Data Science for Data Wranglers Part 4:

The Structure of Visualizations

Data Manipulation

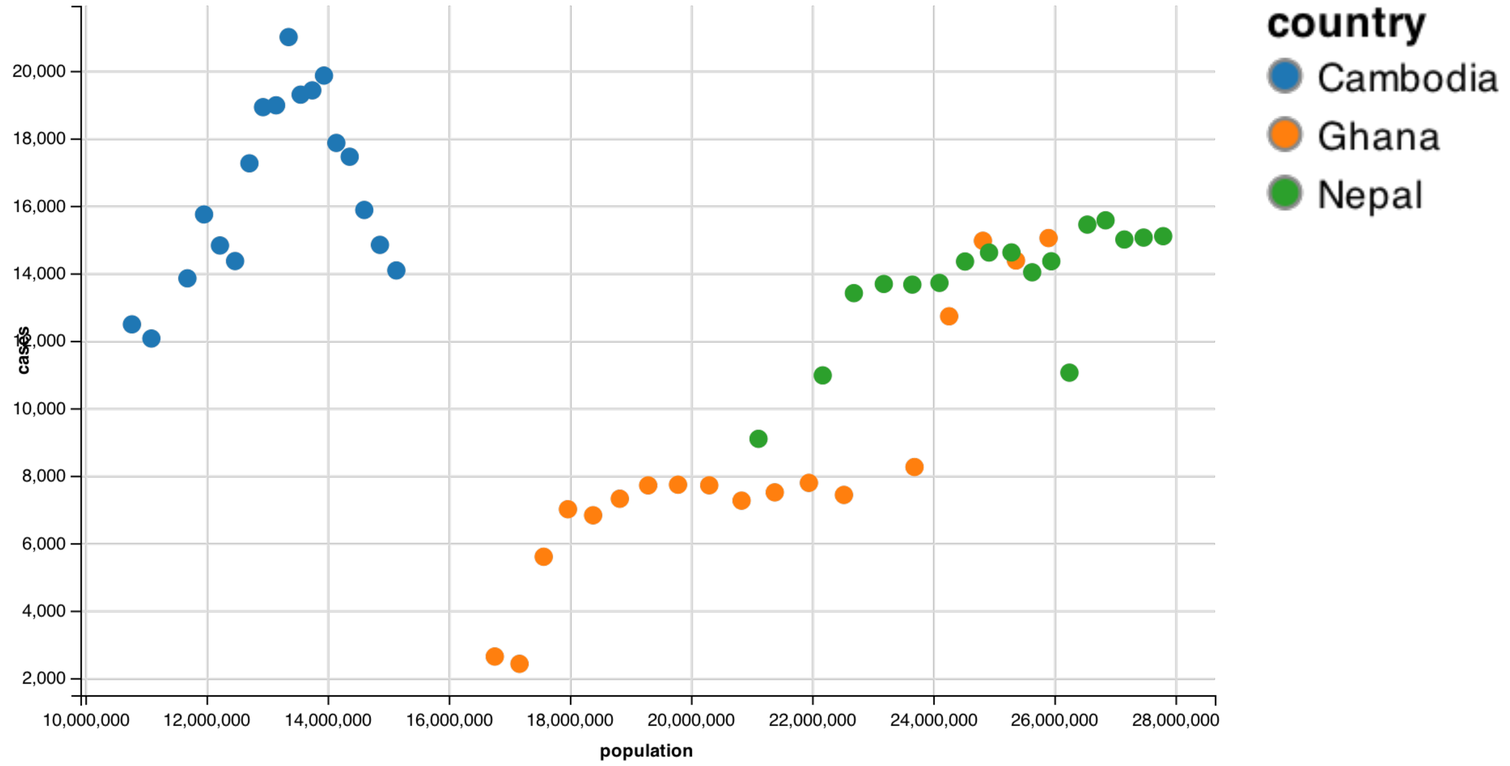
Changing the variables, values, and units of analysis contained in the data set.

Data Tidying

Changing the layout of tabular data to make it suitable for a particular piece of software (R).

Data Visualization

Transforming the data to a visual format that reveals visual patterns.



Visual Space

fill

Blue

Orange

Green

Data Space

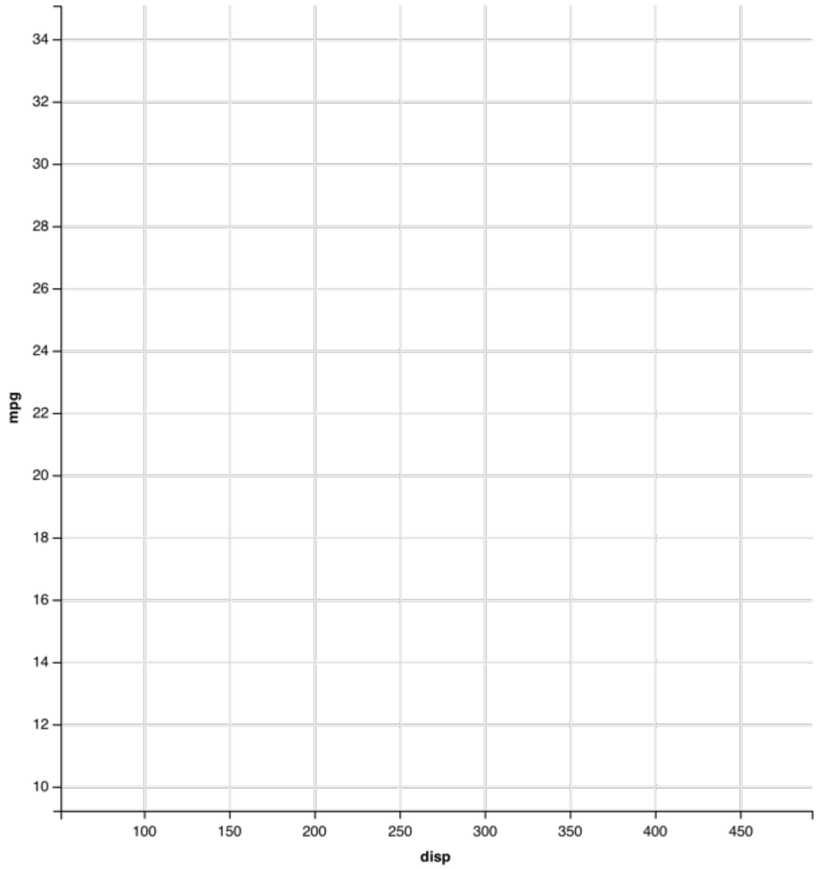
country

Cambodia

Ghana

Nepal

mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9	4	71.1	1

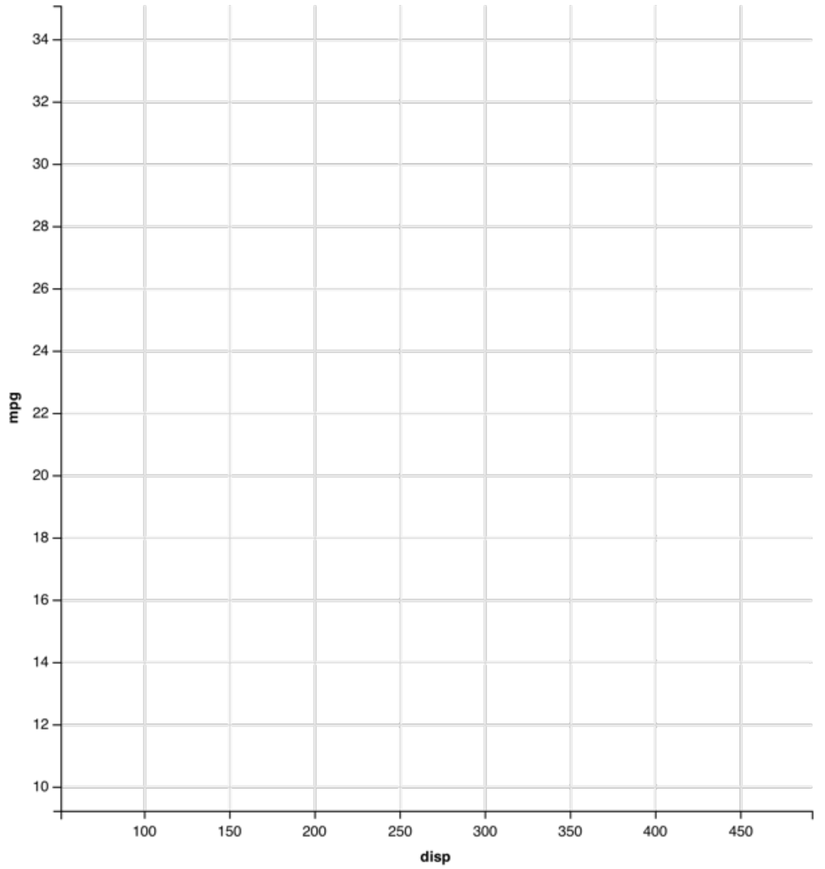


data **mark** **coordinate system**

properties

fill

mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9	4	71.1	1

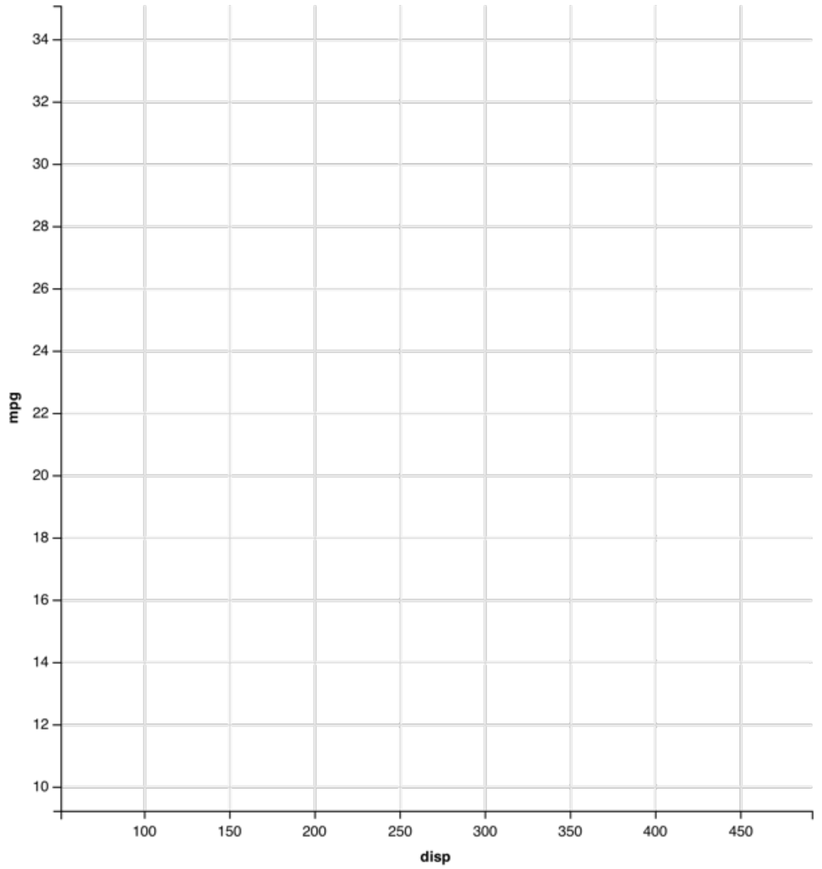


data **mark** **coordinate system**

properties

shape fill

mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9	4	71.1	1



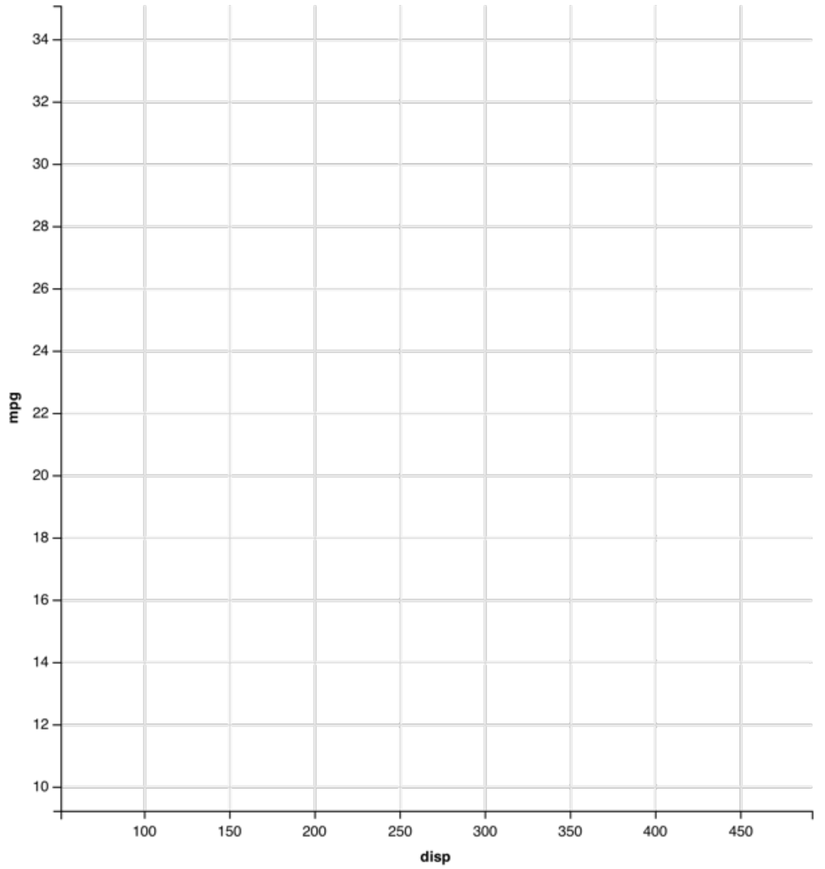
data mark

coordinate system

properties

shape x fill

mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9	4	71.1	1

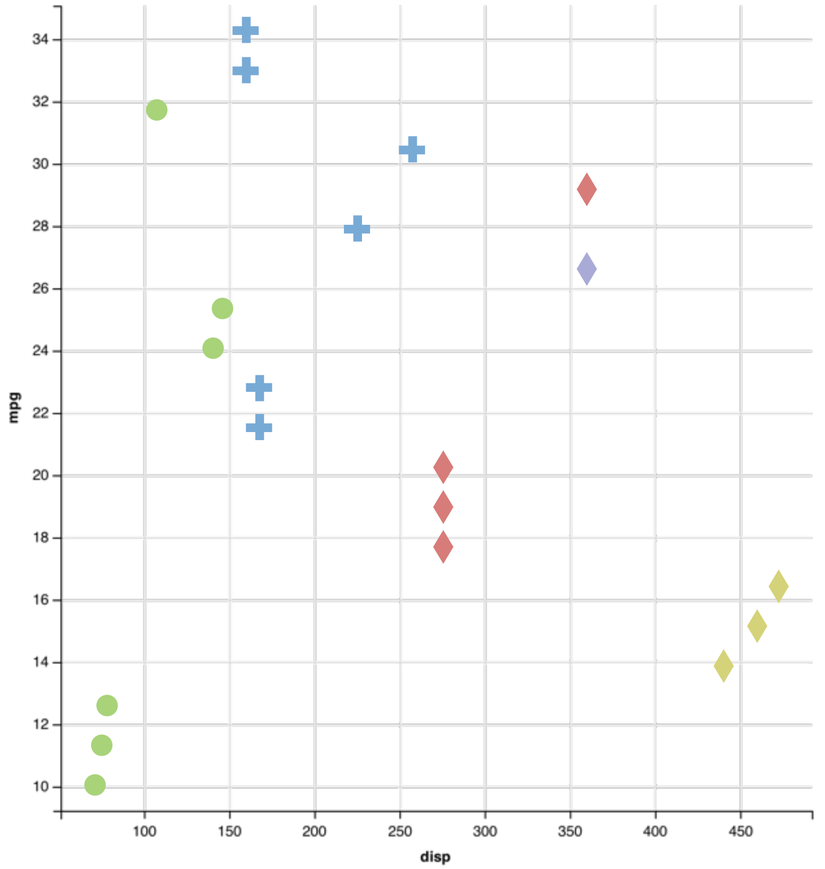


data mark

coordinate system

properties

y	shape	x	fill
mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9	4	71.1	1

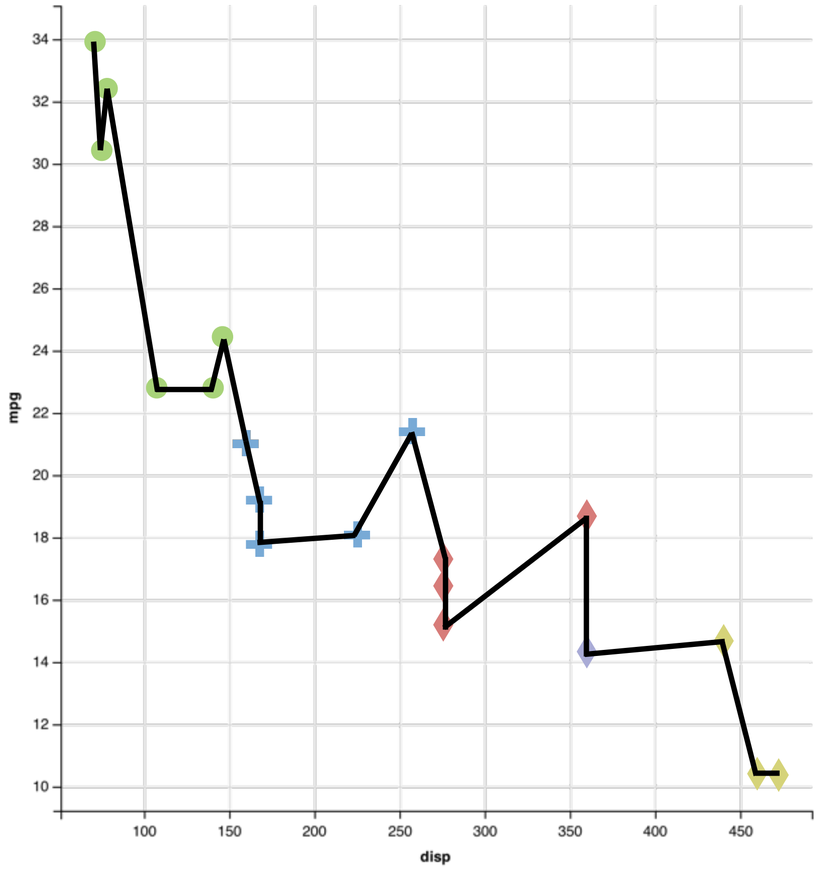


data **mark** **coordinate system**

properties

y shape x fill

mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9	4	71.1	1



data

mark

points

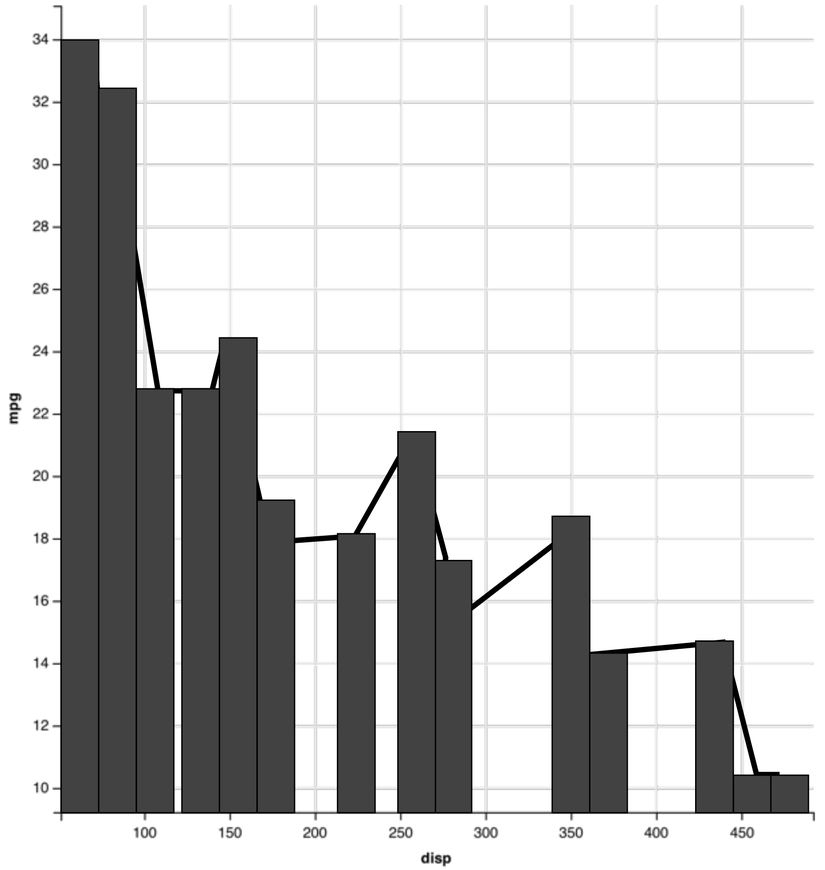
lines

coordinate system

properties

y x

mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9	4	71.1	1



data

mark

points

lines

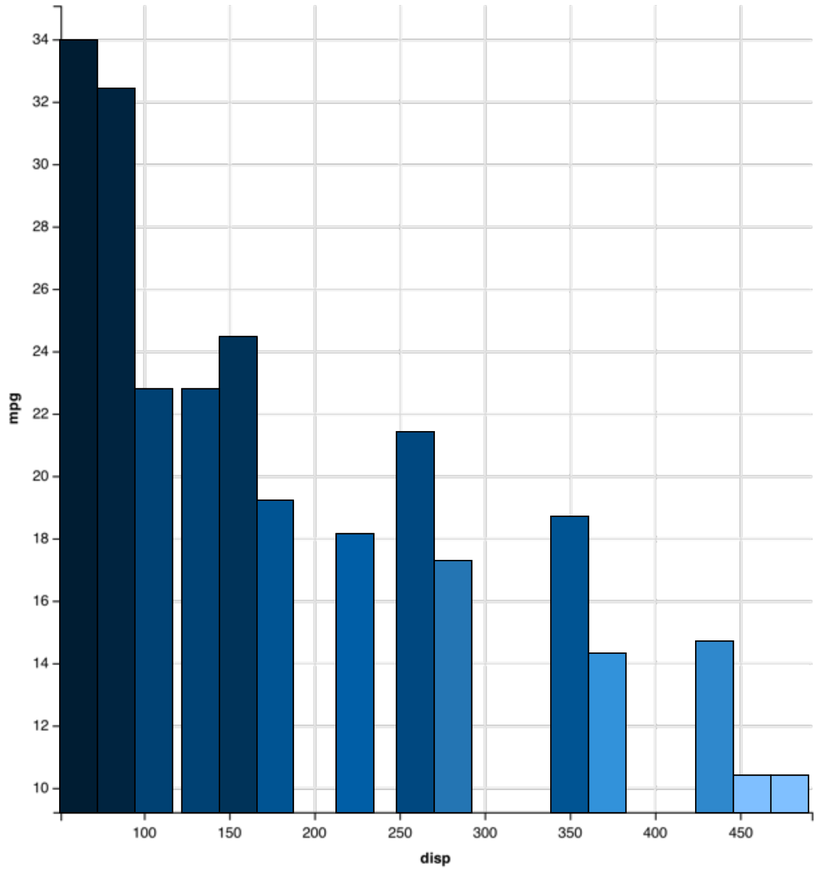
bars

coordinate system

properties

y fill

mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9	4	71.1	1



data

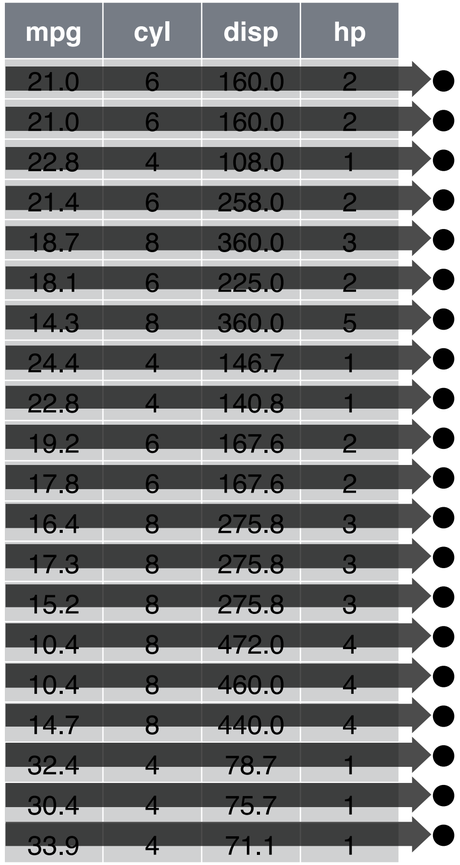
mark

points

lines

bars

coordinate system

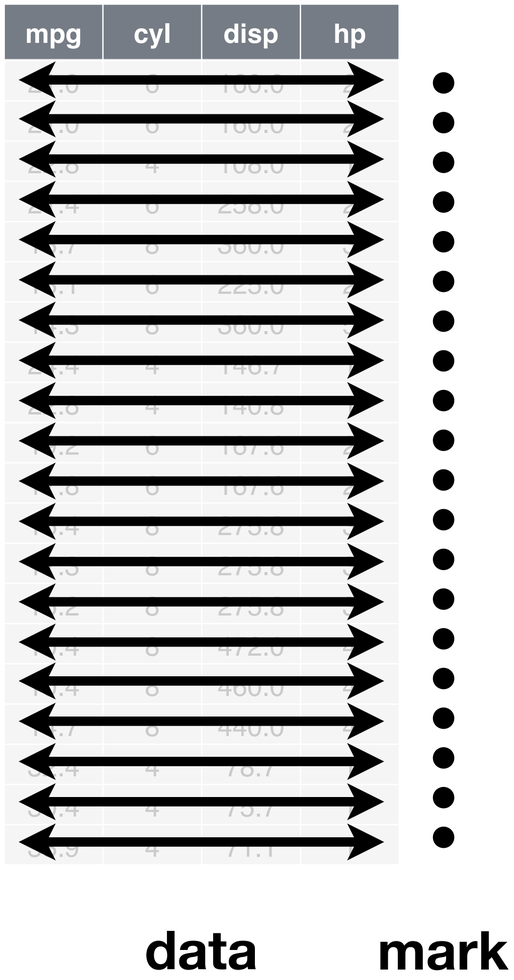


mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9	4	71.1	1

data **mark**

A visualization is a collection of

1. visual marks



A visualization is a collection of

1. visual marks (observations)

properties

fill

mpg	cyl	disp	hp
21.0	6	160.0	2
21.0	6	160.0	2
22.8	4	108.0	1
21.4	6	258.0	2
18.7	8	360.0	3
18.1	6	225.0	2
14.3	8	360.0	5
24.4	4	146.7	1
22.8	4	140.8	1
19.2	6	167.6	2
17.8	6	167.6	2
16.4	8	275.8	3
17.3	8	275.8	3
15.2	8	275.8	3
10.4	8	472.0	4
10.4	8	460.0	4
14.7	8	440.0	4
32.4	4	78.7	1
30.4	4	75.7	1
33.9	4	71.1	1

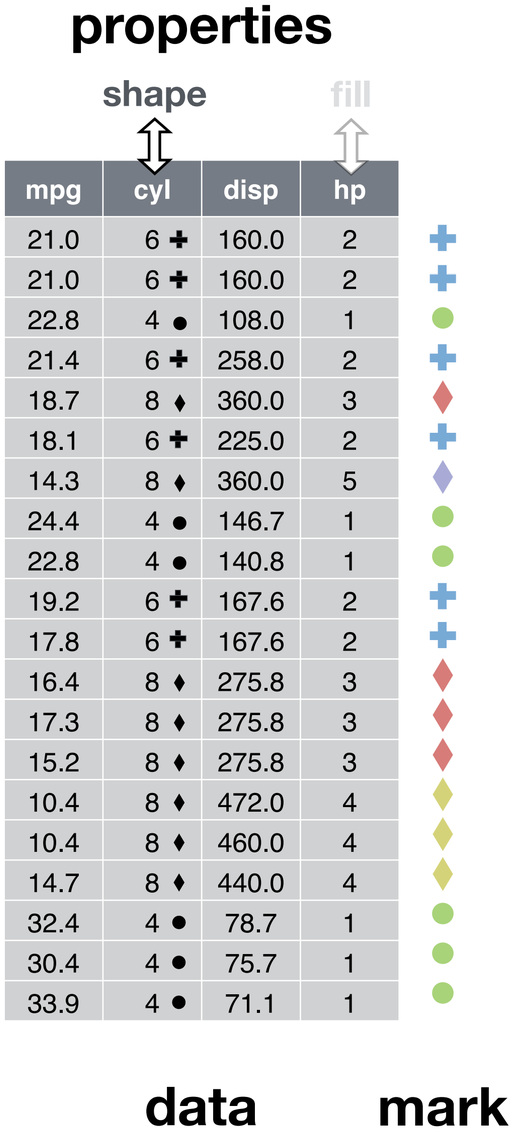
data mark

A visualization is a collection of

1. visual marks (observations)

that have

2. visual properties

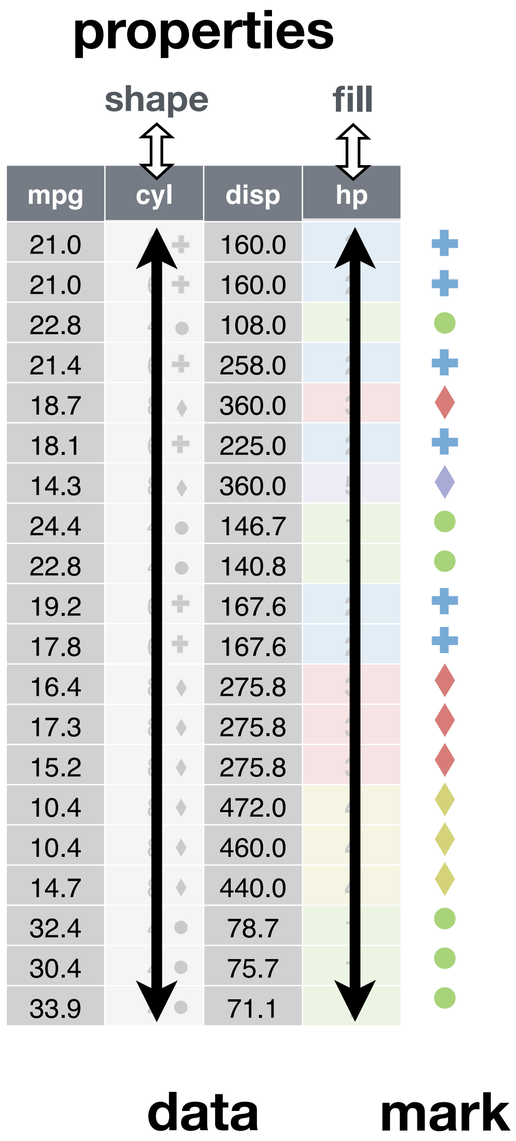


A visualization is a collection of

1. visual marks (observations)

that have

2. visual properties



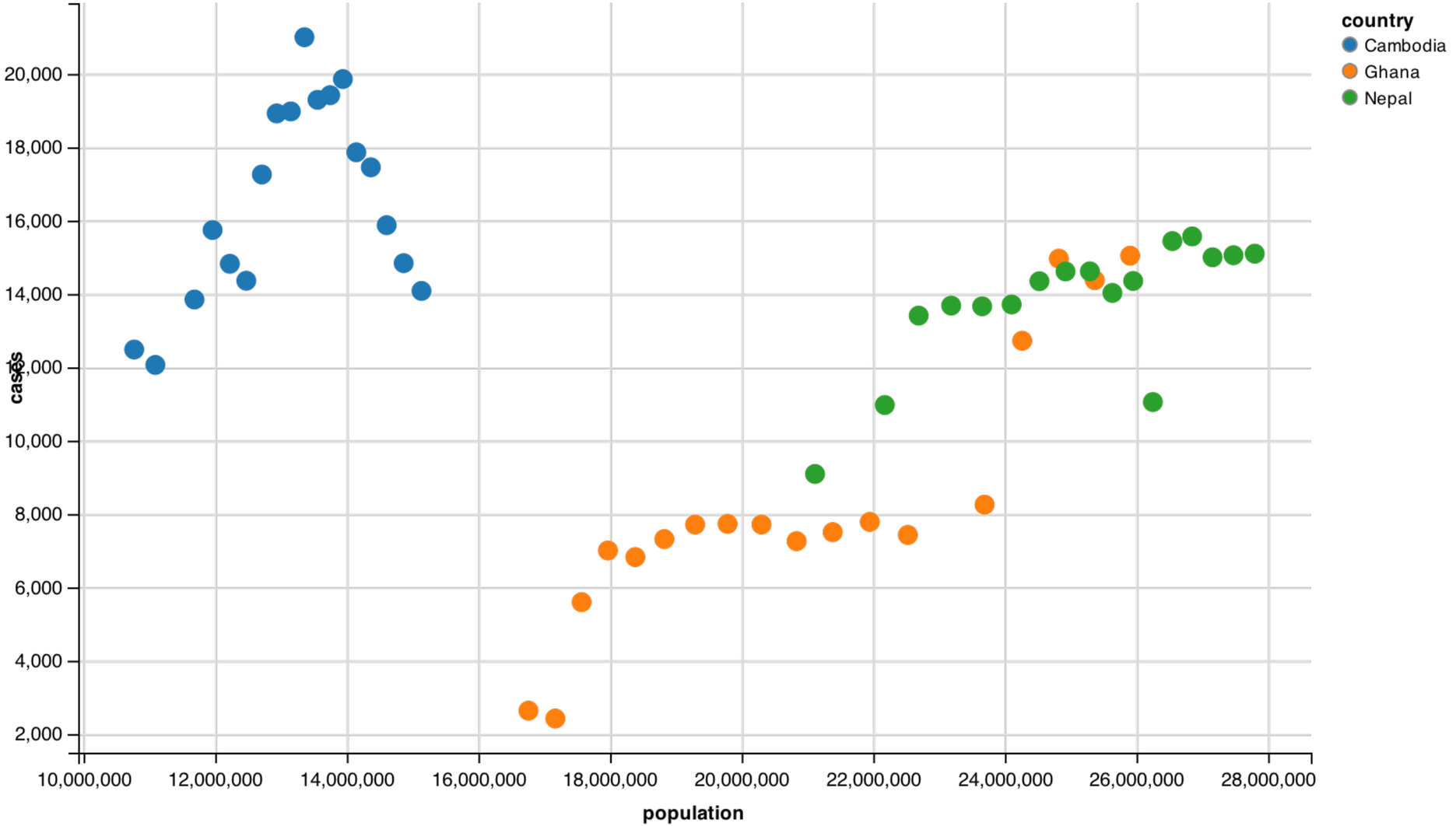
A visualization is a collection of

1. visual marks (observations)

that have

2. visual properties (variables)

The structure of data sets parallels the structure of data visualizations



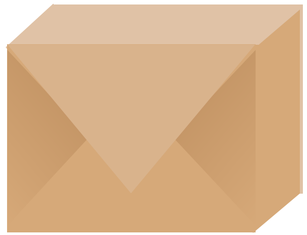
Your Turn

If you do not have tb3, recreate it now to use in the next sections.

```
tb2 <- tb %>%
  mutate(cases = child + adult + elderly) %>%
  select(country:sex, cases) %>%
  filter(!is.na(cases)) %>%
  group_by(country, year) %>%
  summarise(cases = sum(cases)) %>%
  ungroup()

population <- population %>%
  gather("year", "population", -1, convert = TRUE)

tb3 <- tb2 %>%
  left_join(population, by = c("country", "year")) %>%
  mutate(rate = cases / population * 10000) %>%
  select(country, year, rate)
```

Visualizing observations

ggvis



A package that visualizes data.

ggvis implements the *grammar of graphics*, a system for building visualizations that is built around observations and variables.

```
# install.packages("ggvis")
library(ggvis)
```

ggvis()

ggvis begins a graph …with no marks and no properties.

```
ggvis(china, …)
```

data frame to visualize

(optional) list of properties

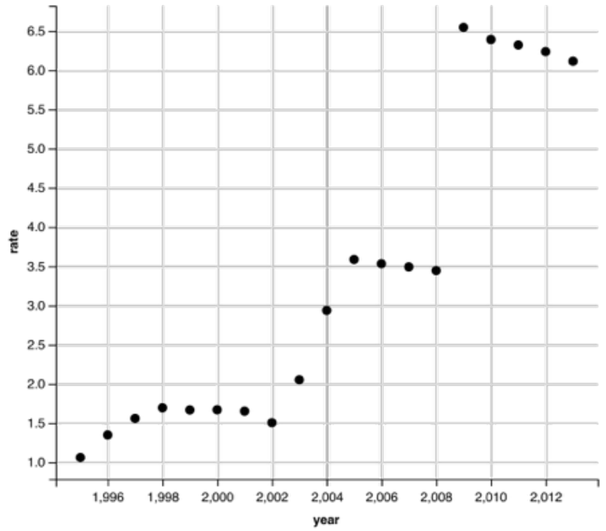
Your Turn

```
china <- tb3 %>% filter(country == "China")
```

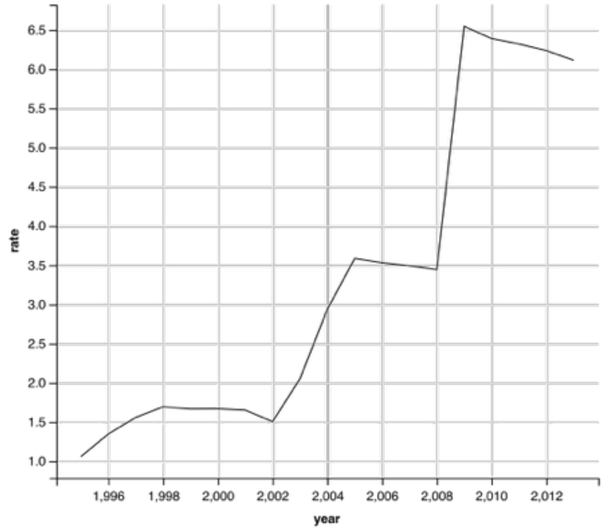
How do the following commands differ?
How does their output differ?

```
china %>% ggvis(x = ~year, y = ~rate) %>% layer_points()
china %>% ggvis(x = ~year, y = ~rate) %>% layer_lines()
china %>% ggvis(x = ~year, y = ~rate) %>% layer_bars()
china %>% ggvis(x = ~year, y = ~rate) %>% layer_smooths()
```

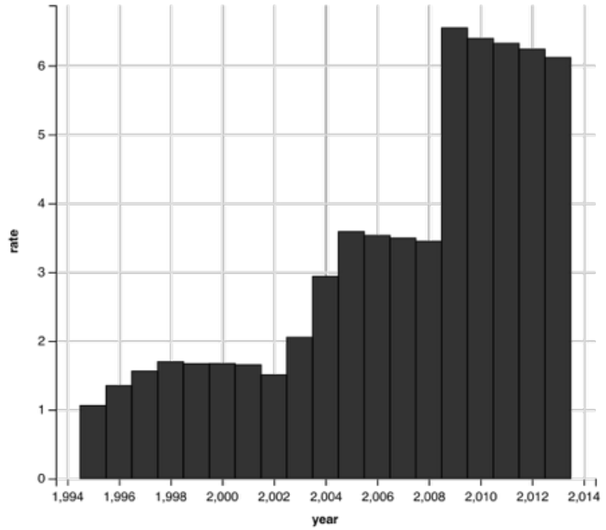
```
china %>%
  ggvis(x = ~year,
        y = ~rate) %>%
  layer_points()
```



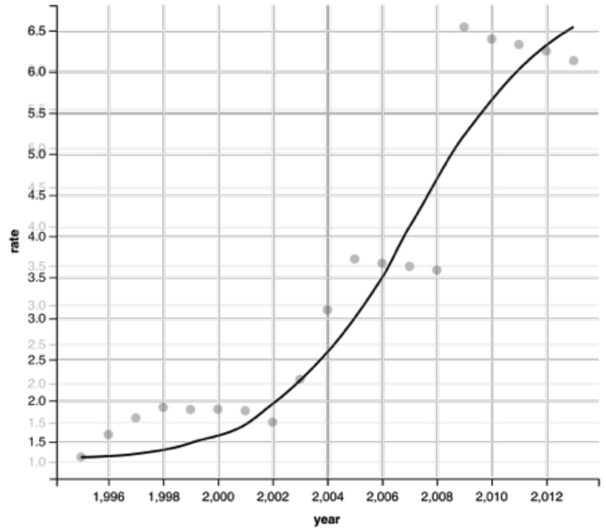

```
china %>%
  ggvis(x = ~year,
        y = ~rate) %>%
  layer_lines()
```




```
china %>%
  ggvis(x = ~year,
        y = ~rate) %>%
  layer_bars()
```

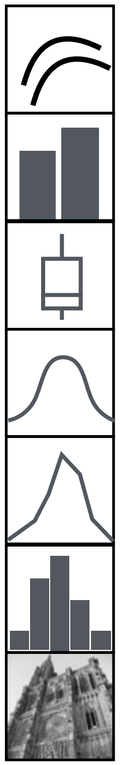



```
china %>%
  ggvis(x = ~year,
        y = ~rate) %>%
  layer_smooths()
```



Layers

Each layer represents observations (or groups of observations) with a different mark. Open a layer's help page to see which properties it uses, e.g. `?layer_arcs`



`layer_arcs`

`layer_bars`

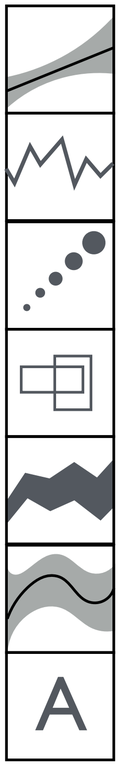
`layer_boxplots`

`layer_densities`

`layer_freqpolys`

`layer_histograms`

`layer_images`



`layer_model_predictions`

`layer_paths`

`layer_points`

`layer_rects`

`layer_ribbons`

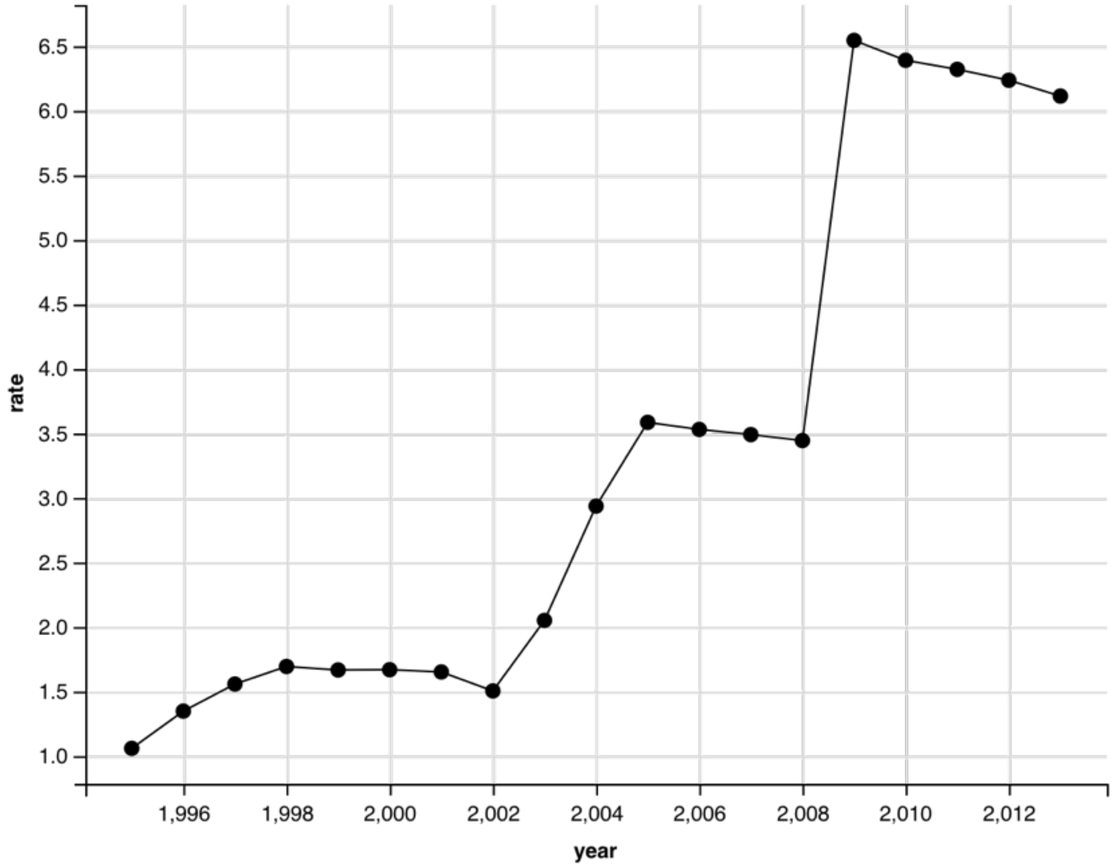
`layer_smooths`

`layer_text`

Layers

Add multiple layers to depict observations in multiple ways.

```
china %>%
  ggvis(x = ~year,
        y = ~rate) %>%
  layer_points() %>%
  layer_lines()
```



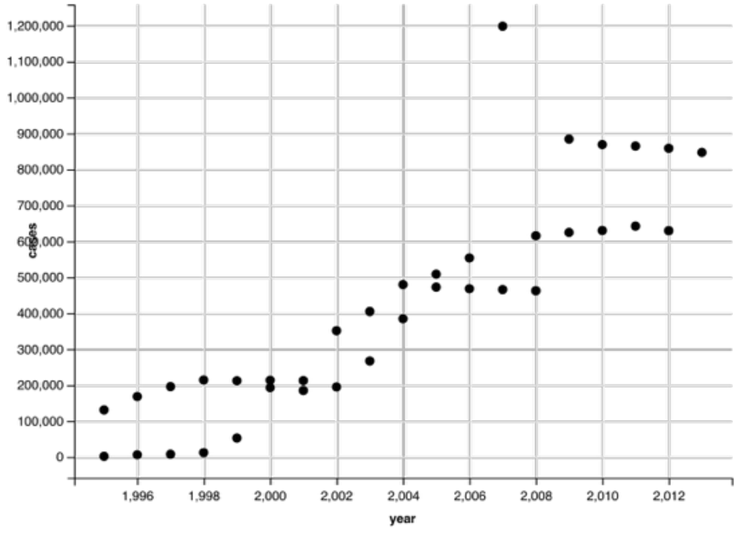
Your Turn

```
indochina <- filter(tb3, country %in% c("India", "China"))
```

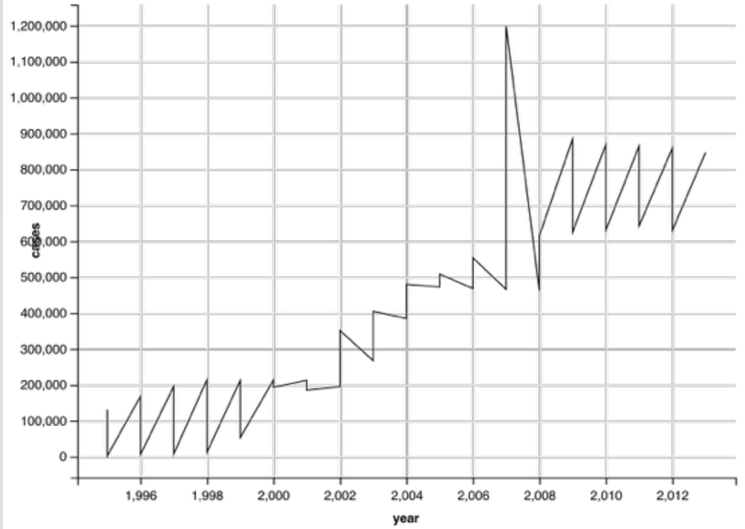
How do the following commands differ?
Can you tell what is happening?

```
indochina %>%
  ggvis(x = ~year, y = ~cases) %>% layer_points()

indochina %>%
  ggvis(x = ~year, y = ~cases) %>% layer_lines()

indochina %>% group_by(country) %>%
  ggvis(x = ~year, y = ~cases) %>%layer_lines()
```

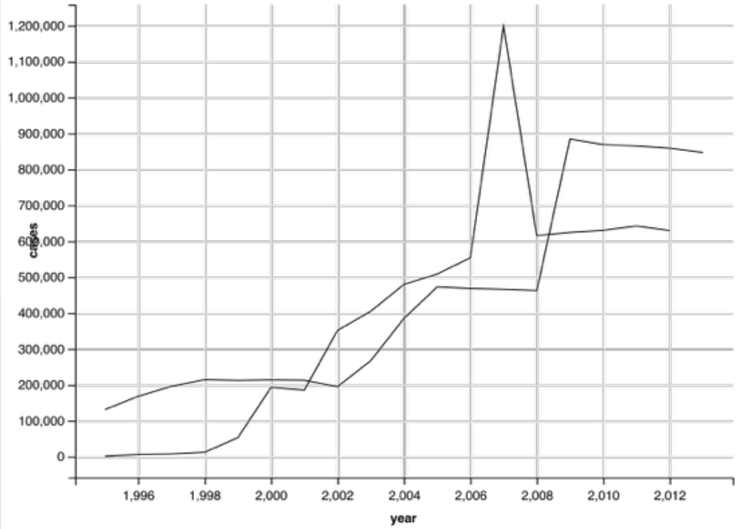
```
indochina <- filter(tb3, country %in% c("India", "China"))
```




```
indochina %>%
  ggvis(x = ~year,
        y = ~cases) %>%
  layer_points()
```




```
indochina %>%
  ggvis(x = ~year,
        y = ~cases) %>%
  layer_lines()
```



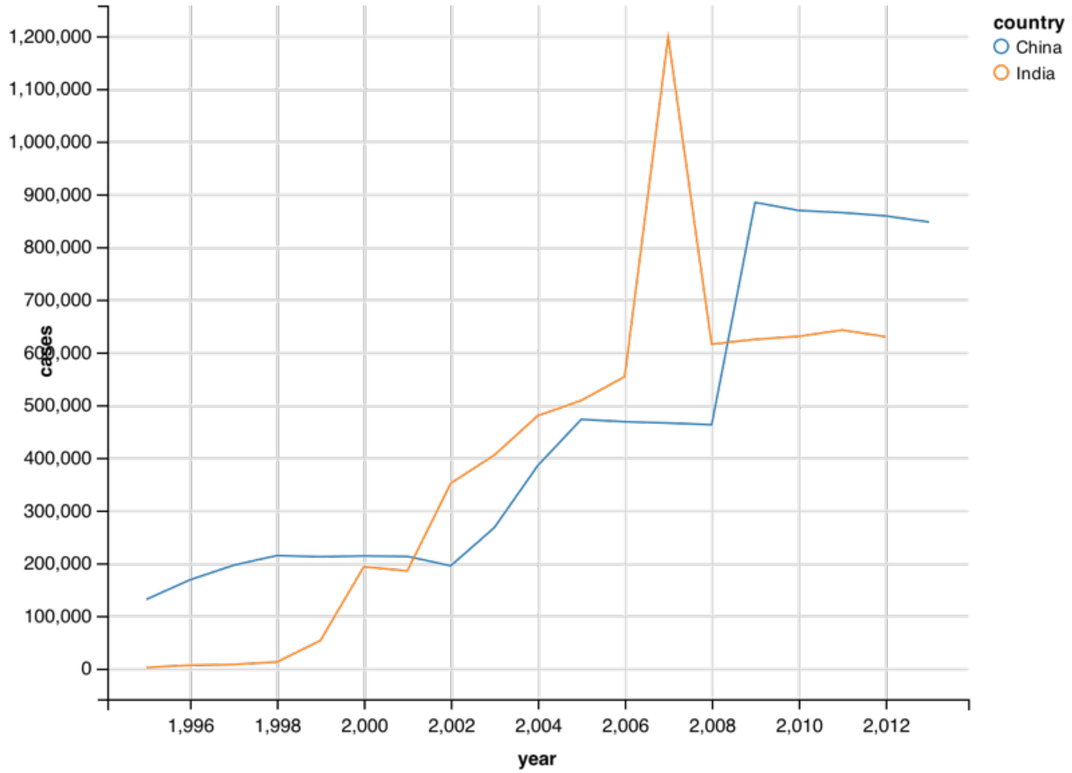

```
indochina %>%
  group_by(country) %>%
  ggvis(x = ~year,
        y = ~cases) %>%
  layer_lines()
```

grouping

ggvis will draw a separate mark for each group of observations in grouped data.

ggvis will automatically group data when necessary, e.g.

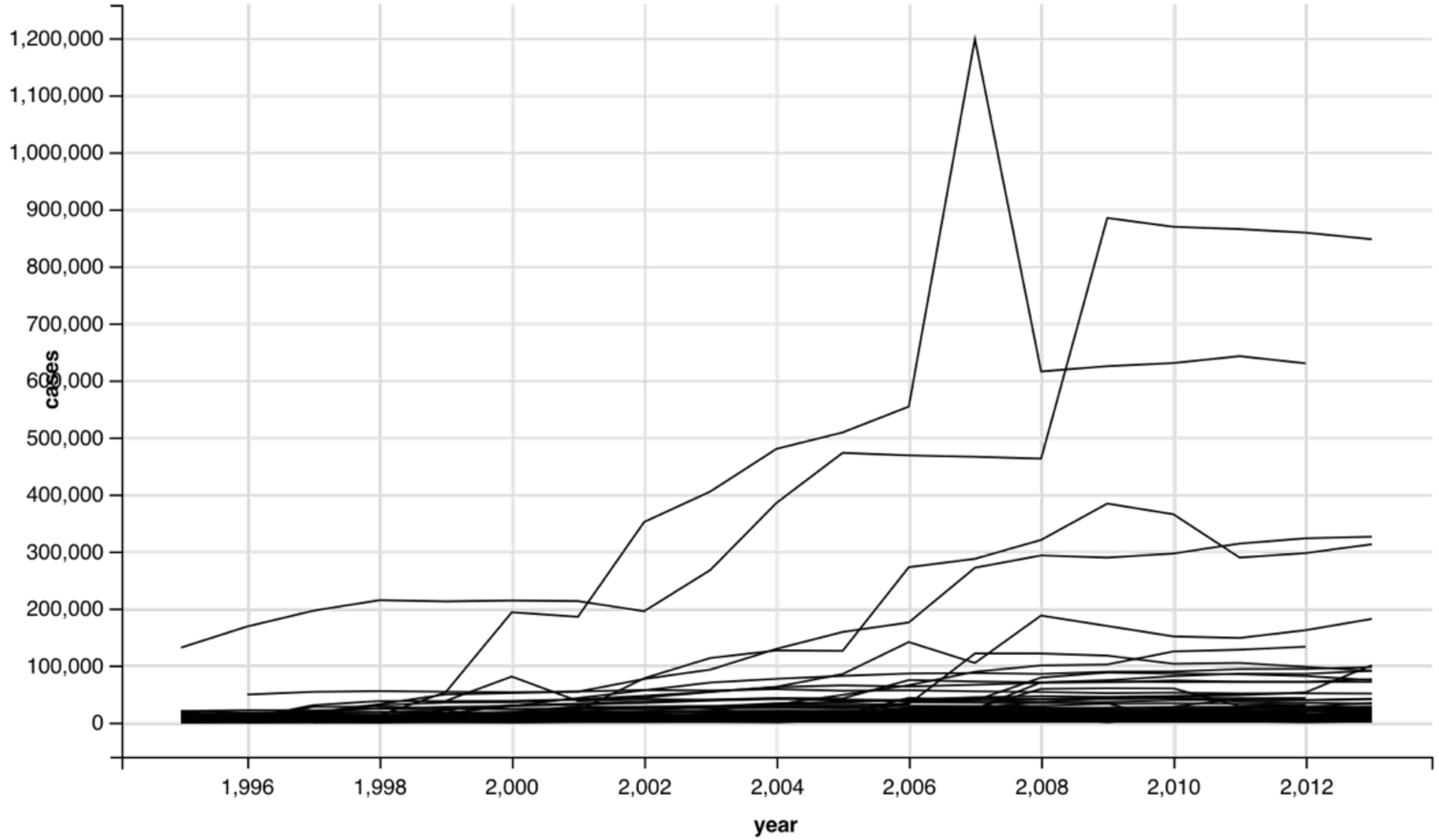
```
indochina %>%
  ggvis(x = ~year,
        y = ~cases,
        stroke = ~country) %>%
  layer_lines()
```



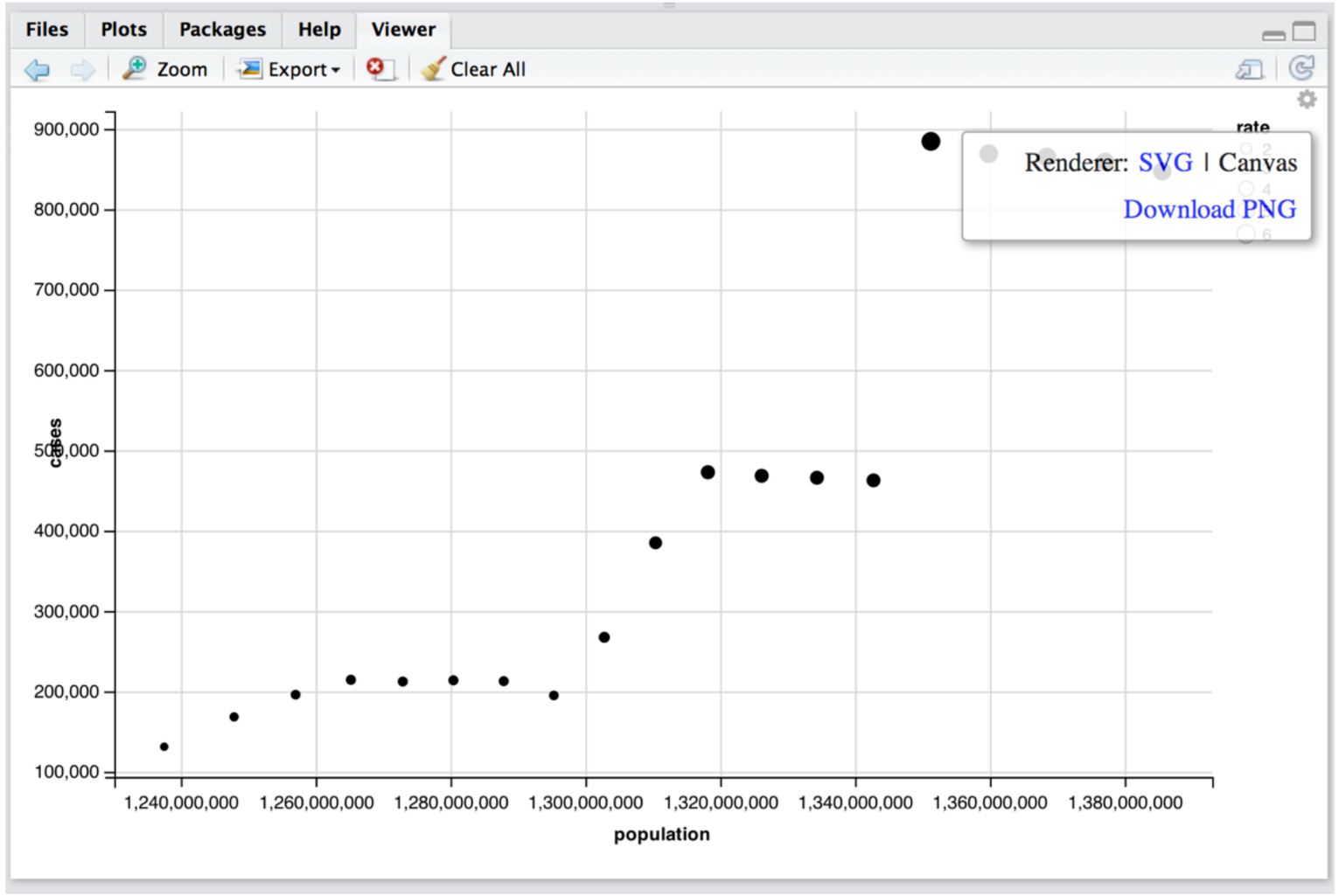
Your Turn

Plot a line graph of the tb3 data. Put year on the x axis, cases on the y axis, and include a separate line for each country.

```
tb3 %>%
  group_by(country) %>%
  ggvis(x = ~year, y = ~cases) %>%
  layer_lines()
```



Saving plots

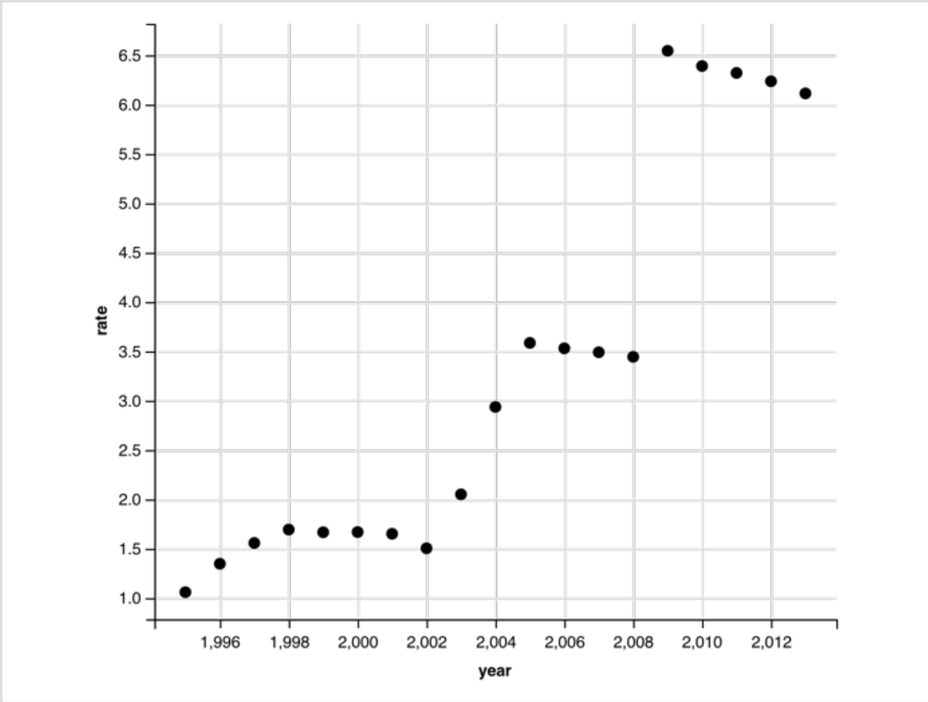


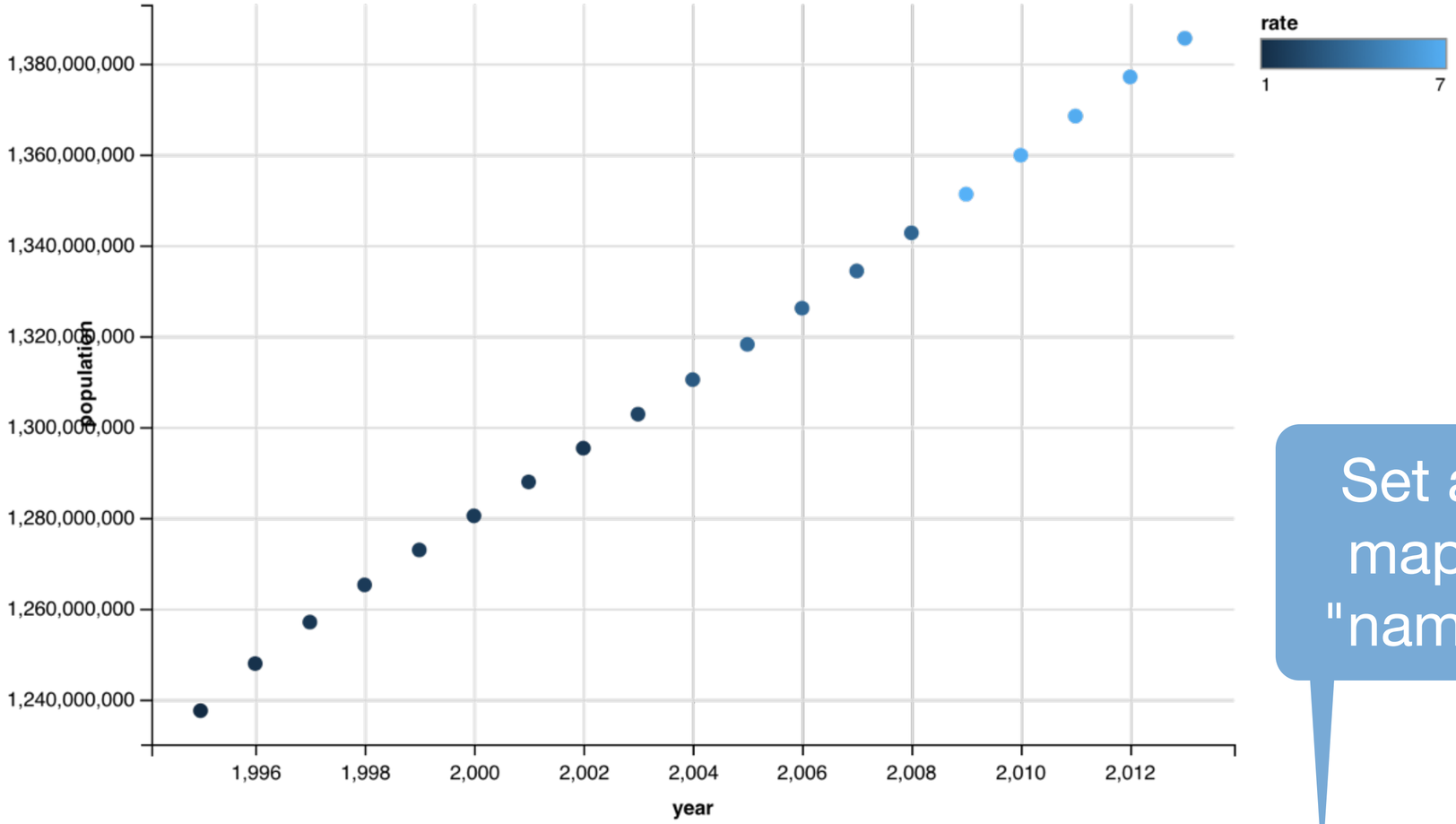
Visualizing variables

Your Turn

This graph sets the **x** location property to **year** and the **y** location property to **rate**. Try to set the **y** location property to **population** and set the **fill** property to **rate.**

```
china %>%
  ggvis(x = ~year, y = ~rate) %>%
  layer_points()
```





Set all property mappings with "name =" syntax

```
china %>%
  ggvis(x = ~year, y = ~population, fill = ~rate) %>%
  layer_points()
```

Quiz

```
x <- c(1, 2, 3)

df <- data.frame(
  x = c("a", "b", "c"),
  y = 1:3, stringsAsFactors = FALSE)
```

x

1
2
3

df

x	y
A	1
B	2
C	3

What will each of these commands return?

x "x" df$x

```
x
```

```
## 1 2 3
```

The object named x

```
"x"
```

```
## "x"
```

The literal value "x" (a string)

```
df$x
```

```
## "a" "b" "c"
```

The column named x in the data frame named df

~

The ~ syntax provides a shortcut for referring to a column in your data frame.

```
fill = ~rate
```

The object named rate

```
fill = "rate"
```

The literal value "rate" (a string)

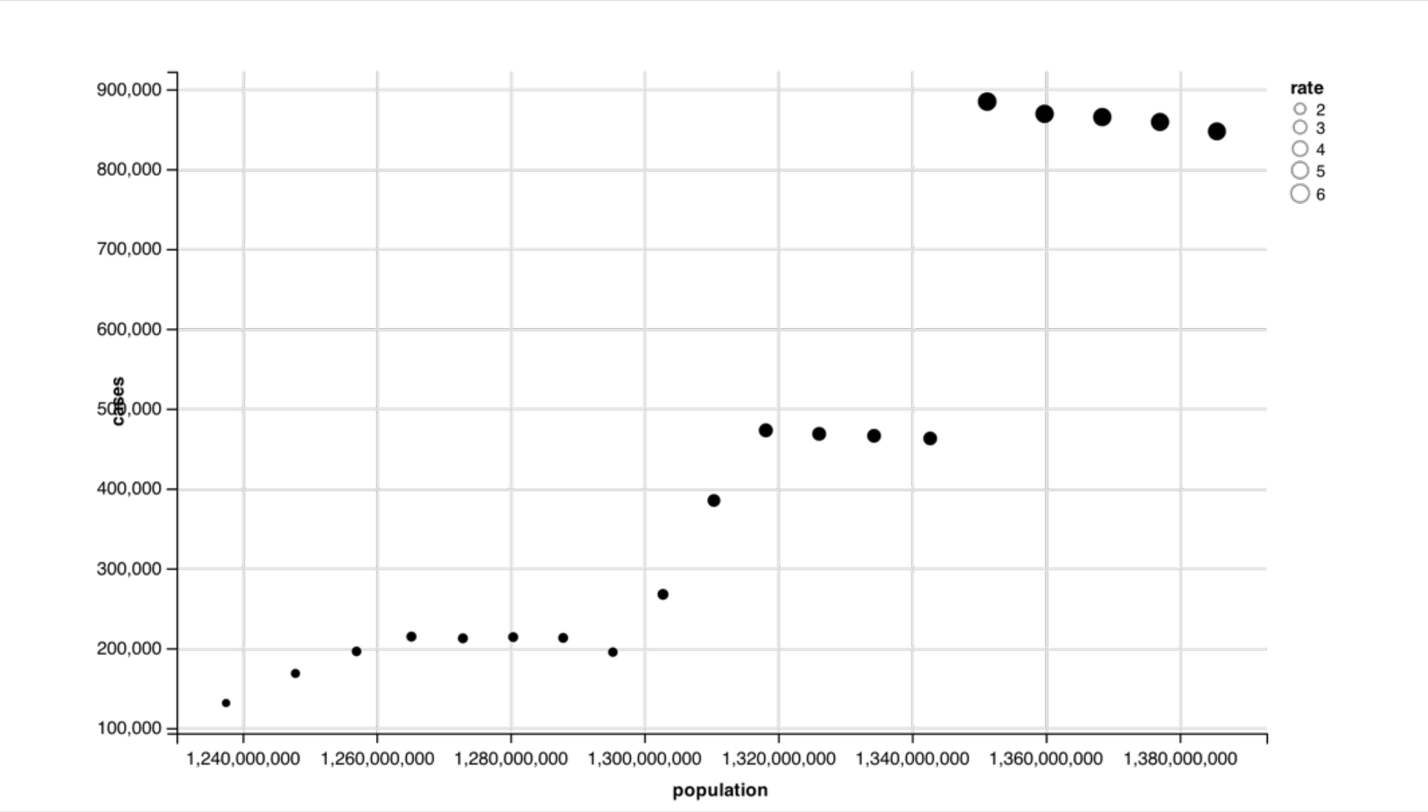
```
fill = ~df$rate
```

The column named rate in the data frame that you passed to ggvis

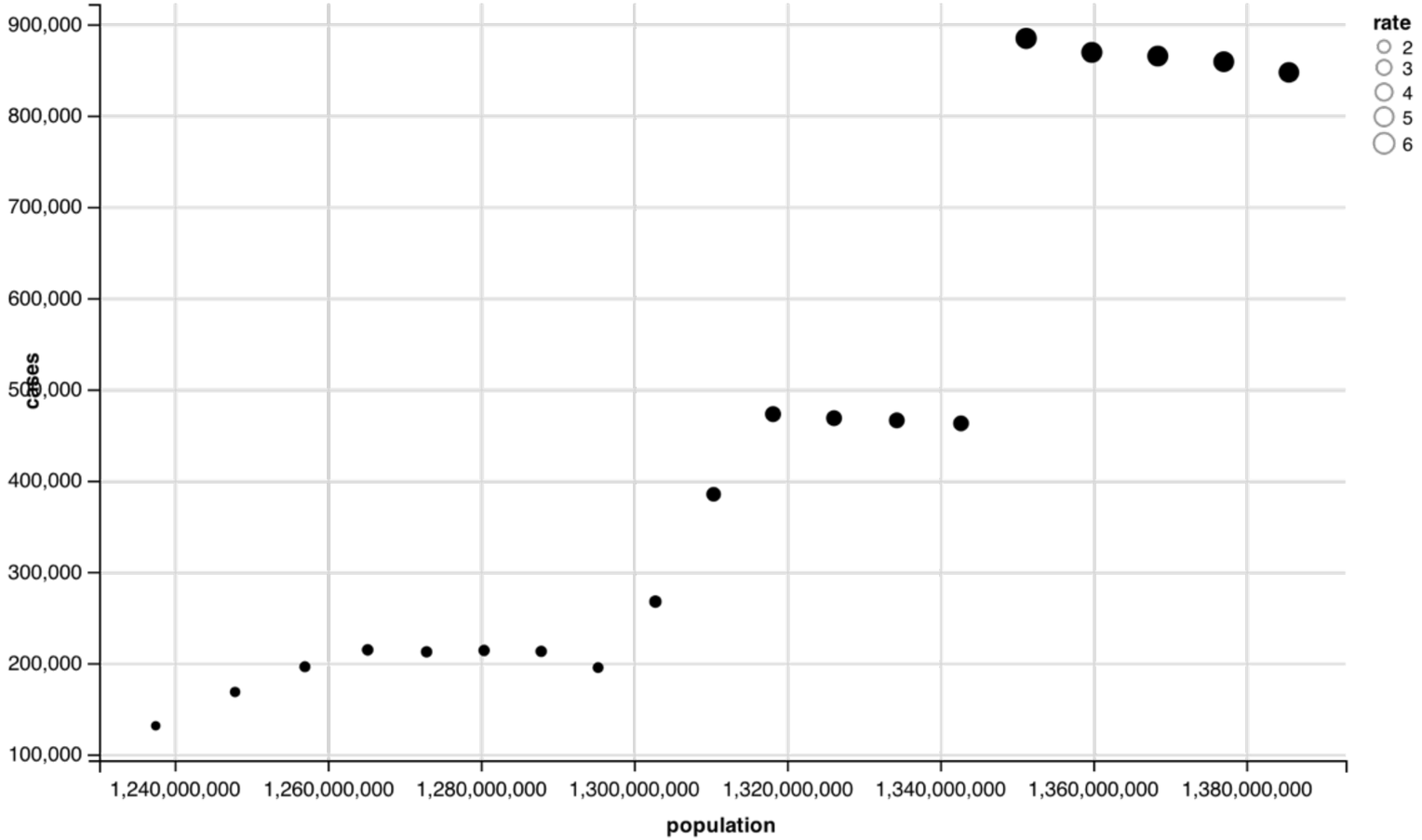
Your Turn

```
china <- tb3 %>% filter(country == "China")
```

Try to create the graph below.



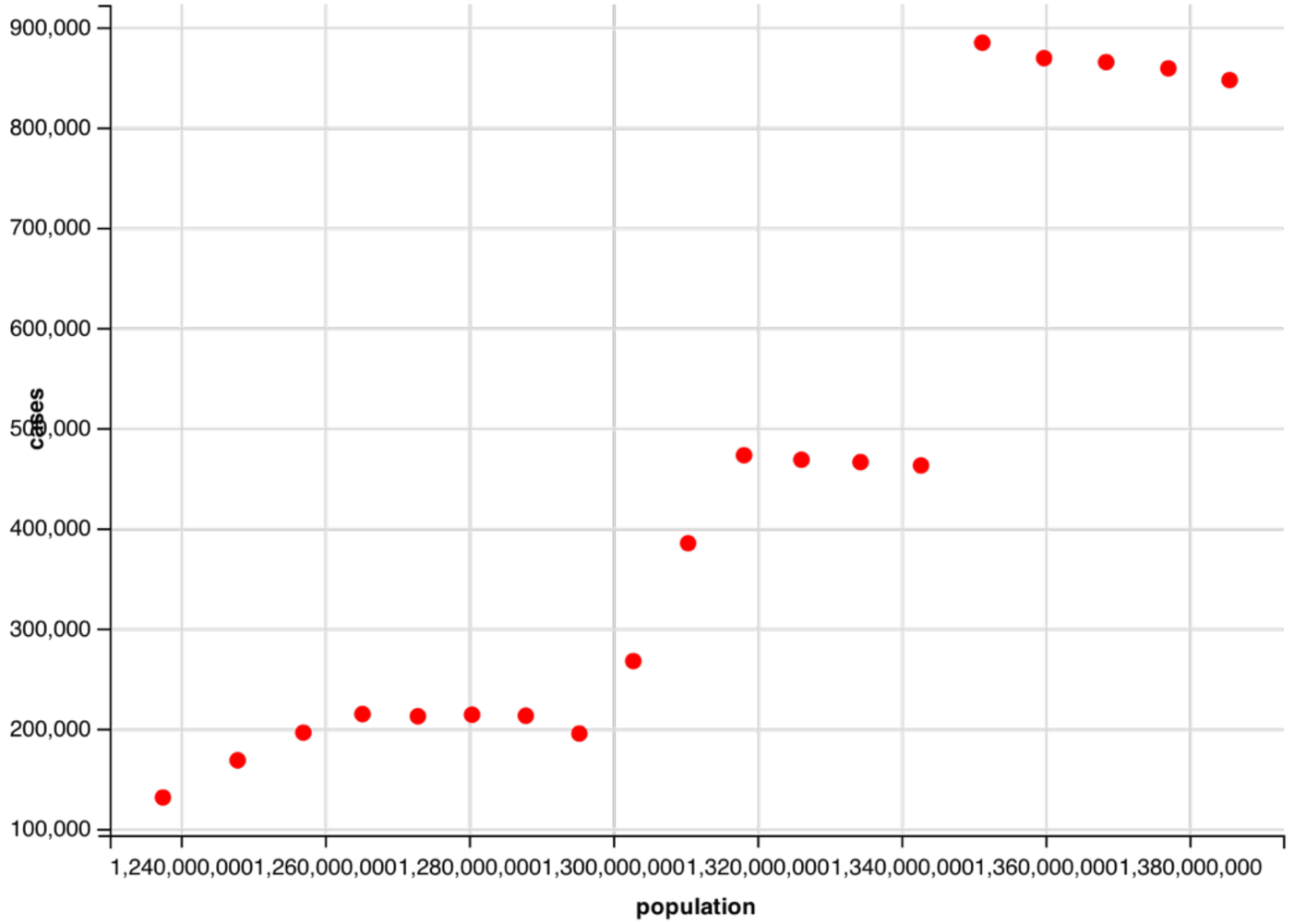
```
china %>%
  ggvis(x = ~population, y = ~cases, size = ~rate) %>%
  layer_points()
```



= vs :=

What if you want to manually set a property?

e.g. make all of the points red?

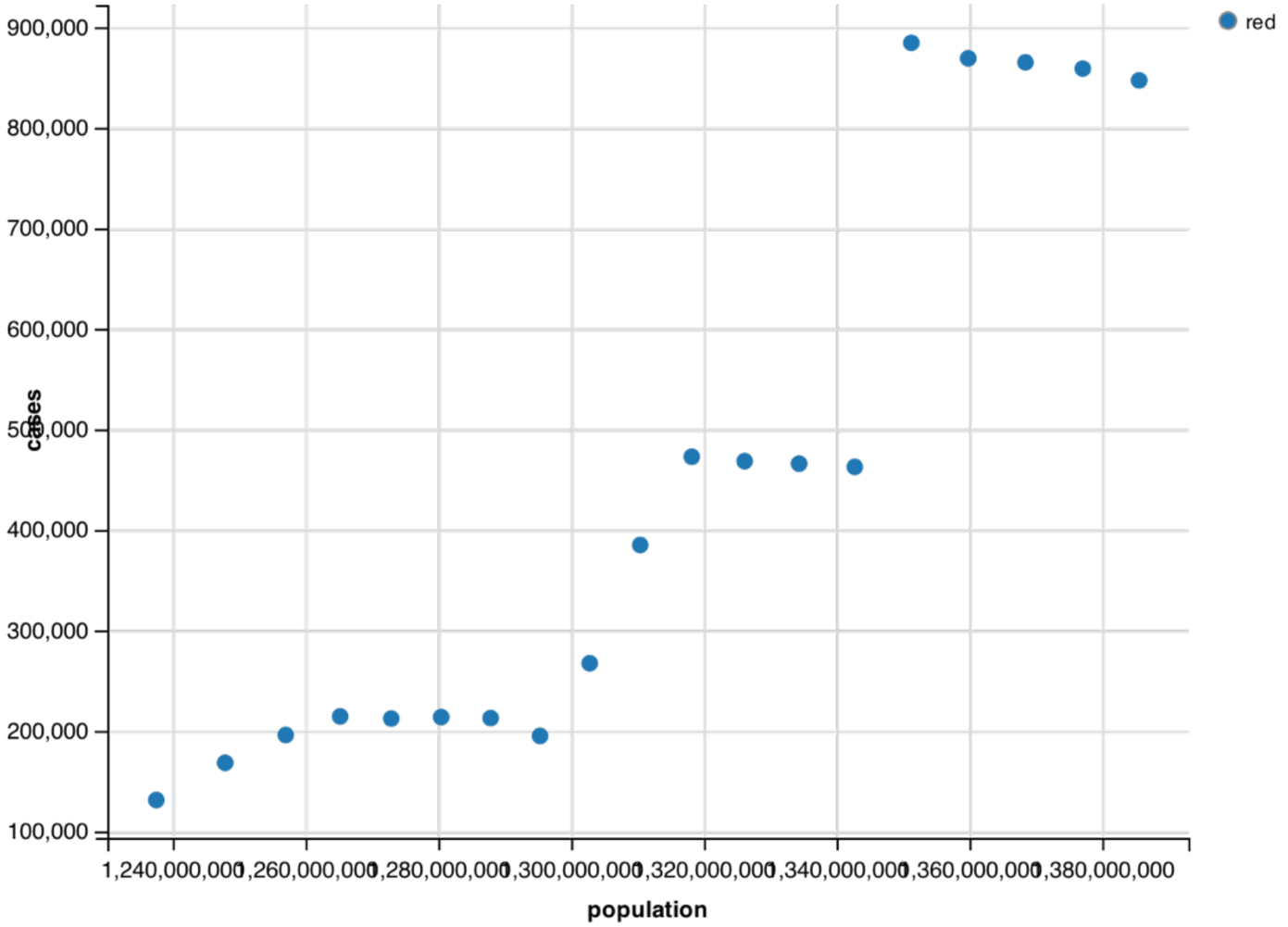


= vs :=

What if you want to manually set a property?

e.g. make all of the points red?

```
china %>%
  ggvis(x = ~population,
        y = ~cases,
        fill = "red") %>%
  layer_points()
```

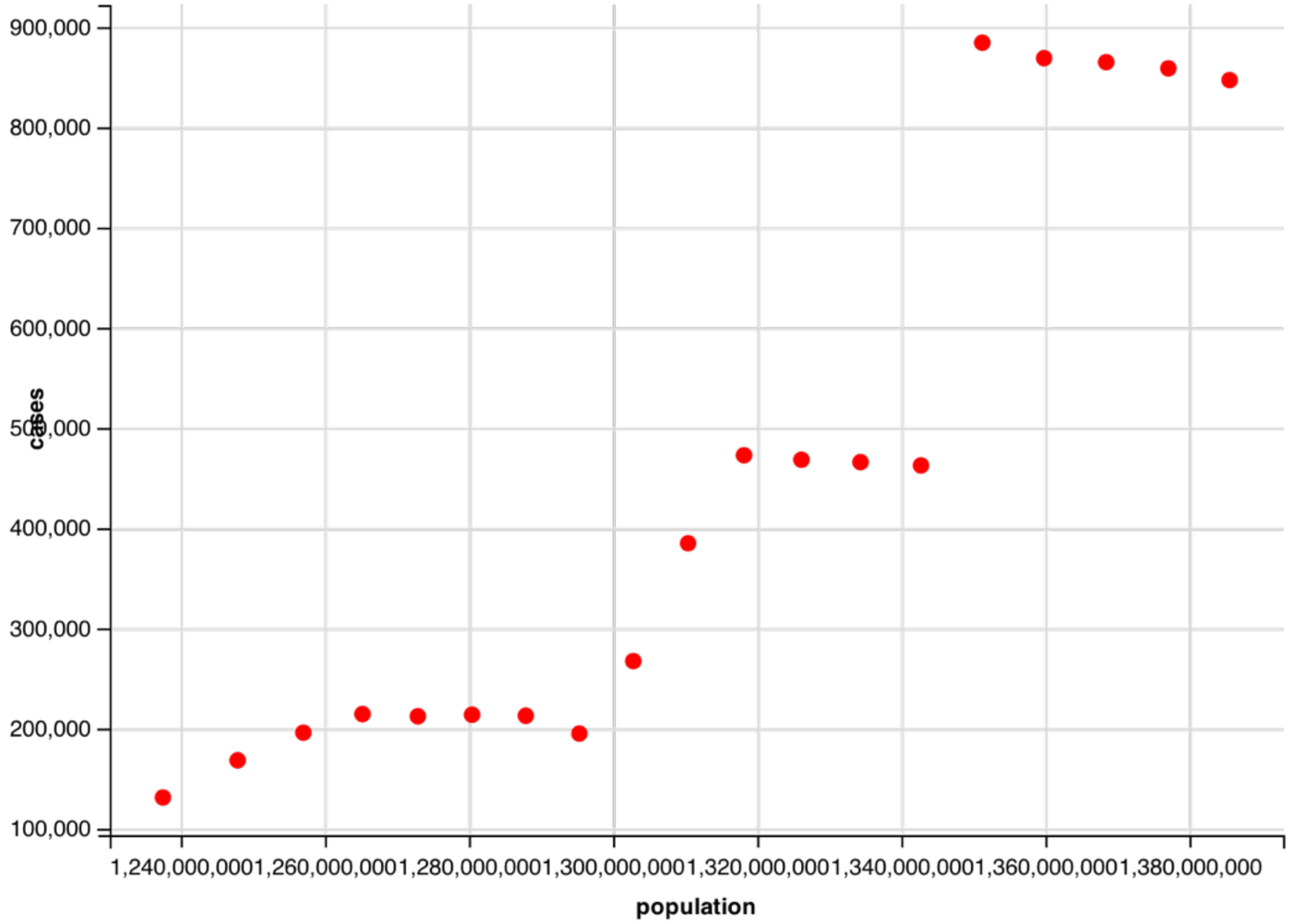


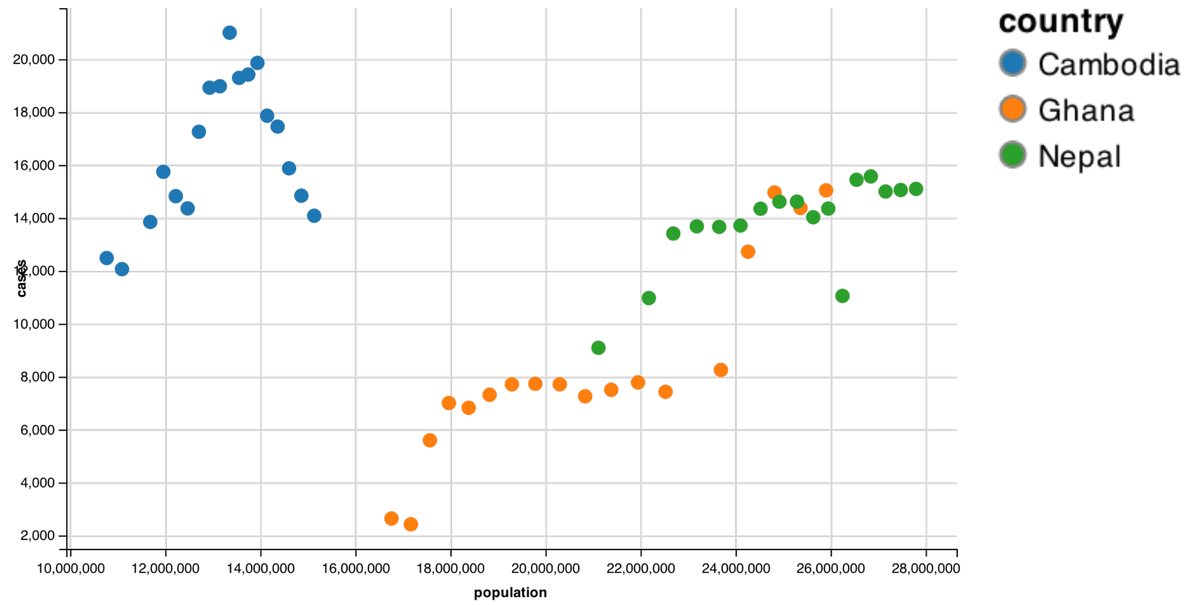
= vs :=

What if you want to manually set a property?

e.g. make all of the points red?

```
china %>%
  ggvis(x = ~population,
        y = ~cases,
        fill := "red") %>%
  layer_points()
```





Data Space

Visual Space

country

Cambodia

Ghana

Nepal

fill

Blue

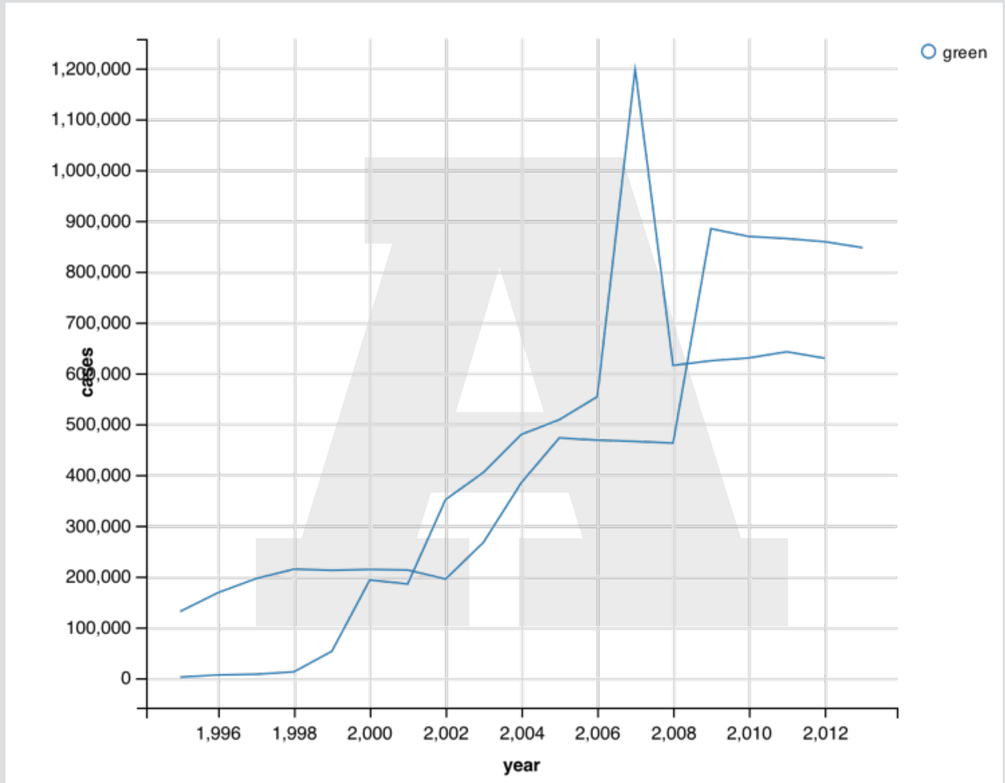
Orange

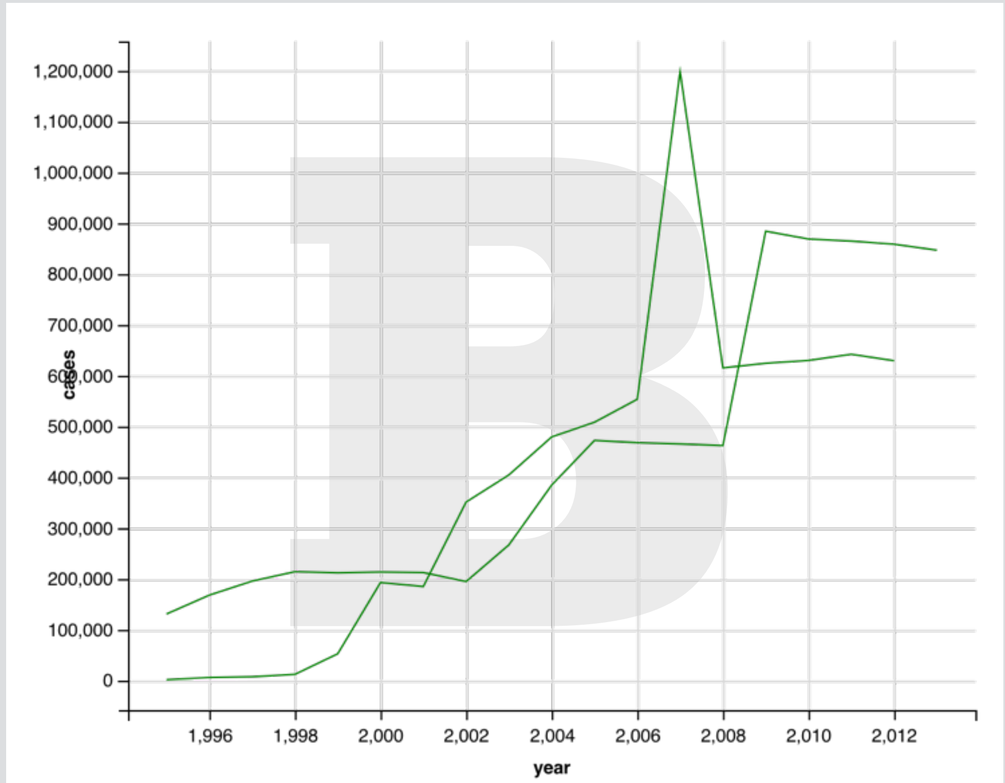
Green

Quiz

Which graph will this code make?

```
indochina %>% group_by(country) %>%
  ggvis(x = ~year, y = ~cases, stroke = "green") %>%
  layer_lines()
```

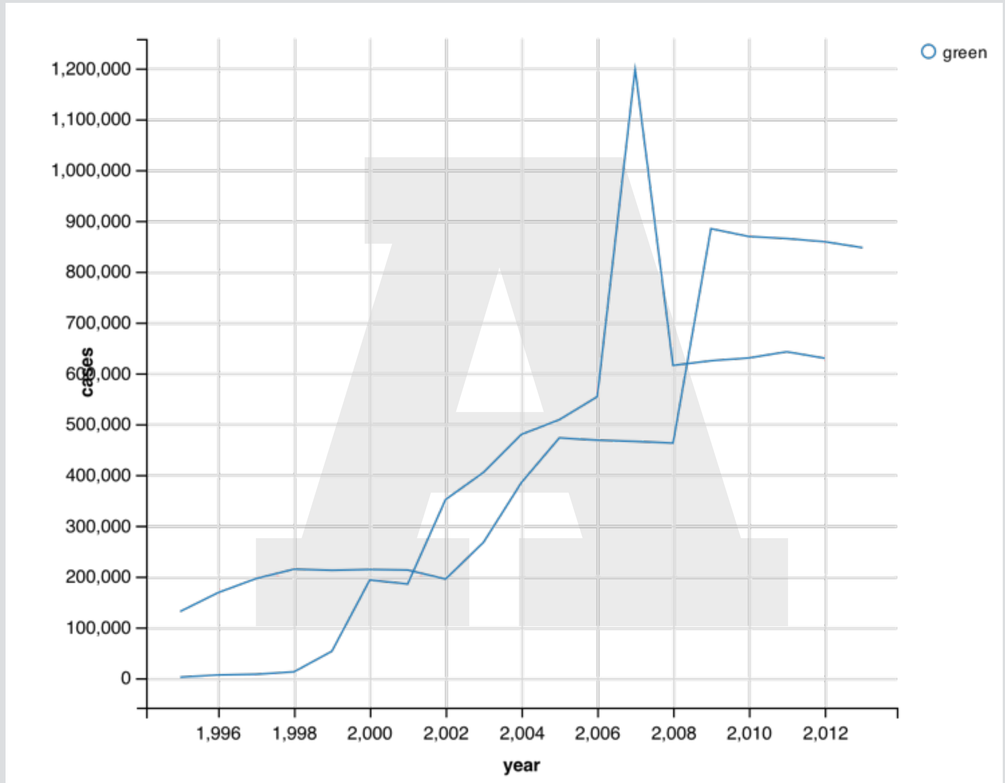


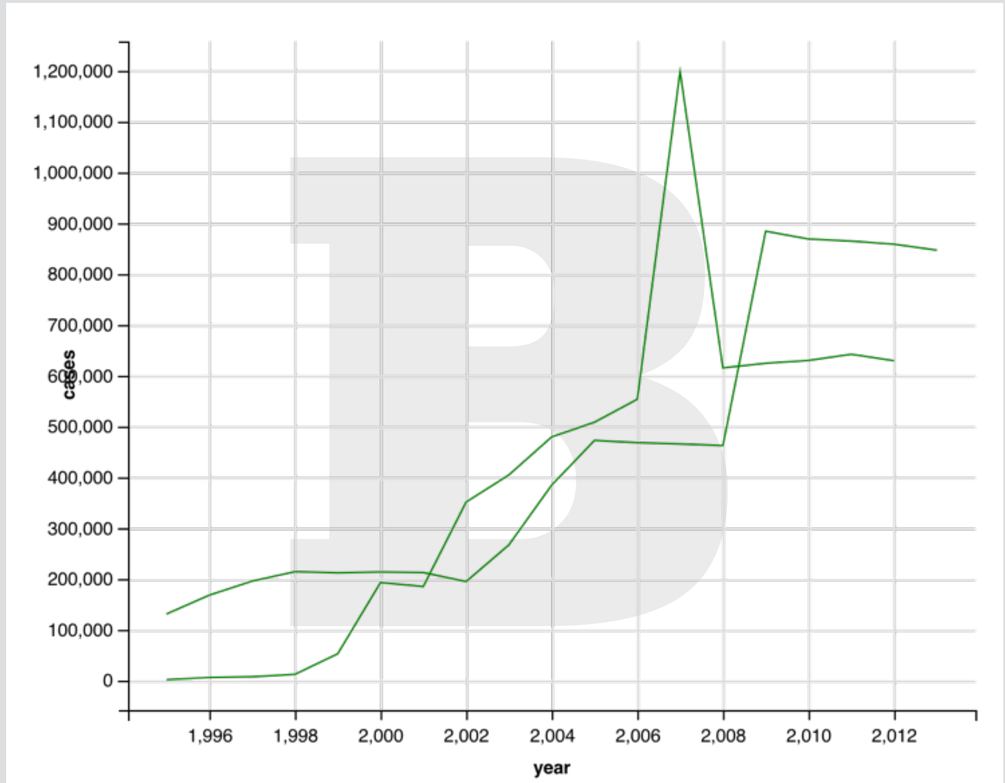


Quiz

Which graph will this code make?

```
indochina %>% group_by(country) %>%
  ggvis(x = ~year, y = ~cases, stroke := "green") %>%
  layer_lines()
```

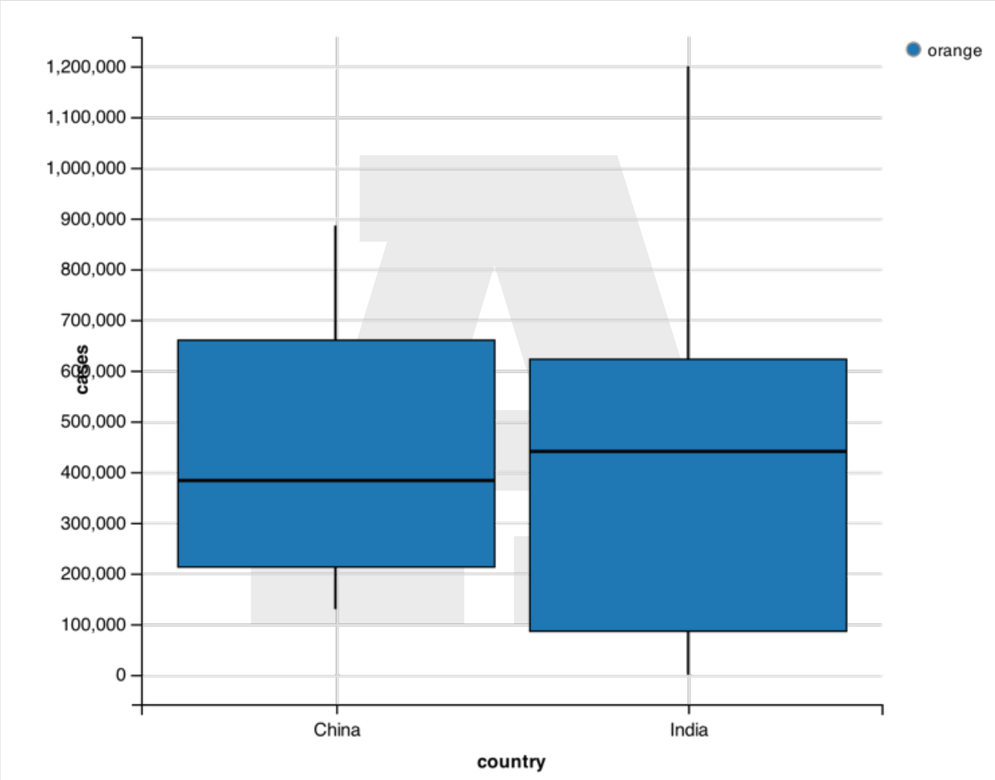


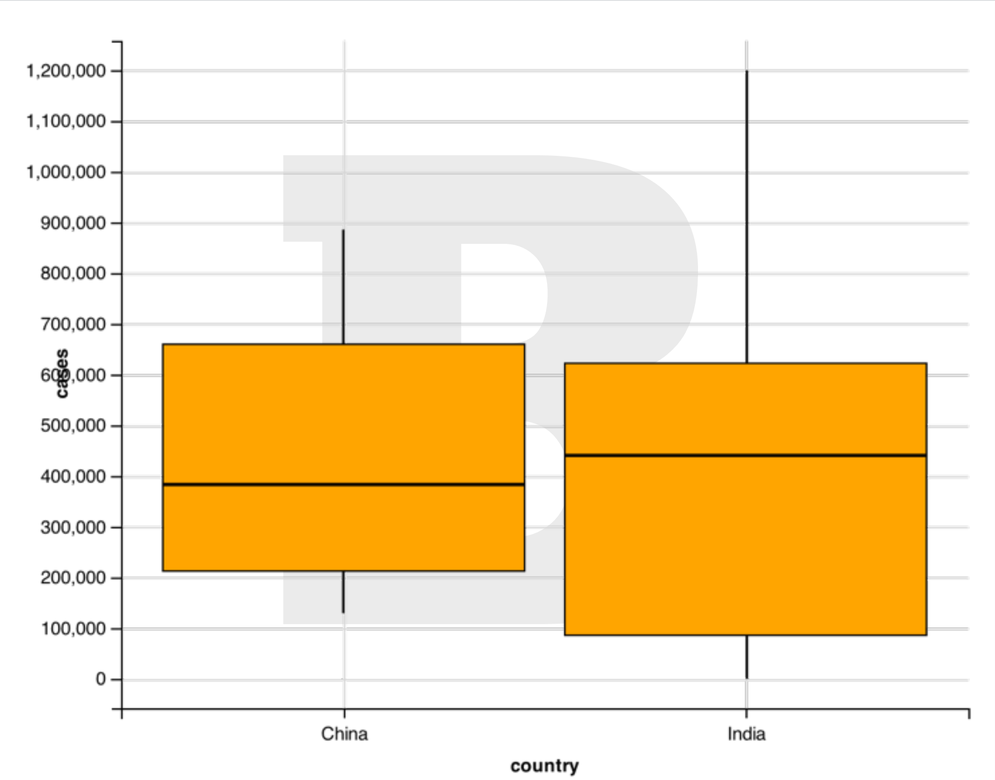


Quiz

Which graph will this code make?

```
indochina %>%
  ggvis(x = ~country, y = ~cases, fill := "orange") %>%
  layer_boxplots()
```

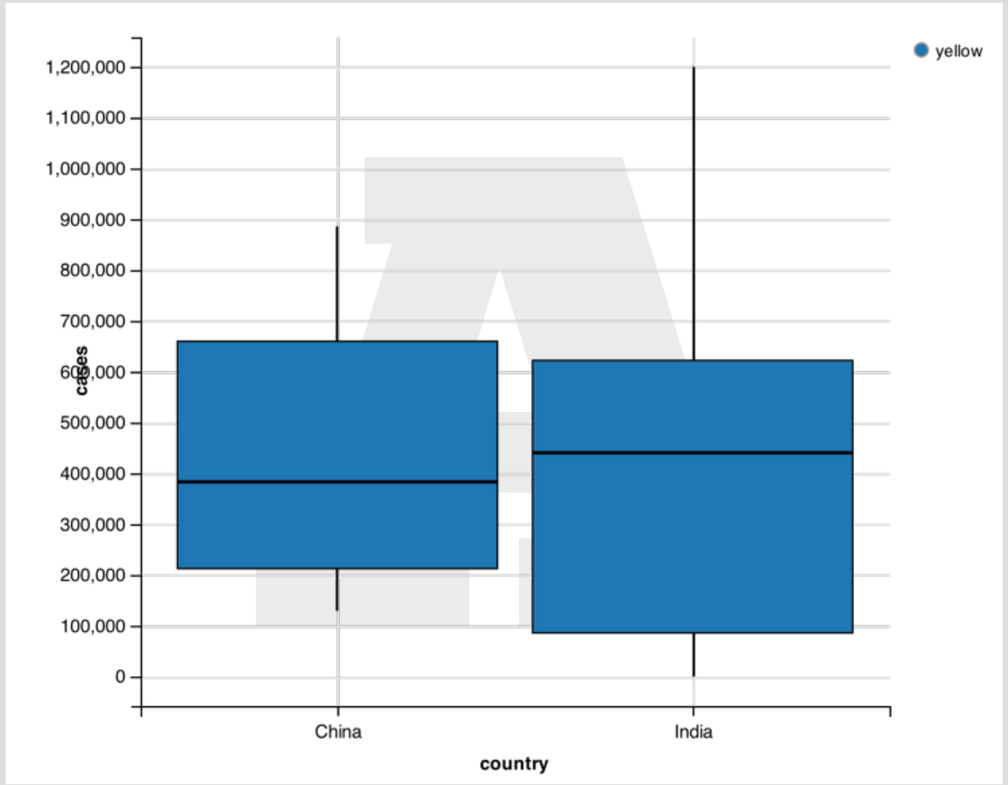


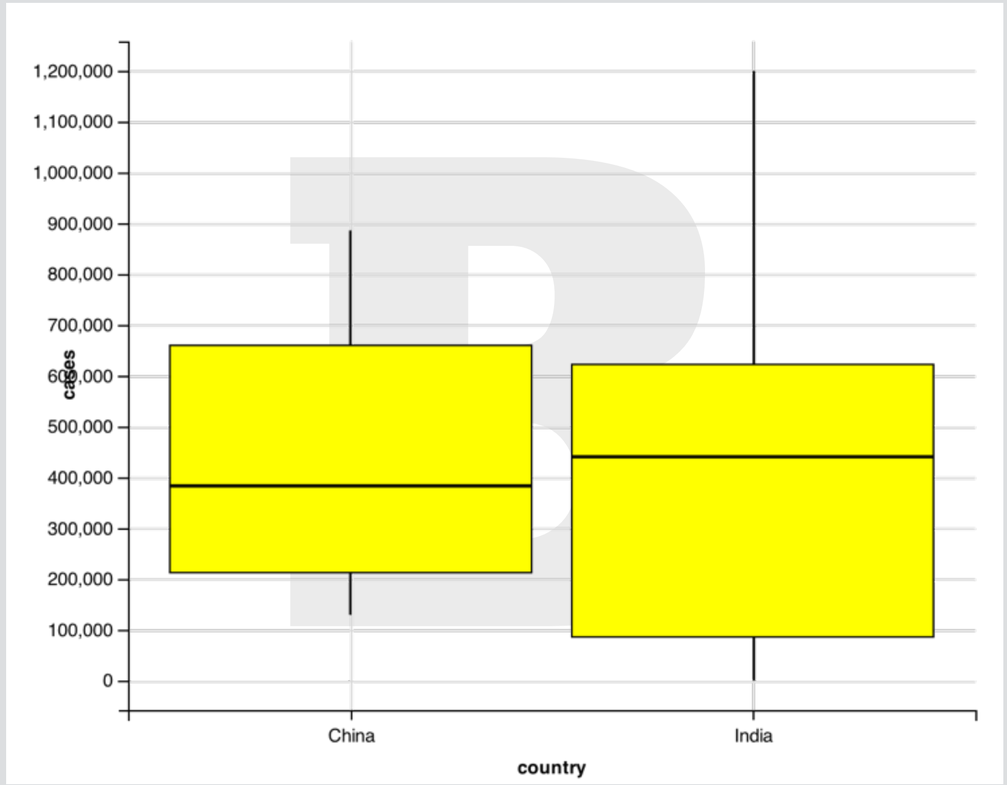


Quiz

Which graph will this code make?

```
color <- "yellow"
indochina %>%
  ggvis(x = ~country, y = ~cases, fill := color) %>% layer_boxplots()
```





interactive graphs

Inputs

Make plots interactive by mapping properties to an input control. Create input controls with an **input_** function.

```
sliderBox <- input_slider(.1, 2, value = 1, step = .1,
  label = "Bandwidth adjustment")

selectBox <- input_select(c("Gaussian" = "gaussian",
  "Epanechnikov" = "epanechnikov","Rectangular" = "rectangular",
  "Triangular" = "triangular","Biweight" = "biweight",
  "Cosine" = "cosine", "Optcosine" = "optcosine"),label = "Kernel")

mtcars %>%
  ggvis(x = ~wt) %>%
  layer_densities(adjust = sliderBox, kernel = selectBox)
```

Inputs

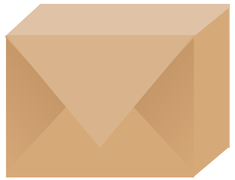
Currently available input functions.

```
input_checkbox
input_checkboxgroup
input_numeric
input_radiobuttons
input_select
input_slider
input_text
```

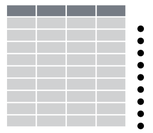
Hover events

```
# This function receives information about the hovered
# point and returns an HTML string to display
all_values <- function(x) {
  if(is.null(x)) return(NULL)
  paste0(names(x), ": ", format(x), collapse = "<br />")
}

mtcars %>% ggvis(x = ~wt, y = ~mpg) %>%
  layer_points(fill.hover := "red") %>%
  add_tooltip(all_values, "hover")
```

Recap: visualization



ggvis: A package that visualizes data.



Depict observations as visual marks with a layer_*() function

Map variables to visual properties

Keep track of data space and visual space with = and := syntax