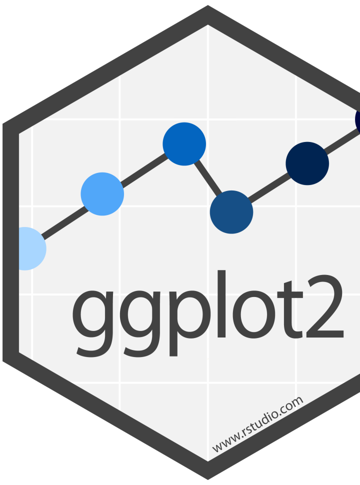
Data Visuali

JSC 370: Data Sc

January 29, 20

Background



This lecture provides an introduction to `ggplo`
vastly better graphics options than R's defaul

Background

`ggplot2` is part of the Tidyverse. The tidyverse
of R packages designed for data science. All p
design philosophy, grammar, and data struct
www.tidyverse.org/)

Core tidyverse includes `dplyr`, which provides
manipulation.

It also includes `stringr` which we will use in a
we saw in lab that helps with dates and times

```
library(tidyverse)
library(nycflights13)
```

```r
library(kableExtra)
```

Layers, dplyr and pipes

- We should take a step back and discuss

- `ggplot2` behaves very similarly to `dplyr`. T

- The first argument in `dplyr` is always a da
data.table).

- Subsequent arguments can also be thou
actions to be taken on the data (verbs).

- The output is always a new data frame.

- Layers are connected by a pipe, which ur

- The new pipe is |>, which works similarly
likely only noticeable by expert users.

The Pipe %>% and now |>

- The pipe passes the object on its left han
of the function on the right hand side.

- We can kind of think of it as saying 'then

Flights data

To illustrate many of today's examples we will the `nycflights13` library. They are all flights tha (JFK, LGA, EWR) in 2013.

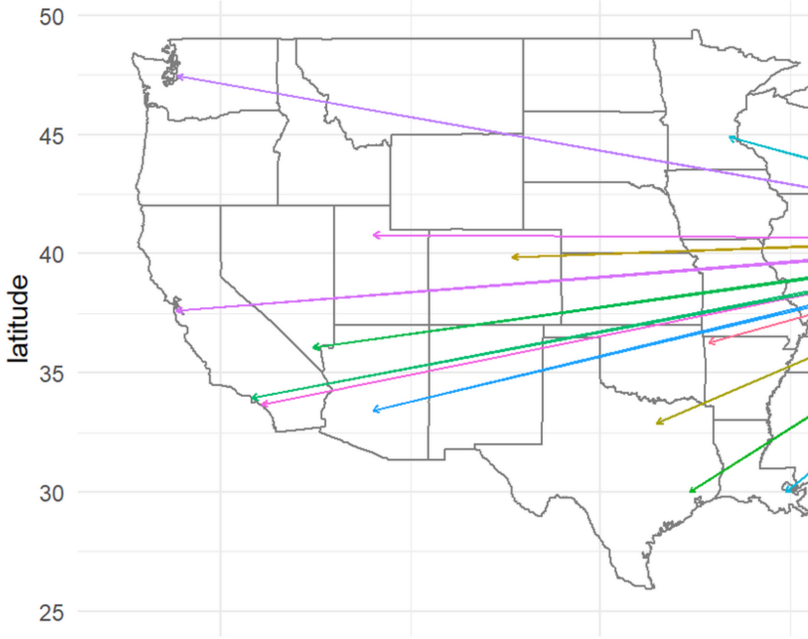
```
names(flights)
```

```
##  [1] "year"           "month"          "day"
##  [5] "sched_dep_time" "dep_delay"      "arr_time"
##  [9] "arr_delay"      "carrier"        "flight"
## [13] "origin"         "dest"           "air_time"
## [17] "hour"           "minute"         "time_hour"
```

```
dim(flights)
```

```
## [1] 336776     19
```

Flights data



The pipe %>%

- An example using pipes: subset the fligh
mean arrival delay times by year, month, c

- We need to start with the data, `filter` to
`group_by` to prepare the groups that we wa
`summarize` to take the mean (or whatever fu
variable we are interested in.

```r
nycflights13::flights %>%
  filter(dest == "LAX") %>%
  group_by(year, month, day) %>%
  summarize(
    arr_delay = mean(arr_delay, na.rm = TRUE)
  )
```

The new pipe |>

Let's do the same thing with the new pipe:

```
nycflights13::flights |>
  filter(dest == "LAX") |>
  group_by(year, month, day) |>
  summarize(
    arr_delay = mean(arr_delay, na.rm = TRUE)
  )
```

A few coding style tips

- Variable names (those created by `<-` anc
  and `summarize()`) should use only lowercas

- Use _ to separate words within a name

- `%>%` or `|>` should always have a space bet
  followed by a new line

- After the first step, each line should be in

- Note on the previous slides that had long
  function may not all fit on one line, put ea
  and indent.

```
short_flights <- flights |>
  filter(air_time < 60)
```

Flights data in more detail

The `nycflights13` library also provides hourly a
three NYC airports (the origin). Let's join the fl
data so we can look at more interesting relati

```
names(weather)
```

```
##  [1] "origin"     "year"       "month"      "day"
##  [6] "temp"       "dewp"       "humid"      "wind_dir"
## [11] "wind_gust"  "precip"     "pressure"   "visib"
```

```
dim(weather)
```

```
## [1] 26115    15
```

Flights data in more detail

Looks like we can join these datasets on year,
(which is the origin airport). We can examine
flight delays and weather at the origin airport

A `left_join` will keep all of the observations in
with the observations in y (weather at origin).
flights originating at the 3 NYC airports on a g
we want to keep the resolution of the x datas

```
flights_weather <-
  left_join(
    flights, weather, by = c("year", "month", "day", "hou
    )
```

Flights data in more detail

```
head(flights_weather)
```

```
## # A tibble: 6 × 29
##    year month   day dep_time sched_dep_time dep_delay a
##   <int> <int> <int>    <int>          <int>     <dbl>
## 1  2013     1     1      517            515         2
## 2  2013     1     1      533            529         4
## 3  2013     1     1      542            540         2
## 4  2013     1     1      544            545        -1
## 5  2013     1     1      554            600        -6
## 6  2013     1     1      554            558        -4
## # ℹ 21 more variables: arr_delay <dbl>, carrier <chr>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <
## #   hour <dbl>, minute <dbl>, time_hour.x <dttm>, temp
## #   humid <dbl>, wind_dir <dbl>, wind_speed <dbl>, wind
## #   precip <dbl>, pressure <dbl>, visib <dbl>, time_hou
```

Daily flights data

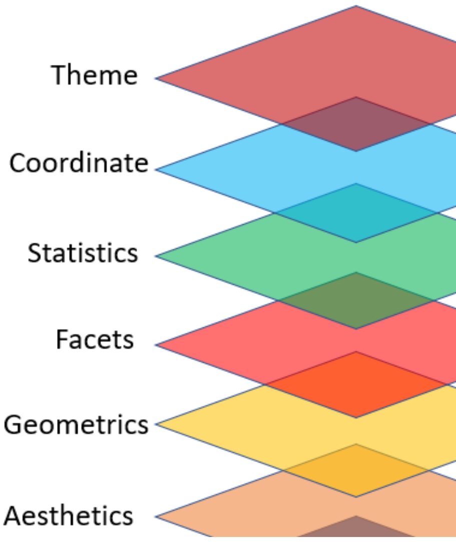
Let's make a more manageable sized dataset
month, day, and origin airport

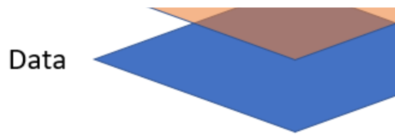
```
flights_weather_day <-
  flights_weather |>
  group_by(year, month, day, origin) |>
  summarize_at(
    vars(dep_delay, arr_delay, temp, dewp, humid, wind_di
             wind_speed, wind_gust, precip, pressure, vis

  )
```

Visualizations

`ggplot2` is designed on the principle of addin





Layers in ggplot2

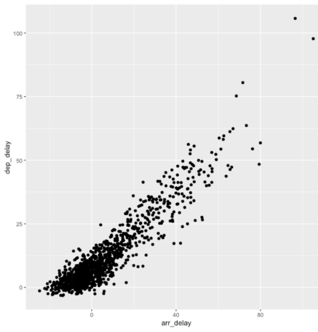
- With `ggplot2` a plot is initiated with the fu
- The first argument of `ggplot()` is the data
- We add aesthetics is always paired with
- The `aes()` mapping takes the x and y axe
- Layers are added to `ggplot()` with `+`
- Layers include `geom` functions such as po

```
ggplot(data = data, mapping = aes(mappings)) +
  geom_function()
```

Basic scatterplot

The first argument of `ggplot()` is the dataset t
With the `+` you add one or more layers.

```
ggplot(data = flights_weather_day, mapping = aes(x = arr_
  geom_point()
```

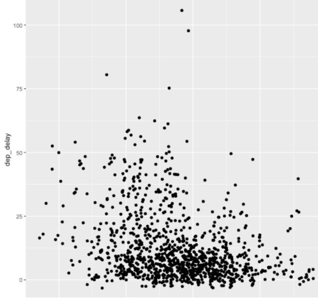


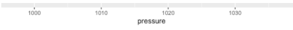
As expected, we see that if a flight has a late c

Another basic scatterplot

We can drop `data =` and `mapping =`. Now let's s
between departure delays and pressure (low
precipitation, high pressure means better we

```
ggplot(flights_weather_day, aes(x = pressure, y = dep_del
  geom_point()
```





Adding to a basic scatterplo

- `geom_point()` adds a layer of points to you

- `ggplot2` comes with many geom functio type of layer to a plot.

- Each `geom` function in `ggplot2` takes a ma

- This defines how variables in your datase properties.

- The mapping argument is always paired arguments of `aes()` specify which variable

- One common problem when creating `g

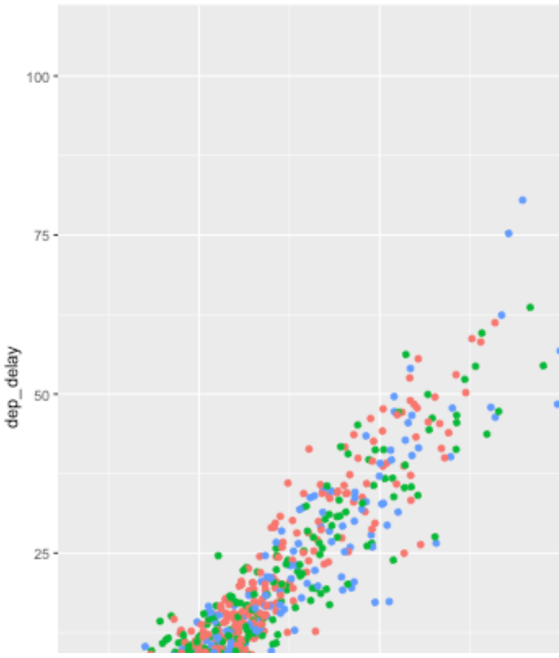
in the wrong place: it has to come at the e

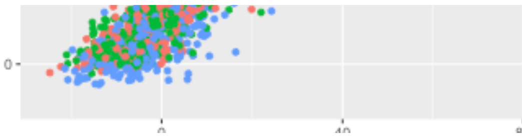
Coloring by a variable - usin

You can convey information about your data your plot to the variables in your dataset. For colors of your points to the class variable to re chooses colors, and adds a legend, automatic

```
ggplot(flights_weather_day, aes(x = arr_delay, y = dep_de
  geom_point()
```

Coloring by a variable - usin



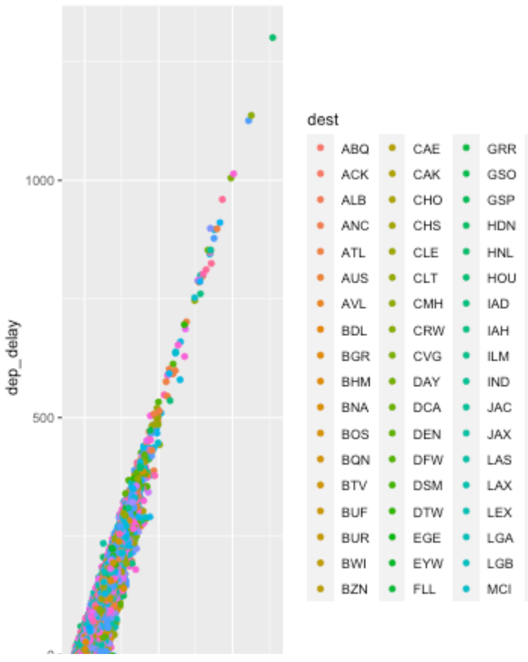


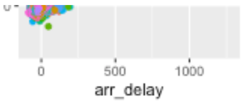
Coloring by a variable - usin

Note when there are a lot of classes or groups
distinguished well

```
ggplot(flights_weather, aes(x = arr_delay, y = dep_delay,
  geom_point()
```

Coloring by a variable - usin



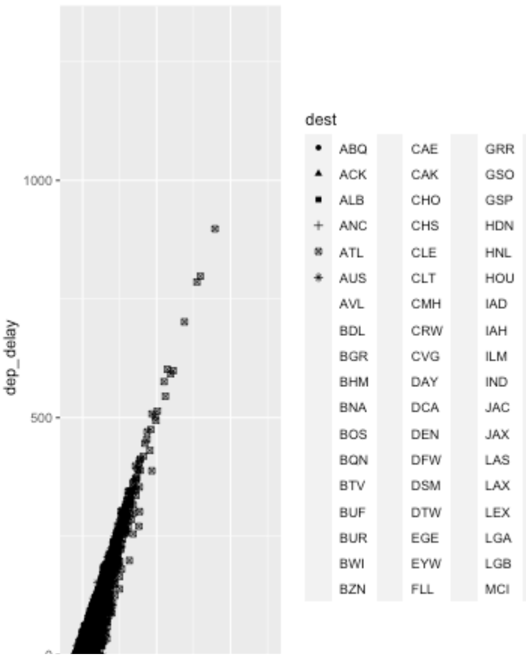


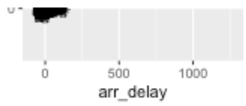
Determining point type usir

By default ggplot uses up to 6 shapes. If there is not plotted!! (At least it warns you.)

```
ggplot(flights_weather, aes(x = arr_delay, y = dep_delay,
  geom_point()
```

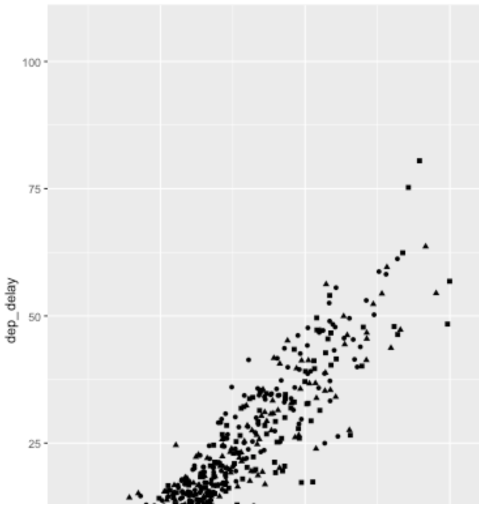
Determining point type usir

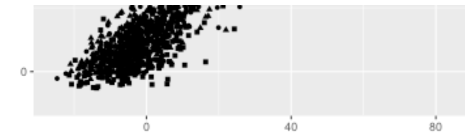




Determining point type usir

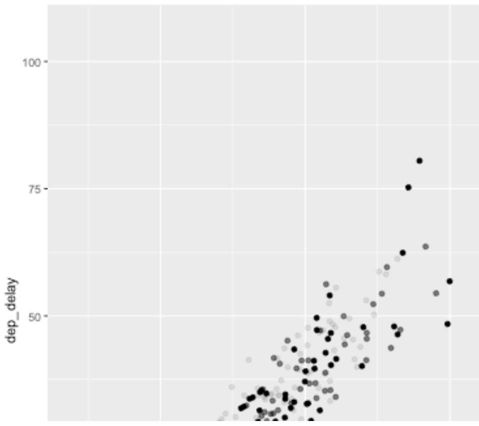
```
ggplot(flights_weather_day, aes(x = arr_delay, y = dep_de
  geom_point()
```

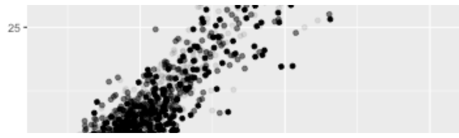




Controlling point transparen "alpha" aesthetic

```
ggplot(flights_weather_day, aes(x = arr_delay, y = dep_de
  geom_point()
```

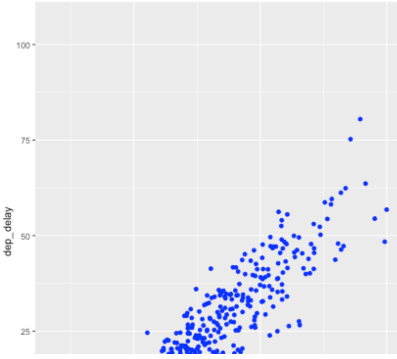


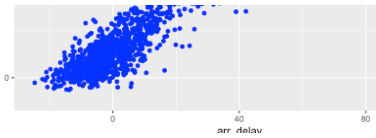


Manual control of aesthetics

To control aesthetics manually, we do it in `geo`
name outside `aes()`

```
ggplot(flights_weather_day, aes(x = arr_delay, y = dep_de
  geom_point(color = "blue")
```





Summary of aesthetics

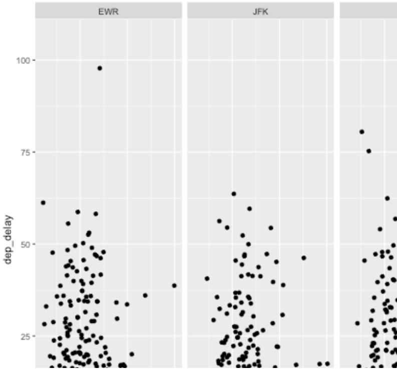
The various aesthetics...

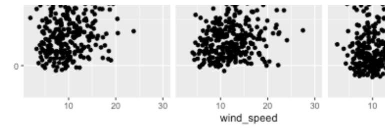
Code	Description
x	position on x-ax
x	position on y-ax
shape	shape
color	color of elemen
fill	color inside ele
size	size
alpha	transparency
linetype	type of line

Facets 1

Facets are particularly useful for categorical v

```
ggplot(flights_weather_day, aes(x = wind_speed, y = dep_d
  geom_point() +
  facet_wrap(~origin, nrow = 1)
```

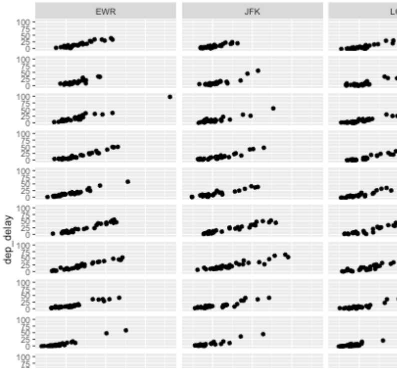


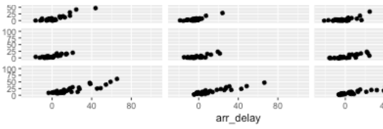


Facets 2

Or you can facet on two variables...

```
ggplot(flights_weather_day, aes(x = arr_delay, y = dep_de
  geom_point() +
  facet_grid(month ~ origin)
```



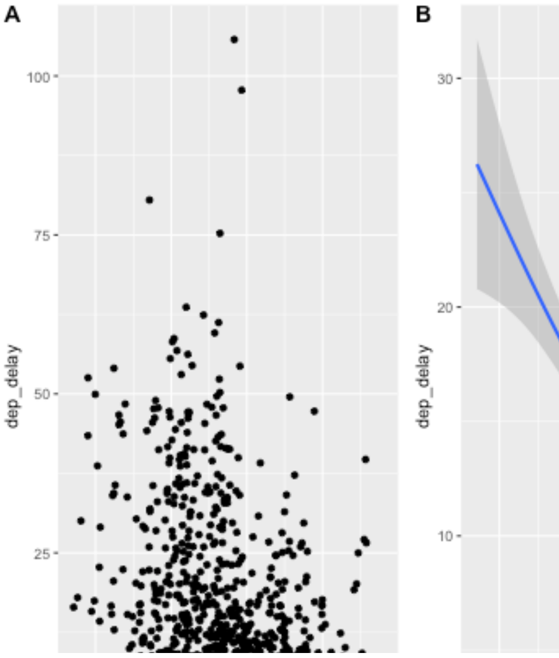


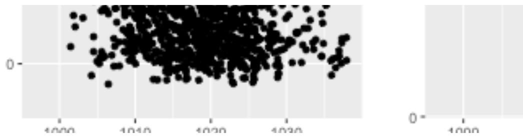
Geometric Objects 1

Geometric objects are used to control the typ
plotting a smoothed line fitted to the data (ar
side plots).

```
library(cowplot)
scatterplot <- ggplot(flights_weather_day, aes(x = pressu
  geom_point()
lineplot <- ggplot(flights_weather_day, aes(x = pressure,
  geom_smooth()
plot_grid(scatterplot, lineplot, labels = "AUTO")
```

Geometric Objects 1

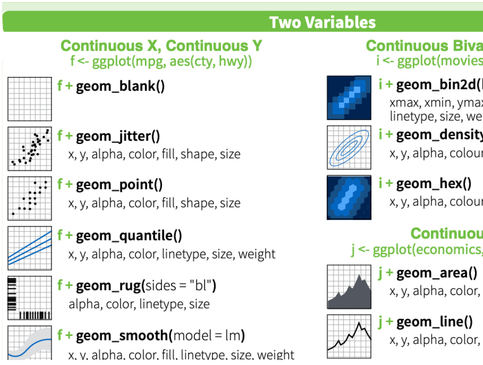




Geoms - Reference

`ggplot2` provides over 40 geoms, and extensic
more (see https://ggplot2.tidyverse.org/referer

The best way to get a comprehensive overvie
which you can find at https://posit.co/resource



Two Variables

Continuous X, Continuous Y
f <- ggplot(mpg, aes(cty, hwy))

f + **geom_blank()**

f + **geom_jitter()**
x, y, alpha, color, fill, shape, size

f + **geom_point()**
x, y, alpha, color, fill, shape, size

f + **geom_quantile()**
x, y, alpha, color, linetype, size, weight

f + **geom_rug(**sides = "bl"**)**
alpha, color, linetype, size

f + **geom_smooth(**model = lm**)**
x, y, alpha, color, fill, linetype, size, weight

Continuous Biva
i <- ggplot(movies

i + **geom_bin2d(**
xmax, xmin, ymax
linetype, size, we

i + **geom_density**
x, y, alpha, colour

i + **geom_hex()**
x, y, alpha, colour

Continuou
j <- ggplot(economics,

j + **geom_area()**
x, y, alpha, color,

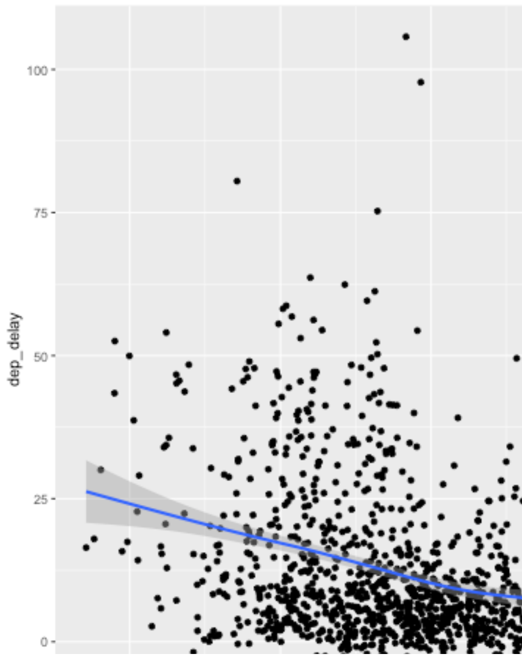
j + **geom_line()**
x, y, alpha, color,

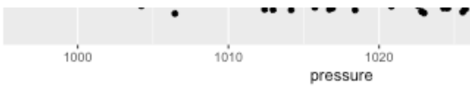
Multiple Geoms 1

To display multiple geoms in the same plot, a
`ggplot()`:

```
ggplot(flights_weather_day, aes(x = pressure, y = dep_del
  geom_point() +
  geom_smooth()
```

Multiple Geoms 1



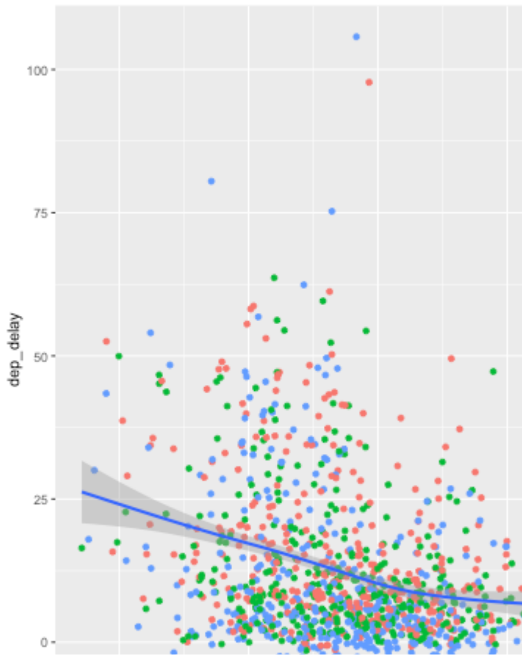


Multiple Geoms 2

If you place mappings in a `geom` function, `ggpl`
extend or overwrite the global mappings for t
possible to display different aesthetics in diffe

```
ggplot(flights_weather_day, aes(x = pressure, y = dep_del
  geom_point(aes(color = origin)) +
  geom_smooth()
```

Multiple Geoms 2



Multiple Geoms 3

You can use the same idea to specify differen
smooth line displays just a subset of the data
JFK. The local data argument in `geom_smooth()`
argument in `ggplot()` for that layer only.

```
ggplot(flights_weather_day, aes(x = pressure, y = dep_del
  geom_point(mapping = aes(color = origin)) +
  geom_smooth(data = filter(flights_weather_day, origin =
```

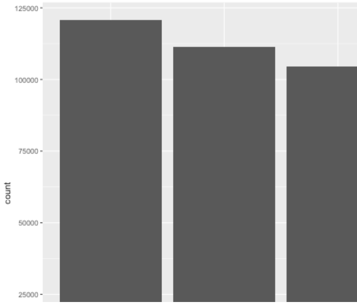
Multiple Geoms 3

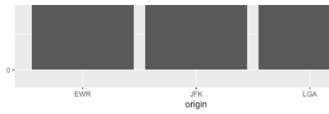
dep_delay
100
75
50
25
0

Statistical Transformations -

Let's make a bar chart of the number of flight
The algorithm uses a built-in statistical transf
calculate the counts.

```
ggplot(flights_weather, aes(x = origin)) +
  geom_bar()
```

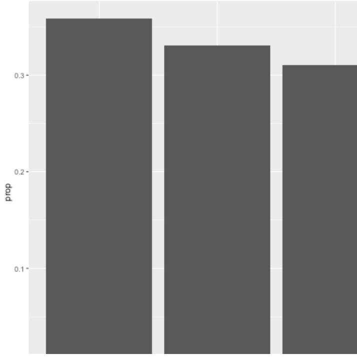




Bar charts 2

You can override the statistic a `geom` uses to co
want to plot proportions, rather than counts:

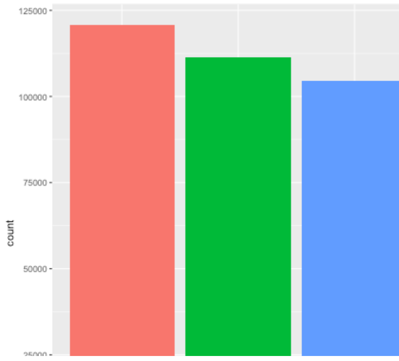
```
ggplot(flights_weather, aes(x = origin, y = stat(prop), g
  geom_bar()
```

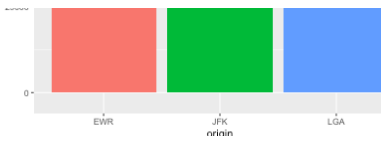


Coloring barcharts

You can color a bar chart using either the `col`
outline), or, more usefully, `fill`:

```
ggplot(flights_weather, aes(x = origin, fill= origin)) +
  geom_bar()
```





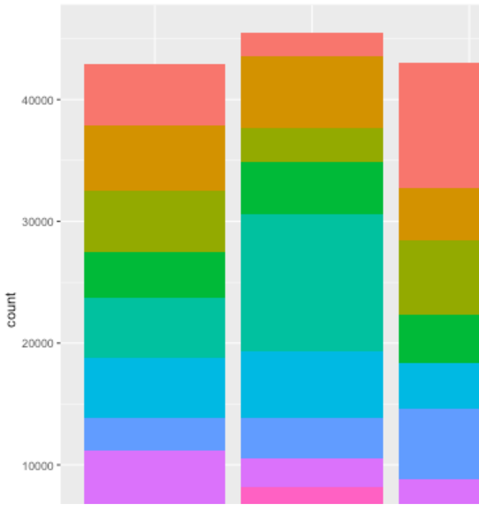
Coloring barcharts

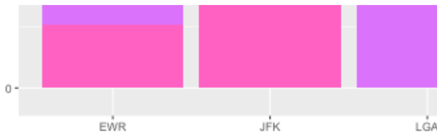
More interestingly, you can fill by another vari
to look at the destination airports with a lot o
flights)

```
flights_weather_ss <- flights_weather |>
  group_by(dest) |>
  filter(n() > 10000)
```

Coloring barcharts

```
ggplot(flights_weather_ss, aes(x = origin, fill= dest)) +
  geom_bar()
```

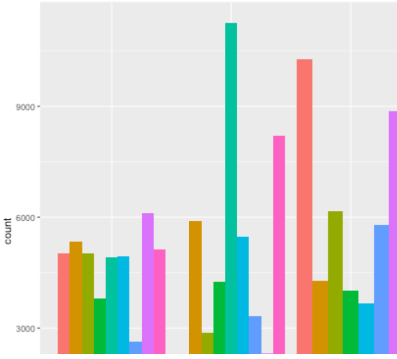


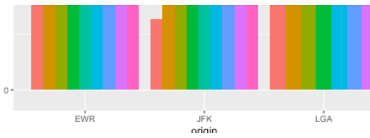


Coloring barcharts

The `position = "dodge"` places overlapping ob
another. This makes it easier to compare indiv

```
ggplot(flights_weather_ss, aes(x = origin, fill= dest)) +
  geom_bar(position="dodge")
```



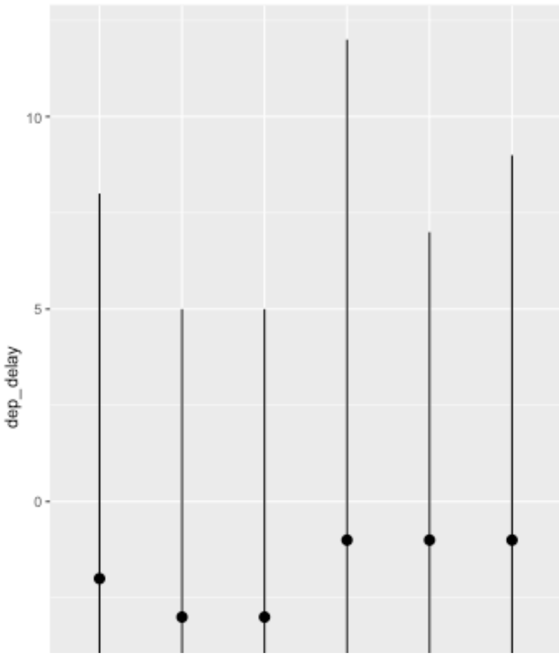


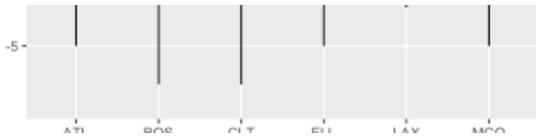
Statistical transformations -

You might want to draw greater attention to t in your code. For example, you might use `sta` the y values for each unique x value, to draw a you're computing:

```
ggplot(flights_weather_ss, aes(x = dest, y = dep_delay))
  stat_summary(fun = median,
               fun.min = function(z) { quantile(z,0.25) }
               fun.max = function(z) { quantile(z,0.75) }
```

Statistical transformations -



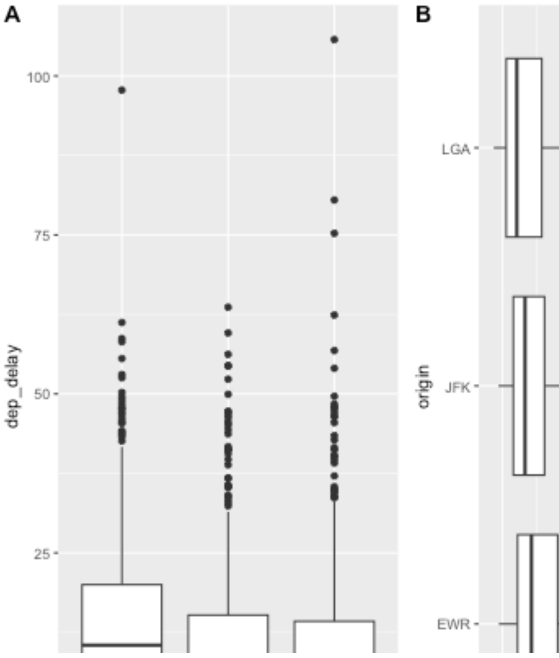


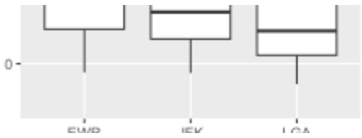
Coordinate systems

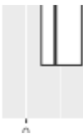
Coordinate systems are one of the more com
start with something simple, here's how to fli

```
unflipped <- ggplot(flights_weather_day, aes(x = origin,
  geom_boxplot()
flipped <- ggplot(flights_weather_day, aes(x = origin, y
  geom_boxplot() +
  coord_flip()
plot_grid(unflipped, flipped, labels = "AUTO")
```

Coordinate systems







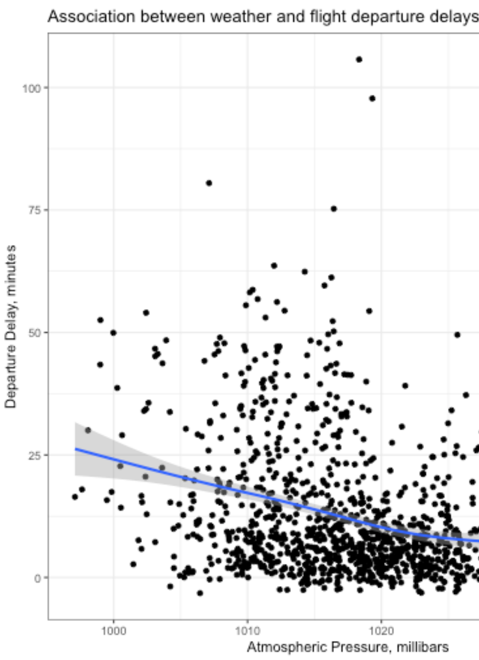
Adding labels

You can make nicer axes and add titles with t

Also showing a minimal theme that removes
`theme_bw()`

```
ggplot(flights_weather_day, aes(x = pressure, y = dep_del
  geom_point() +
  geom_smooth() +
  labs(
    x = "Atmospheric Pressure, millibars",
    y = "Departure Delay, minutes",
    title = "Association between weather and flight depar
  )+
  theme_bw()
```

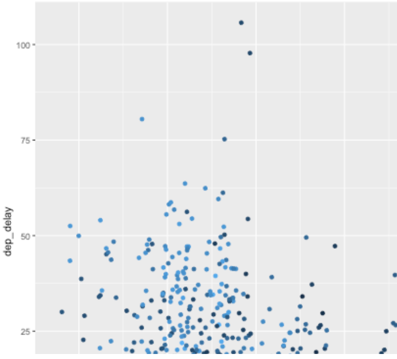
Adding labels

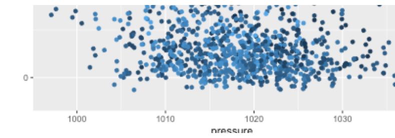


Color ramps

If you add a continuous variable in your color ramp

```
ggplot(flights_weather_day, aes(x = pressure, y = dep_del
  geom_point(aes(colour = temp))
```

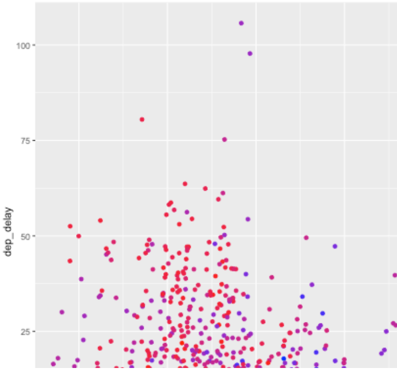


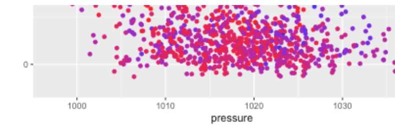


Color palettes

You can define your own color ramp or use or

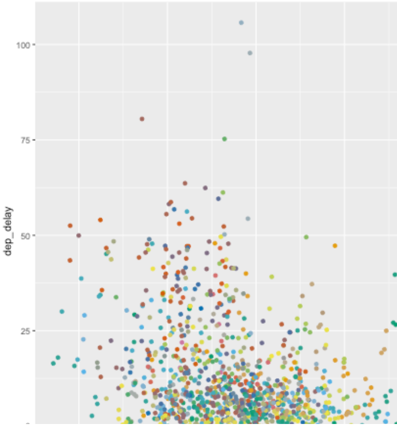
```
ggplot(flights_weather_day, aes(x = pressure, y = dep_del
  geom_point(aes(colour = temp)) +
  scale_colour_gradient(low = "blue", high = "red")
```

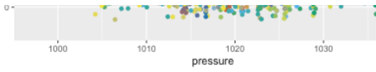




Color palettes

```
cbPalette <- c("#999999", "#E69F00", "#56B4E9", "#009E73"
ggplot(flights_weather_day, aes(x = pressure, y = dep_del
  geom_point(aes(colour = temp)) +
  scale_colour_gradientn(colours = cbPalette)
```





A Great reference

A great (comprehensive) reference for everyth
the R Graphics Cookbook:

https://r-graphics.org/

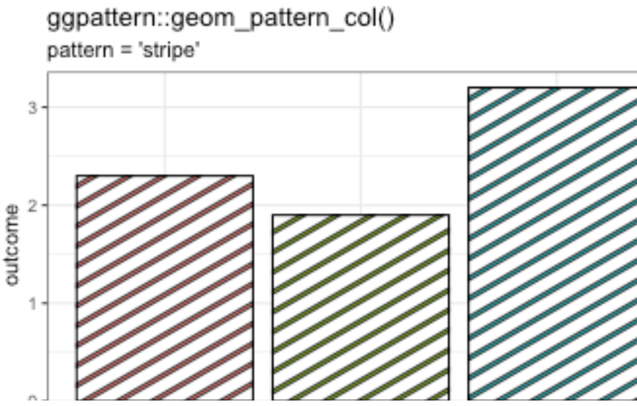
Finally, file under "useless bu

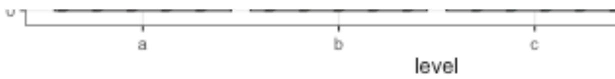
ggpattern - is a library for adding pattern fills

```
  library(ggpattern)
df <- data.frame(level = c("a", "b", "c", "d"), outcome =

ggplot(df, aes(level, outcome, pattern_fill = level)) +
  geom_col_pattern(pattern = "stripe", fill = "white", co
  theme(legend.position = "none") +
  labs(title = "ggpattern::geom_pattern_col()", subtitle
  coord_fixed(ratio = 1 / 2)+
  theme_bw() +
```

Finally, file under "useless bu




0
a
b
c
level