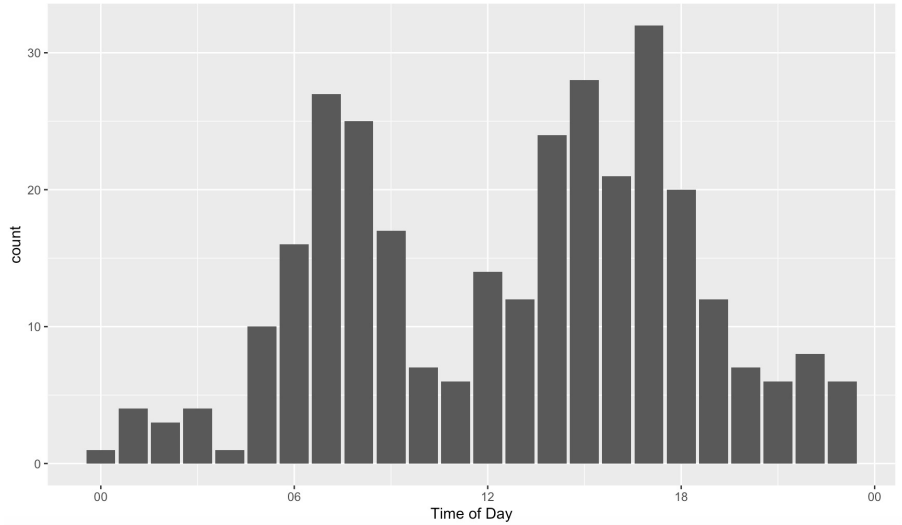
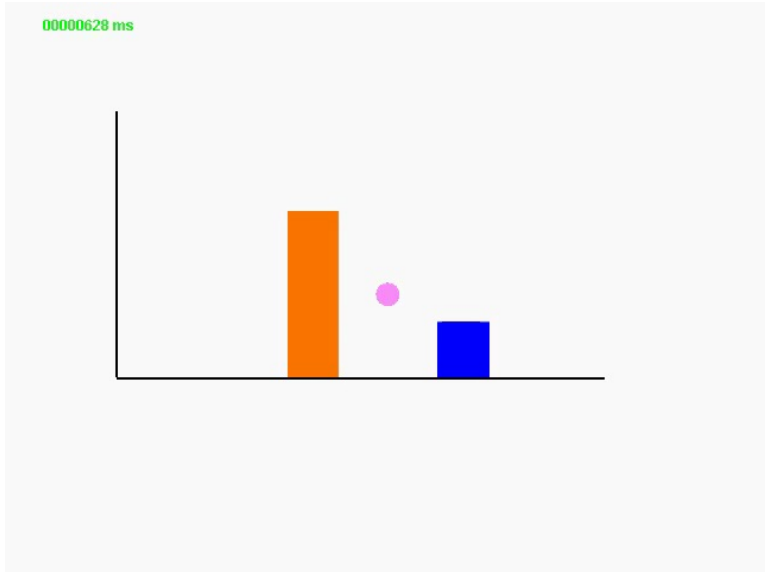


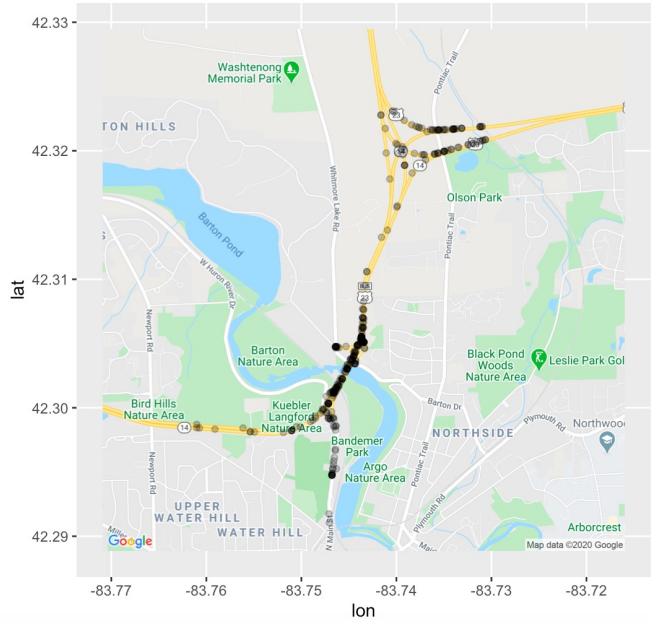
Evidence-Based Data Visualization

Audrey Michal, Ph D.

PDHP Workshop

2/21/20



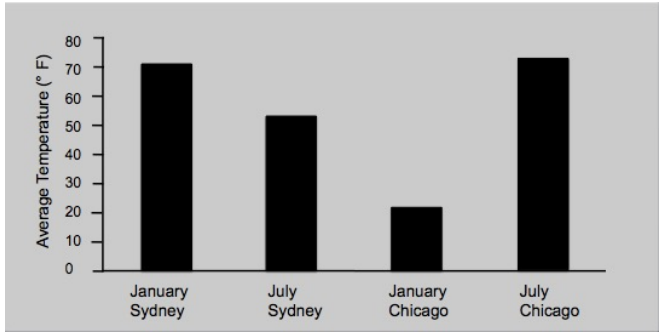


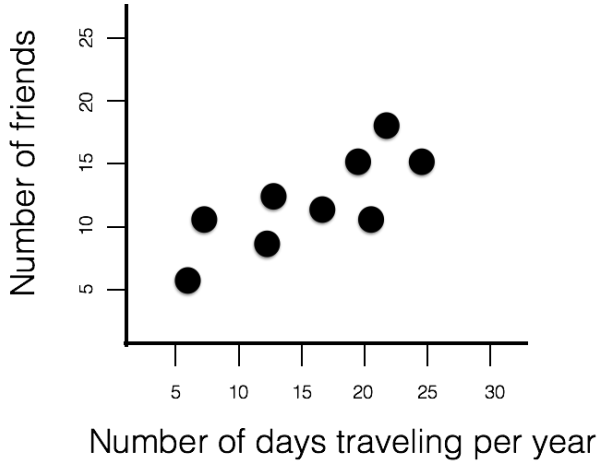
Agenda

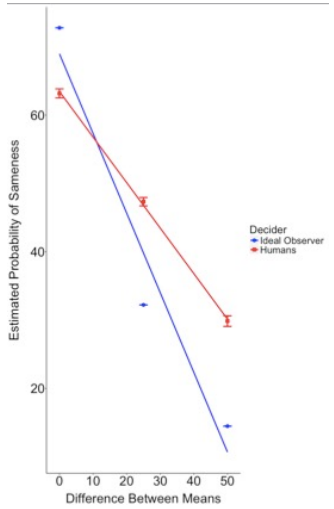
- Introduction to Data Visualization
- Introduction to R and ggplot2
- Data Exploration and Exercises
- Data Explanation, Best Practices and Exercises

What is a data visualization?

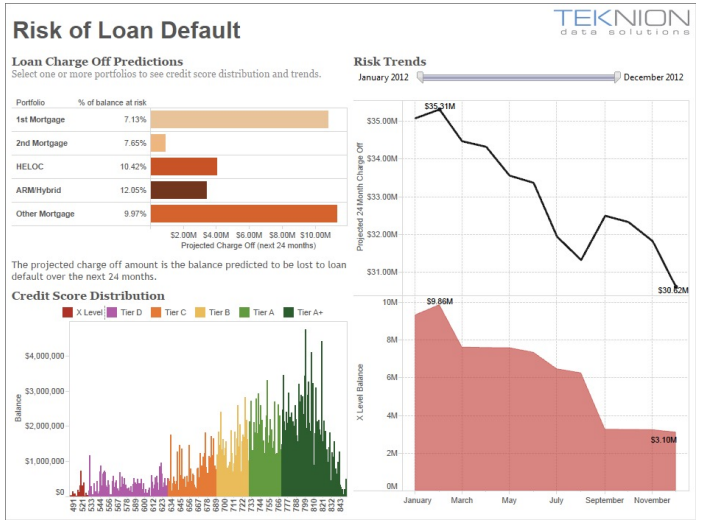
- Visual display of abstracted *data* (information)
- More specific type of **information visualization**

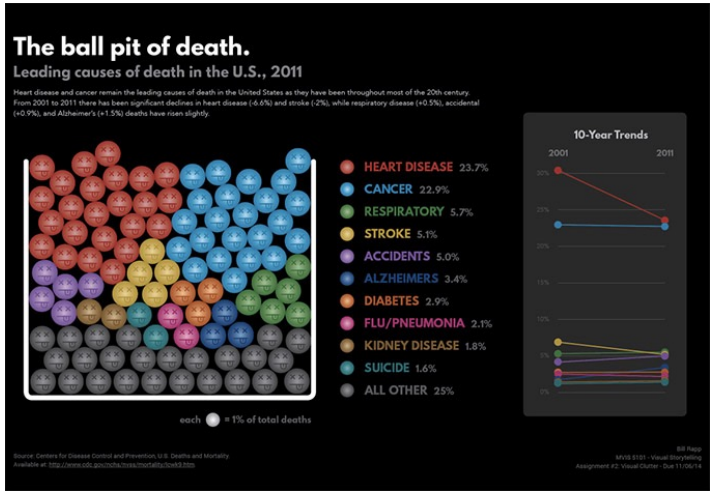


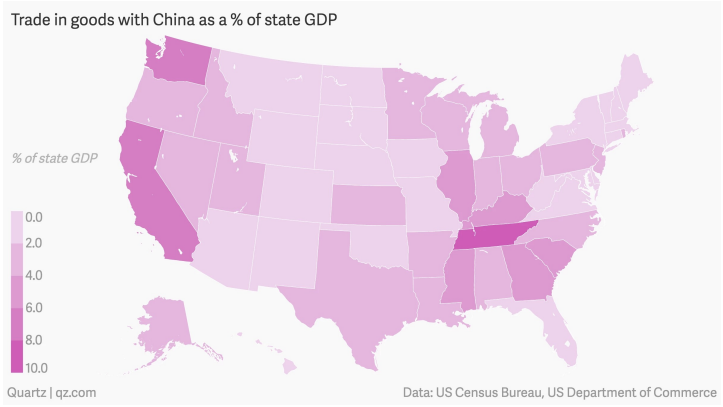


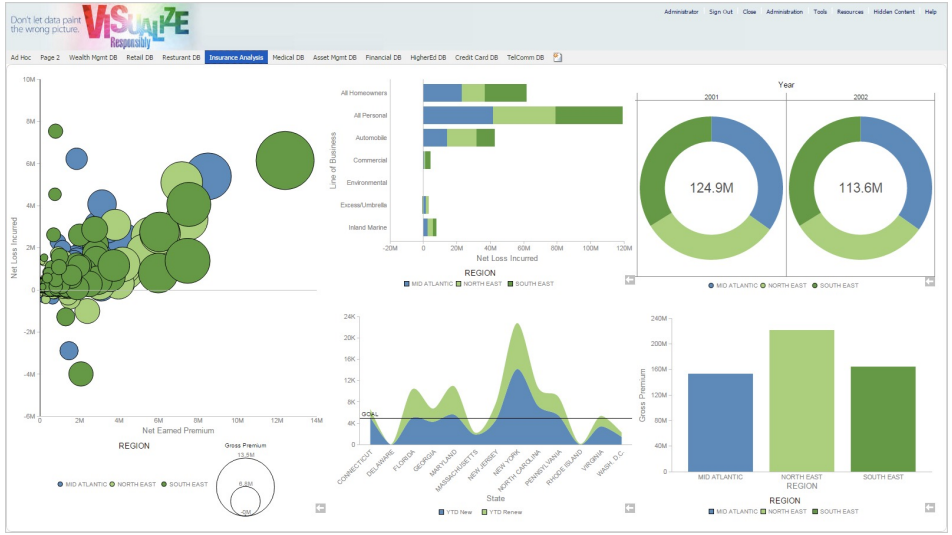


Data: Not just for scientists









Some Goals of Data Visualization

- Communicate data in a clear, efficient, engaging manner
- Allow for data exploration, knowledge/hypothesis generation, insights
- Automatize data visualization, analysis
- Aid decision-making (science, policy, business, medicine, politics etc.)
- Improve learning and comprehension

Cognitive Science Research on Data Visualization

- How do people perceive, interpret and reason about visualizations? What types of errors do they make (biases, etc.)?
- Large previous literature on role of cognition in data viz comprehension (e.g., Carpenter & Shah, 1998; Zacks & Tversky, 1999; Kosslyn, 2006; Freedman & Shah, 2006; Hegarty, 2004)
- Slow, effortful analysis of graphs – *not* automatic, *not* like a 'picture'
- But pervasive perception that data visualizations are intuitive

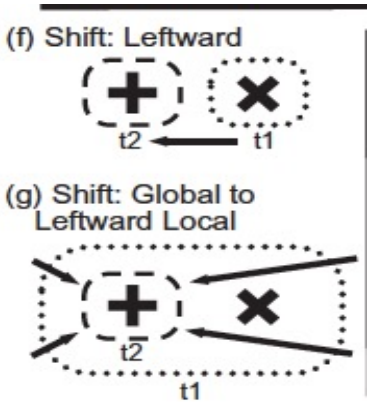
Understanding traditional visualizations

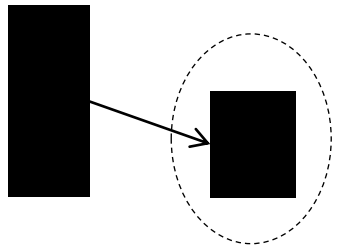
- Bar graphs, line graphs, scatterplots – still much to learn about how we process them
- Visualizations may not always automatically reveal obvious meaningful patterns
- We need better understanding of how people process data visually, even for simple datasets

Interpreting 2-bar Graphs

- Theory: people extract spatial relations by sequentially attending to items (Franconeri et al., 2012)

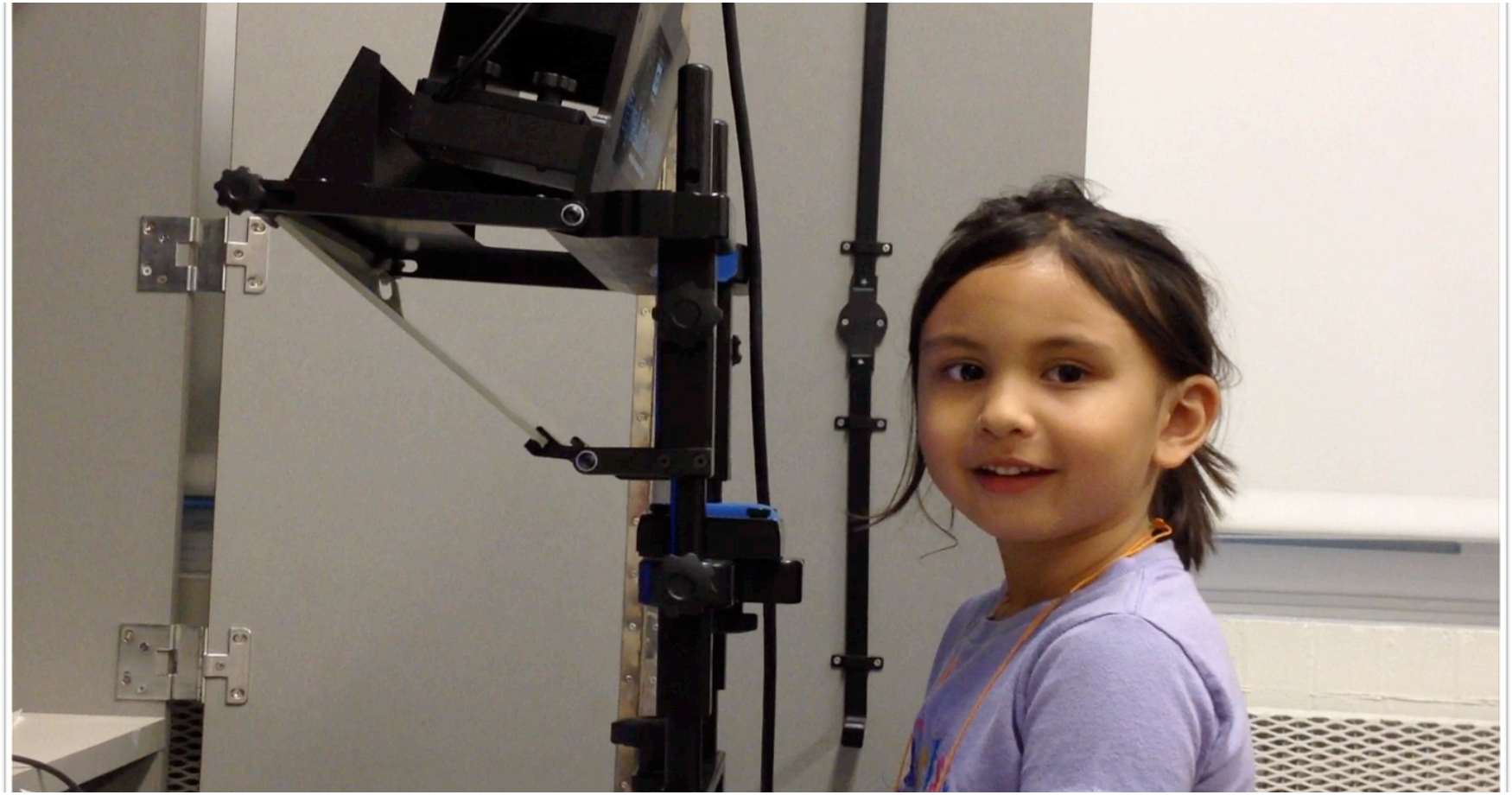
- Does sequential attention also occur for magnitude relations (bar graphs)?



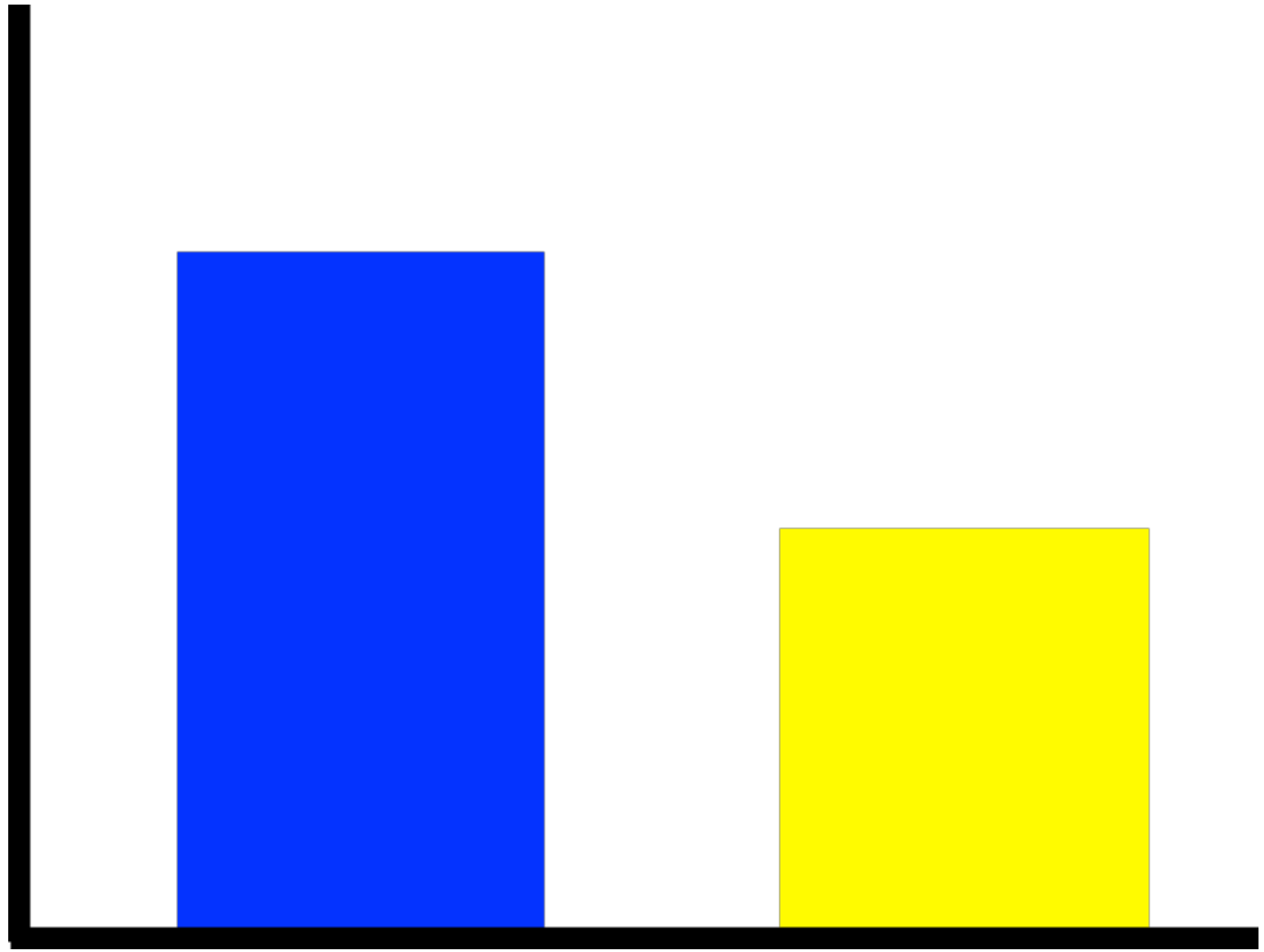


Hypothesis: the order in which data points are compared determines what conclusion is drawn

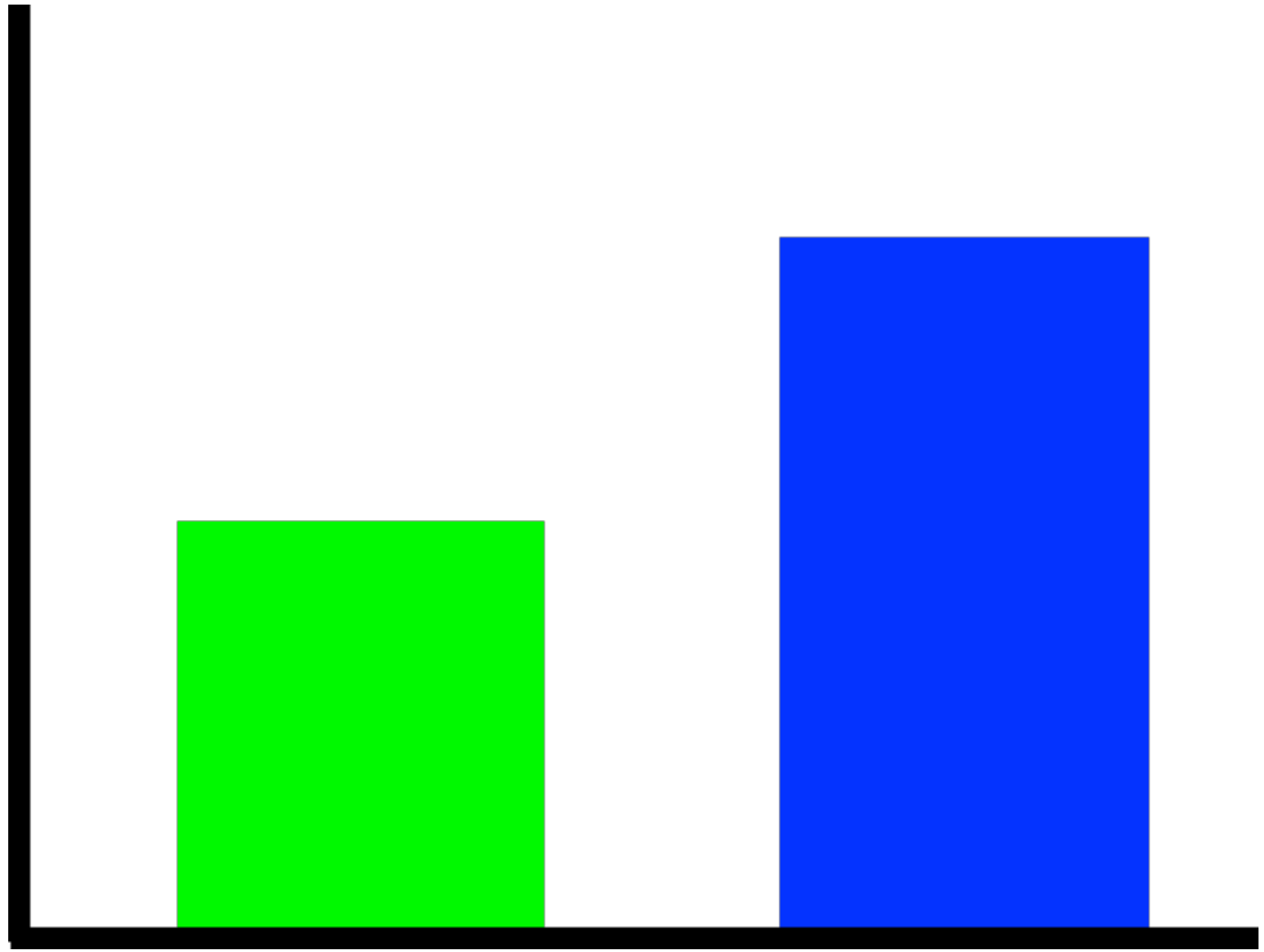
Eye movements? Developmental differences?



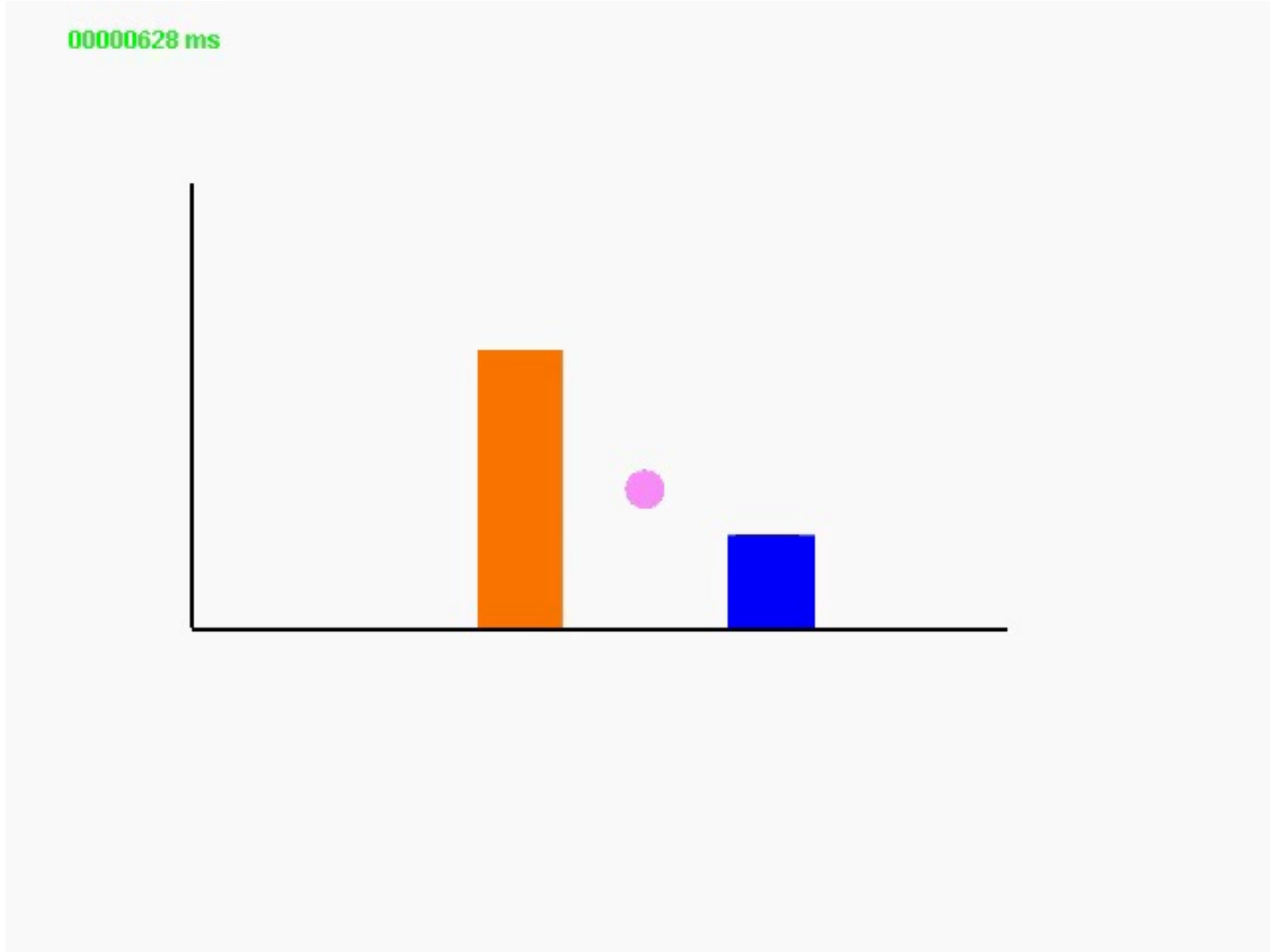
Are there fewer bananas than blueberries?



Are there more limes than blueberries?



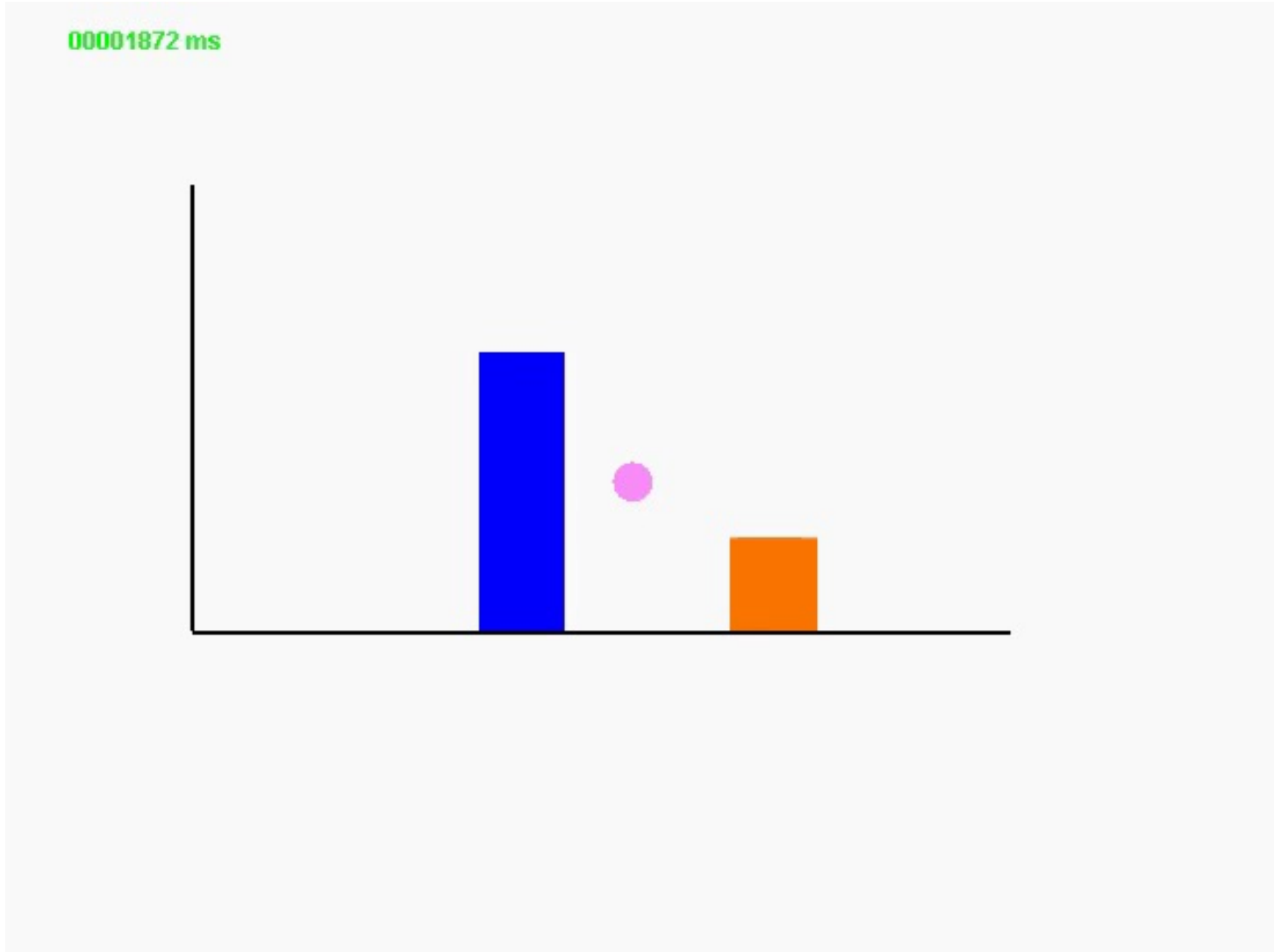
Are there fewer blueberries than oranges?



Slower RT 8 year-old

Left to right perceptual order, regardless of question

Are there fewer oranges than blueberries?



Faster RT 8 year-old

Visual routine order congruent with question

Spatial Routines

Identity Routines

Michal, Shah, Uttal & Franconeri, 2016

Visual Routines: Implications for Data Viz

- People use systematic visual routines when coordinating graphs with language (left bar first, target bar first)
- Visual routines change over the course of development, from inflexible spatial routines in early childhood to flexible, task-based routines
- People may look at the same data visualization in different ways, come away with different conclusions/'insights'

Grouping and Data Viz

Grouping by **proximity**

Within-group chunks invite comparison
(Shah et al., 1999)

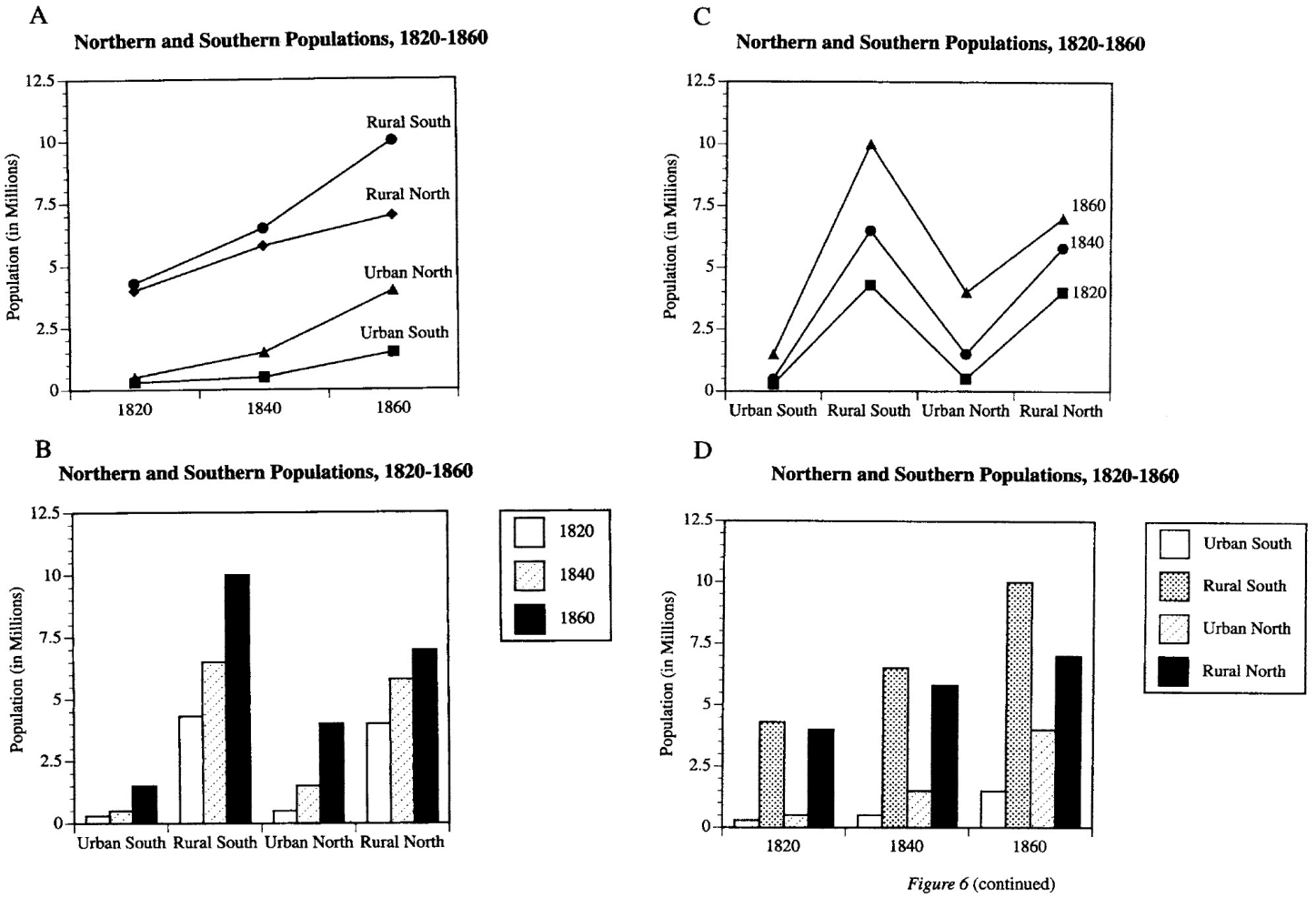
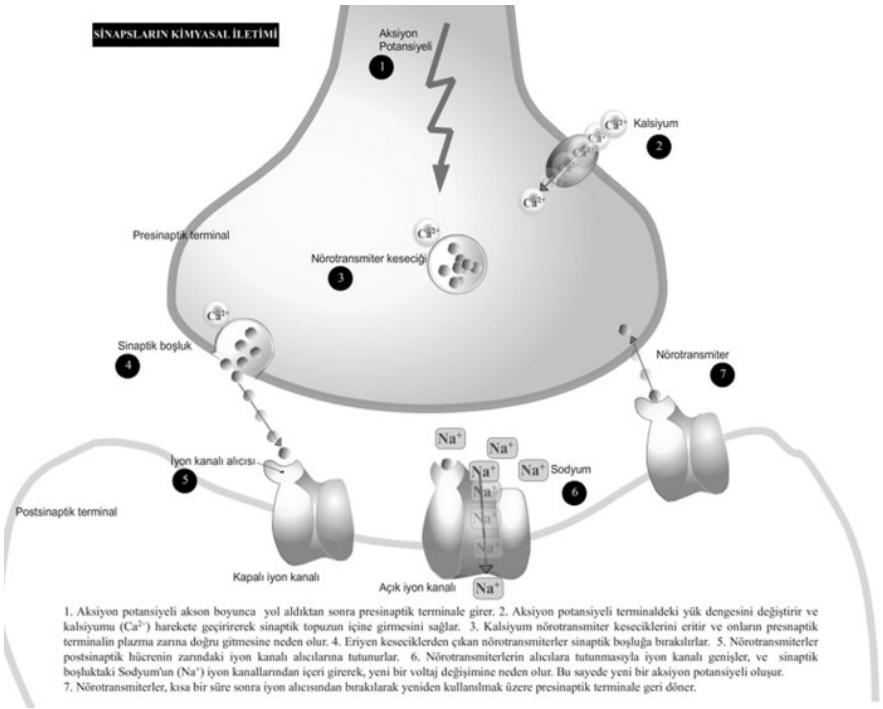
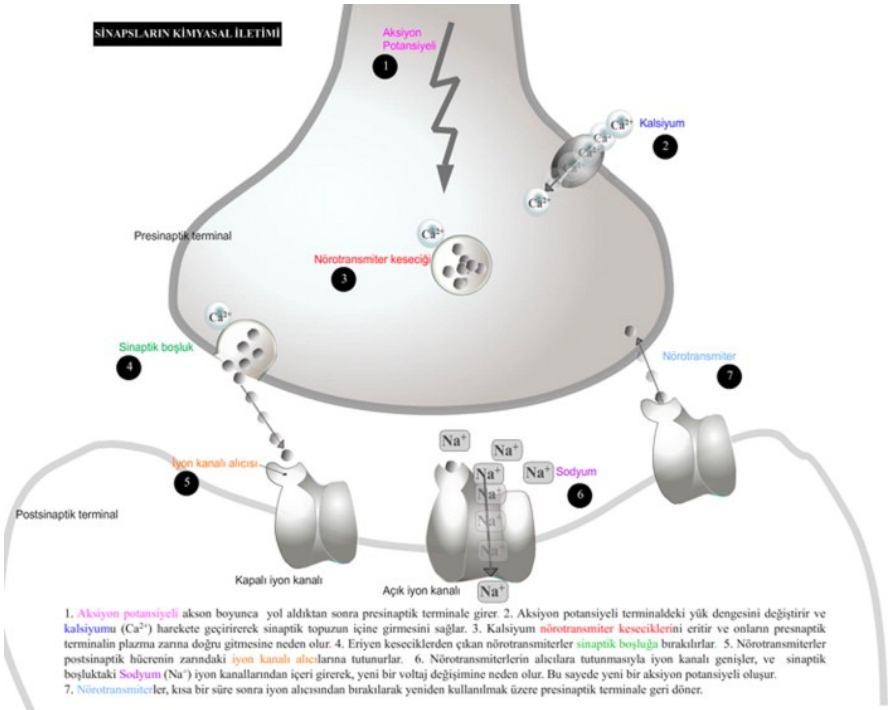


Figure 6 (continued)

Grouping and Data Viz

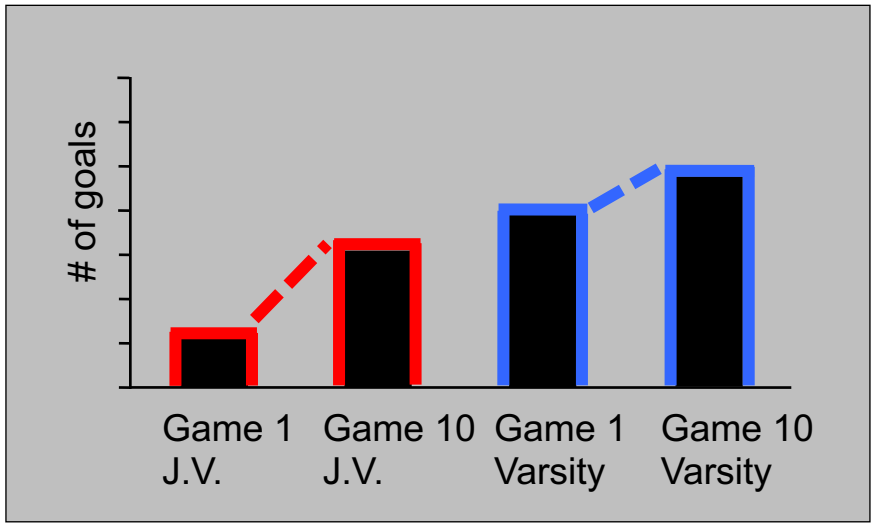
Grouping by **color similarity:** enhance correspondences between text and visualization (Ozcelik et al., 2009)





Grouping and Data Viz

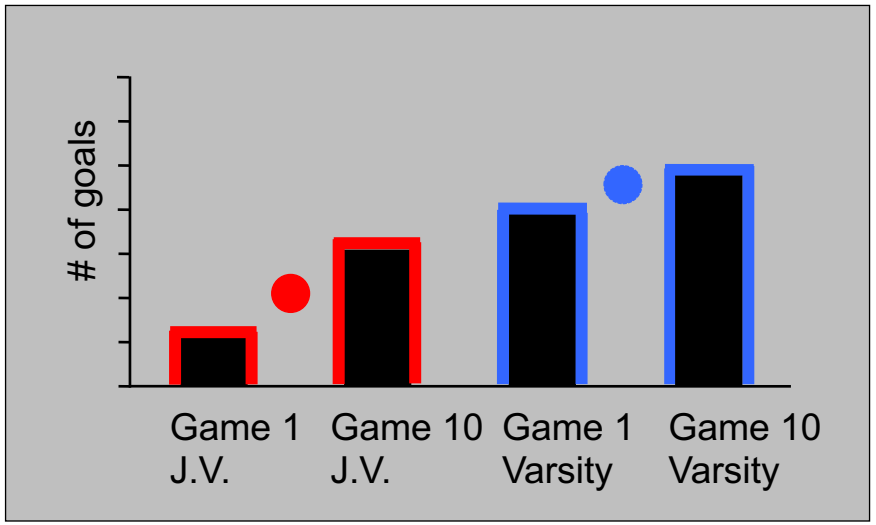
Grouping by **color similarity**: data points that share a color are more easily grouped/compared (Michal et al., 2018)



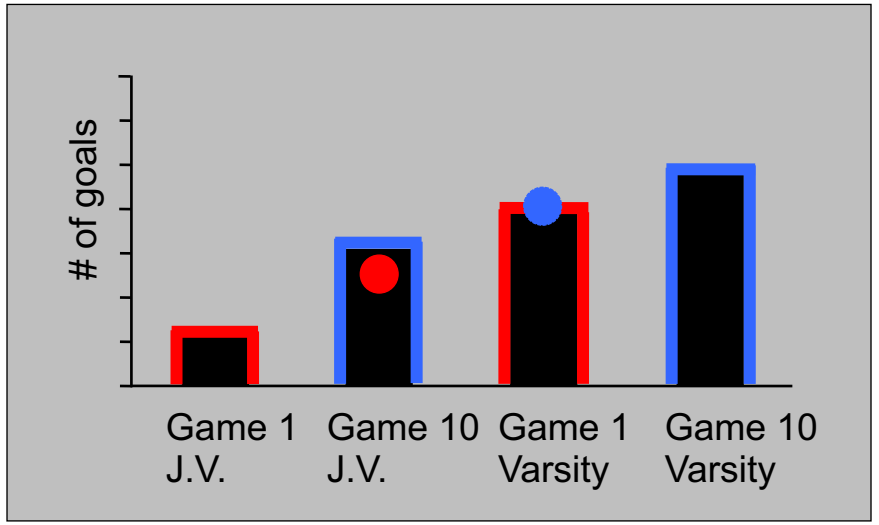
Grouping and Data Viz

- Grouping by **common fate**
- Data points that move together are more easily grouped together (Michal et al., 2018)

Comparison 1

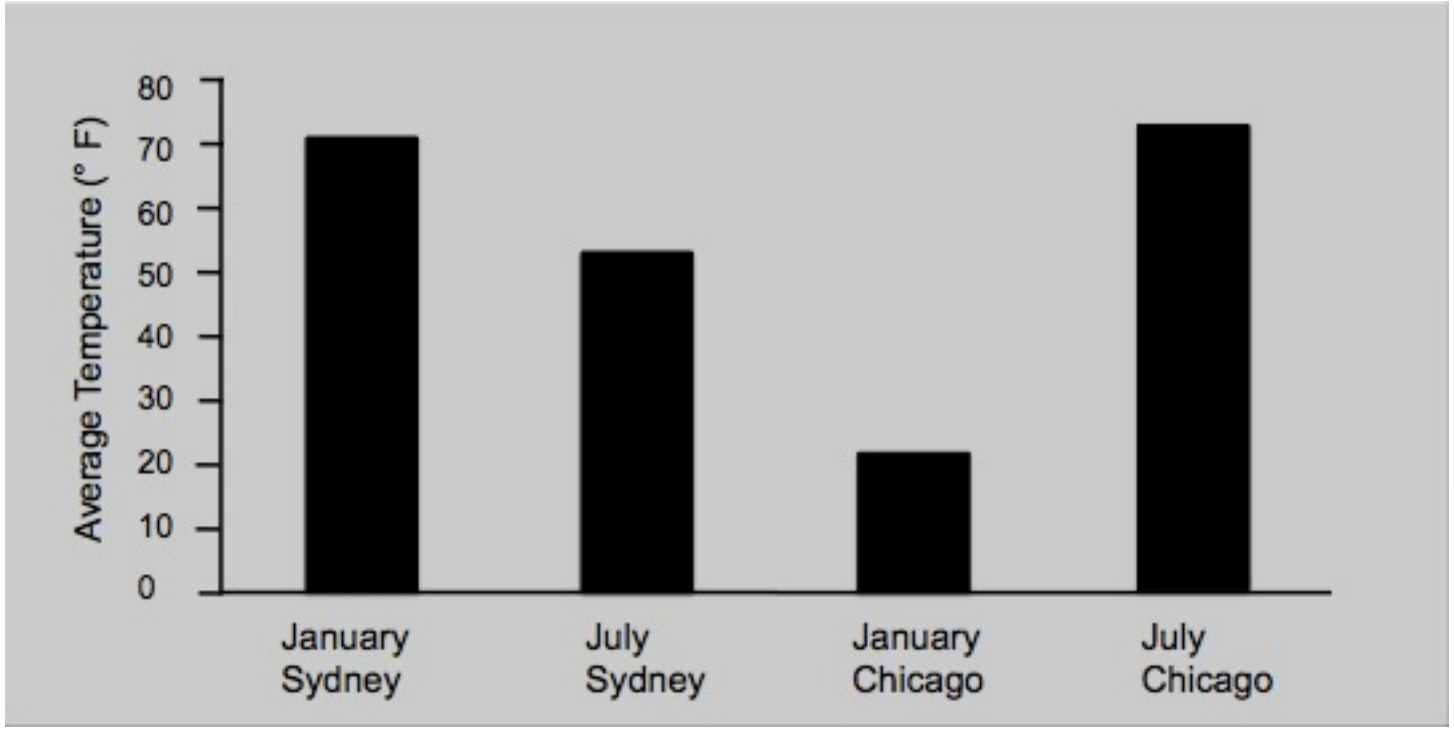


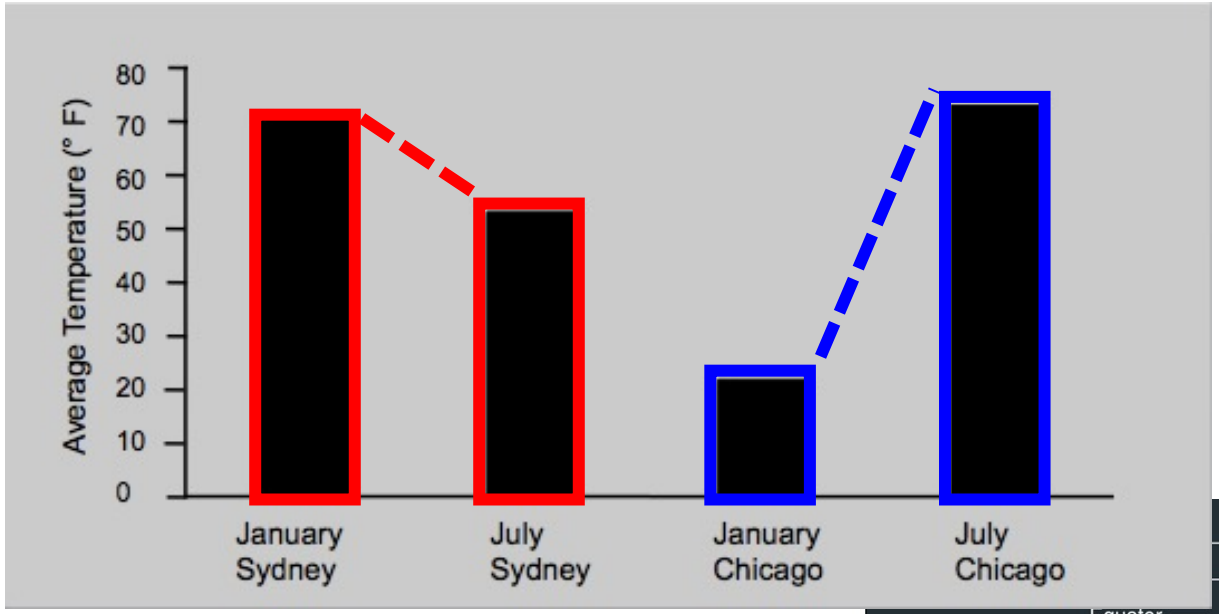
Comparison 2

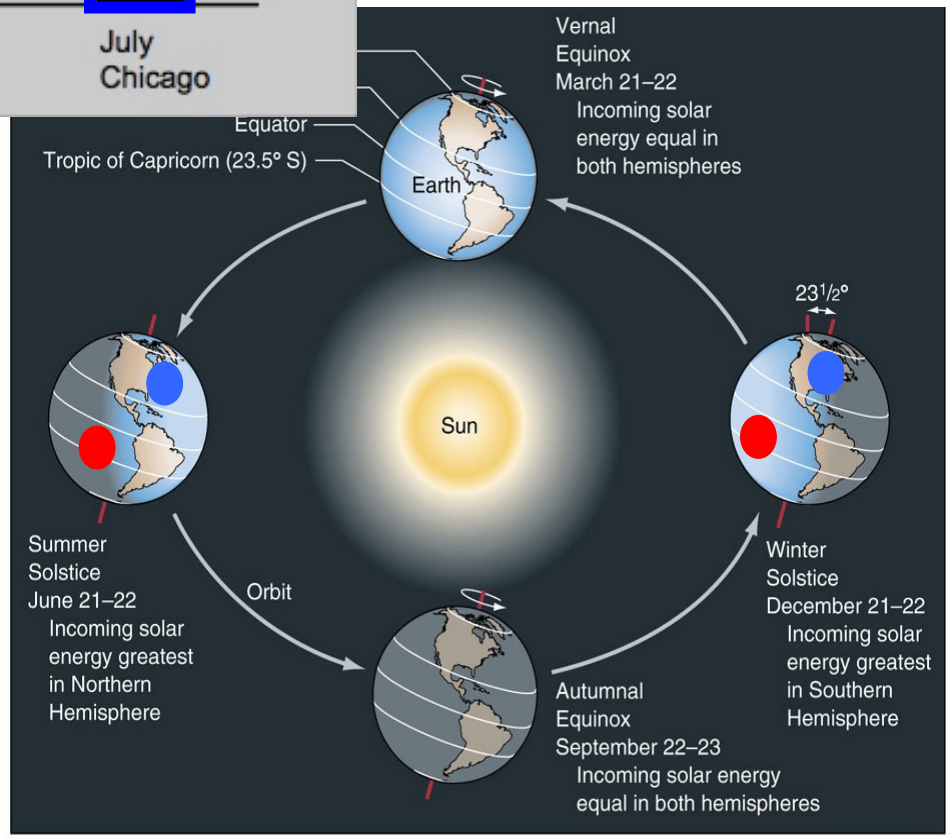


Can Data Visualizations Be Used as a Pedagogical Tool?

Interaction between month and city temperature is key for grasping how Earth's tilt causes seasonal changes




Average Temperature (° F)
80
70
60
50
40
30
20
10
0
January
Sydney
July
Sydney
January
Chicago
July
Chicago


Vernal
Equinox
March 21–22
Incoming solar
energy equal in
both hemispheres
Equator
Tropic of Capricorn (23.5° S)
Earth
23½°
Sun
Summer
Solstice
June 21–22
Incoming solar
energy greatest
in Northern
Hemisphere
Orbit
Winter
Solstice
December 21–22
Incoming solar
energy greatest
in Southern
Hemisphere
Autumnal
Equinox
September 22–23
Incoming solar energy
equal in both hemispheres

Introduction to R and ggplot2

First, downloads

1) Software: Rstudio (https://rstudio.com/products/rstudio/download/)

2) R Packages:

- ggplot2: https://ggplot2.tidyverse.org/

  install.packages(”tidyverse”)
- gganimate: https://cloud.r-project.org/package=gganimate

  install.packages(“gganimate”)

  install.packages(“gifski”)
- ggmap: https://cran.r-project.org/web/packages/ggmap/ggmap.pdf

  install.packages(“ggmap”)
- lubridate

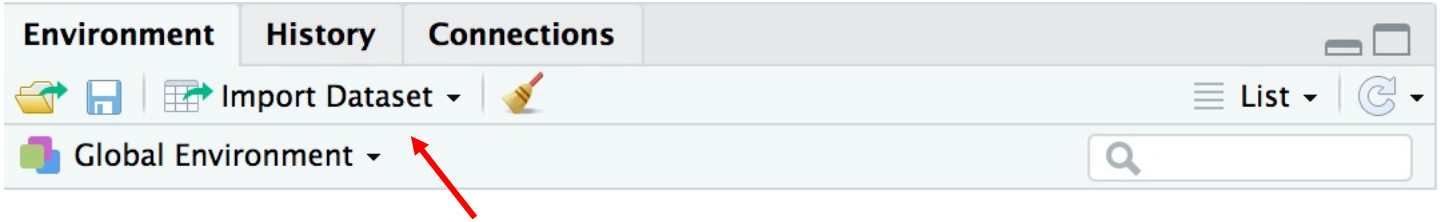
  install.packages(“lubridate”)

3) Data set: https://docs.google.com/spreadsheets/d/19An6THFlgDS8bsf2BA8rYkHUhhjlszqmRzb-xOgbnrs/edit?usp=sharing

- Download onto your computer as Excel spreadsheet, save as “**aacrash.xlsx**”

Import the Data Set

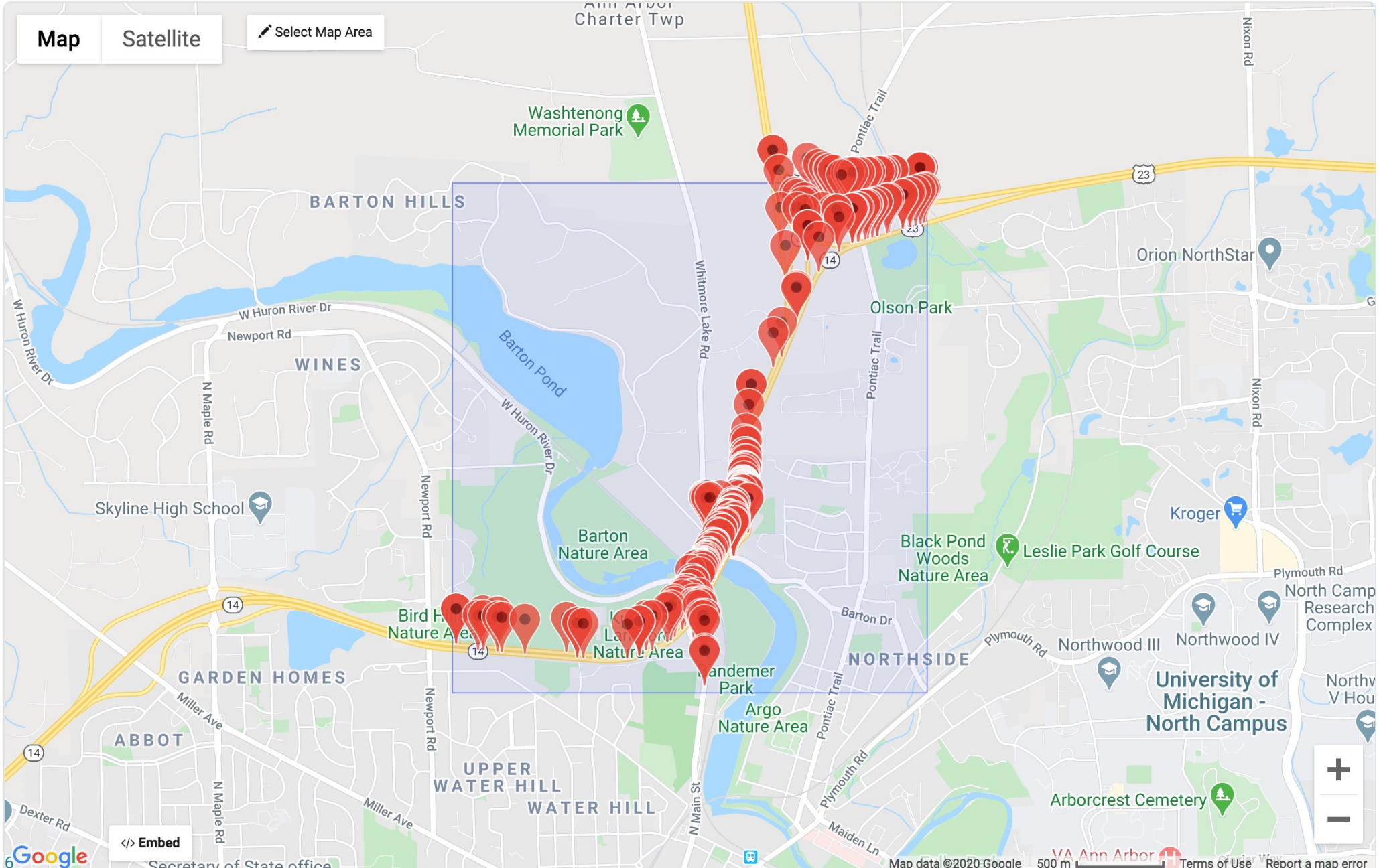
- Open Rstudio and click on the **Import Dataset** button in the top right panel



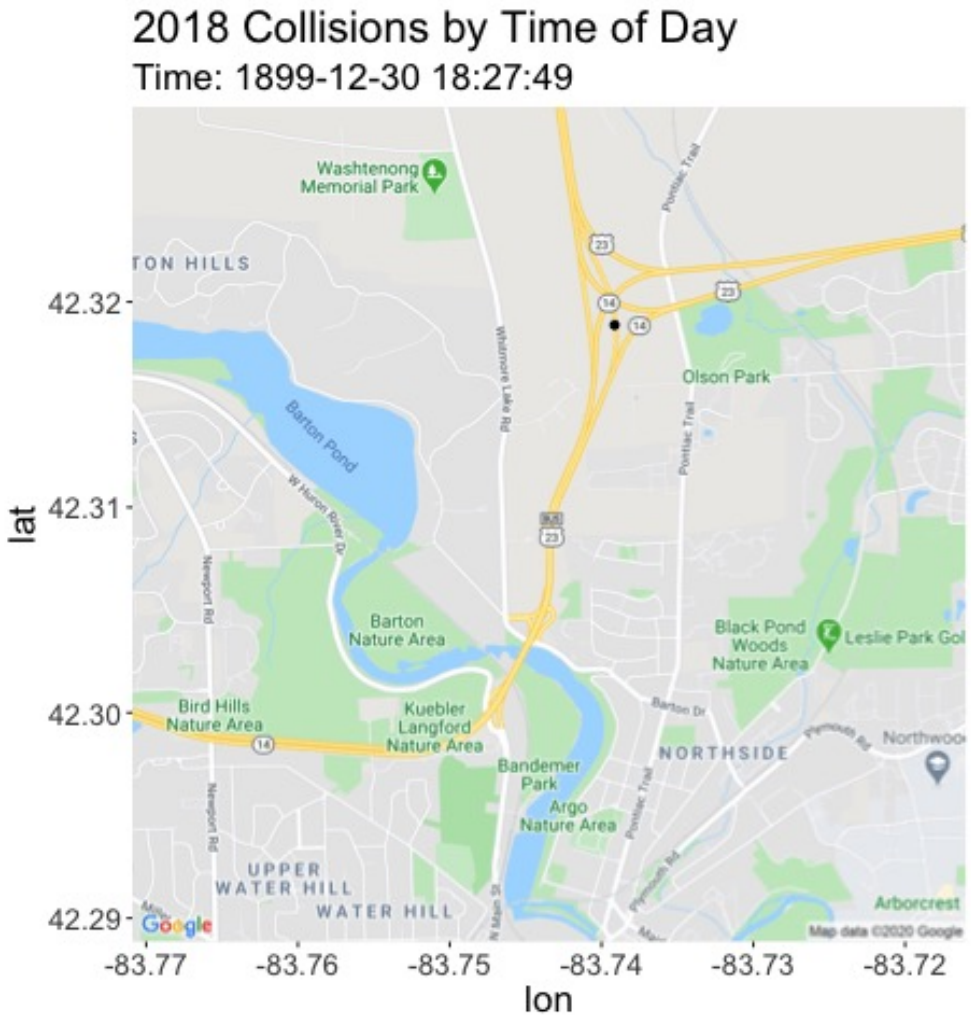
- Import as **Excel Sheet**
- Browse to find your downloaded data set **(aacrash.xlsx)**

296 Crashes

View Crashes by Category: Select ...



By the end of the workshop, you will make this!



Ann Arbor Collision Data: Some Potential Variables of Interest

- Road conditions
- Crash month
- Lighting conditions
- Crash type
- Total motor vehicles
- Hit-and-run
- Truck/Bus
- Time of day
- Day of week
- Driver young/old
- Weather conditions

Data Exploration

Visualization goal: efficiently switch between different formats, variables

Data Exploration

Example goals:

1) What is the overall distribution of my data?
   - E.g., collisions over time, over space
2) Are there outliers or anomalies?
   - E.g., spikes in collisions during certain months?
3) Which combination of specific variables is most informative?

Introduction to ggplot2

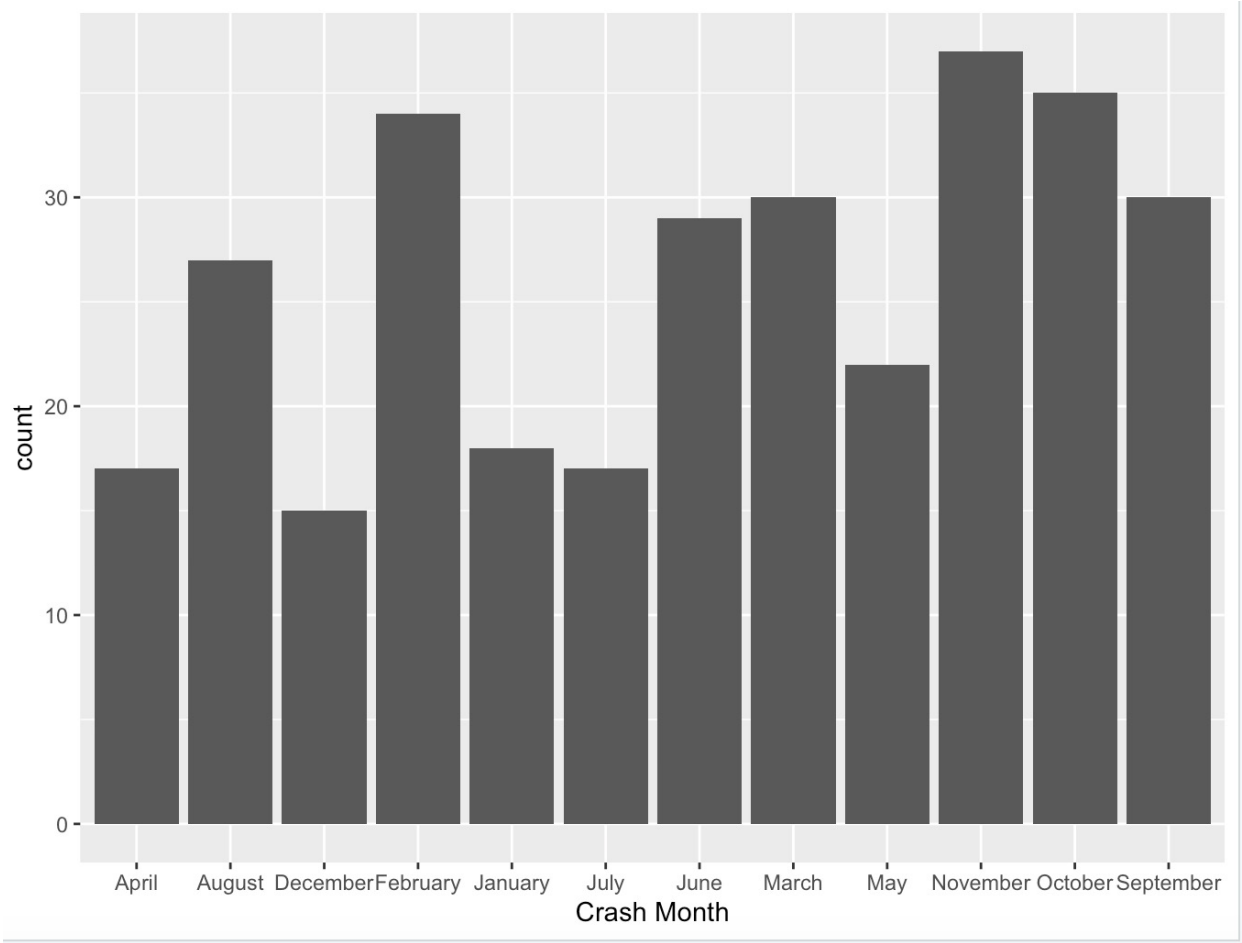
Grammar of graphics: A template

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

- **ggplot:** creates the **plot**, takes argument data (<DATA> is your data set)
- **GEOM_FUNCTION:** creates the **data points** (point, line, bar, etc.), takes argument mapping (controls how x and y variables are mapped to visual properties)
- **aes** ("aesthetic"): controls **visual properties** of the data points (color, shape, size)

Collisions over time: Bar chart

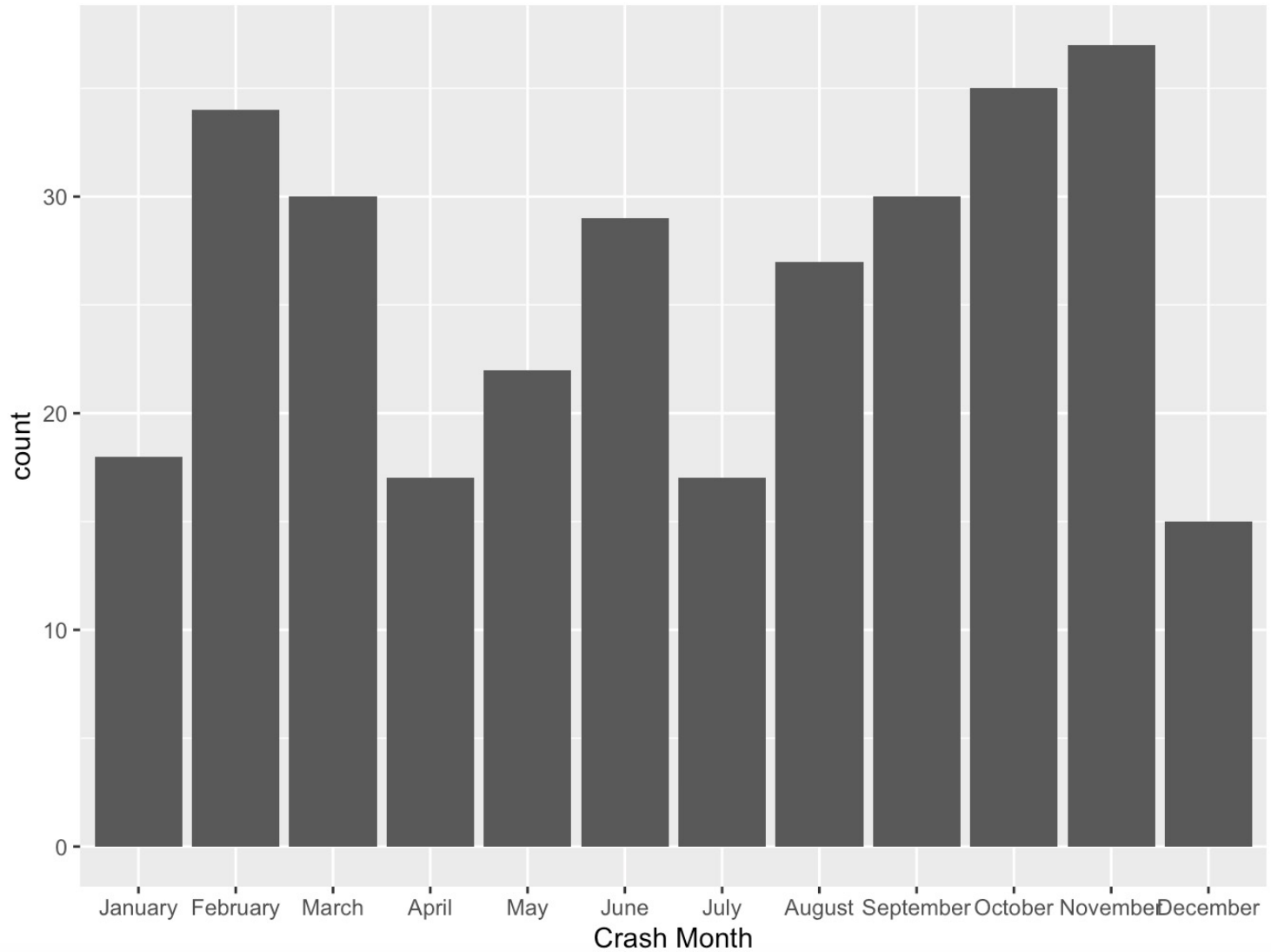
```
ggplot(data=aacrash) +
geom_bar(mapping = aes(x=`Crash Month`))
```



Whoops! Let's fix that label order.

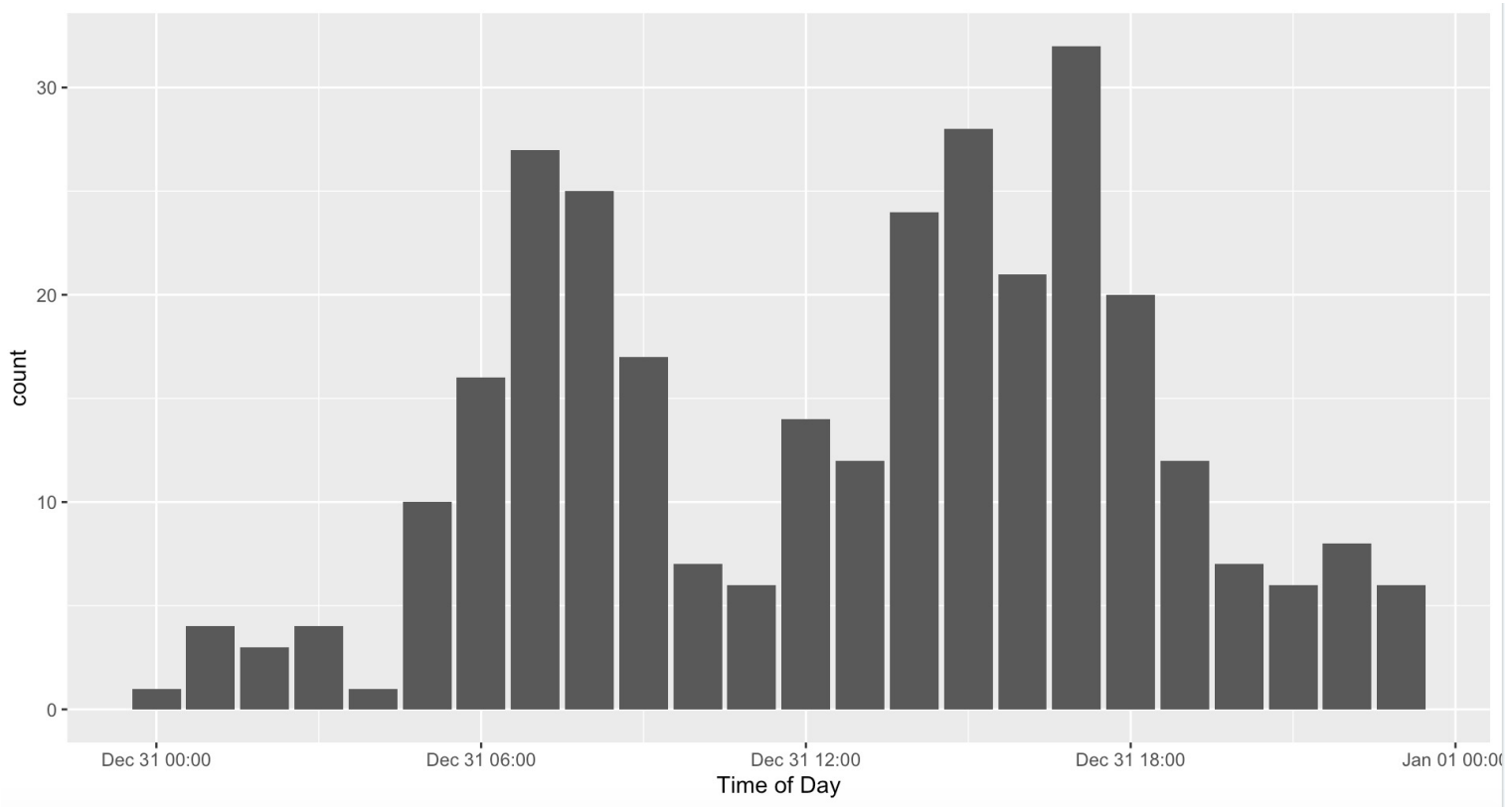
Collisions over time: Ordered by month order, not alphabetically

```
#Change the order of the factor ‘Crash Month’:
aacrash$`Crash Month` <- factor(aacrash$`Crash Month`, levels = c("January", "February", "March”, “April”, “May", "June", "July", "August", "September", "October", "November", "December"))

#Shortcut!
aacrash$`Crash Month` <- factor(aacrash$`Crash Month`, levels = c(month.name))

#Replot:
ggplot(data=aacrash) + geom_bar(mapping = aes(x=`Crash Month`))
```


count
30
20
10
0
January
February
March
April
May
June
July
August
September
October
November
December
Crash Month

Collisions by Time of Day

```
ggplot(data=aacrash) +
geom_bar(mapping =
aes(x=`Time of Day`))
```

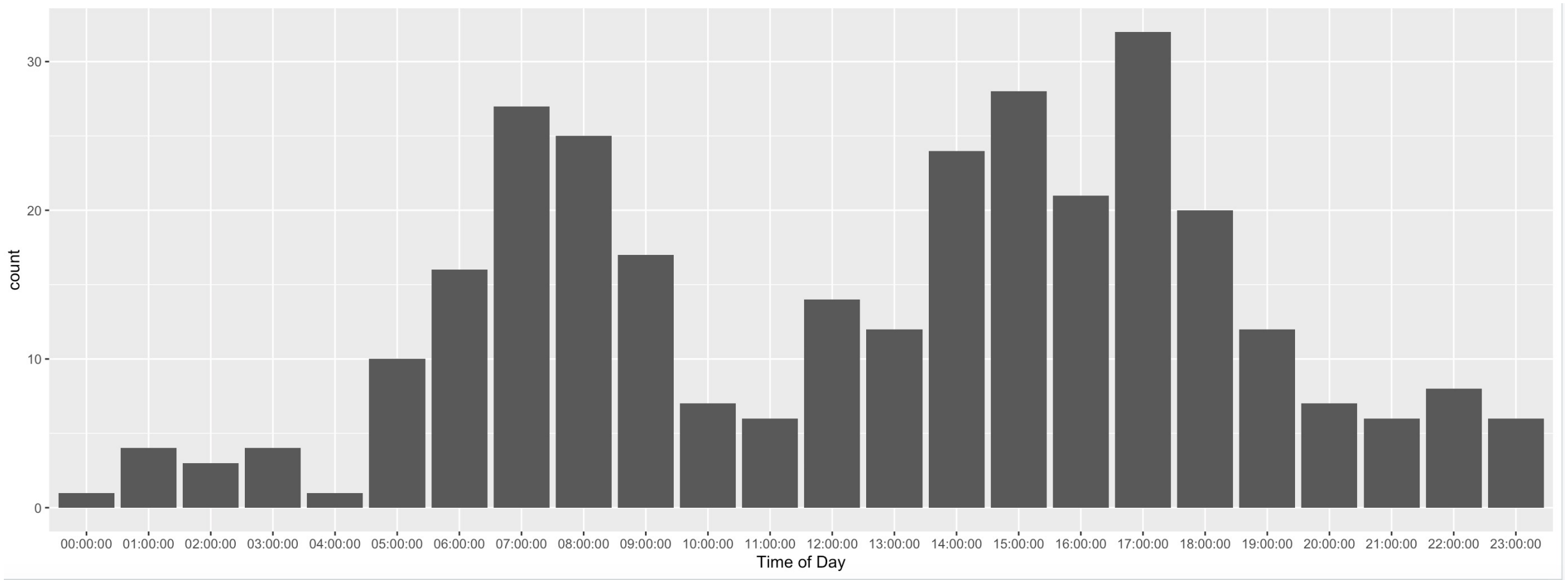


Whoops! Let’s fix that weird time format.

Reformatting Time Variables

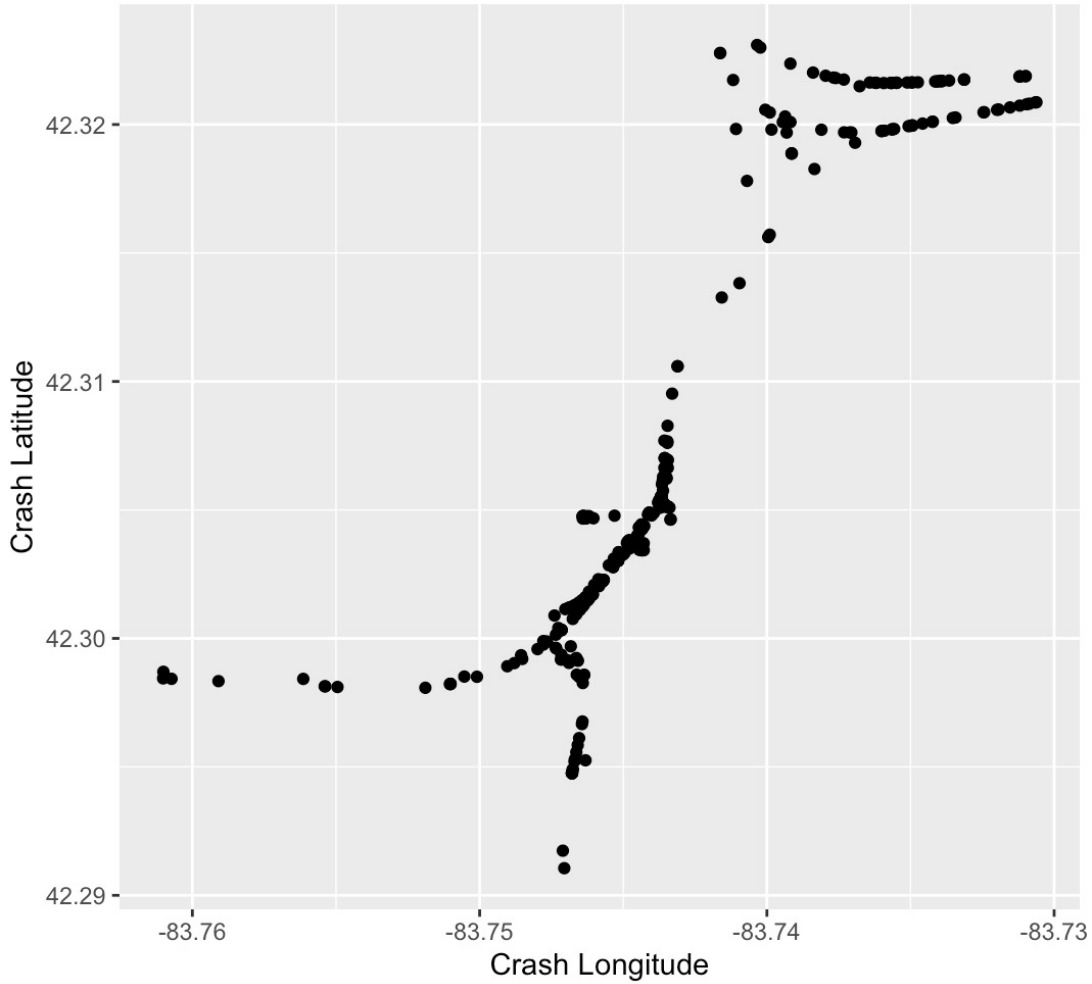
```
#Change format from class POSIXct to character (hms)
aacrash$`Time of Day` <- format(aacrash$`Time of Day`, format = "%H:%M:%S")

#Replot (no scale)
ggplot(data=aacrash) + geom_bar(mapping = aes(x=`Time of Day`))
```

Reformatting Time Variables



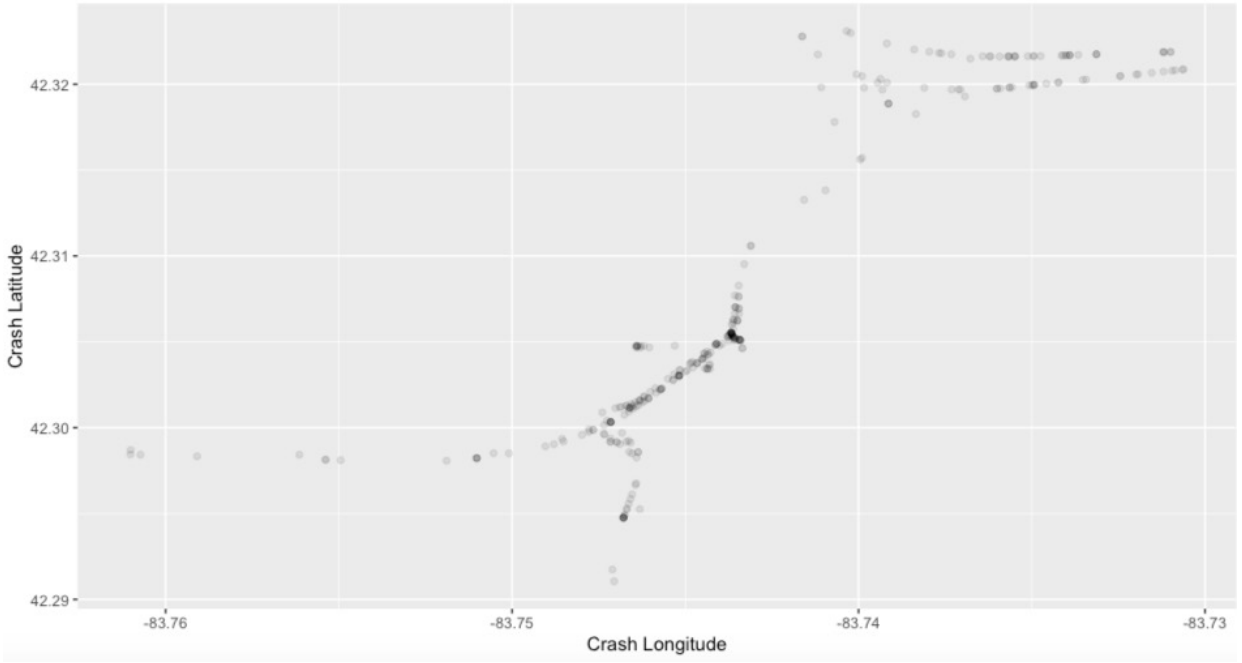
Collisions by Coordinates

```
ggplot(data = aacrash) +
geom_point(mapping = aes(x =
`Crash Longitude`, y = `Crash
Latitude`))
```



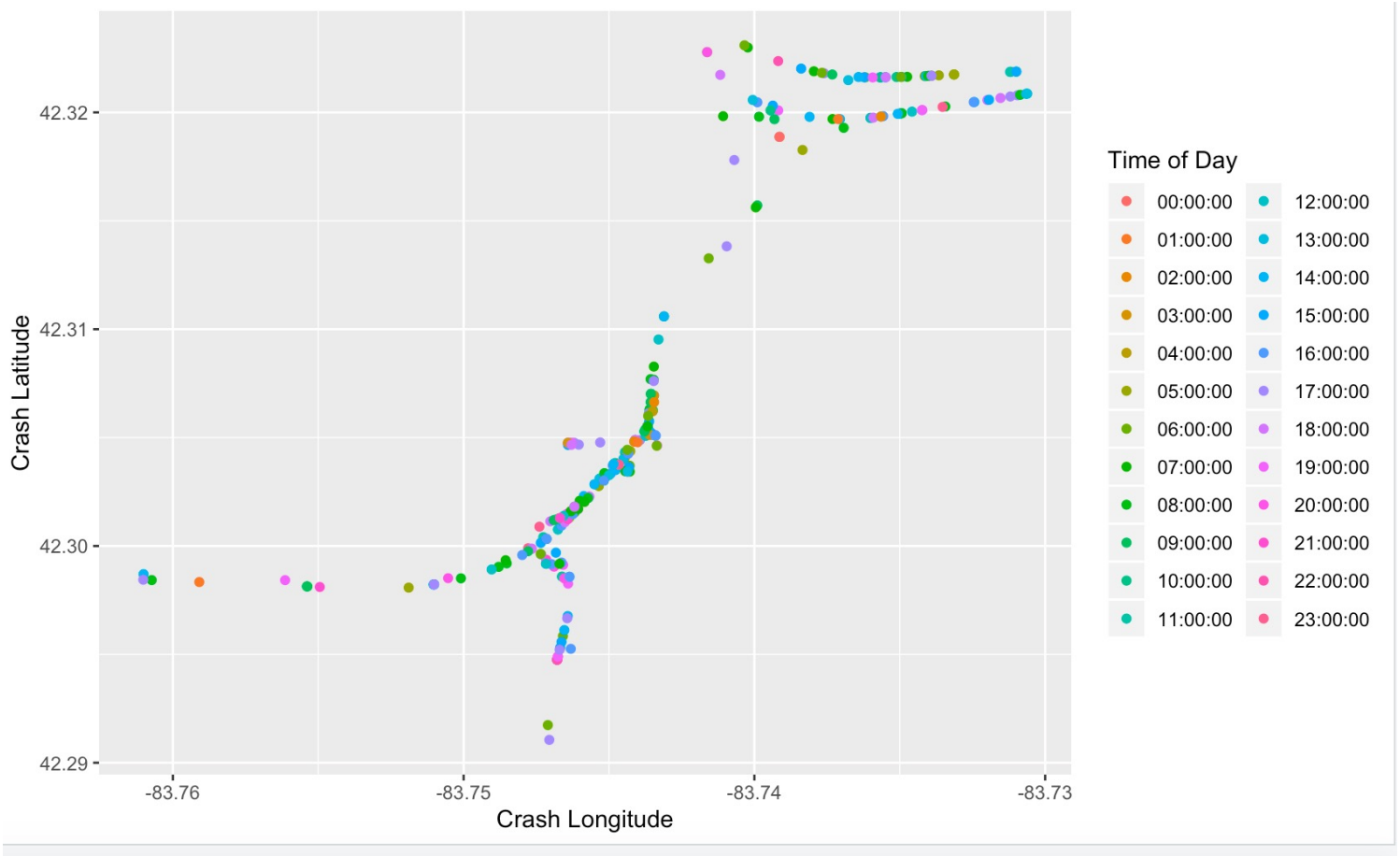
Collisions by Coordinates: Density (transparency)

```
ggplot(data=aacrash) +
geom_point(mapping =
aes(x=`Crash Longitude`,
y=`Crash Latitude`), alpha = .1))
```



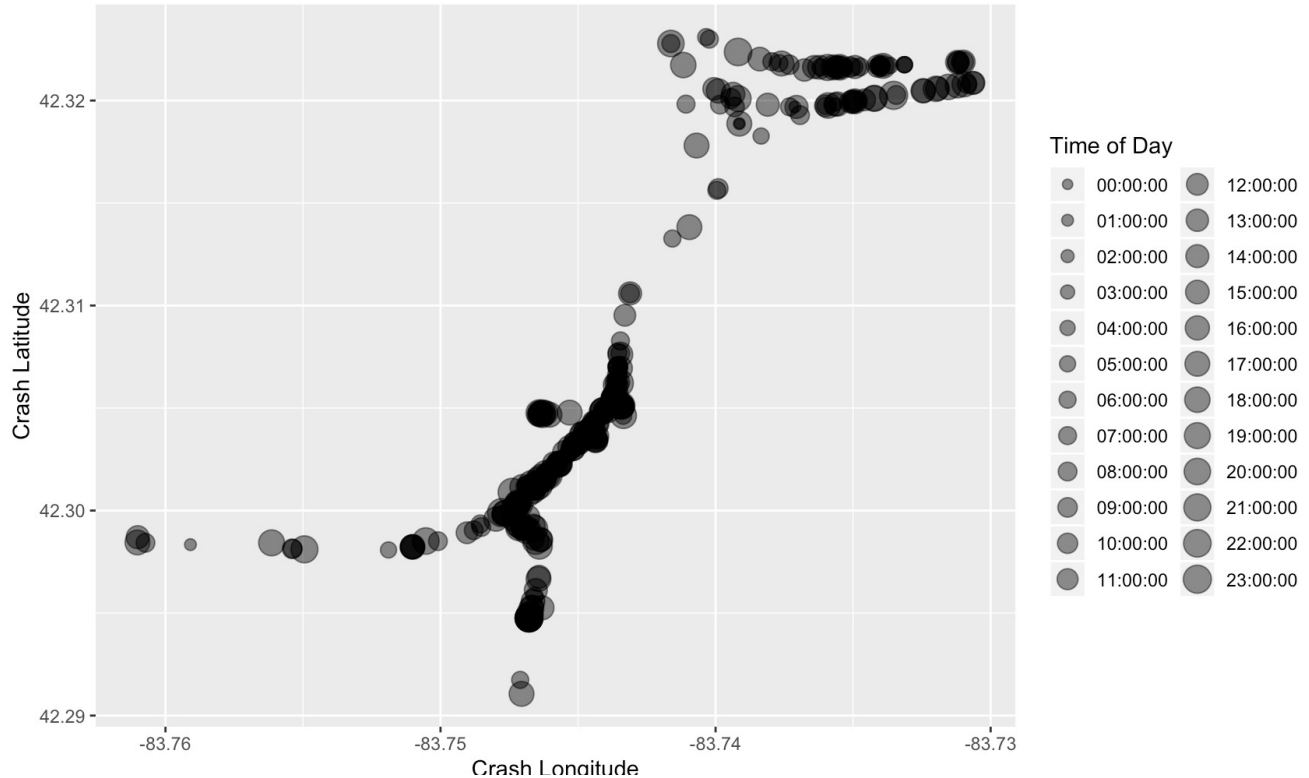
Collisions by Coordinates and Time of Day (color)

```
ggplot(data=aacrash) +
geom_point(mapping =
aes(x=`Crash Longitude`,
y=`Crash Latitude`, color
= `Time of Day`))
```



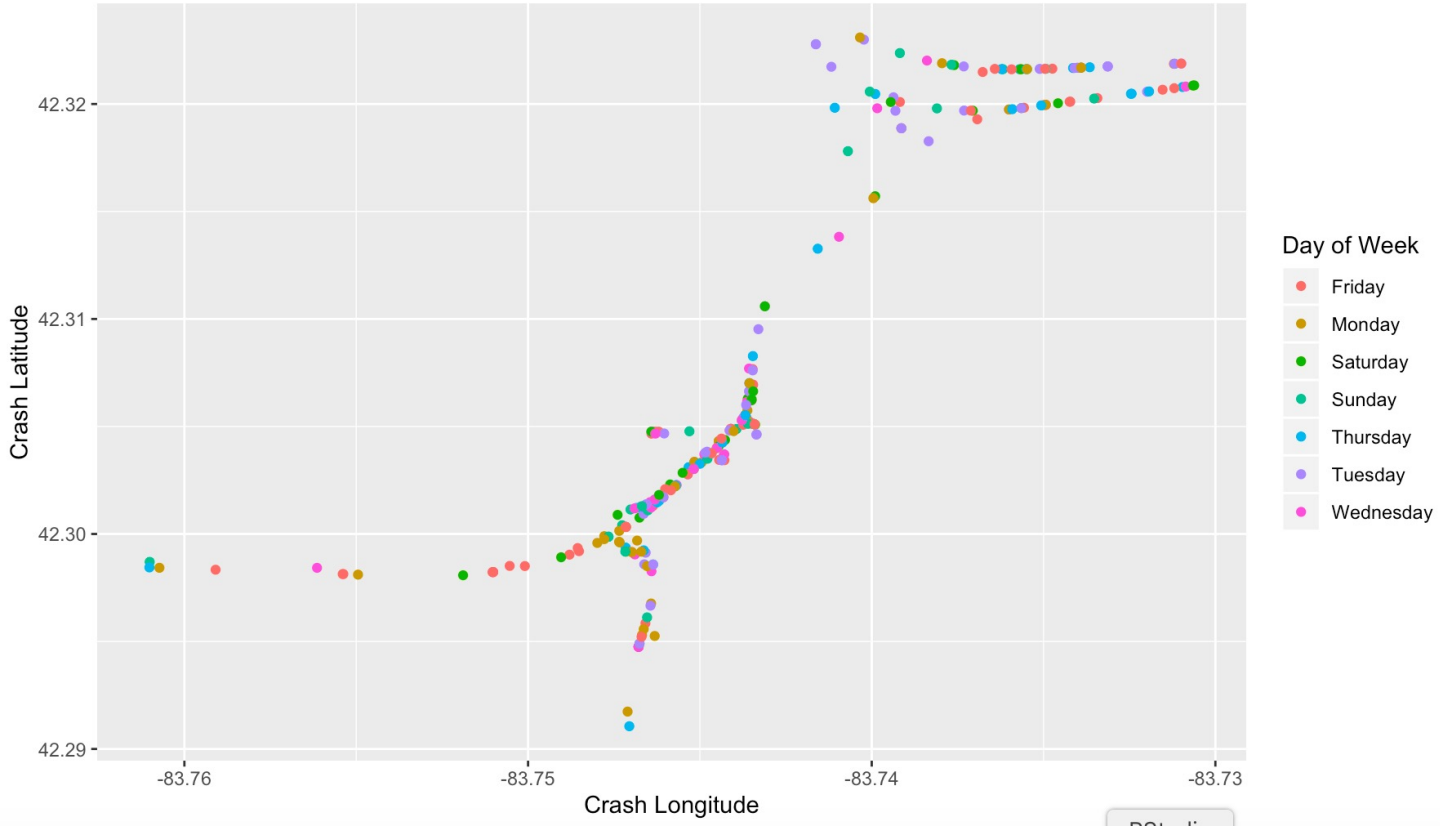
Collisions by Coordinates and Time of Day (size)

ggplot(data=aacrash) + geom_point(mapping = aes(x=`Crash Longitude`, y=`Crash Latitude`, size = `Time of Day`), alpha = 0.5)



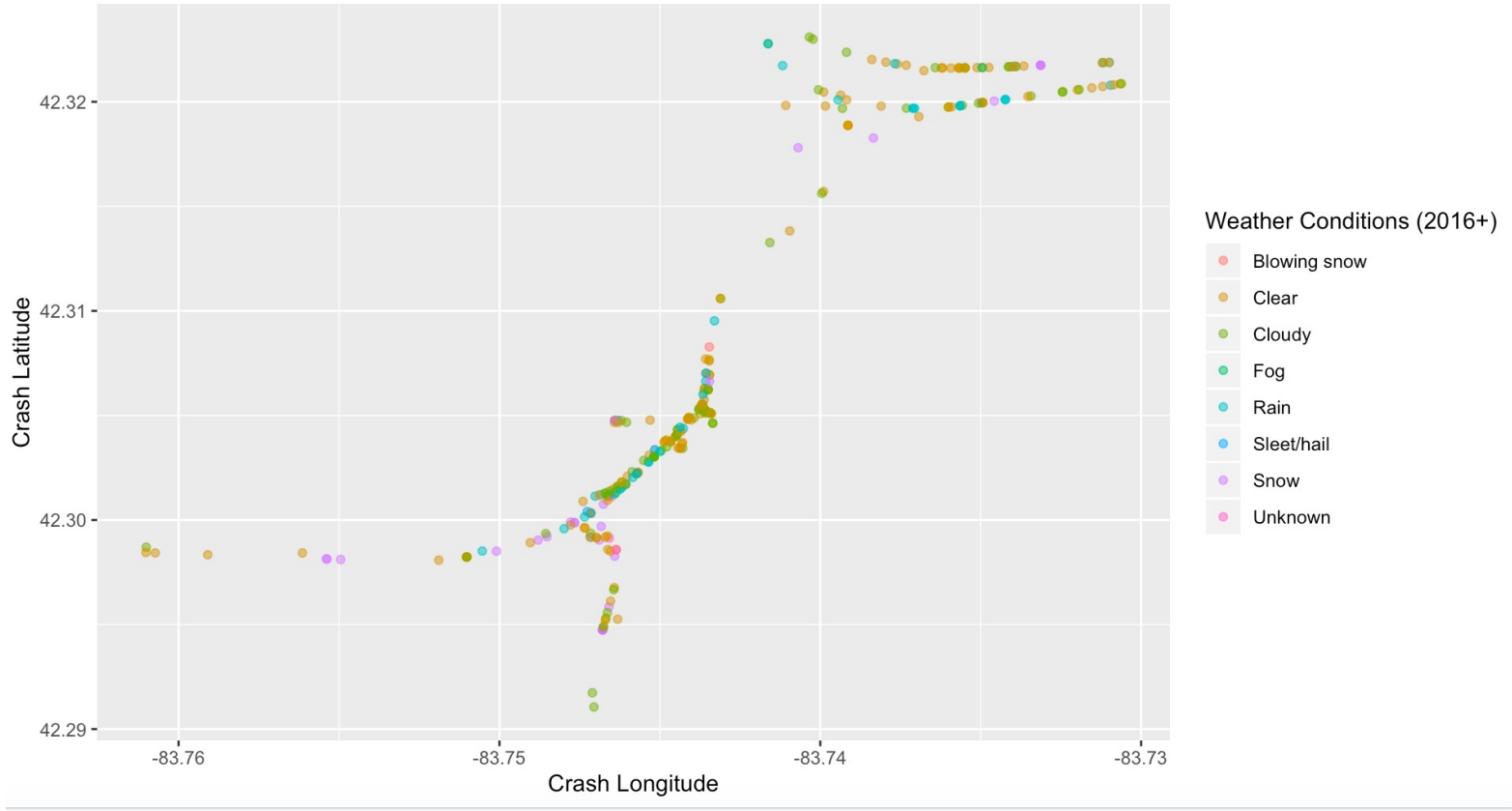
Collisions by Coordinates and Day of Week (color)

ggplot(data=aacrash) + geom_point(mapping = aes(x=`Crash Longitude`, y=`Crash Latitude`, color = `Day of Week`))



Collisions by Weather Condition (color + alpha)

ggplot(data=aacrash) + geom_point(mapping = aes(x=`Crash Longitude`, y=`Crash Latitude`, color = `Weather Conditions (2016+)`), alpha = .5))



Data Visualization with ggplot2 : : CHEAT SHEET

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.

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

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

qplot(x = cty, y = hwy, data = mpg, geom = "point") Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

last_plot() Returns the last plot

ggsave("plot.png", width = 5, height = 5) Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

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=1) - 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()
x, y, alpha, color, fill

c + geom_freqpoly() x, y, alpha, color, group, linetype, size

c + geom_histogram(binwidth = 5) x, y, alpha, color, fill, linetype, size, weight

c2 + geom_qq(aes(sample = hwy)) x, y, alpha, color, fill, linetype, size, weight

discrete

```
d <- ggplot(mpg, aes(fl))
```

d + geom_bar()
x, alpha, color, fill, linetype, size, weight

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

discrete x , discrete y

```
g <- ggplot(diamonds, aes(cut, color))
```

g + geom_count(), x, y, alpha, color, fill, shape, size, stroke

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

```
data <- data.frame(murder = USArrests$Murder,
state = tolower(rownames(USArrests)))
map <- map_data("state")
k <- ggplot(data, aes(fill = murder))
```

k + geom_map(aes(map_id = state), map = map) **+ expand_limits**(x = map$long, y = map$lat), map_id, alpha, color, fill, linetype, size

THREE VARIABLES

```
seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2)); l <- ggplot(seals, aes(long, lat))
```

l + geom_contour(aes(z = z))
x, y, z, alpha, colour, group, linetype, size, weight

l + geom_raster(aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE)
x, y, alpha, fill

l + geom_tile(aes(fill = z)), x, y, alpha, color, fill, linetype, size, width

Stats An alternative way to build a layer

A stat builds new variables to plot (e.g., count, prop).

Visualize a stat by changing the default stat of a geom function, **geom_bar(stat="count")** or by using a stat function, **stat_count(geom="bar")**, which calls a default geom to make a layer (equivalent to a geom function). Use **..name..** syntax to map stat variables to aesthetics.

geom to use | stat function | geommappings

i + stat_density2d(aes(fill = ..level..), geom = "polygon")

variable created by stat

c + stat_bin(binwidth = 1, origin = 10)
x, y | ..count.., ..ncount.., ..density.., ..ndensity..

c + stat_count(width = 1) **x, y,** | ..count.., ..prop..

c + stat_density(adjust = 1, kernel = "gaussian")
x, y, | ..count.., ..density.., ..scaled..

e + stat_bin_2d(bins = 30, drop = T)
x, y, fill | ..count.., ..density..

e + stat_bin_hex(bins=30) **x, y, fill** | ..count.., ..density..

e + stat_density_2d(contour = TRUE, n = 100)
x, y, color, size | ..level..

e + stat_ellipse(level = 0.95, segments = 51, type = "t")

l + stat_contour(aes(z = z)) **x, y, z, order** | ..level..

l + stat_summary_hex(aes(z = z), bins = 30, fun = max)
x, y, z, fill | ..value..

l + stat_summary_2d(aes(z = z), bins = 30, fun = mean)
x, y, z, fill | ..value..

f + stat_boxplot(coef = 1.5) **x, y** | ..lower.., ..middle.., ..upper.., ..width.. , ..ymin.., ..ymax..

f + stat_ydensity(kernel = "gaussian", scale = "area") **x, y** | ..density.., ..scaled.., ..count.., ..n.., ..violinwidth.., ..width..

e + stat_ecdf(n = 40) **x, y** | ..x.., ..y..

e + stat_quantile(quantiles = c(0.1, 0.9), formula = y ~ log(x), method = "rq") **x, y** | ..quantile..

e + stat_smooth(method = "lm", formula = y ~ x, se=T, level=0.95) **x, y** | ..se.., ..x.., ..y.., ..ymin.., ..ymax..

ggplot() + stat_function(aes(x = -3:3), n = 99, fun = dnorm, args = list(sd=0.5)) **x** | ..x.., ..y..

e + stat_identity(na.rm = TRUE)

ggplot() + stat_qq(aes(sample=1:100), dist = qt, dparam=list(df=5)) **sample, x, y** | ..sample.., ..theoretical..

e + stat_sum() x, y, size | ..n.., ..prop..

e + stat_summary(fun.data = "mean_cl_boot")

h + stat_summary_bin(fun.y = "mean", geom = "bar")

e + stat_unique()

Scales

Scales map data values to the visual values of an aesthetic. To change a mapping, add a new scale.

(n <- d + geom_bar(aes(fill = fl)))

scale_ | aesthetic to adjust | prepackaged scale to use | scale-specific arguments

n + scale_fill_manual(
values = c("skyblue", "royalblue", "blue", "navy"),
limits = c("d", "e", "p", "r"), breaks =c("d", "e", "p", "r"),
name = "fuel", labels = c("D", "E", "P", "R"))

range of values to include in mapping | title to use in legend/axis | labels to use in legend/axis | breaks to use in legend/axis

GENERAL PURPOSE SCALES

Use with most aesthetics

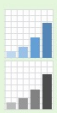
scale_*_continuous() - map cont' values to visual ones
scale_*_discrete() - map discrete values to visual ones
scale_*_identity() - use data values **as** visual ones
scale_*_manual(values = c()) - map discrete values to manually chosen visual ones
scale_*_date(date_labels = "%m/%d"), date_breaks = "2 weeks") - treat data values as dates.
scale_*_datetime() - treat data x values as date times. Use same arguments as scale_x_date(). See ?strptime for label formats.

X & Y LOCATION SCALES

Use with x or y aesthetics (x shown here)

scale_x_log10() - Plot x on log10 scale
scale_x_reverse() - Reverse direction of x axis
scale_x_sqrt() - Plot x on square root scale

COLOR AND FILL SCALES (DISCRETE)

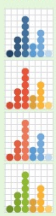


n <- d + geom_bar(aes(fill = fl))

n + scale_fill_brewer(palette = "Blues")
For palette choices:
RColorBrewer::display.brewer.all()

n + scale_fill_grey(start = 0.2, end = 0.8, na.value = "red")

COLOR AND FILL SCALES (CONTINUOUS)



o <- c + geom_dotplot(aes(fill = ..x..))

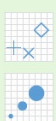
o + scale_fill_distiller(palette = "Blues")

o + scale_fill_gradient(low="red", high="yellow")

o + scale_fill_gradient2(low="red", high="blue", mid = "white", midpoint = 25)

o + scale_fill_gradientn(colours=topo.colors(6))
Also: rainbow(), heat.colors(), terrain.colors(), cm.colors(), RColorBrewer::brewer.pal()

SHAPE AND SIZE SCALES



p <- e + geom_point(aes(shape = fl, size = cyl))
p + scale_shape() + scale_size()
p + scale_shape_manual(values = c(3:7))

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

p + scale_radius(range = c(1,6))
p + scale_size_area(max_size = 6)

Coordinate Systems

r <- d + geom_bar()

r + coord_cartesian(xlim = c(0, 5))
xlim, ylim
The default cartesian coordinate system

r + coord_fixed(ratio = 1/2)
ratio, xlim, ylim
Cartesian coordinates with fixed aspect ratio between x and y units

r + coord_flip()
xlim, ylim
Flipped Cartesian coordinates

r + coord_polar(theta = "x", direction=1)
theta, start, direction
Polar coordinates

r + coord_trans(ytrans = "sqrt")
xtrans, ytrans, limx, limy
Transformed cartesian coordinates. Set xtrans and ytrans to the name of a window function.

π + coord_quickmap()
π + coord_map(projection = "ortho", orientation=c(41, -74, 0))projection, xlim, ylim
Map projections from the mapproj package (mercator (default), azequalarea, lagrange, etc.)

Position Adjustments

Position adjustments determine how to arrange geoms that would otherwise occupy the same space.

s <- ggplot(mpg, aes(fl, fill = drv))

s + geom_bar(position = "dodge")
Arrange elements side by side

s + geom_bar(position = "fill")
Stack elements on top of one another, normalize height

e + geom_point(position = "jitter")
Add random noise to X and Y position of each element to avoid overplotting

e + geom_label(position = "nudge")
Nudge labels away from points

s + geom_bar(position = "stack")
Stack elements on top of one another

Each position adjustment can be recast as a function with manual **width** and **height** arguments
s + geom_bar(position = position_dodge(width = 1))

Themes

r + theme_bw()
White background with grid lines

r + theme_gray()
Grey background (default theme)

r + theme_dark()
dark for contrast

r + theme_classic()
r + theme_light()
r + theme_linedraw()
r + theme_minimal()
Minimal themes
r + theme_void()
Empty theme

Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables.

t <- ggplot(mpg, aes(cty, hwy)) + geom_point()

t + facet_grid(cols = vars(fl))
facet into columns based on fl

t + facet_grid(rows = vars(year))
facet into rows based on year

t + facet_grid(rows = vars(year), cols = vars(fl))
facet into both rows and columns

t + facet_wrap(vars(fl))
wrap facets into a rectangular layout

Set **scales** to let axis limits vary across facets

t + facet_grid(rows = vars(drv), cols = vars(fl), scales = "free")
x and y axis limits adjust to individual facets
"free_x" - x axis limits adjust
"free_y" - y axis limits adjust

Set **labeller** to adjust facet labels

t + facet_grid(cols = vars(fl), labeller = label_both)

fl: c	fl: d	fl: e	fl: p	fl: r

t + facet_grid(rows = vars(fl), labeller = label_bquote(alpha ^ .(fl)))

α^{c}	α^{d}	α^{e}	α^{p}	α^{r}

Labels

t + labs(**x** = "New x axis label", **y** = "New y axis label",
title ="Add a title above the plot",
subtitle = "Add a subtitle below title",
caption = "Add a caption below plot",
<AES> = "New <AES> legend title")

Use scale functions to update legend labels

t + annotate(geom = "text", x = 8, y = 9, label = "A")

geom to place | manual values for geom's aesthetics

Legends

n + theme(legend.position = "bottom")
Place legend at "bottom", "top", "left", or "right"

n + guides(fill = "none")
Set legend type for each aesthetic: colorbar, legend, or none (no legend)

n + scale_fill_discrete(name = "Title", labels = c("A", "B", "C", "D", "E"))
Set legend title and labels with a scale function.

Zooming

Without clipping (preferred)
t + coord_cartesian(
xlim = c(0, 100), ylim = c(10, 20))

With clipping (removes unseen data points)
t + xlim(0, 100) **+ ylim**(10, 20)
t + scale_x_continuous(limits = c(0, 100)) +
scale_y_continuous(limits = c(0, 100))

RStudio

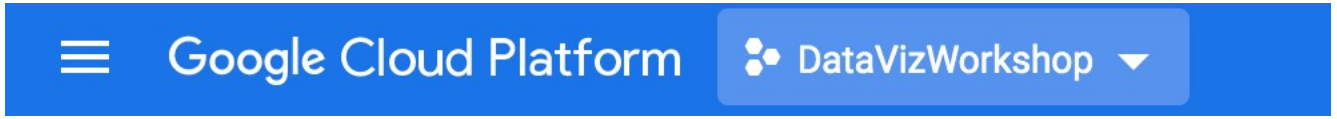
Data Exploration: Exercises

Using AA car collision data set, choose 2-3 other variables, other aesthetics (shape, fill, etc.) to plot as:

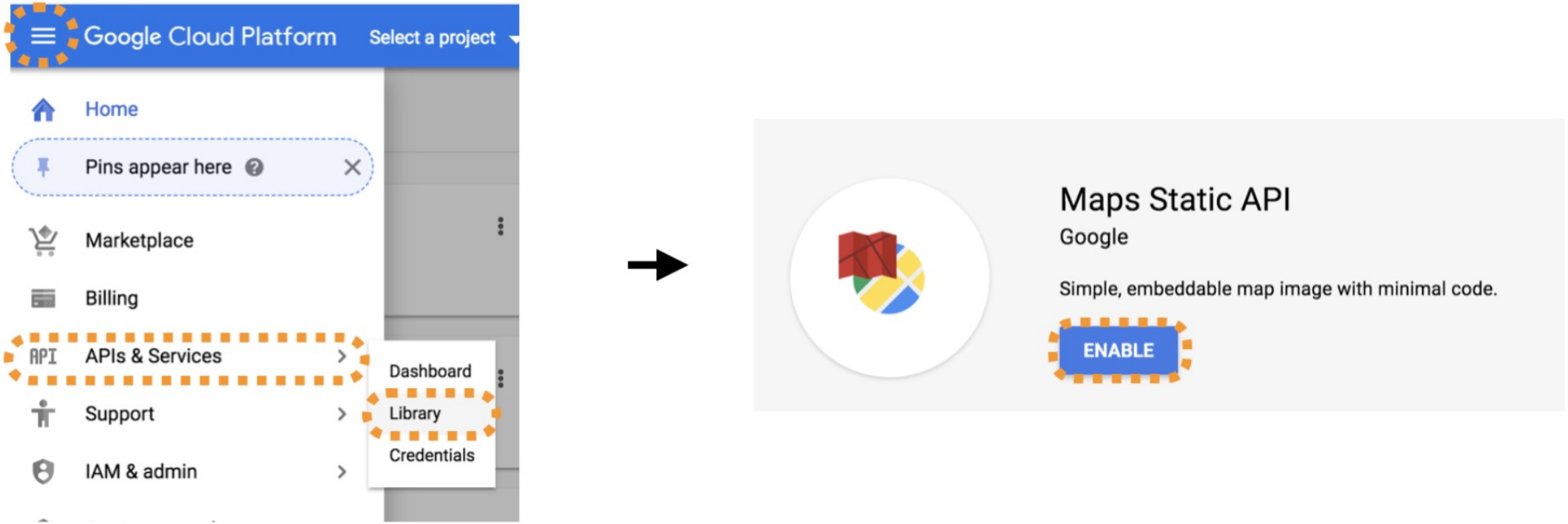
1) Bar chart
2) Scatterplot
3) Your choice!

Setting up ggmap

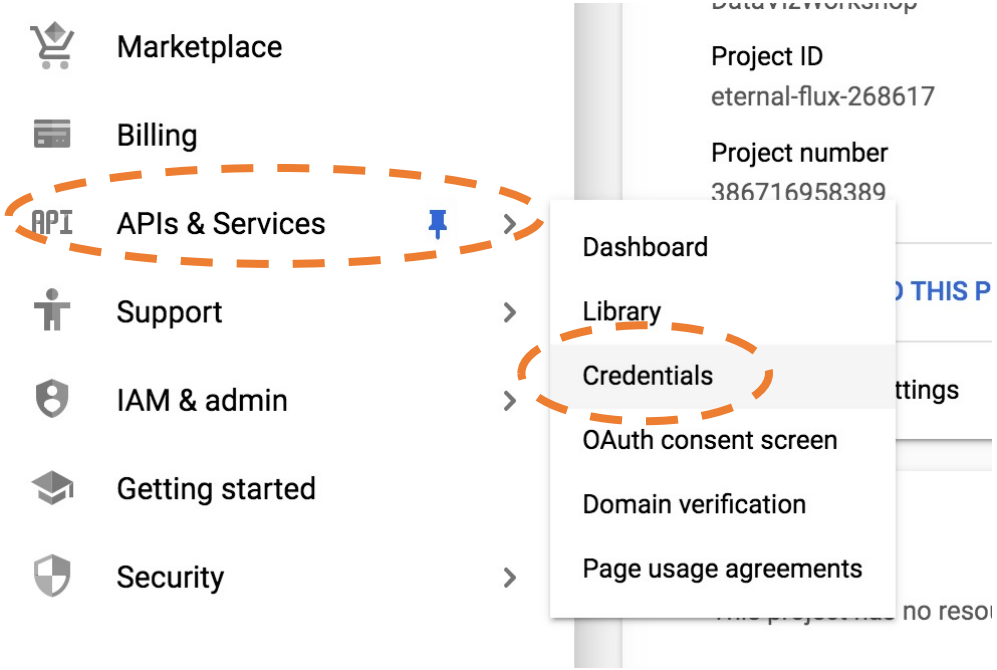
- Visit - https://console.cloud.google.com and sign up for a Google Cloud Platform trial.
- Create a project (top menu)



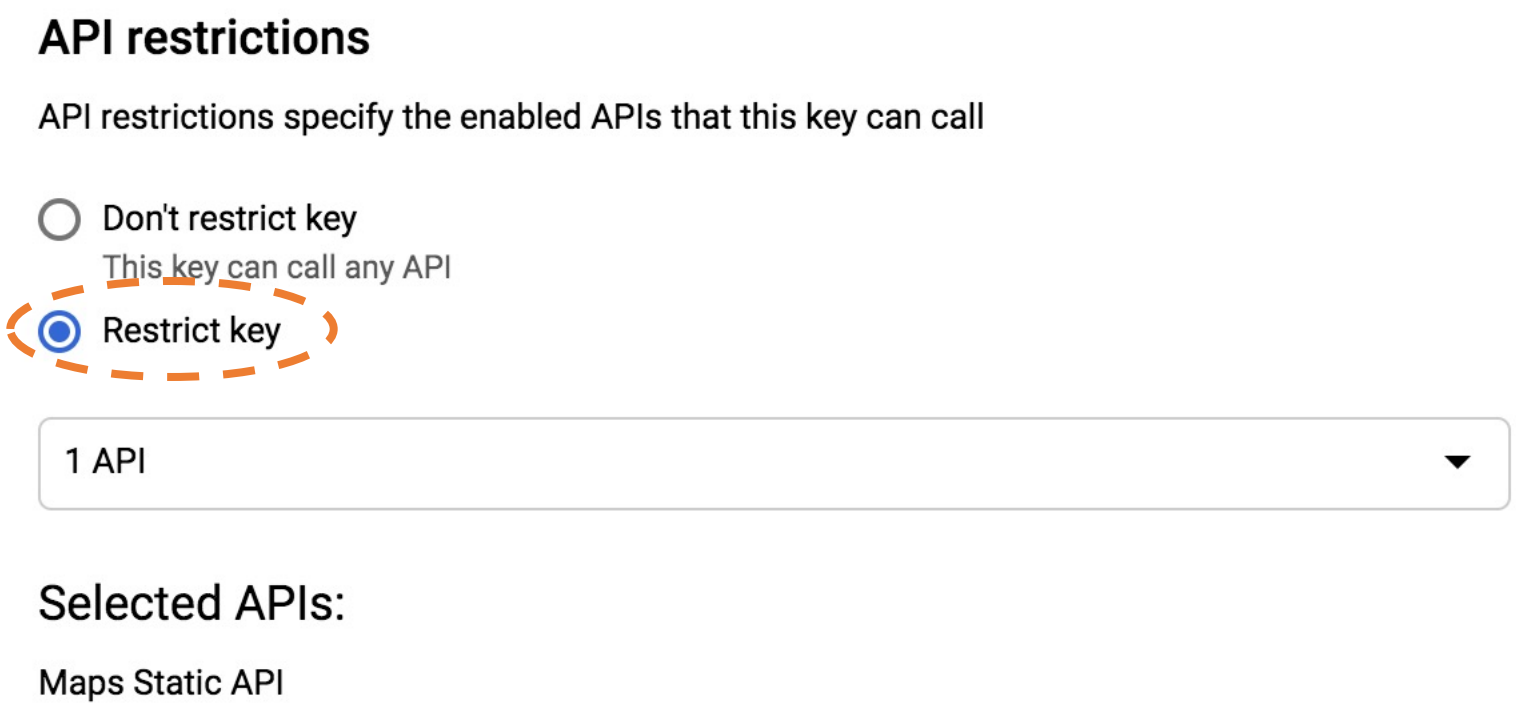
- Add the “Maps Static API” service to the project. Navigate to the library of API services and search for “Maps Static API”. **Enable the service.**



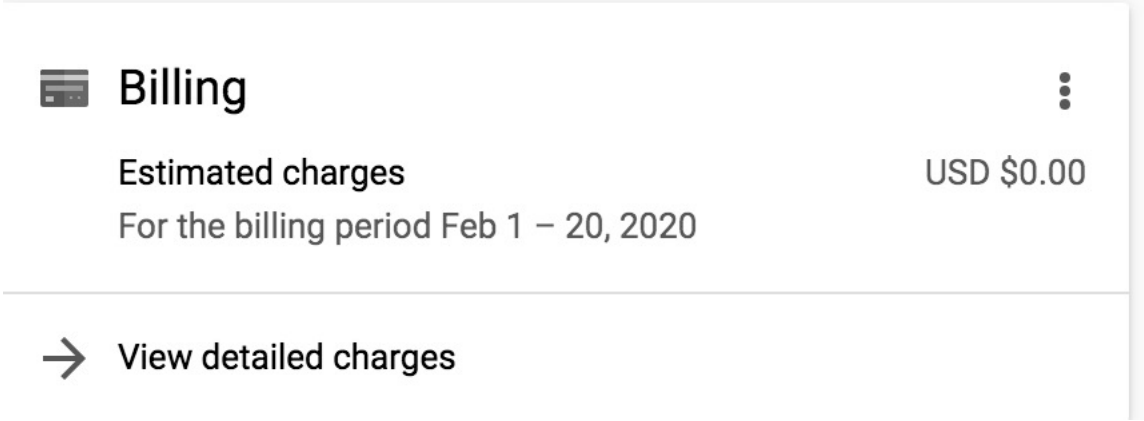
- Generate an API Key. Navigate to the credentials area and select “Create credentials”. Take note of your API key. You will need to register it with the package later.



Under API restrictions, make sure you click the “Restrict key” button for your Maps Static API:



Go the Home Page and click on the Billing menu. **Make sure your project is linked to your billing account.**



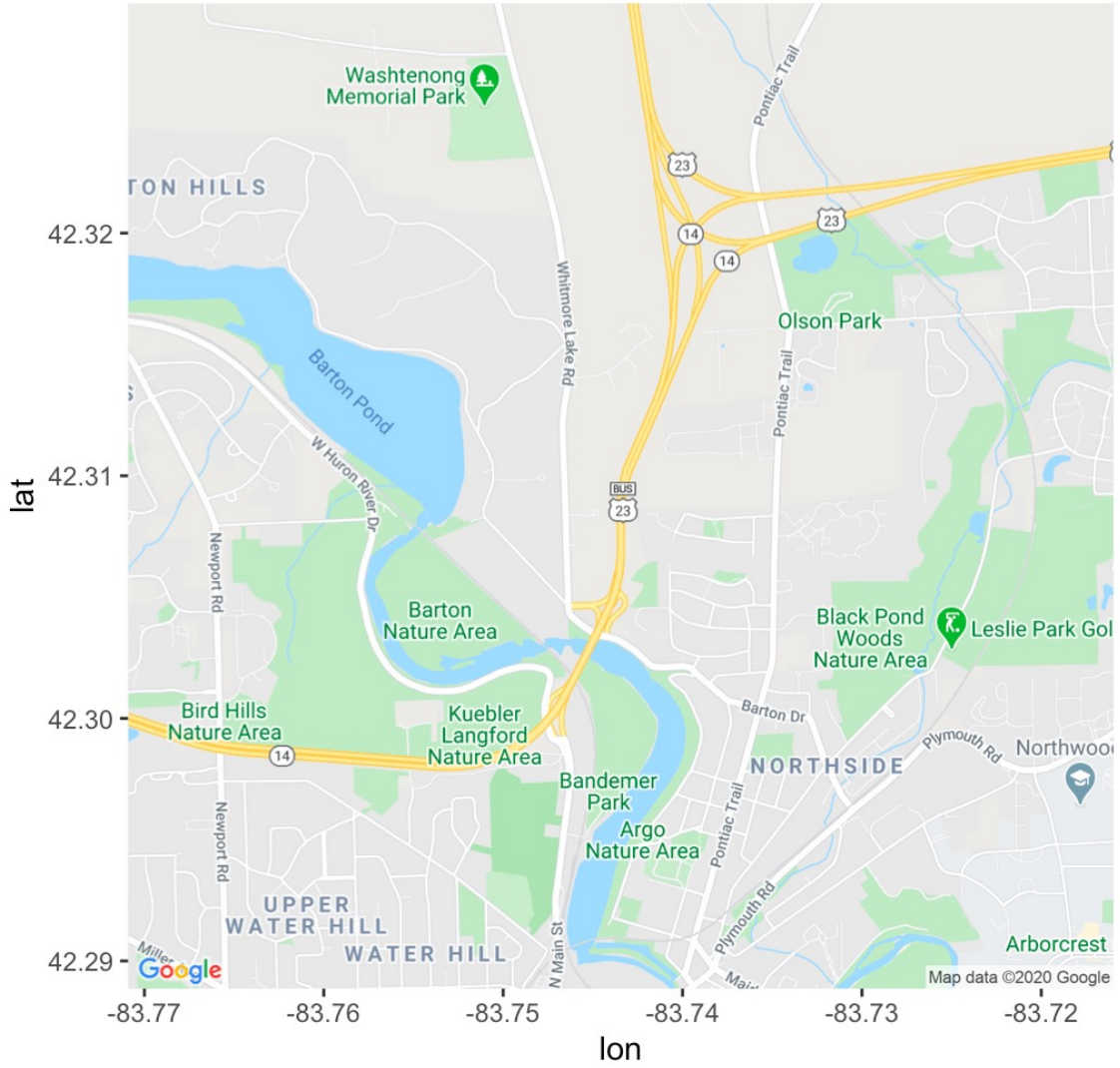
Register your API key

In R, type:

```
ggmap::register_google(key = "SET YOUR KEY HERE")
```

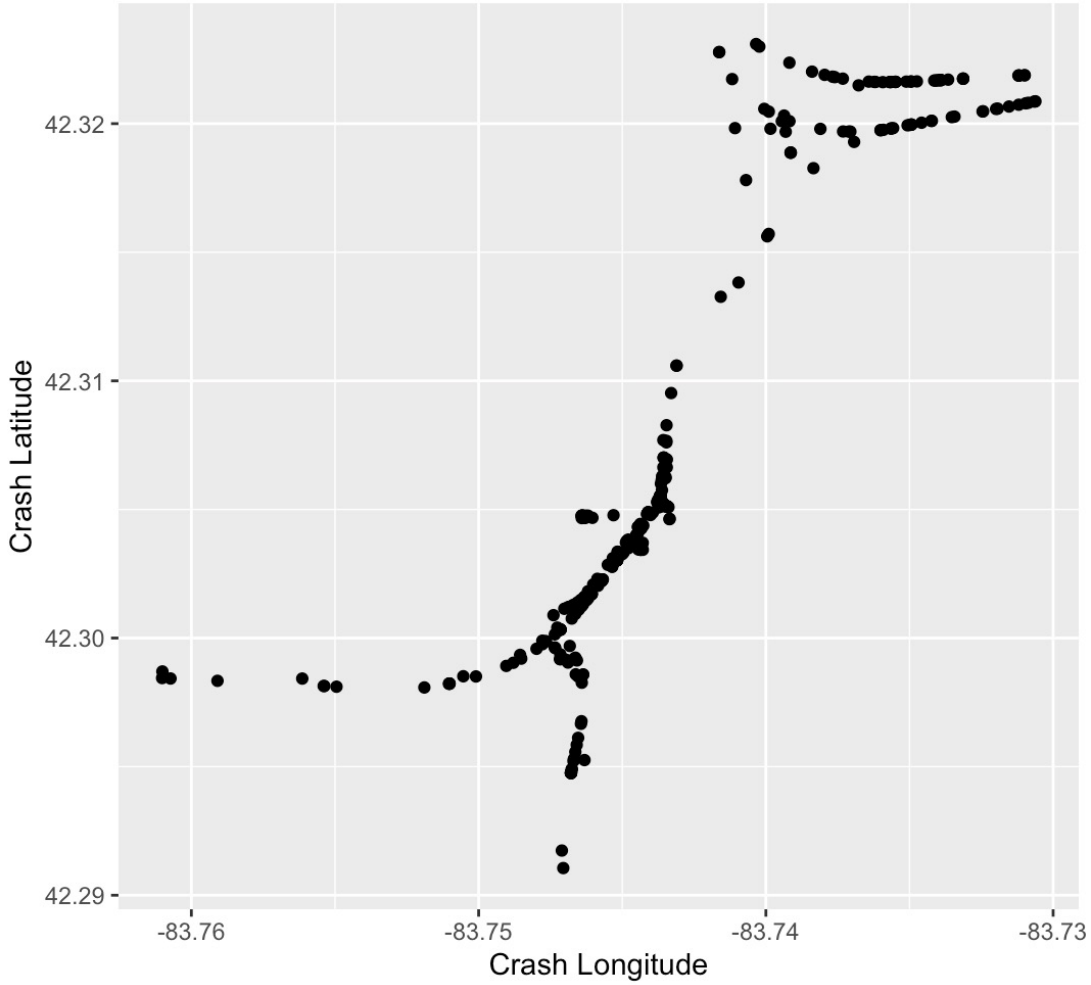
Ggmap: First, let's create our Ann Arbor map

```
aamap <-get_googlemap(center =
c(lon = -83.743485, lat = 42.309204),
zoom = 14, scale = 2, maptype =
"roadmap", color='color')

# zoom: smaller number towards
continent level, larger number
towards building level
```



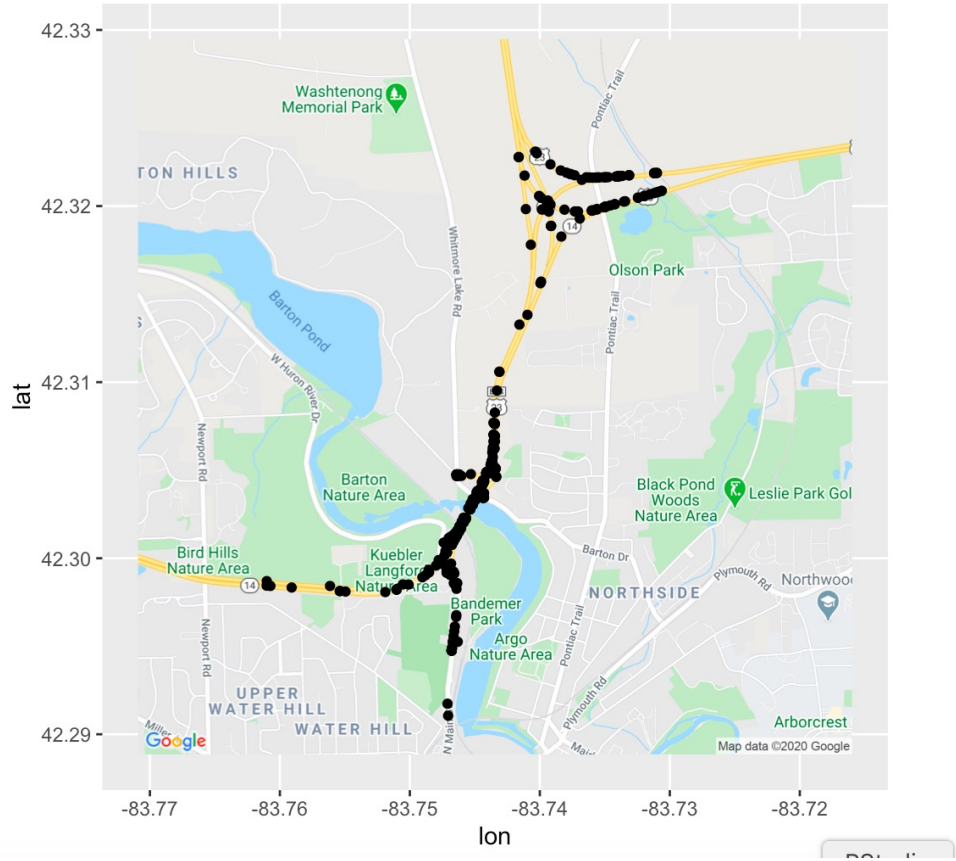
Next, create coordinates data from aacrash

```
ggplot(data = aacrash) +
geom_point(mapping = aes(x =
`Crash Longitude`, y = `Crash
Latitude`))
```



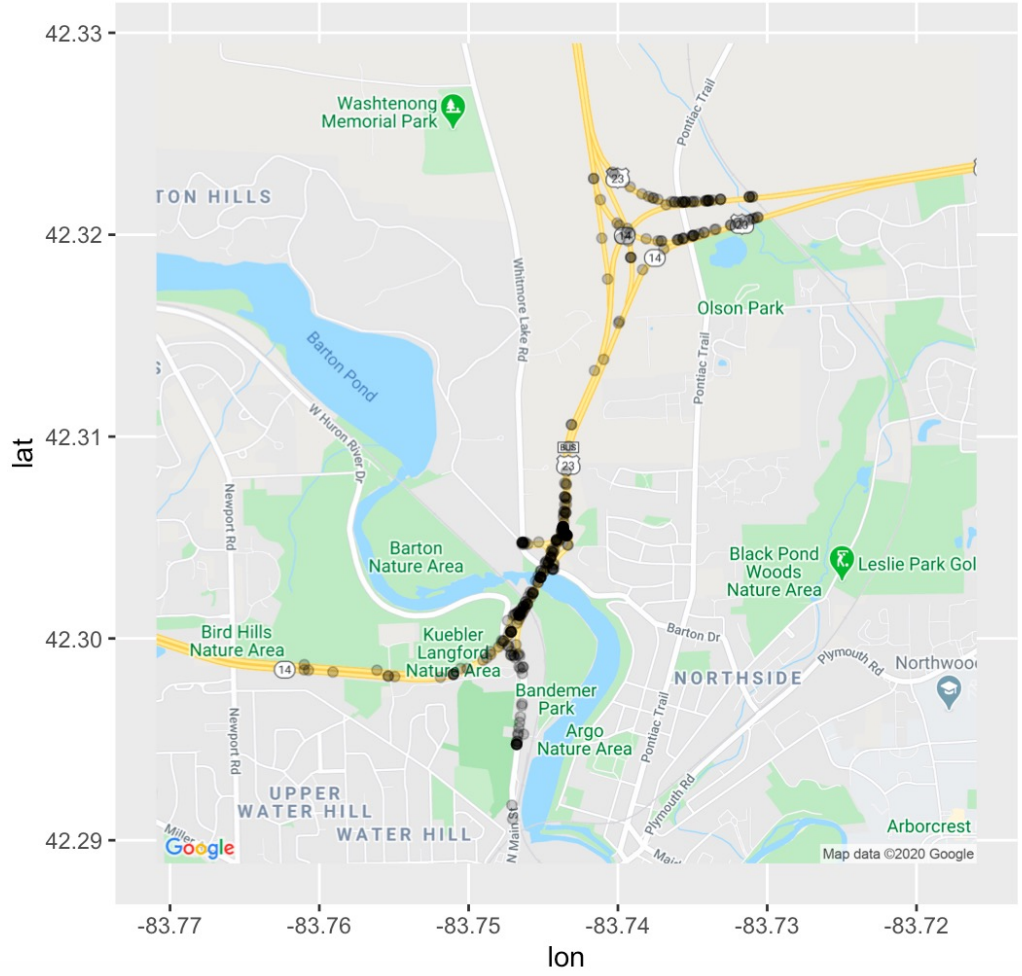
Put the plot on the map!

```
ggmap(aamap,extent = "normal") +
geom_point(aes(x=`Crash Longitude`, y=`Crash Latitude`), data=aacrash)
```



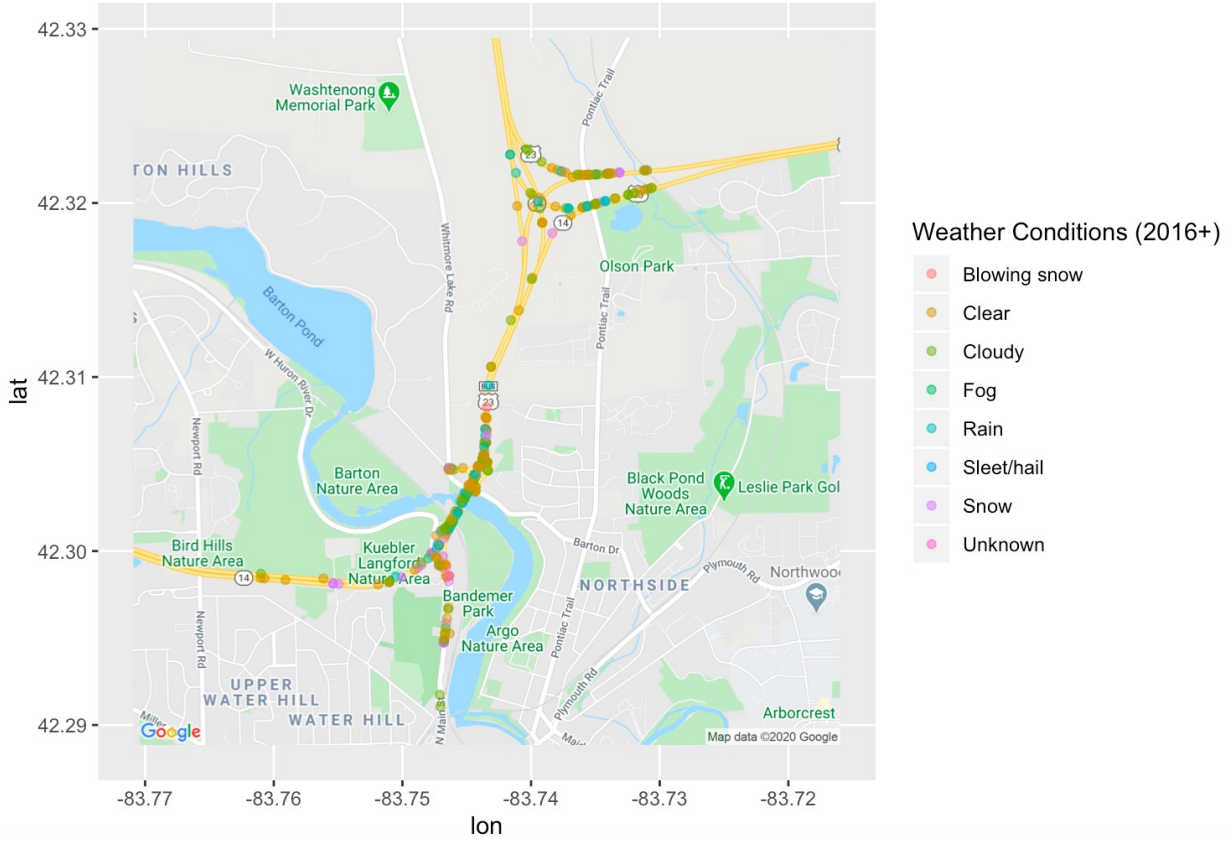
Play with the visual features

```
ggmap(aamap,extent = "normal") +
geom_point(aes(x=`Crash Longitude`, y=`Crash Latitude`), data=aacrash) , alpha = .25)
```



Play with the visual features

ggmap(aamap,extent = "normal") + geom_point(aes(x=`Crash Longitude`, y=`Crash Latitude`, color=`Weather Conditions (2016+)`),data=aacrash, alpha=.5)



Data Explanation

Goals:

1) Choose the most effective **format** for displaying the message I want to communicate
2) Tweak visual design elements (color, amount of ink, labels/legends, grouping, etc.) to make it user-friendly and pretty

Use animation sparingly, carefully

- Can be overwhelming for the visual system
- Keep it simple

Example: use gganimate to show collisions over time of day

```
#First, create code for plot
library(gganimate)
p <- ggplot(aacrash, aes(x=`Crash Longitude`, y=`Crash Latitude`))+geom_point()
plot(p)

#Add the animation code
anim <- p + transition_states(`Time of Day`, transition_length = 1, state_length=2)
> anim

#If you get an error about saving frames, try changing permission of folder to have read and write access
```

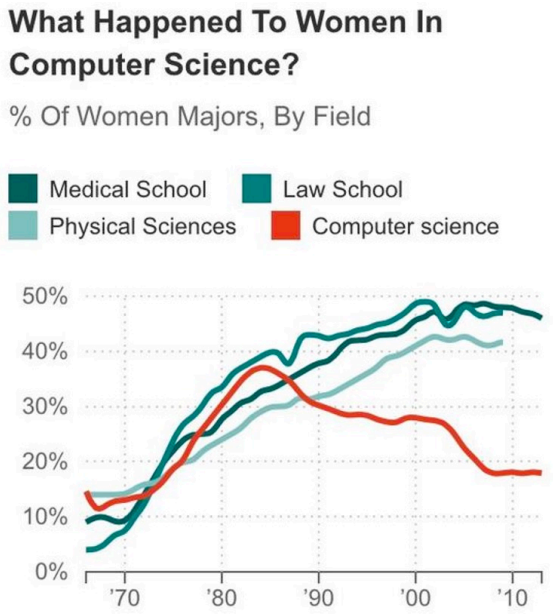
Example: use gganimate to show collisions over time of day

```
#Add a title with the time of day

anim <- p + transition_states(`Time of
Day`, transition_length = 1,
state_length=2) + labs(title = 'Hour:
{closest_state}')


# Save the animation using anim_save

anim_save("aacrash.gif",anim)
```

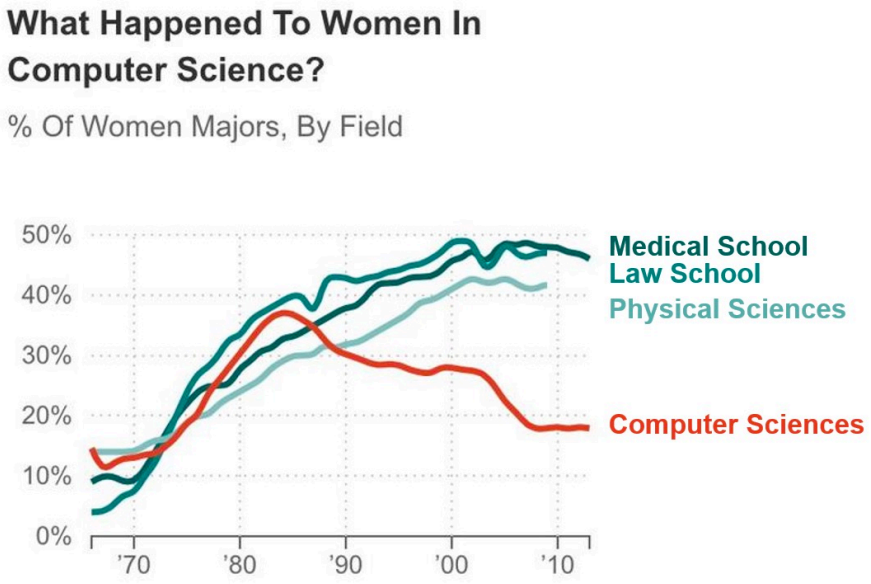
gganimate links

- https://github.com/thomasp85/gganimate/wiki
- https://github.com/ropenscilabs/learngganimate

Use color grouping and direct labels



Vs



Source: https://depictdatastudio.com/directly-labeling-line-graphs/

Use minimal gridlines

ggplot2:

theme_classic()

theme_light()

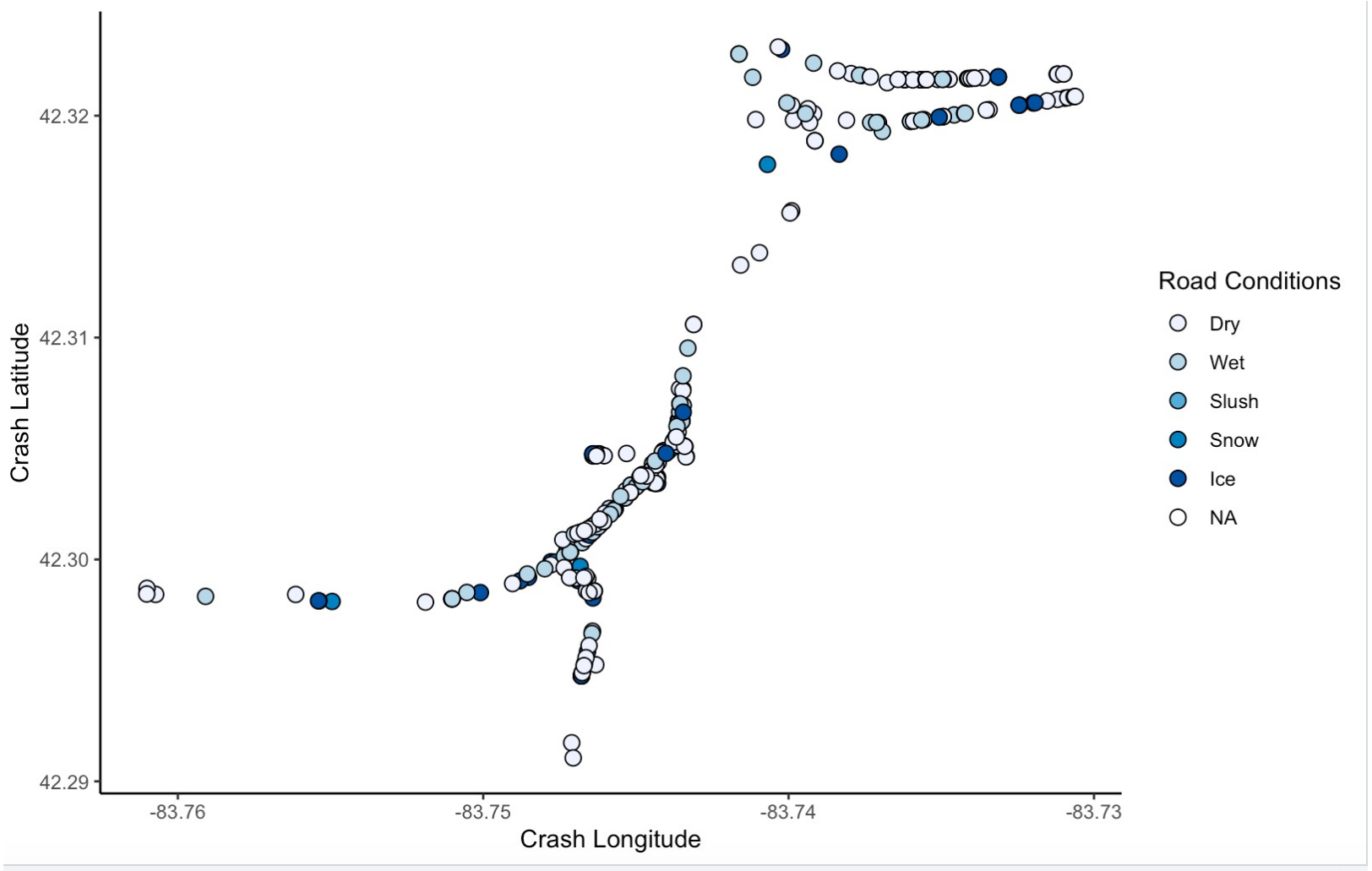
Use color scales that are meaningful, not confusing

Keep *magnitude* and *directionality* of your variables in mind when choosing colors for scale:

- ‘Big’ maps on to dark, ‘small maps’ on to light
- ‘Hot’ = red, ‘cold’ = blue
- Some nice, pretested palettes: http://colorbrewer2.org
  - scale_color_brewer
- Use outlines for colors that are light/hard to see: http://www.sthda.com/english/wiki/ggplot2-point-shapes

Data Explanation: Exercises

1) Try to recreate this visualization, showing collision locations by **road condition** with new colors that better map onto conditions



1) Code to recreate visualization

```
#First, reorder levels of road condition to be in order from Dry to Ice

aacrash$`Road Conditions` <- factor(aacrash$`Road Conditions`, levels
= c("Dry","Wet","Slush","Snow","Ice"))

#Next, plot the data using the “Blues” palette and scale_fill_brewer

ggplot(data=aacrash) + geom_point(mapping = aes(x=`Crash
Longitude`, y=`Crash Latitude`, fill=`Road Conditions`), shape=21, color
= "black", size=3) + scale_fill_brewer(palette = "Blues") +
theme_classic()
```

2) You are tasked with visualizing the mapped locations of the 2018 collisions over the course of an entire day (by hour). Give it a title (“2018 Collisions by Time of Day”) and subtitle (“Time: {time}”). Use ggmap and gganimate to create this visualization.

Solution

#Plot the map and data

```
m <- ggmap(aamap) +
geom_point(aes(x=`Crash Longitude`,
y=`Crash Latitude`), data=aacrash)
```

Animate

```
map_anim <- m + transition_states(`Time of
Day`,transition_length = 1,state_length = 2)
+ labs(title = "2018 Collisions by Time of
Day",subtitle = "Hour: {closest_state}")
```