Lesson 08 - Creating Intermediate graphics

Robin Donatello

Last Updated 01-06-2019

Introduction

The first lesson on graphing focused on creating the correct graphics for the data types. This lesson expands on those tools and demonstrates how to customize some features and add enhancements to clarify or to add information to a plot.

Student Learning Outcomes

After completing this lesson students will be able to

- Create several multivariable graphics
- Change the color of plots
- Create a grid/panel of plots.

Preparation

Prior to this lesson students should

- Download the [[08_plots2_notes.Rmd]] R markdown file and save into your Math130 folder.
- Install the `scales` and `gridExtra` packages.
- Import both the NCbirths, and the email data sets.
- Load the `ggplot2` and `dplyr` packages.

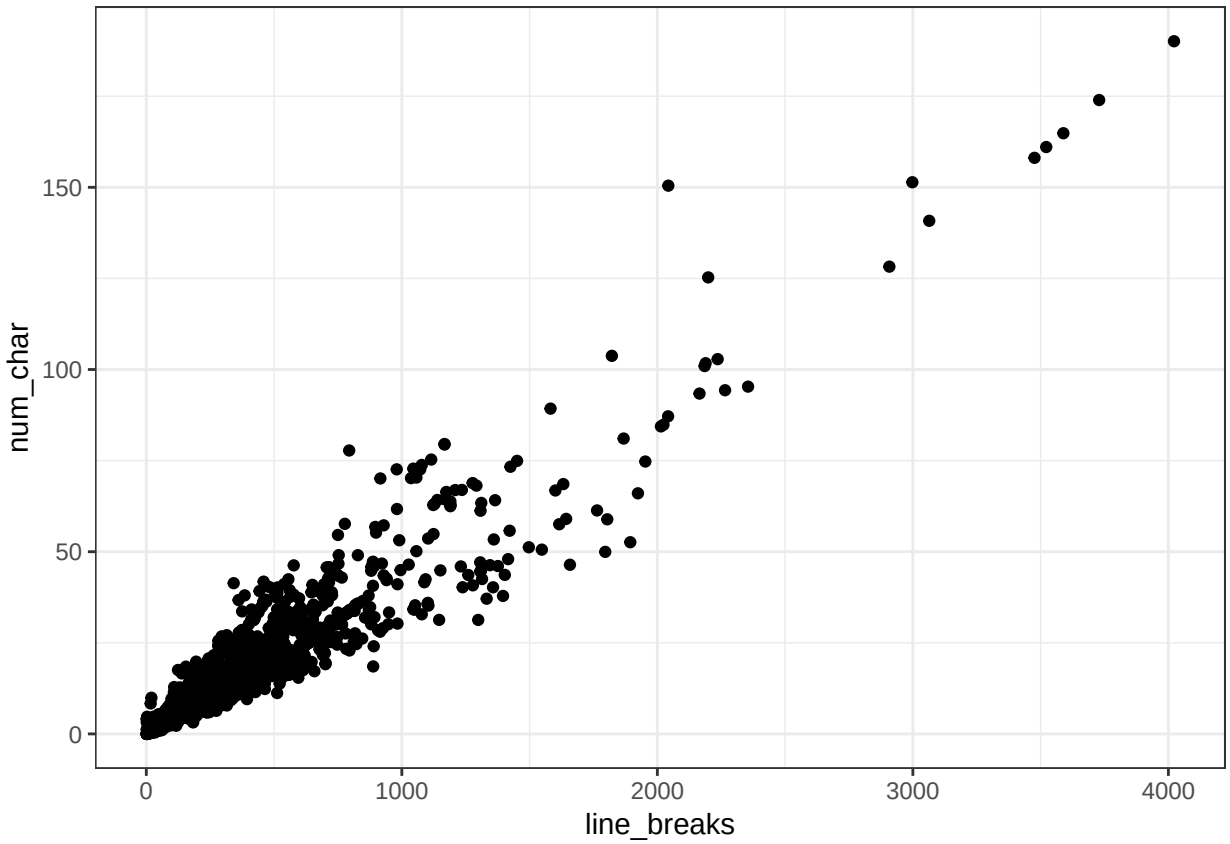
```
library(ggplot2); library(dplyr)
knitr::opts_chunk$set(warning=FALSE, message=FALSE)
email <- read.table("../data/email.txt", header=TRUE, sep="\t")
NCbirths <- read.csv("../data/NCbirths.csv", header=TRUE)
```

Appearance enhancements

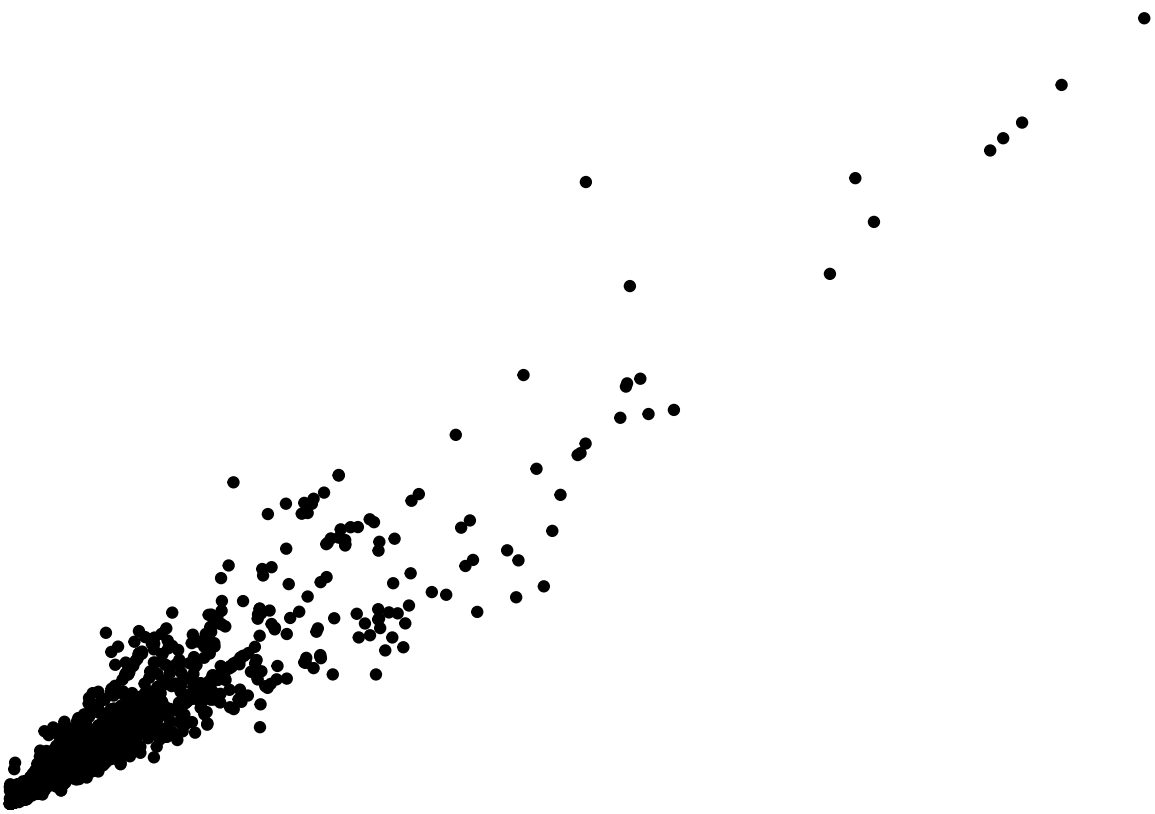
Themes

The standard theme has a grey background, white grid lines etc. Themes can be changed by adding `theme_X()` where `X` has several options

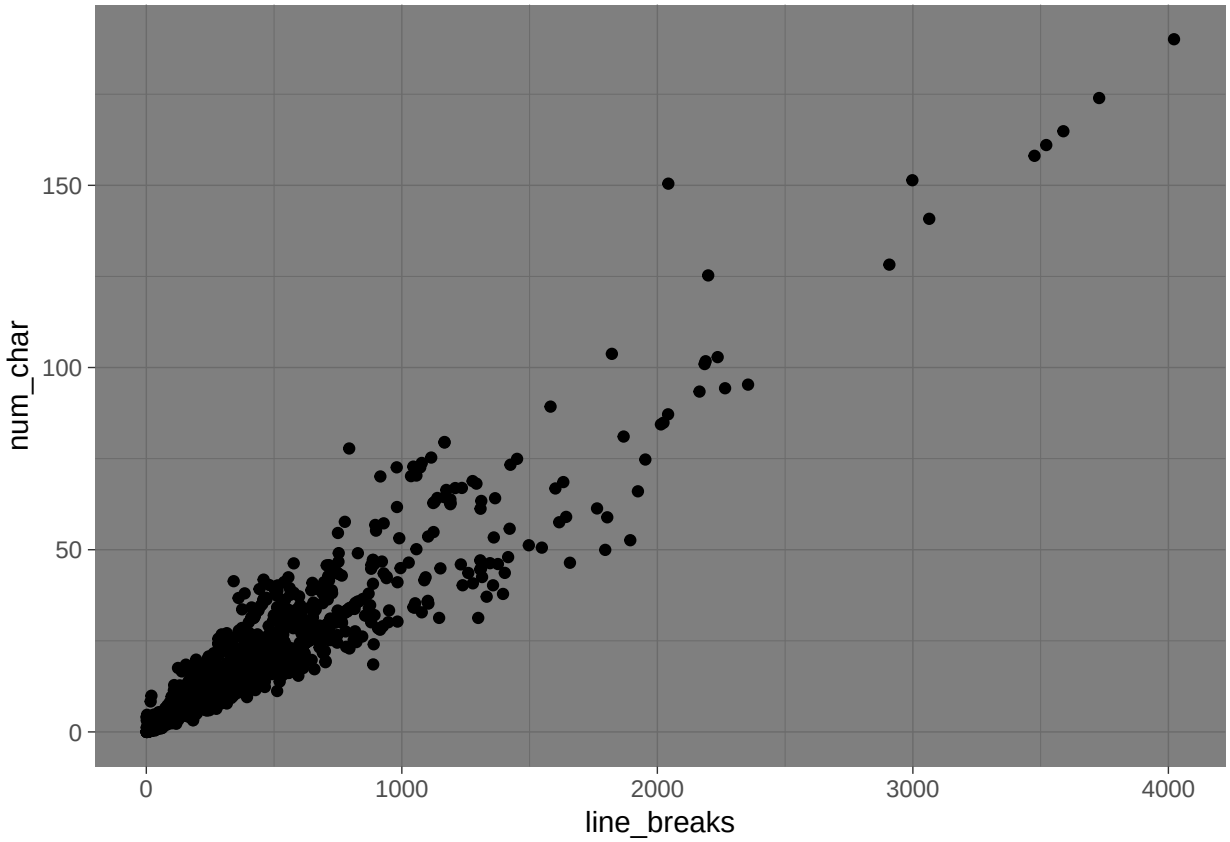
```
ggplot(email, aes(x=line_breaks, y=num_char)) + geom_point() + theme_bw()
```




```
ggplot(email, aes(x=line_breaks, y=num_char)) + geom_point() + theme_void()
```



```
ggplot(email, aes(x=line_breaks, y=num_char)) + geom_point() + theme_dark()
```

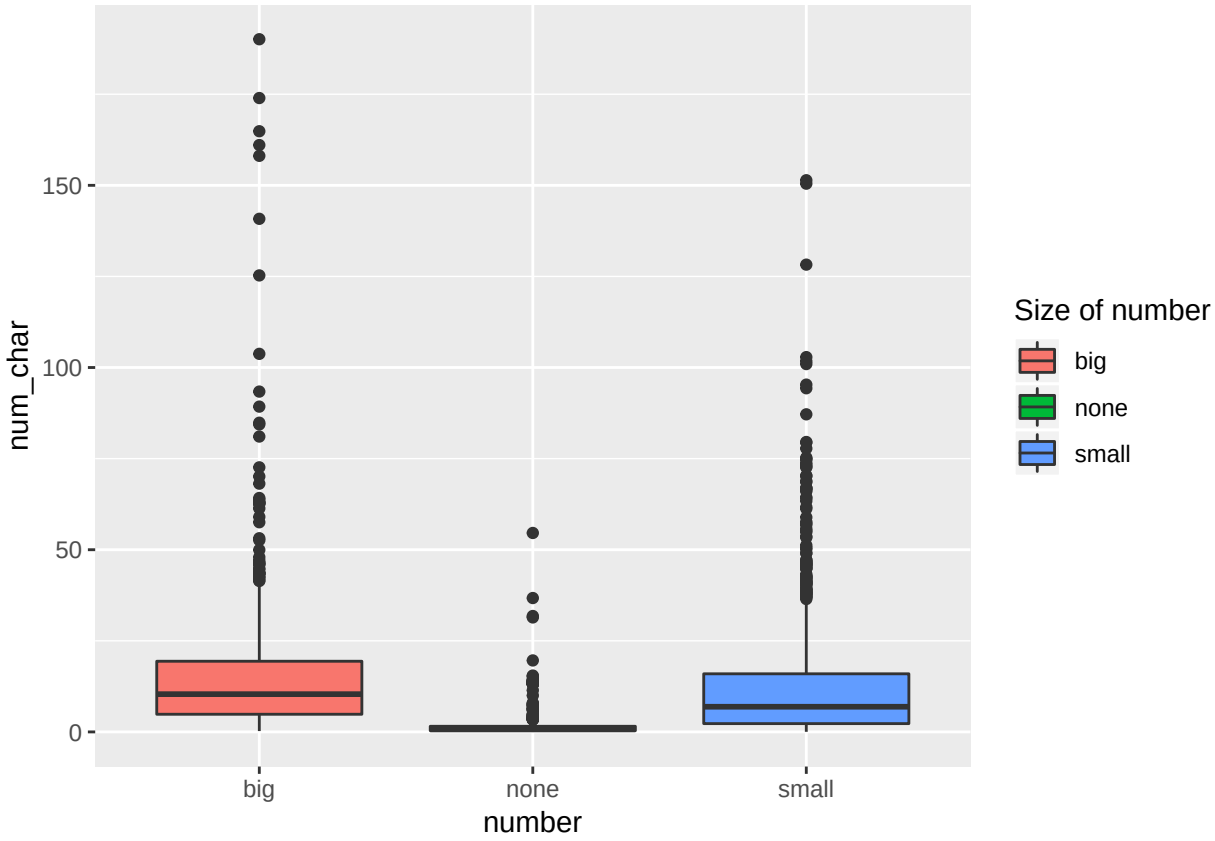


See more about themes here: http://www.sthda.com/english/wiki/ggplot2-themes-and-background-colors-the-3-elements

Change legend titles

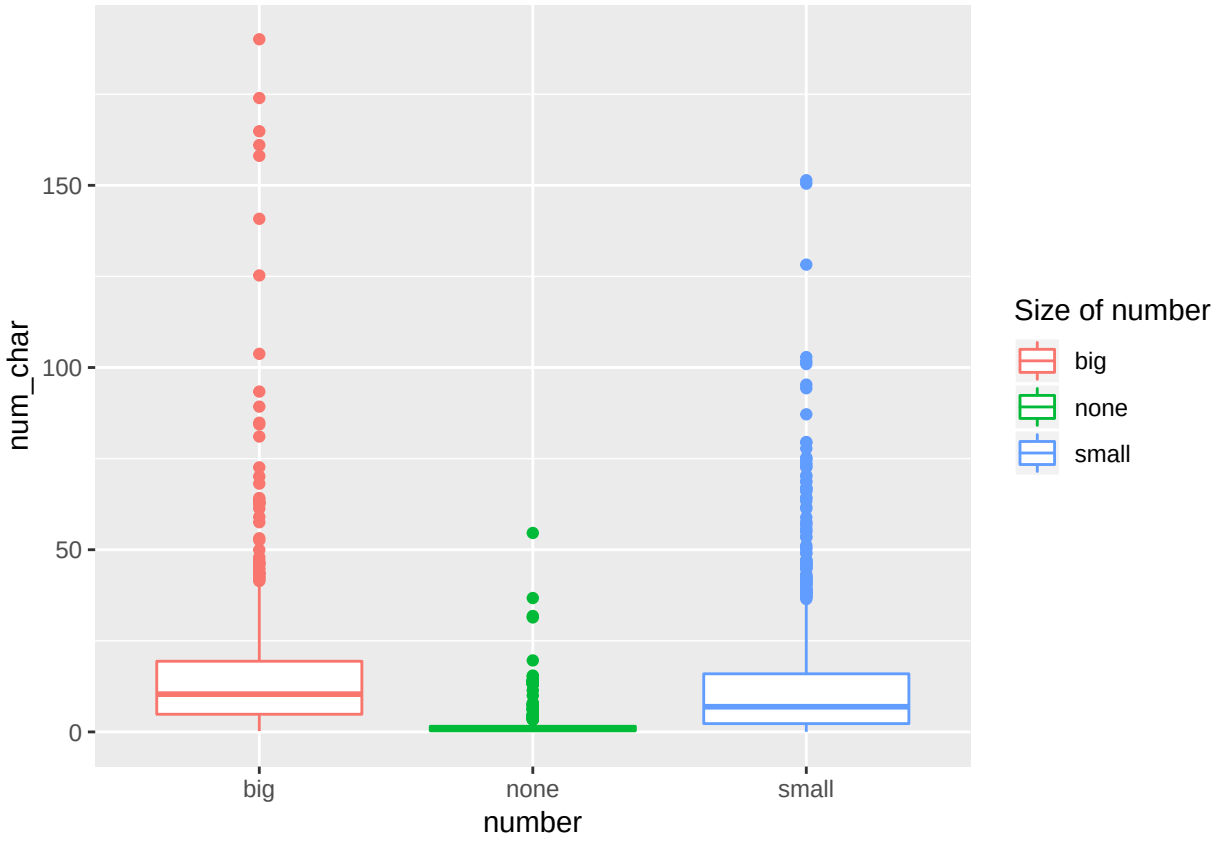
Add the `name=` argument to whatever layer you added that created the legend. Here I specified a `fill`, and it was a `discrete` variable. So I use the `scale_fill_discrete()` layer.

```
ggplot(email, aes(y=num_char, x=number, fill=number)) + geom_boxplot() +
  scale_fill_discrete(name="Size of number")
```



Here, I colored the points by a discrete variable, so the layer is `scale_color_discrete()`.

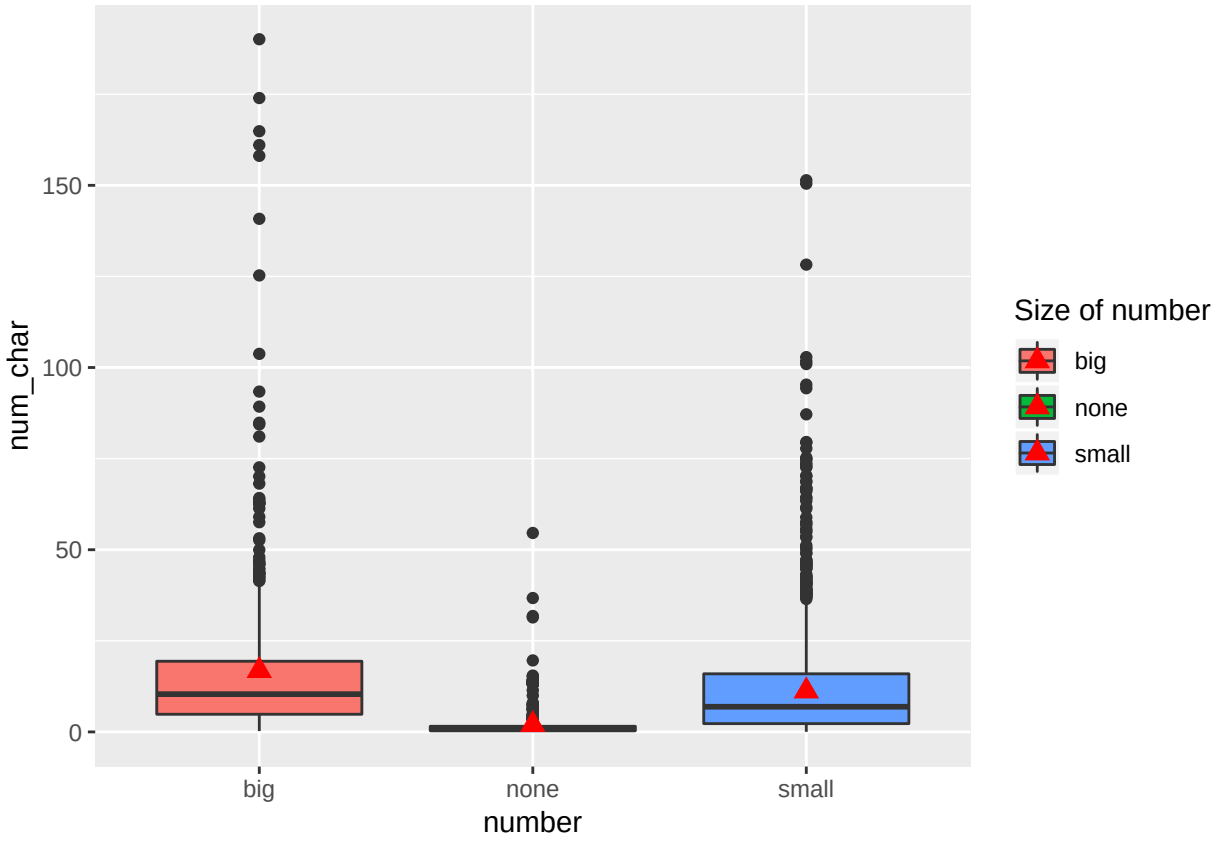
```
ggplot(email, aes(y=num_char, x=number, col=number)) + geom_boxplot() +
  scale_color_discrete(name="Size of number")
```



Add means to boxplots.

Boxplots are great. Even better with violin overlays. Know what makes them even better than butter? Adding a point for the mean. `stat_summary` is the layer you want to add.

```
ggplot(email, aes(y=num_char, x=number, fill=number)) + geom_boxplot() +
  scale_fill_discrete(name="Size of number") +
  stat_summary(fun.y="mean", geom="point", size=3, pch=17,color="red")
```



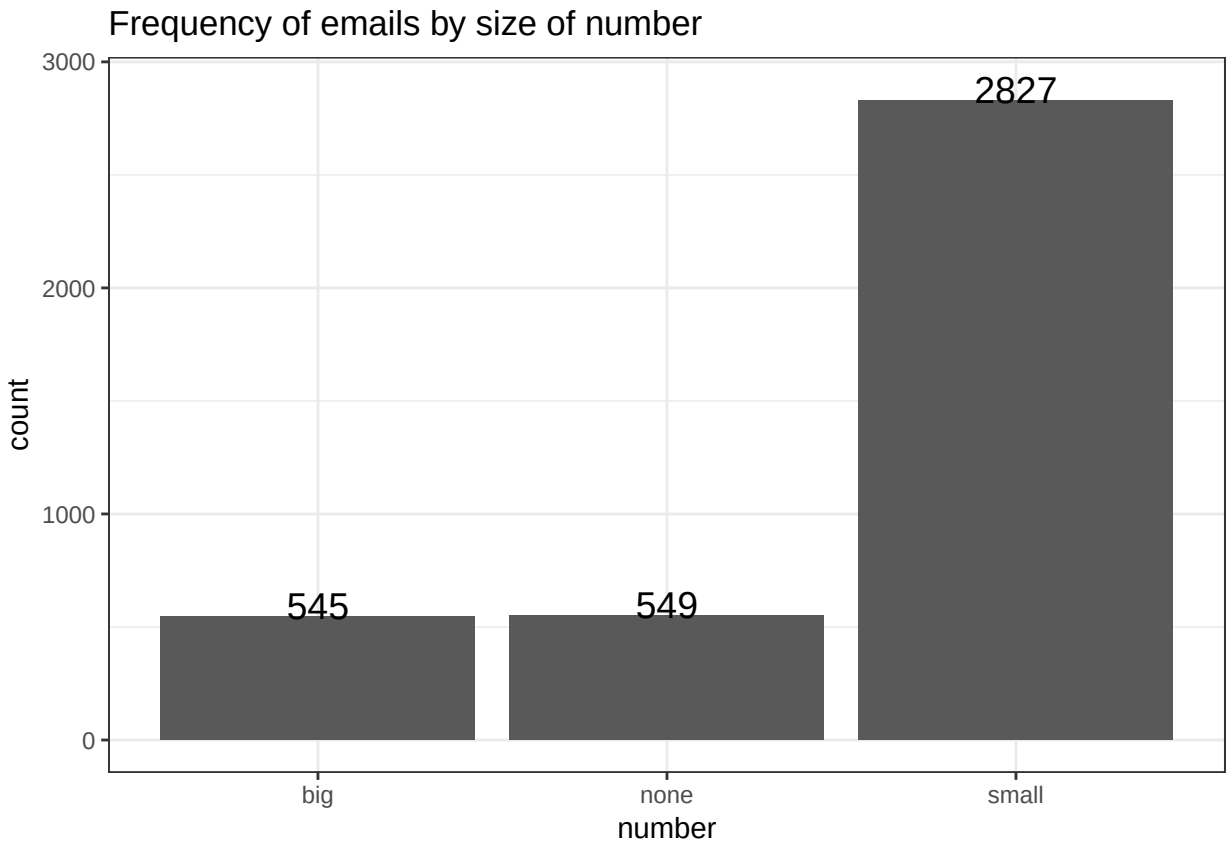
I suggest playing around with `size` and plotting character `pch` to get a feel for how these work. You can also look at `?pch` (and scroll down in the help file) to see the 25 default plotting characters.

Barchart enhancements

Numbers to the top of bars.

The biggest addition to a barchart is the numbers on top of the bars.

```
ggplot(email, aes(x=number)) + theme_bw() +
    geom_bar(aes(y = ..count..)) + ggtitle("Frequency of emails by size of number") +
    geom_text(aes(y=..count.. + 50, label=..count..), stat='count', size = 5)
```



- Play with the `+50` on the y axis to move the numbers up/down the bars.
- Play with the `size` modifier to find something readable.

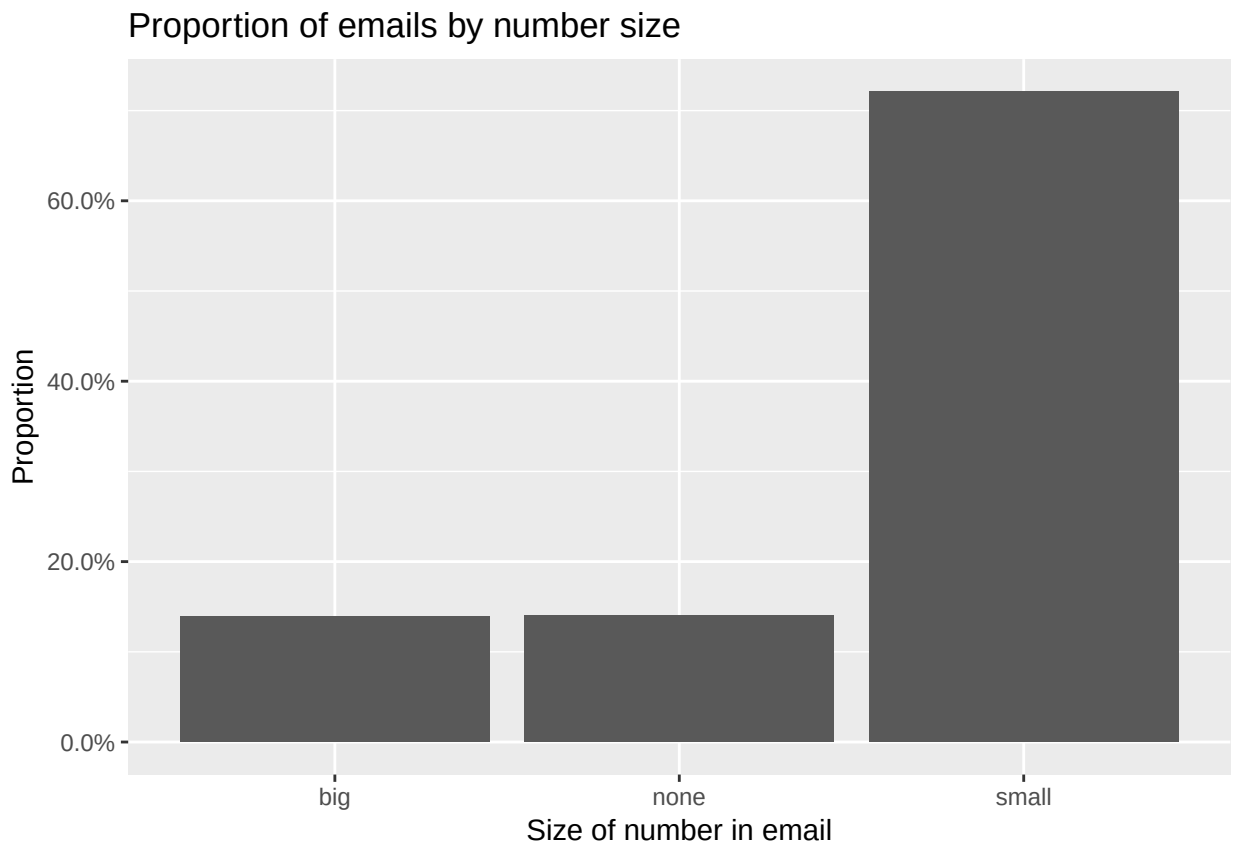
Plotting Proportions

Often you don't want to compare counts but percents. To accomplish this, we have to aggregate the data to calculate the proportions first, then plot the aggregated data using `geom_col` to create the columns. Here we also use `scale_y_continuous` to make the y axis labels display as percents. You can adjust the scale with just ggplot, but specifically showing percentages requires the `scales` package to be loaded.

```
library(scales)
cut.props <- table(email$number) %>% prop.table() %>% data.frame()
cut.props # what does this data look like?
```

```
##    Var1      Freq
## 1   big 0.1389952
## 2  none 0.1400153
## 3 small 0.7209895
```

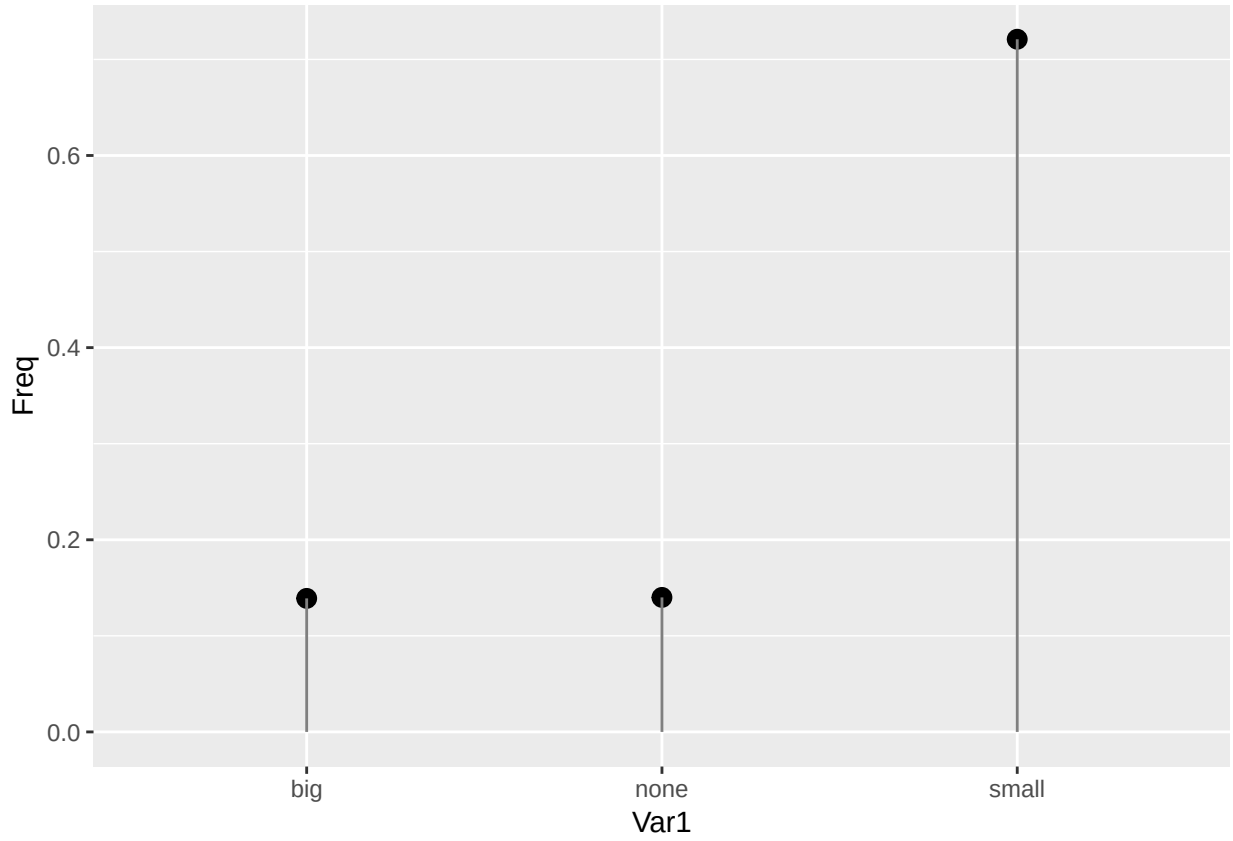
```
ggplot(cut.props, aes(x=Var1, y=Freq)) +
  geom_col() +
  scale_y_continuous(labels=percent) +
  ylab("Proportion") +
  xlab("Size of number in email") +
  ggtitle("Proportion of emails by number size")
```



Cleveland Dot Plots

Another way to visualize categorical data that takes up less ink than bars is a Cleveland dot plot. Here again we are plotting summary data instead of the raw data. This uses a new `geom_segment` that draws the lines from x=0 to the dot, and that it should be placed on the y-axis at the value of `Freq`.

```
ggplot(cut.props, aes(x=Var1, y=Freq)) +
  geom_point(size = 3) +
  geom_segment(aes(x=Var1, xend=Var1, y=0, yend=Freq), color='grey50')
```



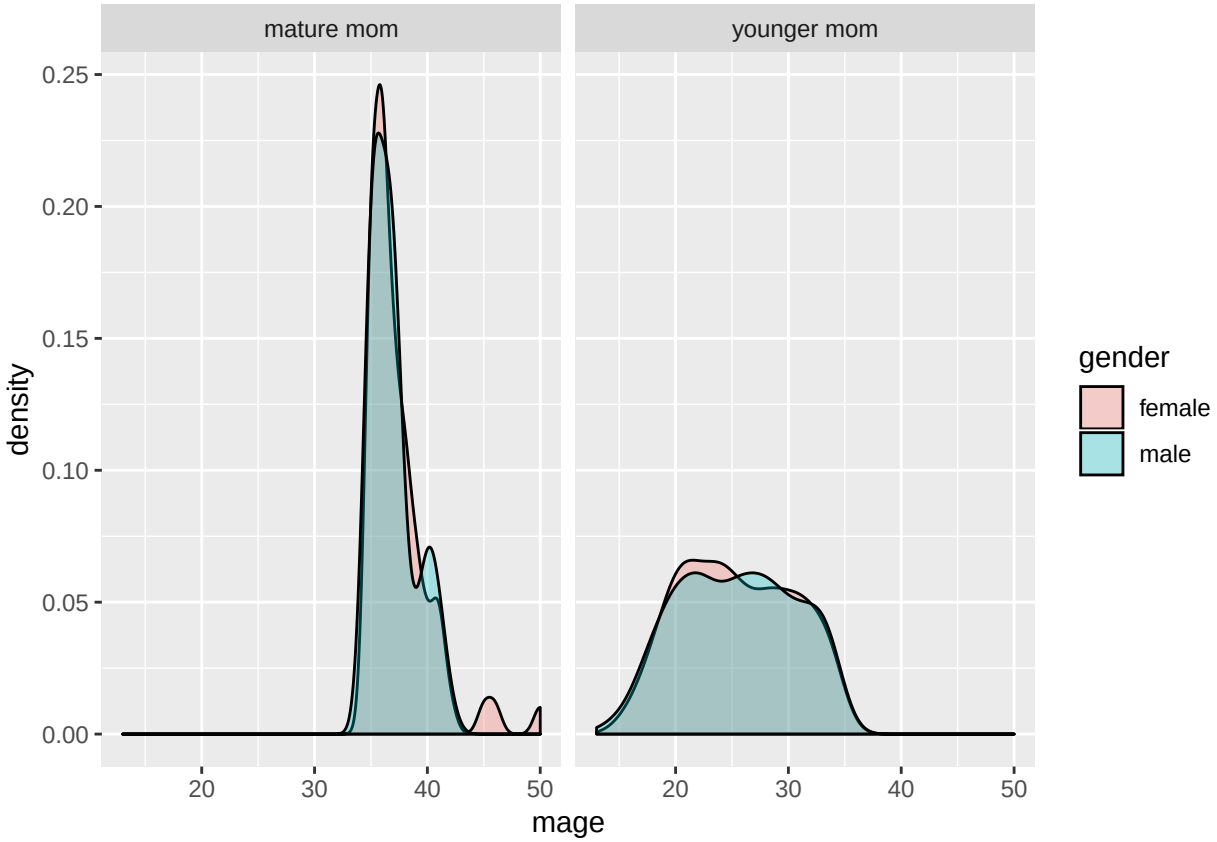
More examples can be found here: https://uc-r.github.io/cleveland-dot-plots

Faceting / paneling

ggplot introduces yet another term called `faceting`. The definition is *a particular aspect or feature of something*, or *one side of something many-sided, especially of a cut gem.* Basically instead of plotting the grouped graphics on the same plotting area, we let each group have it's own plot, or facet.

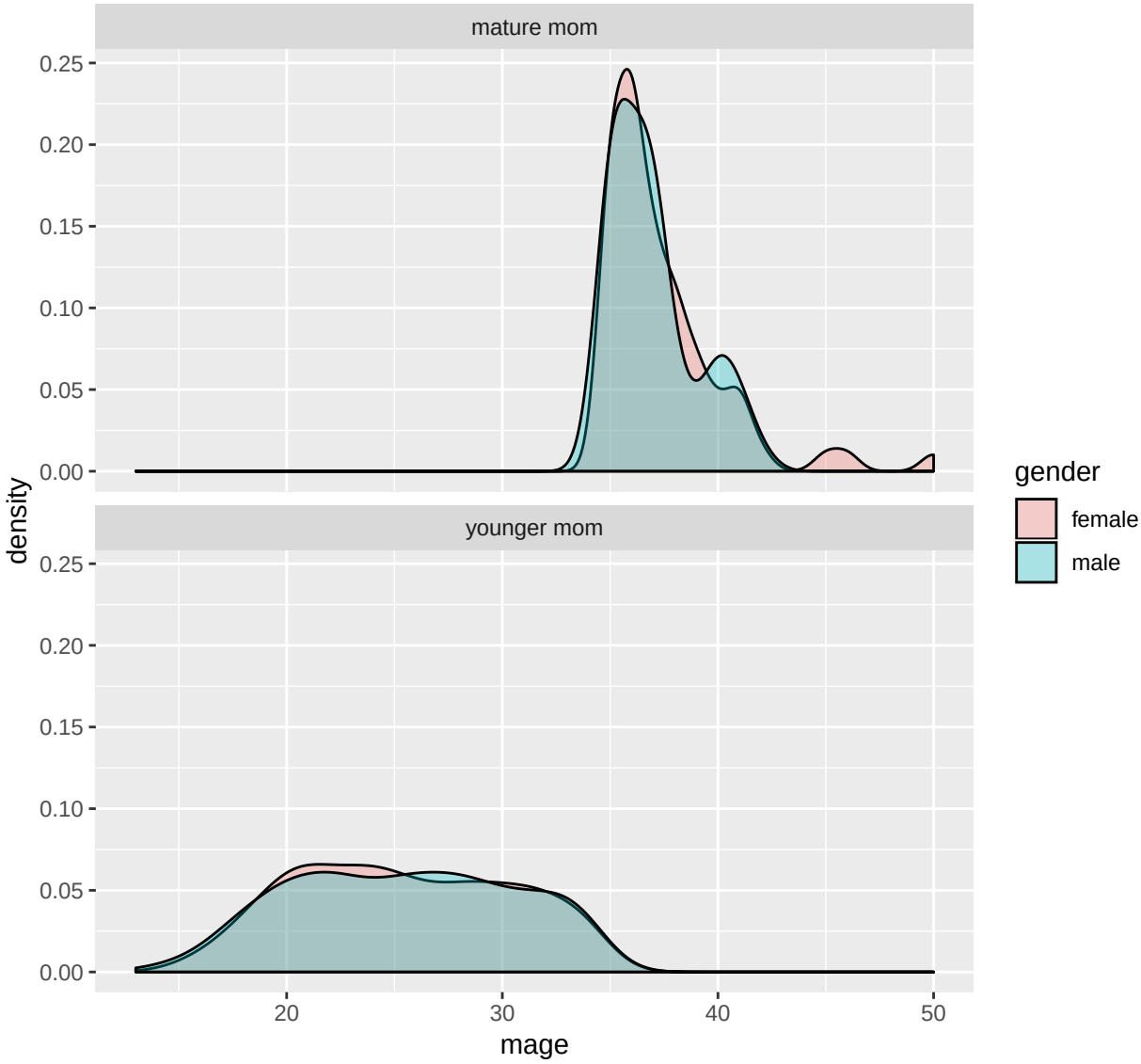
We add a `facet_wrap()` and specify that we want to panel on the color group. Note the twiddle in front of color. (Not here we're switching to the `NCbirths` data set.)

```
ggplot(NCbirths, aes(x=mage, fill=gender)) + geom_density(alpha=.3) + facet_wrap(~mature)
```



The grid placement can be semi-controlled by using the `ncol` argument in the `facet_wrap()` statement.

```
ggplot(NCbirths, aes(x=mage, fill=gender)) + geom_density(alpha=.3) + facet_wrap(~mature, ncol=1)
```



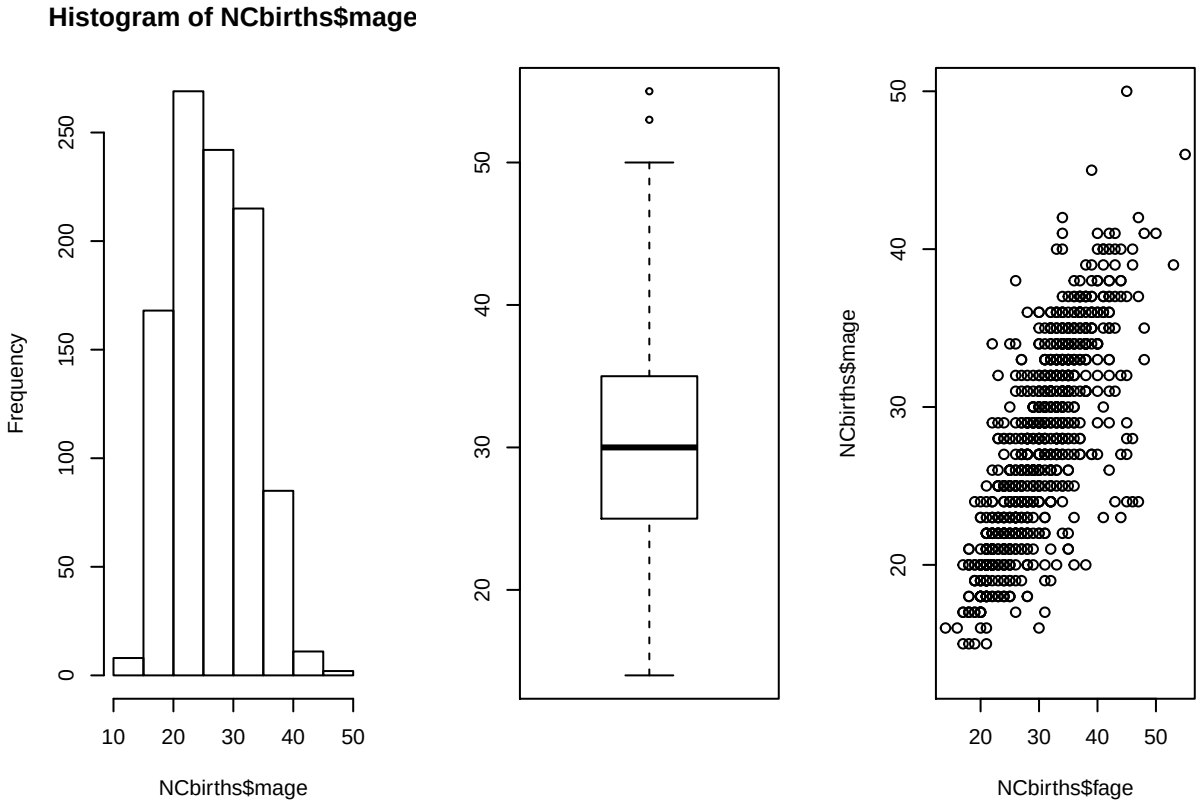
It is important to compare distributions across groups on the same scale, and our eyes can compare items vertically better than horizontally.

Multiple plots per window

Base

I use `par(mfrow=c(r,c))` for base graphics, where `r` is the number of rows and `c` the number of columns.

```
par(mfrow=c(1,3))
hist(NCbirths$mage)
boxplot(NCbirths$fage)
plot(NCbirths$mage ~ NCbirths$fage)
```

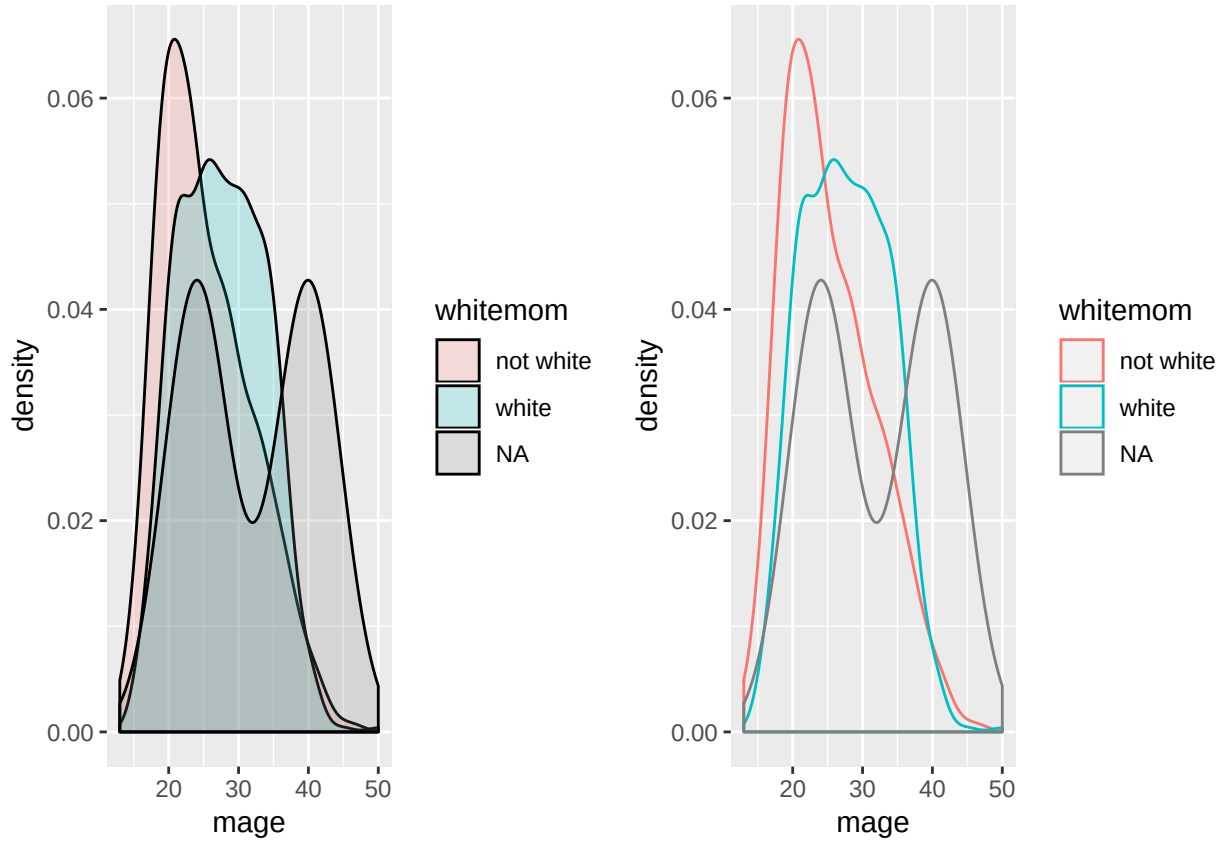


Other resources including learning about `layouts`. Multipanel plotting with base graphics http://seananderson.ca/courses/11-multipanel/multipanel.pdf

ggplot

Use the `grid.arrange` function in the `gridExtra` package. You assign the output of a ggplot object to an object (here it's `plot1` and `plot2`). Then you use `grid.arrange()` to arrange them either side by side or top and bottom.

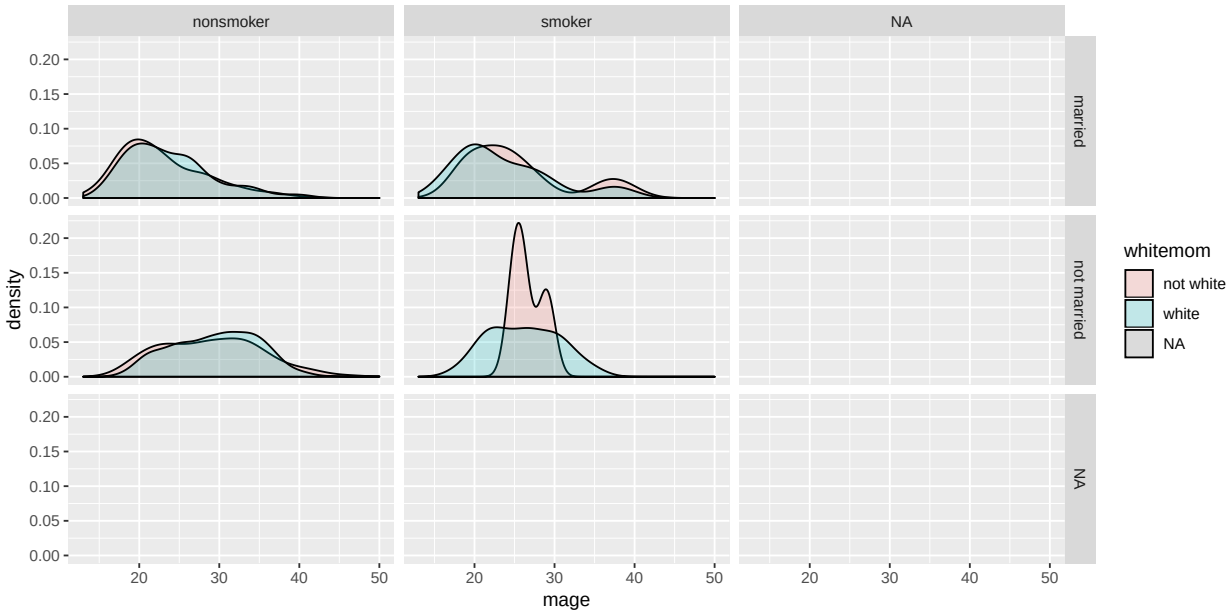
```
library(gridExtra)
plot1 <- ggplot(NCbirths, aes(x=mage, fill=whitemom)) + geom_density(alpha=.2)
plot2 <- ggplot(NCbirths, aes(x=mage, col=whitemom)) + geom_density()
grid.arrange(plot1,plot2, ncol=2)
```



Paneling on two variables

Who says we're stuck with only faceting on one variable? A variant on `facet_wrap` is `facet_grid`. Here we can specify multiple variables to panel on.

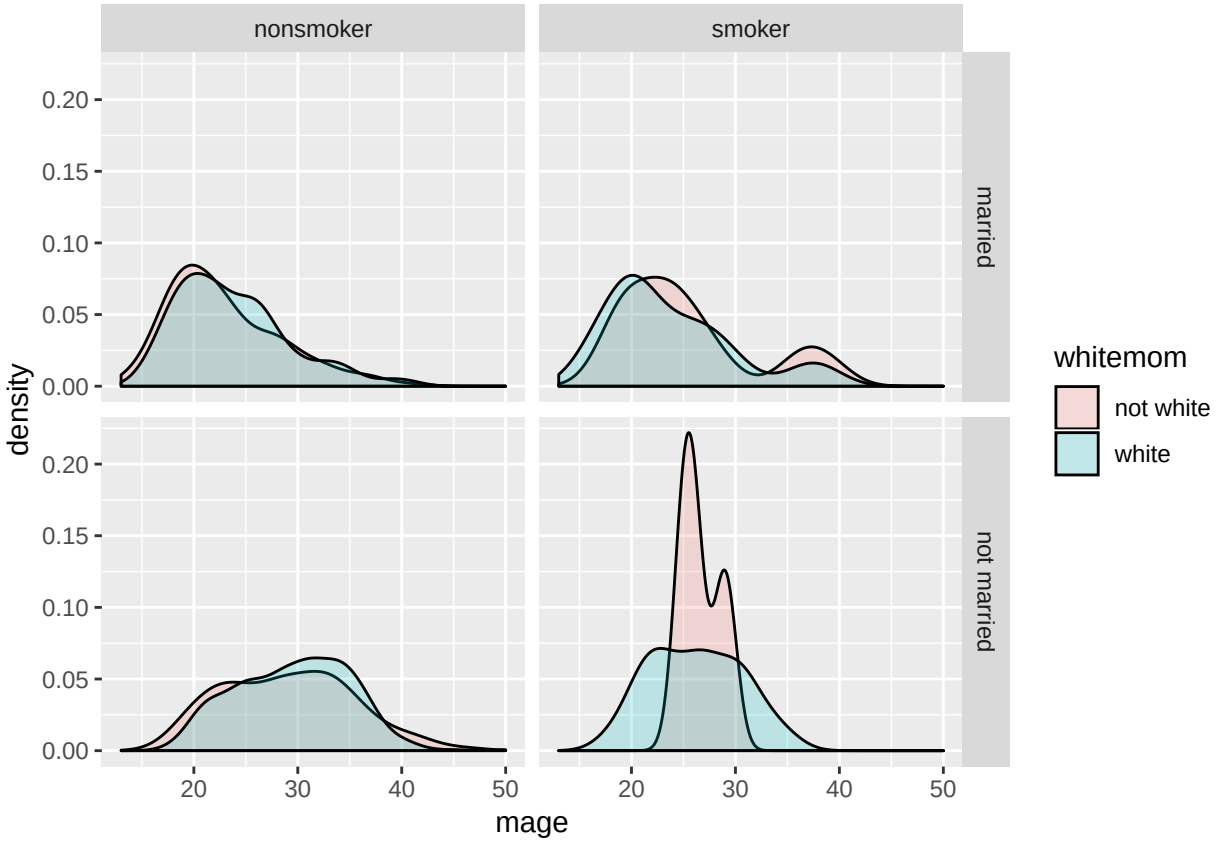
```
ggplot(NCbirths, aes(x=mage, fill=whitemom)) + geom_density(alpha=.2) + facet_grid(marital~habit)
```



Removing NA categories.

This happens when there is no data is present for one or more levels. One way to get around this is to use `select` to only choose variables that we are going to use in the plot directly, delete all rows with any missing values using `na.omit()`, then pipe `ggplot` directly in after deleting all rows with missing data.

```
NCbirths %>% select(mage, whitemom, marital, habit) %>% na.omit() %>%
  ggplot(aes(x=mage, fill=whitemom)) +
  geom_density(alpha=.2) + facet_grid(marital~habit)
```

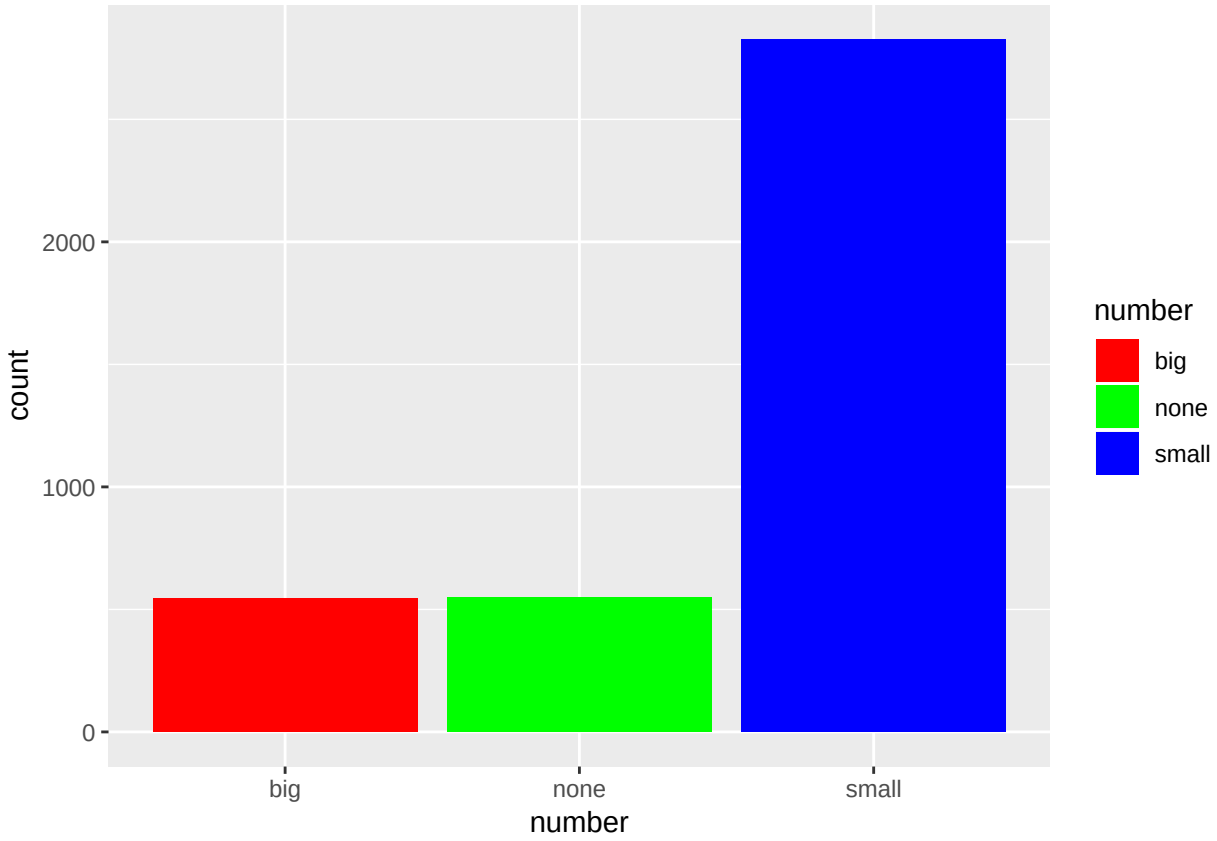


Warning: `na.omit()` can be dangerous to use. It will drop EVERY row with ANY value missing. We are using it safely here because we are only selecting the variables that we will use in the plot.

Changing colors

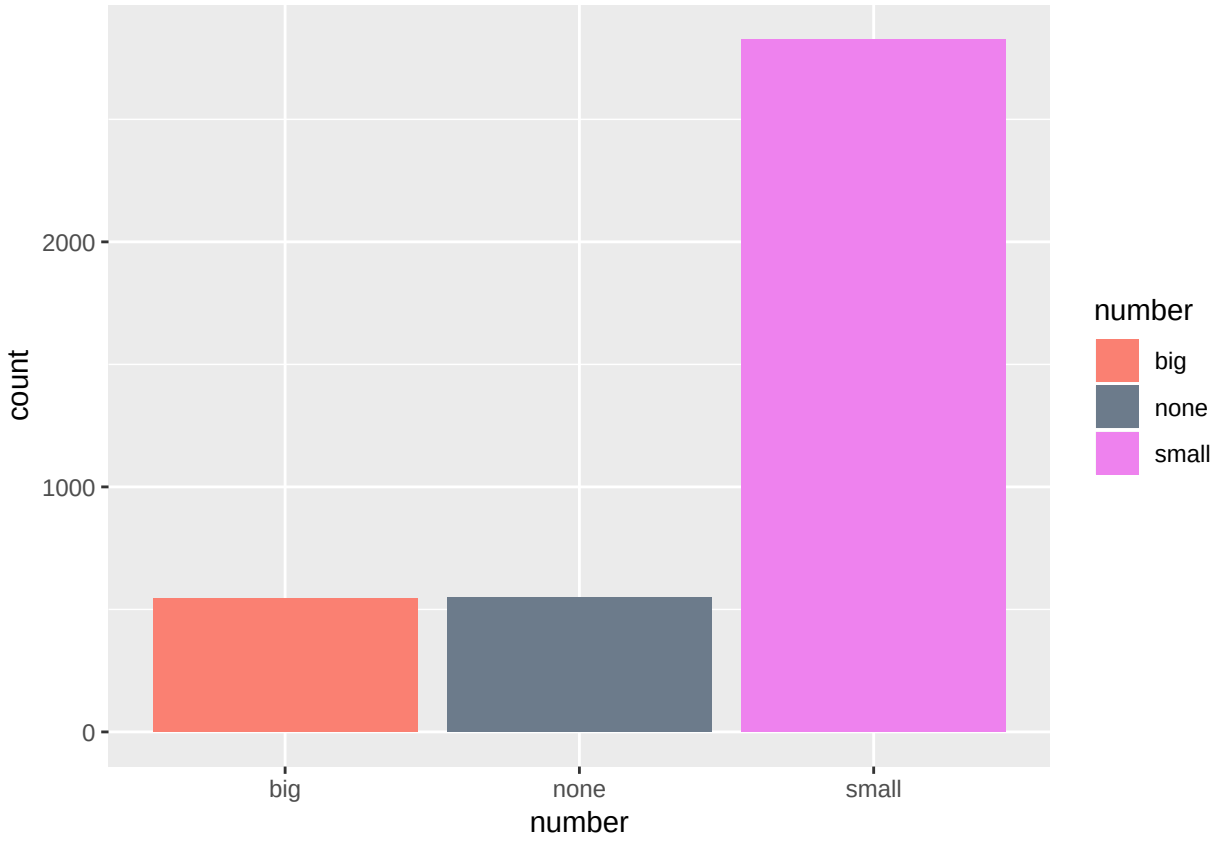
Manual

```
ggplot(email, aes(x=number, fill=number)) + geom_bar() +
    scale_fill_manual(values=c("red", "green", "blue"))
```



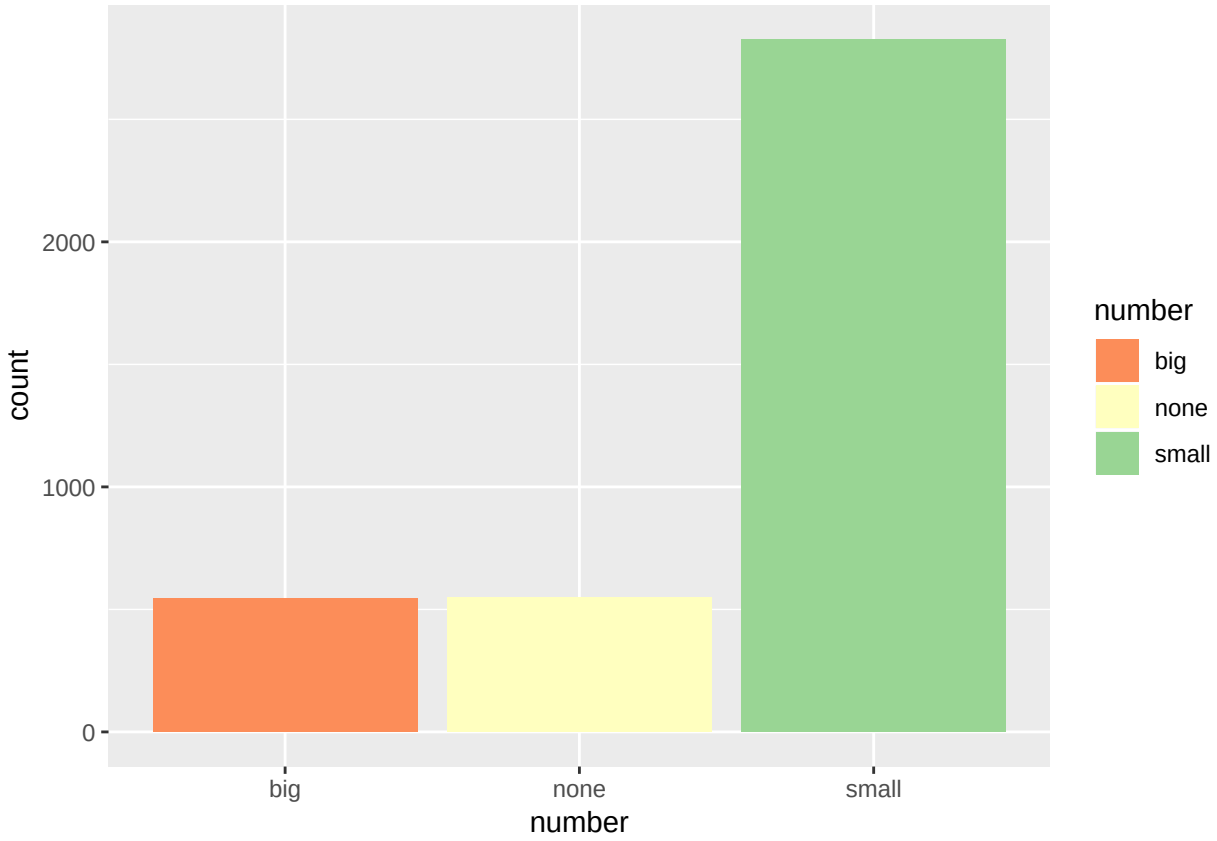
There's some spectacularly named colors as well,

```
ggplot(email, aes(x=number, fill=number)) + geom_bar() +
    scale_fill_manual(values=c("salmon", "slategray4", "violet"))
```



Using a color palette (recommended)

```
library(RColorBrewer)
ggplot(email, aes(x=number, fill=number)) + geom_bar() +
    scale_fill_brewer(palette="Spectral")
```



More guidance on colors can be found here: https://www.nceas.ucsb.edu/~frazier/RSpatialGuides/colorPaletteCheatsheet.pdf