

Themes from scratch

INTRODUCTION TO DATA VISUALIZATION WITH GGPLOT2



Rick Scavetta

Founder, Scavetta Academy

The themes layer

- All non-data ink
- Visual elements not part of the data

The themes layer

- All non-data ink
- Visual elements not part of the data

Three types

type
text
line
rectangle

The themes layer

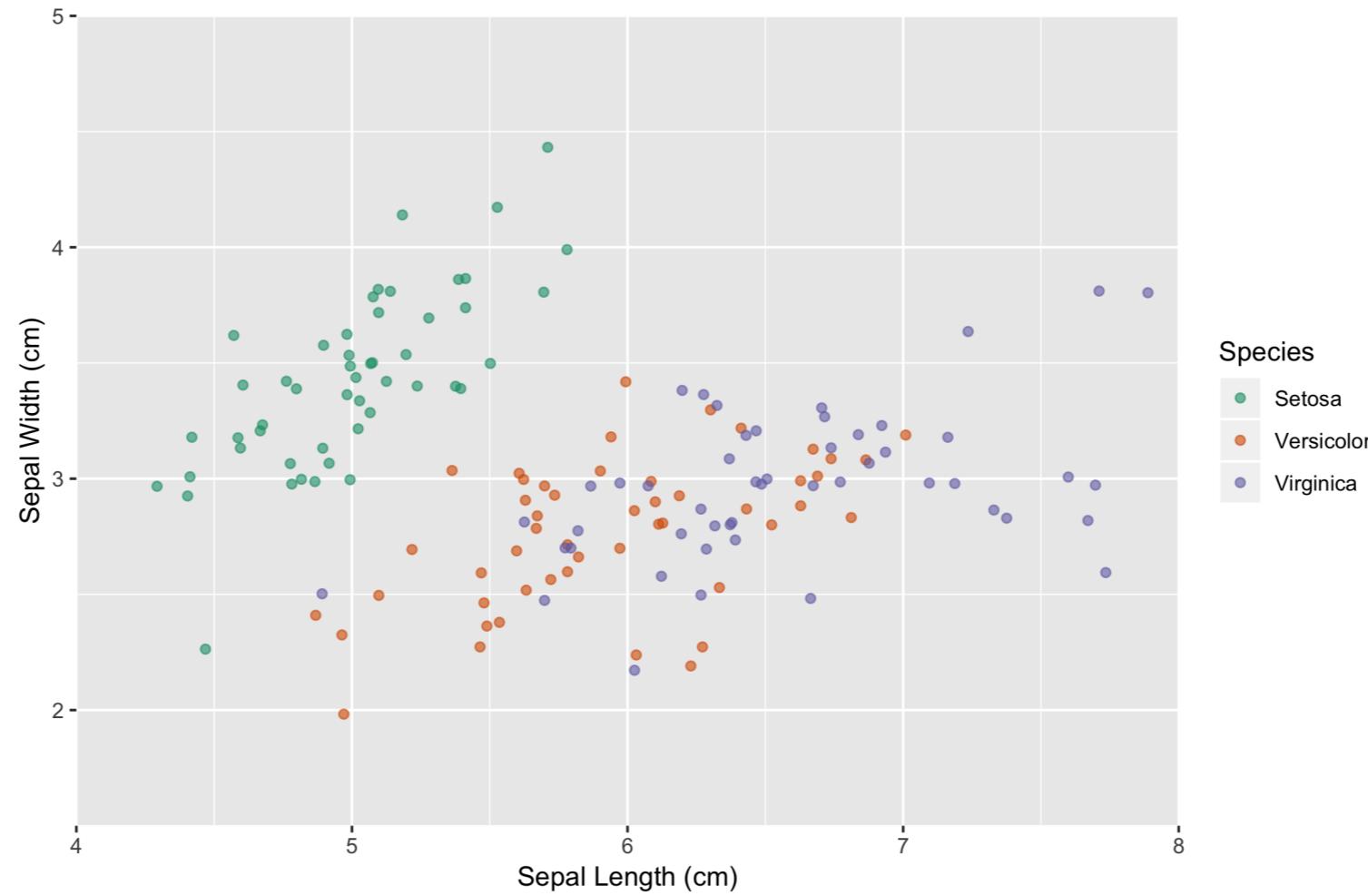
- All non-data ink
- Visual elements not part of the data

Three types

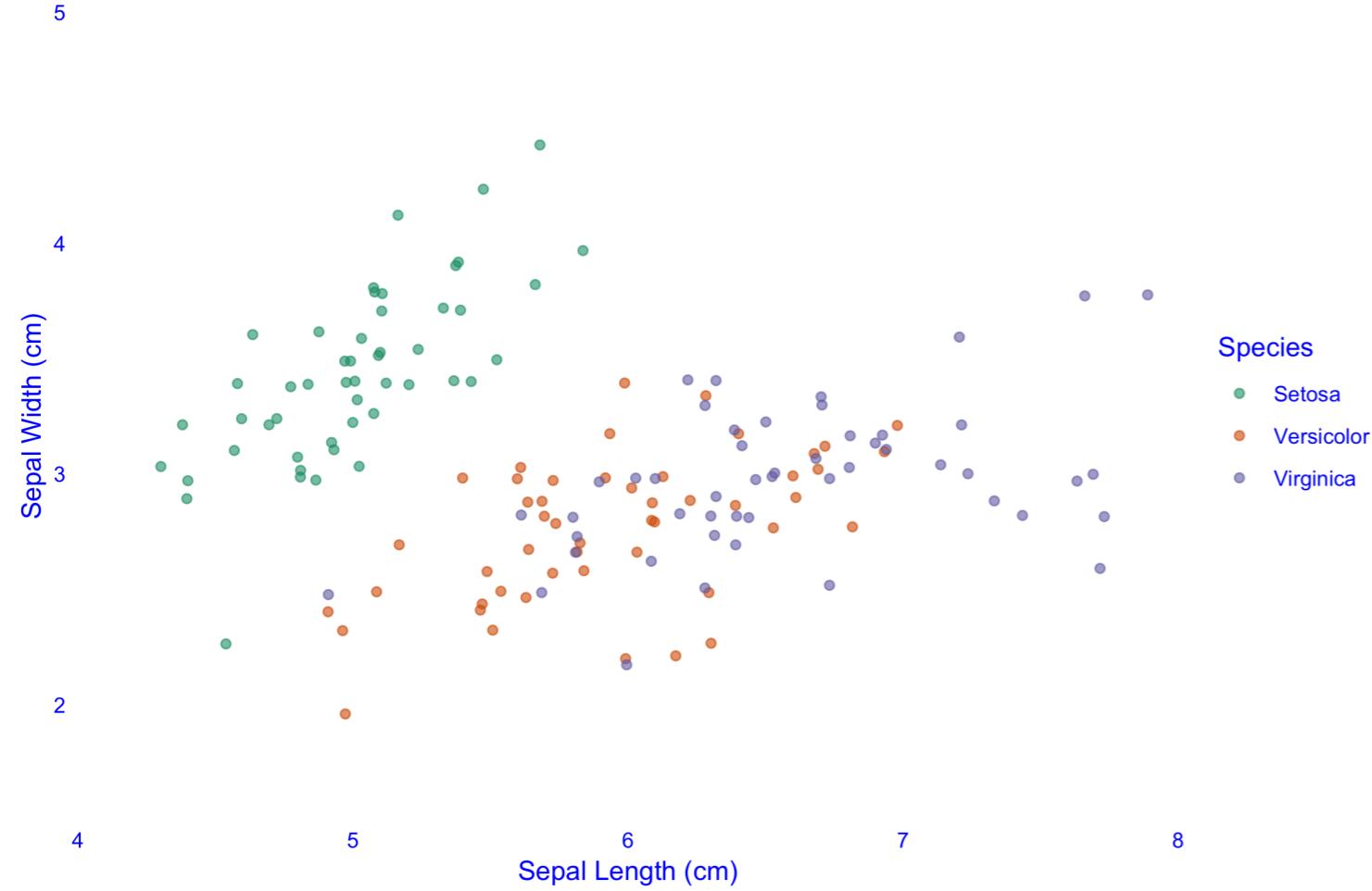
type	modified using
text	<code>element_text()</code>
line	<code>element_line()</code>
rectangle	<code>element_rect()</code>

A starting plot...

```
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +  
  geom_jitter(alpha = 0.6)
```

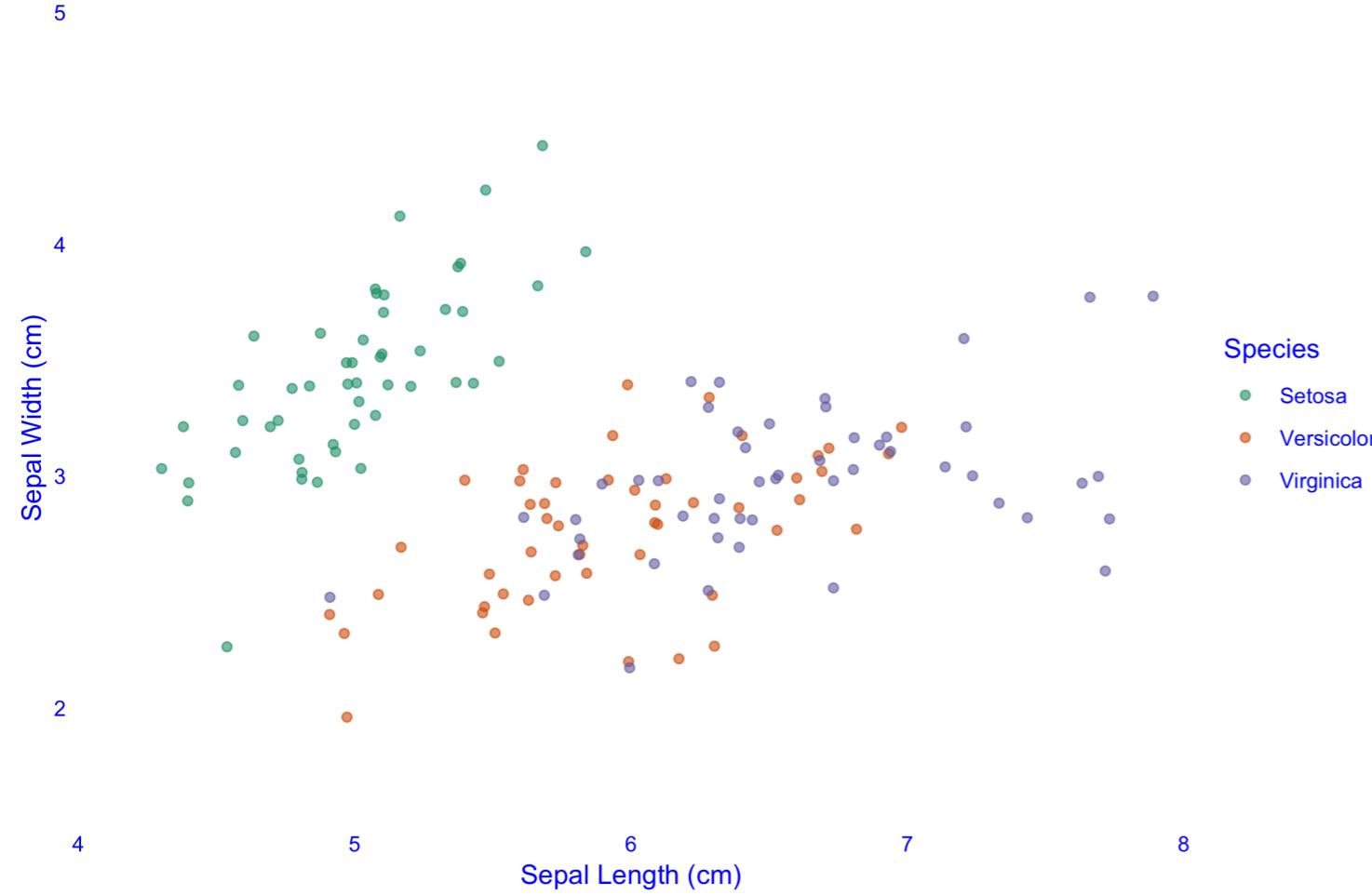


The text elements



```
text
axis.title
axis.title.x
axis.title.x.top
axis.title.x.bottom
axis.title.y
axis.title.y.left
axis.title.y.right
title
legend.title
plot.title
plot.subtitle
plot.caption
plot.tag
axis.text
axis.text.x
axis.text.x.top
axis.text.x.bottom
axis.text.y
axis.text.y.left
axis.text.y.right
legend.text
strip.text
strip.text.x
strip.text.y
```

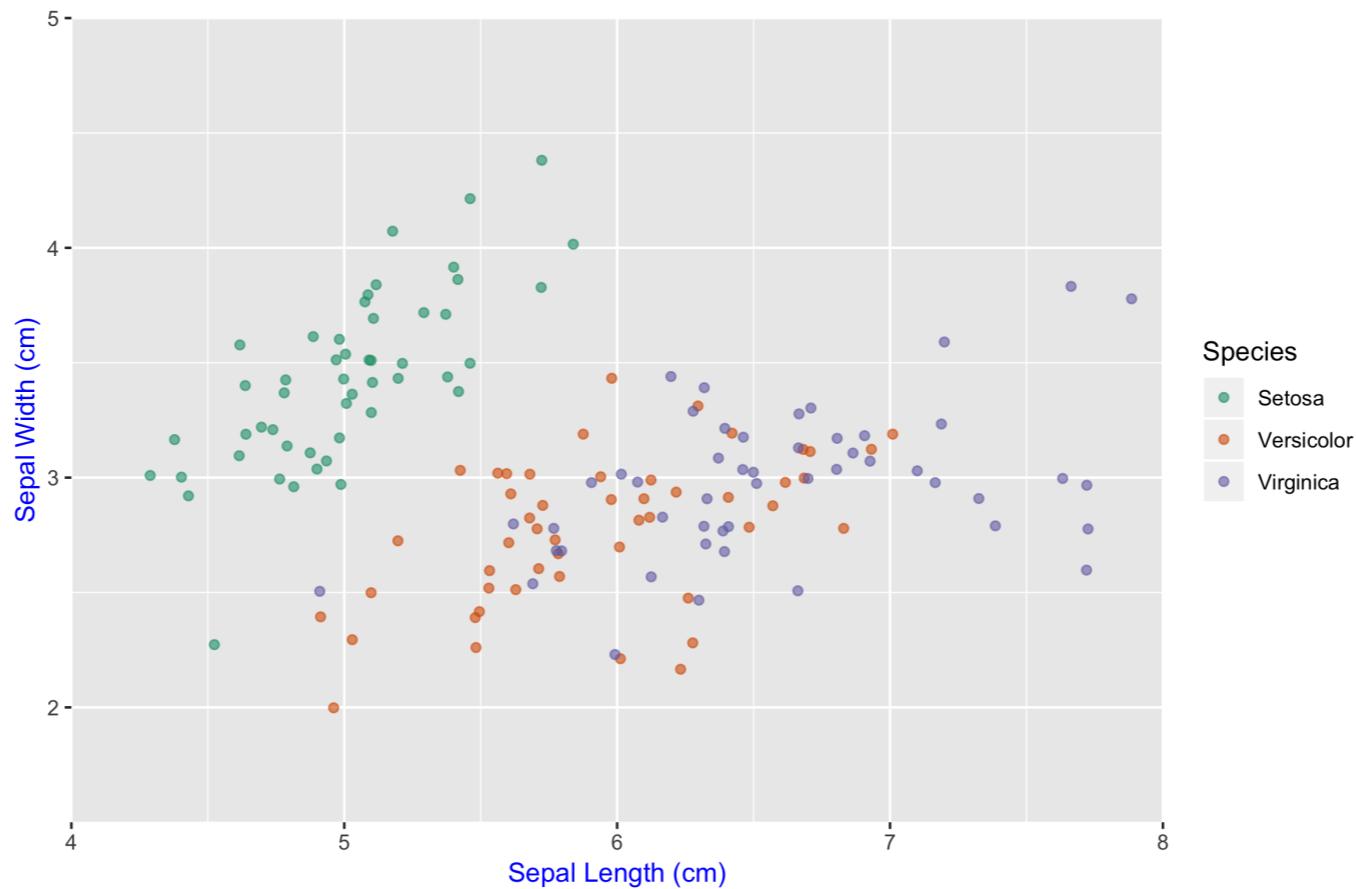
The text elements



```
theme(  
text,  
axis.title,  
axis.title.x,  
axis.title.x.top,  
axis.title.x.bottom,  
axis.title.y,  
axis.title.y.left,  
axis.title.y.right,  
title,  
legend.title,  
plot.title,  
plot.subtitle,  
plot.caption,  
plot.tag,  
axis.text,  
axis.text.x,  
axis.text.x.top,  
axis.text.x.bottom,  
axis.text.y,  
axis.text.y.left,  
axis.text.y.right,  
legend.text,  
strip.text,  
strip.text.x,  
strip.text.y)
```

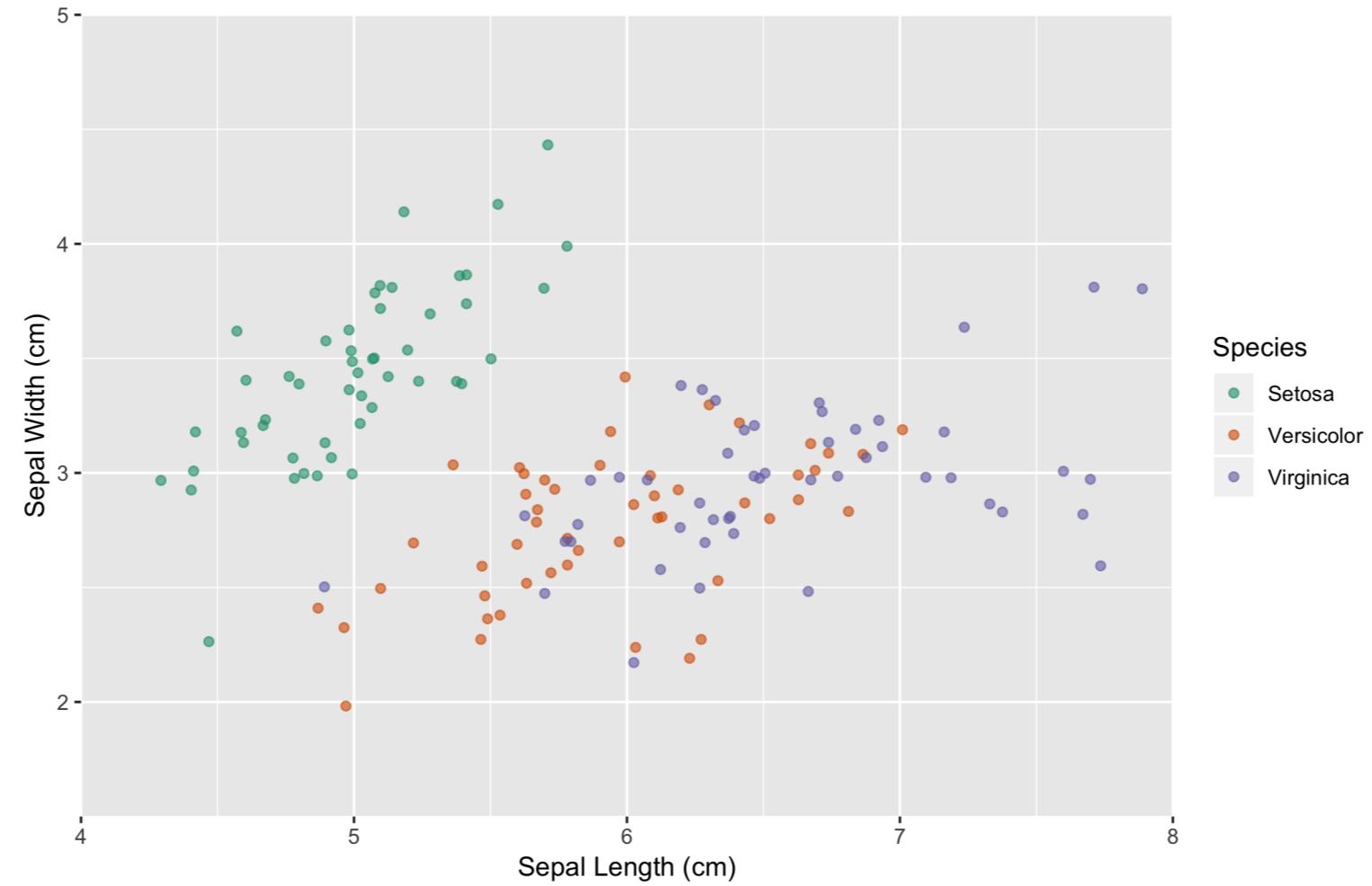
Adjusting theme elements

```
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +  
  geom_jitter(alpha = 0.6) +  
  theme(axis.title = element_text(color = "blue"))
```

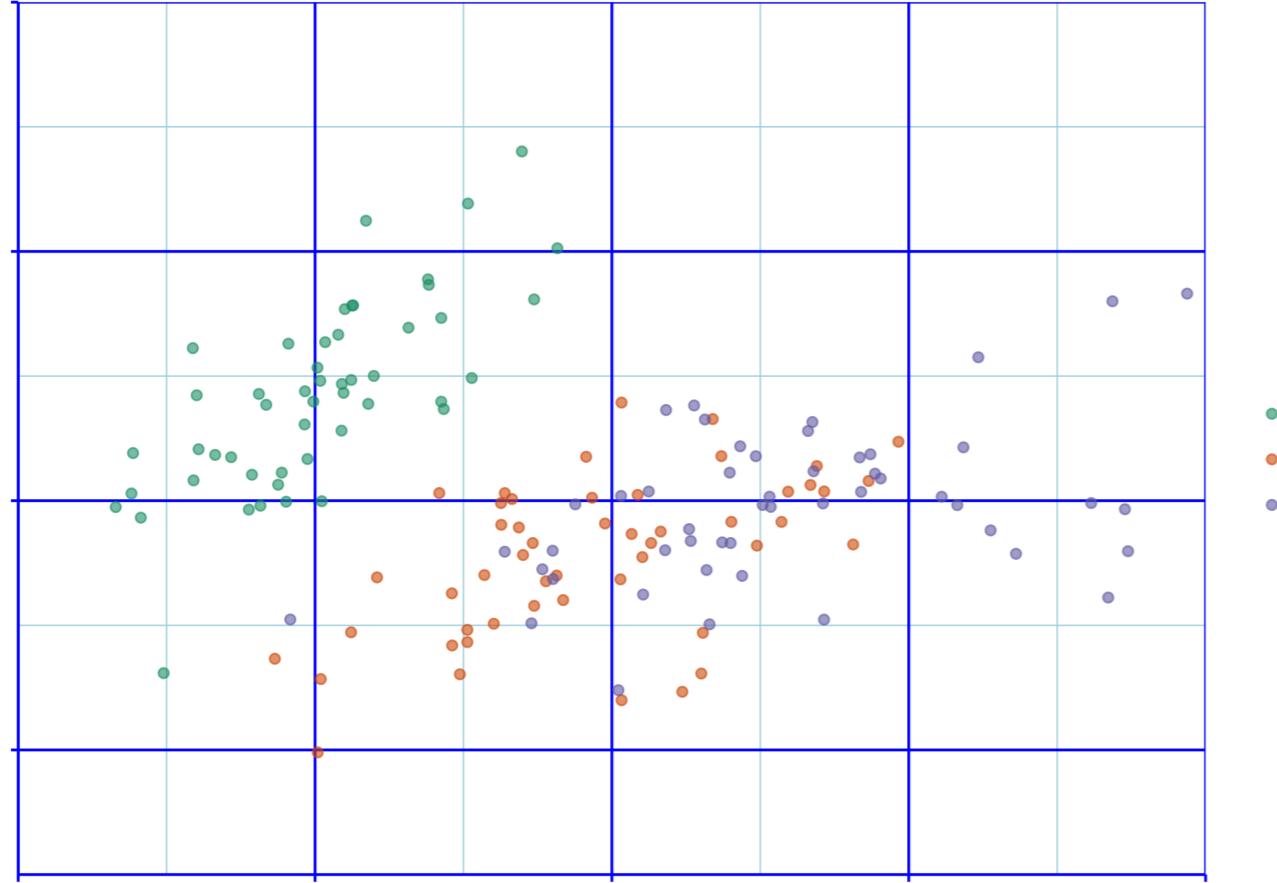


A starting plot...

```
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +  
  geom_jitter(alpha = 0.6)
```



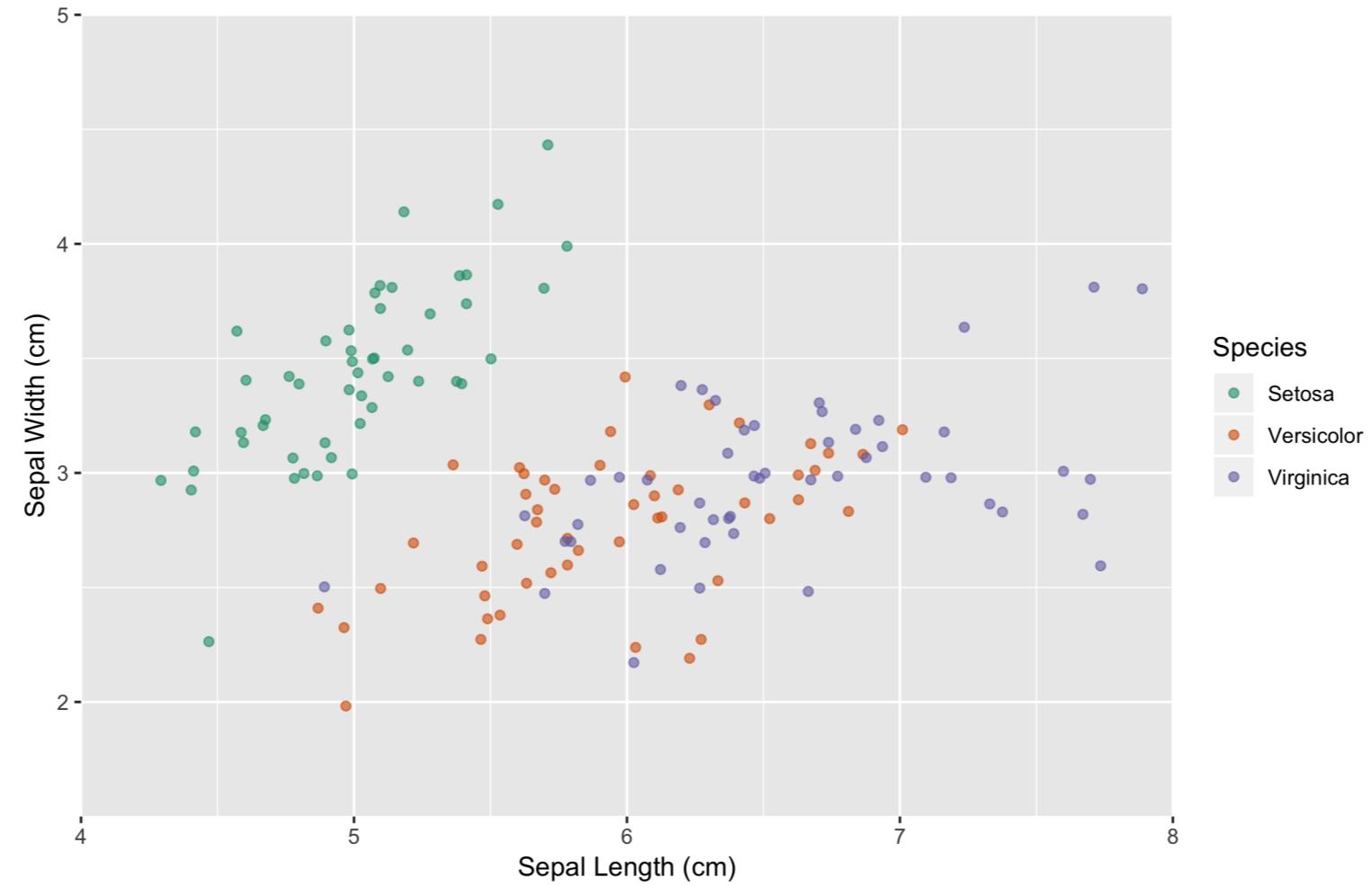
Line elements



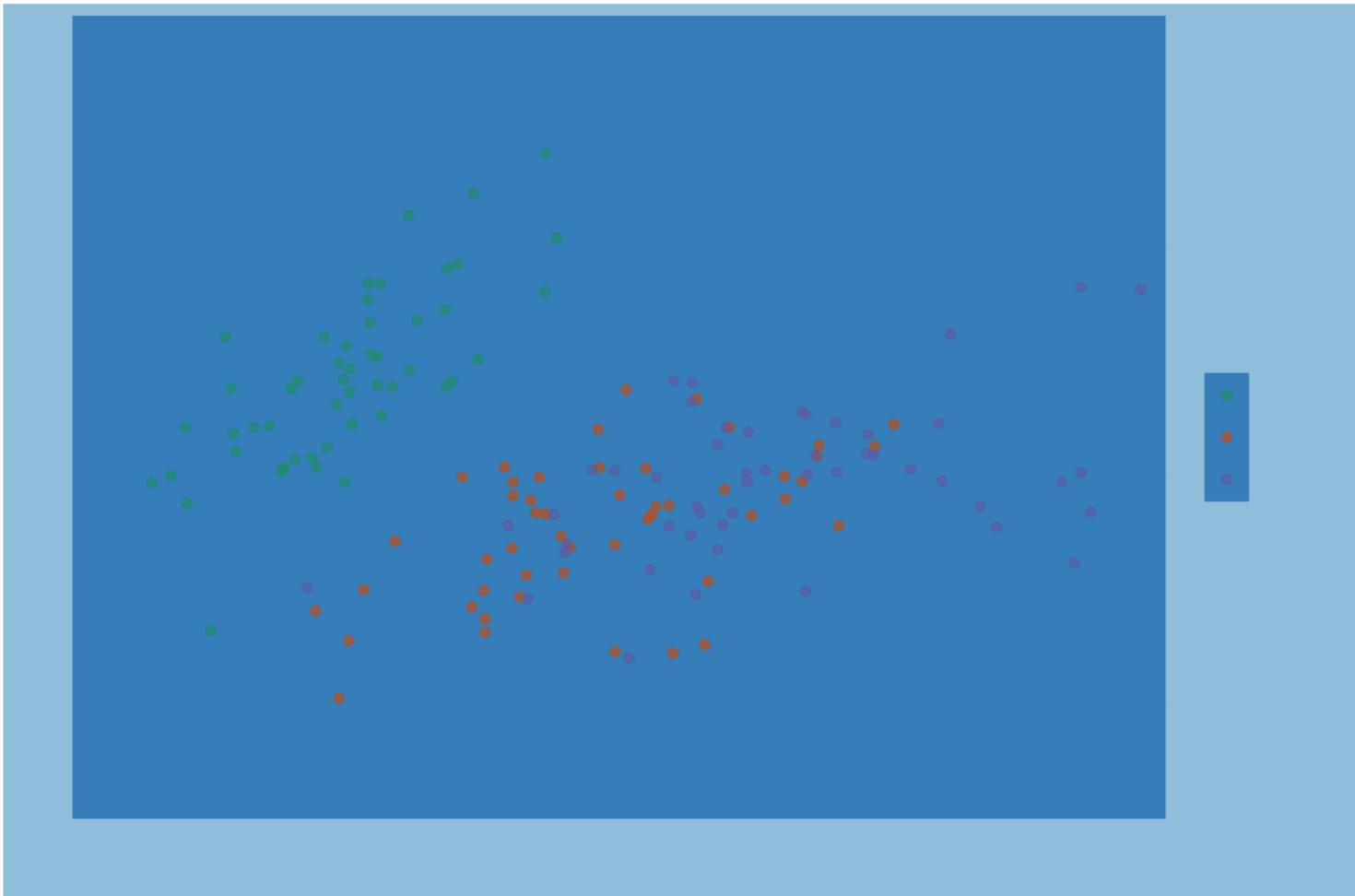
```
theme(  
  line,  
  axis.ticks,  
  axis.ticks.x,  
  axis.ticks.x.top,  
  axis.ticks.x.bottom,  
  axis.ticks.y,  
  axis.ticks.y.left,  
  axis.ticks.y.right,  
  axis.line,  
  axis.line.x,  
  axis.line.x.top,  
  axis.line.x.bottom,  
  axis.line.y,  
  axis.line.y.left,  
  axis.line.y.right,  
  panel.grid,  
  panel.grid.major,  
  panel.grid.major.x,  
  panel.grid.major.y,  
  panel.grid.minor,  
  panel.grid.minor.x,  
  panel.grid.minor.y)
```

A starting plot...

```
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +  
  geom_jitter(alpha = 0.6)
```



Rect elements



```
theme(  
  rect,  
  legend.background,  
  legend.key,  
  legend.box.background,  
  panel.background,  
  panel.border,  
  plot.background,  
  strip.background,  
  strip.background.x,  
  strip.background.y)
```

Hierarchical naming reflects inheritance rules

e.g. Text

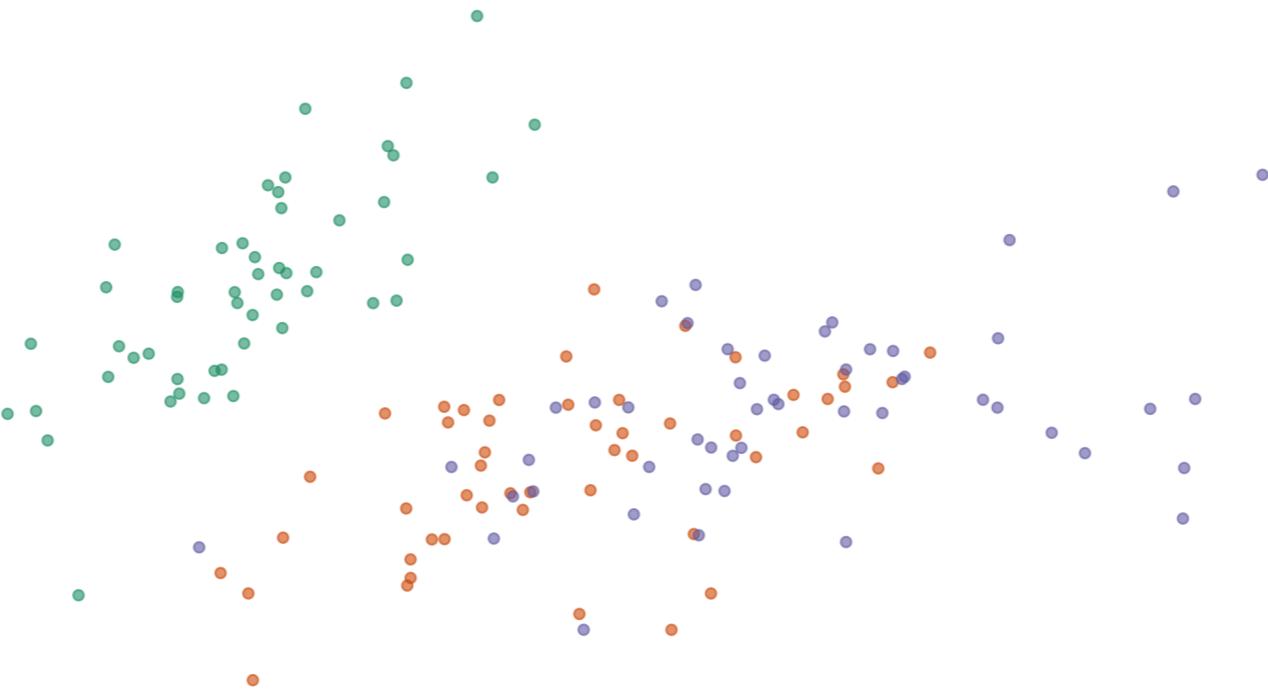
```
text  
  axis.title  
    axis.title.x  
      axis.title.x.top  
      axis.title.x.bottom  
    axis.title.y  
      axis.title.y.left  
      axis.title.y.right
```

e.g. Lines

```
line  
  axis.ticks  
    axis.ticks.x  
      axis.ticks.x.top  
      axis.ticks.x.bottom  
    axis.ticks.y  
      axis.ticks.y.left,  
      axis.ticks.y.right  
  axis.line  
    axis.line.x  
      axis.line.x.top  
      axis.line.x.bottom  
    axis.line.y  
      axis.line.y.left  
      axis.line.y.right
```

element_blank()

```
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +  
  geom_jitter(alpha = 0.6) +  
  theme(line = element_blank(),  
        rect = element_blank(),  
        text = element_blank())
```



Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH GGPLOT2

Theme flexibility

INTRODUCTION TO DATA VISUALIZATION WITH GGPLOT2



Rick Scavetta

Founder, Scavetta Academy

Ways to use themes

1. From scratch (last video)

Ways to use themes

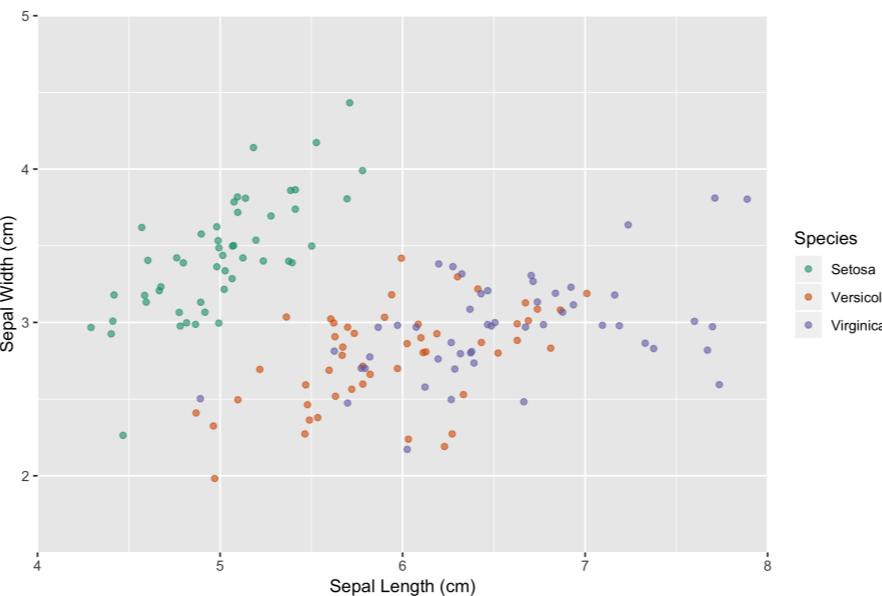
1. From scratch (last video)
2. **Theme layer object**
3. Built-in themes
 - ggplot2 or ggthemes packages
4. Built-in themes from other packages
5. Update/Set default theme

Defining theme objects

- Useful when you have many plots
- Provides consistency in style
- Apply a specific theme everywhere

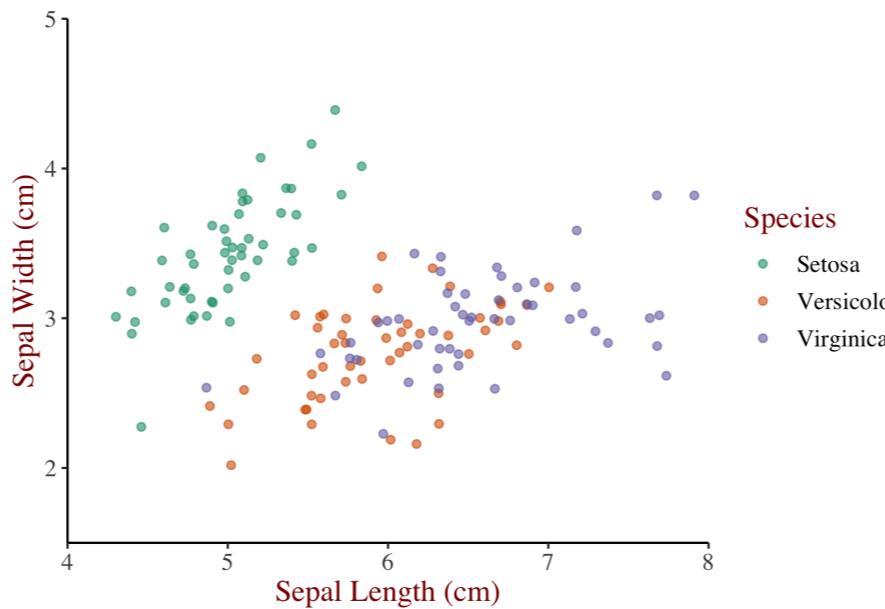
Defining theme objects

```
z <- ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +  
  geom_jitter(alpha = 0.6) +  
  scale_x_continuous("Sepal Length (cm)", limits = c(4,8), expand = c(0,0)) +  
  scale_y_continuous("Sepal Width (cm)", limits = c(1.5,5), expand = c(0,0)) +  
  scale_color_brewer("Species", palette = "Dark2", labels = c("Setosa", "Versicolor", "Virginica"))
```



Defining theme objects

```
z + theme(text = element_text(family = "serif", size = 14),  
          rect = element_rect(),  
          panel.grid = element_rect(),  
          title = element_text(color = "#8b0000"),  
          axis.line = element_line(color = "black"))
```

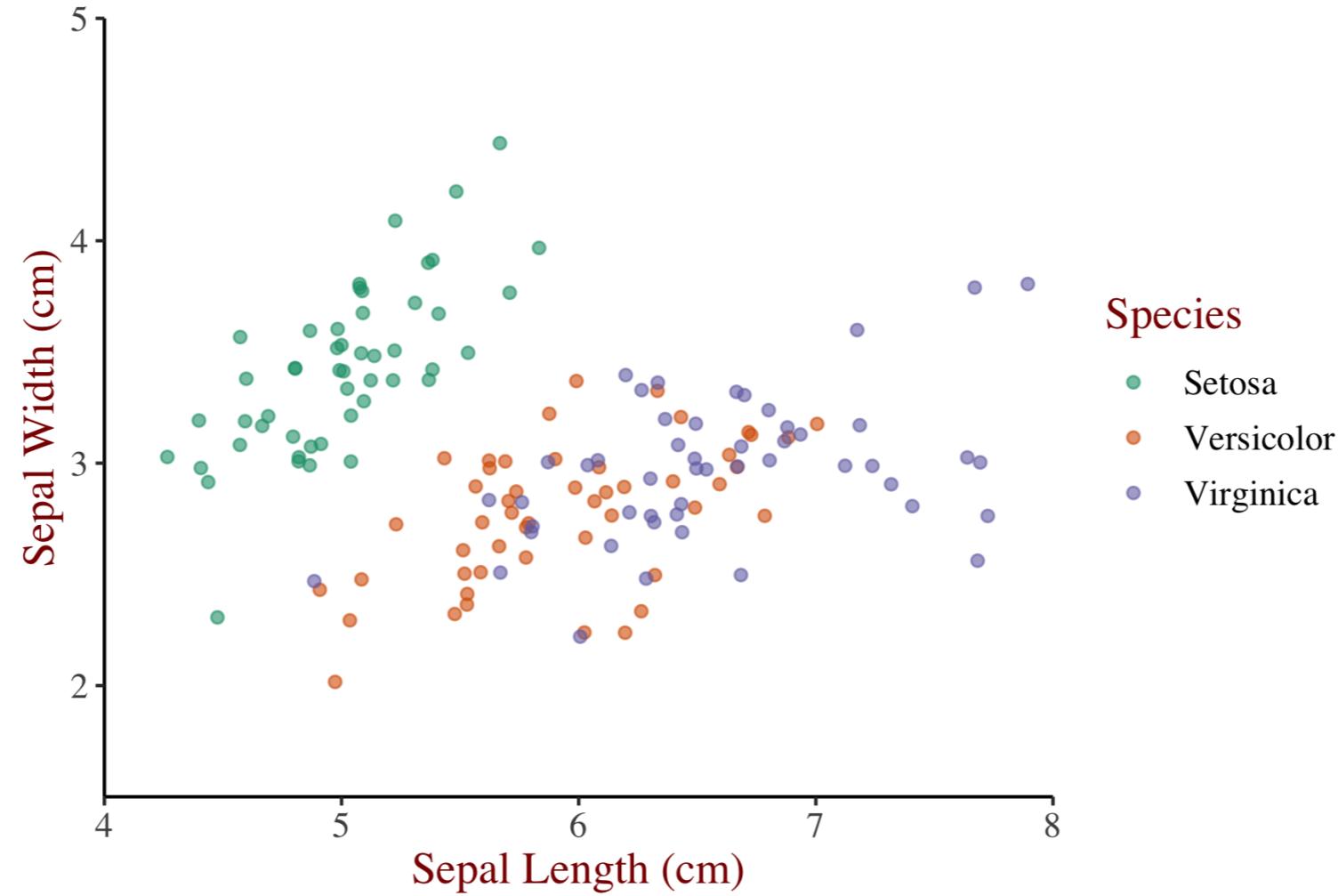


Defining theme objects

```
theme_iris <- theme(text = element_text(family = "serif", size = 14),  
                    rect = element_blank(),  
                    panel.grid = element_blank(),  
                    title = element_text(color = "#8b0000"),  
                    axis.line = element_line(color = "black"))
```

Reusing theme objects

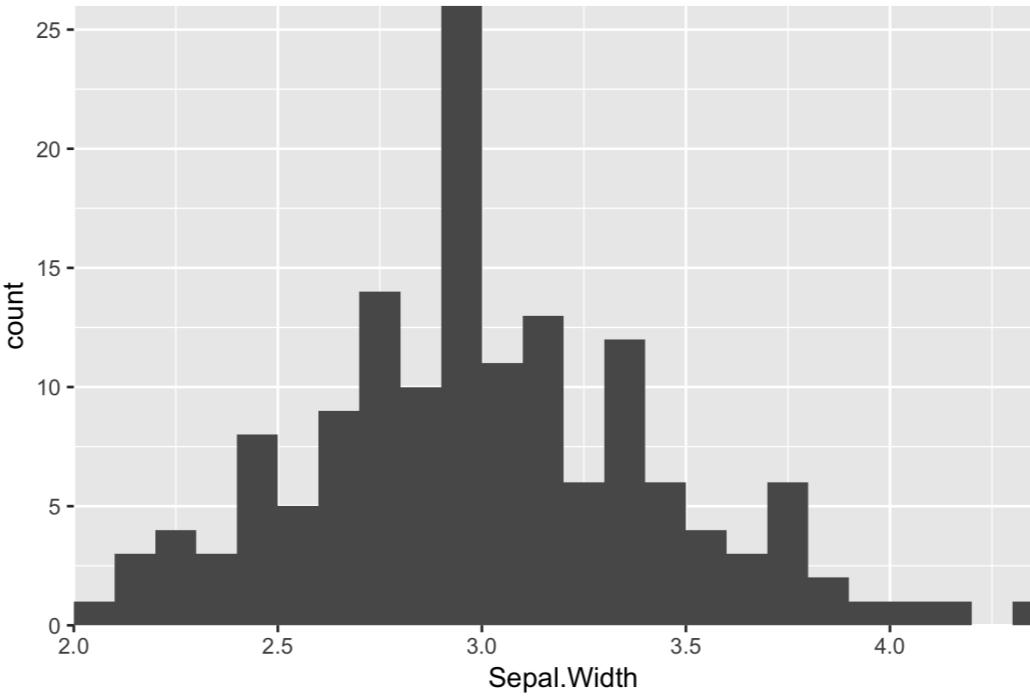
`z + theme_iris`



Reusing theme objects

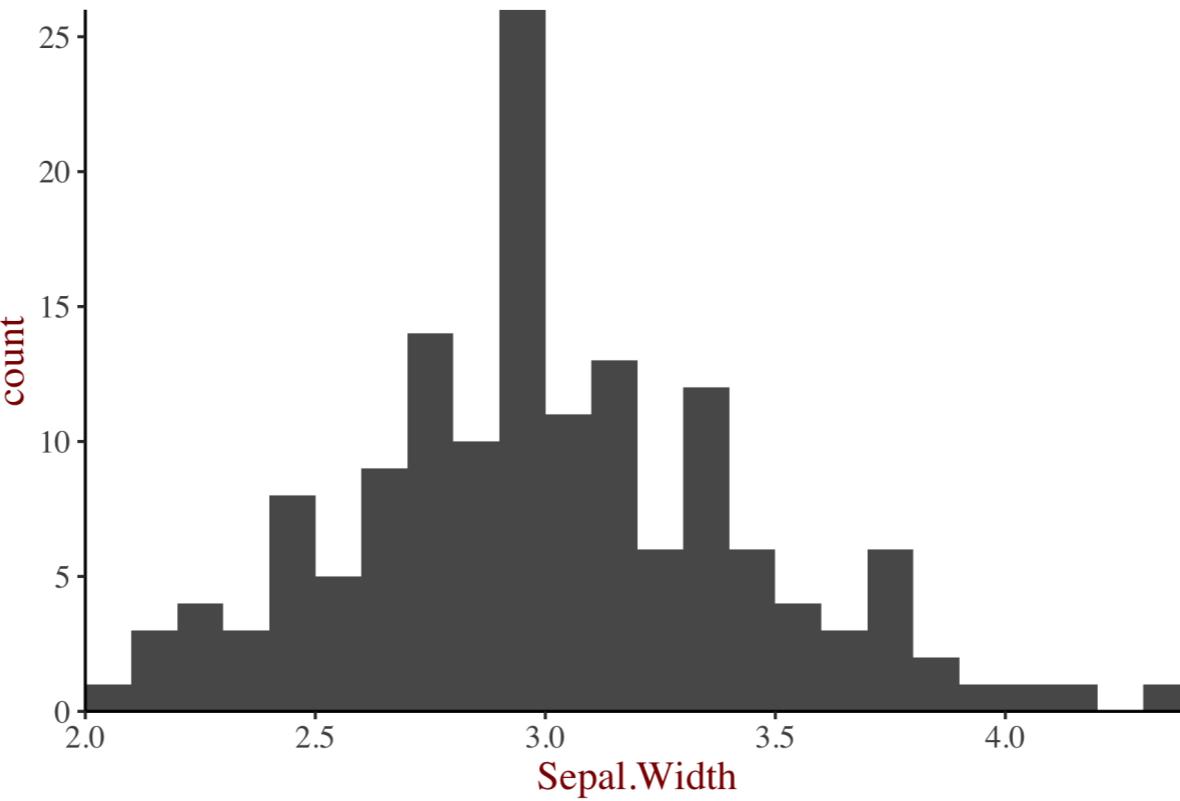
```
m <- ggplot(iris, aes(x = Sepal.Width)) +  
  geom_histogram(binwidth = 0.1,  
                 center = 0.05)
```

```
m
```



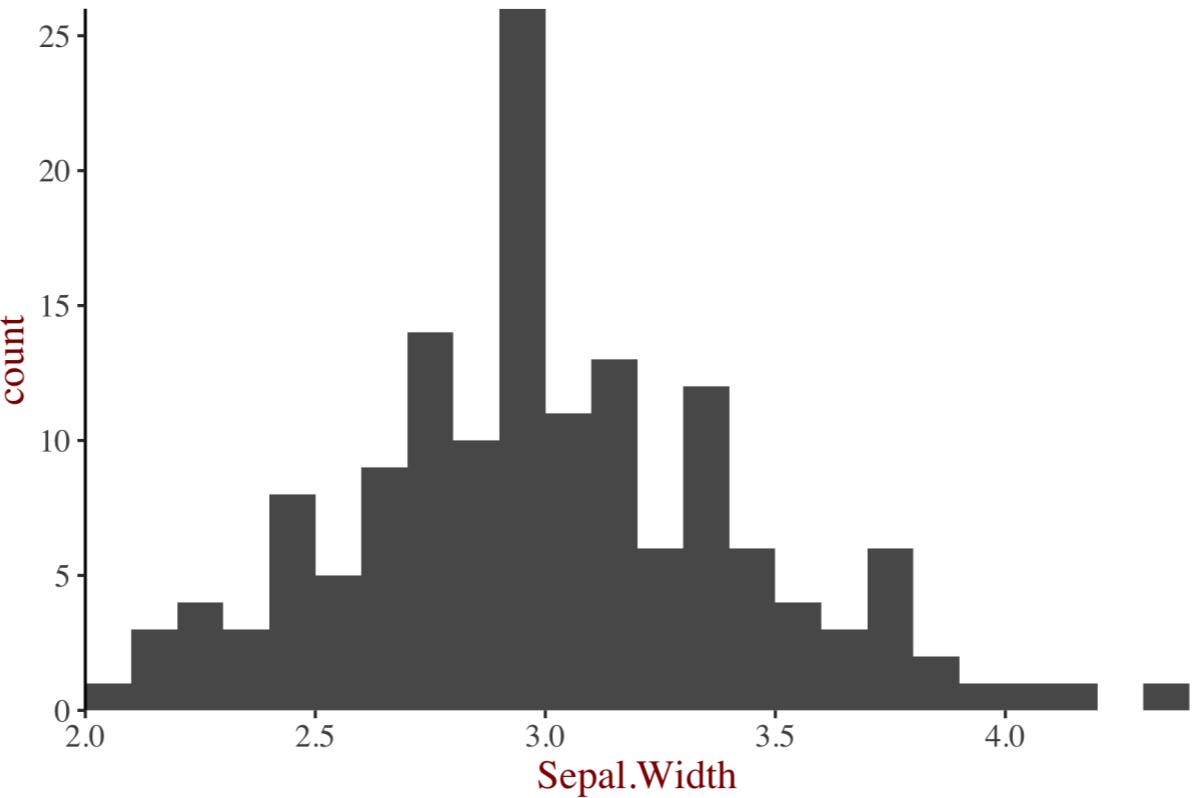
Reusing theme objects

```
m +  
  theme_iris
```



Reusing theme objects

```
m +  
  theme_iris +  
  theme(axis.line.x = element_blank())
```



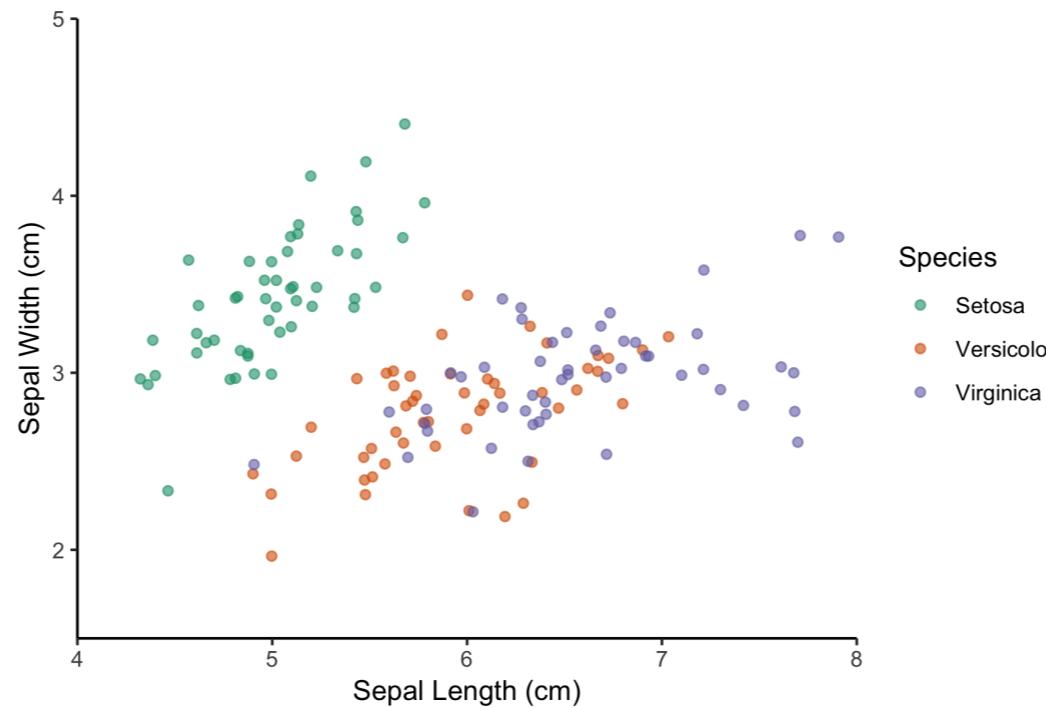
Ways to use themes

1. From scratch (last video)
2. Theme layer object
3. **Built-in themes**
 - ggplot2 or ggthemes packages
4. Built-in themes from other packages
5. Update/Set default theme

Using built-in themes

Use `theme_*`() functions to access built-in themes.

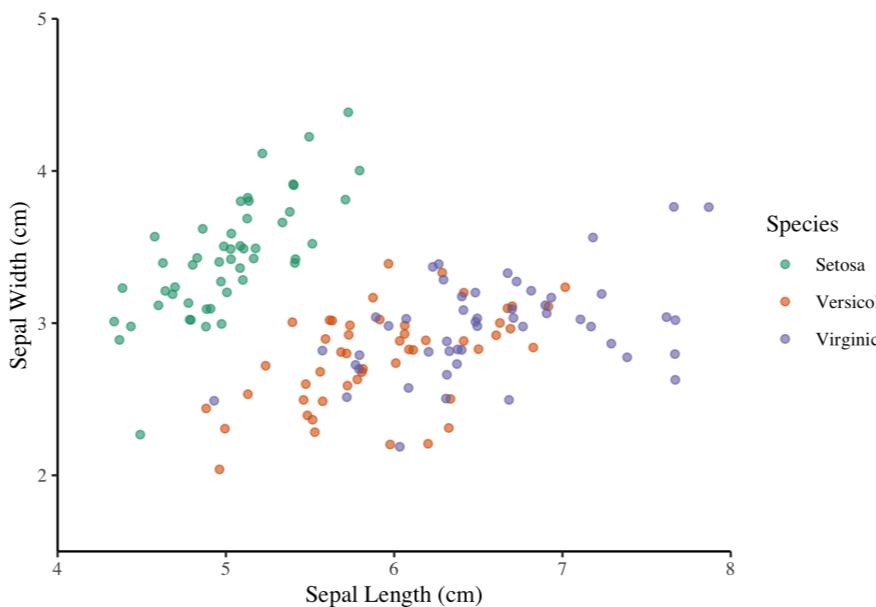
```
z +  
  theme_classic()
```



Using built-in themes

Use `theme_*`() functions to access built-in themes.

```
z +  
  theme_classic() +  
  theme(text = element_text(family = "serif"))
```



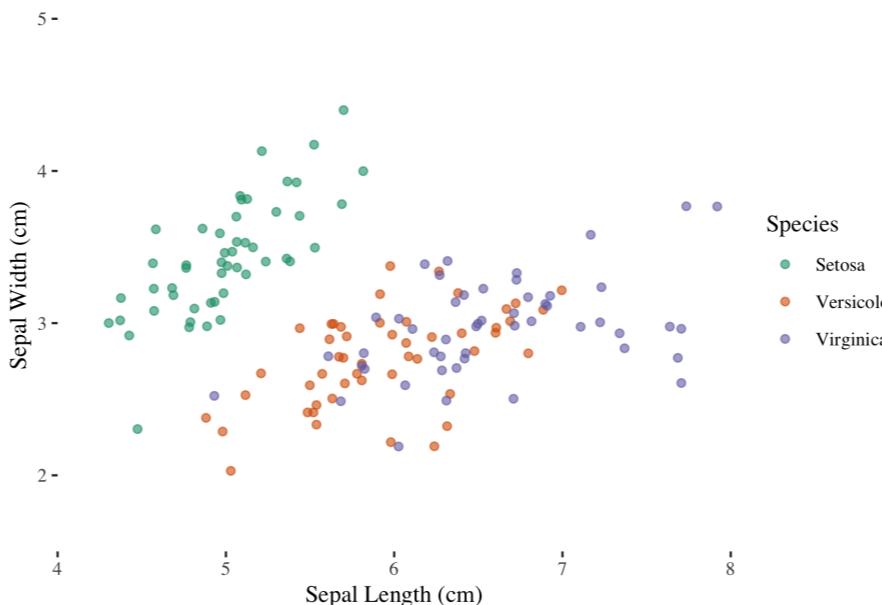
Ways to use themes

1. From scratch (last video)
2. Theme layer object
3. Built-in themes
 - ggplot2 or ggthemes packages
4. **Built-in themes from other packages**
5. Update/Set default theme

The ggthemes package

Use the `ggthemes` package for more functions.

```
library(ggthemes)  
+  
theme_tufte()
```



Ways to use themes

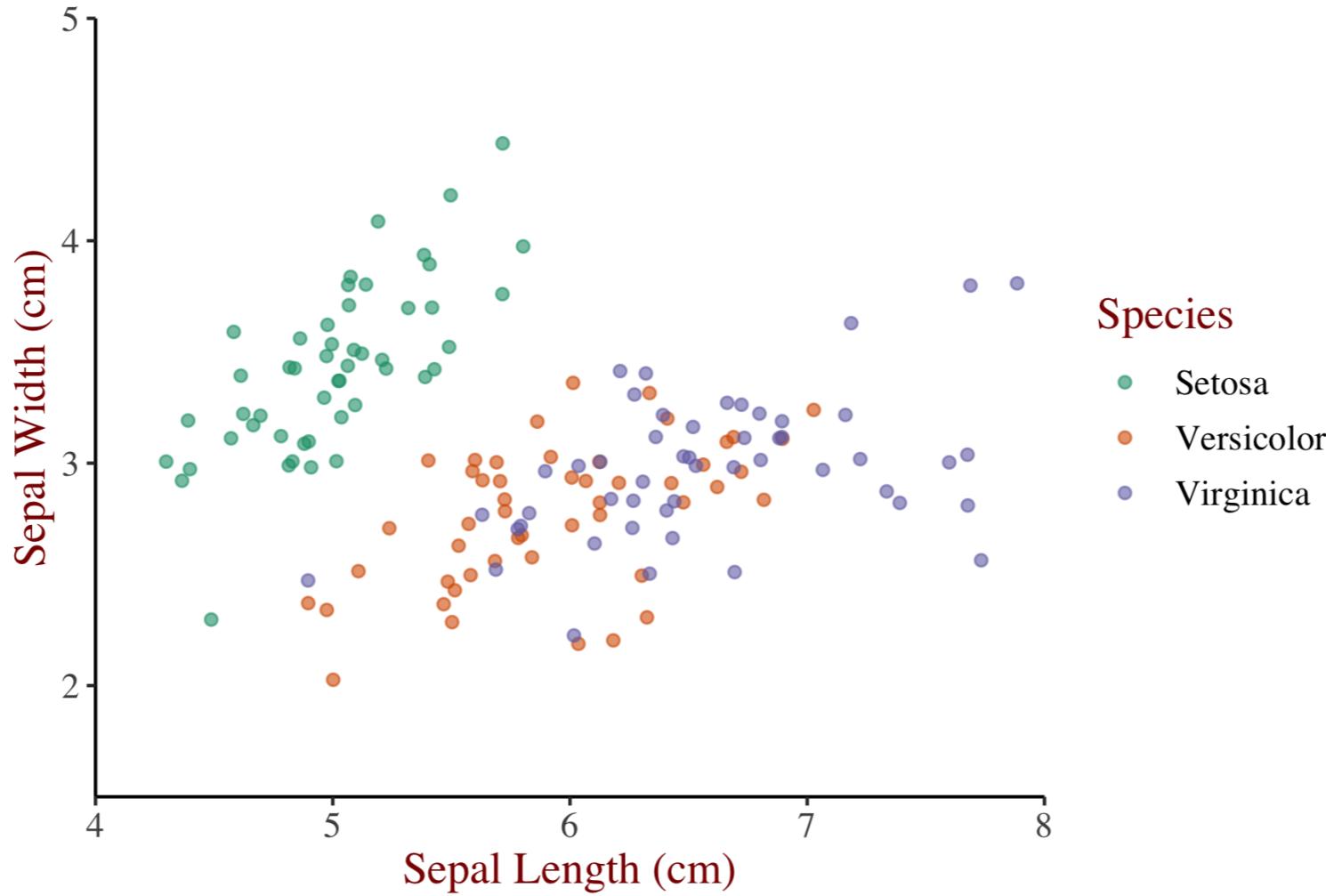
1. From scratch (last video)
2. Theme layer object
3. Built-in themes
 - ggplot2 or ggthemes packages
4. Built-in themes from other packages
5. **Update/Set default theme**

Updating themes

```
original <- theme_update(text = element_text(family = "serif", size = 14),  
                         rect = element_blank(),  
                         panel.grid = element_blank(),  
                         title = element_text(color = "#8b0000"),  
                         axis.line = element_line(color = "black"))
```

Updating themes

z

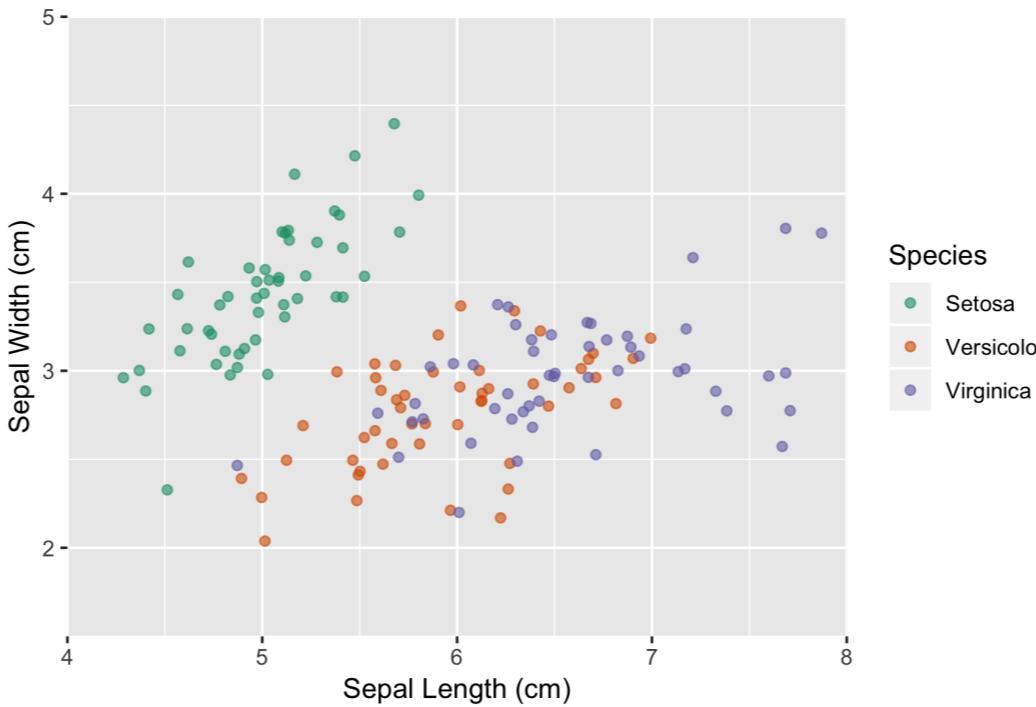


Setting themes

```
theme_set(original)
```

```
# Alternatively
```

```
# theme_set(theme_grey())
```



Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH GGPLOT2

Effective explanatory plots

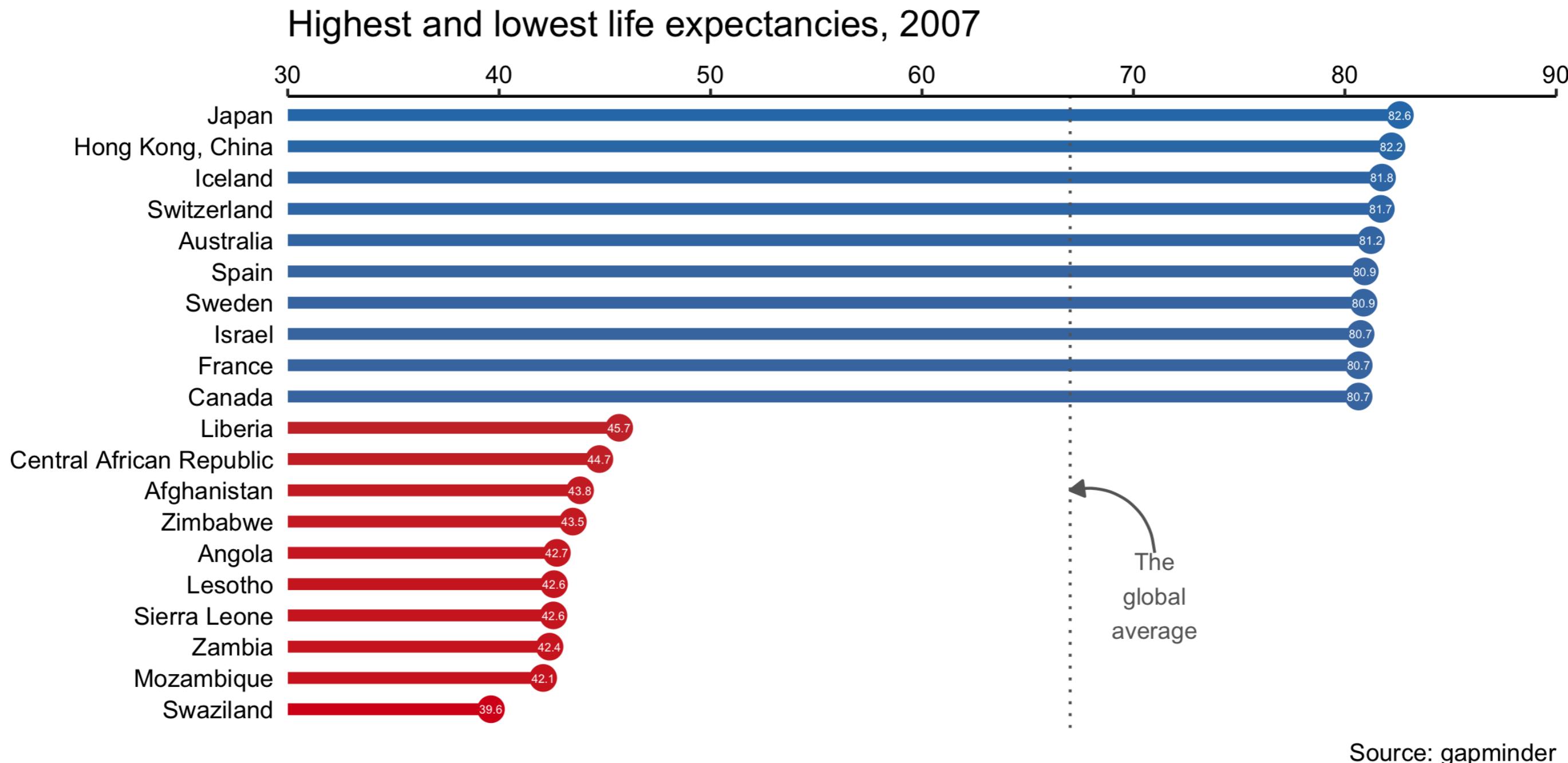
INTRODUCTION TO DATA VISUALIZATION WITH GGPLOT2



Rick Scavetta

Founder, Scavetta Academy

Our goal, an effective explanatory plot



Complete data

```
dplyr::glimpse(gm2007_full)
```

Observations: 142

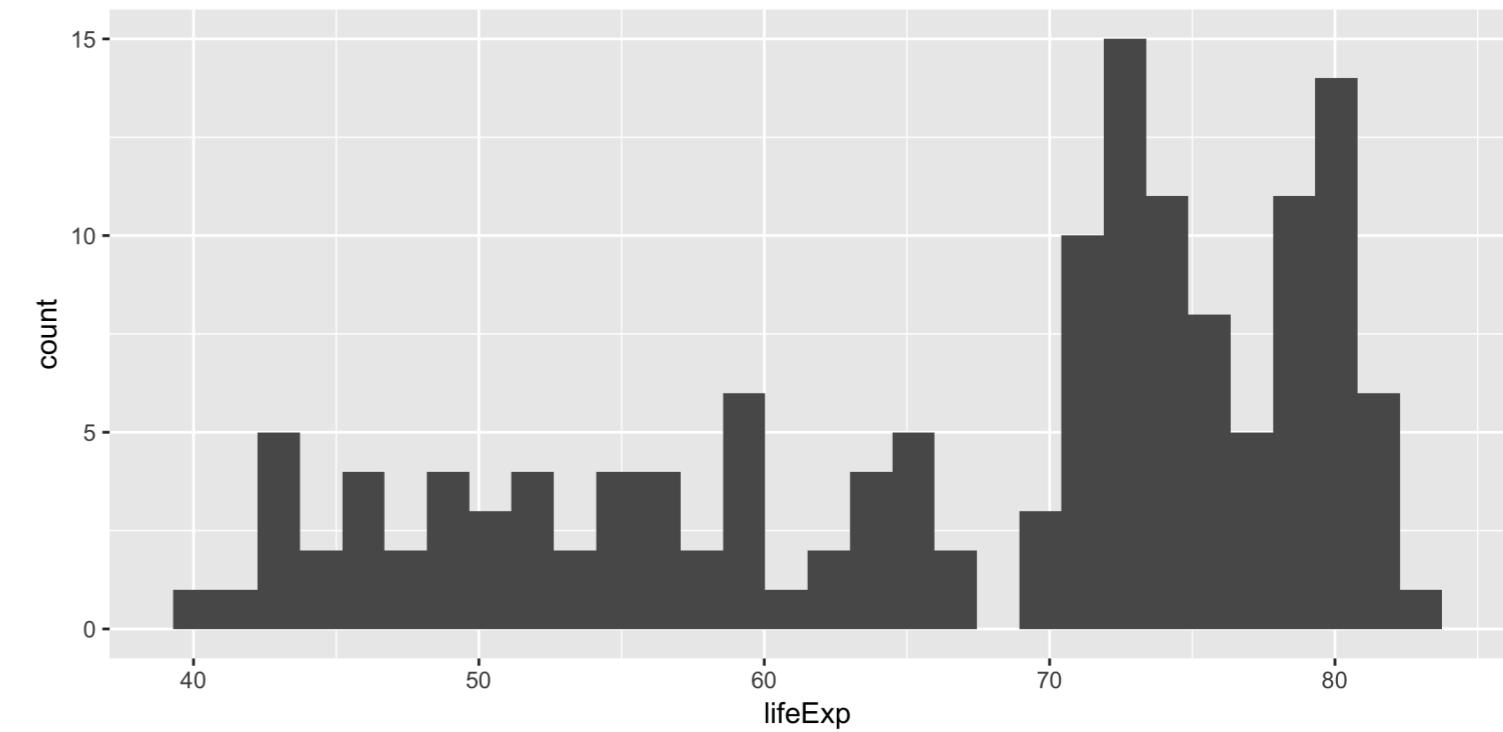
Variables: 3

```
$ country    <fct> "Afghanistan", "Albania", "Algeria", "Angola", "Argentina", "Au...  
$ lifeExp    <dbl> 43.828, 76.423, 72.301, 42.731, 75.320, 81.235, 79.829, 75.635,...  
$ continent <fct> Asia, Europe, Africa, Africa, Americas, Oceania, Europe, Asia, ...
```

¹ We would begin with our complete data set, which contains three variables for 142 countries.

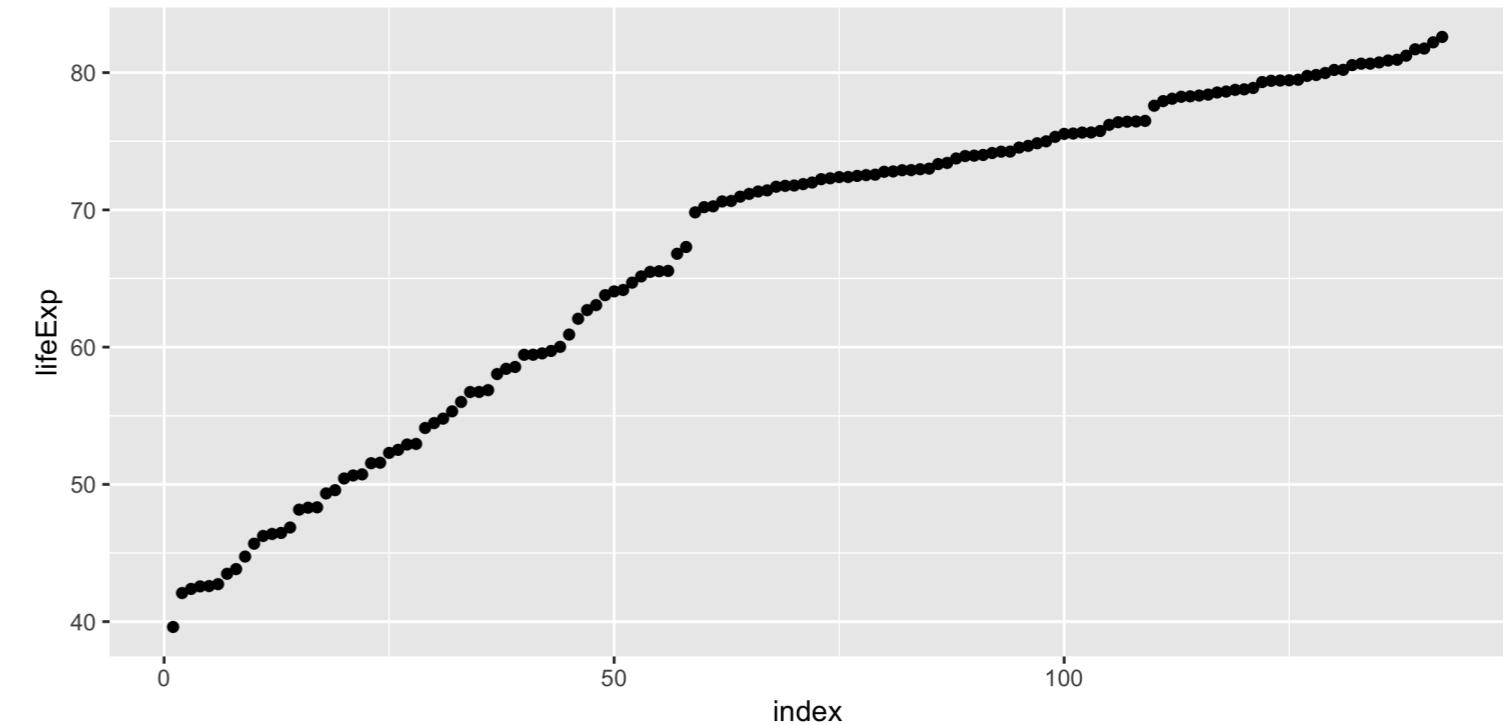
First exploratory plots - distributions

```
ggplot(gm2007_full, aes(lifeExp)) +  
  geom_histogram()
```



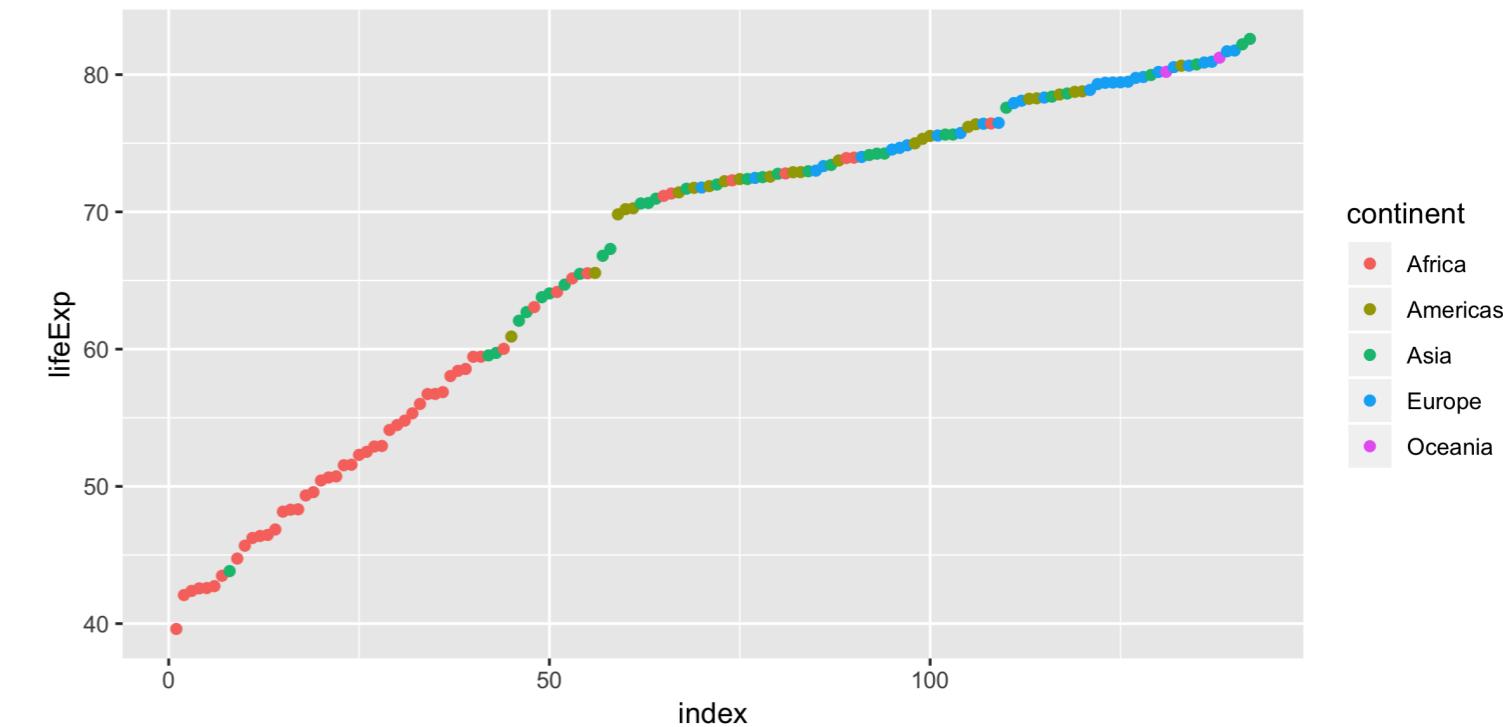
First exploratory plots - distributions

```
ggplot(gm2007_full_arranged,  
       aes(index, lifeExp)) +  
  geom_point()
```



First exploratory plots - distributions

```
ggplot(gm2007_full_arranged,  
       aes(index, lifeExp, col = continent)) +  
  geom_point()
```



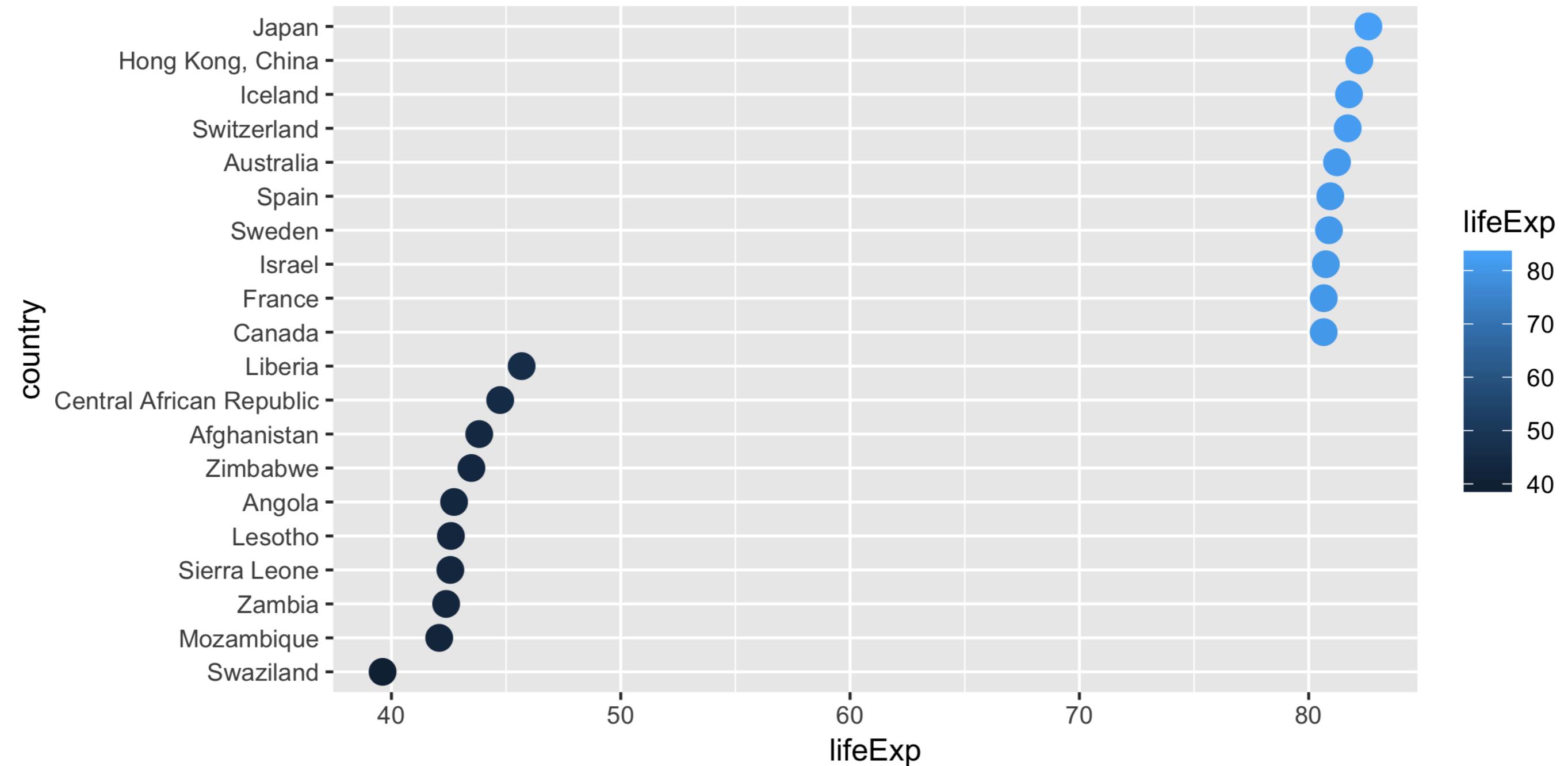
Our data

```
dplyr::glimpse(gm2007)
```

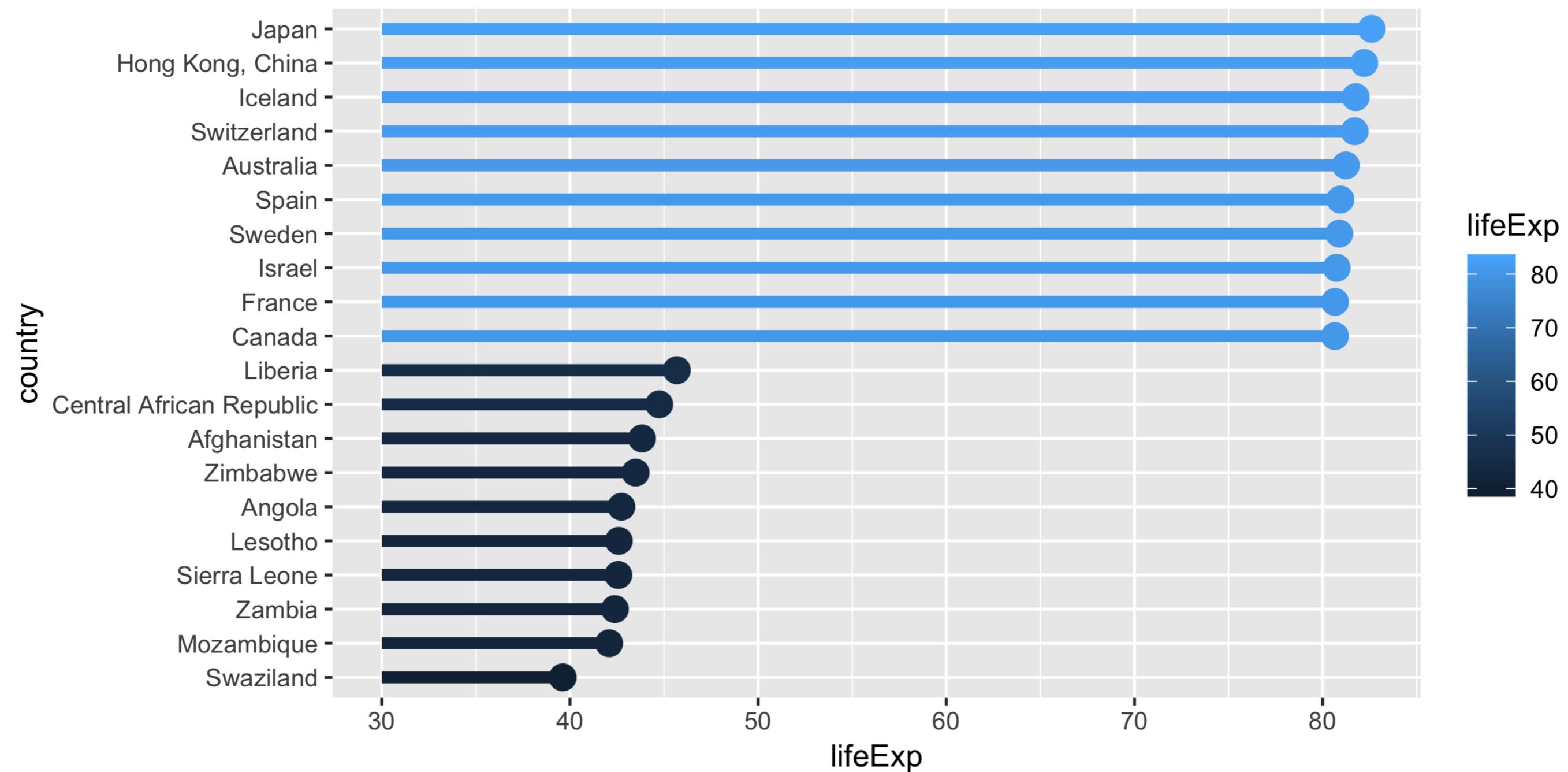
Observations: 20

Variables: 3

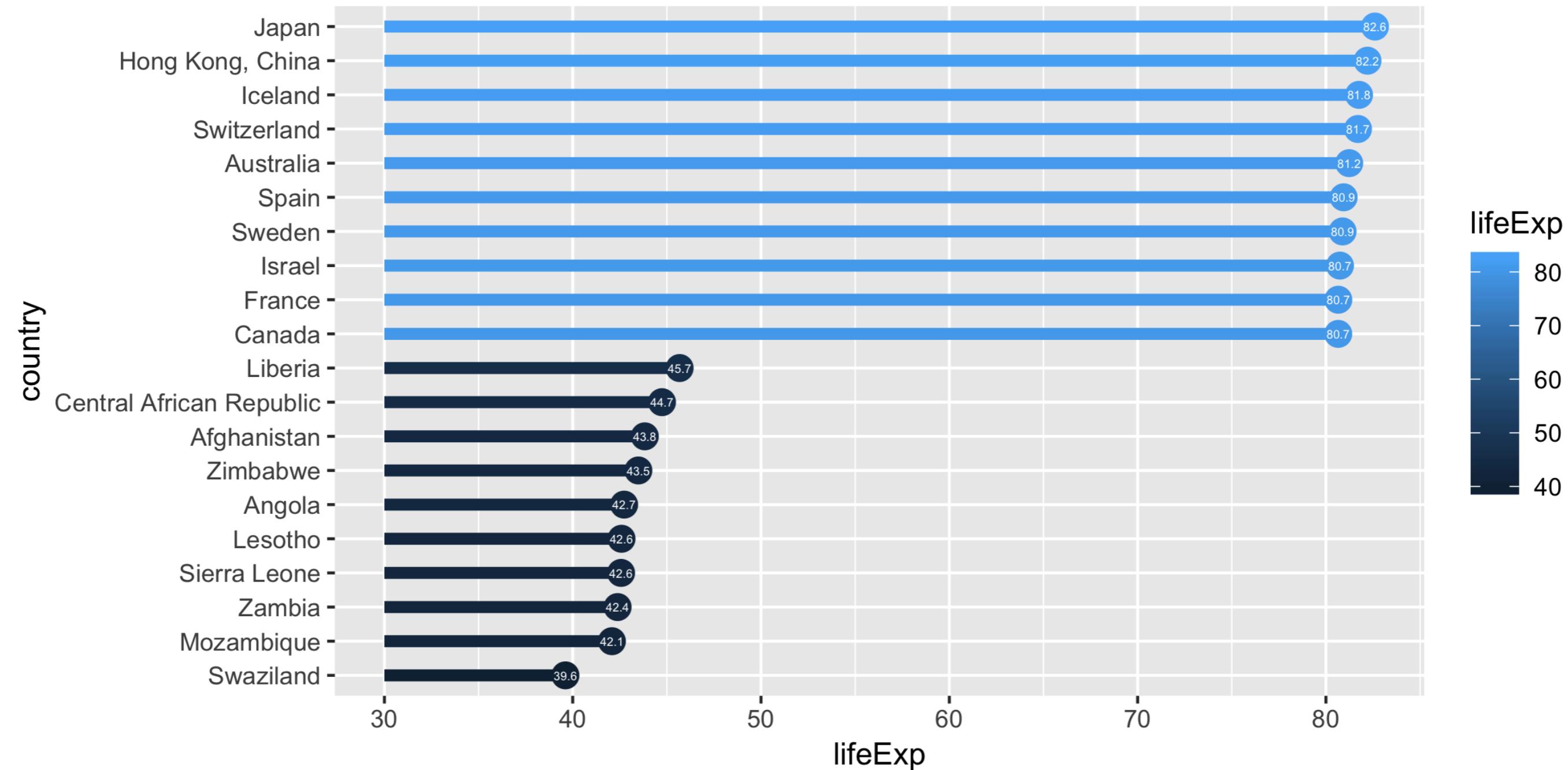
```
$ country    <fct> "Swaziland", "Mozambique", "Zambia", "Sierra Leone", "Lesotho..."  
$ lifeExp    <dbl> 39.613, 42.082, 42.384, 42.568, 42.592, 42.731, 43.487, 43.82...  
$ continent <fct> Africa, Africa, Africa, Africa, Africa, Africa, Africa, Asia,...  
>
```



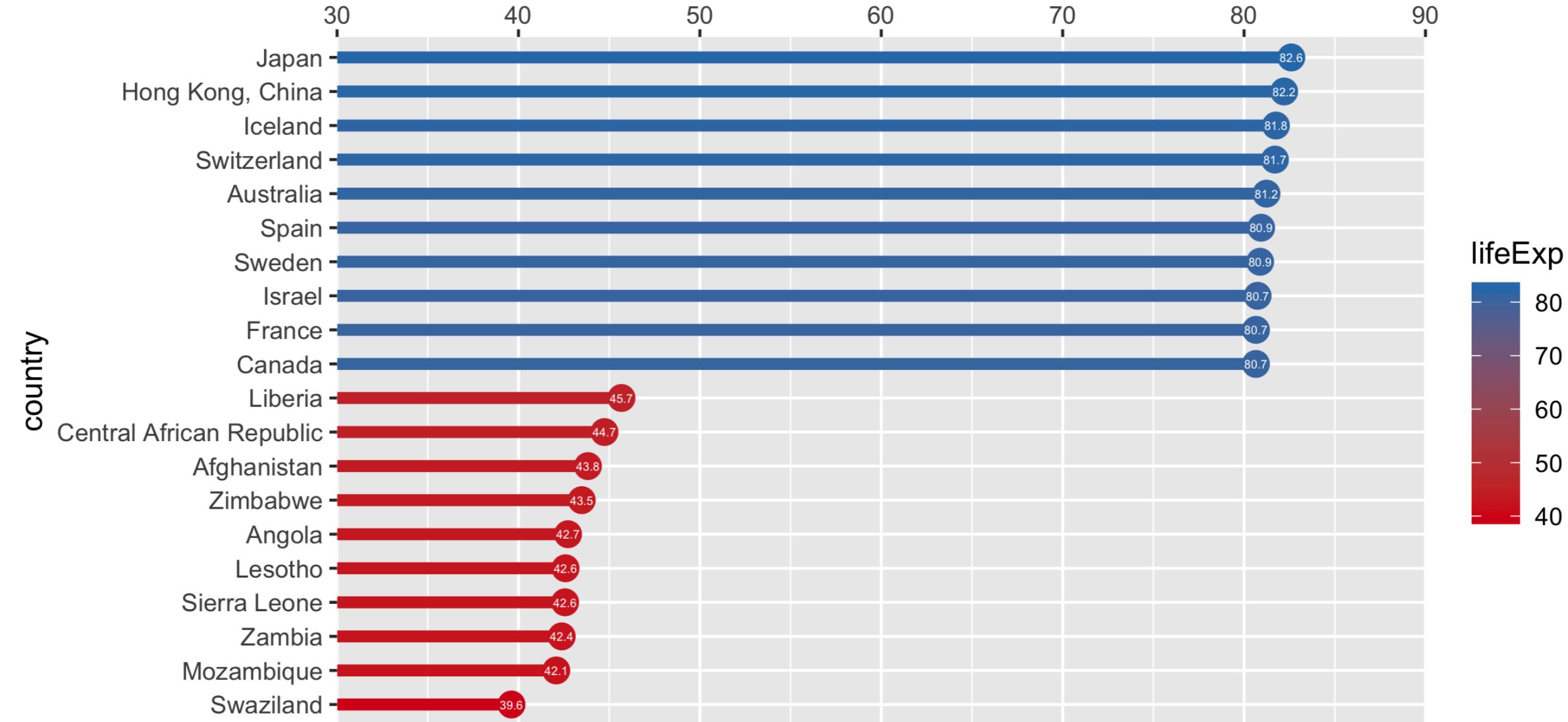
Use intuitive and attractive geoms



Add text labels to your plot

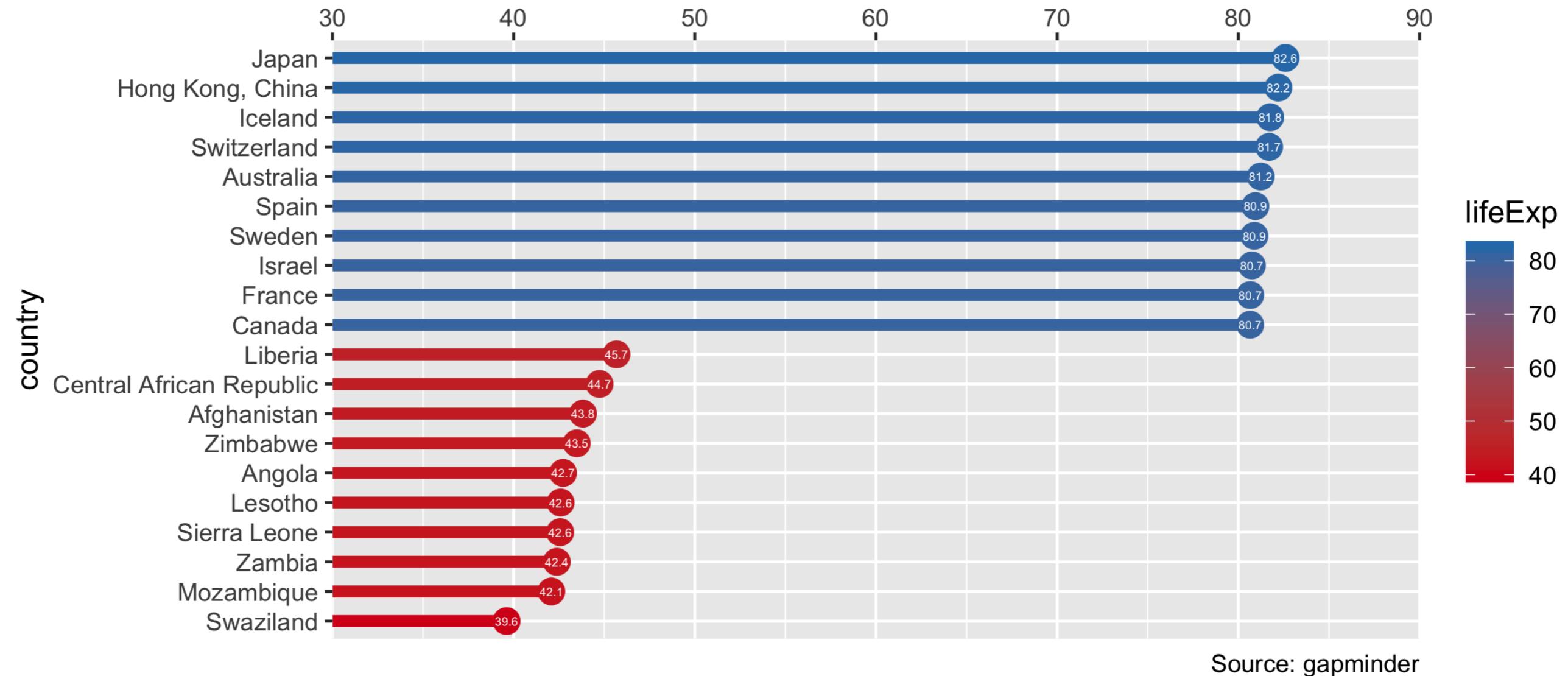


Use appropriate scales

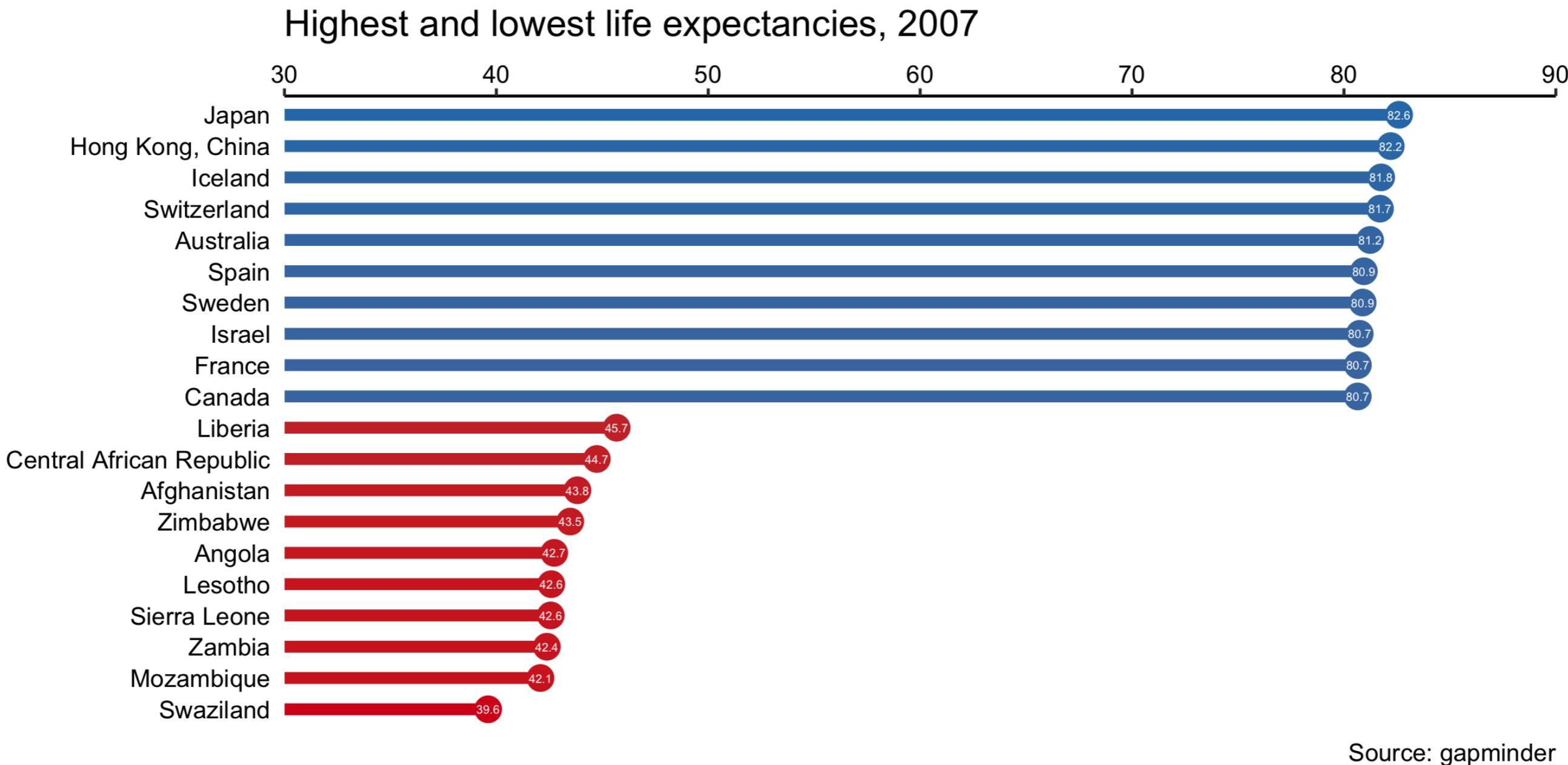


Add useful titles and citations

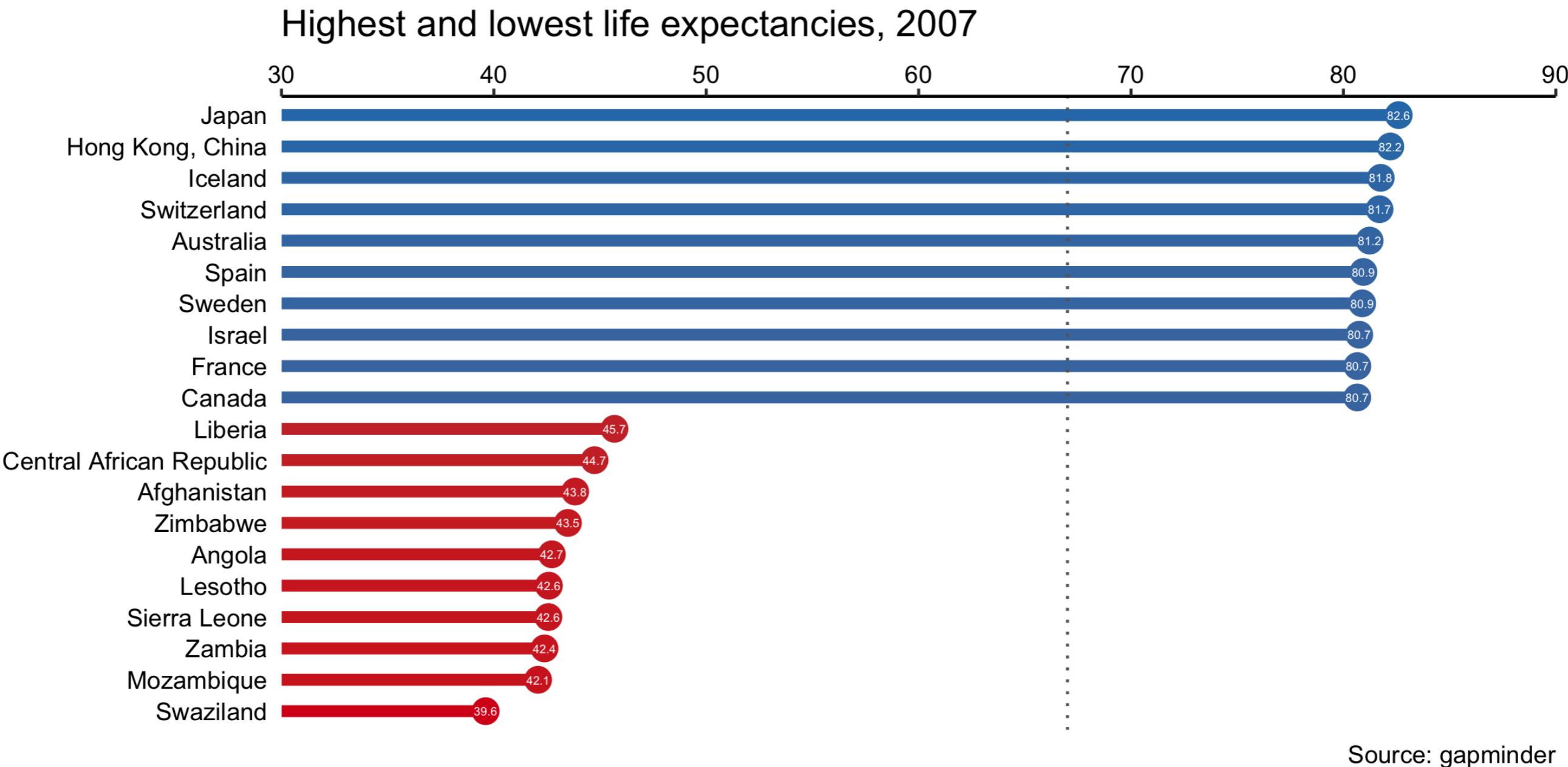
Highest and lowest life expectancies, 2007



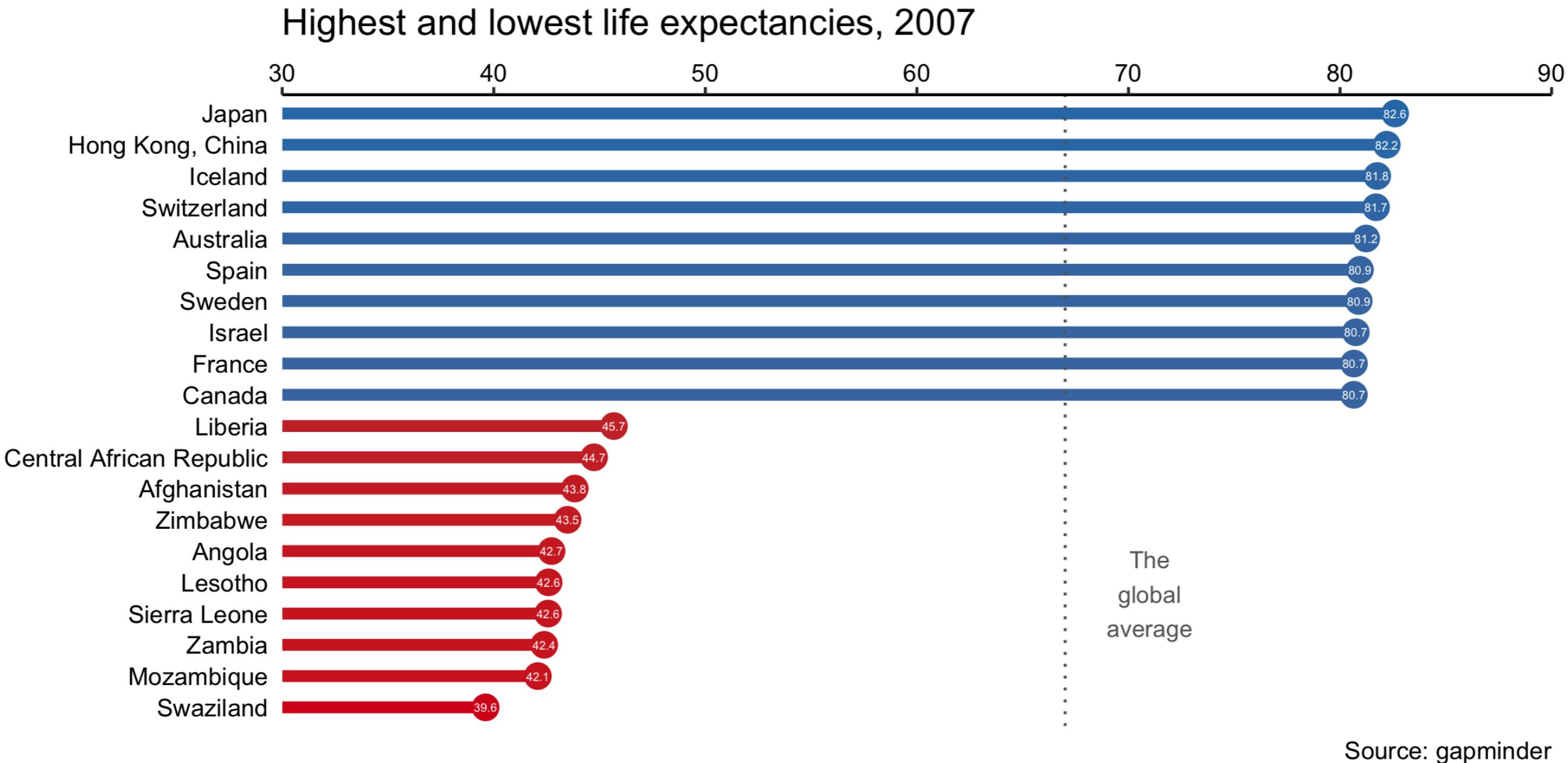
Remove non-data ink



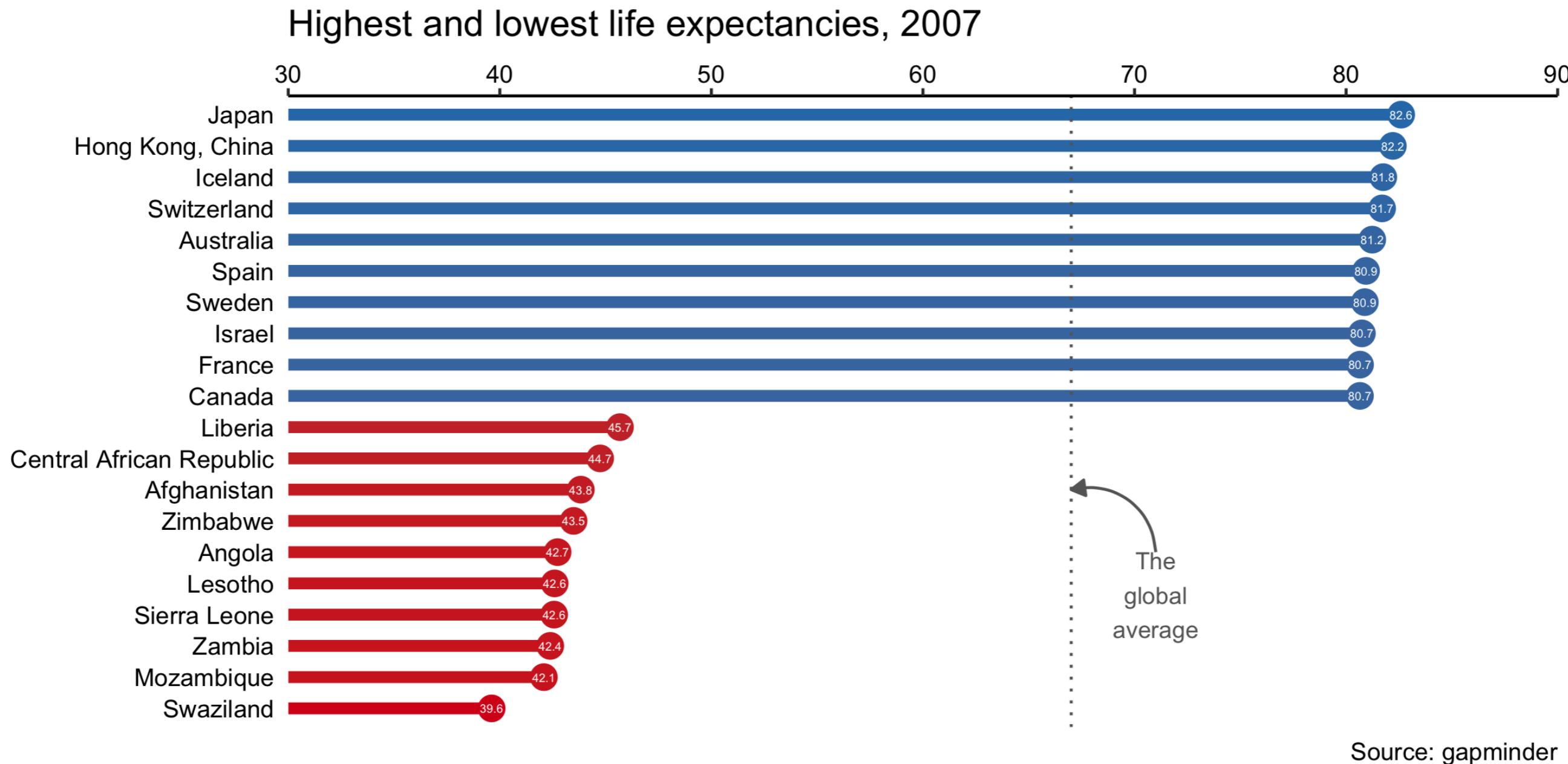
Add threshold lines



Add informative text



Add embellishments



Let's practice!

INTRODUCTION TO DATA VISUALIZATION WITH GGPLOT2