes Package

```r
ggplot(data = Births2015,
       mapping = aes(x = date, y = births,
                     color = wday)) +
  geom_point()  +
  theme_wsj()
```



Saving/exporting your figures

```r
p1 <- ggplot(data = Births2015,
        mapping = aes(x = date, y = births,
                      color = wday)) +
  geom_point() +
  theme_bw()

p1
```



```r
ggsave(filename = "my_nice_figure.png",
       plot = p1,
       width = 10,
       height = 6,
       units = "in")
```

Reproducible Workflow

- One where if you shared your data and work with someone else, they could reproduce your results.
- Not the same as **replication**: Where someone collects new data following your same design to see if they get the same results.

- `Quarto` documents allow us to include our R code, output, and narrative in the same place.
  - Load the **raw** data.
  - Be transparent about all the analysis steps.
  - Even if you don’t showcase the R code in the output file, it is contained in the `qmd` file.

Creating `repr`oducible `ex`amples with `reprex`

Why do I need to learn to create reproducible technical examples?

- So that you can ask and answer questions in our class Slack Workspace or Stack Overflow or other R help sites!

First, let’s take a look at some bad examples!

What is wrong with this coding question?

I am trying to create a plot and I can’t get the bars to do what I want them to. Help?!

What is wrong with this coding question?

I want to do the following but it isn’t working:

thing <- read.csv(“long/file/path/thing.csv”)

ggplot(thing, aes(x = factor(that))) + geom_bar()

Help?!

What is wrong with this coding question?

I want to reorder the bars of my plot but can’t get it working. Help!

```r
library(tidyverse)
library(palmerpenguins)

penguins <- penguins %>%
  group_by(species) %>%
  mutate(mean_flipper = mean(flipper_length_mm)) %>%
  ungroup() %>%
  mutate(long = case_when(flipper_length_mm < mean(flipper_length_mm) ~ "no",
                          flipper_length_mm >= mean(flipper_length_mm) ~ "yes"))

penguins %>%
  ggplot(mapping = aes(x = factor(species))) +
  geom_bar()
```

```r
penguins %>%
  count(species)
```

What is wrong with this coding question?

I want to reorder the bars of my plot but can’t get it working. Help!

```r
rm(list = ls())

library(tidyverse)
library(palmerpenguins)

penguins %>%
  ggplot(mapping = aes(x = factor(species))) +
  geom_bar()
```

What makes a good coding question?

- It uses a **minimal** dataset to reproduce the issue.
- It includes the **shortest** amount of **runnable** code necessary to reproduce the issue.
- It doesn't wreak havoc on other people's computers.
- It includes code **and output** so that others don't have to run it!

- It includes any necessary information on the used packages, R version, system, etc.
  - Should not be a concern for our class Slack since we are all on the same RStudio Server.
  - Can use `packageVersion("tidyverse")` or `sessionInfo()` to find this information.

Minimal Dataset: two good options

Create a toy data frame.

```r
dat <- data.frame(animal = c("cat", "dog", "mouse"),
                  weight = c(5, 10, 0.5))
dat
```

```
  animal weight
1    cat    5.0
2    dog   10.0
3  mouse    0.5
```

Use a built-in dataset or a dataset from a particular package.

```r
library(palmerpenguins)
penguins
```

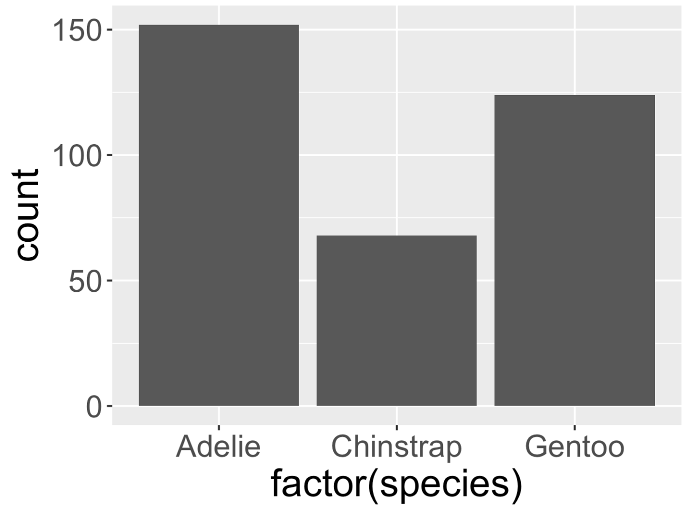
```
# A tibble: 344 × 10
   species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
   <fct>   <fct>              <dbl>         <dbl>             <int>       <int>
 1 Adelie  Torgersen           39.1          18.7               181        3750
 2 Adelie  Torgersen           39.5          17.4               186        3800
 3 Adelie  Torgersen           40.3          18                 195        3250
 4 Adelie  Torgersen           NA            NA                  NA          NA
 5 Adelie  Torgersen           36.7          19.3               193        3450
 6 Adelie  Torgersen           39.3          20.6               190        3650
 7 Adelie  Torgersen           38.9          17.8               181        3625
 8 Adelie  Torgersen           39.2          19.6               195        4675
 9 Adelie  Torgersen           34.1          18.1               193        3475
10 Adelie  Torgersen           42            20.2               190        4250
# ℹ 334 more rows
```

Minimal Code

Include the **necessary** libraries.

Test run the code in a restarted R session to make sure it is runnable!

```r
library(tidyverse)
library(palmerpenguins)

penguins %>%
  ggplot(mapping = aes(x = factor(species))) +
  geom_bar()
```

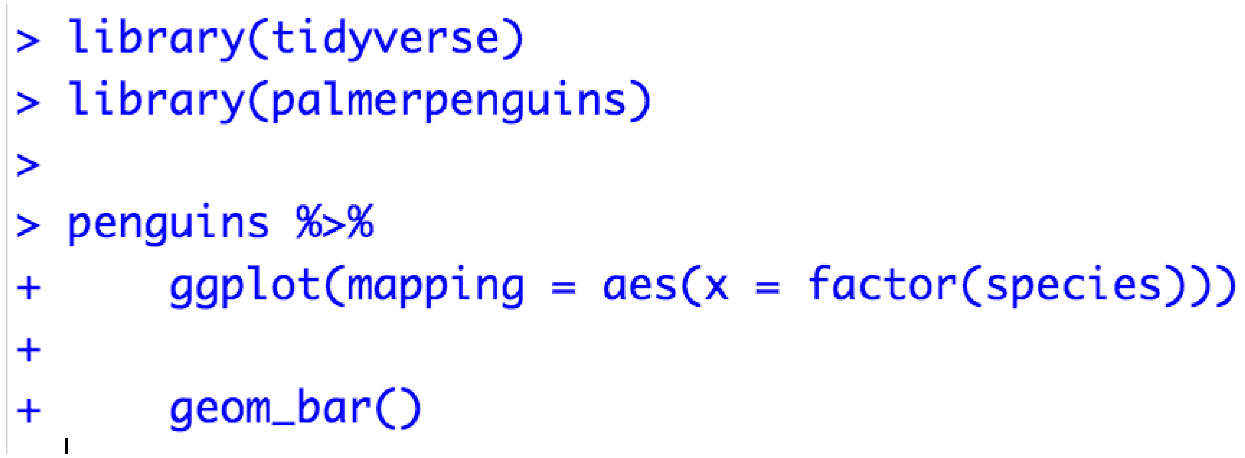


Make sure your code is copy-and-paste-able!

Don’t copy from the console.

```
> library(tidyverse)
> library(palmerpenguins)
>
> penguins %>%
+     ggplot(mapping = aes(x = factor(species))) +
+     geom_bar()
```

Make sure your code is copy-and-paste-able!

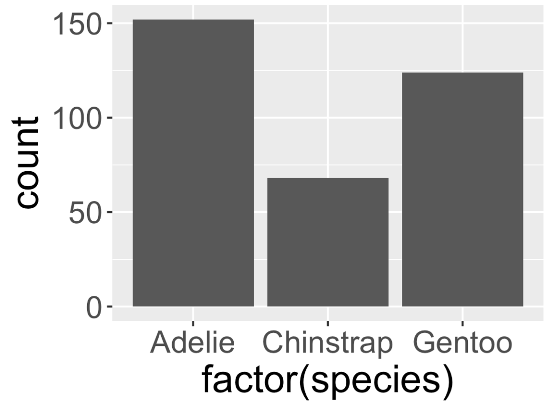


```
> library(tidyverse)
> library(palmerpenguins)
>
> penguins %>%
+     ggplot(mapping = aes(x = factor(species)))
+
+     geom_bar()
```

Now we have our reproducible example:

How can I reorder the bars in the ggplot to go from the most frequent to the least frequent category?

```r
library(tidyverse)
library(palmerpenguins)

penguins %>%
  ggplot(mapping = aes(x = factor(species))) +
  geom_bar()
```



How can we easily share it?

- Using the `reprex()` function in the `reprex` package.

`reprex` Practice Time!

But first: Q: What is an R script file?

- A text file for entering R commands.

Q: How is an R script file different from a Quarto or RMarkdown document?

- You only put code in an R script.
- If you add any text you must comment it out with #.
- Think of it as a single R chunk that you won’t knit into an output document.
- Useful when writing a lot of code and want to compartmentalize.

`reprex` Practice Time!

1. In **Session**, select "Clear Workspace" and then "Restart R".
2. Open a script file and include in the top line:

```r
library(reprex)
```

3. Put the code you want to use in the script file and make sure it runs.

```r
library(tidyverse)
library(palmerpenguins)

penguins %>%
  ggplot(mapping = aes(x = factor(species))) +
  geom_bar()
```

4. Surround the code with `reprex({ ... }, venue = "slack")` and run it.
5. An md file will pop up. Copy all the contents of that file.
6. Head over to the `#coding-qa` channel and paste in the contents as a reply to the `reprex` practice message. A text box will pop-up and select "Apply".
7. Above your code, type your question. Then hit "Send".

Next week

- Learn to make animated figures!
- Hone our data wrangling skills
- P-set 1 due on Thursday 2/12