Lecture 5: Data Visualization Part 2

James Sears*
AFRE 891 SS 24
Michigan State University

*Parts of these slides are adapted from "Advanced Data Analytics" by Nick Hagerty and "Introduction to Data Science" by Rafael A. Irizarry, used under CC BY-NC-SA 4.0.

Table of Contents

Last Lecture:

This Lecture:

Prologue

Prologue

Packages we'll use for today's examples:

```
pacman::p_load(dslabs, ggrepel, tidyverse)
```

Prologue

Additional packages if you want to replicate the color schemes later on:

```
pacman::p_load(RColorBrewer, viridis, Polychrome, broman)
```

Prologue

Additional packages if you want to replicate the theme/plot extensions:

```
pacman::p_load(ggtheme, showtext, gganimate, ggExtra)
```

Common Charts

Common Charts

Now that we've seen an example of the complex plots we can create with **ggplot2**, let's take a step back and talk through the **geoms** we'll need for common charts and some useful **customization settings**.

This Lecture

- Line
- Scatterplot
- Histogram
- Ridge Plots
- Kernel Densities

Later this Semester

- Ribbons
- Dot and Whisker Plots
- Event Study Plots
- Maps

Other Common Charts

Now that we're up to speed with the syntax of **ggplot2**, let's work through some more techniques with other chart types.

To practice, let's load in a more complete version of the **gapminder data**:

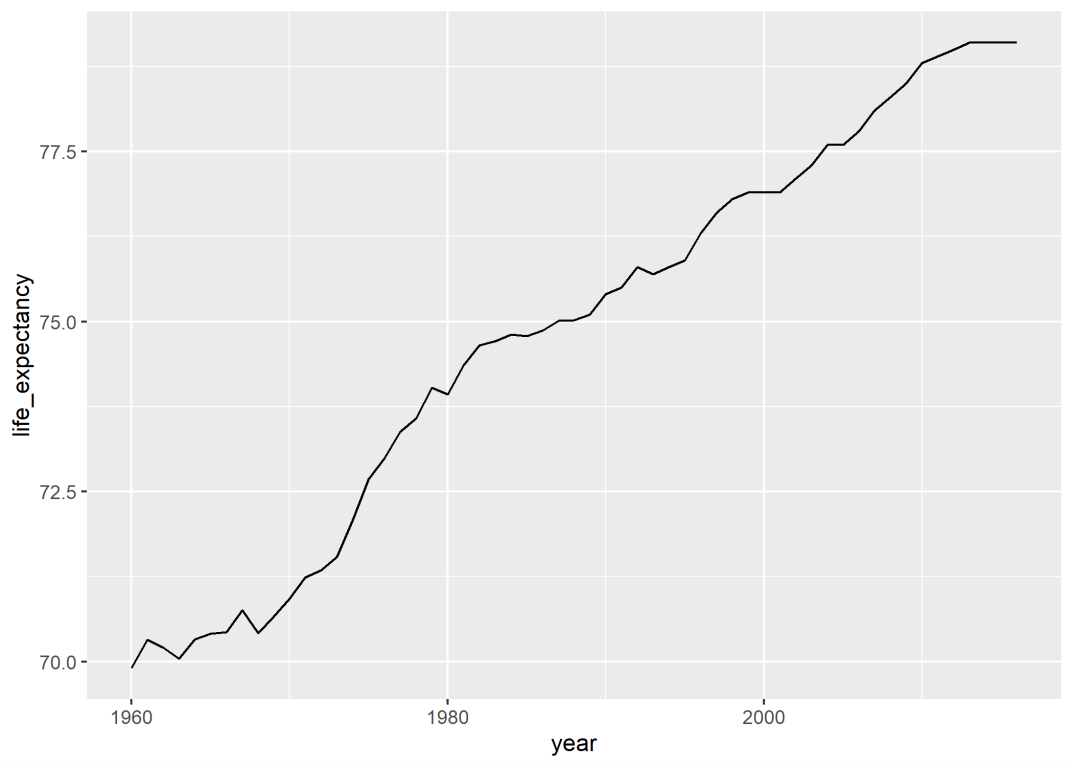
```
# run below line if you have gapminder package loaded
#try(detach("package:gapminder", unload = TRUE))

data(gapminder)
```

Line Chart (*geom_line*)

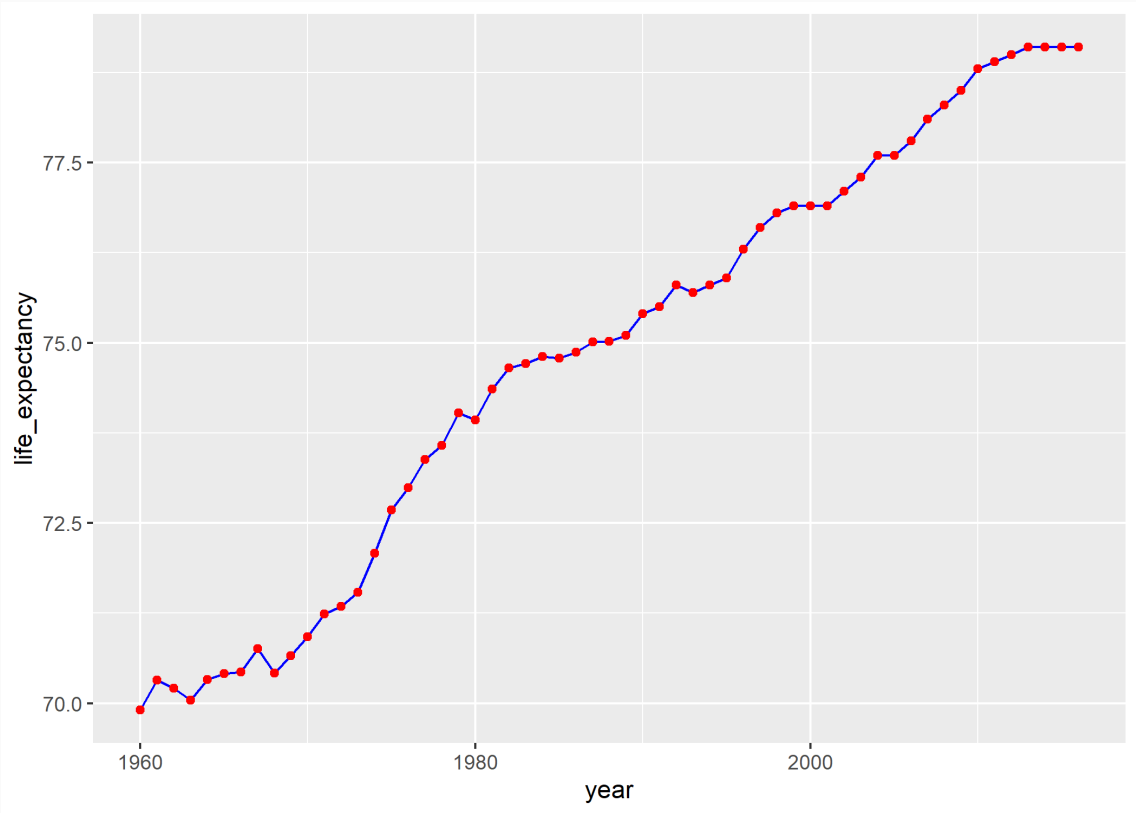
How has average life expectancy in the US evolved from 1960-2016?

```
filter(gapminder, country == "United States") %>%
  ggplot()+
  geom_line(aes(x = year, y = life_expectancy))
```



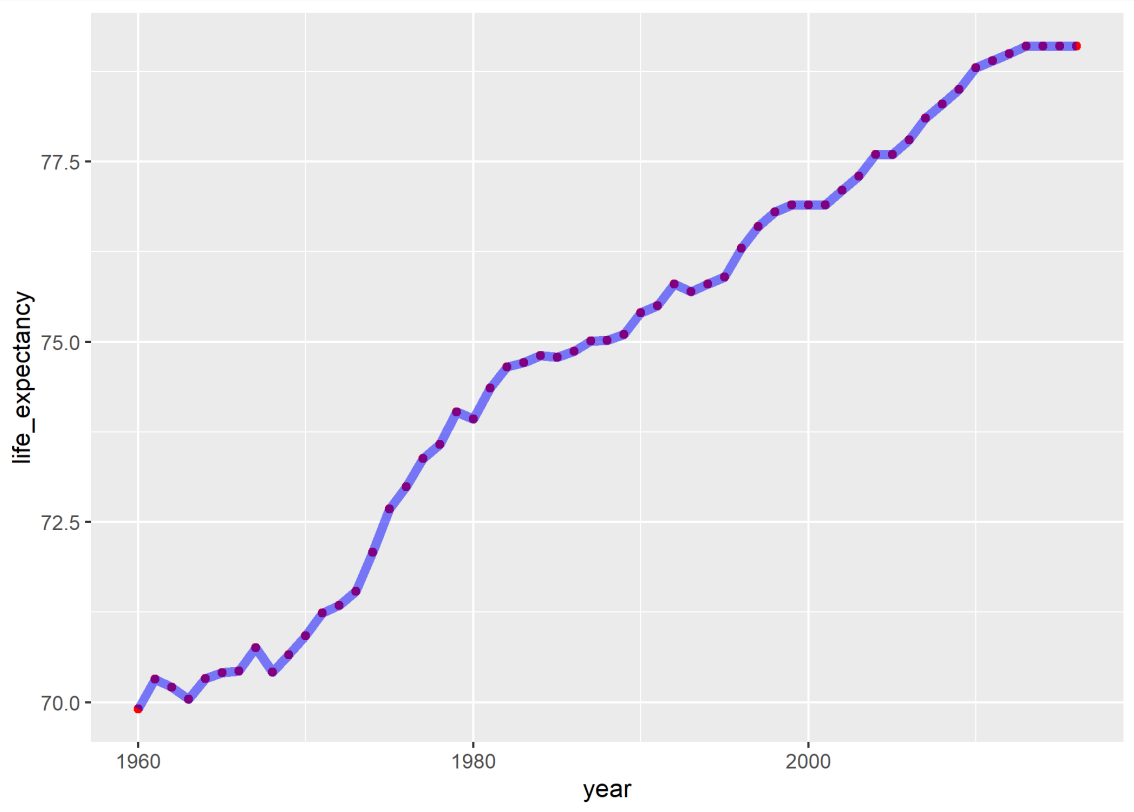
Points Connected with a Line

```
filter(gapminder, country == "United States") %>%
  ggplot(aes(x = year, y = life_expectancy)) +
  geom_line(color = "blue") +
  geom_point(color = "red")
```



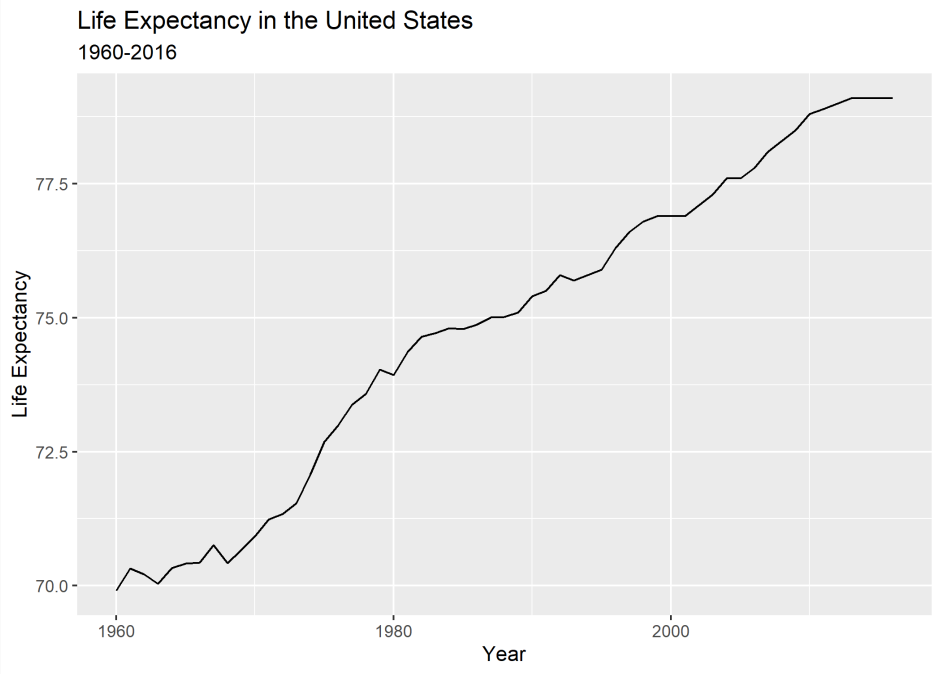
Layer Order: Lowest Layer on Top

```
filter(gapminder, country == "United States") %>%
  ggplot(aes(x = year, y = life_expectancy)) +
  geom_point(color = "red") +
  geom_line(color = "blue", linewidth = 2, alpha = 0.5)
```



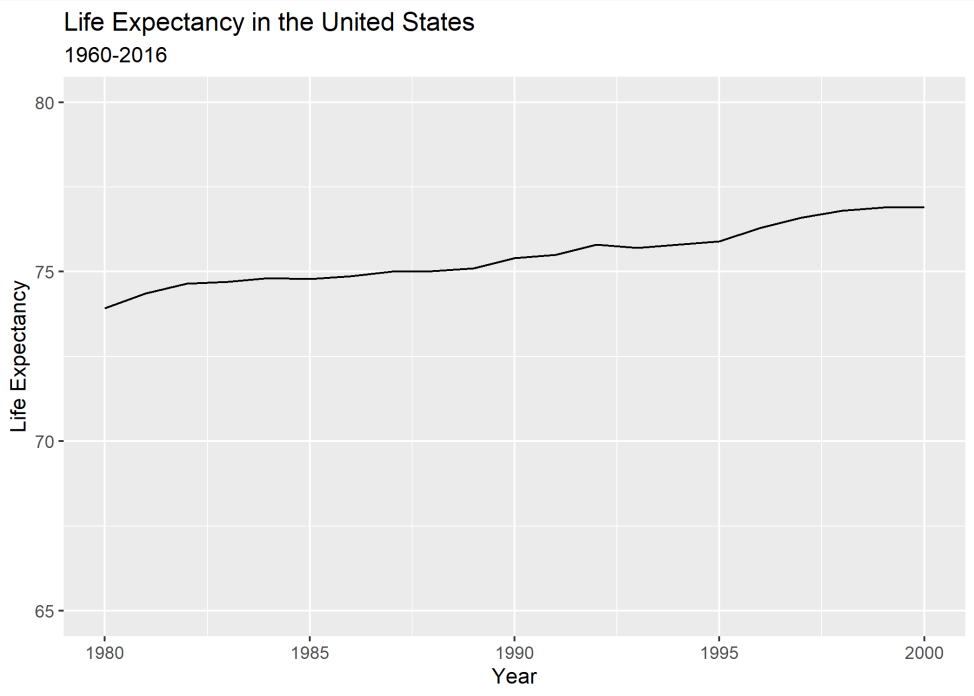
Modify Titles with *labs()*

```
filter(gapminder, country == "United States") %>%
  ggplot() +
  geom_line(aes(x = year, y = life_expectancy)) +
  labs(title = "Life Expectancy in the United States",
       subtitle = "1960-2016",
       x = "Year",
       y = "Life Expectancy")
```



Change Axis Limits with *lims()*

```
filter(gapminder, country == "United States") %>%
  ggplot() +
  geom_line(aes(x = year, y = life_expectancy)) +
  labs(title = "Life Expectancy in the United States",  subtitle = "1960-2
  lims(y = c(65, 80), x = c(1980, 2000))
```

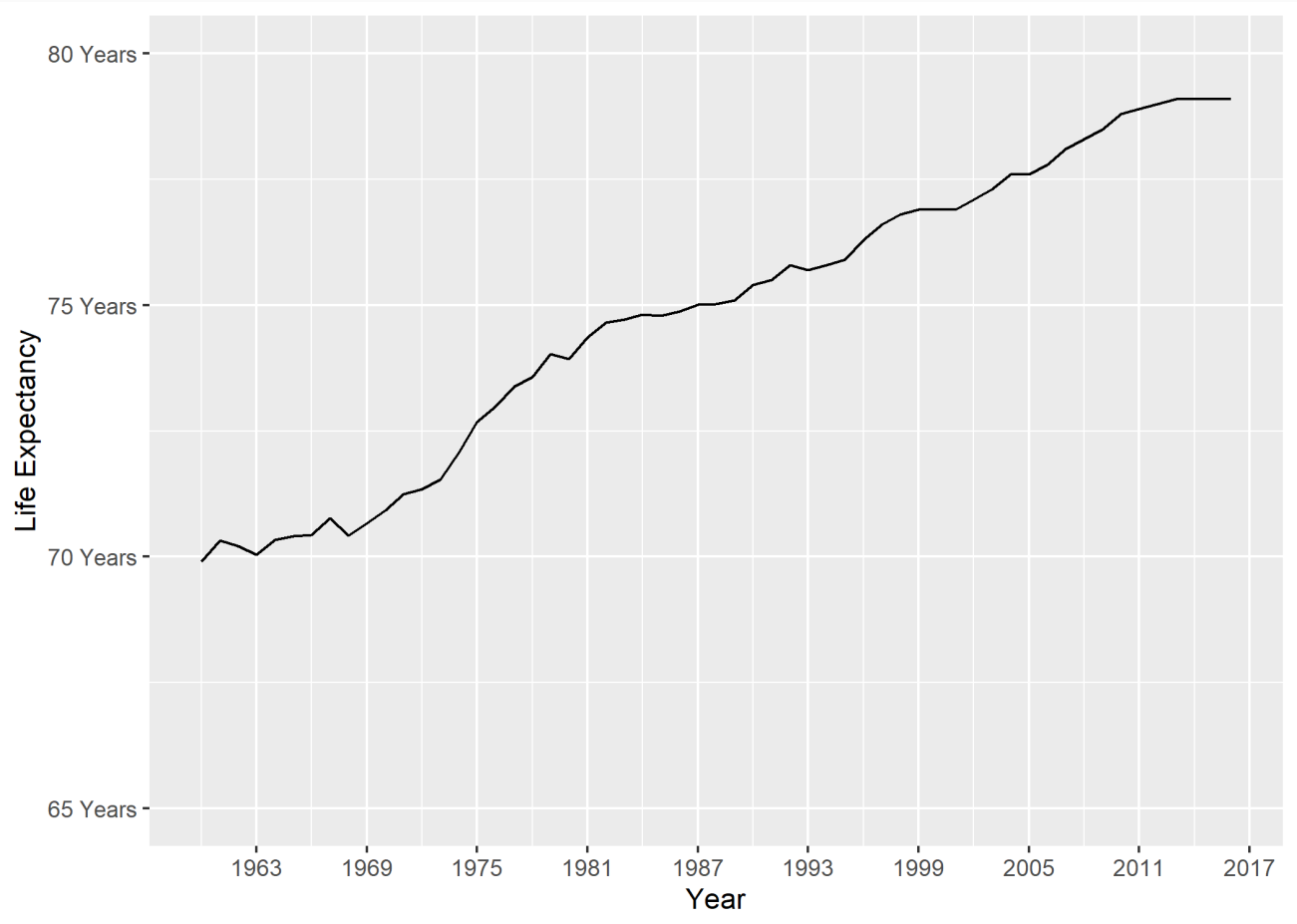


Customize Axis Scales

Change limits/breaks/labels for a **specific axis** with `scale_x/y_type()` layers (`type` one of "discrete", "continuous", "date", or "manual")

```
pacman::p_load(lubridate)
filter(gapminder, country == "United States") %>%
  # add a date-formatted version of our year variable
  mutate(year_date = paste("01/01/",year) %>% mdy())  %>%
  ggplot() +
  geom_line(aes(x = year_date, y = life_expectancy)) +
  # Format continuous y axis
  scale_y_continuous(name = "Life Expectancy",
                     breaks = seq(65, 80, 5),
                     labels = paste0( seq(65, 80, 5), " Years"),
                     limits = c(65, 80)) +
  # format date x axis
  scale_x_date(name = "Year",
               date_breaks = "6 years",
               date_labels = "%Y")
```

Customize Axis Scales: The Chart



Mappings: Multiple Series

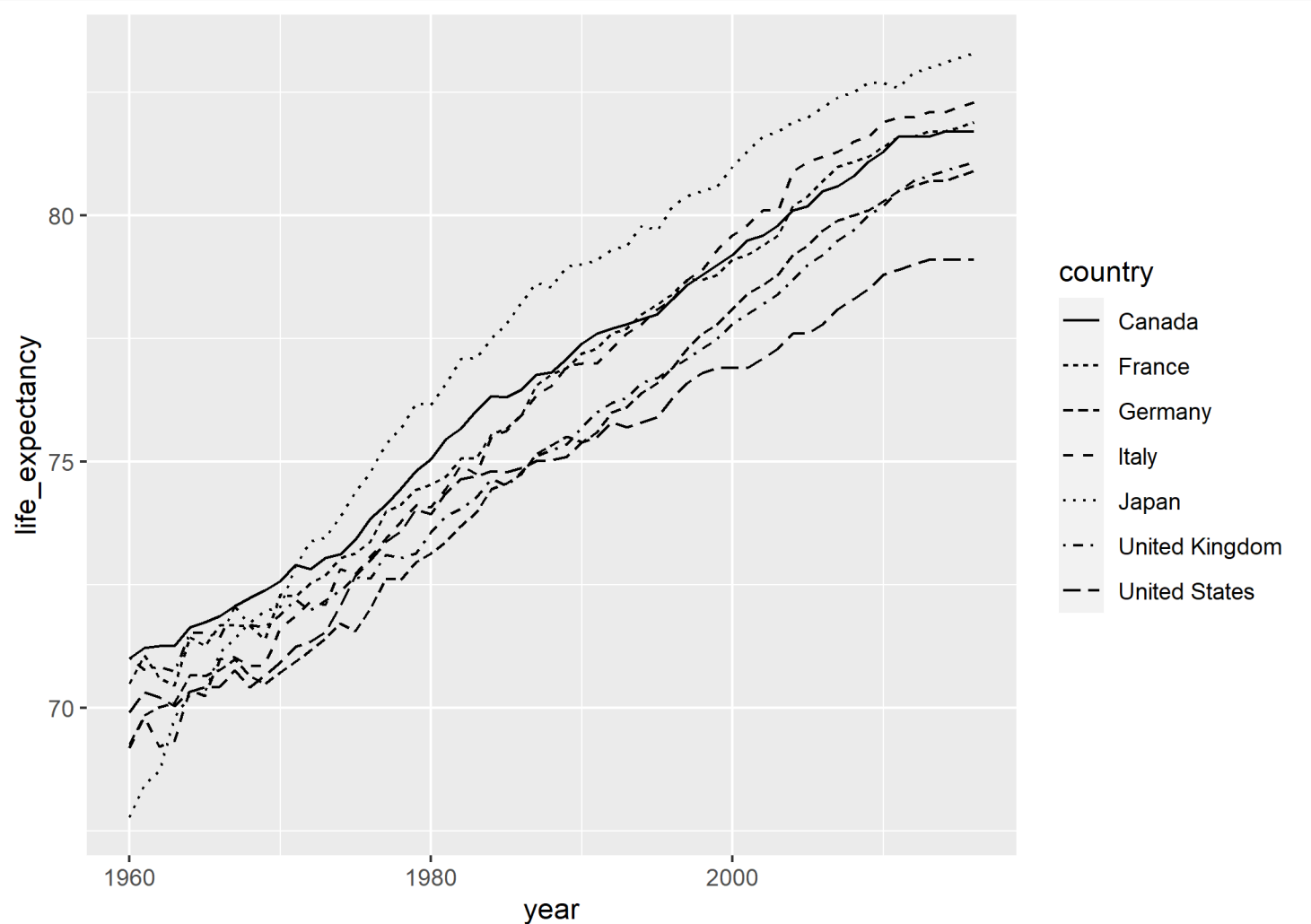
Add a `linetype` or `color` aesthetic to add a separate line for each value of a categorical variable].

What are trends in life expectancy for the G-7 countries?

```
g7 ← c("United States", "United Kingdom", "Germany", "Italy", "France",
#filter to G-7 countries
filter(gapminder, country %in% g7) %>%
  ggplot() +
  geom_line(aes(x = year, y = life_expectancy,
                linetype = country
  ))
```

Mappings: Multiple Series (*linetype*)

What are trends in life expectancy for the G-7 countries?



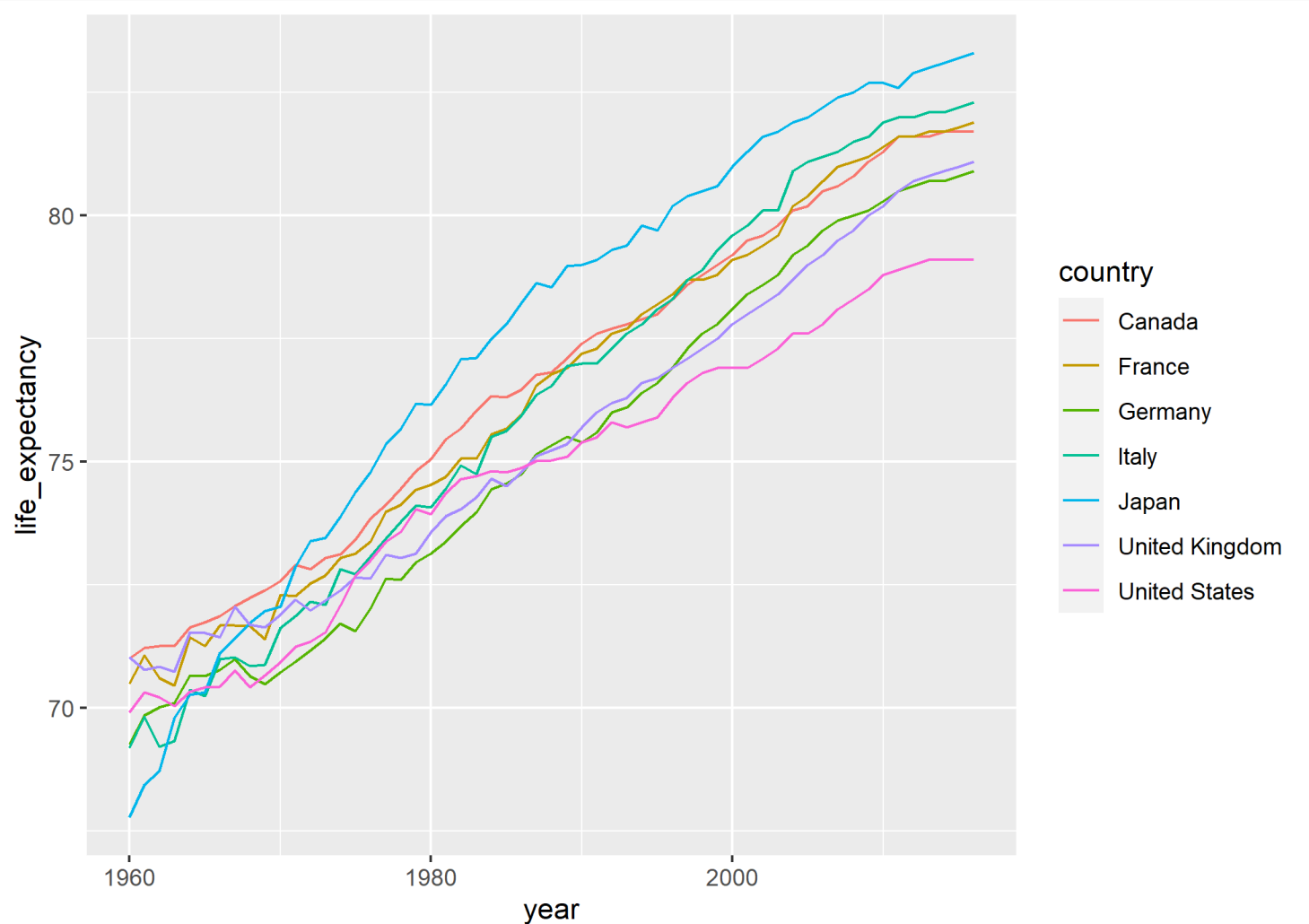
Mappings: Multiple Series (*color*)

What are trends in life expectancy for the G-7 countries?

```
# filter to G-7 countries
filter(gapminder, country %in% g7) %>%
  ggplot() +
  geom_line(aes(x = year, y = life_expectancy,
                color = country
  ))
```

Mappings: Multiple Series (*color*)

What are trends in life expectancy for the G-7 countries?



Adding Text Labels

Text labels are generally more effective than legends

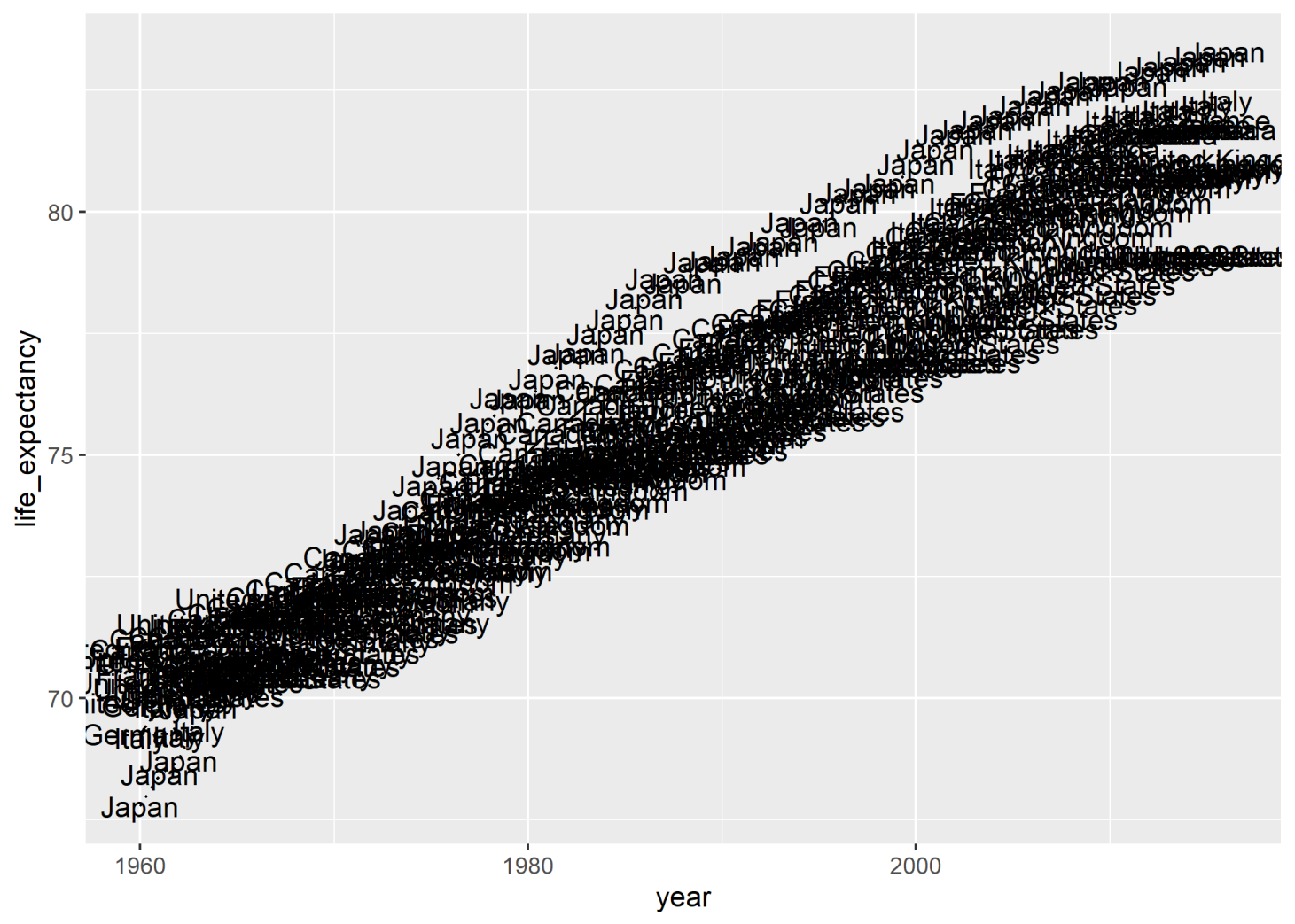
- `geom_text` from **ggplot** or `geom_text_repel` from **ggrepel** for text
- `geom_label_repel` from **ggrepel** for text + boxes

Try adding text labels to our chart:

```
# filter to G-7 countries
filter(gapminder, country %in% g7) %>%
  ggplot(aes(x = year, y = life_expectancy)) +
  geom_line(aes(linetype = country)) +
  geom_text(aes(label = country)) +
  theme(legend.position = "none")
```

Adding Text Labels

Try adding text labels to our chart:



Adding Text Labels (At Specific Points)

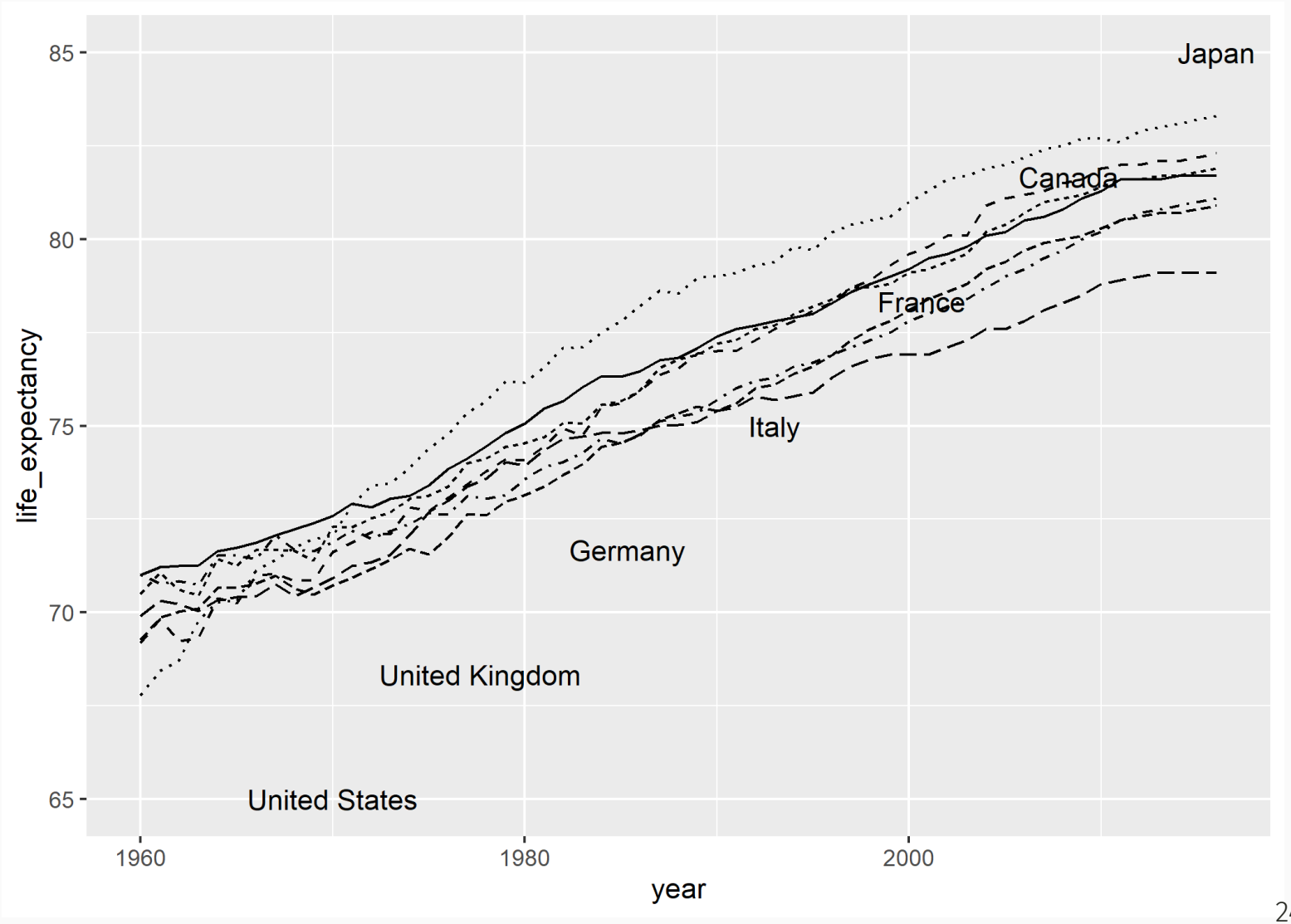
Whoops! By default `geom_text` adds text labels **at every data point.**

To add labels at **specific points**, we can **manually define the x/y locations** for `geom_text`

```
g7_labs ← data.frame(country = g7,
                     x = seq(1970, 2016, length.out = length(g7)),
                     y = seq(65, 85, length.out = length(g7)))

# filter to G-7 countries
filter(gapminder, country %in% g7) %>%
  ggplot(aes(x = year, y = life_expectancy)) +
  geom_line(aes(linetype = country)) +
  geom_text(data = g7_labs, # switch data object
            aes(x=x, y=y, label = country)) +
  theme(legend.position = "none")
```

Adding Text Labels (At Specific Points)



Adding Text Labels (At Specific Points)

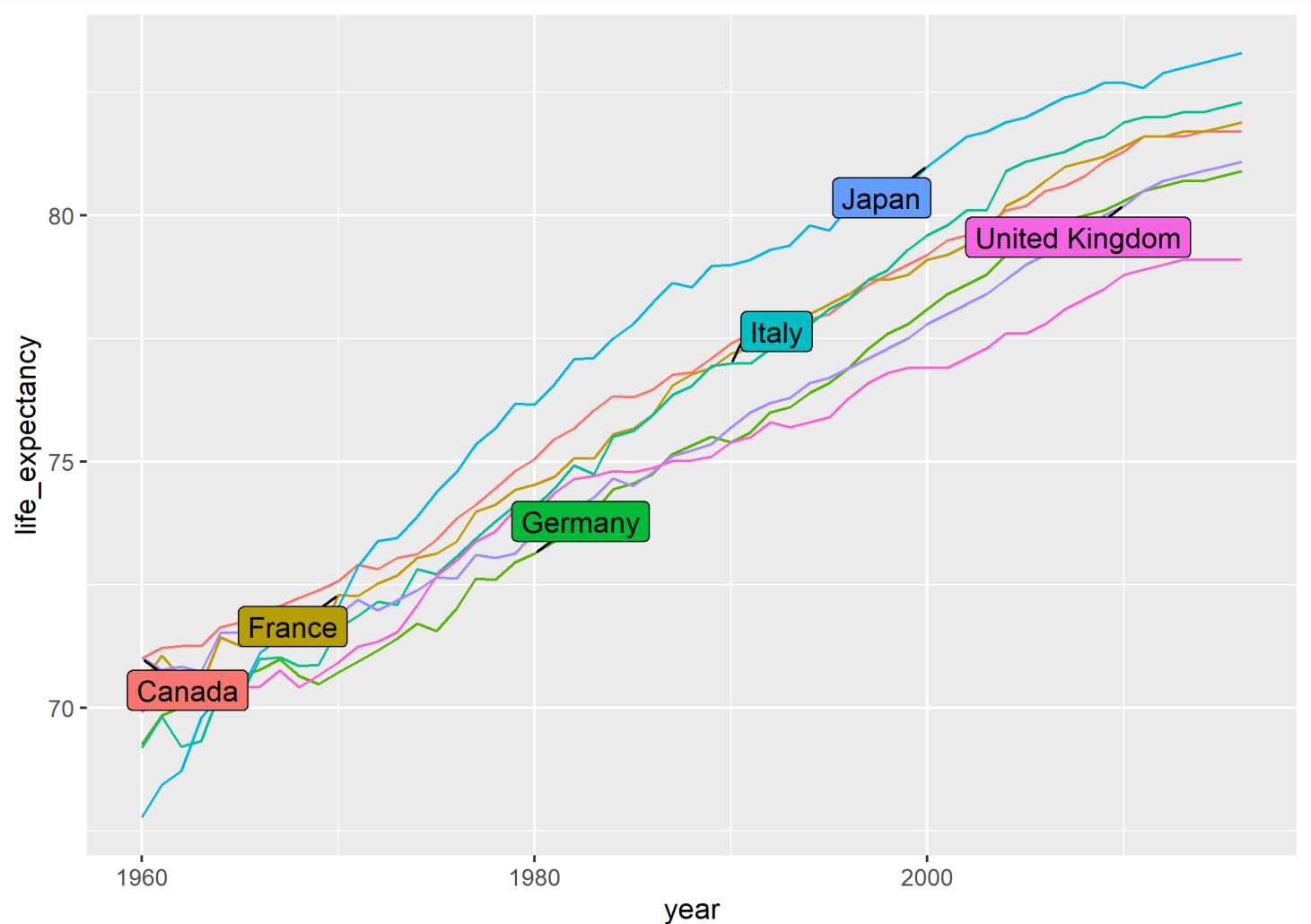
Add text + boxes with `geom_label_repel`

```
# get a label x/y point for each country
g7_labs ← filter(gapminder, country %in% g7) %>%
  mutate(order = rep(seq(1:7), 57)) %>%
  filter(year == 1960+10*(order-1))

filter(gapminder, country %in% g7) %>%
  ggplot(aes(x = year, y = life_expectancy)) +
  geom_line(aes(color = country)) +
  geom_label_repel(data = g7_labs, # switch data object
                   aes(label = country, fill = country),
                   min.segment.length = unit(0, "lines")) +
  theme(legend.position = "none")
```

Adding Text Labels (At Specific Points)

Add text + boxes with `geom_label_repel`

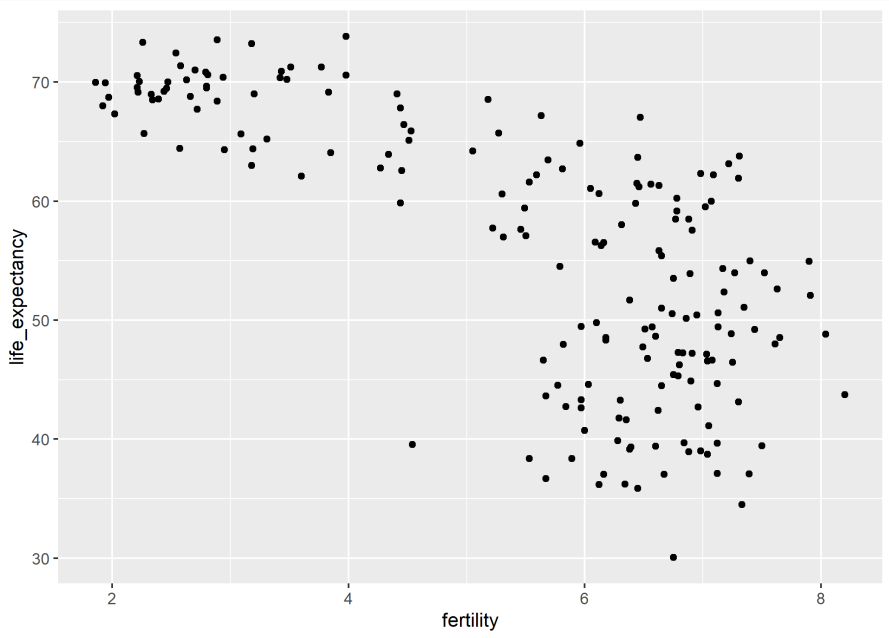


Scatterplot (*geom_point*)

How have fertility rates and life expectancies co-evolved over time?

First, plotting the data from 1962:

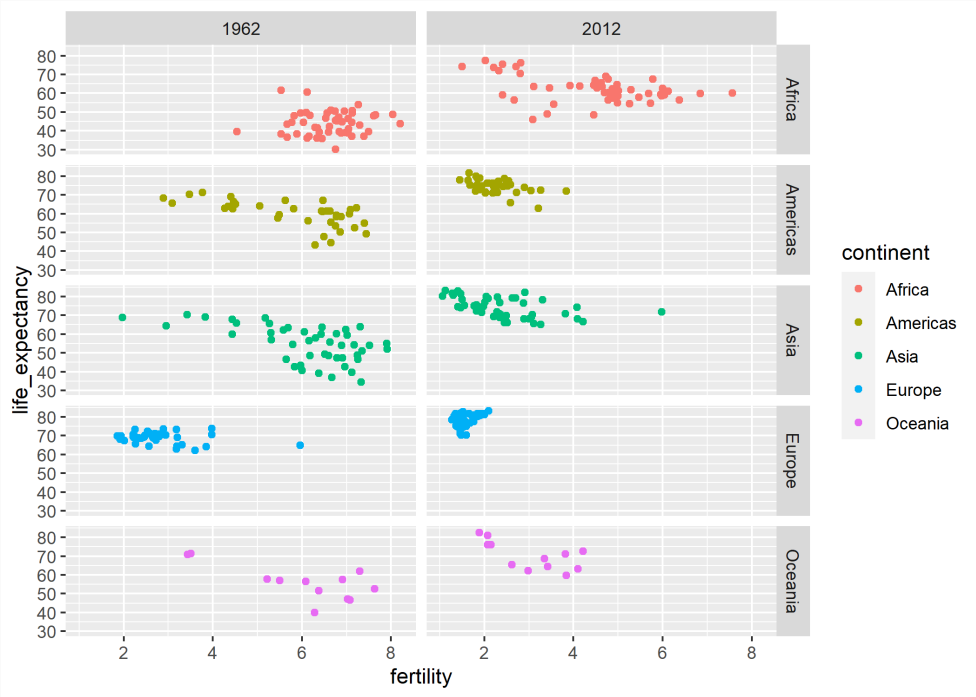
```
filter(gapminder, year == 1962) %>%
  ggplot(aes(fertility, life_expectancy)) + # omitting x/y = since in exp
  geom_point()
```



Faceting

Stratify (by continent, and compare 1962 to 2012) with `facet_grid`:

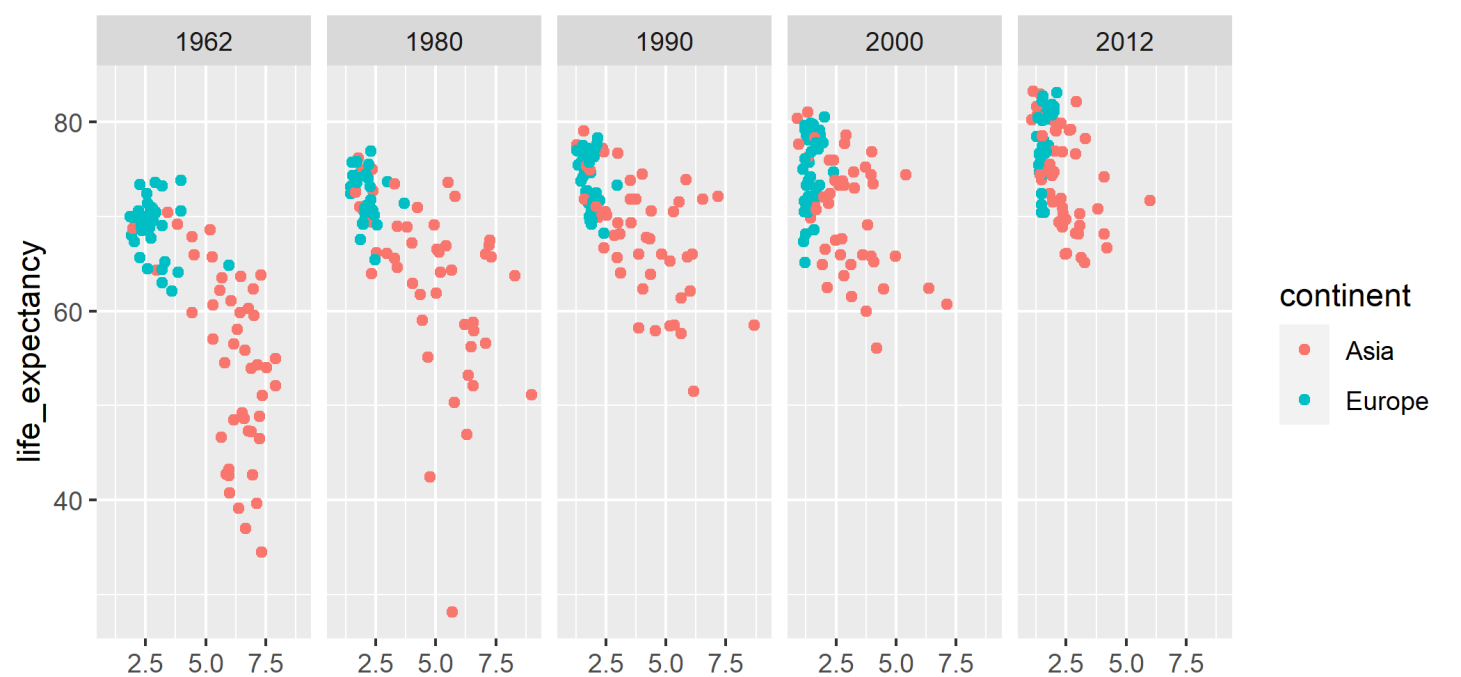
```
filter(gapminder, year %in% c(1962, 2012)) %>%
  ggplot(aes(fertility, life_expectancy, col = continent)) +
  geom_point() +
  facet_grid(continent ~ year)  # "row ~ column"
```



Faceting

Show Europe vs. Asia for 5 different years:

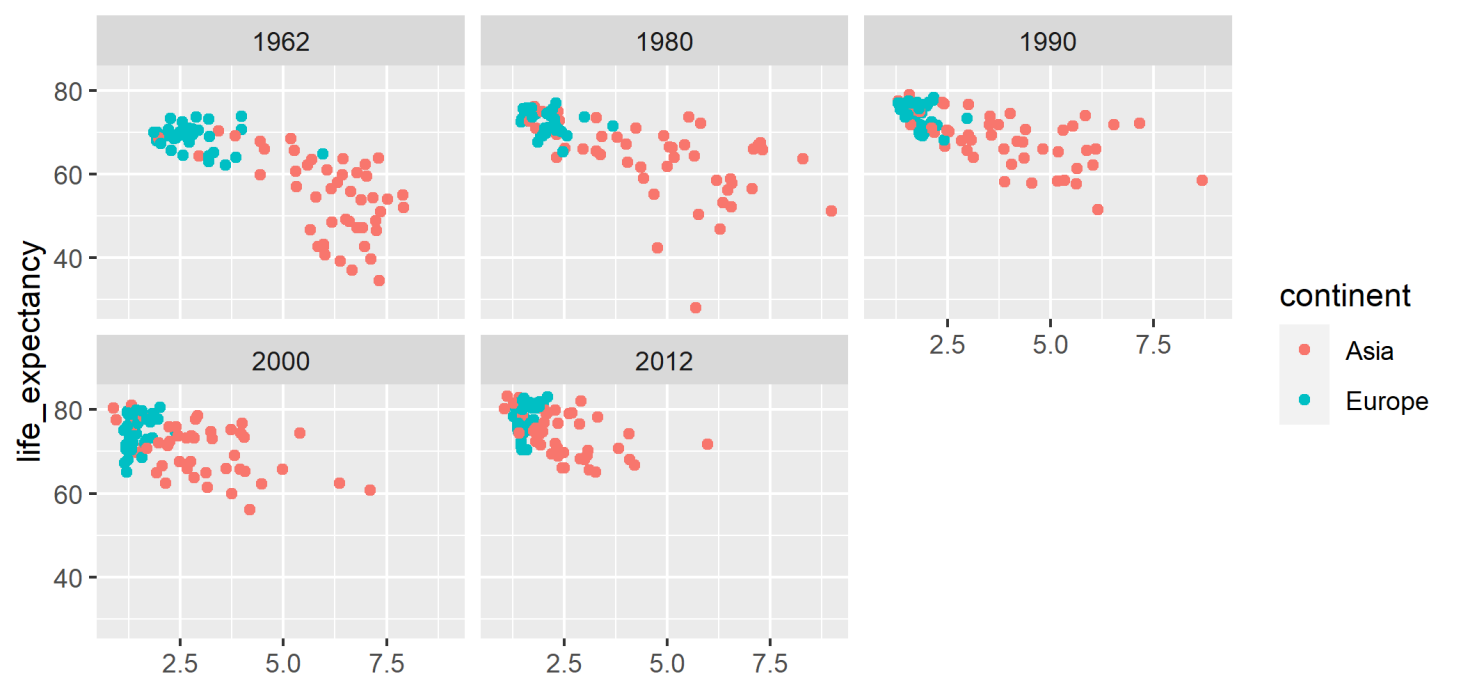
```
years = c(1962, 1980, 1990, 2000, 2012)
continents = c("Europe", "Asia")
gapminder %>%
  filter(year %in% years & continent %in% continents) %>%
  ggplot( aes(fertility, life_expectancy, col = continent)) +
    geom_point() +
    facet_grid(. ~ year)
```



Faceting

Too narrow? Wrap rows with `facet_wrap`:

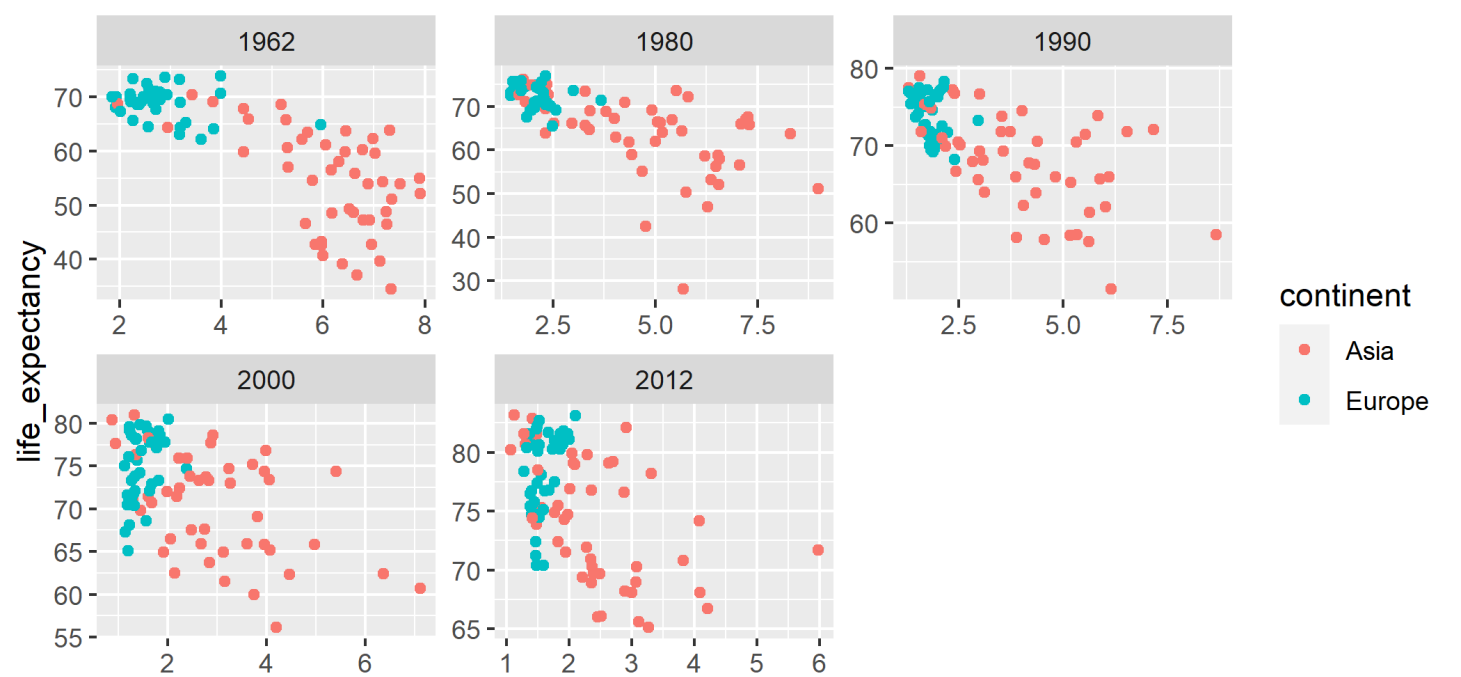
```
years = c(1962, 1980, 1990, 2000, 2012)
continents = c("Europe", "Asia")
gapminder %>%
  filter(year %in% years & continent %in% continents) %>%
  ggplot( aes(fertility, life_expectancy, col = continent)) +
    geom_point() +
    facet_wrap(~year, nrow = 2) # can also use ncol
```



Faceting

An important thing `facet` gives us is **common axis scales**. Otherwise graphs look like this:

```
years = c(1962, 1980, 1990, 2000, 2012)
continents = c("Europe", "Asia")
gapminder %>%
  filter(year %in% years & continent %in% continents) %>%
  ggplot( aes(fertility, life_expectancy, col = continent)) +
    geom_point() +
    facet_wrap(~year, scales = "free")
```



Cleveland Dot Plots

Cleveland dot plots are uncluttered and can be **more effective** than **bar/column charts**.

- Especially when the x-intercept doesn't mean much.
- Or when plotting multiple values per category.

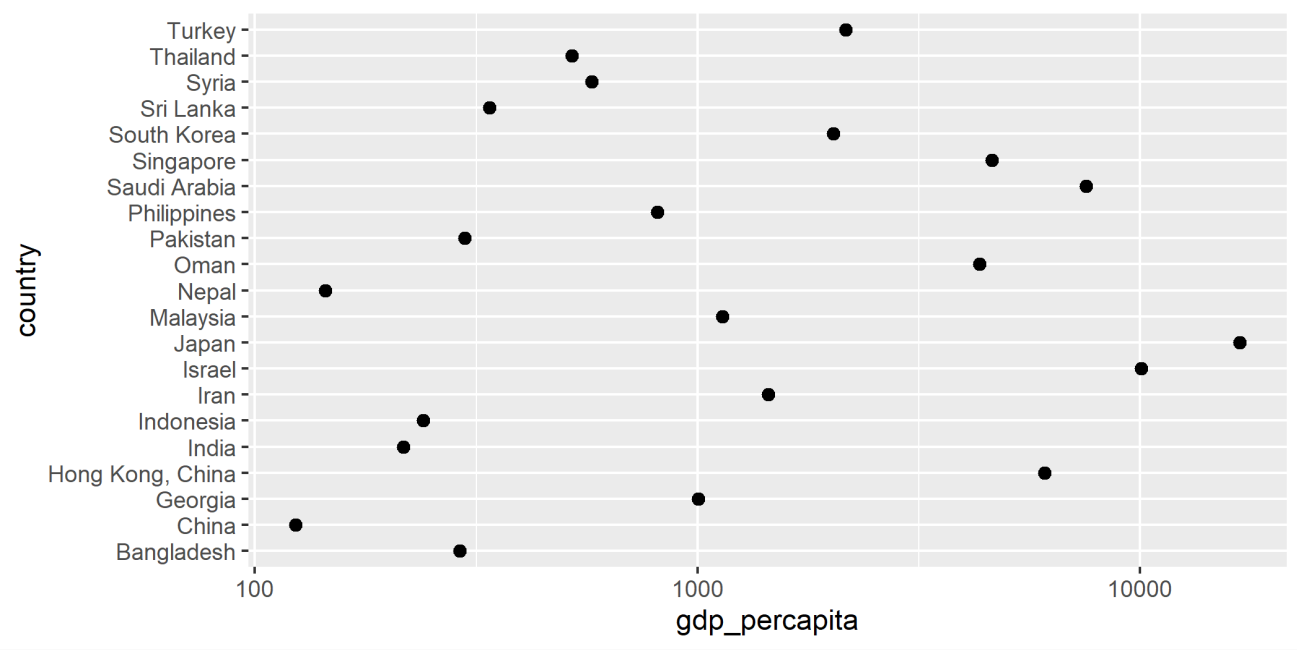
Structure:

- Categorical variable on y axis (easiest with factor)
- points as values on x axis

```
gapminder %>% mutate(gdp_percapita = gdp/population) %>%
  filter(year == 1970 & !is.na(gdp_percapita) & continent=="Asia") %>%
  mutate() %>%
  ggplot(aes(gdp_percapita, country)) +
    geom_point(size=2) +
    scale_x_log10() # log 10 scale for ease of viewing
```

Cleveland Dot Plots

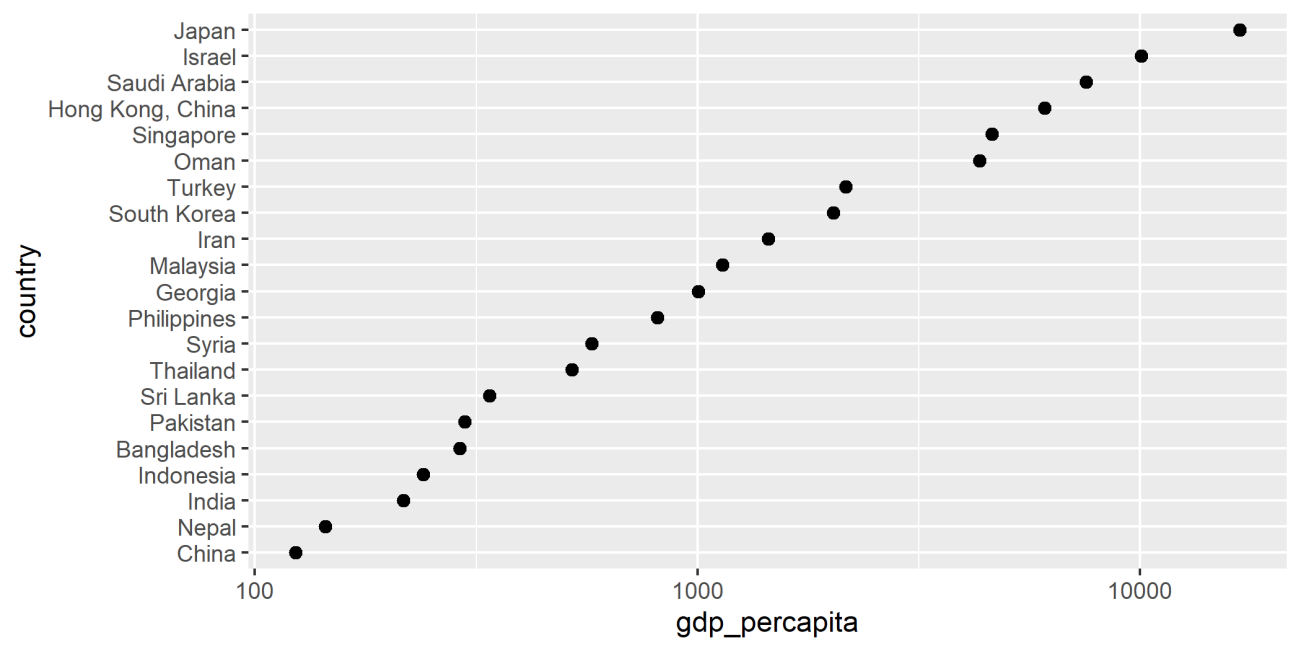
Cleveland dot plots are uncluttered and can be more effective than bar charts.



Cleveland Dot Plots

Use `reorder` to conditionally reorder a factor (i.e. countries by GDP)

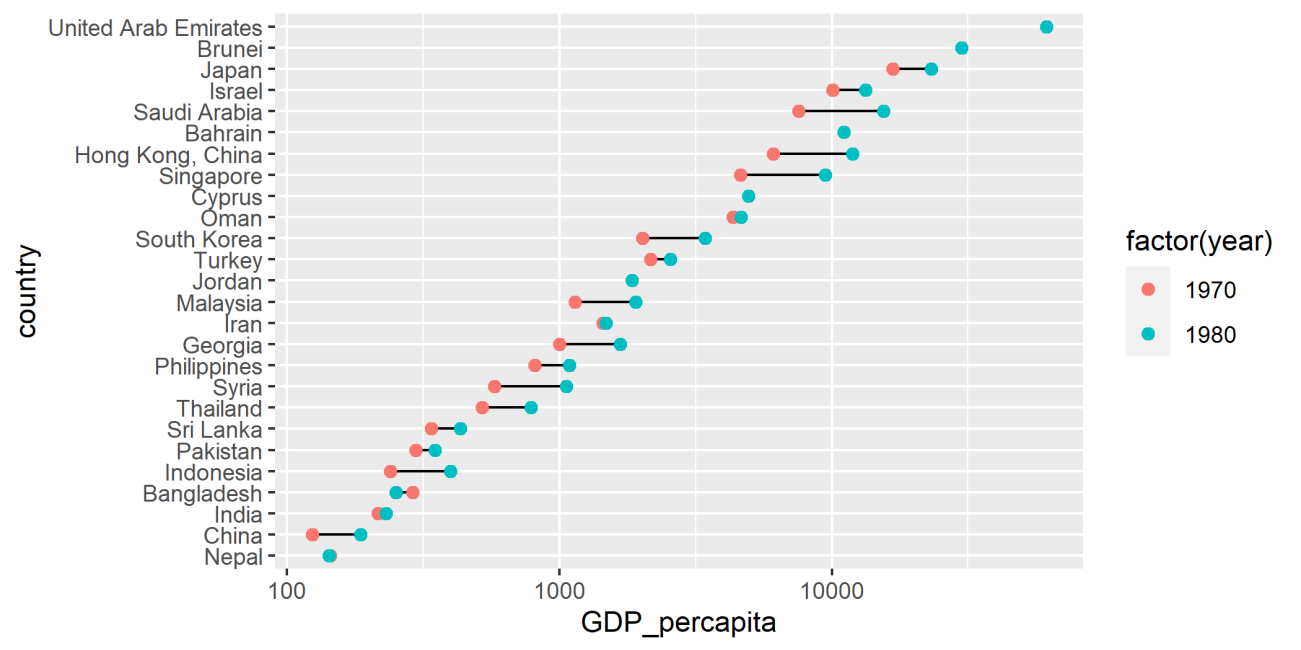
```
gapminder %>% mutate(gdp_percapita = gdp/population) %>%
  filter(year == 1970 & !is.na(gdp_percapita) & continent=="Asia") %>%
  mutate(country = reorder(country, gdp_percapita)) %>%
  ggplot(aes(gdp_percapita, country)) +
    geom_point(size=2) +
    scale_x_log10()
```



Cleveland Dot Plots

Also useful when plotting **multiple values per category**.

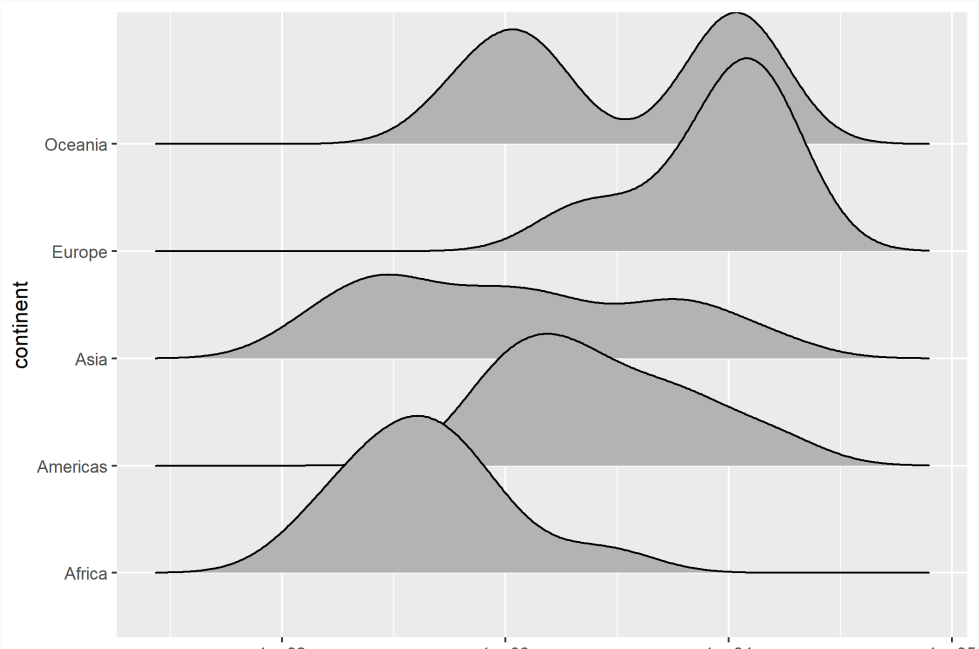
```
gapminder %>% mutate(GDP_percapita = gdp/population) %>%
  filter(year %in% c(1970, 1980) & !is.na(GDP_percapita) & continent=="Asia") %>%
  mutate(country = reorder(country, GDP_percapita)) %>%
  ggplot(aes(GDP_percapita, country)) +
    geom_line(aes(group = country)) + # add line connecting gdp values per country
    geom_point(size=2, aes(color = factor(year))) + # add point for 1970 + 1980 values on top of l
    scale_x_log10()
```



Ridge Plots

Using **ggridges** for staggered densities (a la **Joy Division's "Unknown Pleasures"**)

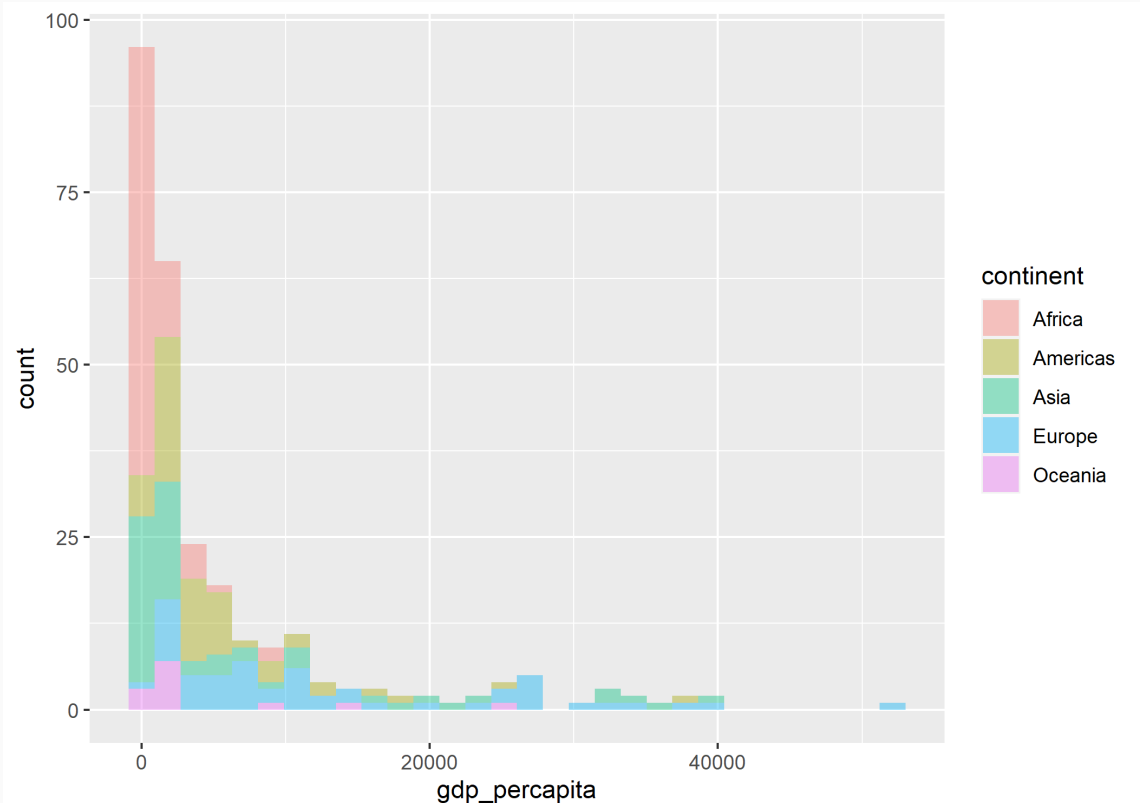
```
pacman::p_load(ggridges)
gapminder %>% mutate(gdp_percapita = gdp/population) %>%
  filter(year == 1970 & !is.na(gdp_percapita)) %>%
  ggplot(aes(gdp_percapita, continent)) +
    geom_density_ridges() +
    scale_x_log10()
```



Histograms with *geom_hist*

In this case histograms are hard to view with so many continents

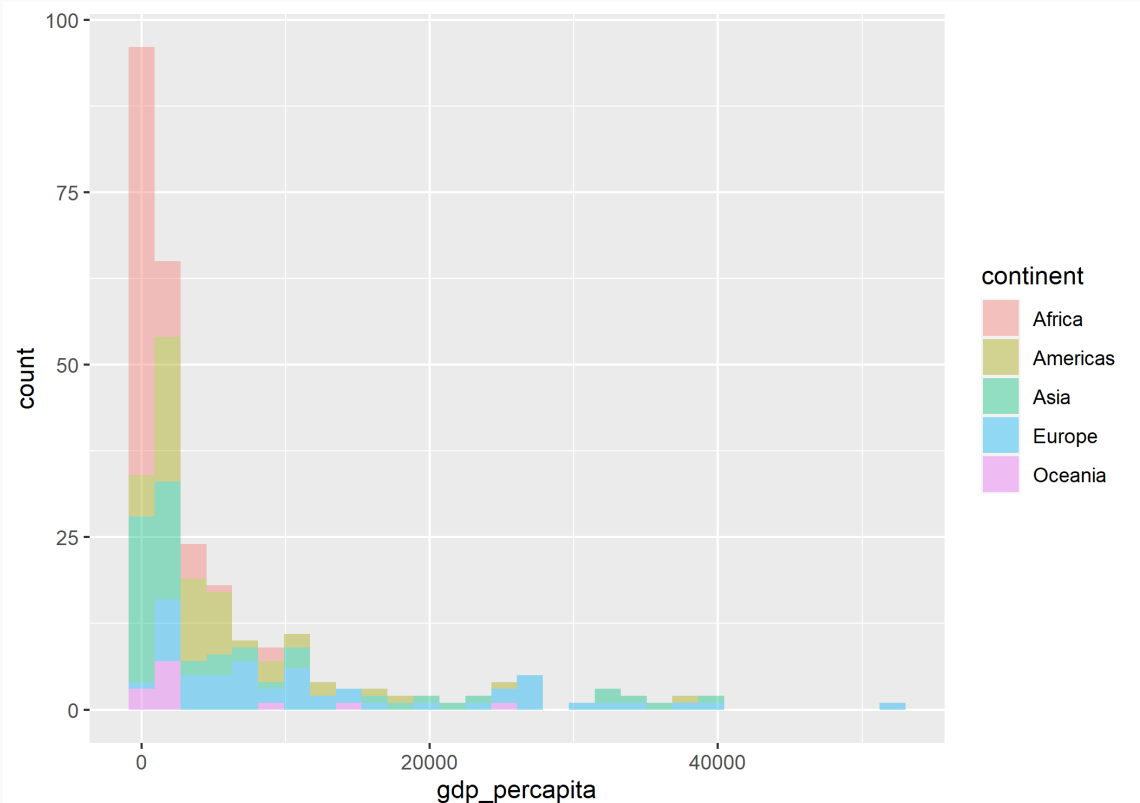
```
gapminder %>% mutate(gdp_percapita = gdp/population) %>%
  filter(year %in% c(1960, 2010) & !is.na(gdp_percapita)) %>%
  ggplot(aes(gdp_percapita, fill = continent)) +
    geom_histogram(alpha=0.4)
```



Histograms with *geom_hist*

In this case histograms are hard to view with so many continents

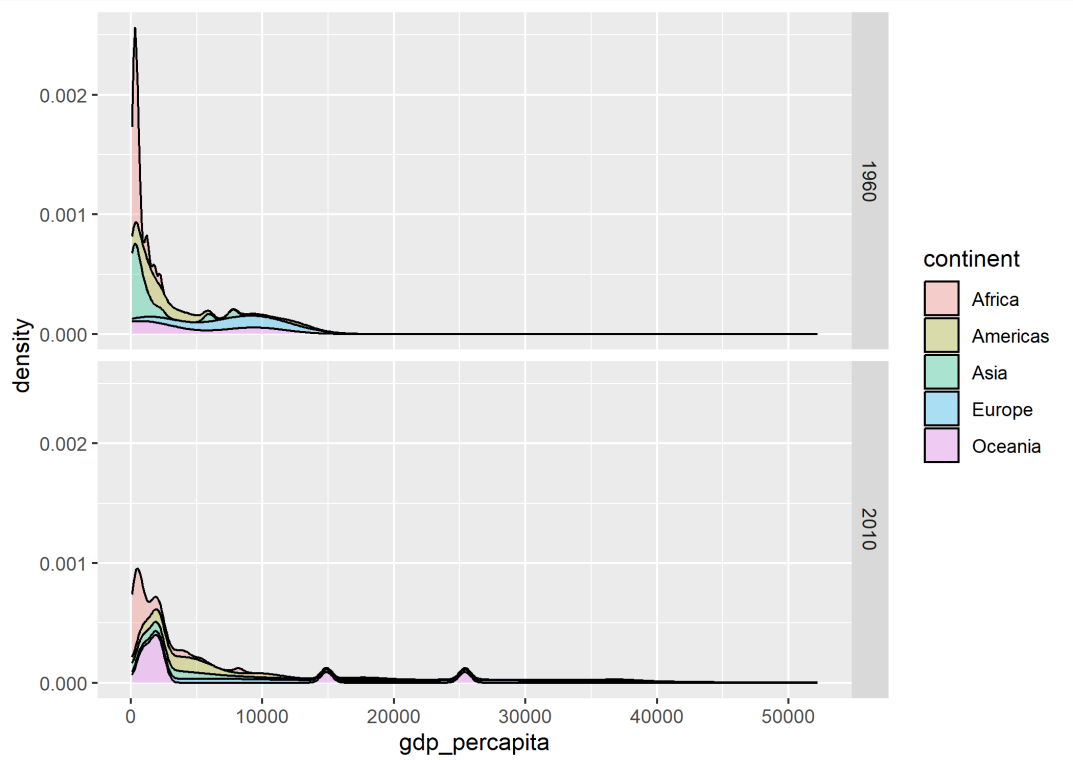
```
gapminder %>% mutate(gdp_percapita = gdp/population) %>%
  filter(year %in% c(1960, 2010) & !is.na(gdp_percapita)) %>%
  ggplot(aes(gdp_percapita, fill = continent)) +
    geom_histogram(alpha=0.4)
```



Stacked Density Plots

Stacked densities on [0,1] scale are similarly hard to view

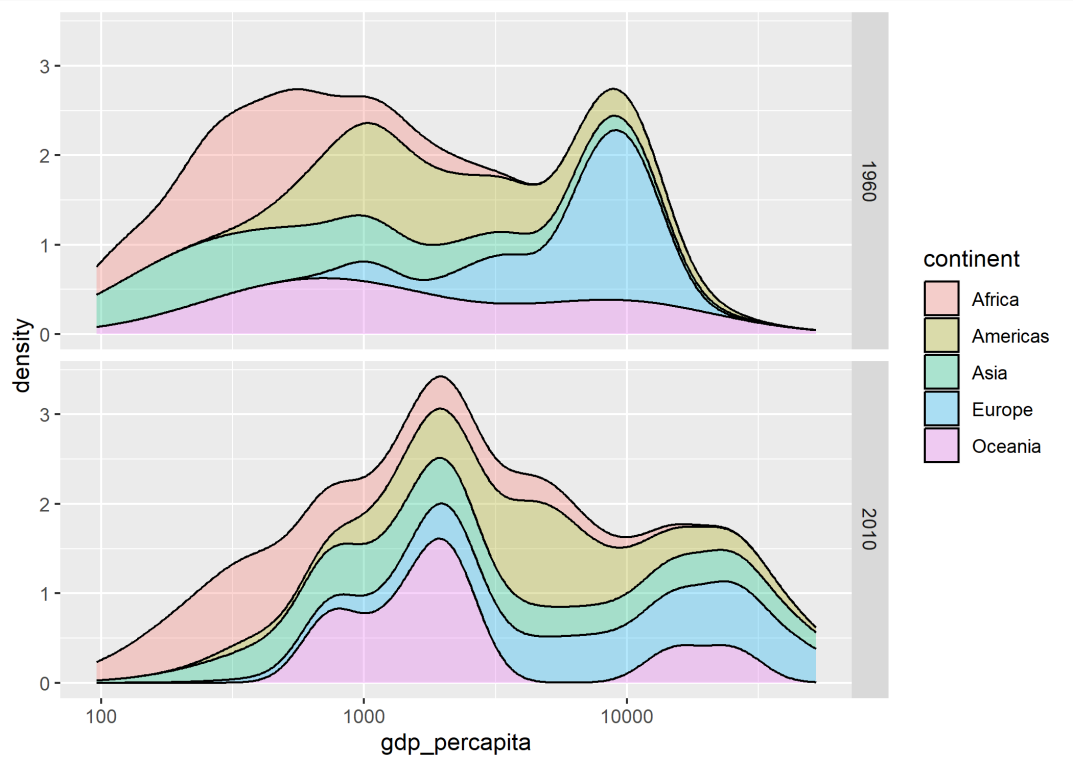
```
gapminder %>% mutate(gdp_percapita = gdp/population) %>%
  filter(year %in% c(1960, 2010) & !is.na(gdp_percapita)) %>%
  ggplot(aes(gdp_percapita, fill = continent)) +
    geom_density(alpha=0.3, position = "stack") +
    facet_grid(year ~ .)
```



Stacked Density Plots

Easier to compare with log 10 transformation

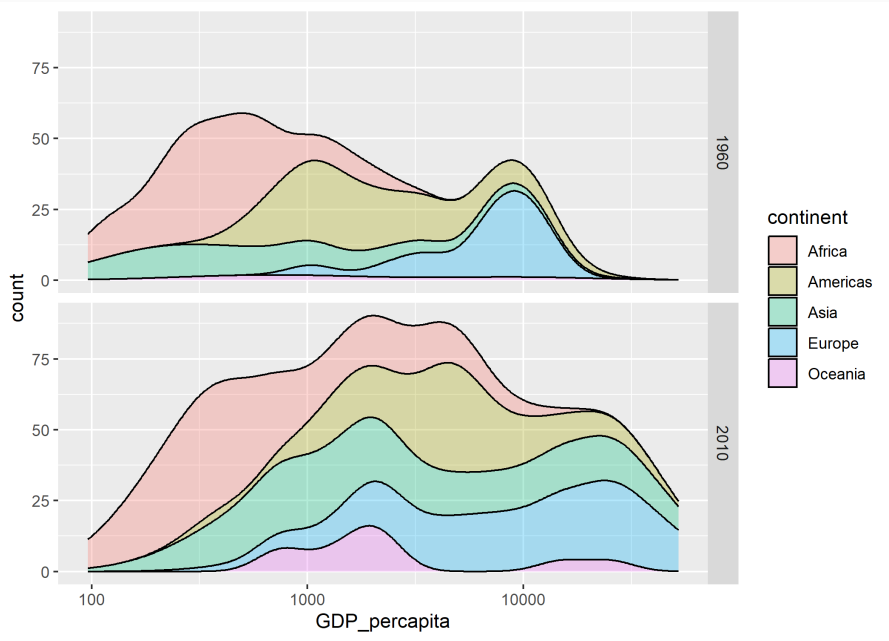
```
gapminder %>% mutate(gdp_percapita = gdp/population) %>%
  filter(year %in% c(1960, 2010) & !is.na(gdp_percapita)) %>%
  ggplot(aes(gdp_percapita, fill = continent)) +
    geom_density(alpha=0.3, position = "stack") +
    facet_grid(year ~ .) + scale_x_log10()
```



Stacked Density Plots

Scale each continent by its number of countries using the **computed variable** `..count..`:

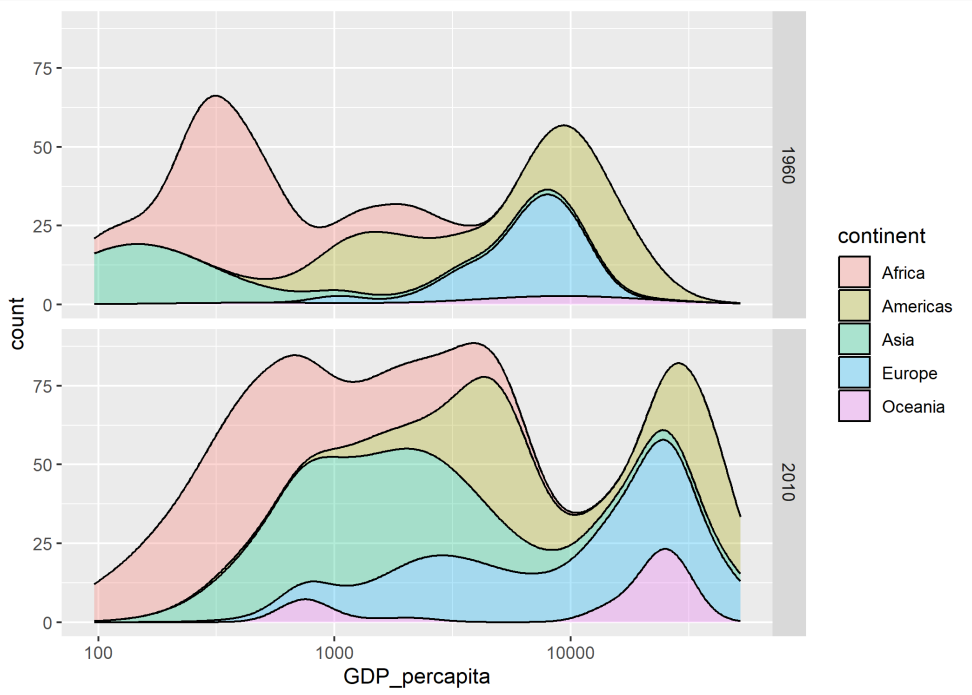
```
gapminder %>% mutate(GDP_percapita = gdp/population) %>%
  filter(year %in% c(1960, 2010) & !is.na(GDP_percapita)) %>%
  ggplot(aes(GDP_percapita, y=..count.., fill = continent)) +
    scale_x_continuous(trans = "log10") +
    geom_density(alpha=0.3, position = "stack") +
    facet_grid(year ~ .) + scale_x_log10()
```



Stacked Density Plots

And weight countries within each continent by its population:

```
gapminder %>% mutate(GDP_percapita = gdp/population) %>%
  filter(year %in% c(1960, 2010) & !is.na(GDP_percapita)) %>%
  ggplot(aes(GDP_percapita, y=..count.., weight=population, fill = continent)) +
    geom_density(alpha=0.3, position = "stack") +
    facet_grid(year ~ .) +
    scale_x_log10()
```



Exporting Graphs

Exporting Graphs

You can save your **ggplot2**-based figures using `ggsave()`.

Option 1: By default, `ggsave()` saves the last plot printed to the screen.

- i.e. what's currently displayed in the "Plots" viewer
  - saves to the working directory

```
# Create a simple scatter plot
filter(gap_full, year == 2000) %>%
ggplot(aes(x = fertility, y = life_expectancy)) +
geom_point()

# Save our simple scatter plot
ggsave(filename = "simple_scatter.pdf")
```

- This example creates a PDF. Change to `".png"` for PNG, etc.
- Optional arguments: `path`, `width`, `height`, `dpi`.

Exporting Graphs

For example, to save a print quality (300 dpi) 8in by 6in `".png"` file to the "output" subfolder, we can modify our `ggsave` call to

```
# Save our simple scatter plot
ggsave(filename = "simple_scatter.png",
       path = "output/",
       width = 8,
       height = 6,
       units = "in",
       dpi = "print"
       )
```

Exporting Graphs

Option 2: You can first assign your `ggplot()` objects to memory:

```
# Create a simple scatter plot named 'gg_points'
points_2010 ← filter(gap_full, year == 2010) %>%
  ggplot(aes(x = fertility, y = life_expectancy)) +
  geom_point()
```

And then save this figure by name using the `plot` argument:

```
# Save our simple scatter plot name 'ggsave'
ggsave(
  # can add subfolder directly in filename
  filename = "output/simple_scatter.pdf",
  plot = points_2010
)
```

Exporting Graphs

In what format should you save your graphics?

Vector graphics are composed of **formulas or paths**.

- "Draw a straight line from (0, 0) to (13, 4)."
- Infinitely zoom-able. Preserves all underlying information.
- May be slow to load when complex.
- Fully modifiable in vector art software (i.e. Adobe Illustrator)
- `.pdf` or `.svg`.

Raster graphics are composed of **pixels** (a grid of squares with color information).

- Only an approximation to the underlying shapes or points.
- Work better with Microsoft Office and HTML.
- The original format of photographs.
- Usually best: `.png`. Also `.jpeg`, `.gif`.

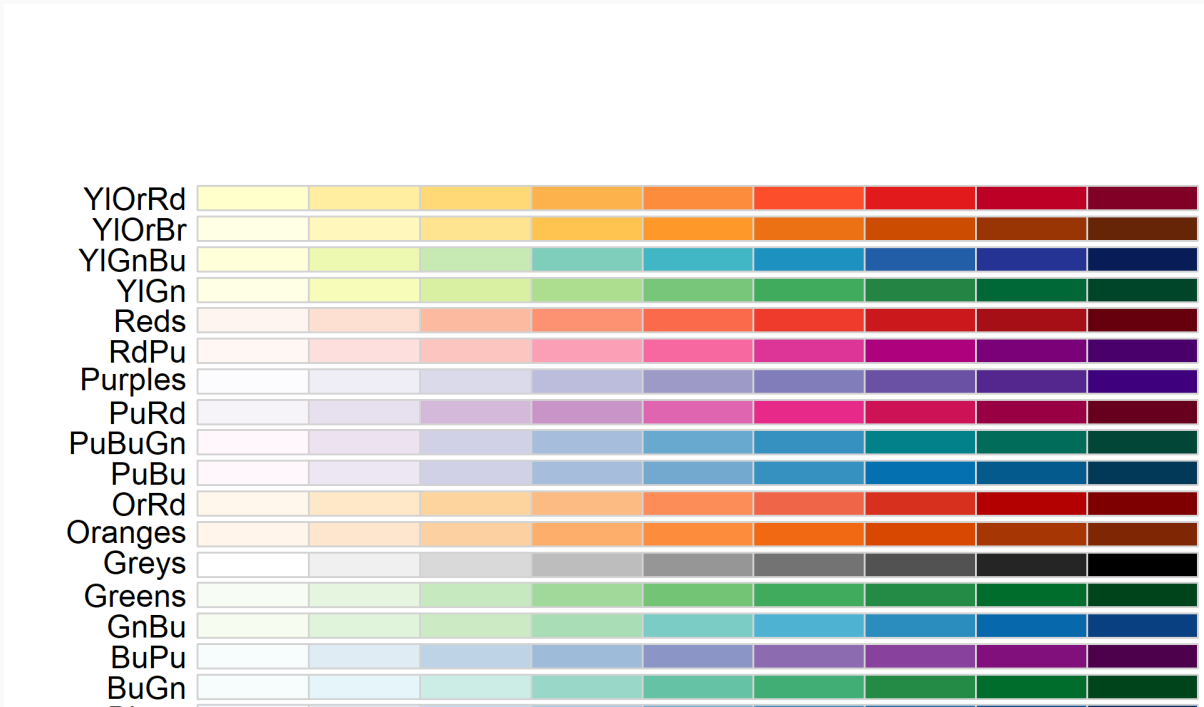
Colors Schemes

Color Schemes

Choose a **sequential** color scheme when your values are ordered in **only one direction**.

- Low to high; values are all positive; zero is defined arbitrarily.

```
pacman::p_load(RColorBrewer)
display.brewer.all(type="seq")
```

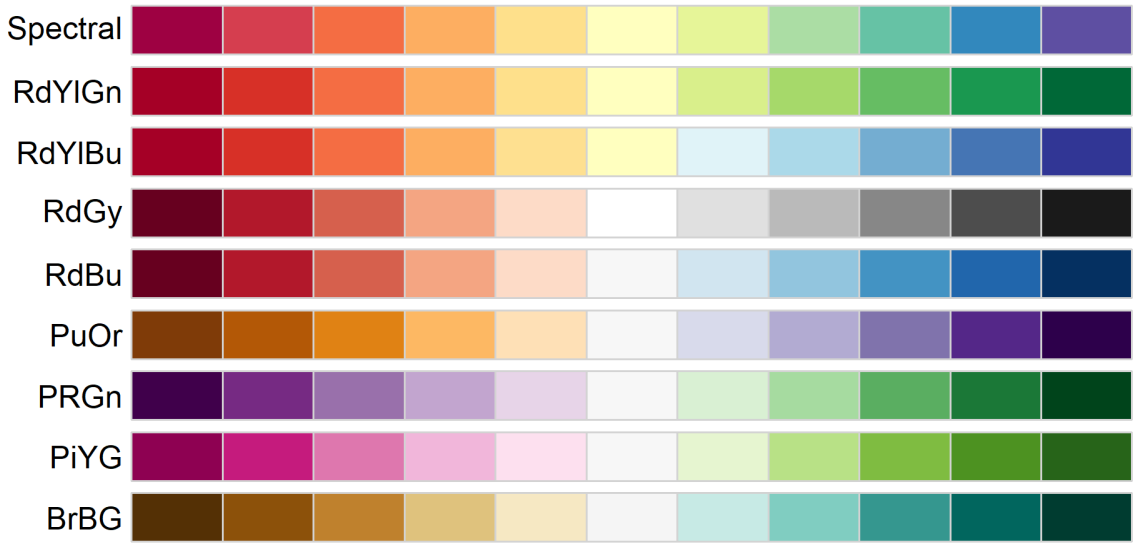


Color Schemes

Choose a **diverging** color scheme when your values are ordered in **two directions relative to a center**.

- Positive vs. negative; vote shares relative to 50%.

```
display.brewer.all(type="div")
```



Color Schemes

Choose a **qualitative** color scheme when your values have **no ordering**.

- Only need to distinguish among categories.

```
display.brewer.all(type="qual")
```

Use Established Color Schemes

There are several great color schemes available in R created by professional visual designers.

- **RColorBrewer** is based on the research of cartographer Cynthia Brewer. Her **ColorBrewer** website lets you choose a color scheme by value ordering and whether you need it to be colorblind safe, printer friendly, or photocopy safe.
- **viridis** schemes are designed to span a large perceptual range while remaining perceptually uniform, robust to colorblindness, and pretty. (The next few slides show diagrams from the the package's **vignette**.)

Use Established Color Schemes

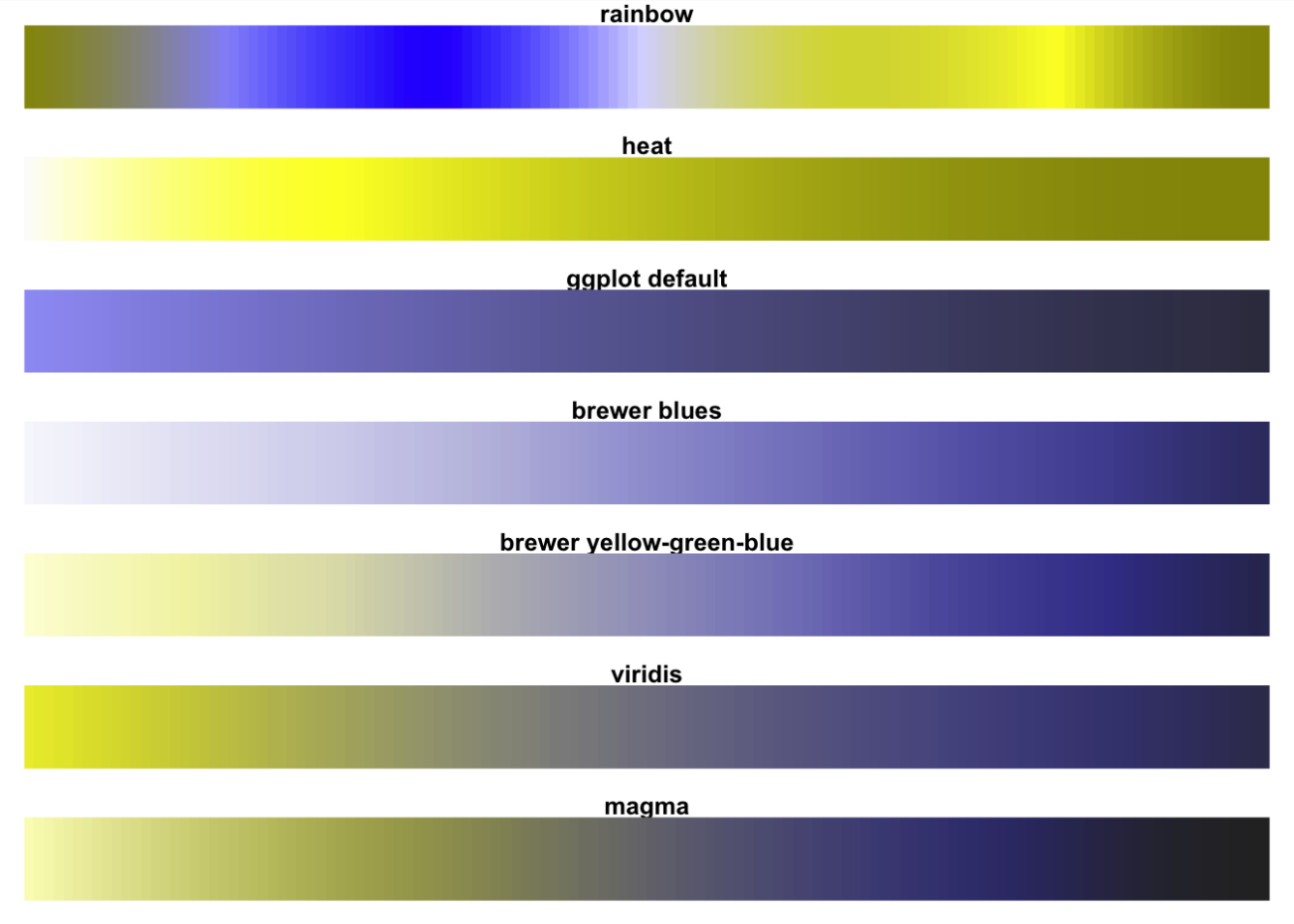
Palettes available in **viridis**:

Comparing Palettes

Compare `rainbow` and `heat` from base R, the default **ggplot2** palette, and palettes from **RColorBrewer** and **viridis**:

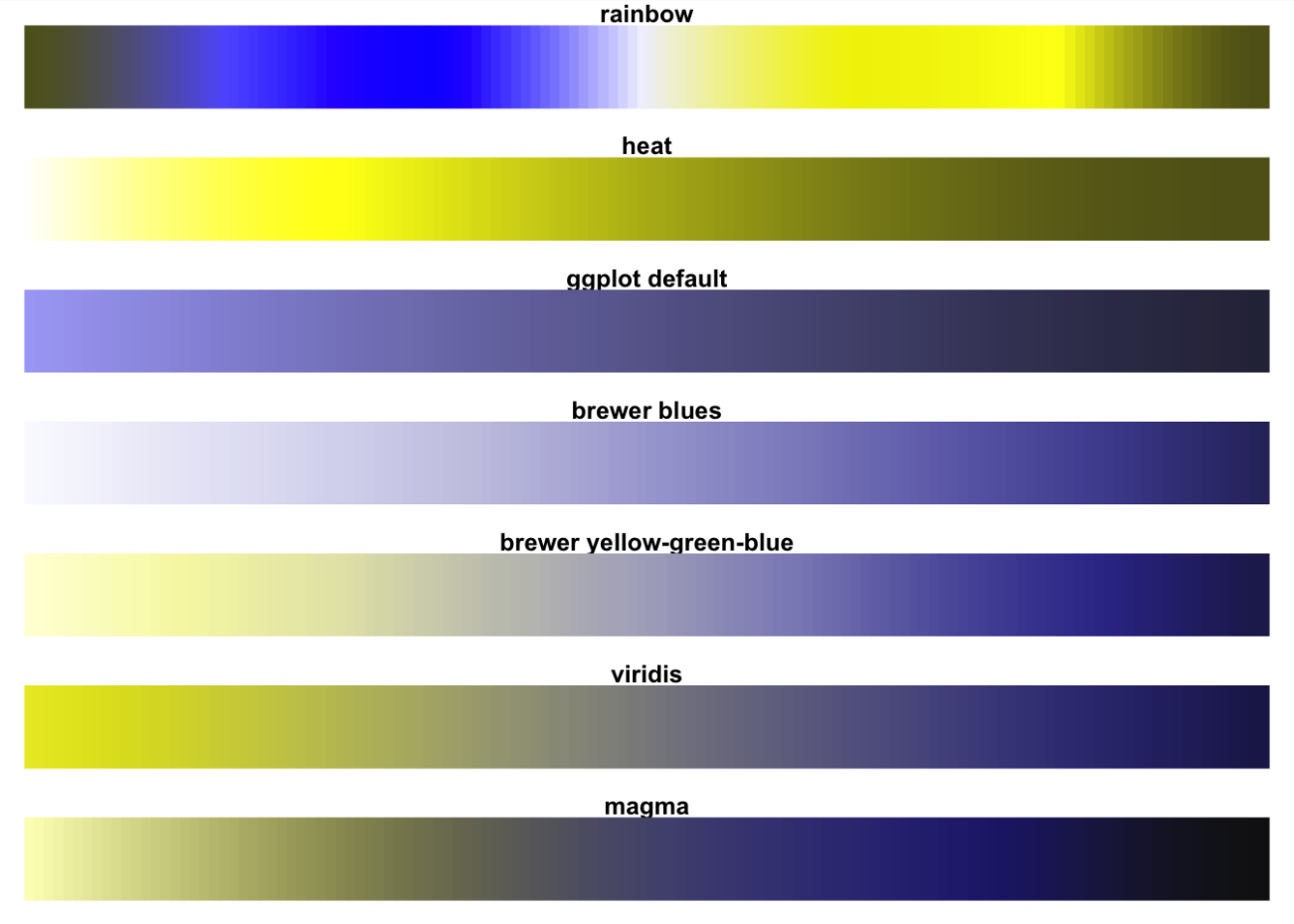
Consider Colorblindness

Green-Blind (Deuteranopia):



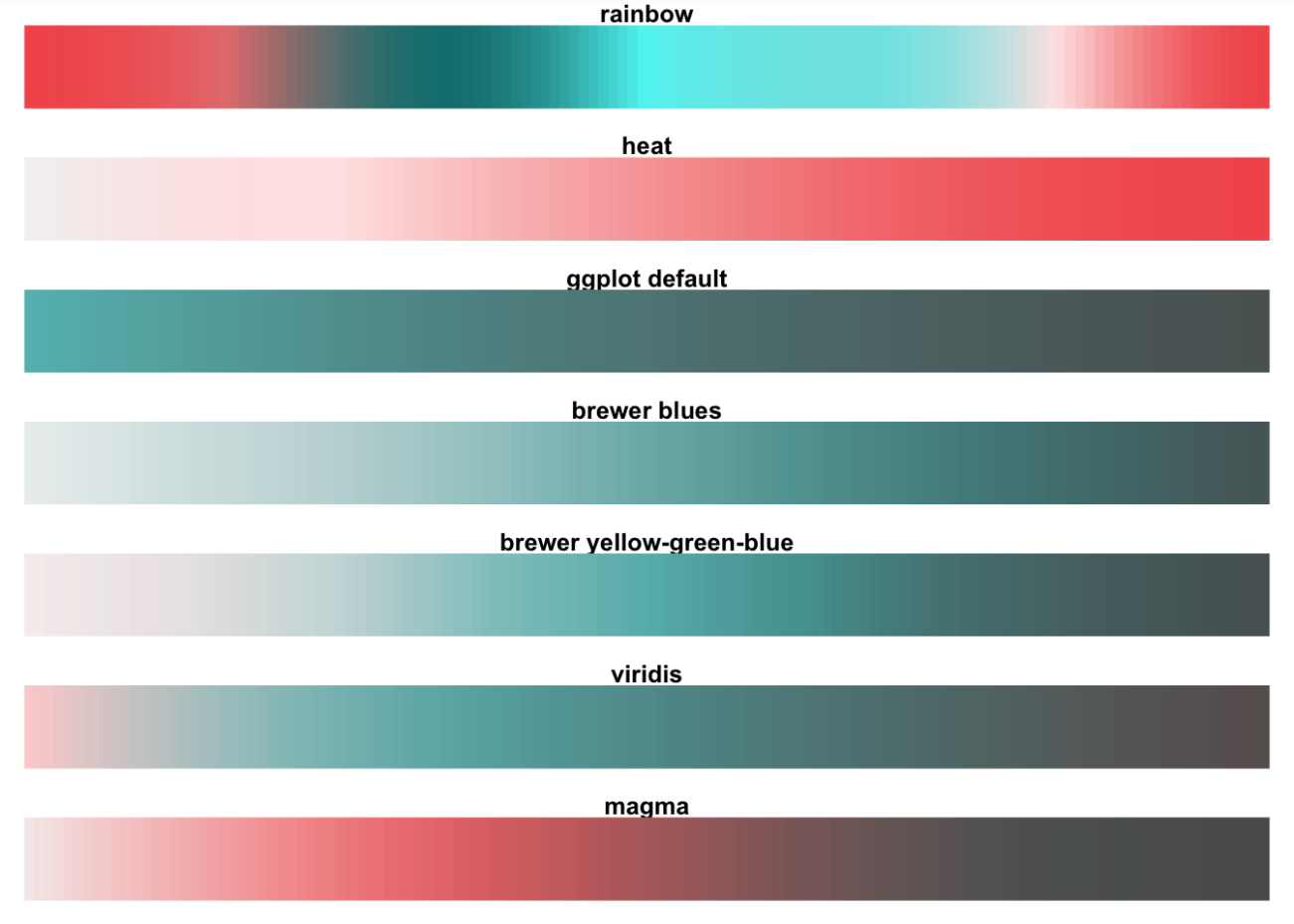
Consider Colorblindness

Red-Blind (Protanopia):



Consider Colorblindness

Blue-Blind (Tritanopia):



Consider Printer-Friendliness

Grayscale:

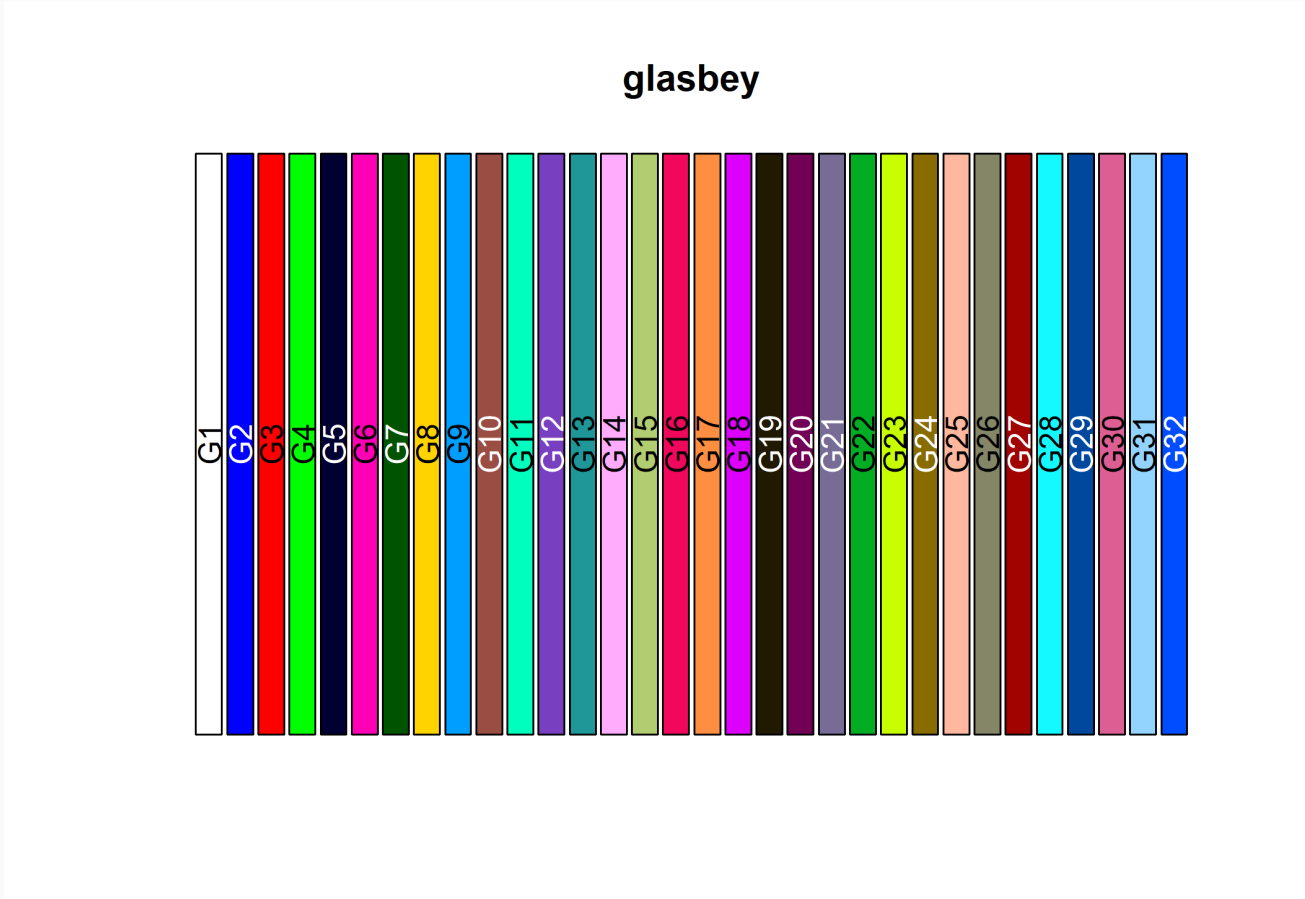
Other Established Color Schemes

Other packages for useful/fun color schemes:

- **Polychrome** has large **qualitative palettes** and functions for checking how palettes will look to a person with color deficit vision
- **broman** has Crayola crayon colors.

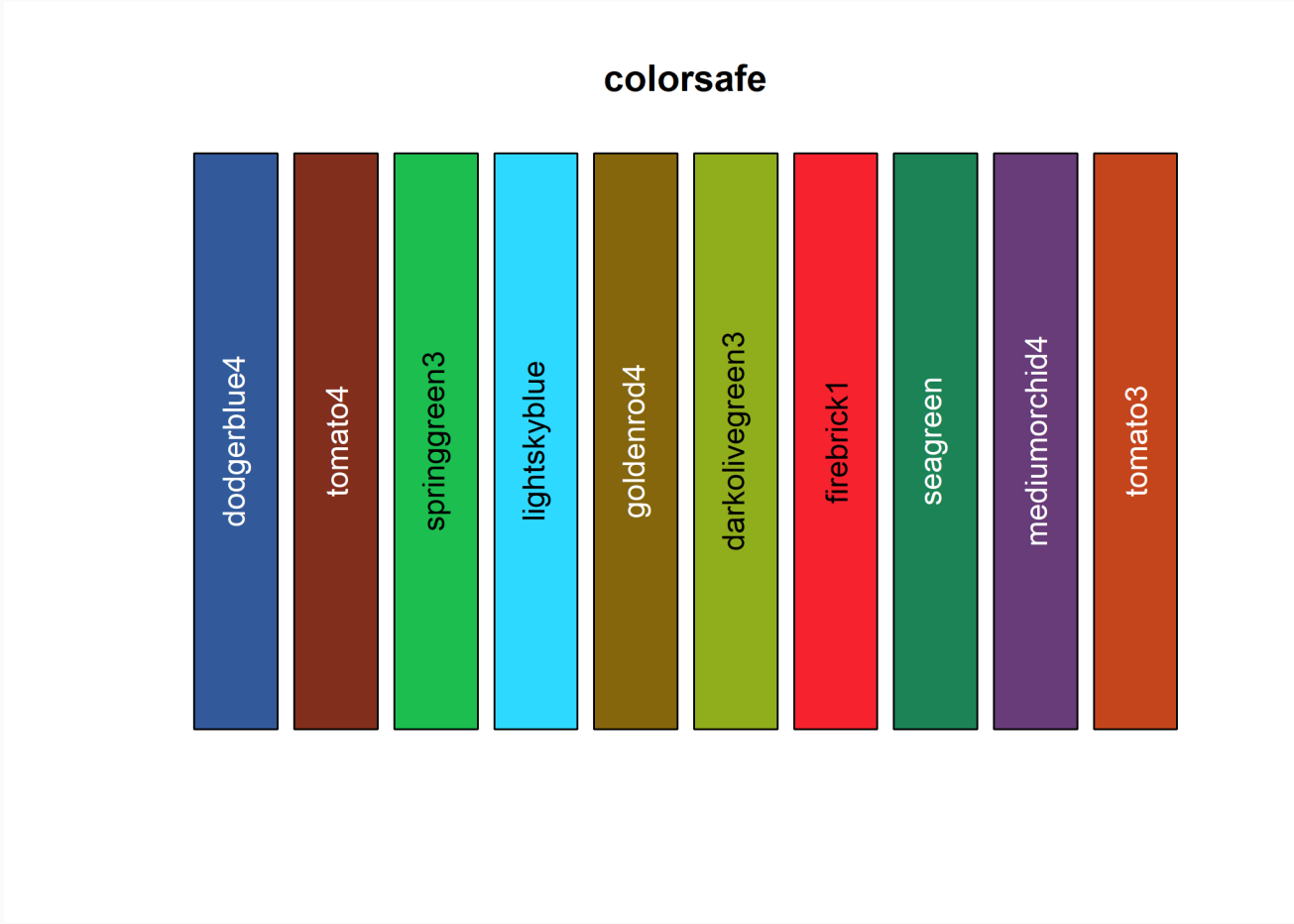
Polychrome: glasbey Palette

```
pacman::p_load(Polychrome)
data(glasbey)
swatch(glasbey)
```



Polychrome: colorsafe Palette

```
pacman::p_load(Polychrome)
data(colorsafe)
swatch(colorsafe)
```



Broman: brocolors

```
pacman::p_load(broman)
plot_crayons()
```

Themes

Themes

ggplot2's default theme is now fairly iconic - but that doesn't mean we can't do better.

Option 1: Pre-Existing Themes

- Default **ggplot2** choice is `theme_gray`
- Alternate **ggplot2** themes: `theme_bw`, `theme_linedraw`, `theme_light`, `theme_dark`, `theme_minimal`, `theme_classic`, `theme_void`
- Additional themes in **ggthemes**
  - i.e. want to replicate known appearances from .hi-dkorange[Excel/Stata, The Economist, FiveThirtyEight, or WSJ]

Themes

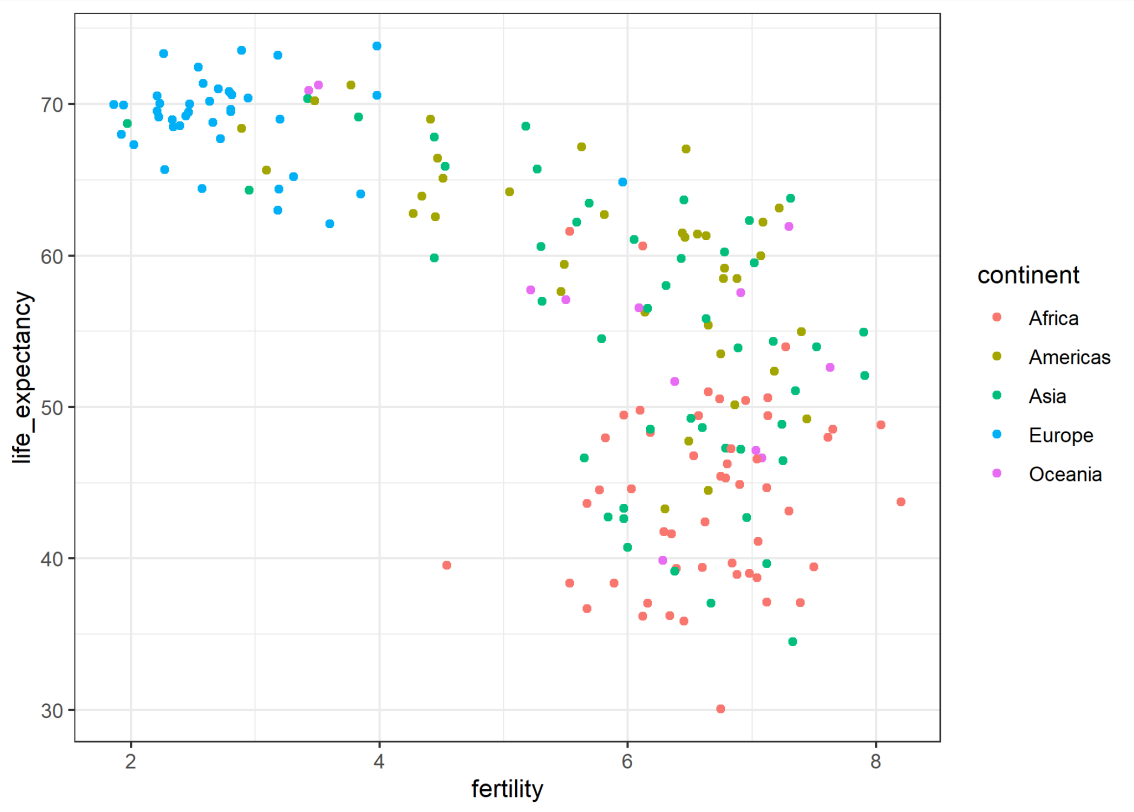
ggplot2's default theme is now fairly iconic - but that doesn't mean we can't do better.

Option 2: Create Your Own!

- Every element of the plot is customizable (gridlines, fonts, legends, margins, etc.)
- Use `theme` either on-the-fly for small tweaks or store a complete custom theme in memory for regular use

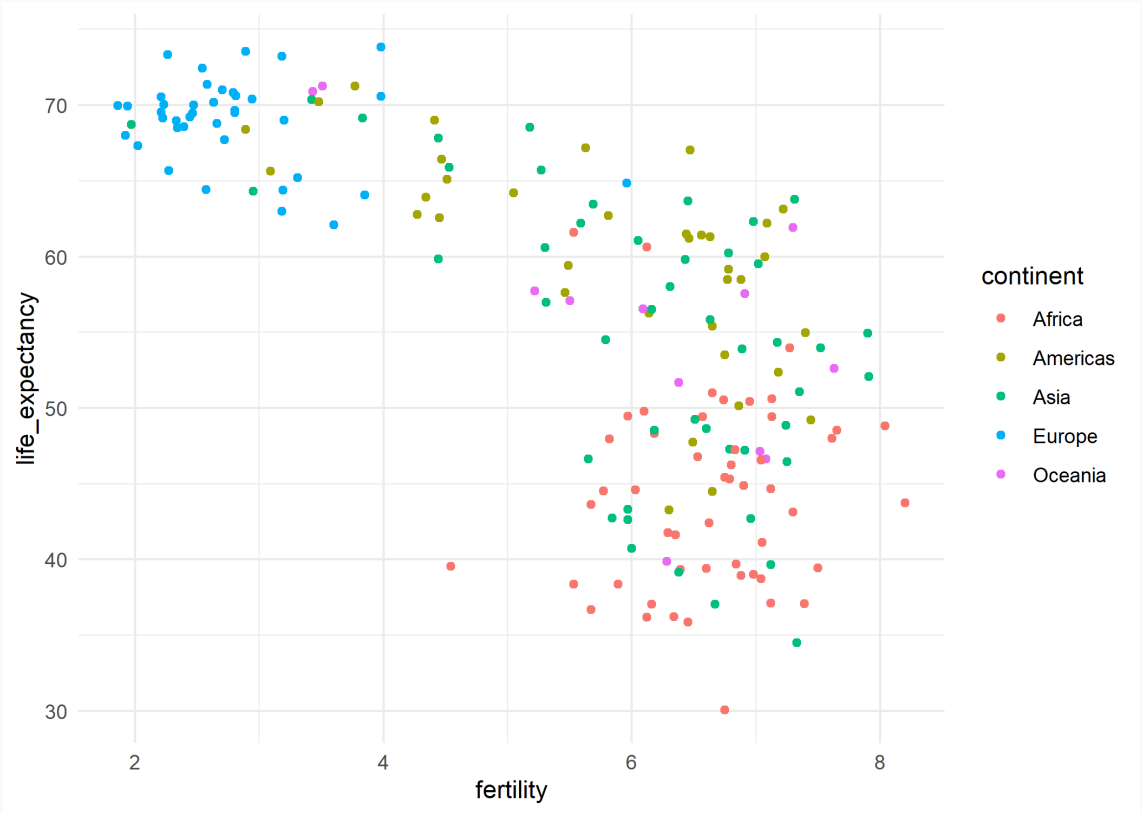
Other ggplot2 Themes: *theme_bw*

```
filter(gapminder, year == 1962) %>%
  ggplot(aes(fertility, life_expectancy, color = continent)) +
    geom_point() +
    theme_bw()
```



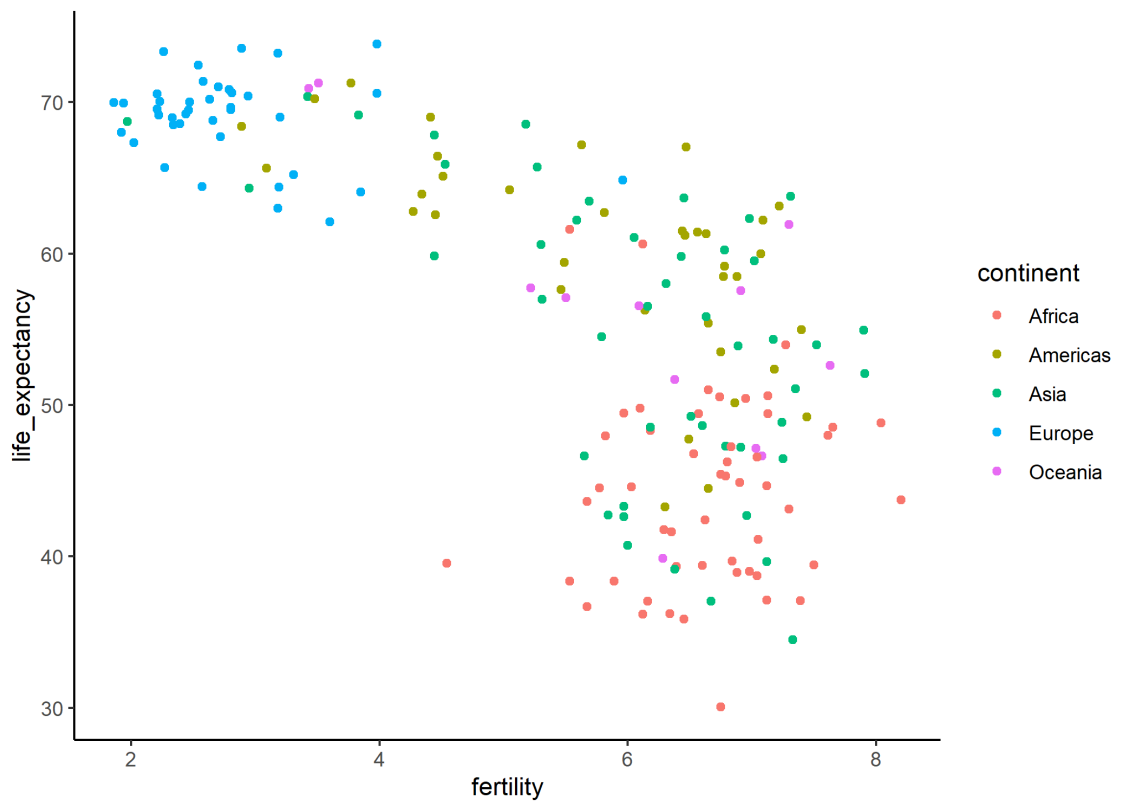
Other ggplot2 Themes: *theme_minimal*

```
filter(gapminder, year == 1962) %>%
  ggplot(aes(fertility, life_expectancy, color = continent)) +
    geom_point() +
    theme_minimal()
```



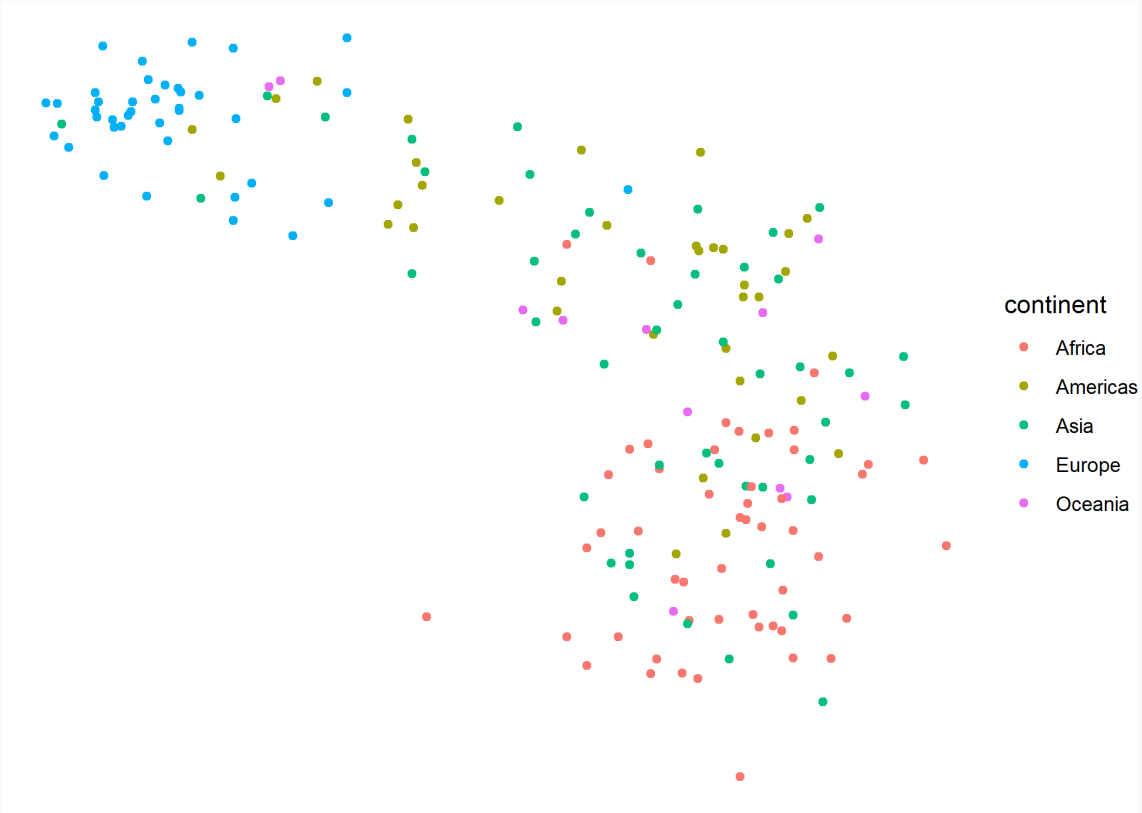
Other ggplot2 Themes: *theme_classic*

```
filter(gapminder, year == 1962) %>%
  ggplot(aes(fertility, life_expectancy, color = continent)) +
    geom_point() +
    theme_classic()
```



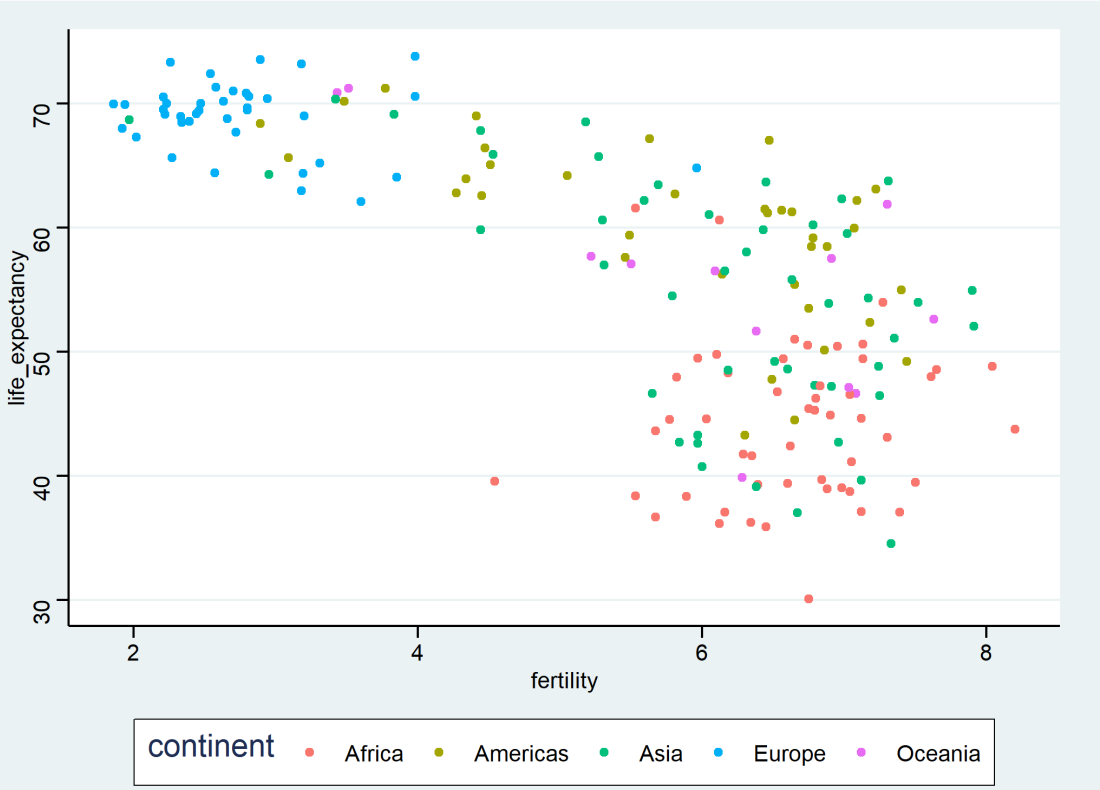
Other ggplot2 Themes: *theme_void*

```
filter(gapminder, year == 1962) %>%
  ggplot(aes(fertility, life_expectancy, color = continent)) +
    geom_point() +
    theme_void()
```



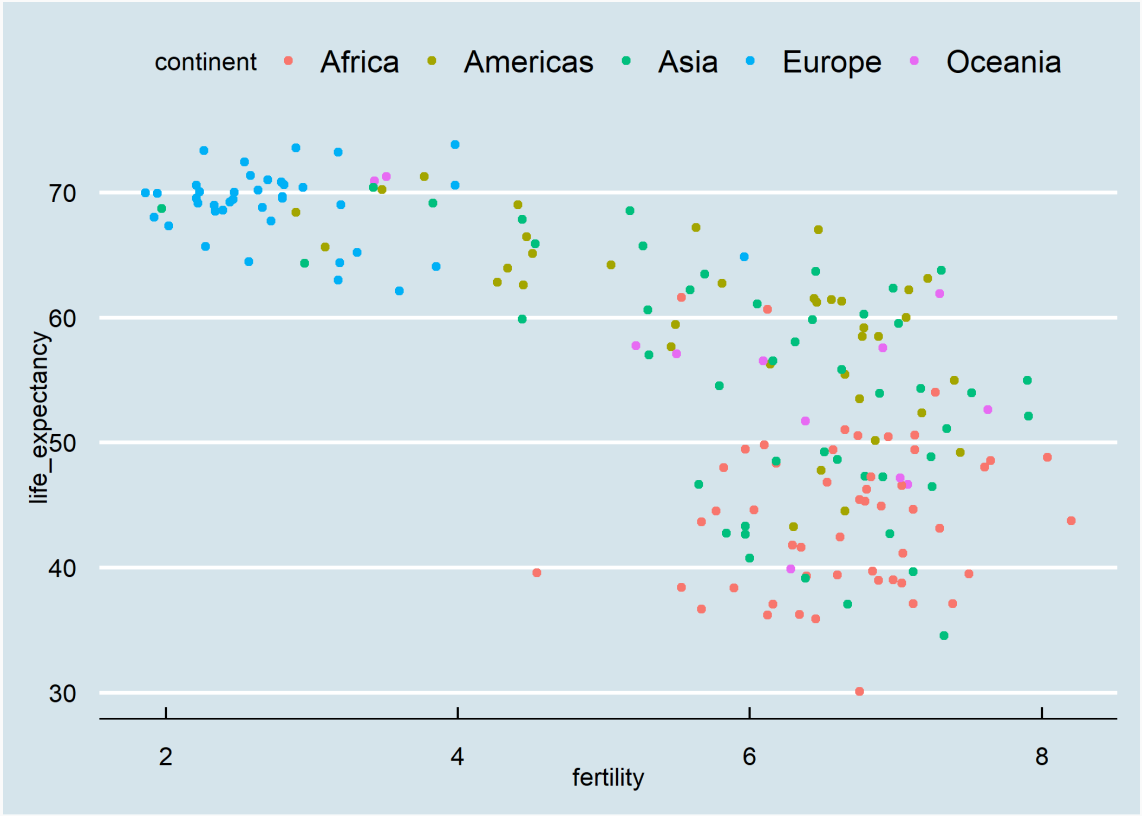
ggtheme Themes: *theme_stata*

```
pacman::p_load(ggthemes)
filter(gapminder, year == 1962) %>%
  ggplot(aes(fertility, life_expectancy, color = continent)) +
    geom_point() +
    theme_stata()
```



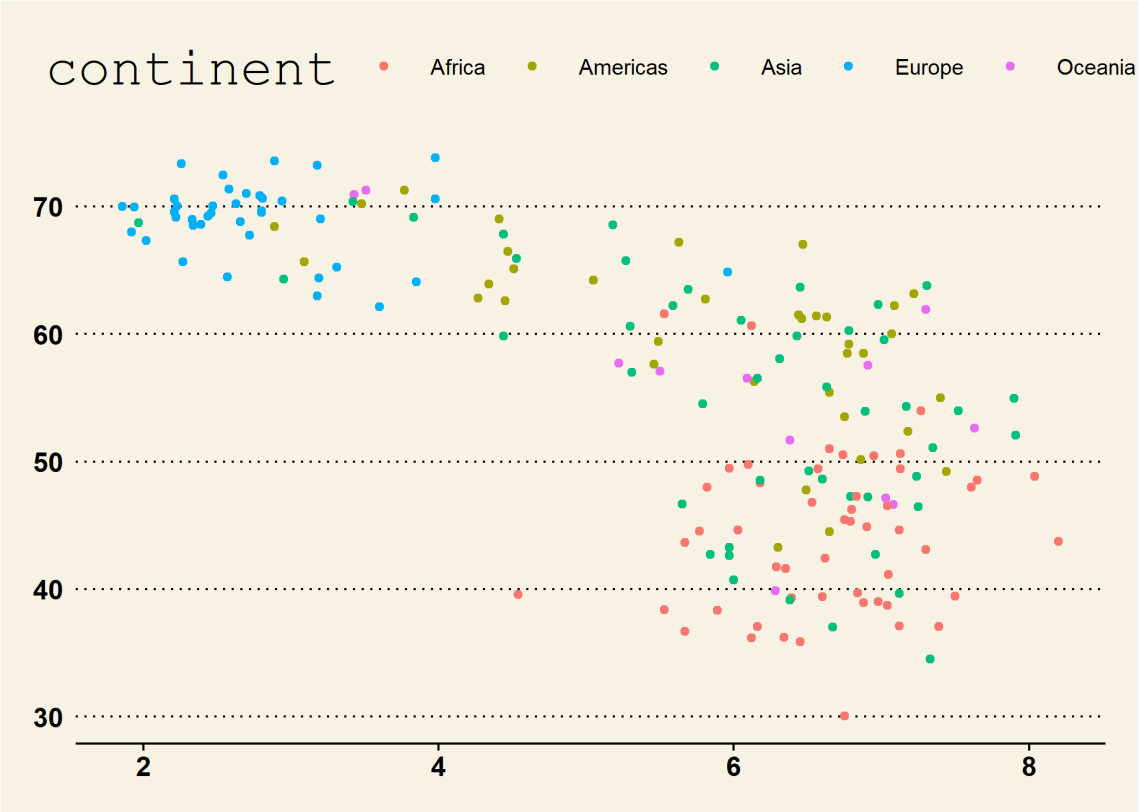
ggtheme Themes: *theme_economist*

```
filter(gapminder, year == 1962) %>%
  ggplot(aes(fertility, life_expectancy, color = continent)) +
    geom_point() +
    theme_economist()
```



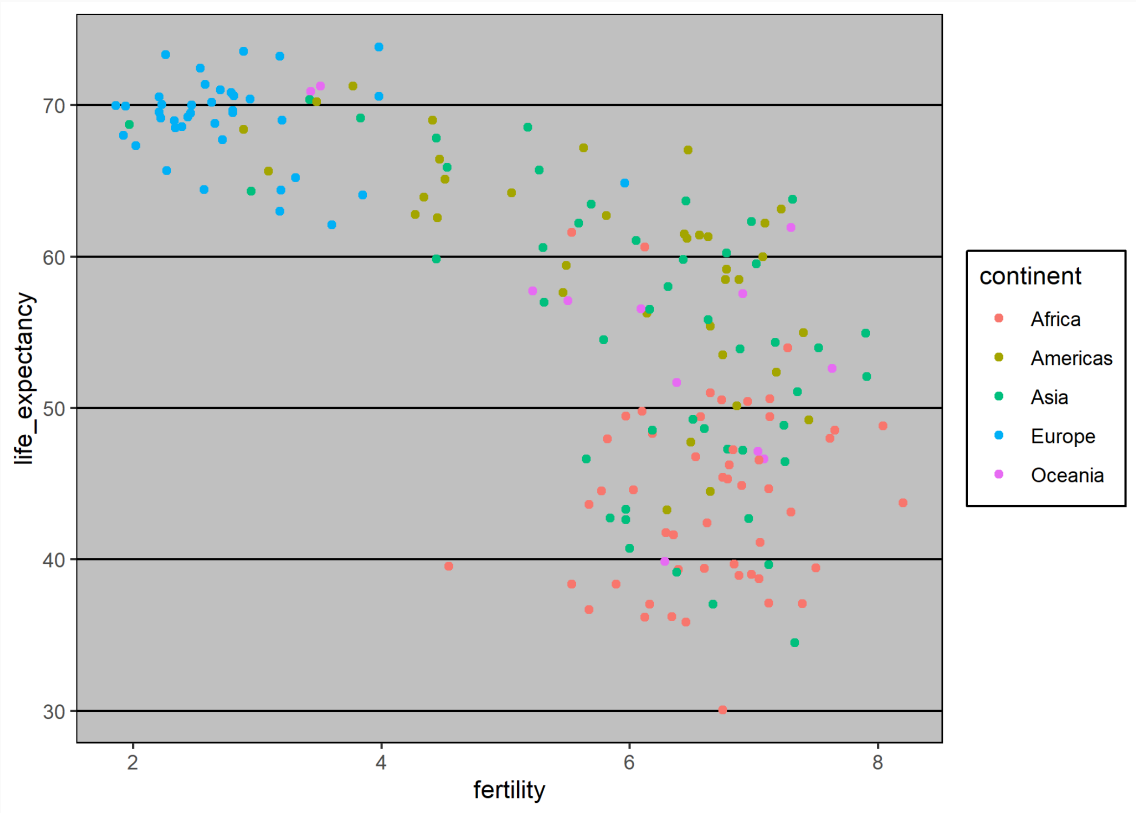
ggtheme Themes: *theme_wsj*

```
filter(gapminder, year == 1962) %>%
  ggplot(aes(fertility, life_expectancy, color = continent)) +
    geom_point() +
    theme_wsj()
```



ggtheme Themes: *theme_excel*

```
filter(gapminder, year == 1962) %>%
  ggplot(aes(fertility, life_expectancy, color = continent)) +
    geom_point() +
    theme_excel()
```



Creating Custom Themes

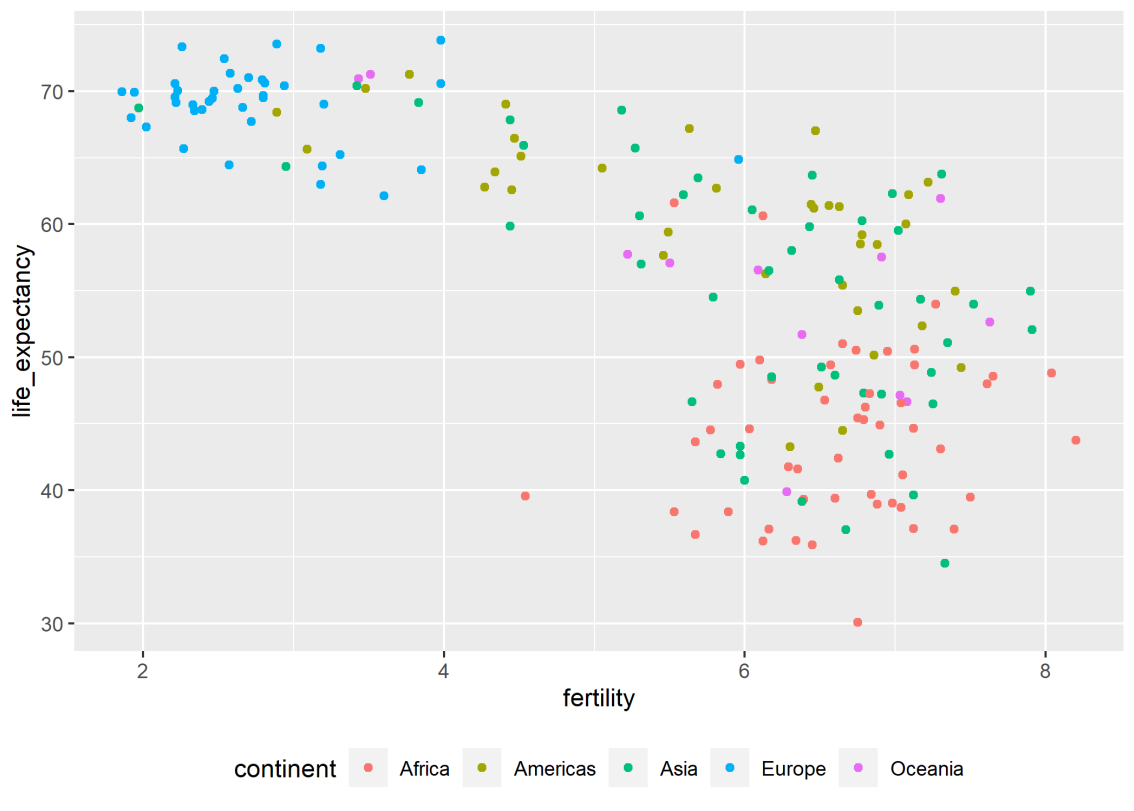
Alternatively, you can create a **custom theme** by tweaking **any and all plot elements**

- Specify elements and specifications within `theme`
- Either inline or as separate theme object in memory
- Full list of elements **here**

Custom Themes: Legend Position

```
theme_plot ← filter(gapminder, year == 1962) %>%
  ggplot(aes(fertility, life_expectancy, color = continent)) +
    geom_point()

theme_plot +
  theme(legend.position = "bottom")
```

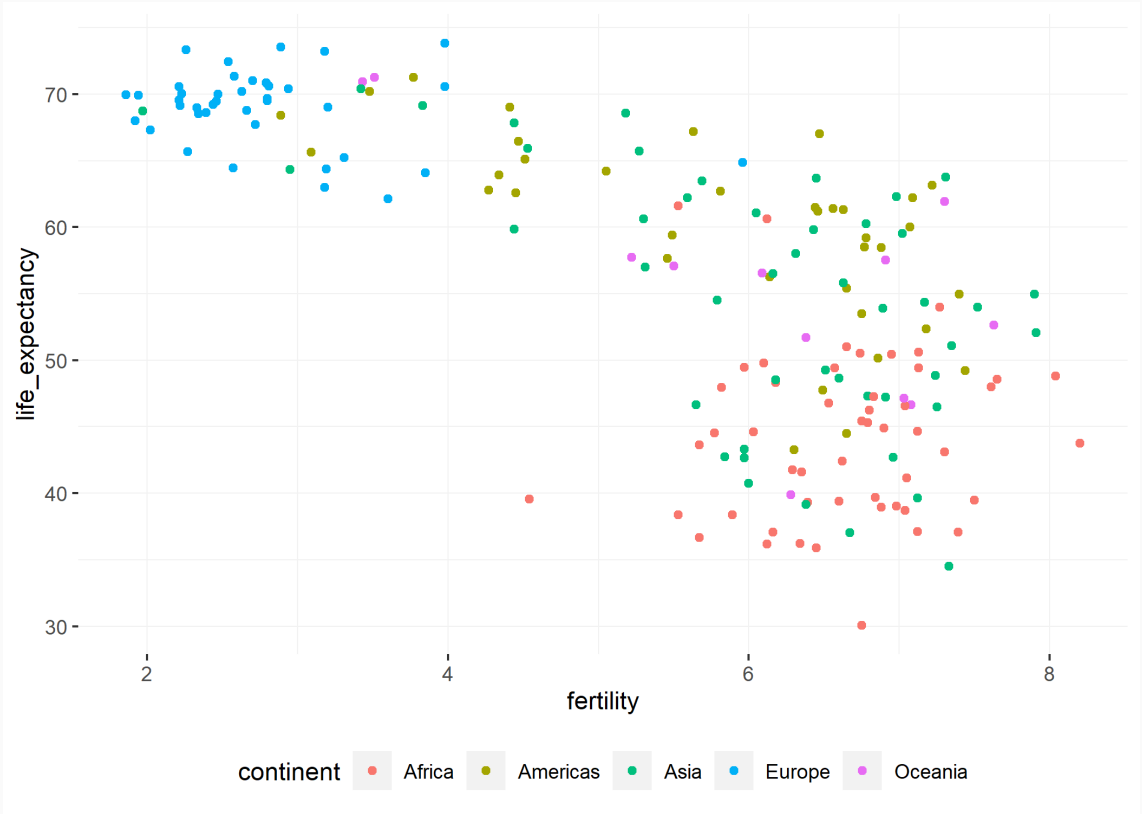


Custom Themes: Background Fill Color

```
theme_plot +
  theme(legend.position = "bottom",
        panel.background = element_rect(fill = NA)) # change to white
```

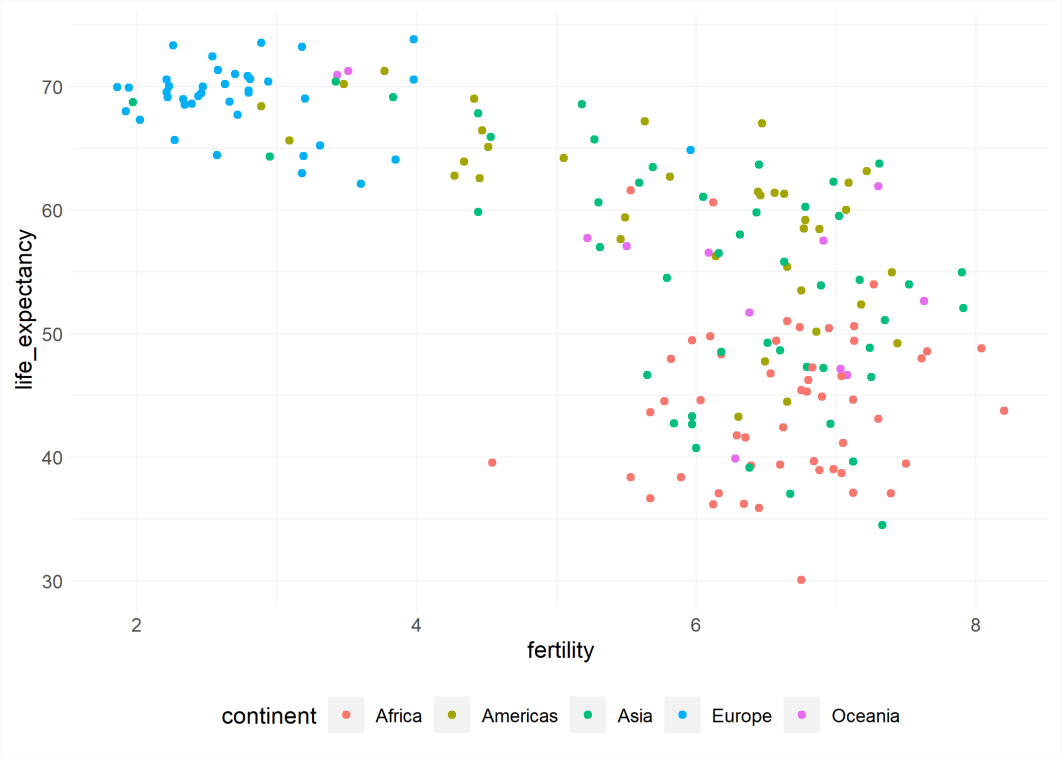
Custom Themes: Gridlines

```
theme_plot +
  theme(legend.position = "bottom",
        panel.background = element_rect(fill = NA), # change to white
         panel.grid.major = element_line(color = "grey95", linewidth = 0.3), # change color of maj
  panel.grid.minor = element_line(color = "grey95", linewidth = 0.3)) #change color of minor grid
```



Custom Themes: Tickmarks

```
theme_plot +
  theme(legend.position = "bottom",
        panel.background = element_rect(fill = NA), # change to white
         panel.grid.major = element_line(color = "grey95", linewidth = 0.3), # change color of maj
  panel.grid.minor = element_line(color = "grey95", linewidth = 0.3), #change color of minor grid
    axis.ticks = element_line(color = "grey95", linewidth = 0.3)) # make axis tick marks the same
```



Custom Themes: Fonts

Use **showtext** to add a custom font

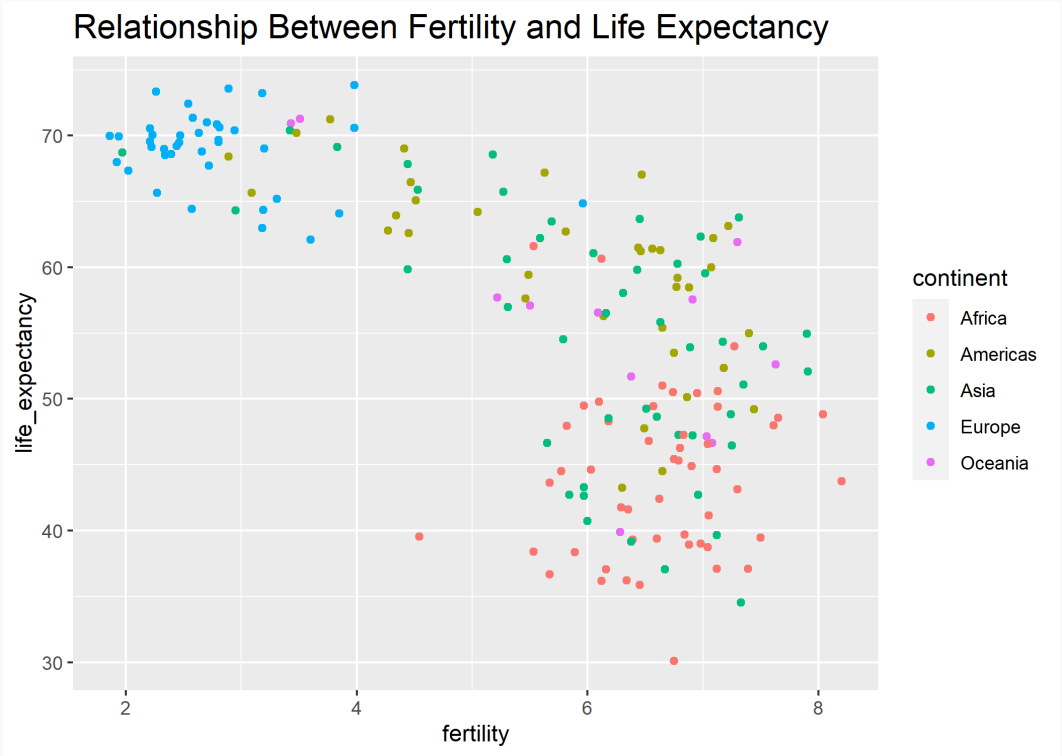
- i.e. add any **Google Font**
  - I like Lato
- `showtext_auto` to automatically use **showtext** in all plots
- `showtext_begin/end` to turn on and off when desired

```
# Name of Font Family, what we'll refer to it as in R
font_add_google("Schoolbell", "bell")
```

Custom Themes: Fonts

With default font:

```
theme_plot +
  labs(title = "Relationship Between Fertility and Life Expectancy") +
  theme(plot.title=element_text(size=16))
```

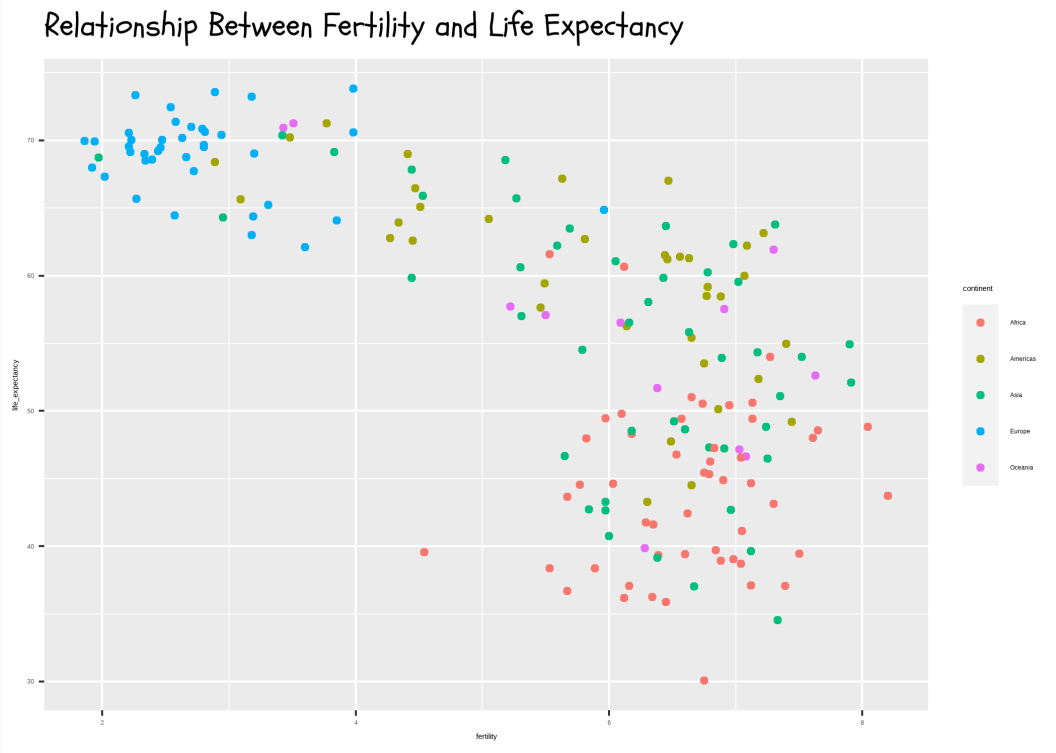


Custom Themes: Fonts

With custom font:

```
showtext_auto()

theme_plot +
  labs(title = "Relationship Between Fertility and Life Expectancy") +
  theme(plot.title=element_text(size=48, family = "bell"))
```



Extending ggplot2

Animations with *gganimate*

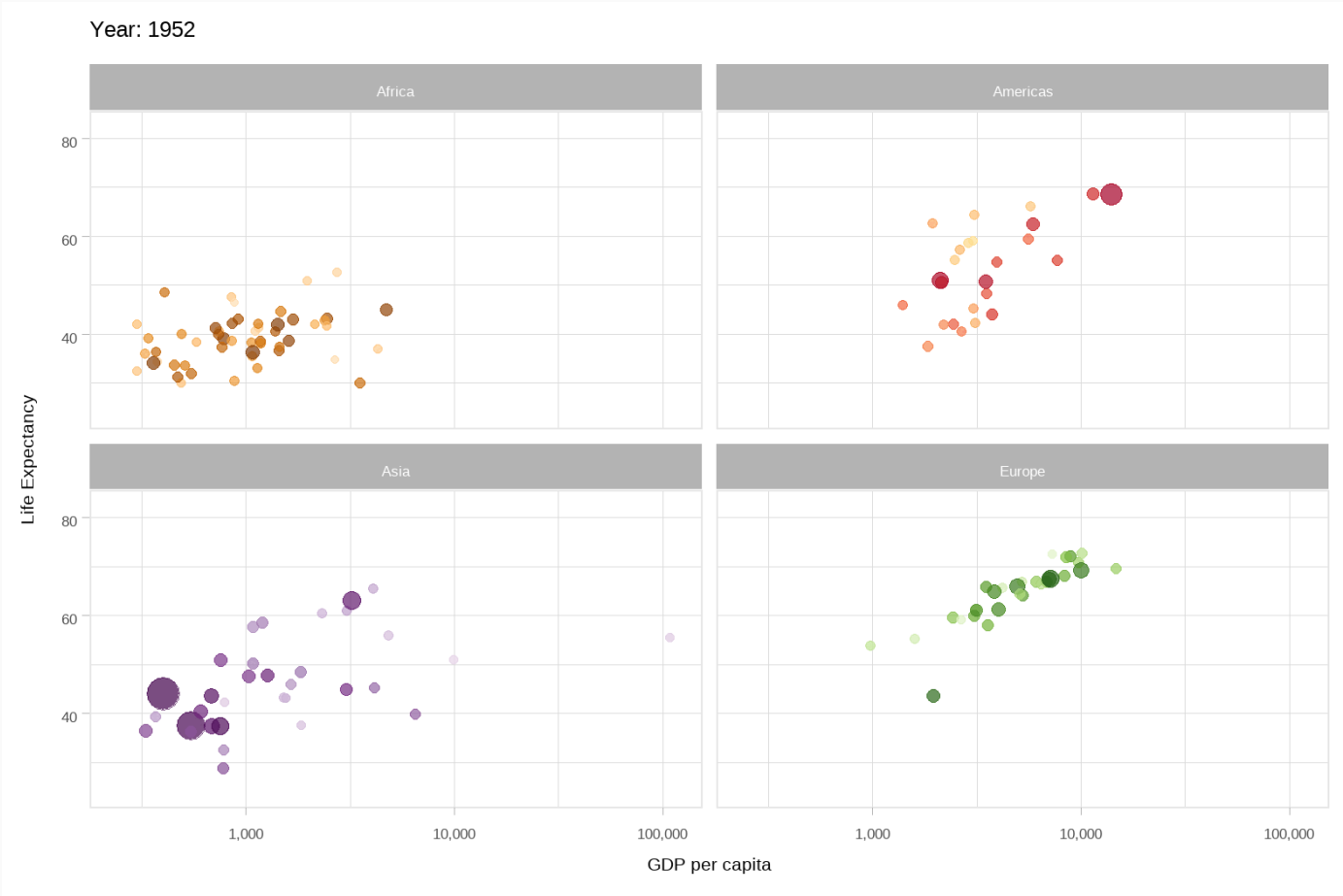


Image and code from **"Plotting in R"** by Edward Rubin, used with permission, and excluded from the overall CC license.

Marginal Distributions

Add marginal distributions to plot axes with **ggExtra**

```
# add marginal distribution to X axis
ggExtra::ggMarginal(theme_plot, margins = "x",
           type = "histogram", size = 2, fill = "aquamarine")
```

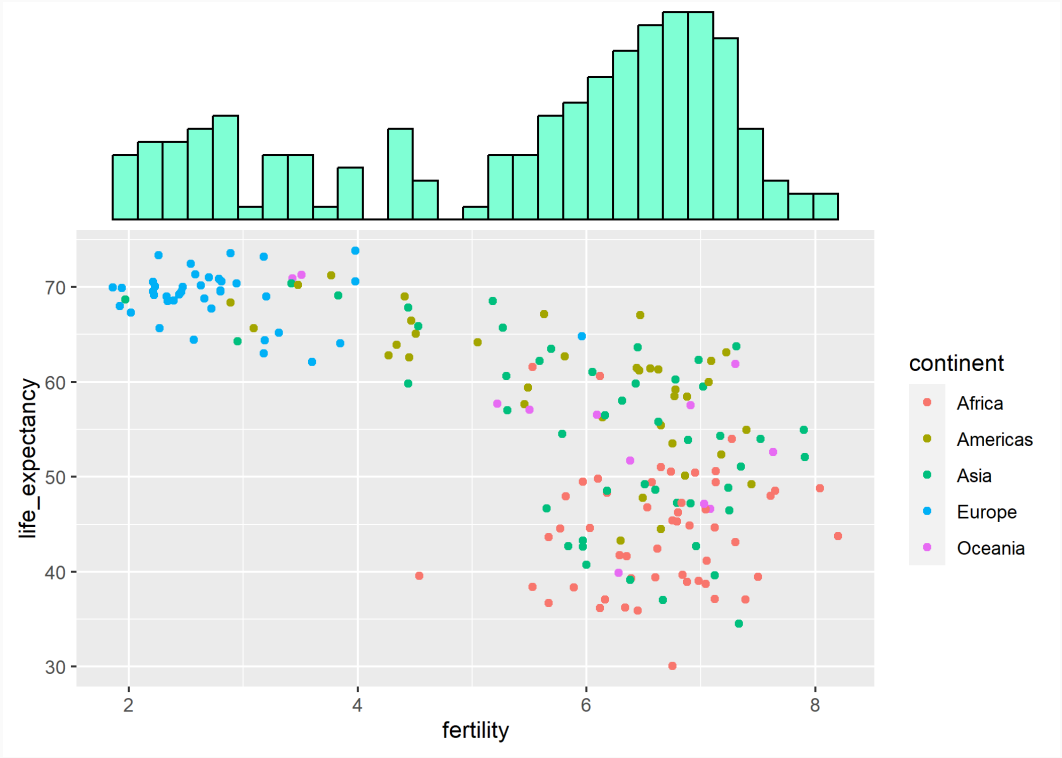


Table of Contents

Last Lecture:

This Lecture: