Data visualization and manipulation in R:

ggplot2, dplyr, and the tidyverse

Joseph K Aicher (Barash/Bhoj Labs)

Derived from **bit.ly/gentle-ggplot2**

2019-08-23

Brought to you by the letter...

Learning objectives

- Why you should learn ggplot2 and dplyr
- Mechanics and getting started with **ggplot2**
- Transforming data using **dplyr**
- What is the tidyverse?

What is *ggplot2*?

Hadley Wickham

"[ggplot2 is] a powerful way of thinking about visualisation, as a way of **mapping between variables and the visual properties of geometric objects** that you can perceive."

http://disq.us/p/sv640d

What is *dplyr*?

dplyr provides a set of tools for efficiently manipulating datasets in R.

Key ideas:

- Tabular data is tabular data regardless of source
- Most tasks are a composition of simple operations

Adapted from https://blog.rstudio.com/2014/01/17/introducing-dplyr/

What is the *tidyverse*?

"The tidyverse is an opinionated **collection of R packages** designed for data science. All packages have an underlying design philosophy, grammar, and data structures" (tidyverse.org)

- `tidyr`: clean and reshape your data
- `readr`: better file parsing
- `tibble`: better data frames
- `purrr`: better functional programming
- … and many more (i.e. `broom`)…

Incredibly accessible and deep online textbook: R for Data Science

Getting started with ggplot2

What are we getting into?

`ggplot2` is a huge package: philosophy + functions
...but it's very well organized

Lots of examples of not-so-great plots in these slides
...but that's okay

Going to throw a lot at you
...but you'll know *where* and *what* to look for

What is the general process for using *ggplot2*?

- Functional data visualization
  1. Define data being used
  2. Map data to visual elements
  3. Tweak scales, guides, axis, labels, theme

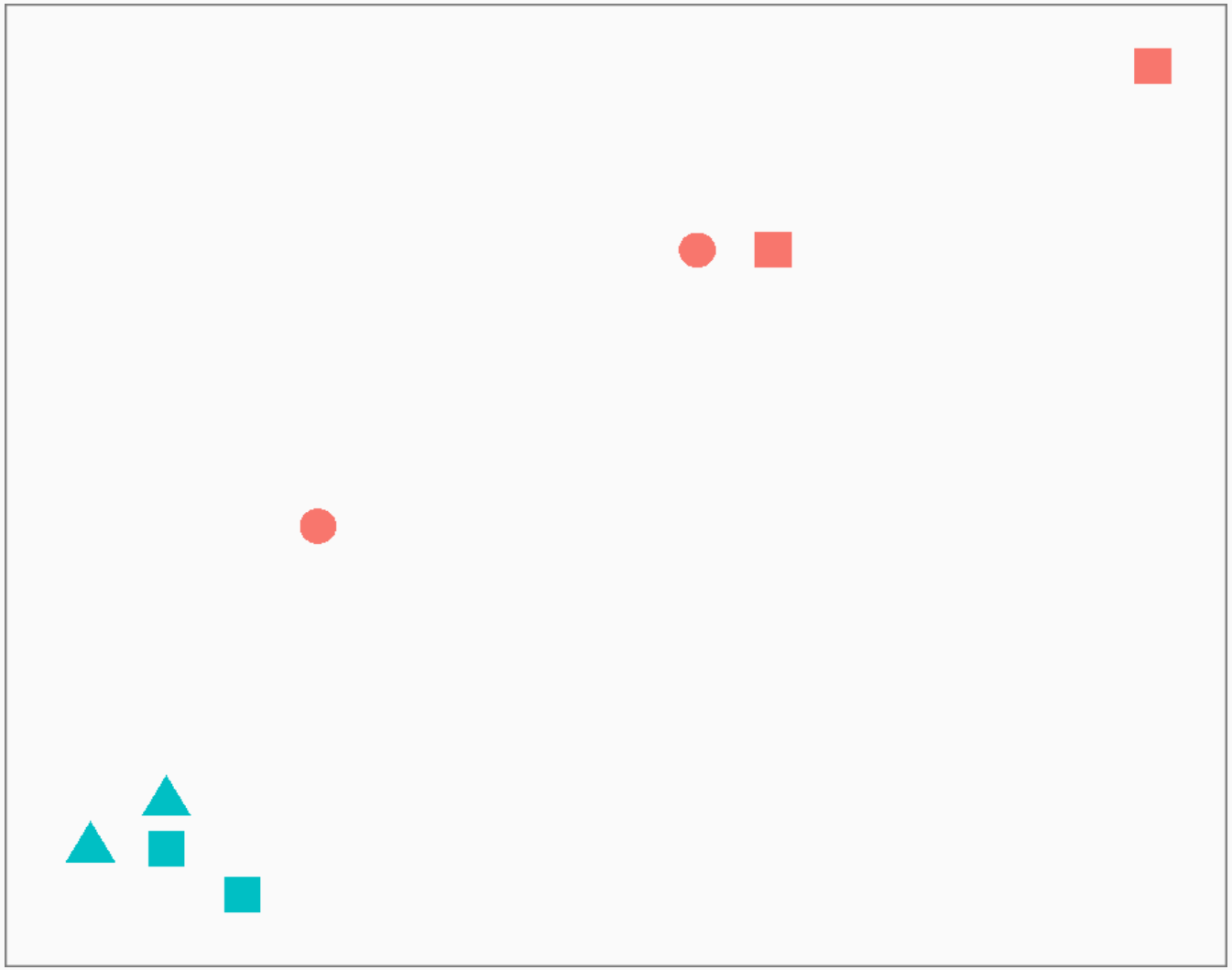
- Reason about how data drives the visualization

- Iterate on visualization

gg is for *grammar of graphics*

"Good grammar is just the first step in creating a good sentence."

How is the drawing on the right connected to data?

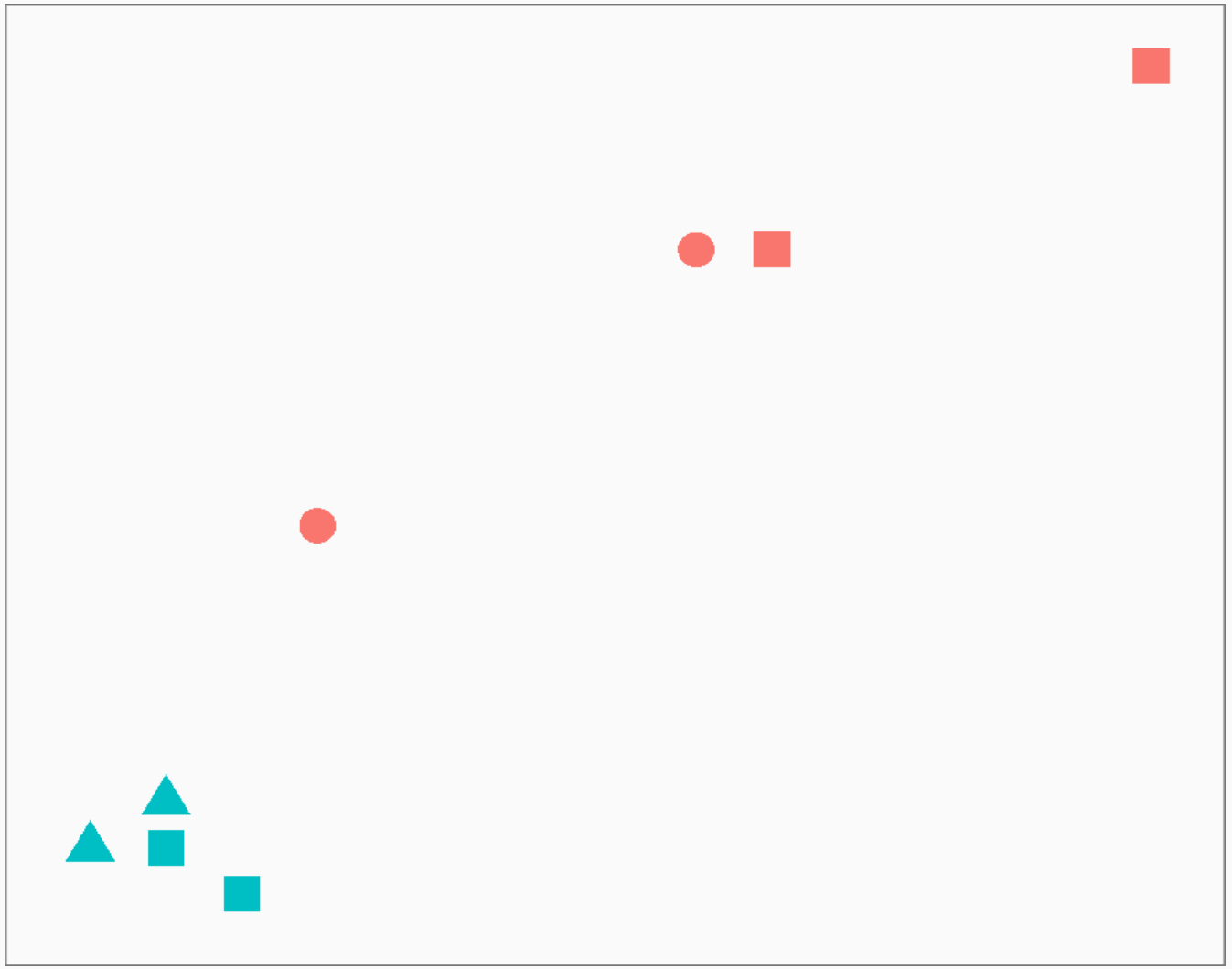


http://vita.had.co.nz/papers/layered-grammar.pdf

Guess the data behind this plot?

MPG Ratings of Cars

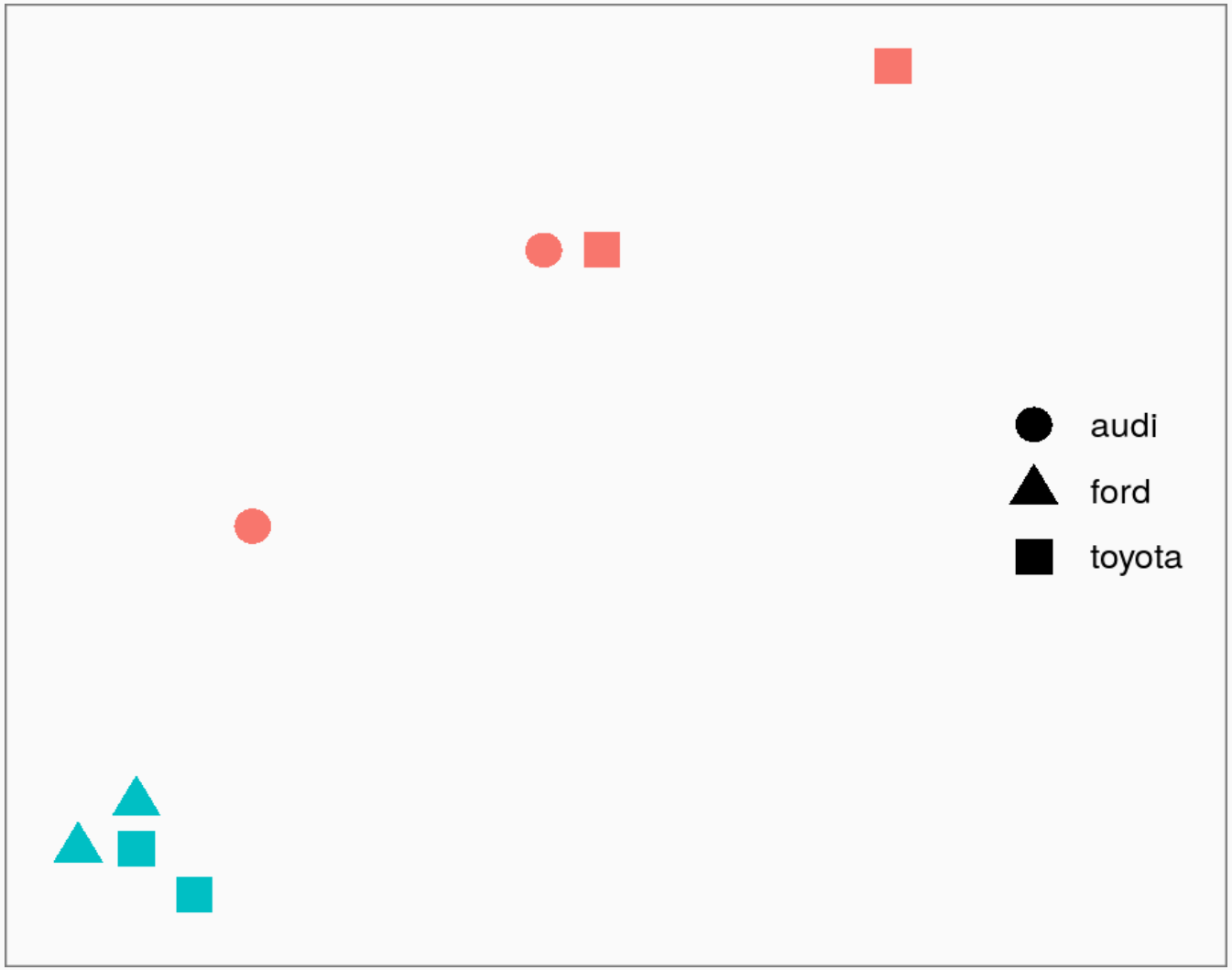
- Manufacturer
- Car Type (Class)
- City MPG
- Highway MPG



Guess the data behind this plot?

MPG Ratings of Cars

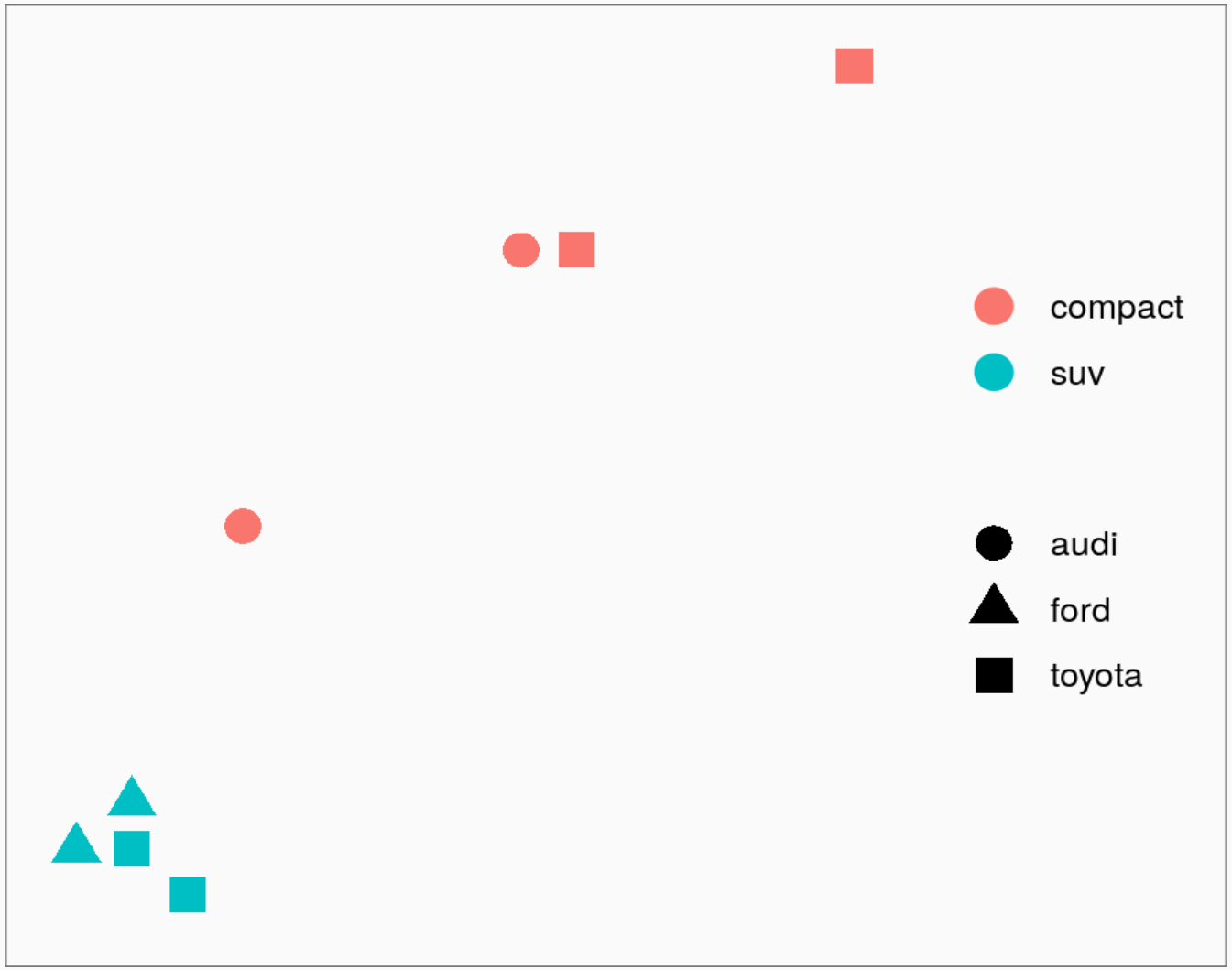
- Manufacturer
- Car Type (Class)
- City MPG
- Highway MPG



Guess the data behind this plot?

MPG Ratings of Cars

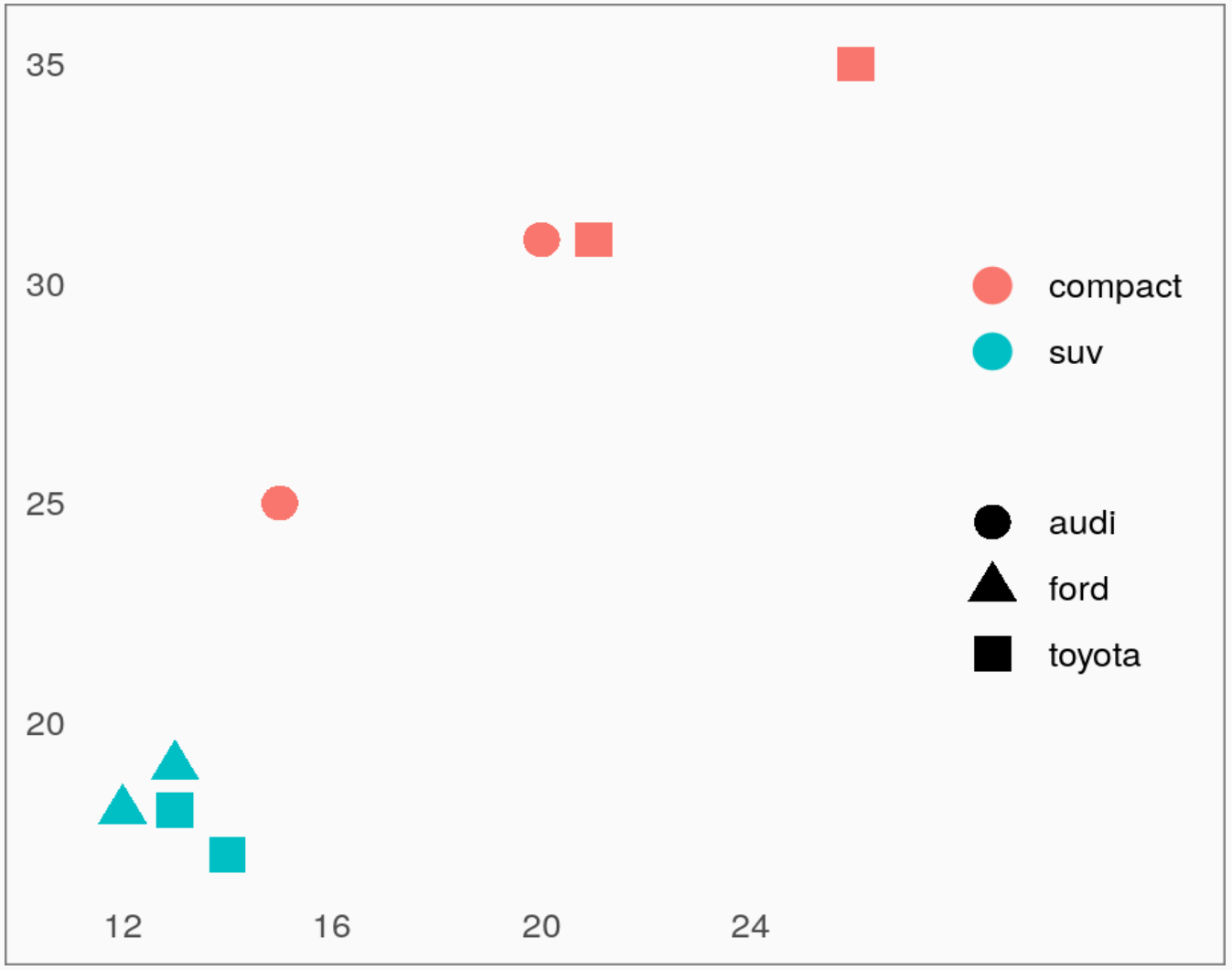
- Manufacturer
- Car Type (Class)
- City MPG
- Highway MPG



Guess the data behind this plot?

MPG Ratings of Cars

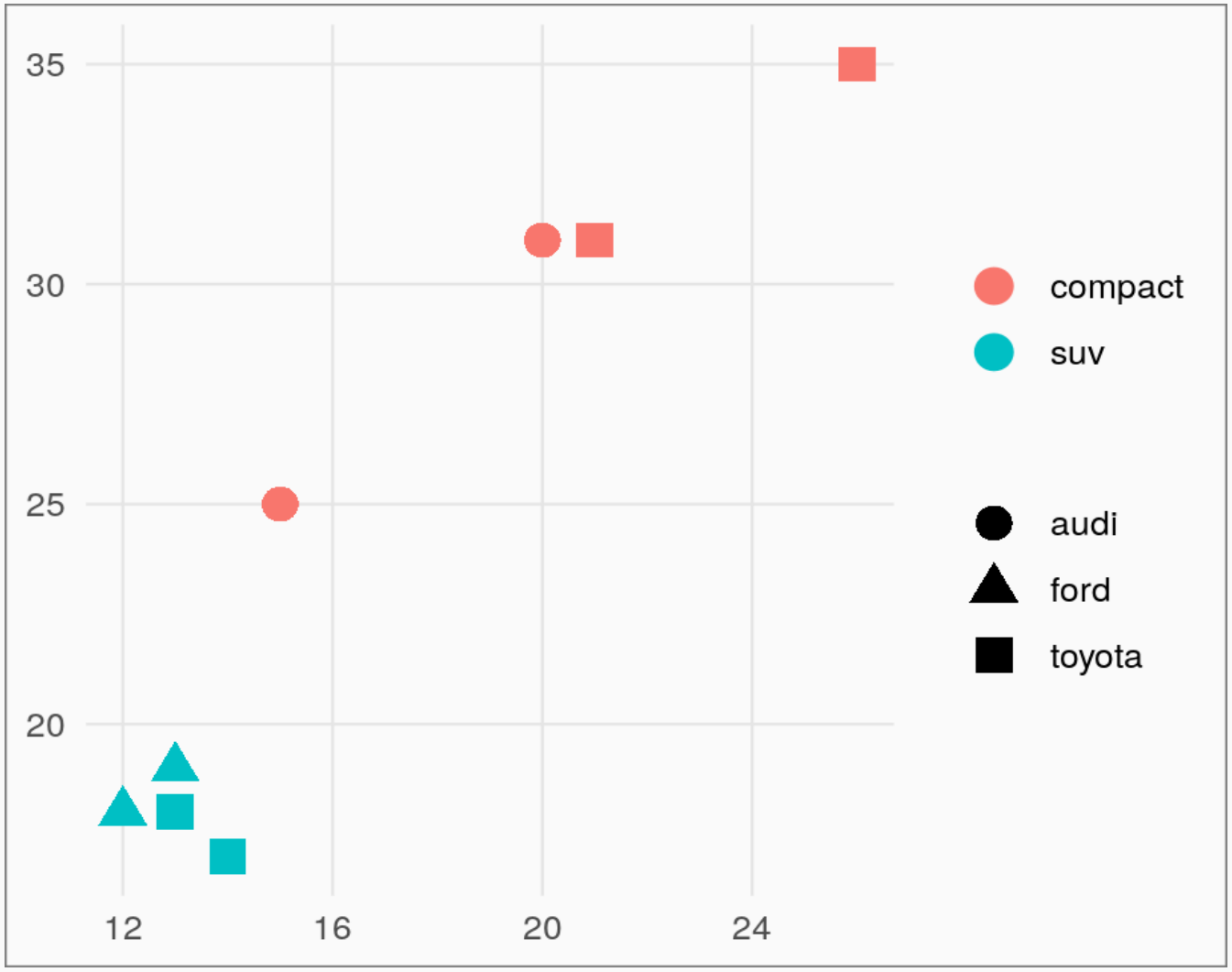
- Manufacturer
- Car Type (Class)
- City MPG
- Highway MPG



Guess the data behind this plot?

MPG Ratings of Cars

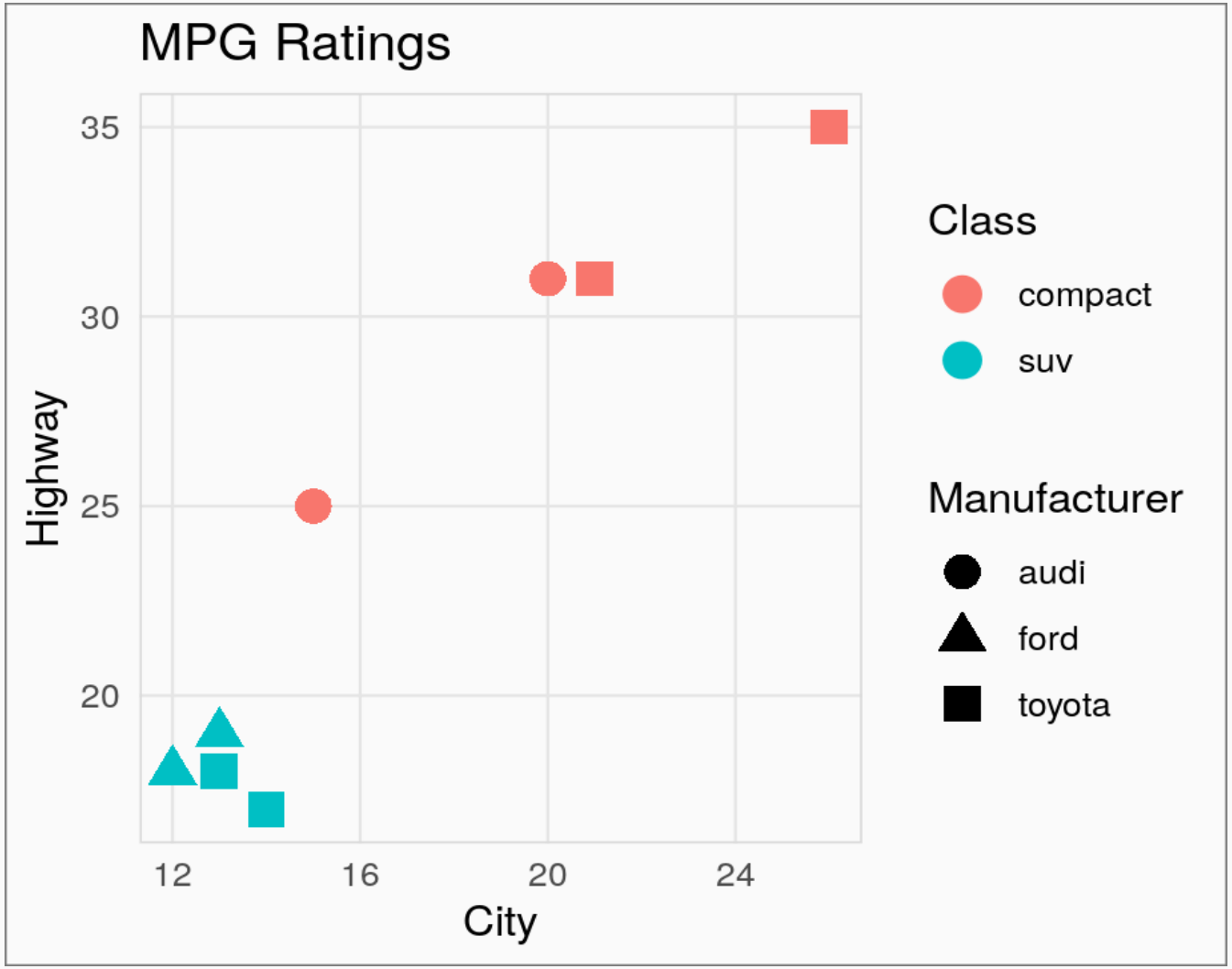
- Manufacturer
- Car Type (Class)
- City MPG
- Highway MPG



Guess the data behind this plot?

MPG Ratings of Cars

- Manufacturer
- Car Type (Class)
- City MPG
- Highway MPG



Guess the data behind this plot?

MPG Ratings of Cars

- Manufacturer
- Car Type (Class)
- City MPG
- Highway MPG

manufacturer	class	cty	hwy	model
audi	compact	20	31	a4
audi	compact	15	25	a4 quattro
ford	suv	12	18	expedition 2wd
ford	suv	13	19	explorer 4wd
toyota	suv	14	17	4runner 4wd
toyota	compact	21	31	camry solara
toyota	compact	26	35	corolla
toyota	suv	13	18	land cruiser wagon 4wd

How do we express visuals in words?

- **Data** to be visualized
- **Geometric objects** that appear on the plot
- **Aesthetic mappings** from data to visual component
- **Statistics** transform data on the way to visualization
- **Coordinates** organize location of geometric objects
- **Scales** define the range of values for aesthetics
- **Facets** group into subplots

gg is for Grammar of Graphics

Data

```
ggplot(data)
```

Tidy Data

1. Each variable forms a column
2. Each observation forms a row
3. Each observational unit forms a table

Start by asking

1. What information do I want to use in my visualization?
2. Is that data contained in one column/row for a given data point?

gg is for Grammar of Graphics

Data

```
ggplot(data)
```

country	1997	2002	2007
Canada	30.30584	31.90227	33.39014
China	1230.07500	1280.40000	1318.68310
United States	272.91176	287.67553	301.13995

```
# NOTE: using `tidyr` here, pivoting "longer"
tidy_pop ← gather(messy_pop, 'year', 'pop', -country)
```

country	year	pop
Canada	1997	30.306
China	1997	1230.075
United States	1997	272.912

gg is for Grammar of Graphics

Data

Aesthetics

```
+ aes()
```

Map data to visual elements or parameters

- year
- pop
- country

gg is for Grammar of Graphics

Data

Aesthetics

```
+ aes()
```

Map data to visual elements or parameters

- year → **x**
- pop → **y**
- country → *shape*, *color*, etc.

gg is for Grammar of Graphics

Data

Aesthetics

```
+ aes()
```

Map data to visual elements or parameters

```
aes(
  x = year,
  y = pop,
  color = country
)
```

gg is for Grammar of Graphics

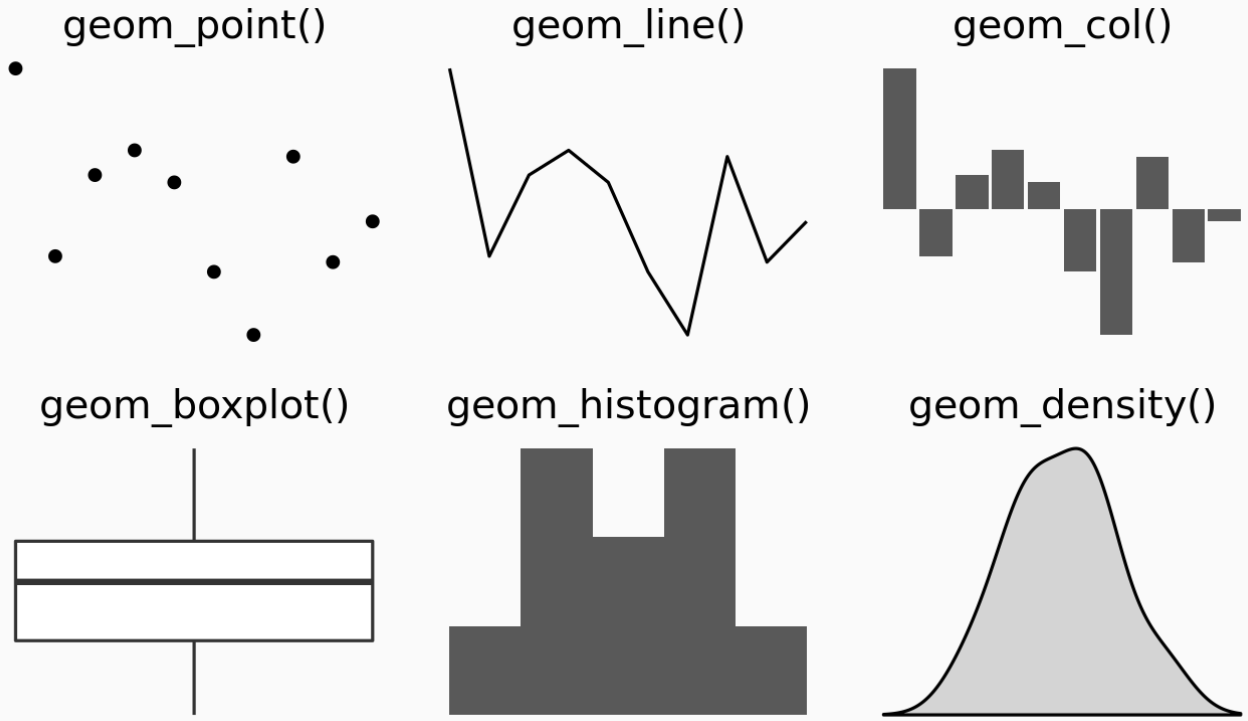
Data

Aesthetics

Geoms

```
+ geom_*()
```

Geometric objects displayed on the plot



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

```
+ geom_*()
```

Here are the some of the most widely used geoms

Type	Function
Point	`geom_point()`
Line	`geom_line()`
Bar	`geom_bar()`, `geom_col()`
Histogram	`geom_histogram()`
Regression	`geom_smooth()`
Boxplot	`geom_boxplot()`
Text	`geom_text()`
Vert./Horiz. Line	`geom_{vh}line()`
Count	`geom_count()`
Density	`geom_density()`

https://eric.netlify.com/2017/08/10/most-popular-ggplot2-geoms/

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

```
+ geom_*()
```

See http://ggplot2.tidyverse.org/reference/ for many more options

```
##  [1] "geom_abline"     "geom_area"       "geom_bar"        "geom_bin2d"
##  [5] "geom_blank"      "geom_boxplot"    "geom_col"        "geom_contour"
##  [9] "geom_count"      "geom_crossbar"   "geom_curve"      "geom_density"
## [13] "geom_density_2d" "geom_density2d"  "geom_dotplot"    "geom_errorbar"
## [17] "geom_errorbarh"  "geom_freqpoly"   "geom_hex"        "geom_histogram"
## [21] "geom_hline"      "geom_jitter"     "geom_label"      "geom_line"
## [25] "geom_linerange"  "geom_map"        "geom_path"       "geom_point"
## [29] "geom_pointrange" "geom_polygon"    "geom_qq"         "geom_qq_line"
## [33] "geom_quantile"   "geom_raster"     "geom_rect"       "geom_ribbon"
## [37] "geom_rug"        "geom_segment"    "geom_sf"         "geom_sf_label"
## [41] "geom_sf_text"    "geom_smooth"     "geom_spoke"      "geom_step"
## [45] "geom_text"       "geom_tile"       "geom_violin"     "geom_vline"
```

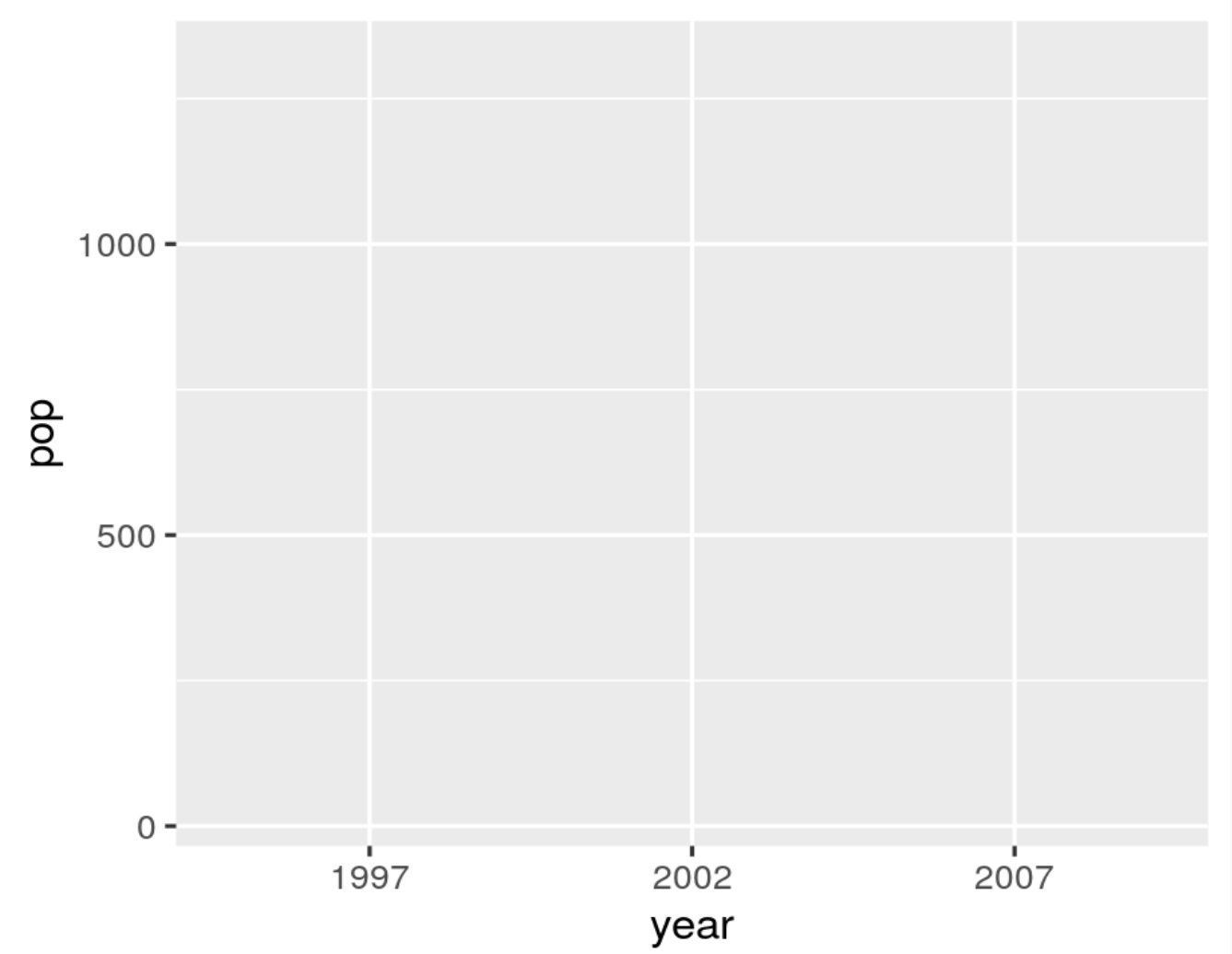
Or just start typing `geom_` in RStudio

Our first plot!

```
ggplot(tidy_pop)
```

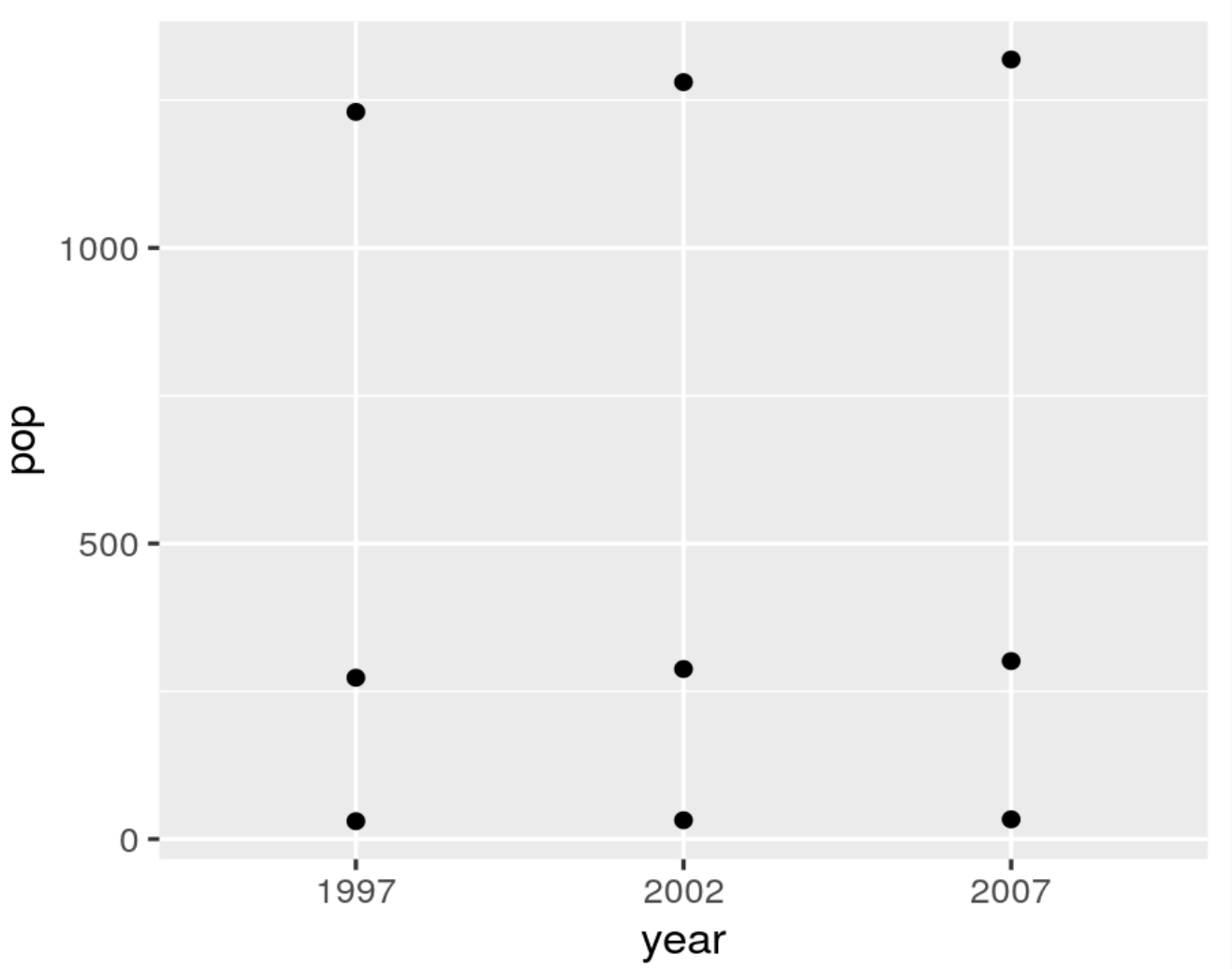
Our first plot!

```
ggplot(tidy_pop) +
  aes(x = year,
      y = pop)
```



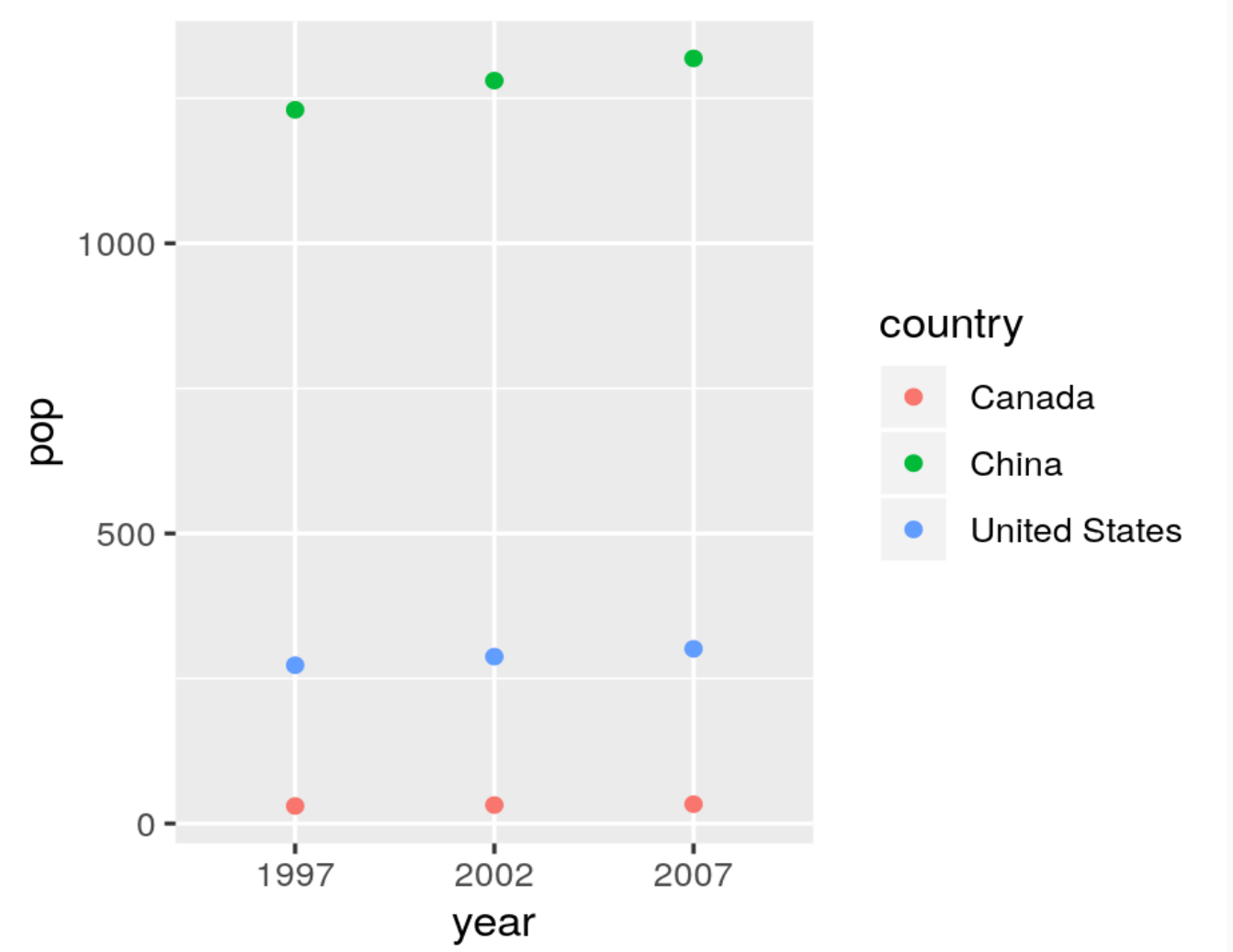
Our first plot!

```
ggplot(tidy_pop) +
  aes(x = year,
      y = pop) +
  geom_point()
```



Our first plot!

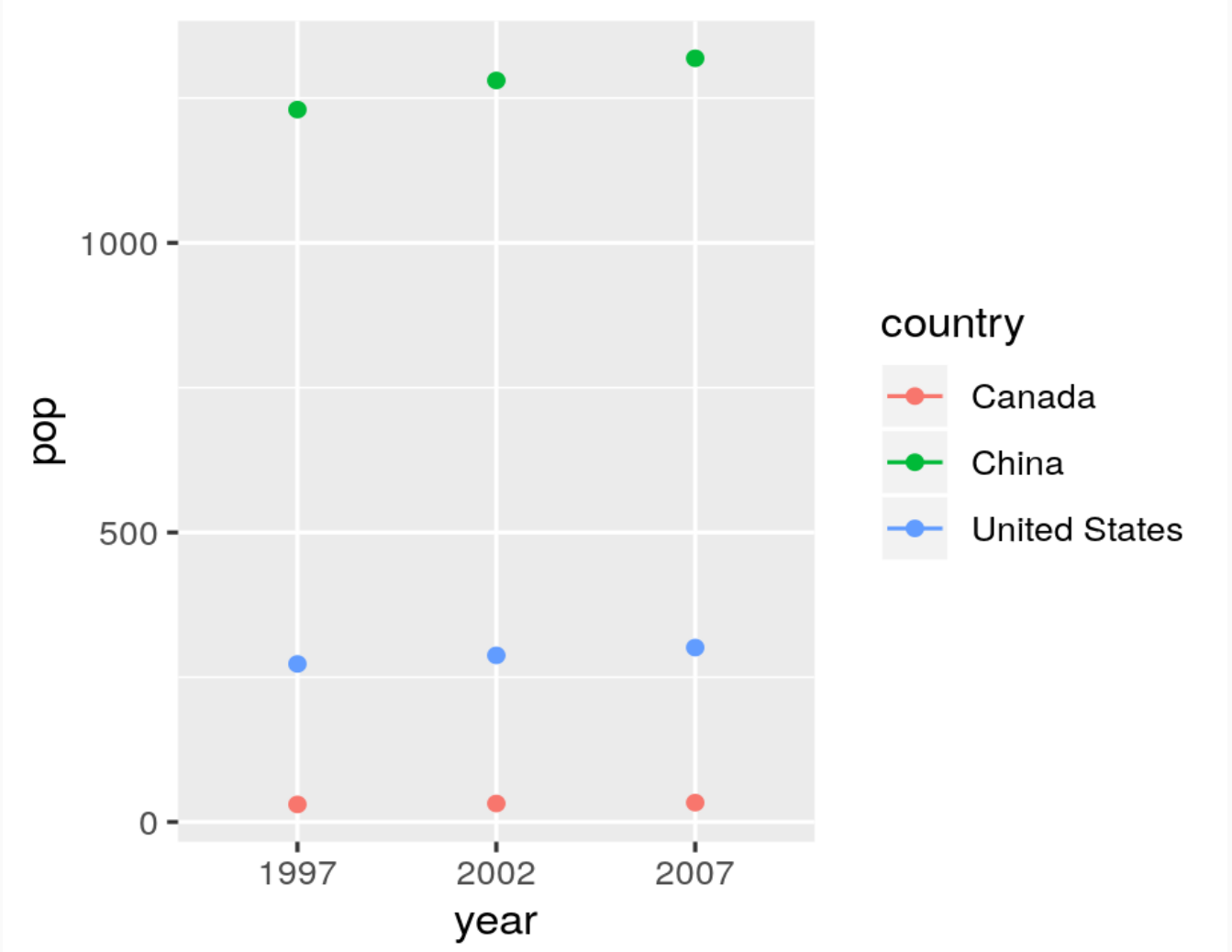
```
ggplot(tidy_pop) +
  aes(x = year,
      y = pop,
      color = country) +
  geom_point()
```



Our first plot!

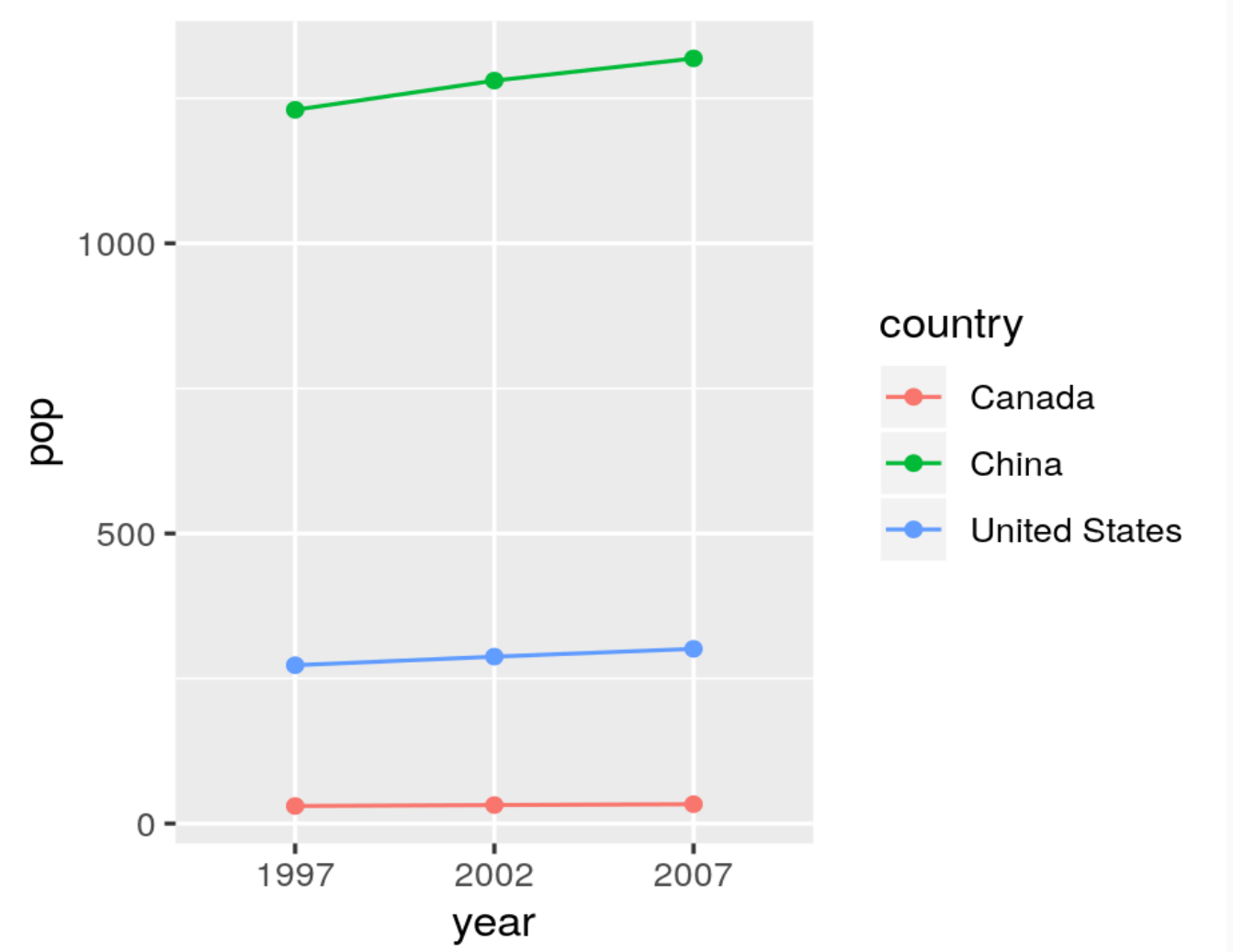
```
ggplot(tidy_pop) +
  aes(x = year,
      y = pop,
      color = country) +
  geom_point() +
  geom_line()
```

geom_path: Each group consists of only one observation. Do you need to adjust the group aesthetic?



Our first plot!

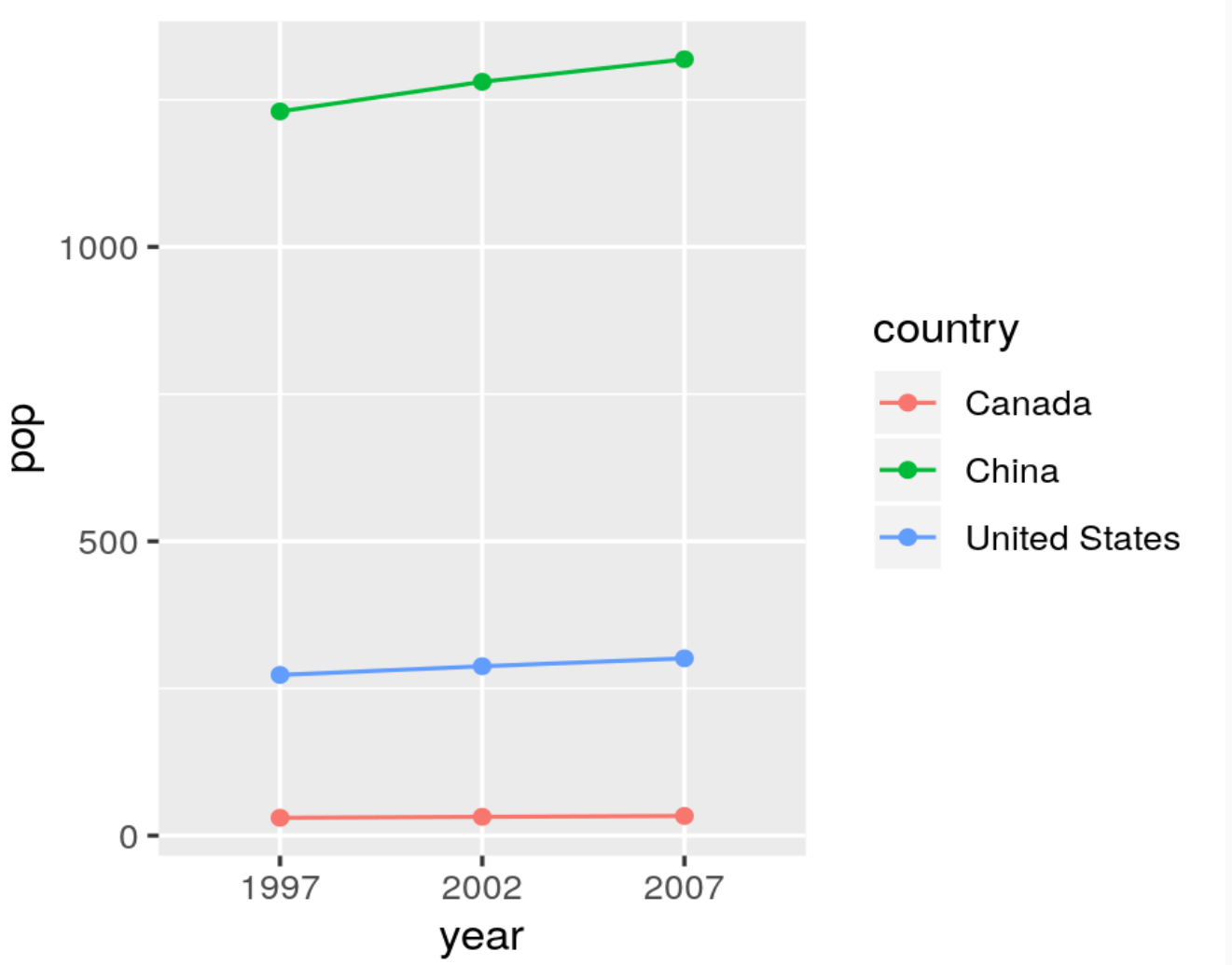
```
ggplot(tidy_pop) +
  aes(x = year,
      y = pop,
      color = country) +
  geom_point() +
  geom_line(
    aes(group = country))
```



Our first plot!

```
g ← ggplot(tidy_pop) +
  aes(x = year,
      y = pop,
      color = country) +
  geom_point() +
  geom_line(
    aes(group = country))

g
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

```
+ geom_*()
```

```
geom_*(mapping, data, stat, position)
```

- `data` Geoms can have their own data
  - Has to map onto global coordinates
- `map` Geoms can have their own aesthetics
  - Inherits global aesthetics
  - Have geom-specific aesthetics
    - `geom_point` needs `x` and `y`, optional `shape`, `color`, `size`, etc.
    - `geom_ribbon` requires `x`, `ymin` and `ymax`, optional `fill`
  - `?geom_ribbon`

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

```
+ geom_*()
```

```
geom_*(mapping, data, stat, position)
```

- `stat` Some geoms apply further transformations to the data
  - All respect `stat = 'identity'`
  - Ex: `geom_histogram` uses `stat_bin()` to group observations
- `position` Some adjust location of objects
  - `'dodge'`, `'stack'`, `'jitter'`

gg is for Grammar of Graphics

Data

Aesthetics

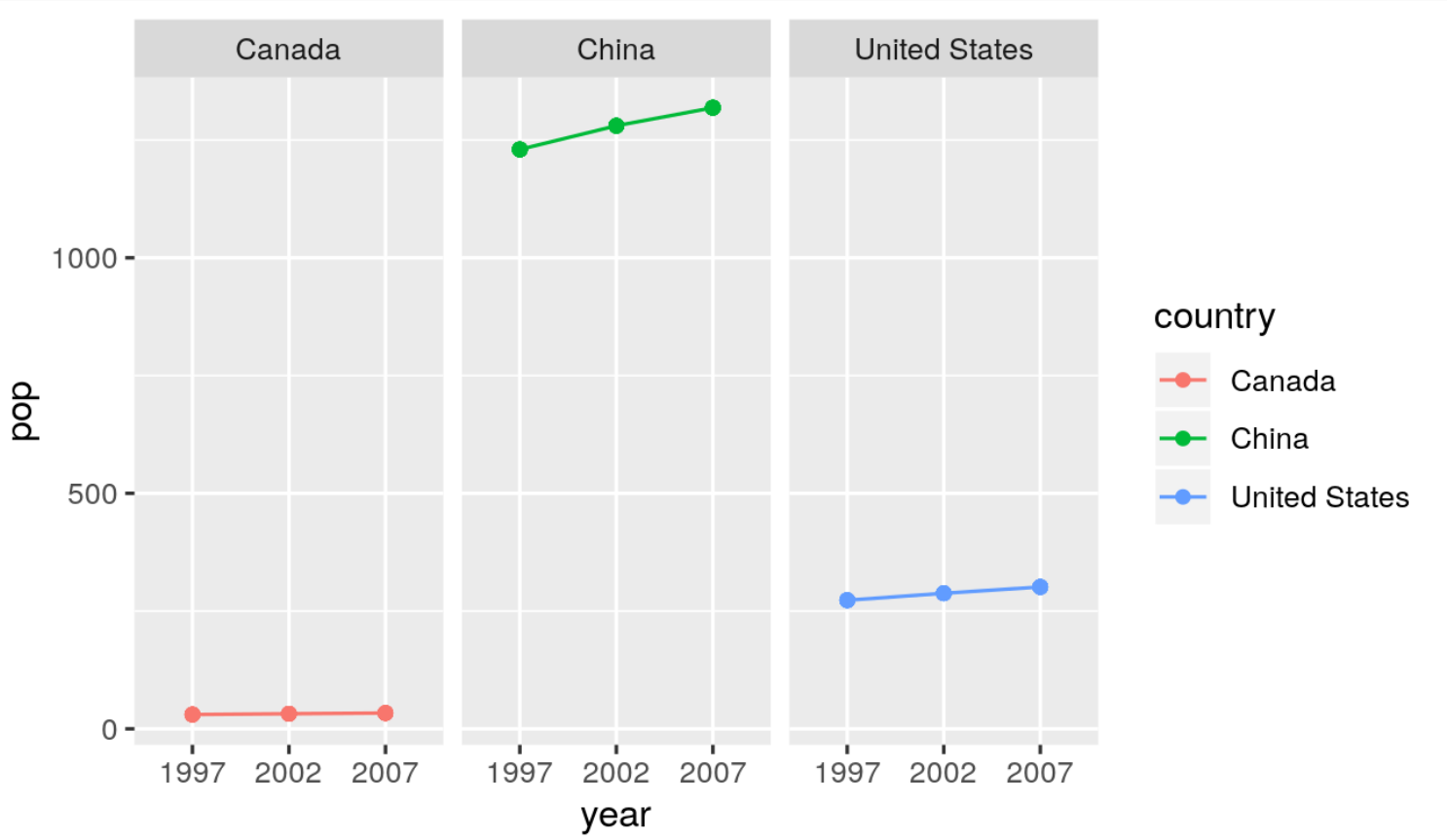
Geoms

Facet

```
+facet_wrap()

+facet_grid()
```

```
g + facet_wrap(~ country)
```



gg is for Grammar of Graphics

Data

Aesthetics

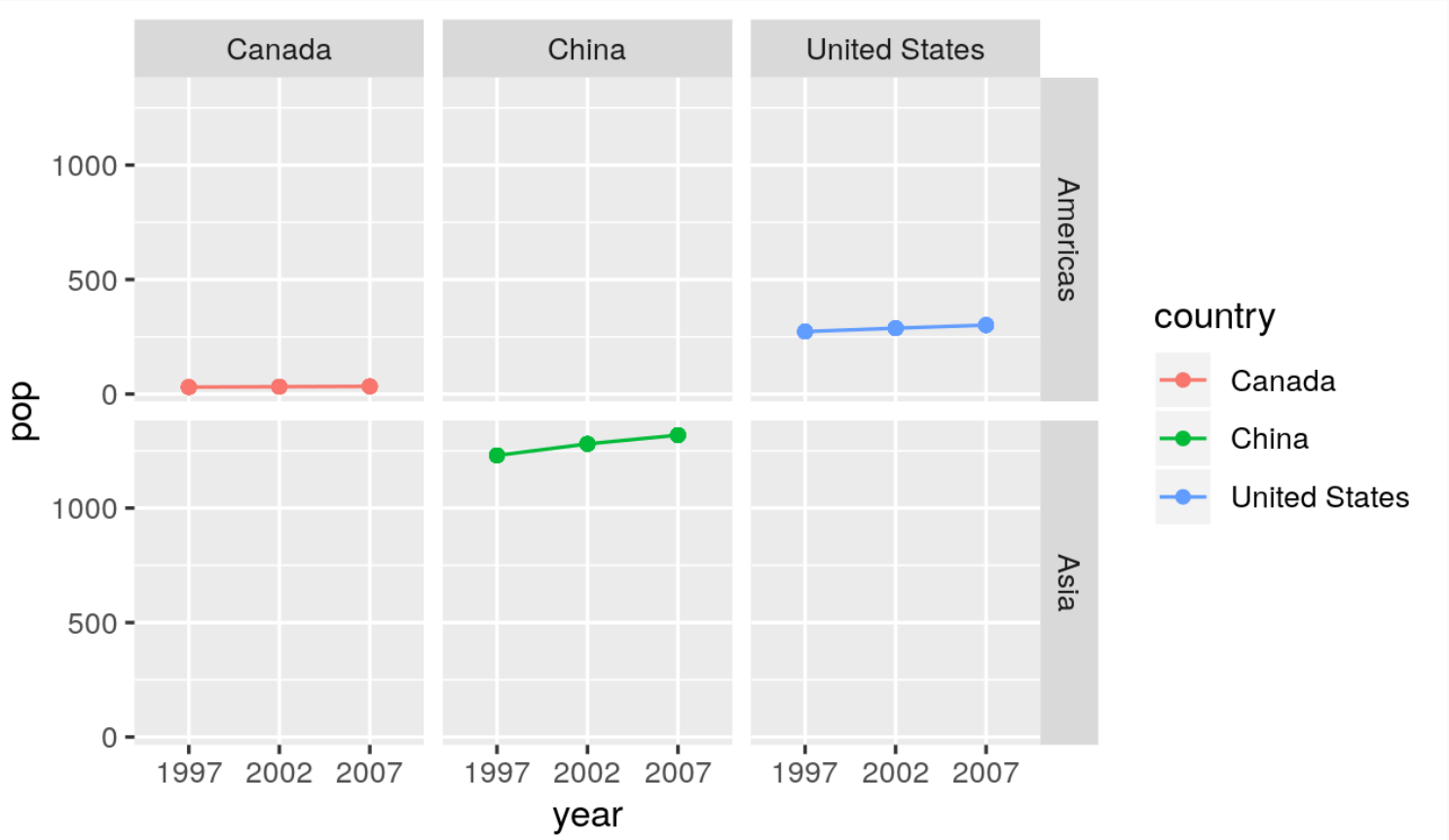
Geoms

Facet

```
+facet_wrap()

+facet_grid()
```

```
g + facet_grid(continent ~ country)
```



gg is for Grammar of Graphics

Data

Aesthetics

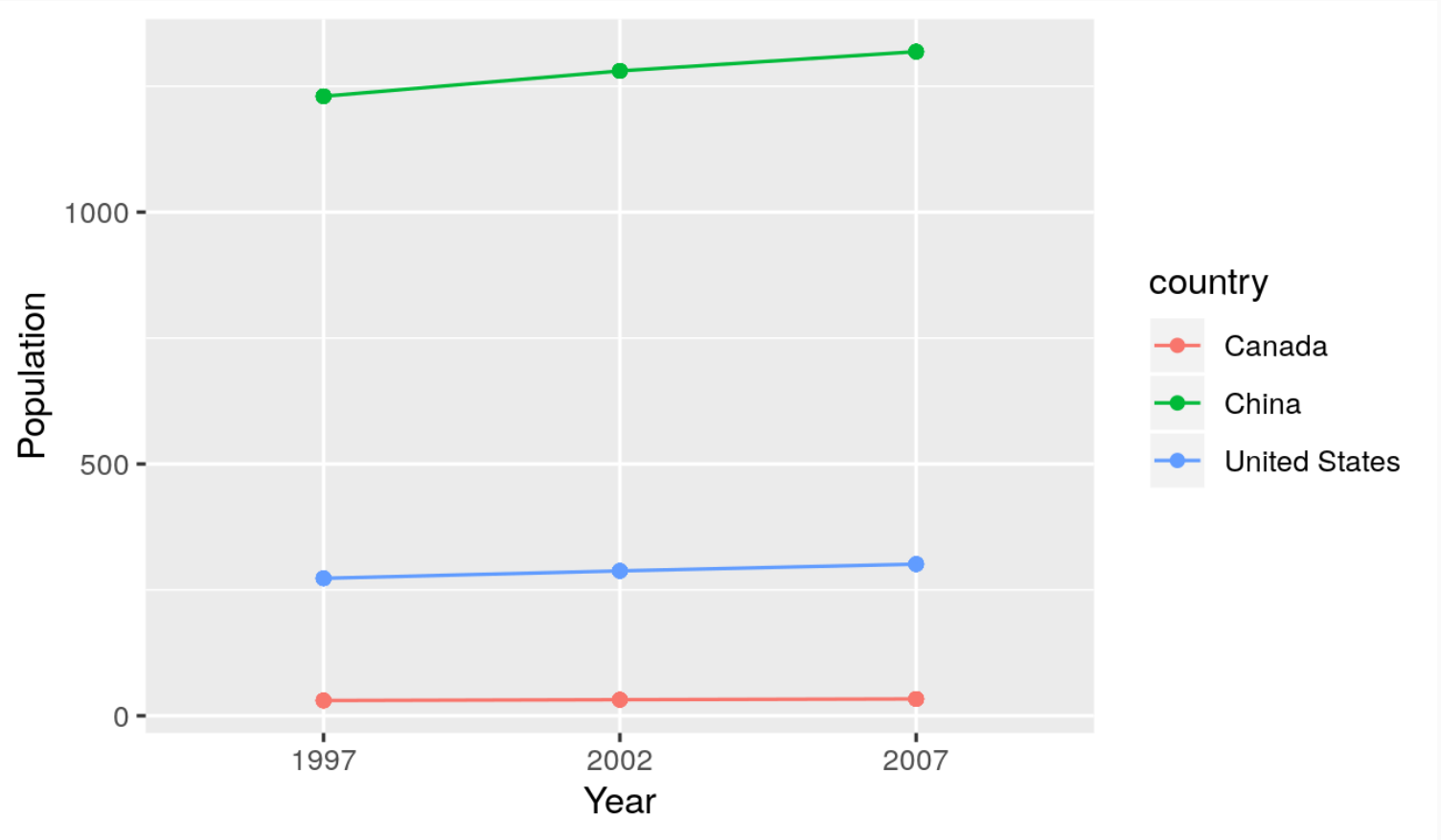
Geoms

Facet

Labels

```
+ labs()
```

```
g + labs(x = "Year", y = "Population")
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

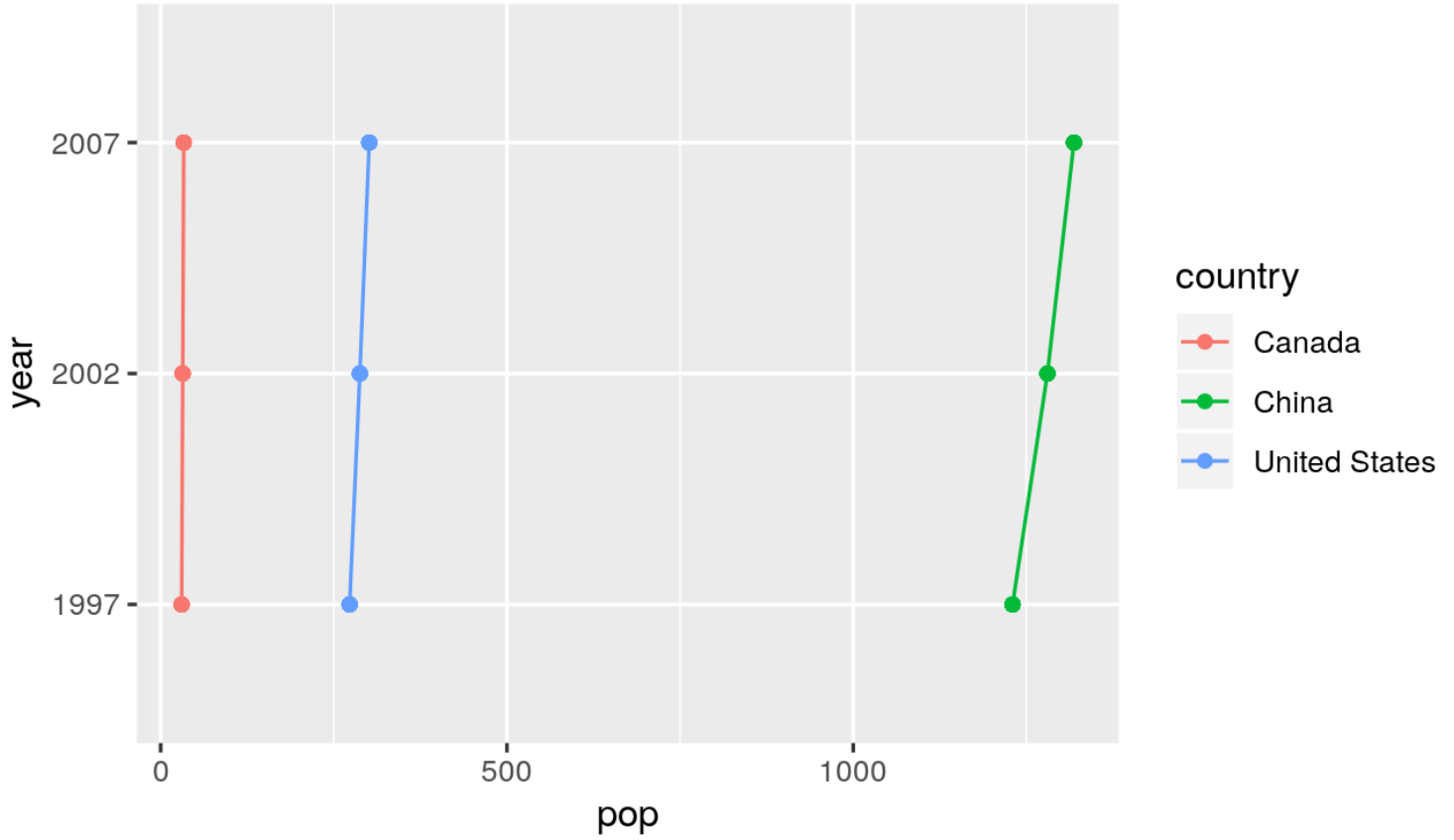
Facet

Labels

Coords

```
+ coord_*()
```

```
g + coord_flip()
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

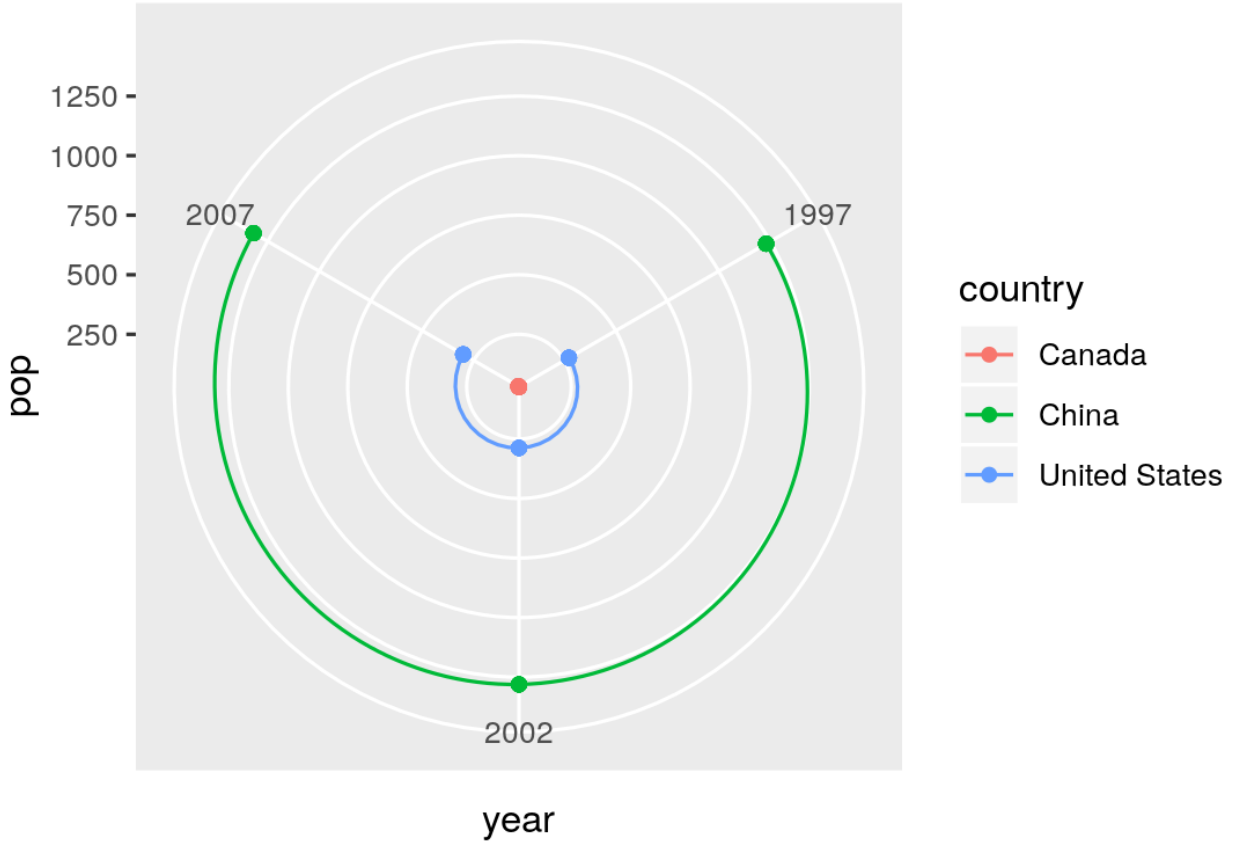
Facet

Labels

Coords

```
+ coord_*()
```

```
g + coord_polar()
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

Coords

Scales

```
+ scale_*_*()
```

`scale` + `_` + `<aes>` + `_` + `<type>` + `()`

What parameter do you want to adjust? → `<aes>`
What type is the parameter? → `<type>`

- I want to change my discrete x-axis
  `scale_x_discrete()`
- I want to change range of point sizes from continuous variable
  `scale_size_continuous()`
- I want to rescale y-axis as log
  `scale_y_log10()`
- I want to use a different color palette
  `scale_fill_discrete()`
  `scale_color_manual()`

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

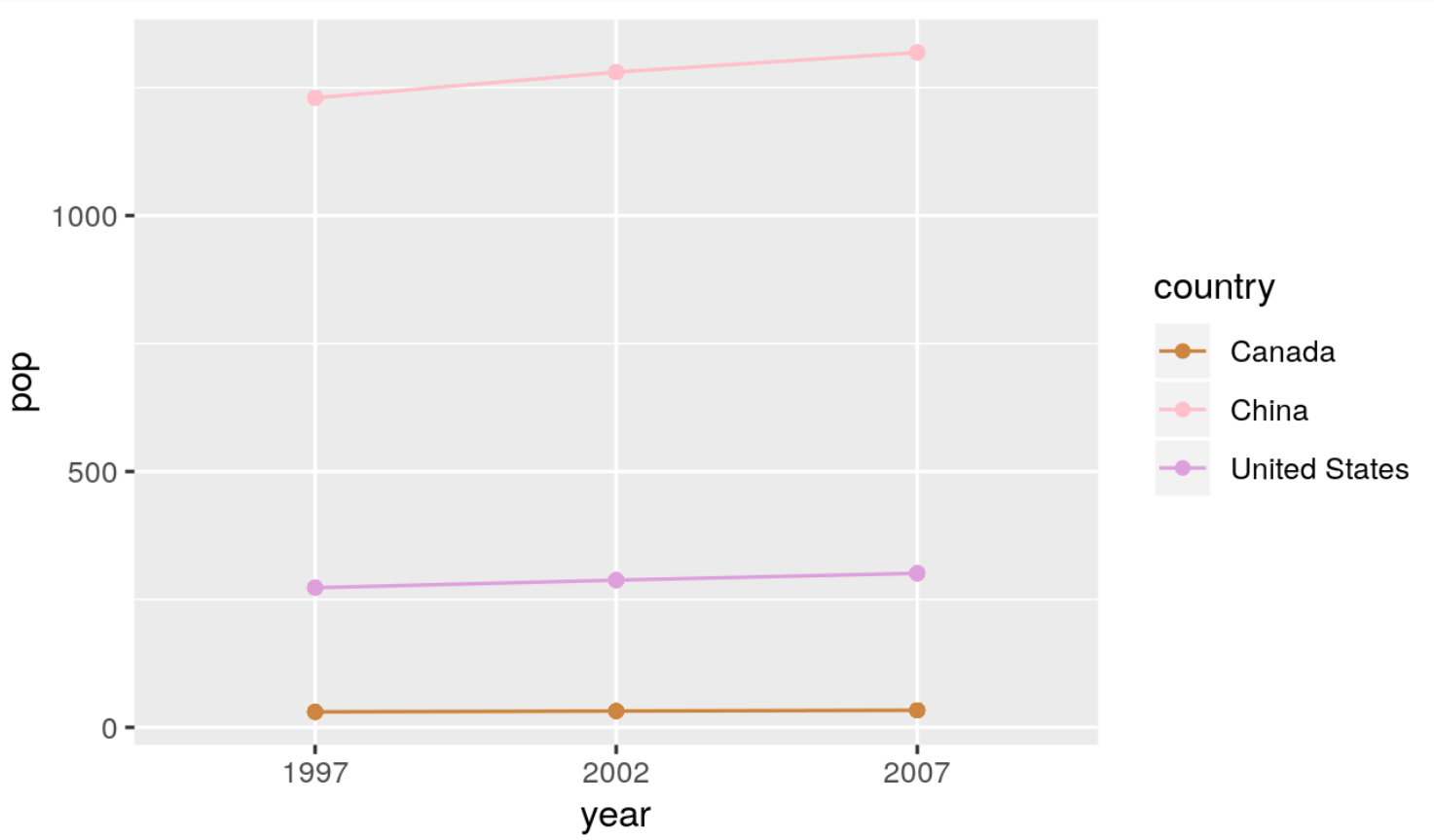
Labels

Coords

Scales

```
+ scale_*_*()
```

```
g + scale_color_manual(values = c("peru", "pink", "plum"))
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

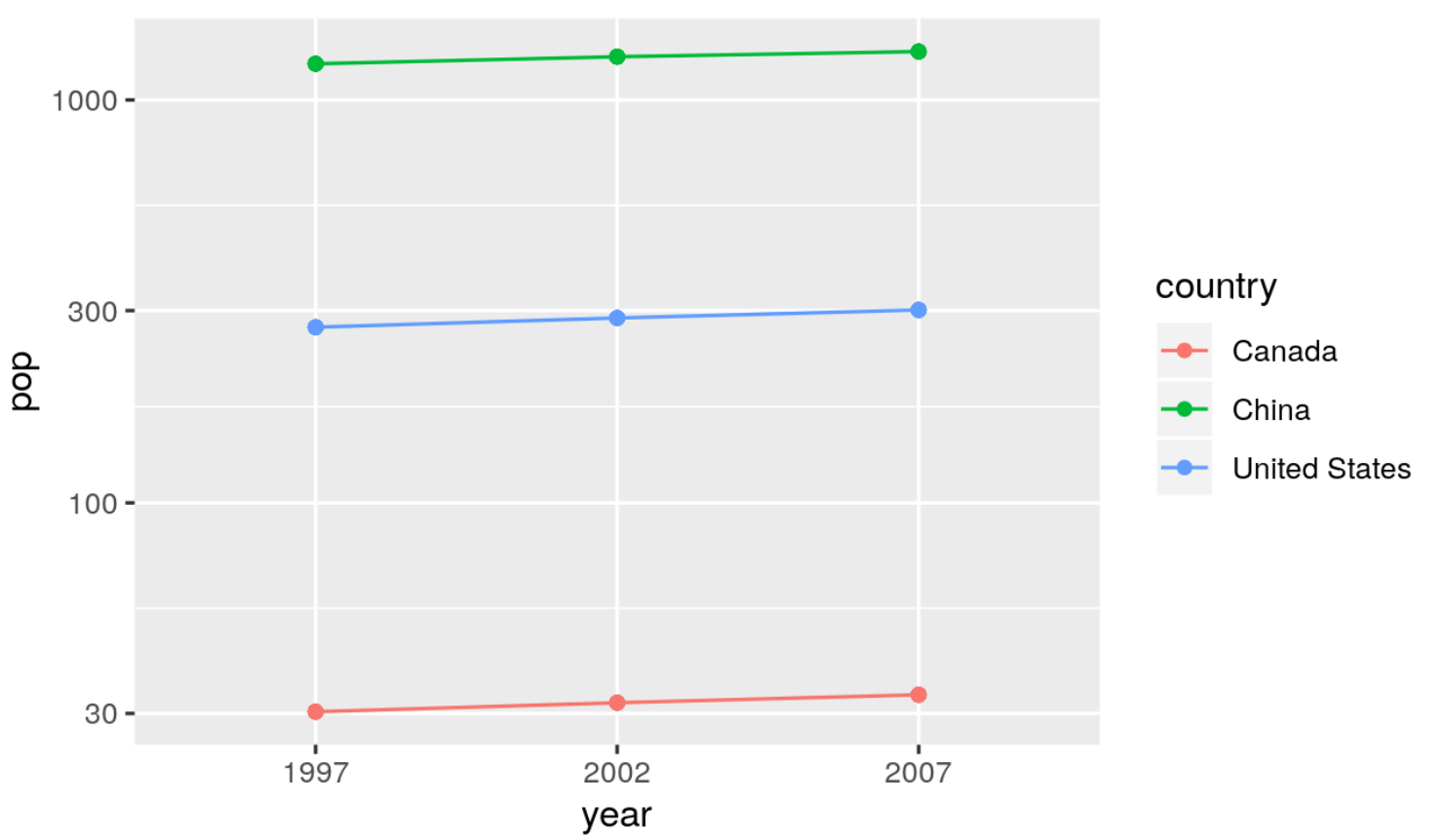
Labels

Coords

Scales

```
+ scale_*_*()
```

```
g + scale_y_log10()
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

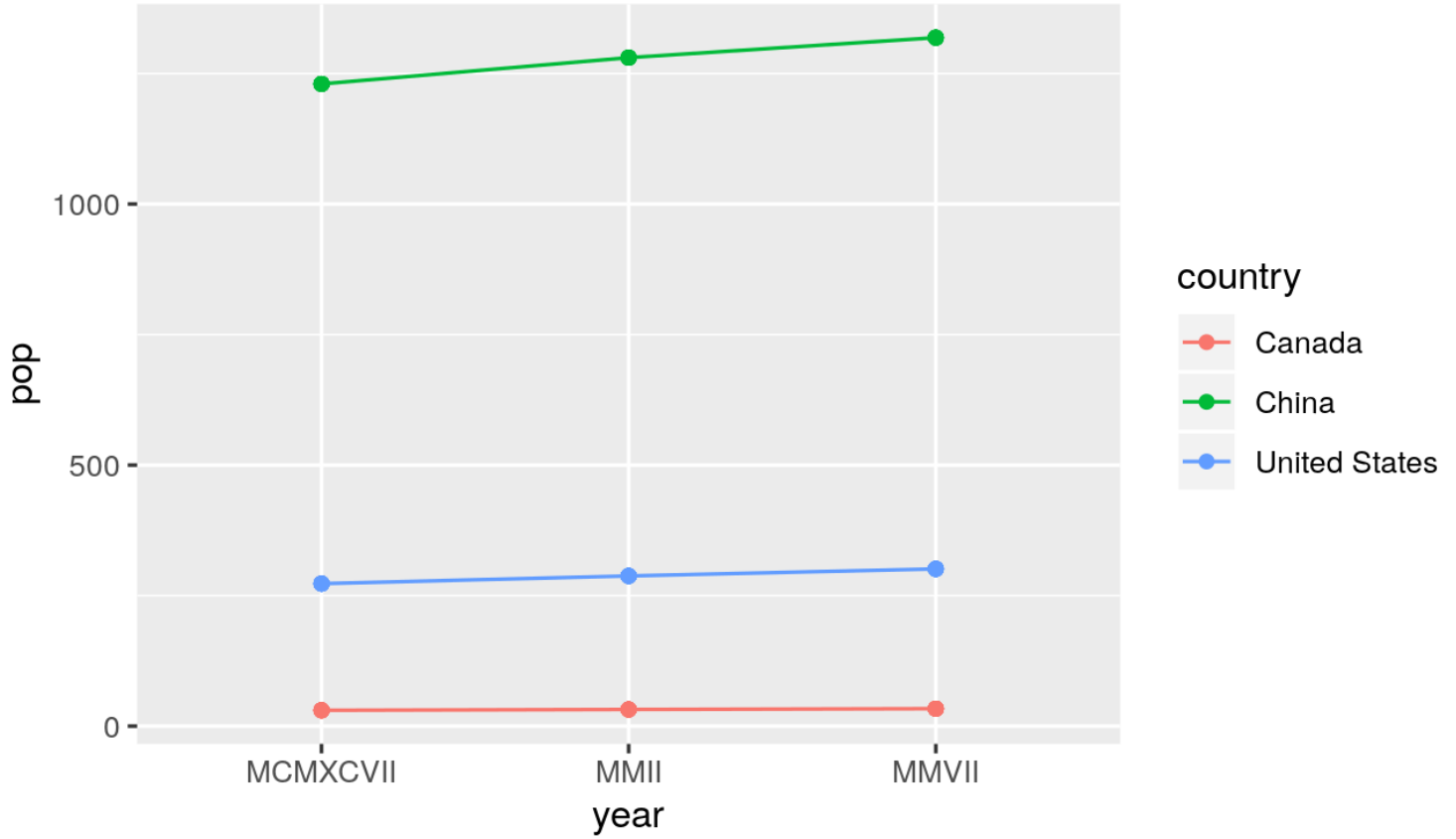
Labels

Coords

Scales

```
+ scale_*_*()
```

```
g + scale_x_discrete(labels = c("MCMXCVII", "MMII", "MMVII"
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

Coords

Scales

Theme

```
+ theme()
```

Change the appearance of plot decorations
i.e. things that aren't mapped to data

A few "starter" themes ship with the package

- `g + theme_bw()`
- `g + theme_dark()`
- `g + theme_gray()`
- `g + theme_light()`
- `g + theme_minimal()`

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

Coords

Scales

Theme

```
+ theme()
```

Huge number of parameters, grouped by plot area:

- Global options: `line`, `rect`, `text`, `title`
- `axis`: x-, y- or other axis title, ticks, lines
- `legend`: Plot legends
- `panel`: Actual plot area
- `plot`: Whole image
- `strip`: Facet labels

Check out3 emilyriederer/ugliest-ggplot-theme.R!

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

Coords

Scales

Theme

```
+ theme()
```

Theme options are supported by helper functions:

- `element_blank()` removes the element
- `element_line()`
- `element_rect()`
- `element_text()`

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

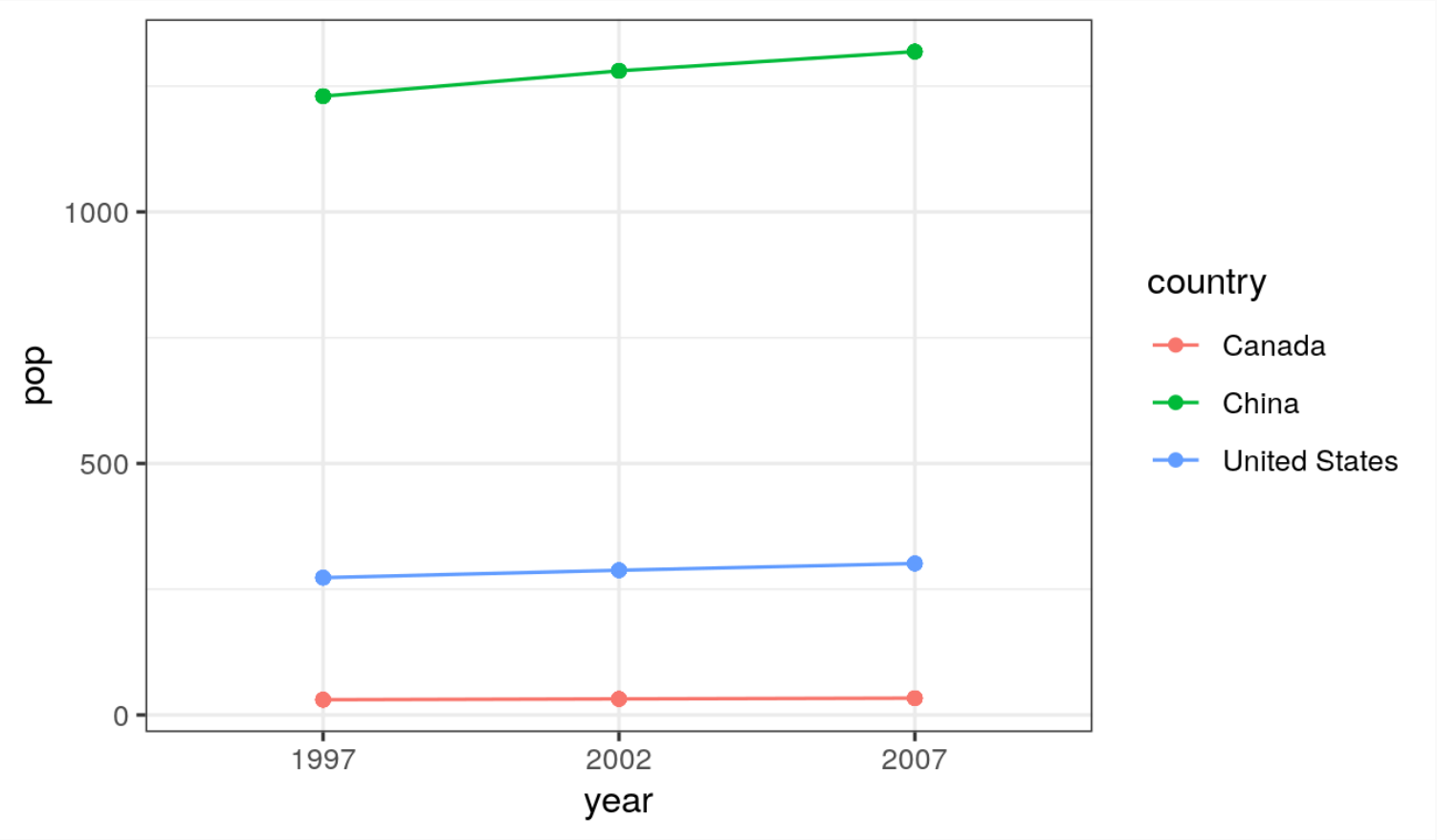
Coords

Scales

Theme

```
+ theme()
```

```
g + theme_bw()
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

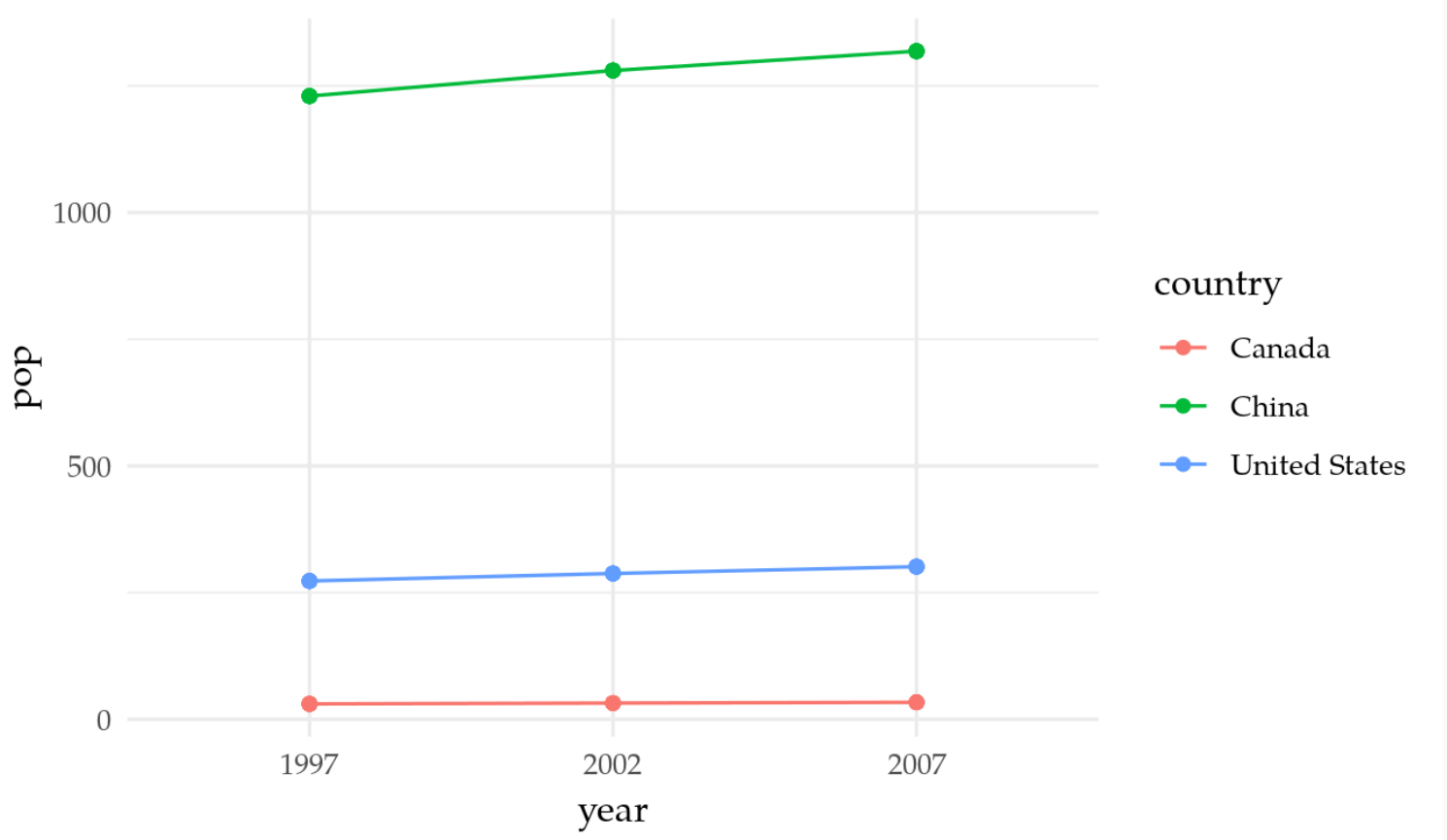
Coords

Scales

Theme

```
+ theme()
```

```
g + theme_minimal() + theme(text = element_text(family = "Palatino"))
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

Coords

Scales

Theme

```
+ theme()
```

You can also set the theme globally with `theme_set()`

```
my_theme ← theme_bw() +
  theme(
    text = element_text(family = "Palatino", size = 12),
    panel.border = element_rect(colour = 'grey80'),
    panel.grid.minor = element_blank()
  )

theme_set(my_theme)
```

All plots will now use this theme!

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

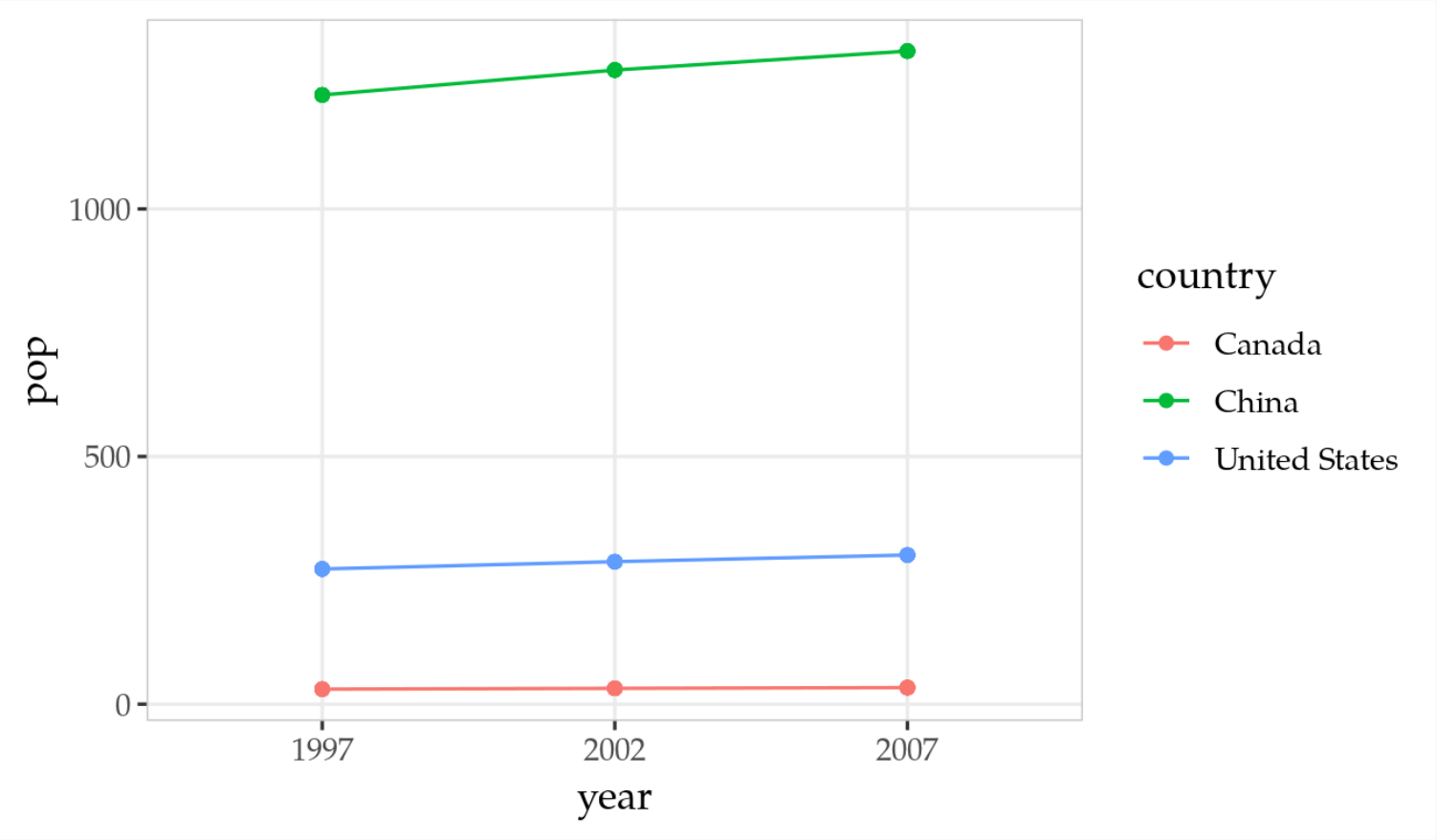
Labels

Coords

Scales

Theme

```
g
```




```
+ theme()
```

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

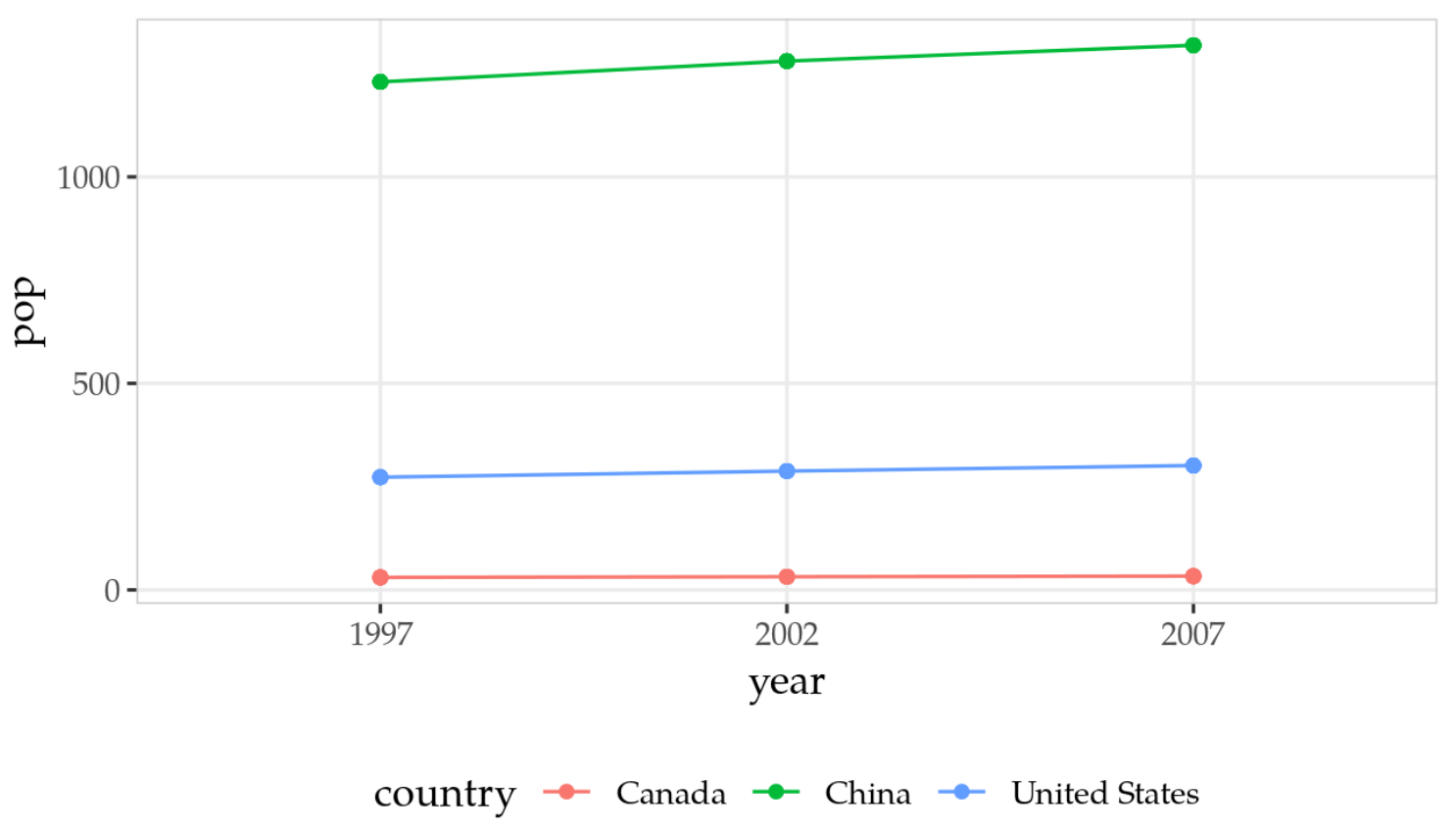
Coords

Scales

Theme

```
+ theme()
```

```
g + theme(legend.position = 'bottom')
```



Save Your Work

To save your plot, use **ggsave**

```
ggsave(
  filename = "my_plot.png",
  plot = my_plot,
  width = 10,
  height = 8,
  dpi = 100,
  device = "png"
)
```


You have the power!

Getting started with dplyr

Inspired by dplyr slides

What is *dplyr*?

dplyr combines:

1. tables
2. verbs that do things to tables
3. pipes that string together tables and verbs

Most tasks can be written as a sequence of simple verbs.

The pipe %>%

I want to apply `A`, `B`, then `C` on `df`.

without pipes

```
C(B(A(df)))  # like, please don't.
```

with pipes

```
df %>%  # start with df
  A() %>%  # then apply A
  B() %>%  # then apply B
  C()  # then apply C
```

Single-table verbs

Most comon set of verbs:

- `mutate()`: new variables as functions of existing variables
- `select()`: select subset of variables (columns)
- `filter()`: select subset of observations (rows)
- `arrange()`: reorder observations
- `summarize()`: reduce multiple observations to single summary

Combine with `group_by()` which allows performing operations "by group"

Single-table verbs

country	1997	2002	2007
Canada	30.306	31.902	33.390
China	1230.075	1280.400	1318.683
United States	272.912	287.676	301.140

Examples operating on the table `messy_pop`...

Single-table verbs

Mutate

```
mutate( ... )
```

country	1997	2002	2007
Canada	30.306	31.902	33.390
China	1230.075	1280.400	1318.683
United States	272.912	287.676	301.140

```
messy_pop %>%
  mutate(average=(`1997` + `2002` + `2007`) / 3)
```

country	1997	2002	2007	average
Canada	30.306	31.902	33.390	31.866
China	1230.075	1280.400	1318.683	1276.386
United States	272.912	287.676	301.140	287.242

Single-table verbs

Mutate

Select

```
select( ... )
```

country	1997	2002	2007
Canada	30.306	31.902	33.390
China	1230.075	1280.400	1318.683
United States	272.912	287.676	301.140

```
messy_pop %>%
  select(`1997`, `2007`)
```

1997	2007
30.306	33.390
1230.075	1318.683
272.912	301.140

Single-table verbs

Mutate

Select

Filter

`filter( ... )`

country	1997	2002	2007
Canada	30.306	31.902	33.390
China	1230.075	1280.400	1318.683
United States	272.912	287.676	301.140

```
messy_pop %>%
  filter(`2002` < 500)
```

country	1997	2002	2007
Canada	30.306	31.902	33.39
United States	272.912	287.676	301.14

Single-table verbs

Mutate

Select

Filter

Arrange

```
arrange( ... )
```

country	1997	2002	2007
Canada	30.306	31.902	33.390
China	1230.075	1280.400	1318.683
United States	272.912	287.676	301.140

```
messy_pop %>%
  arrange(desc(`2002`))
```

country	1997	2002	2007
China	1230.075	1280.400	1318.683
United States	272.912	287.676	301.140
Canada	30.306	31.902	33.390

Single-table verbs

Mutate

Select

Filter

Arrange

Summarize

```
summarize( ... )
```

country	1997	2002	2007
Canada	30.306	31.902	33.390
China	1230.075	1280.400	1318.683
United States	272.912	287.676	301.140

```
messy_pop %>%
  summarize(`1997` = mean(`1997`), `2002` = mean(`2002`))
```

1997	2002
511.098	533.326

Single-table verbs

Mutate

Select

Filter

Arrange

Summarize

Group by

```
group_by( ... )
```

```
americas %>%  # has 48 rows
  sample_n(6)
```

country	continent	year	lifeExp	pop	gdpPercap
United States	Americas	1952	68.440	157553000	13990.482
Canada	Americas	1952	68.750	14785584	11367.161
Mexico	Americas	1952	50.789	30144317	3478.126
Canada	Americas	1997	78.610	30305843	28954.926
Mexico	Americas	2007	76.195	108700891	11977.575
Ecuador	Americas	1977	61.310	7278866	6679.623

Single-table verbs

Mutate

Select

Filter

Arrange

Summarize

Group by

```
group_by( ... )
```

```
americas %>%
  summarize(lifeExp=mean(lifeExp), gdbPercap=mean(gdpPercap
  arrange(lifeExp)
```

lifeExp	gdbPercap
69.152	15532.41

Single-table verbs

Mutate

Select

Filter

Arrange

Summarize

Group by

```
group_by( ... )
```

```
americas %>%
  group_by(country) %>%
  summarize(lifeExp=mean(lifeExp), gdbPercap=mean(gdpPercap
  arrange(lifeExp)
```

country	lifeExp	gdbPercap
Ecuador	62.81683	5733.625
Mexico	65.40883	7724.113
United States	73.47850	26261.151
Canada	74.90275	22410.746

Two-table verbs

When we have multiple tables contributing to an analysis...

- Mutating joins: Add variables from one table to another with matching columns
- Filtering joins: Keep/remove observations using matches in another table
- Set operations: Treat rows as elements of a mathematical set

See `vignette("two-table")` for more details...

Tidyverse and other packages

Not enough time to go over all the great packages...

Tidyverse

- `tidyr`: tidy the messiest of data
- `purrr`: take apply and Reduce to the next level
- `stringr`: sane string handling (have you ever tried `"hello" + "world"`?)
- `forcats`: sane factors and handy releveling (R's most confusing datatype)
- `readxl`: when your wet-lab collaborator gives you an Excel spreadsheet
- `broom`: turn statistical/ML models into tidy tables (*sorry I wasn't able to do this*!)

Other packages

- ggplot2 extensions (link): especially `cowplot` or `egg` for multi-plot figures
- `data.table` and/or `sparklyr`: when you have to work with really large data
- Bioconductor (link): open source software for bioinformatics

Broom + tidyr + purrr can be used to do some serious cross-validation: see tutorial

Last tips and tricks, other resources

- Subsample your data (`dplyr::sample_{frac,n}`) when testing code
- Simulate random data to gain intuition/check assumptions
  - Especially as a sanity check when thinking about weird distributions for GCB 533
- Look at your data when analyzing it (pathologic cases)
- Organizing analyses (Noble et al 2009)
- Learn to use version control (slides)
- Ask for help!

Questions and practicum