Una introducción “amigable” al uso de `ggplot2`

Fernanda Miron

Material de:
https://github.com/gadenbuie/gentle-ggplot2/tree/main

Why *ggplot2*?

My personal reasons

- Functional data visualization
    1. Wrange data
    2. Map data to visual elements
    3. Tweak scales, guides, axis, labels, theme
- Easy to reason about how data drives visualization
- Easy to iterate
- Easy to be consistent

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

G is for getting started

Easy: install the tidyverse

```
install.packages('tidyverse')
```

Medium: install just `ggplot2`

```
install.pacakages('ggplot2')
```

Expert: install from GitHub

```
devtools :install_github('tidyverse/ggplot2')
```

G is for getting started

Load the tidyverse

```r
library(tidyverse)
```

```
## ── Attaching packages ──────────────────────────────────────────

## ✓ ggplot2 3.1.0     ✓ purrr   0.2.5
## ✓ tibble  1.4.2     ✓ dplyr   0.7.7
## ✓ tidyr   0.8.1     ✓ stringr 1.3.1
## ✓ readr   1.1.1     ✓ forcats 0.3.0

## ── Conflicts ─────────────────────────────────────────────────────
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

G is for getting started

Other packages you'll need for this adventure

We'll use an excerpt of the gapminder dataset provided by the `gapminder` package by Jenny Bryan.

https://github.com/jennybc/gapminder

```r
# install.packages("gapminder")
library(gapminder)
```

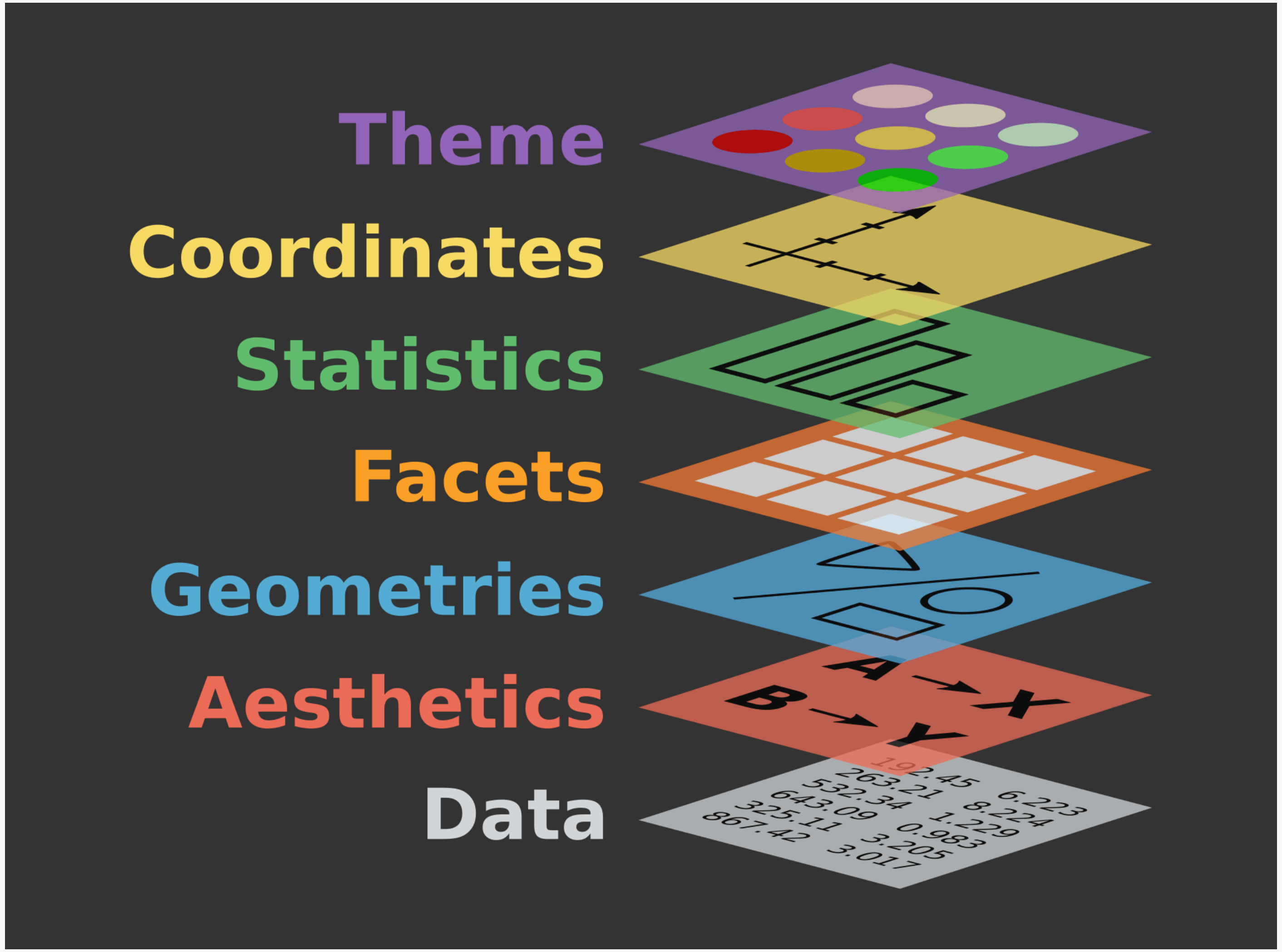
gg is for
Grammar of Graphics

How do we express visuals in words?

- Data to be visualized

How do we express visuals in words?

- Data to be visualized



How do we express visuals in words?

- Data to be visualized
- Geometric objects that appear on the plot

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

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

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

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

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

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

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

```
tidy_pop <- gather(messy_pop, 'year', 'pop', -country)
```

country	year	pop
Canada	1997	30.306
China	1997	1230.075
United States	1997	272.912
Canada	2002	31.902

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

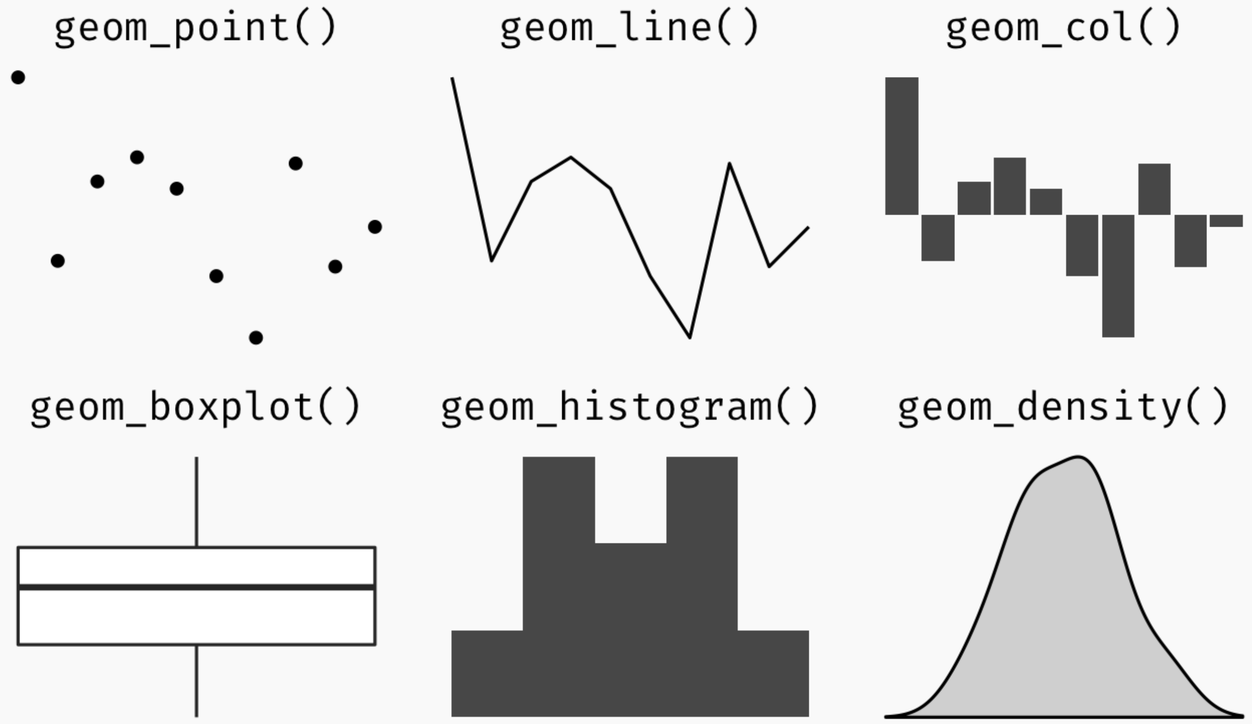
Data

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"
```

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

```
+ geom_*()
```

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

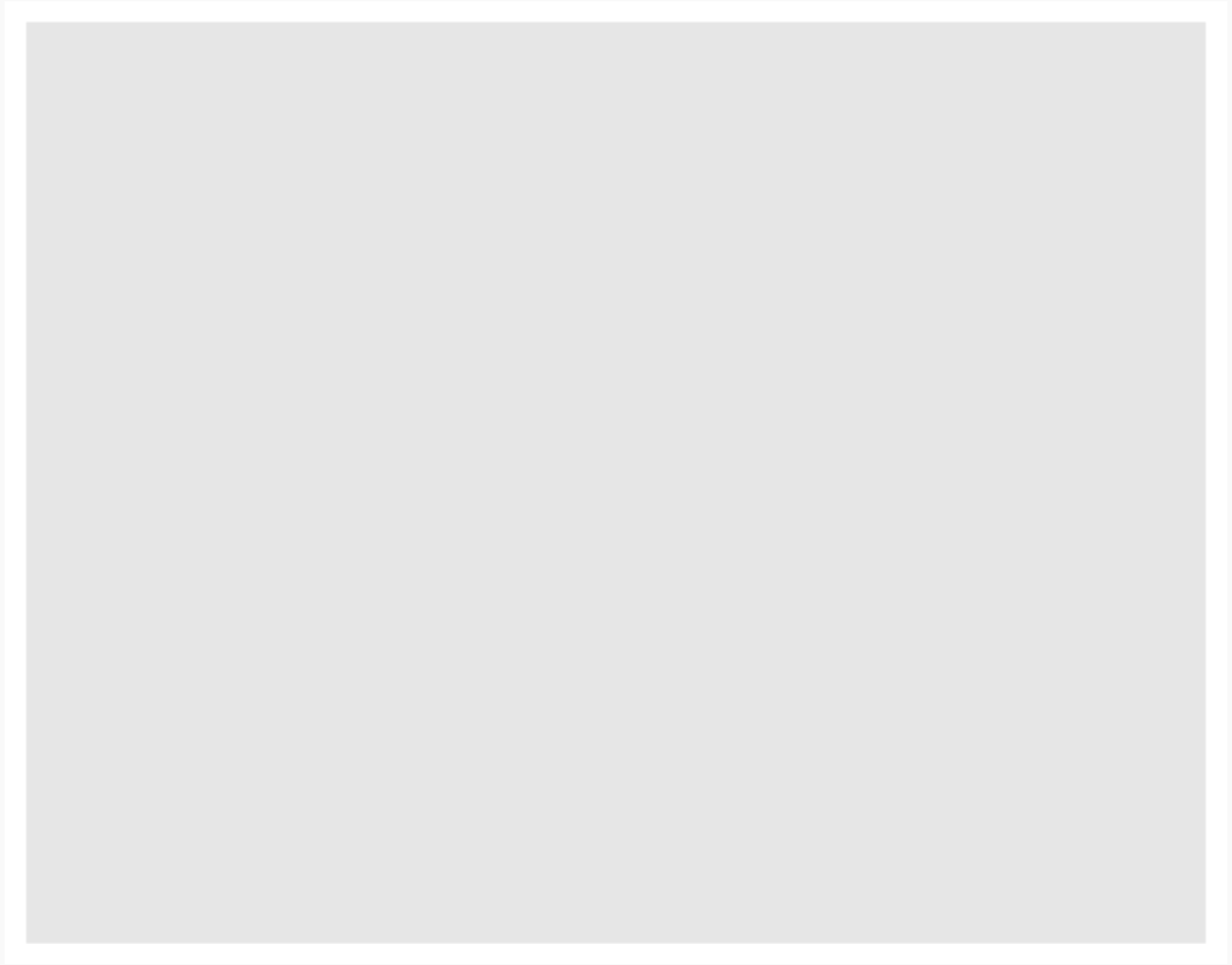
```
#  [1] "geom_abline"     "geom_area"      "geom_bar"       "geom_bin2d"
#  [5] "geom_blank"      "geom_boxplot"   "geom_col"       "geom_contour"
#  [9] "geom_count"      "geom_crossbar"  "geom_curve"     "geom_density"
# [13] "geom_density_2d" "geom_density2d" "geom_dotplot"   "geom_errorbar"
# [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

```
ggplot(df_geom) +
  aes(x, y) +
```

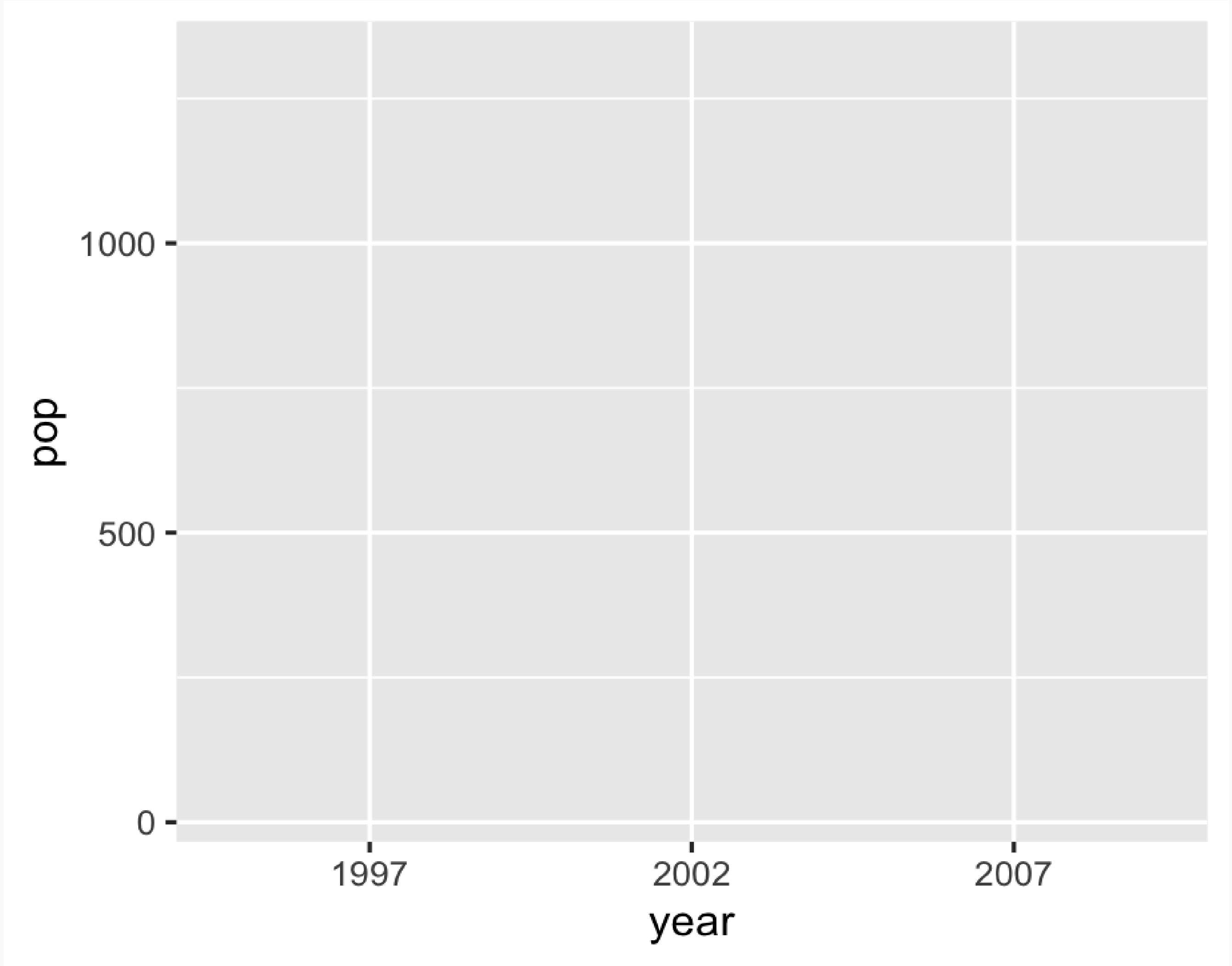
Our first plot!

```
ggplot(tidy_pop)
```



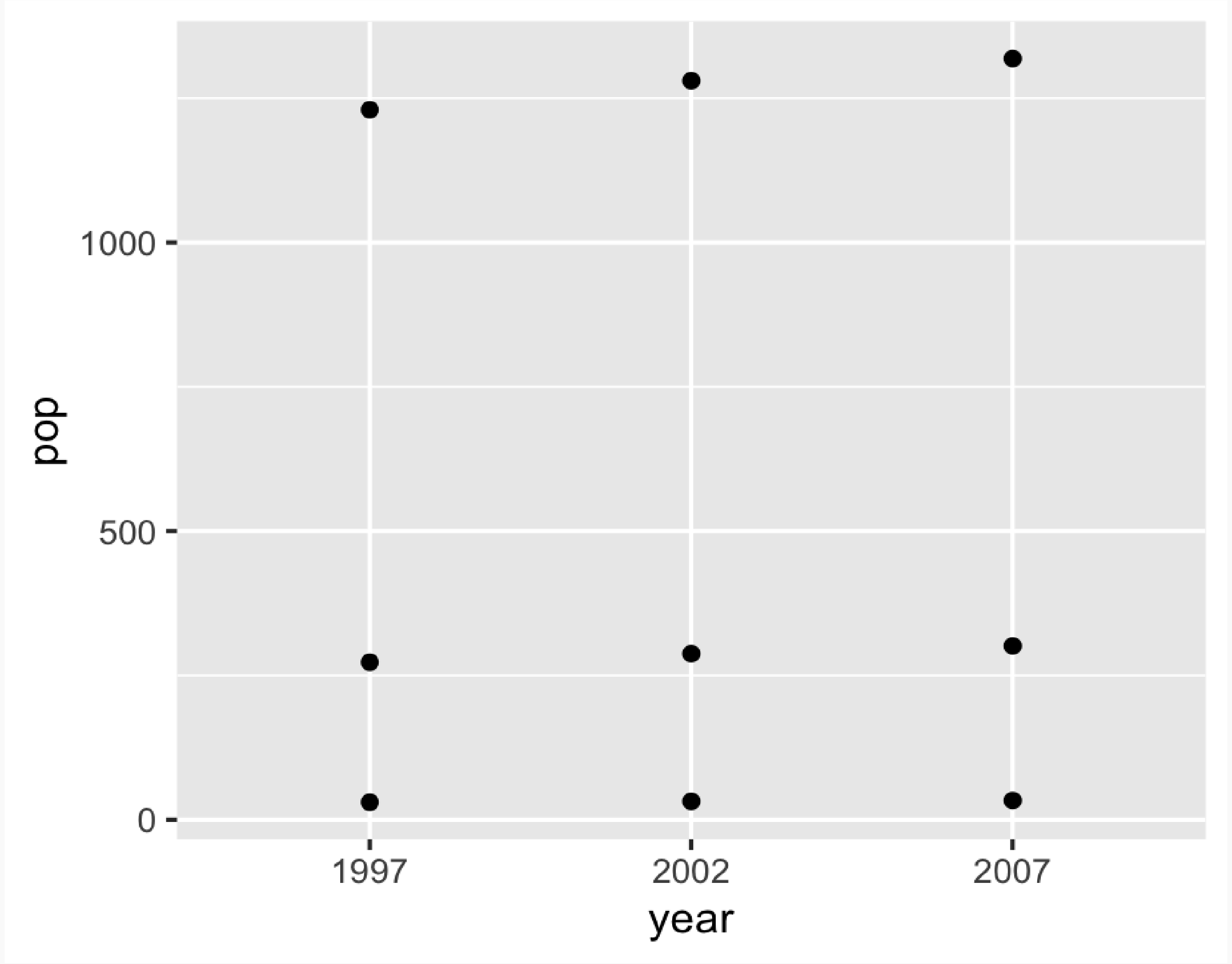
Our first plot!

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



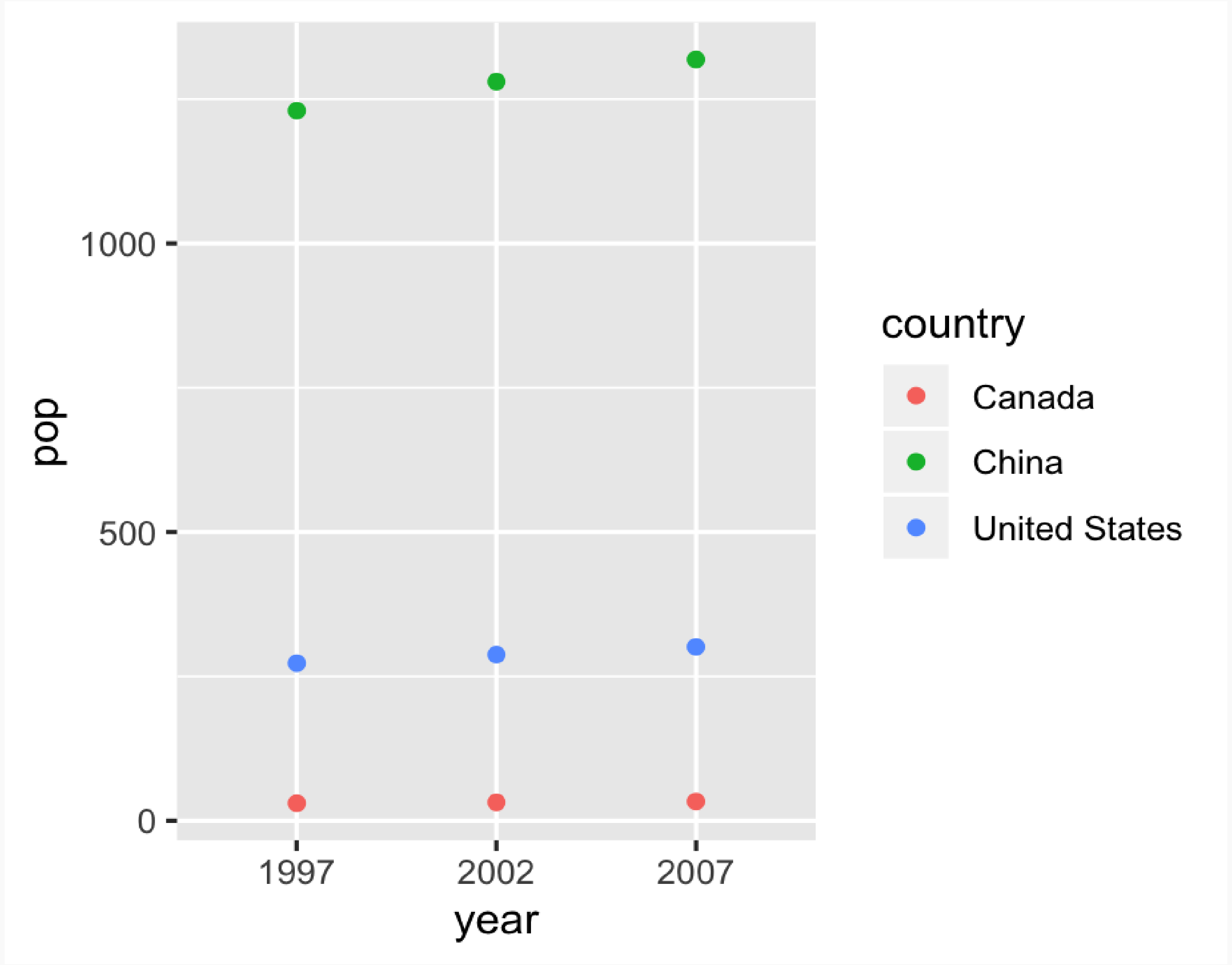
Our first plot!

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



Our first plot!

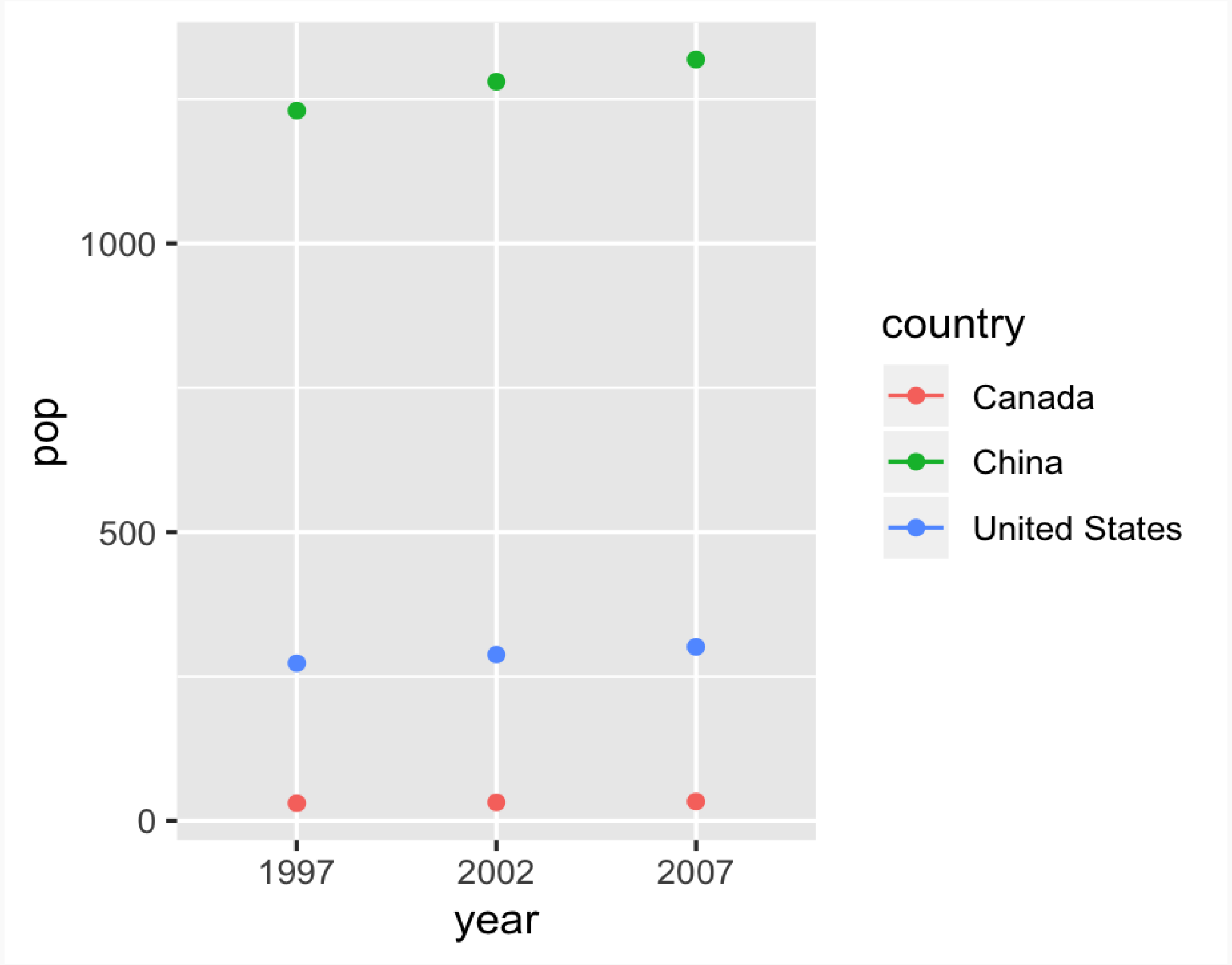
```
ggplot(tidy_pop) +
  aes(x = year,
      y = pop,
      
  geom_point()
```



Our first plot!

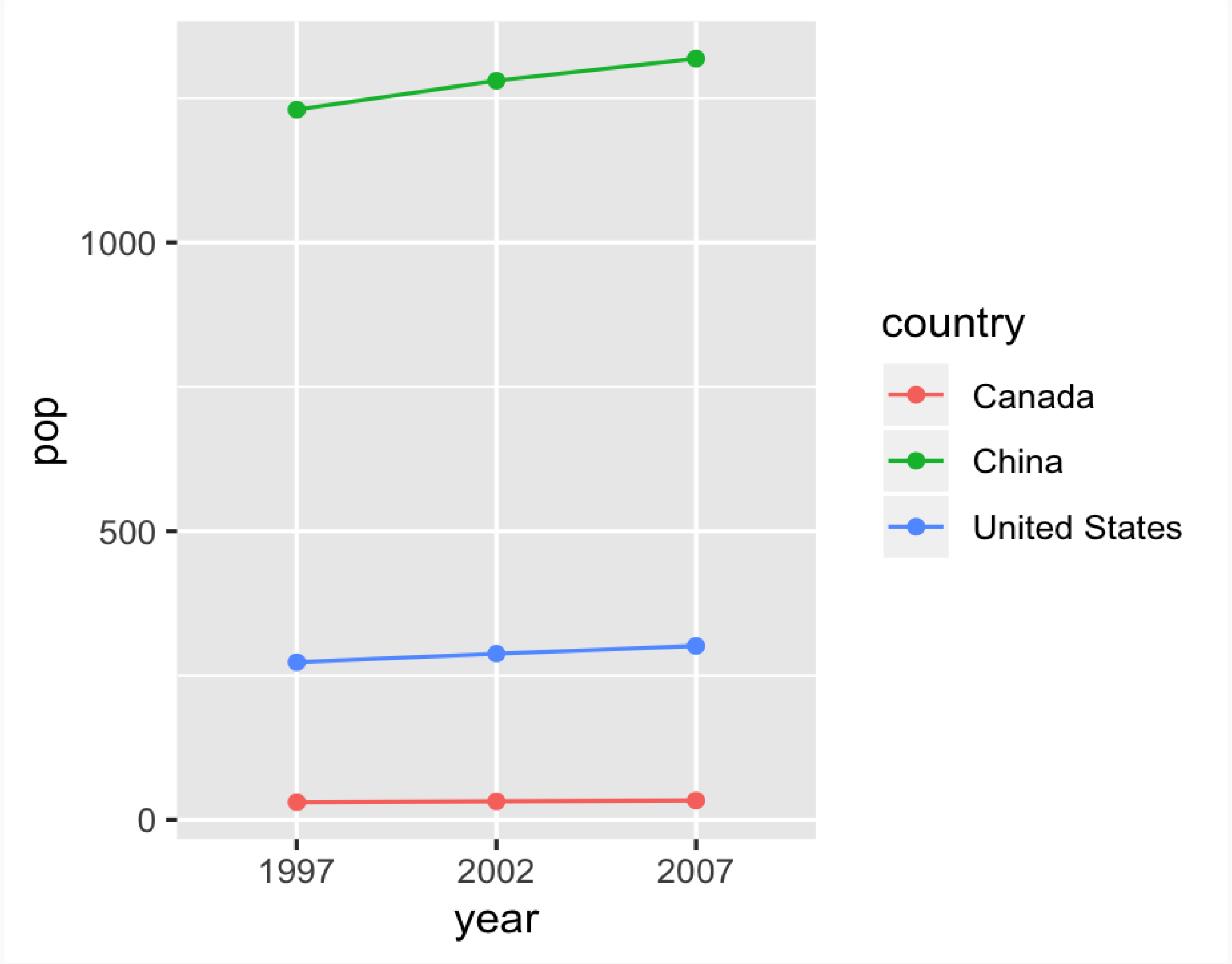
```
ggplot(tidy_pop) +
  aes(x = year,
      y = pop,
      color = country) +
  geom_point() +
  geom_line()
```

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



Our first plot!

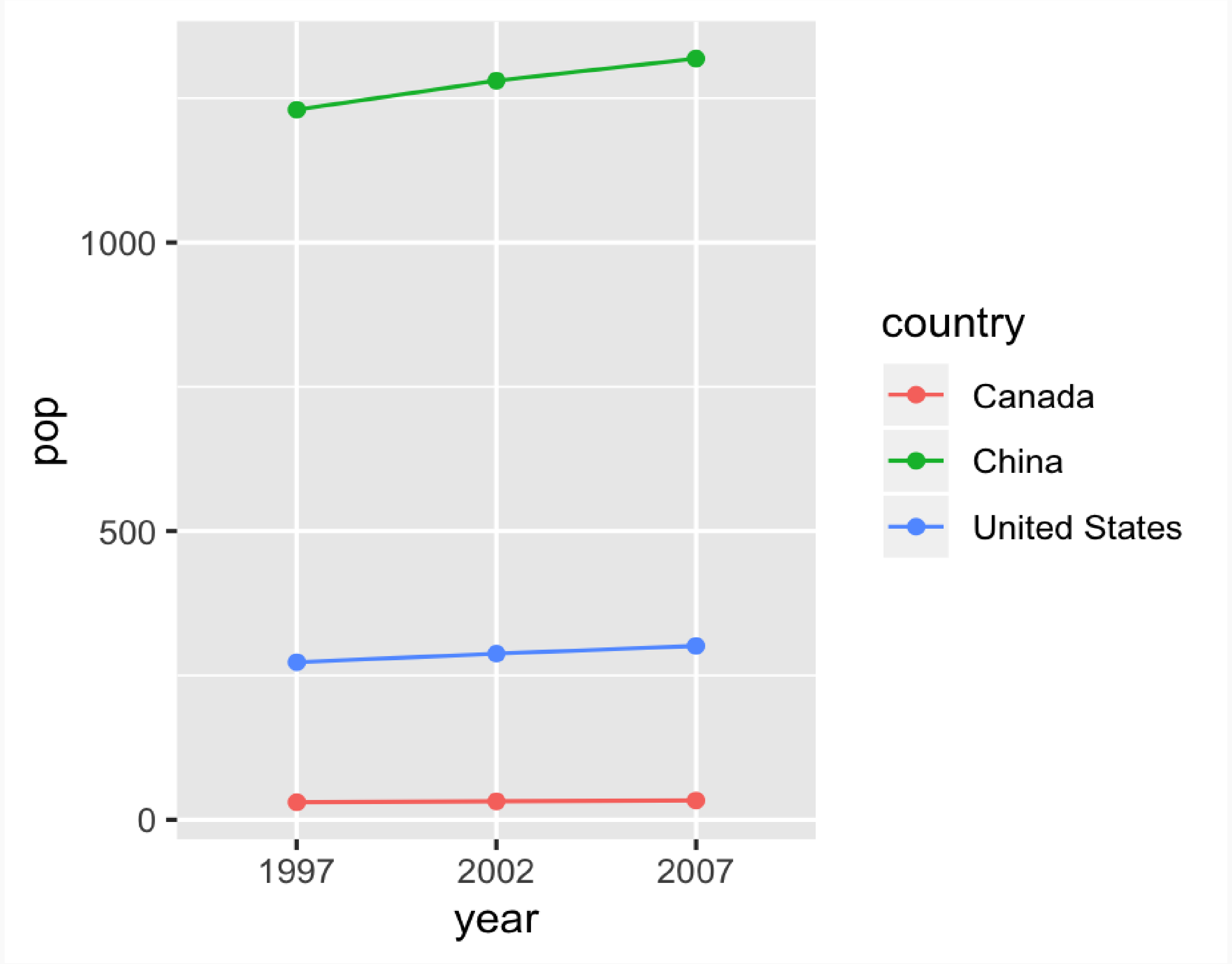
```
ggplot(tidy_pop) +
  aes(x = year,
      y = pop,
      color = country) +
  geom_point() +
  geom_line(
    aes(group = country))
```



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

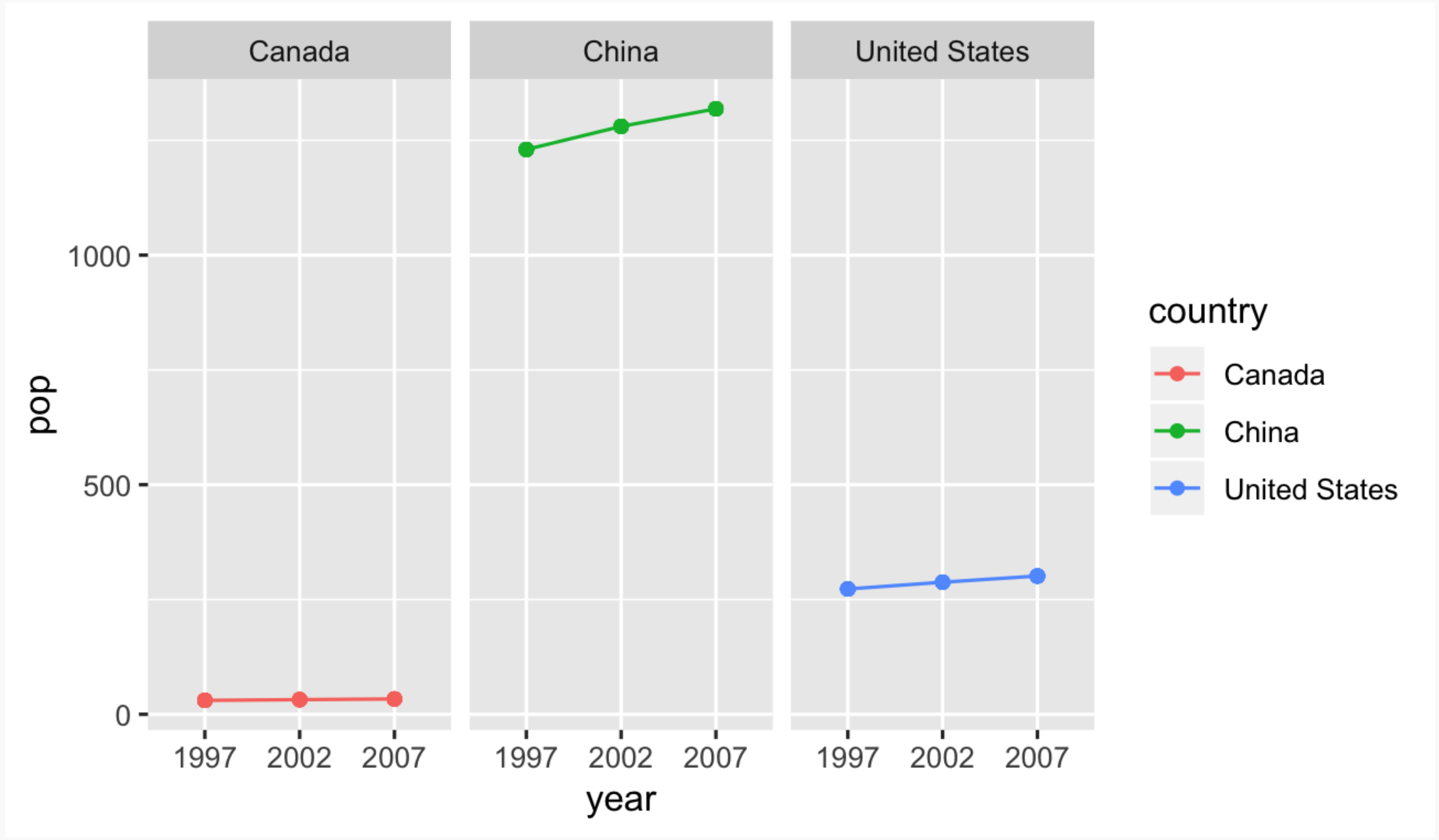
Geoms

Facet

```
+facet_wrap()

+facet_grid()
```

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



gg is for Grammar of Graphics

Data

Aesthetics

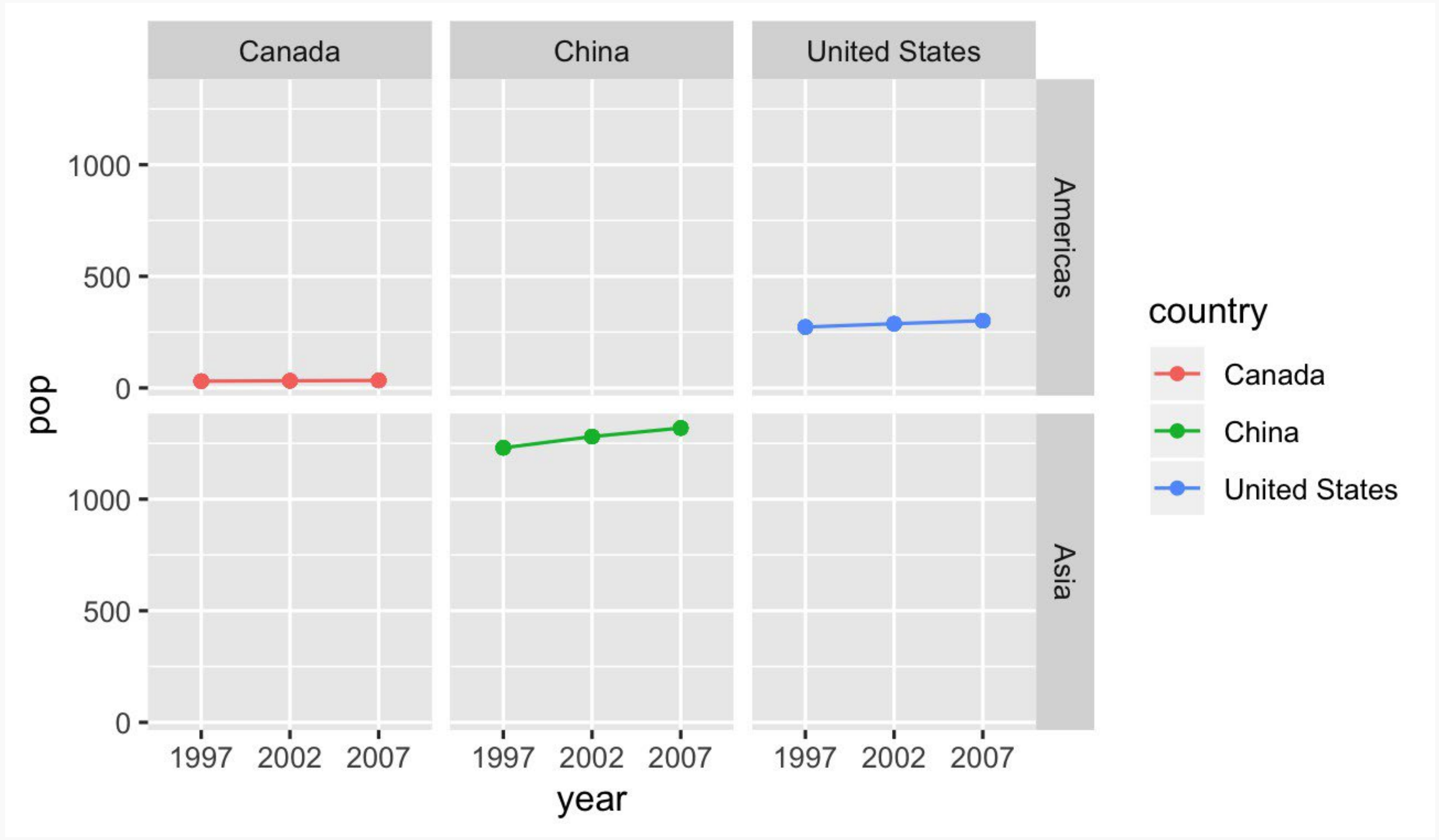
Geoms

Facet

```
+facet_wrap()

+facet_grid()
```

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



gg is for Grammar of Graphics

Data

Aesthetics

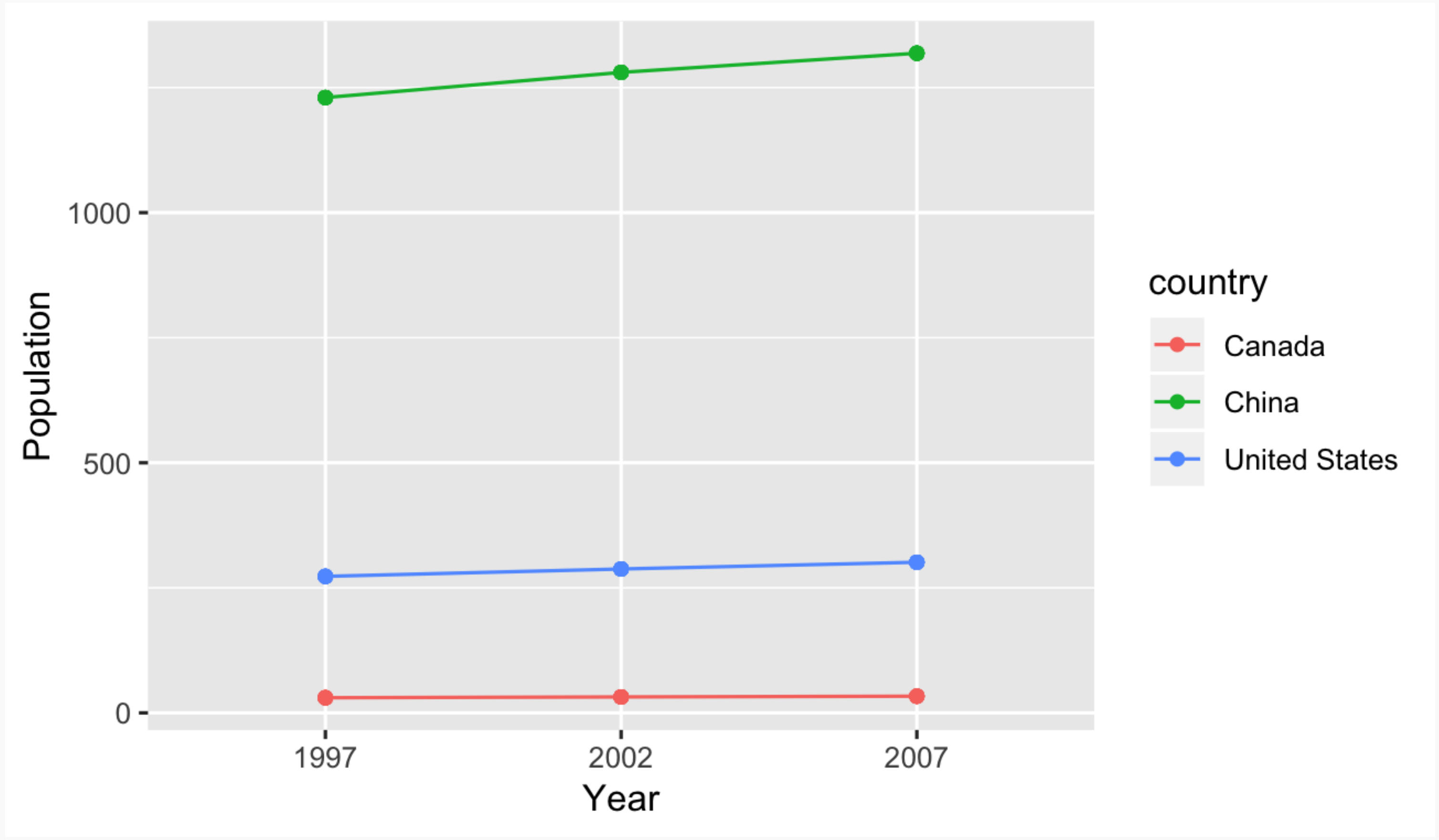
Geoms

Facet

Labels

```
+ labs()
```

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



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

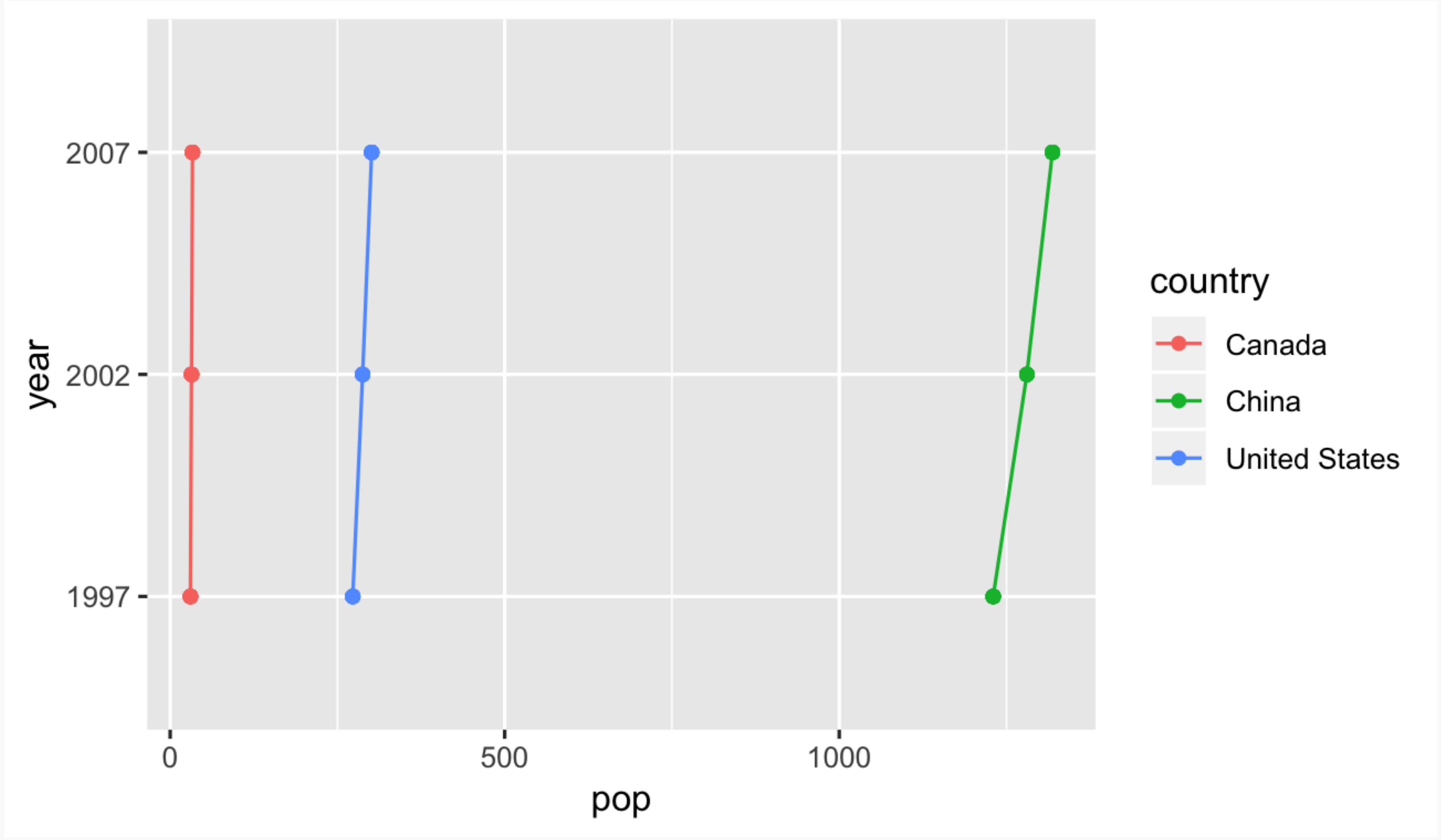
Facet

Labels

Coords

```
+ coord_*()
```

```
g + coord_flip()
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

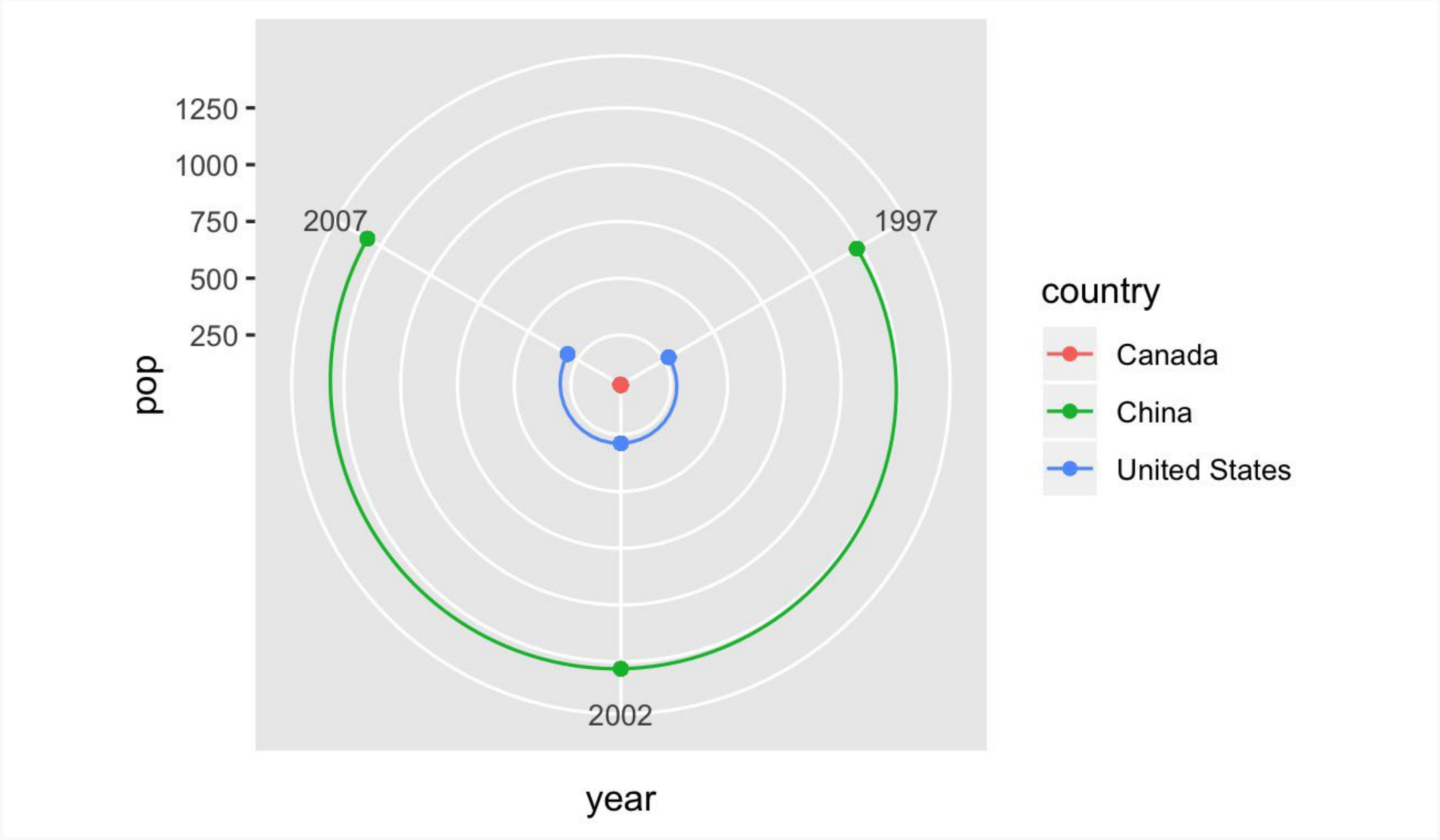
Facet

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

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

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

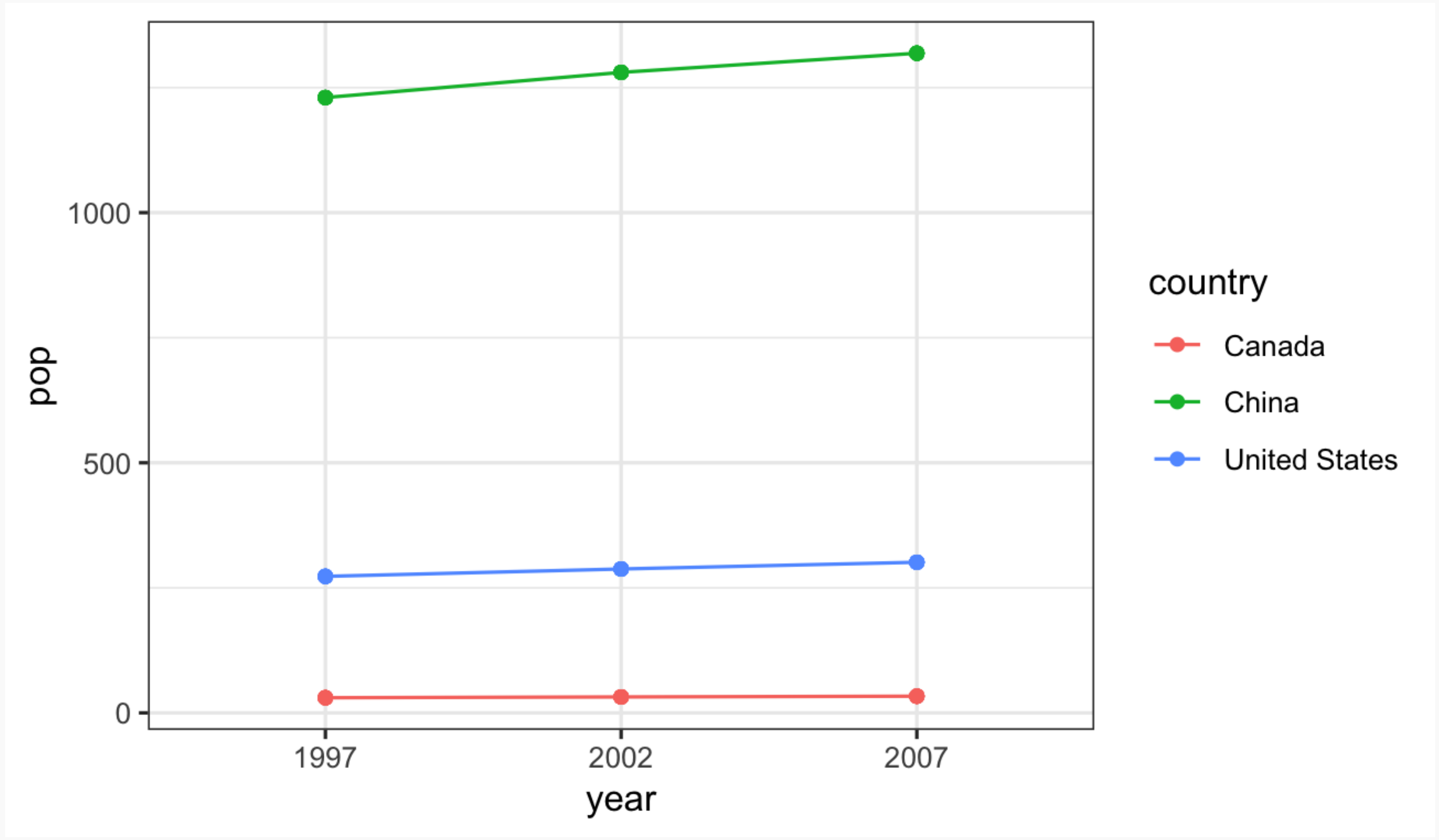
Coords

Scales

Theme

```
+ theme()
```

```
g + theme_bw()
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

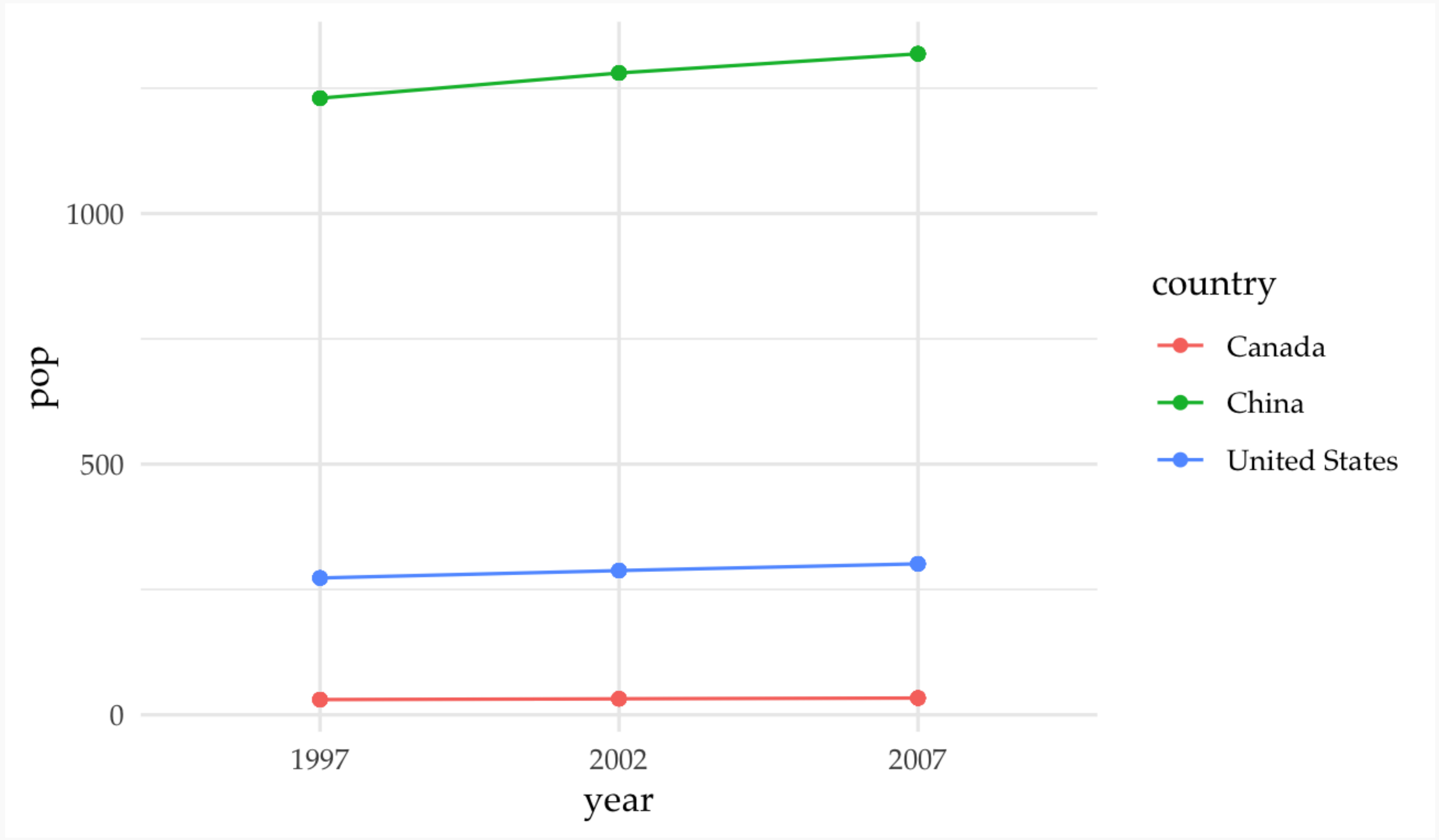
Coords

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

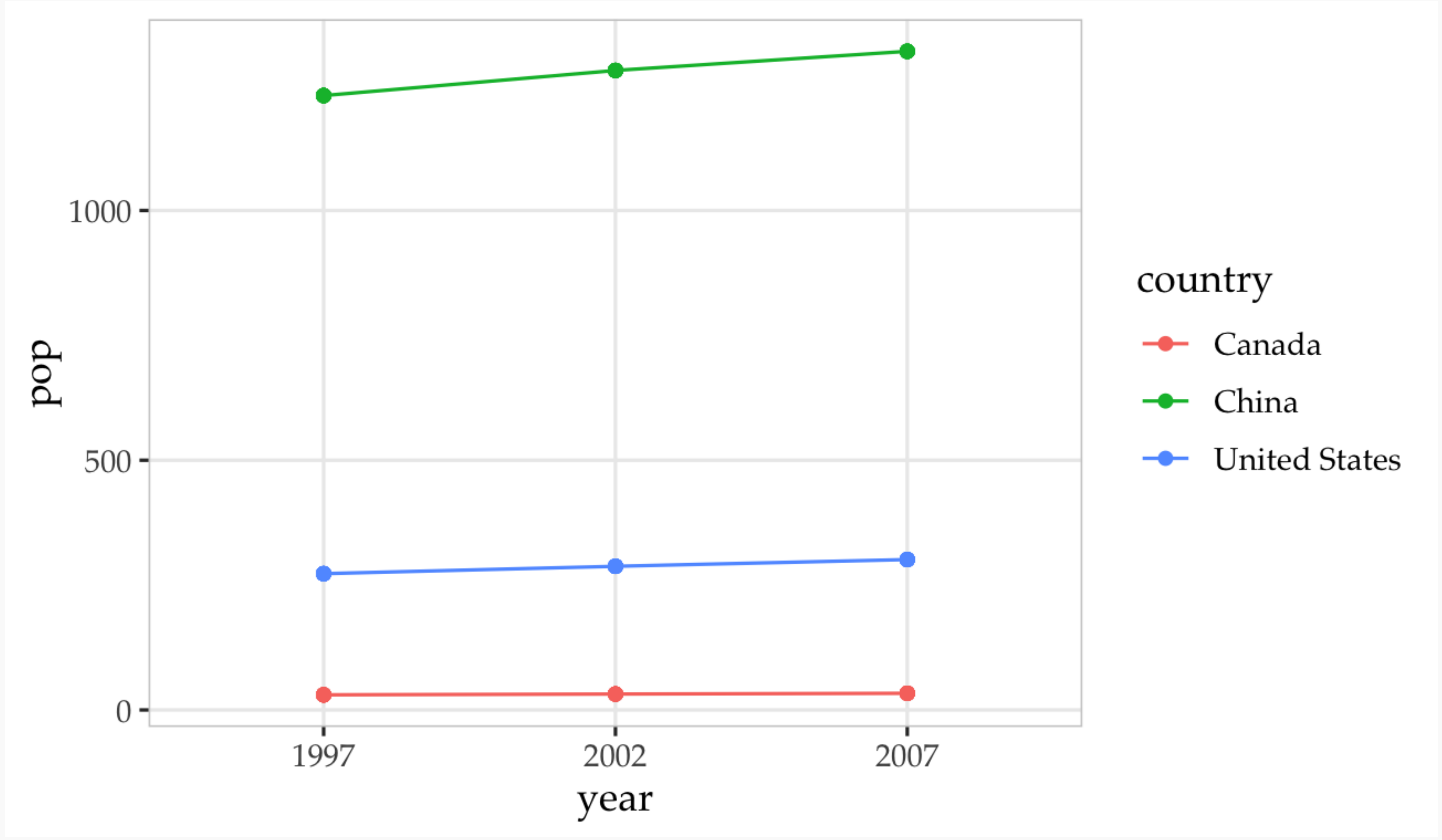
Coords

Scales

Theme

```
+ theme()
```

```
g
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

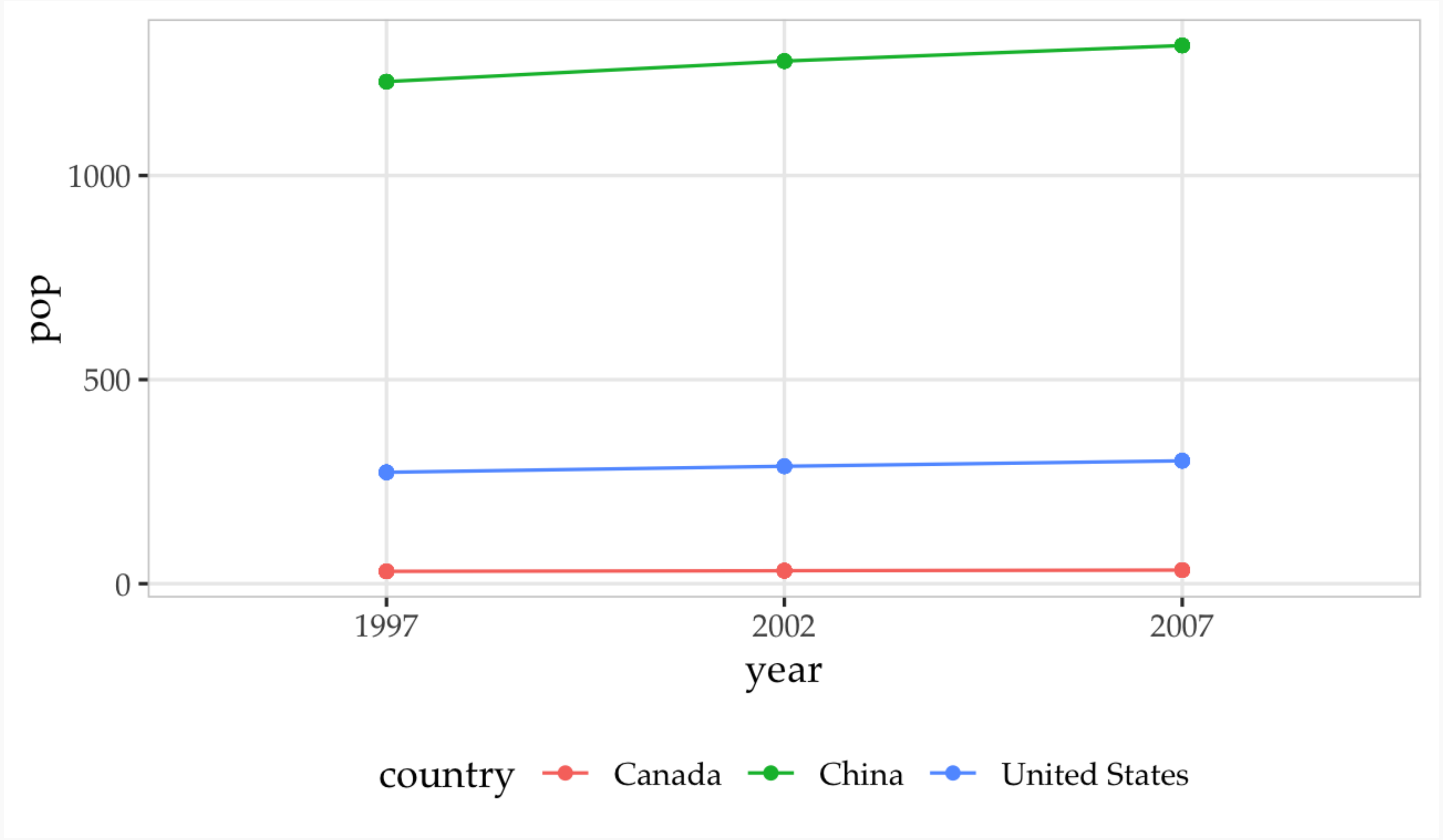
Coords

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!

"Live" Coding

```r
library(gapminder)
```