ggplot2

Matthew Flickinger, Ph.D.

University of Michigan

BDSI June 24, 2019

Why plot?

Raw Data

X	Y
55.38	97.18
51.53	96.02
46.15	94.49
42.82	91.41
40.76	88.33
38.71	84.87
35.64	79.87
…	…

Summaries

Statistic	X	Y
Mean	54.26	47.83
SD	16.7	26.9

Regression Statistic Y=X	Value
Intercept	53.4
Slope	-0.10
Correlation	-0.06

Why plot?

Summaries

Statistic	X	Y
Mean	54.26	47.83
SD	16.7	26.9

Regression Statistic Y=X	Value
Intercept	53.4
Slope	-0.10
Correlation	-0.06

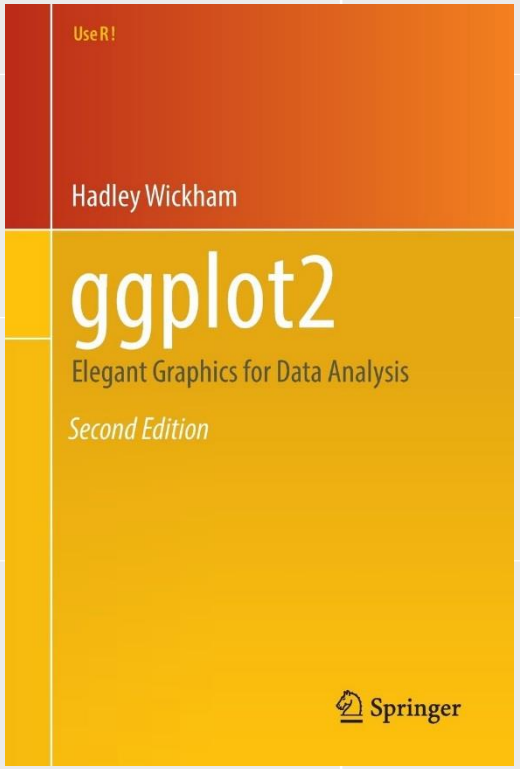
Data from https://cran.r-project.org/package=datasauRus

Installing ggplot2

- Even though the package is sometimes just referred to as "ggplot", the package name is "ggplot2"
- ggplot is included in the tidyverse package. To load the tidyverse package, run
  - `library(tidyverse)`
- If you get the message "there is no package 'tidyverse' " you must install it first
  - `install.packages("tidyverse")`
  - `library(tidyverse)`
- Be sure to load the package at the start of your session

ggplot2 help

- Use R help "?ggplot"
- Use website (has pictures)
  - http://ggplot2.tidyverse.org/reference/
  - [open now]
- Read Hadley's book



Second Edition published June 2016

Gapminder Data

- Dataset tracking life expectancy and per-capita GDP of 142 countries
- Data reported every five years from 1952-2007

- Available in R package on CRAN
  - `install.packages("gapminder")`
  - `library(gapminder)`
  - `View(gapminder)`

Gapminder Data

```
# A tibble: 1,704 x 6
   country     continent  year lifeExp      pop gdpPercap
   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
 1 Afghanistan Asia       1952    28.8  8425333      779.
 2 Afghanistan Asia       1957    30.3  9240934      821.
 3 Afghanistan Asia       1962    32.0 10267083      853.
 4 Afghanistan Asia       1967    34.0 11537966      836.
 5 Afghanistan Asia       1972    36.1 13079460      740.
 6 Afghanistan Asia       1977    38.4 14880372      786.
 7 Afghanistan Asia       1982    39.9 12881816      978.
 8 Afghanistan Asia       1987    40.8 13867957      852.
 9 Afghanistan Asia       1992    41.7 16317921      649.
10 Afghanistan Asia       1997    41.8 22227415      635.
# ... with 1,694 more rows
```

#start-plot

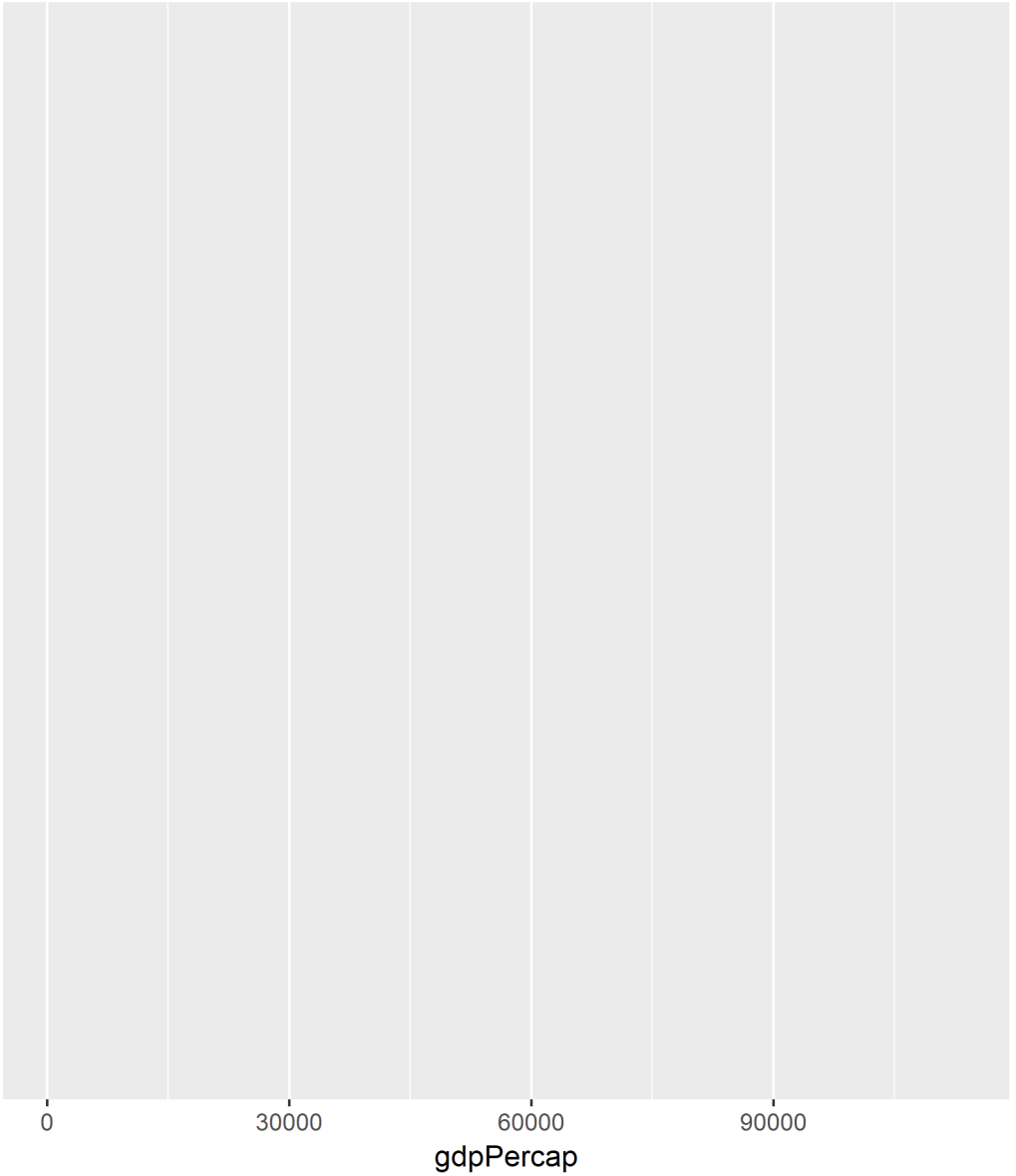
Start Plot

```
ggplot(data = gapminder)
```

#add-x

Add x variable

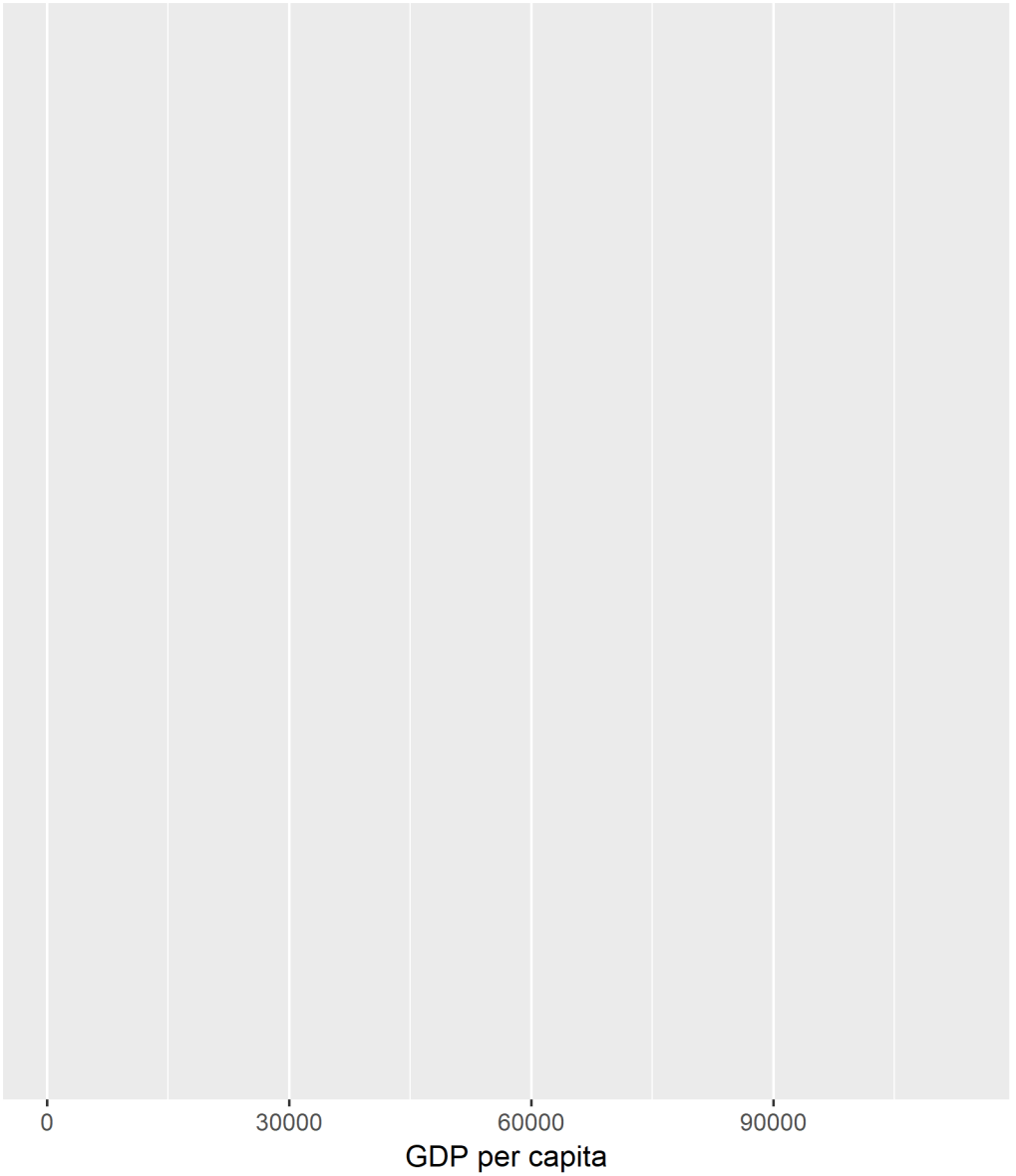
```
ggplot(data = gapminder) +
  aes(x = gdpPercap)
```



#add-x-label

Fix x label

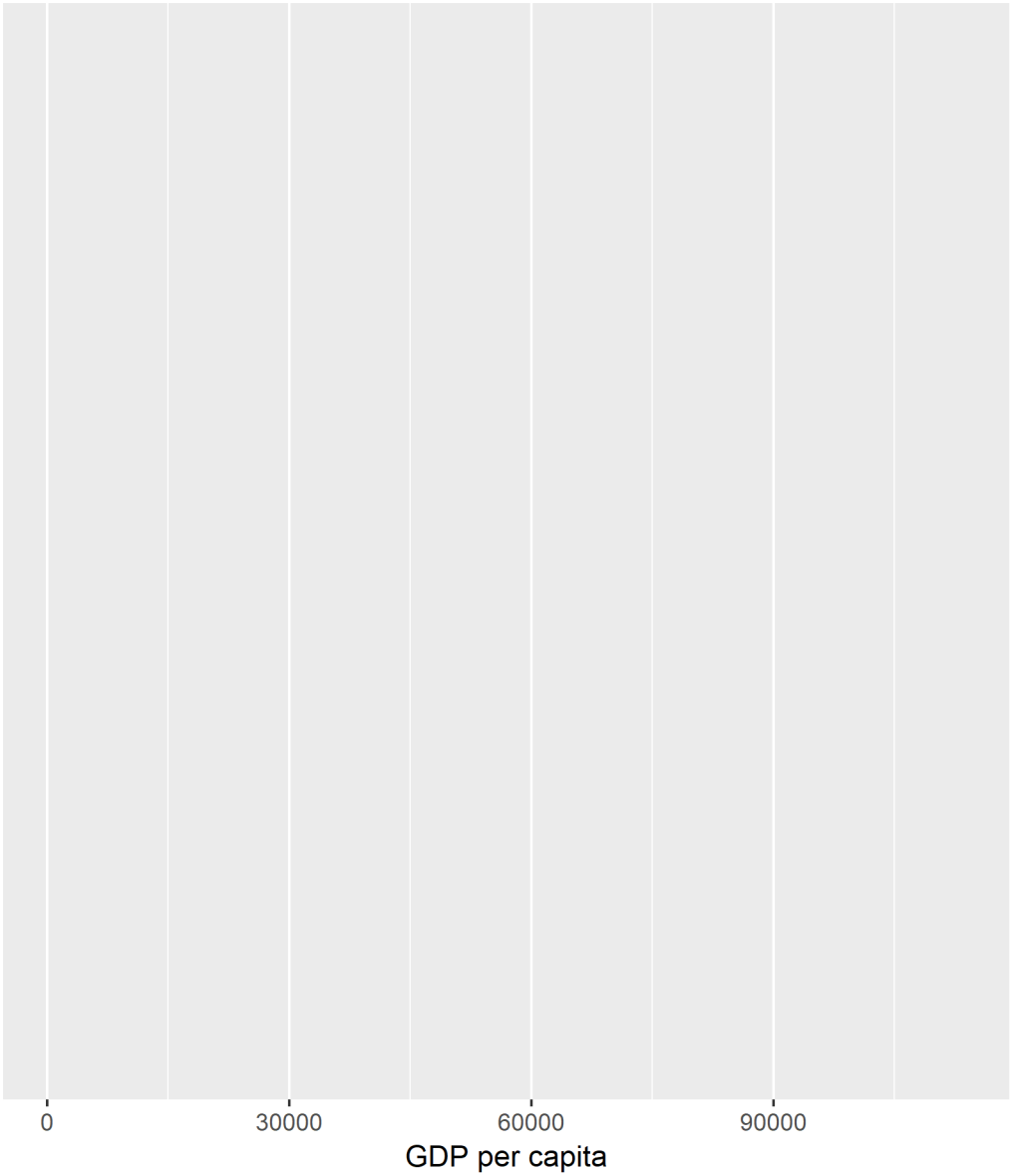
```
ggplot(data = gapminder) +
  aes(x = gdpPercap) +
  labs(x = "GDP per capita")
```



#add-y

Add y variable

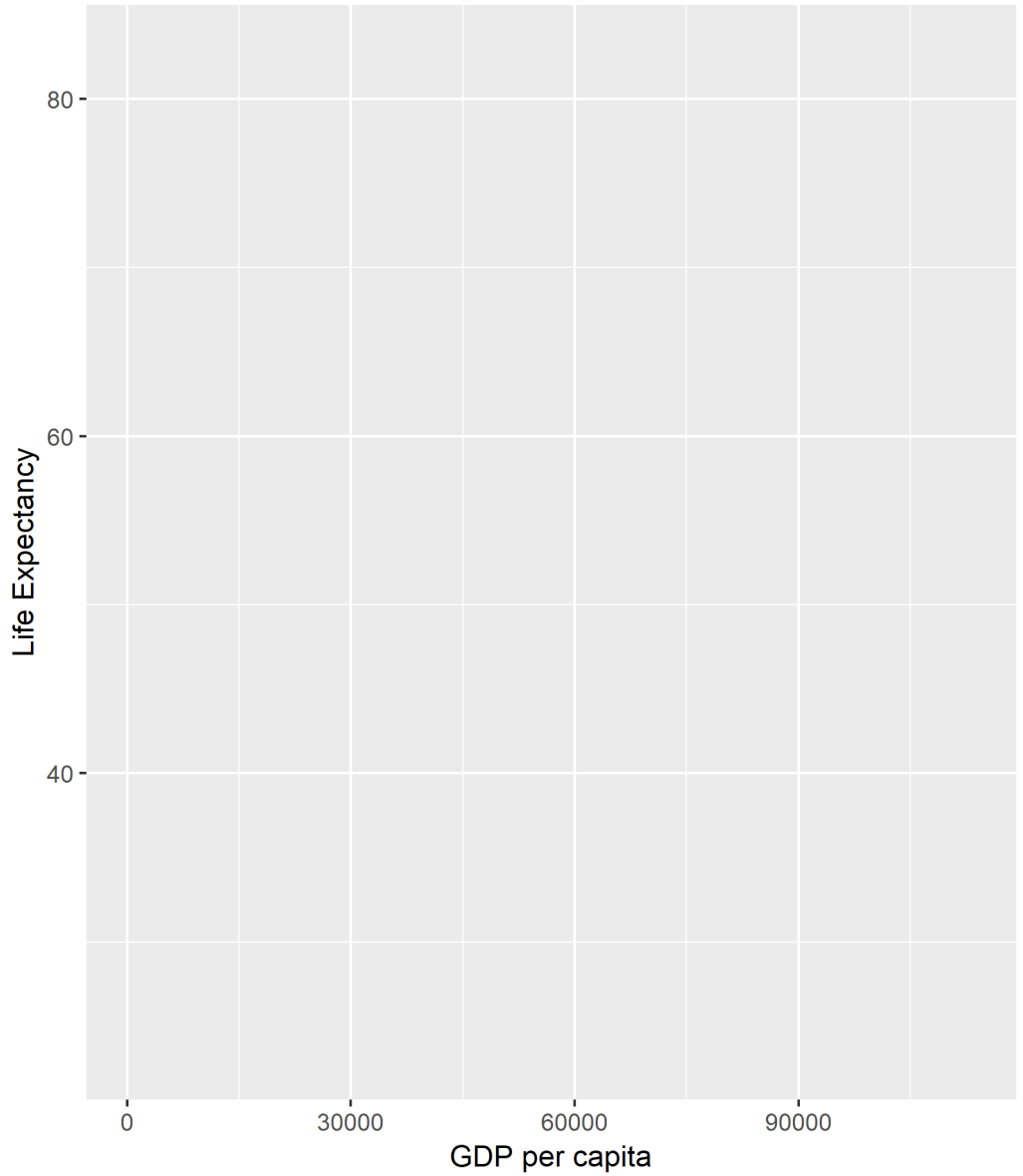
```
ggplot(data = gapminder) +
  aes(x = gdpPercap) +
  labs(x = "GDP per capita") +
  aes(y = lifeExp)
```



#add-y-label

Fix x label

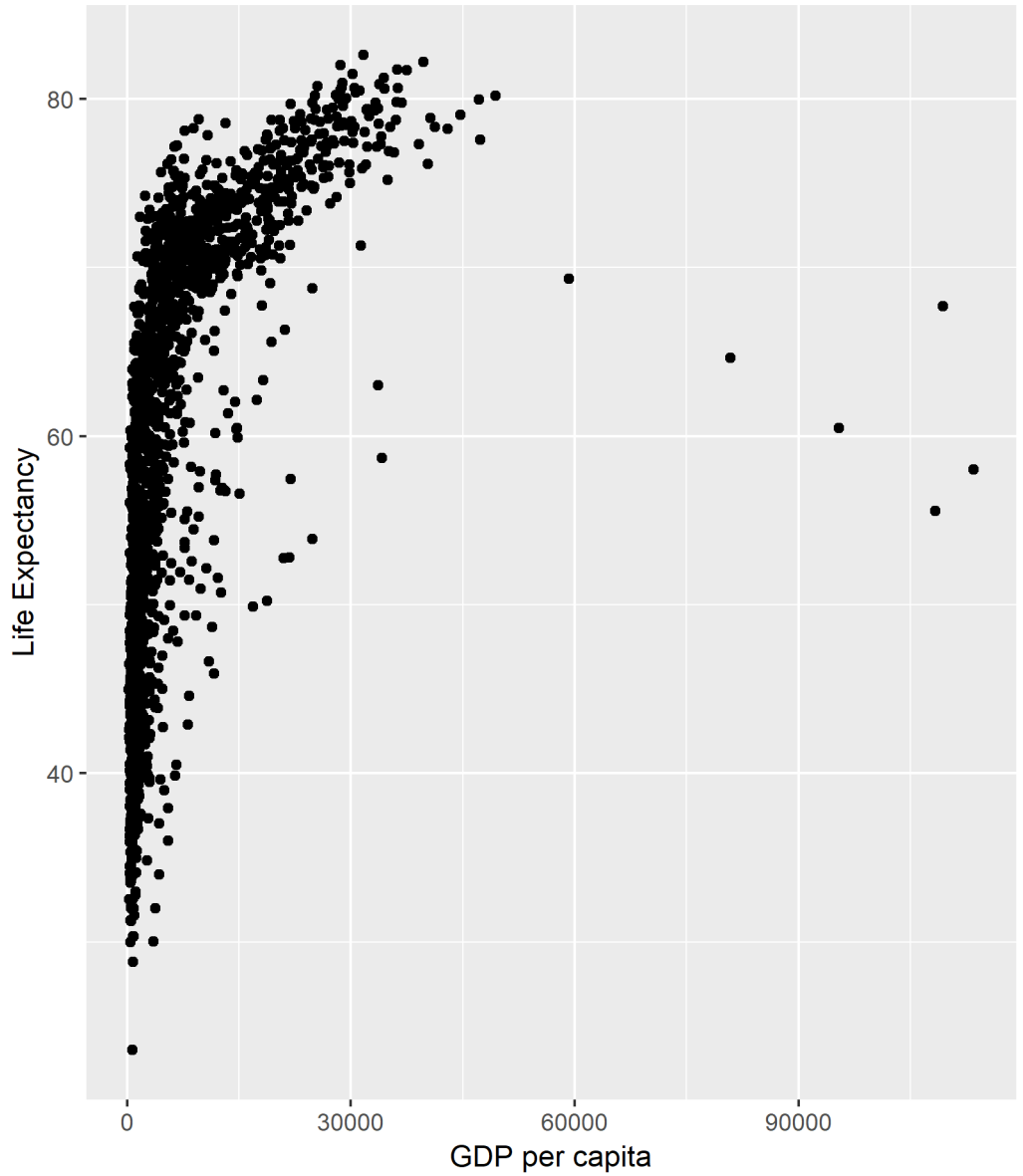
```
ggplot(data = gapminder) +
  aes(x = gdpPercap) +
  labs(x = "GDP per capita") +
  aes(y = lifeExp) +
  labs(y = "Life Expectancy")
```



#add-points

Draw points

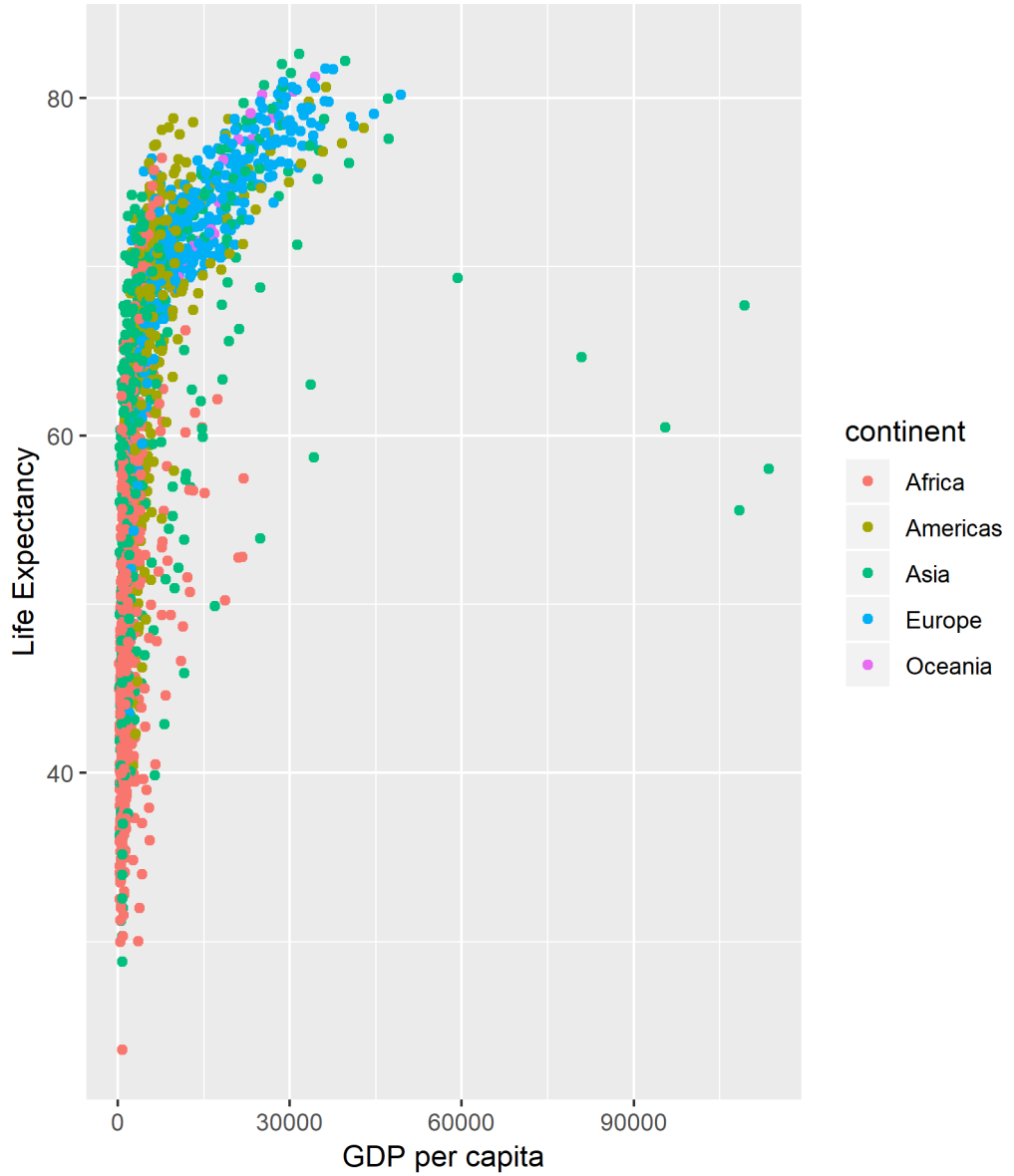
```
ggplot(data = gapminder) +
  aes(x = gdpPercap) +
  labs(x = "GDP per capita") +
  aes(y = lifeExp) +
  labs(y = "Life Expectancy") +
  geom_point()
```



#add-color

Color by continent

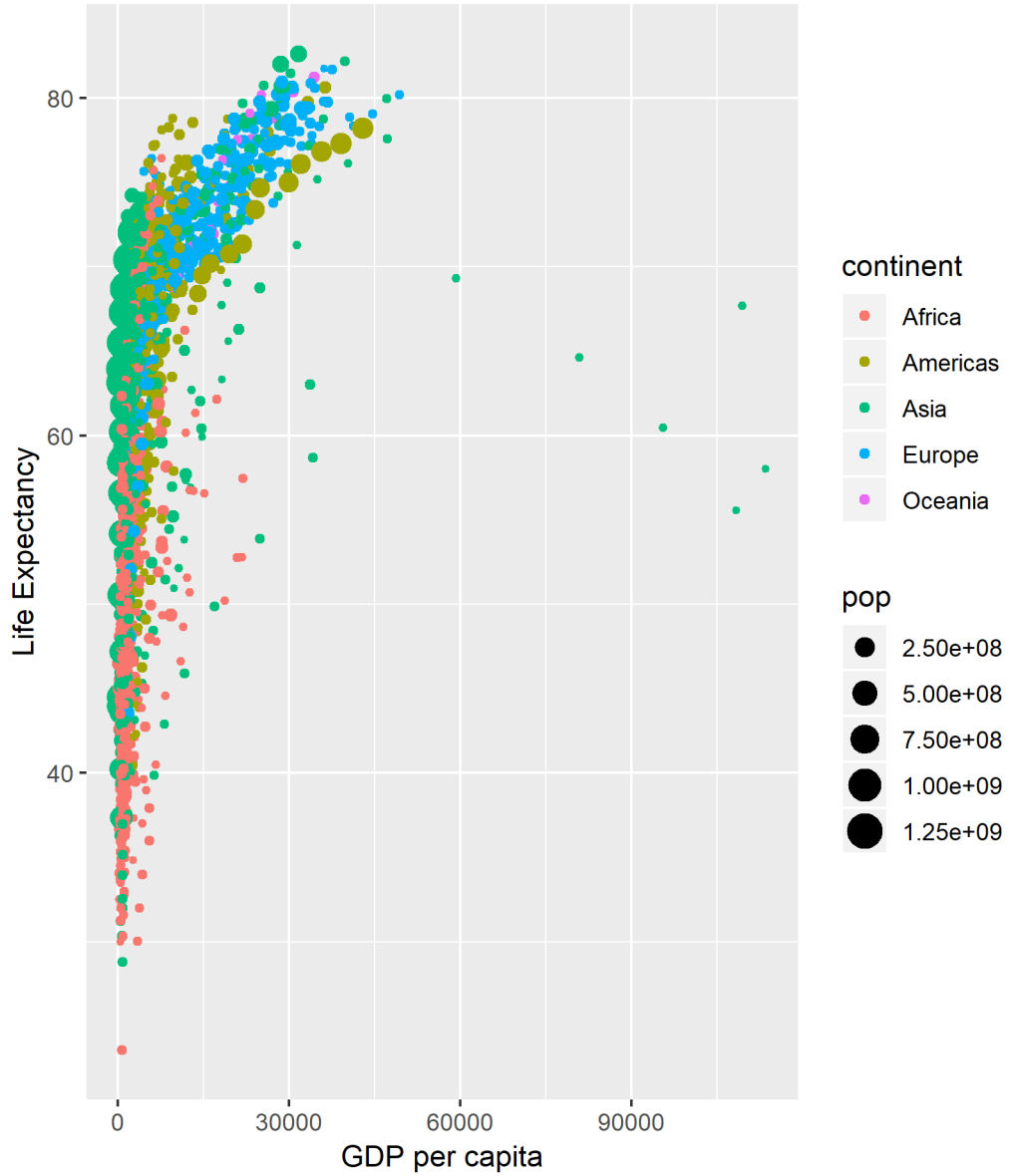
```
ggplot(data = gapminder) +
  aes(x = gdpPercap) +
  labs(x = "GDP per capita") +
  aes(y = lifeExp) +
  labs(y = "Life Expectancy") +
  geom_point() +
  aes(color=continent)
```



#add-size

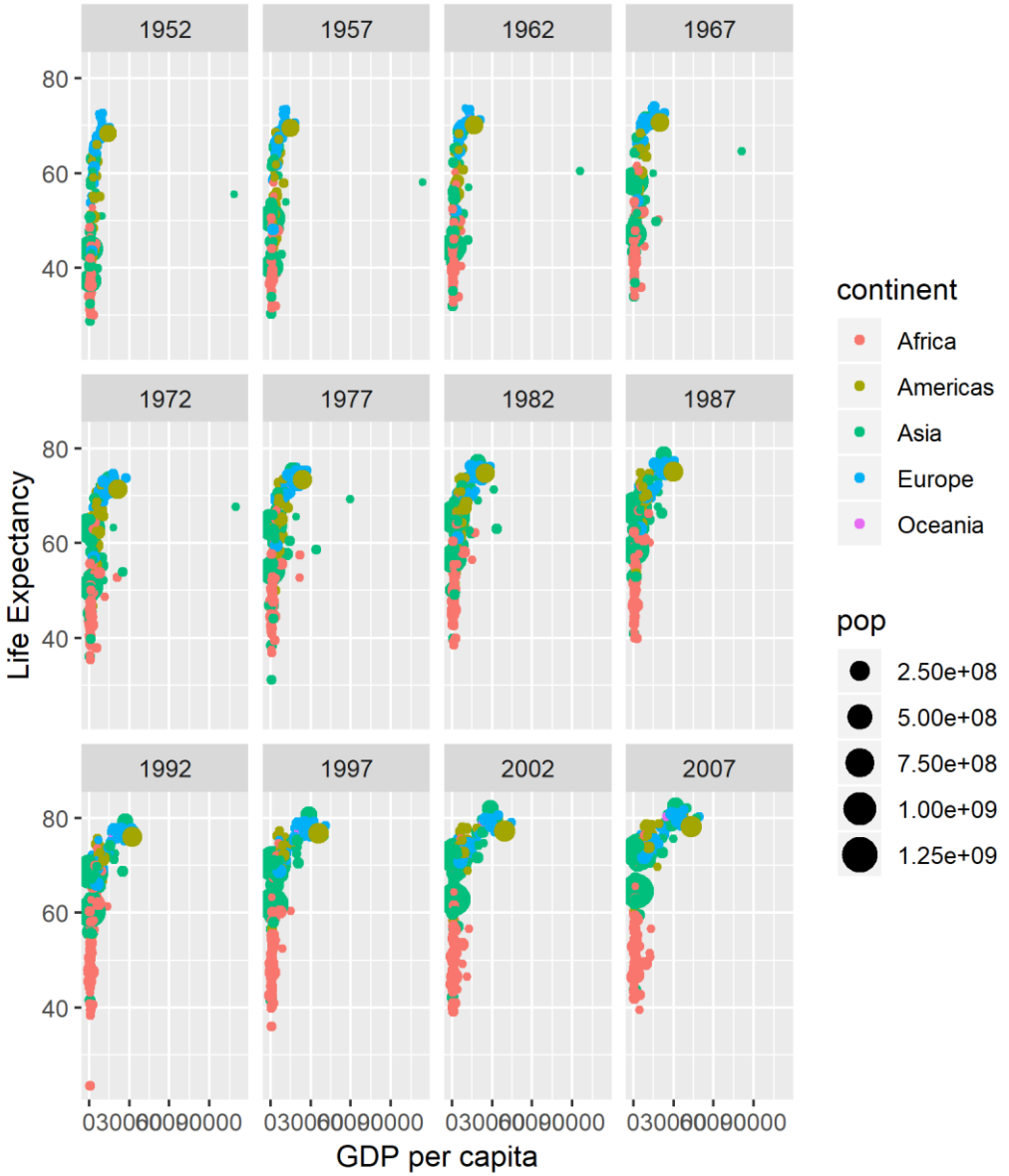
Size by population

```
ggplot(data = gapminder) +
  aes(x = gdpPercap) +
  labs(x = "GDP per capita") +
  aes(y = lifeExp) +
  labs(y = "Life Expectancy") +
  geom_point() +
  aes(color=continent) +
  aes(size = pop)
```



Small plots by year

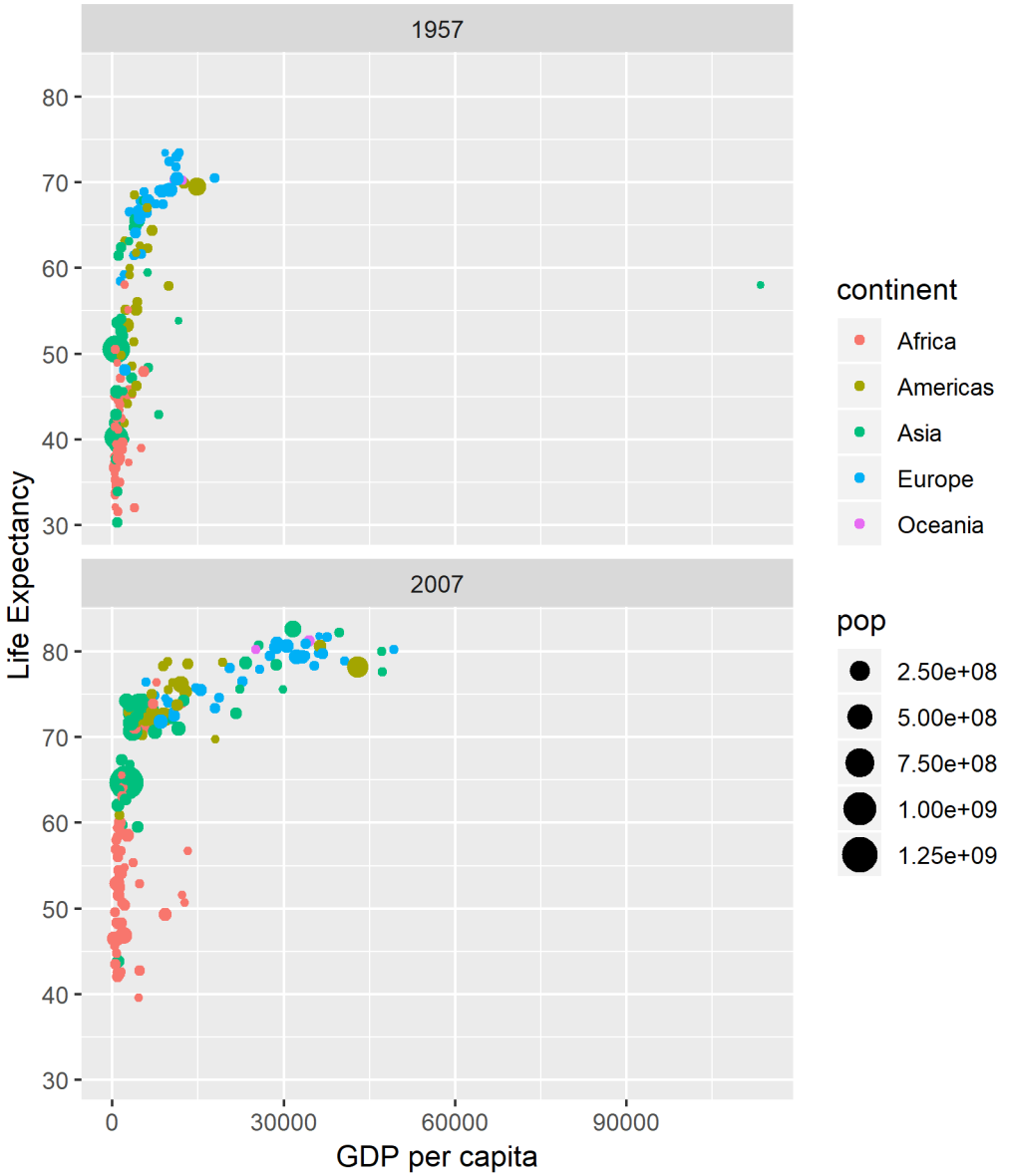
```
ggplot(data = gapminder) +
  aes(x = gdpPercap) +
  labs(x = "GDP per capita") +
  aes(y = lifeExp) +
  labs(y = "Life Expectancy") +
  geom_point() +
  aes(color=continent) +
  aes(size = pop) +
  facet_wrap(vars(year))
```



#add-data-filter

Filter years with dplyr

```
gapminder %>%
  filter(
    year %in% c(1957, 2007)
  ) %>%
ggplot(data = .) +
  aes(x = gdpPercap) +
  labs(x = "GDP per capita") +
  aes(y =lifeExp) +
  labs(y = "Life Expectancy") +
  geom_point() +
  aes(color=continent) +
  aes(size = pop) +
  facet_wrap(vars(year), nrow=2)
```



Reorganize code (optional)

```
gapminder %>%
  filter(
    year %in% c(1957, 2007)
  ) %>%
ggplot(data = .) +
geom_point(mapping = aes(
    x = gdpPercap, y = lifeExp,
    color=continent, size = pop)) +
  facet_wrap(vars(year), nrow=2) +
  labs(x = "GDP per capita", y = "Life Expectancy")
```

Combine all the aes() options into one. Pass as the mapping= arguments of the geometry

Combine the labs() together

Aesthetics, aes()

- Mappings between a column of your data and some property of the geometry being drawn
- Can pass as the `mapping=` argument
  - `ggplot(data=, mapping=)`
  - `geom_xxx(mapping=, data=, ...)`
- If unnamed, aes() assumes the first two arguments are x and y
  - `aes(gdpPercap, lifeExp)`
  - `aes(x = gdpPercap, y = lifeExp)`
  - `aes(y = lifeExp, x = gdpPercap)`

  } Equivalent
- Never use "$" inside `aes()`

What other aes() does geom_point() know?

- Help page: `"?geom_point"`
- Check out the “Aesthetics” section of the help page
- Running `vignette("ggplot2-specs")` brings up more documentation

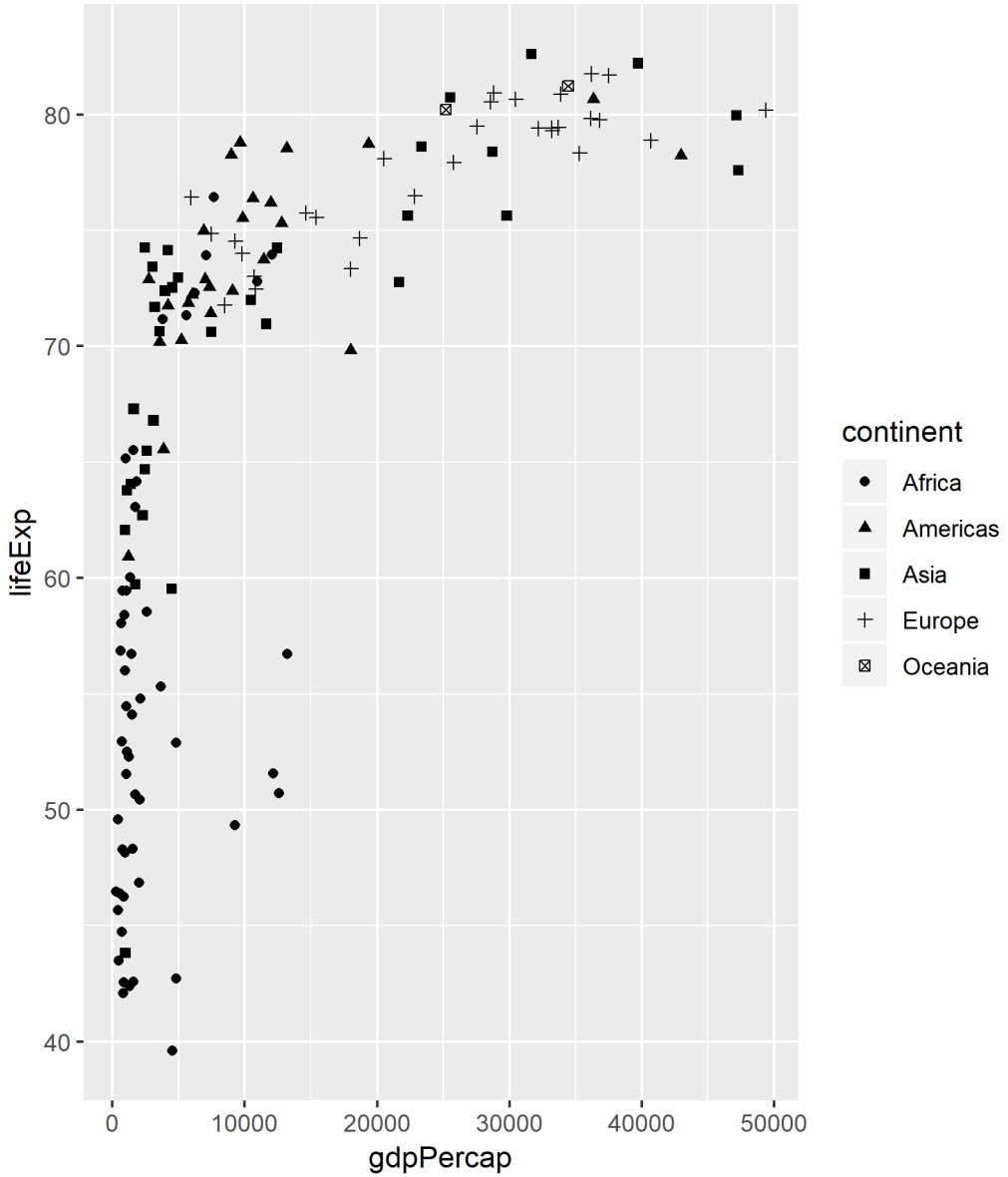
- **x** (required)
- **y** (required)
- alpha
- colour (color)
- fill
- group
- shape
- size
- stroke

#shape-poll

Change point shape?

```
gap2007 <- gapminder %>%
  filter(year == 2007)

ggplot(data = gap2007) +
  aes(x = gdpPercap) +
  aes(y = lifeExp) +
  geom_point() +
  aes( <??> = <??> )
```

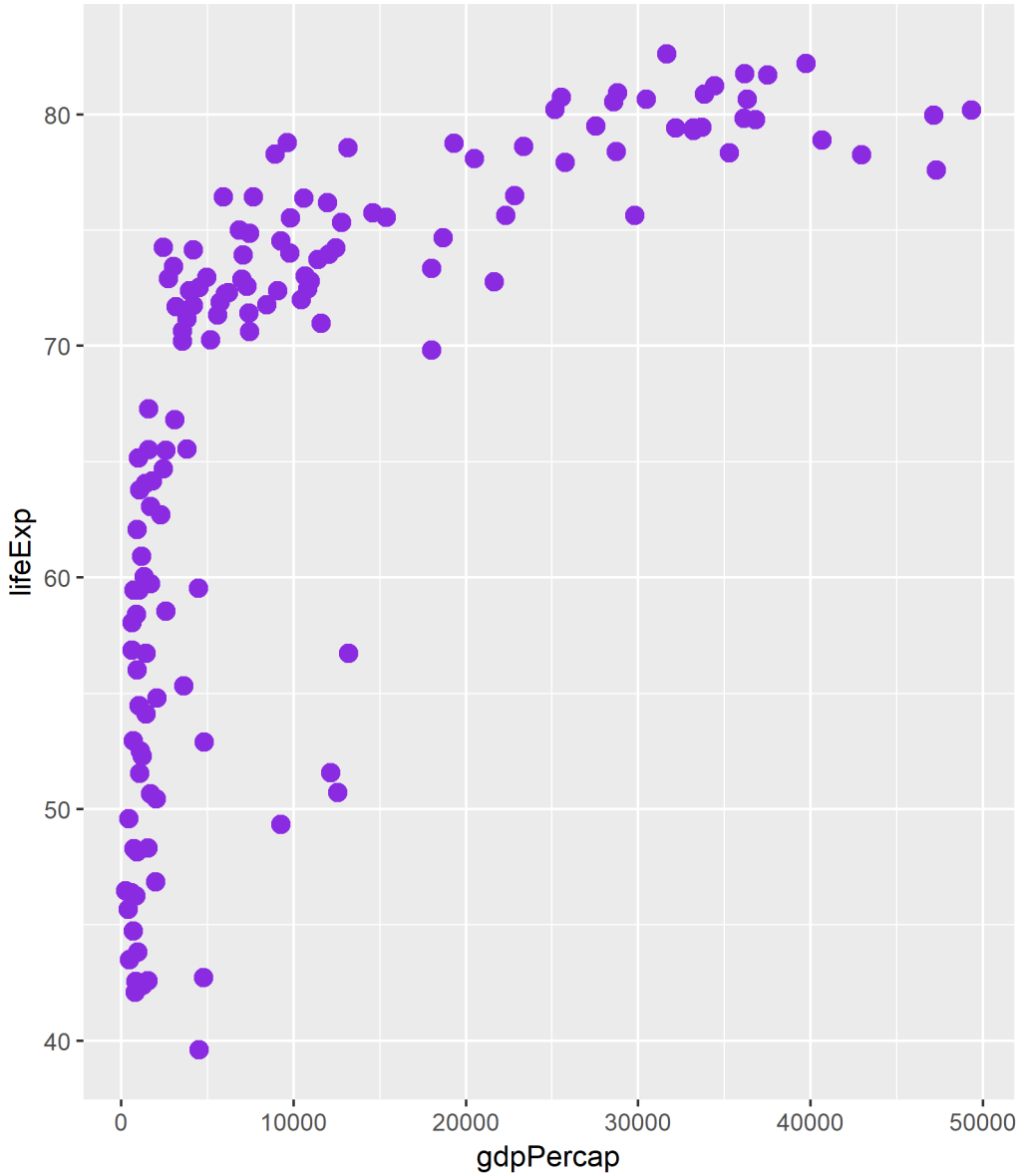
How can you get different shapes for each continent?



Fix values outside aes()

```
ggplot(data = gap2007) +
  aes(x = gdpPercap) +
  aes(y = lifeExp) +
  geom_point(
    size = 3,
    color = "blueviolet"
  )
```

If you want to set a value not related to your data, do so in the geometry layer, outside of `aes()`



Pick your favorite color

```
ggplot(data = gap2007) +
  aes(x = gdpPercap) +
  aes(y = lifeExp) +
  geom_point(
    size = 3,
    color = "    "
  )
```

```
# list all R color names
colors()

# choose 10 random colors
sample(colors(), 10)

# or specify a hex value
"#8A2BE2"
```

Find a cool color

Geometries

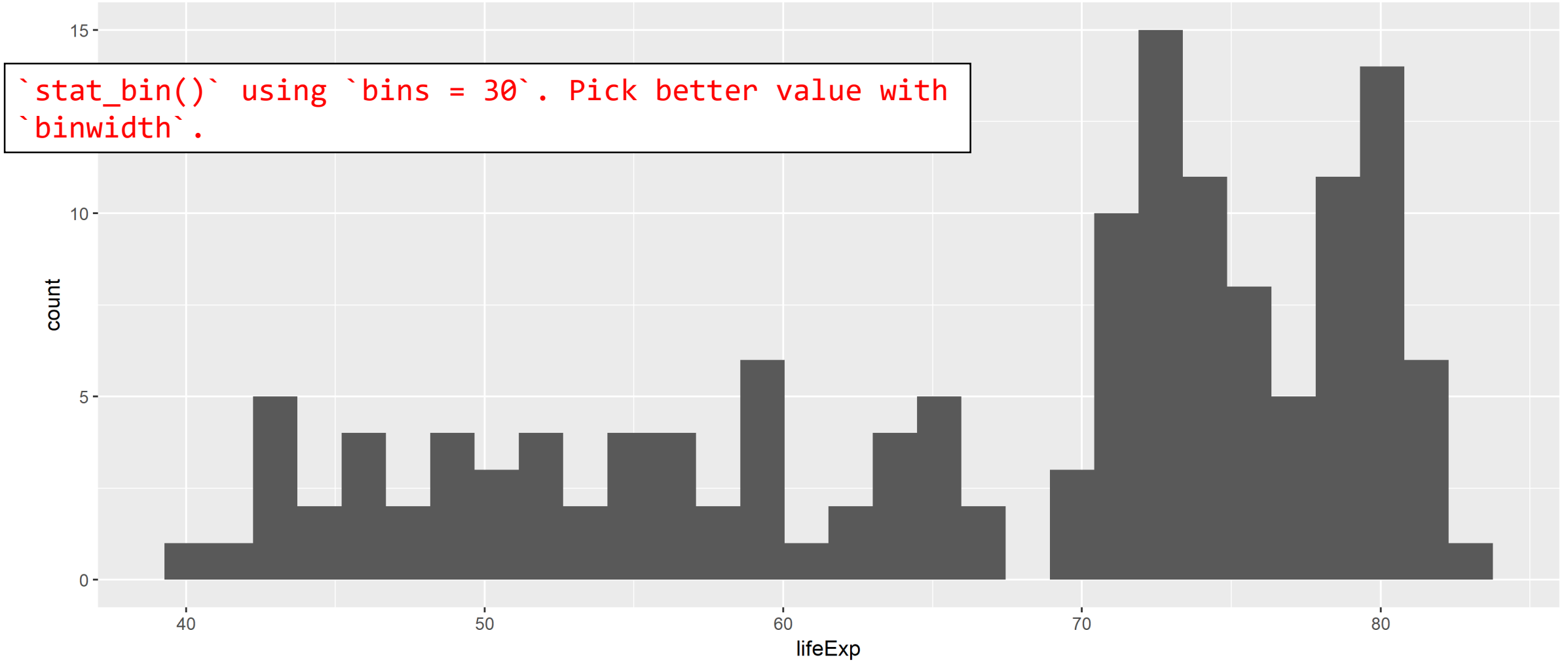
- `geom_point()` is just one of many geometries
- It is used to make scatter plots
- Works best with two continuous variables

- What if we wanted to look at a distribution for a single continuous variable?

#histogram

Single Variable Plot

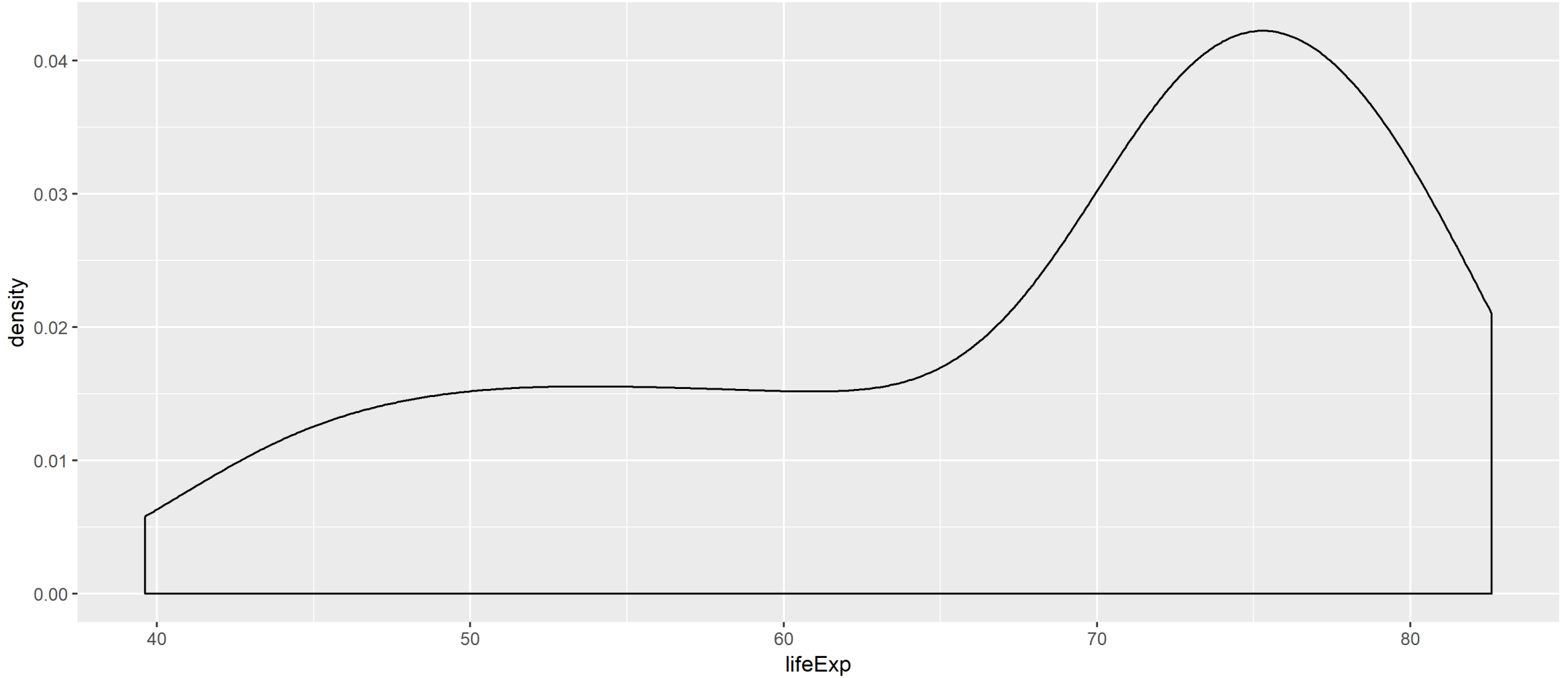
```
ggplot(gap2007) + aes(x=lifeExp) + geom_histogram()
```



#density

Single Variable Plot

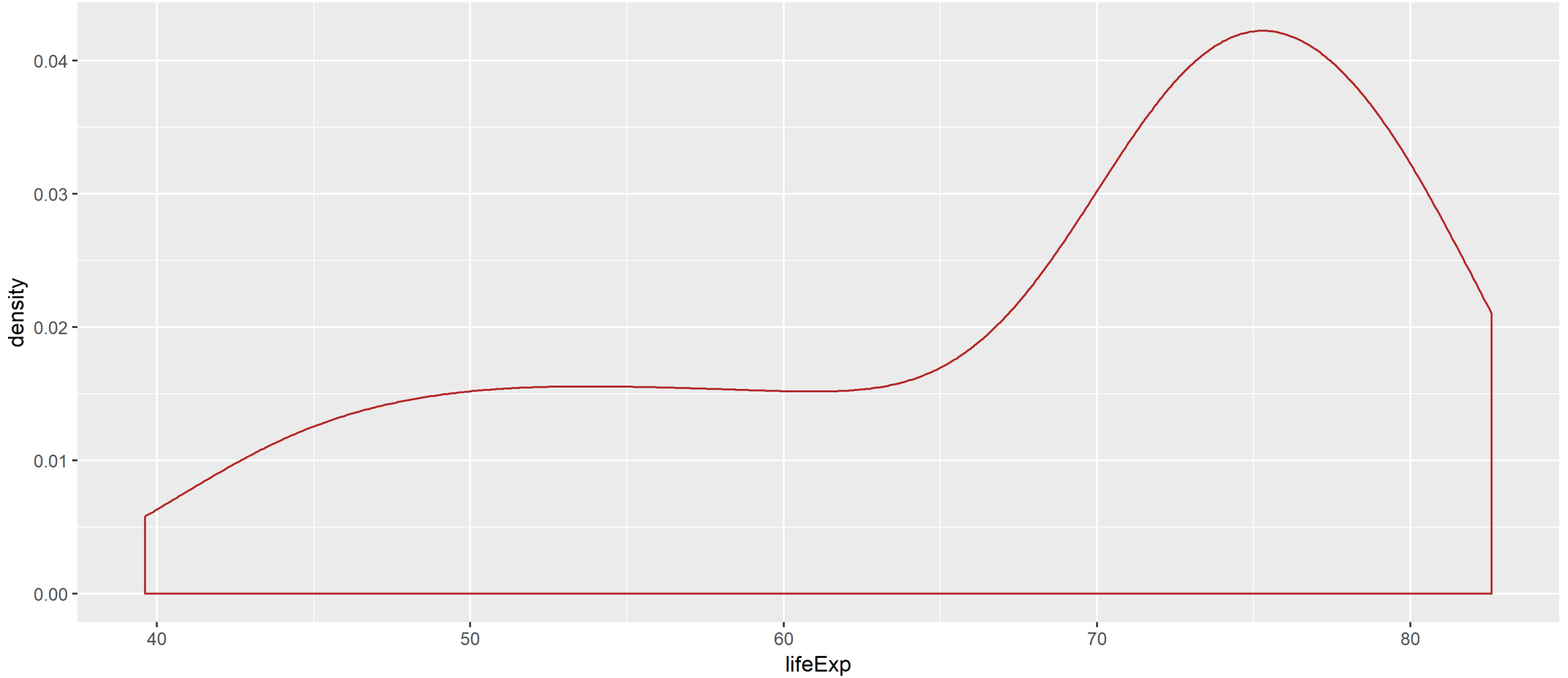
```
ggplot(gap2007) + aes(x=lifeExp) + geom_density()
```



#density-color

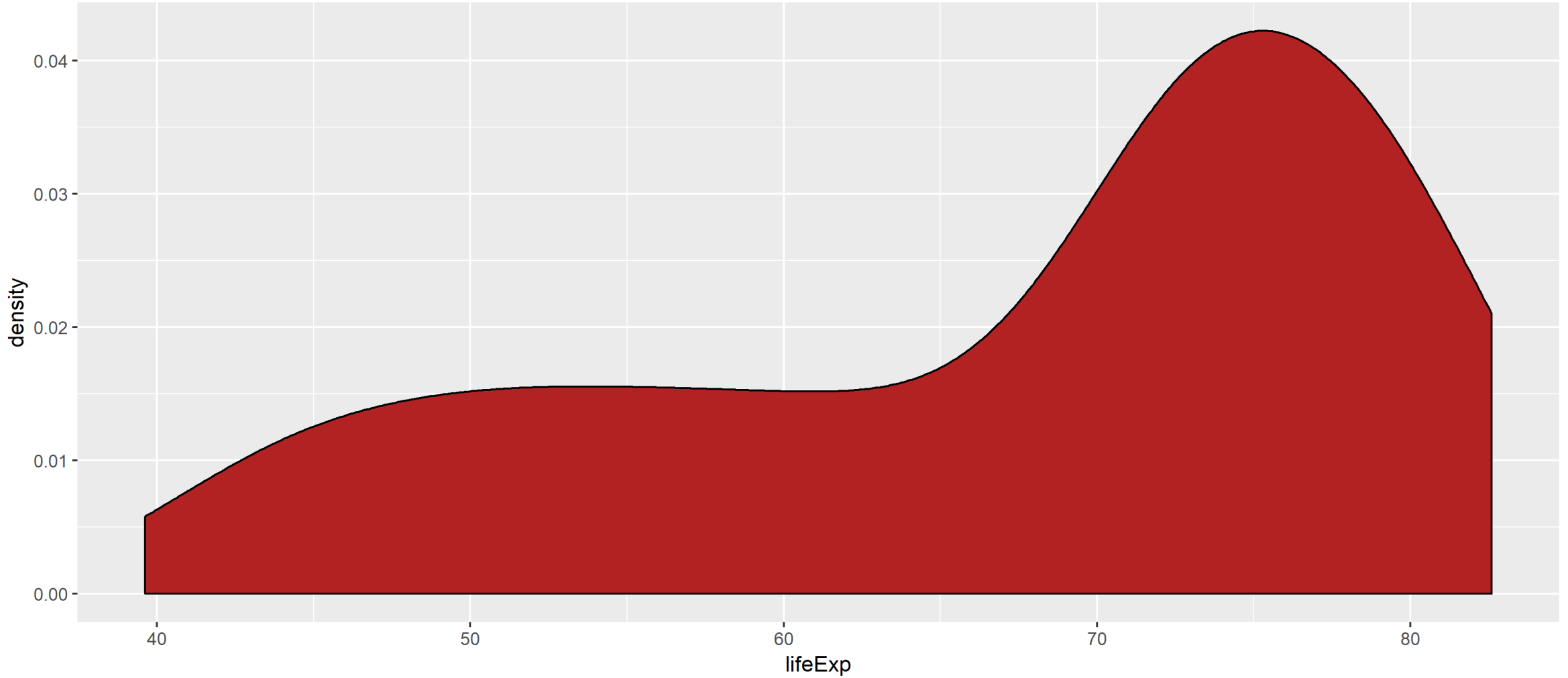
Single Variable Plot

```
ggplot(gap2007) + aes(x=lifeExp) + geom_density(color="firebrick")
```



Single Variable Plot

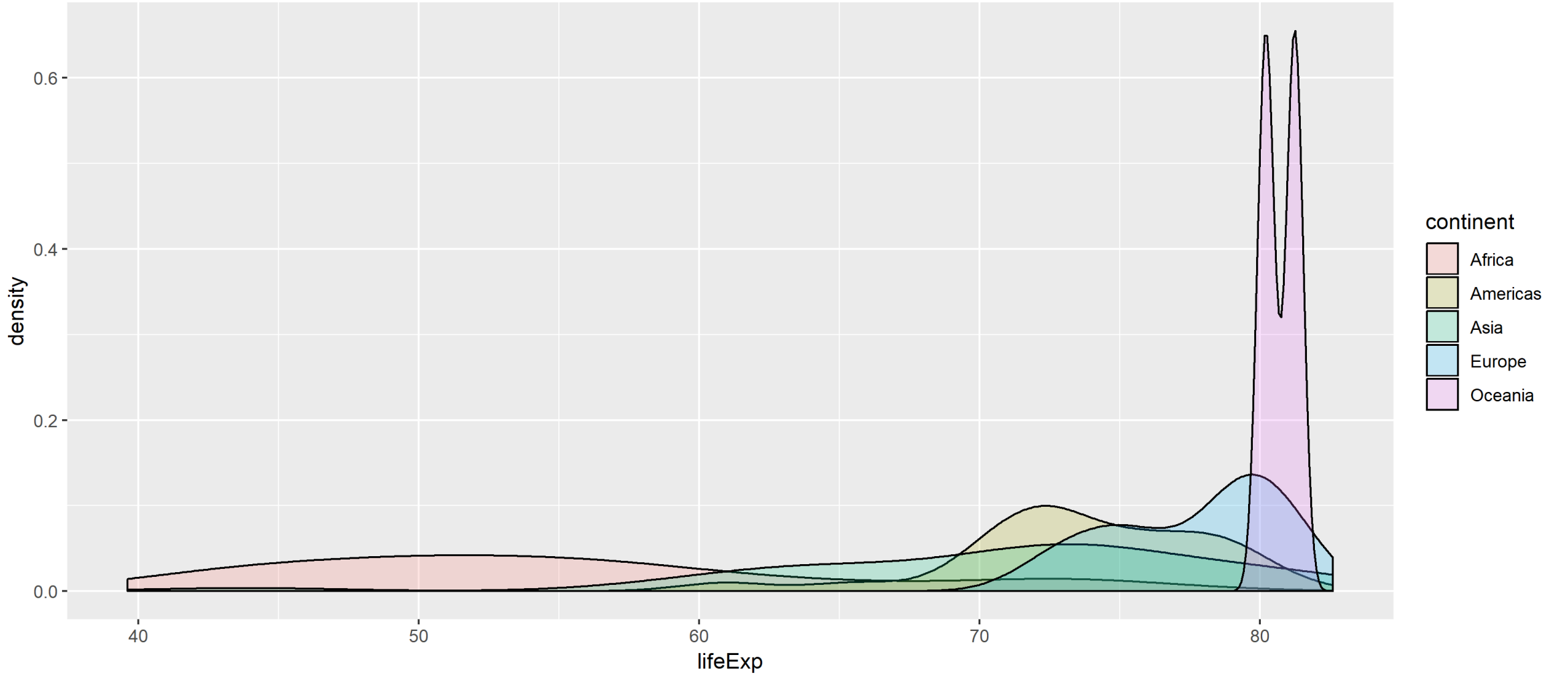
```
ggplot(gap2007) + aes(x=lifeExp) + geom_density(fill="firebrick")
```



#density-groups

Single Variable Plot

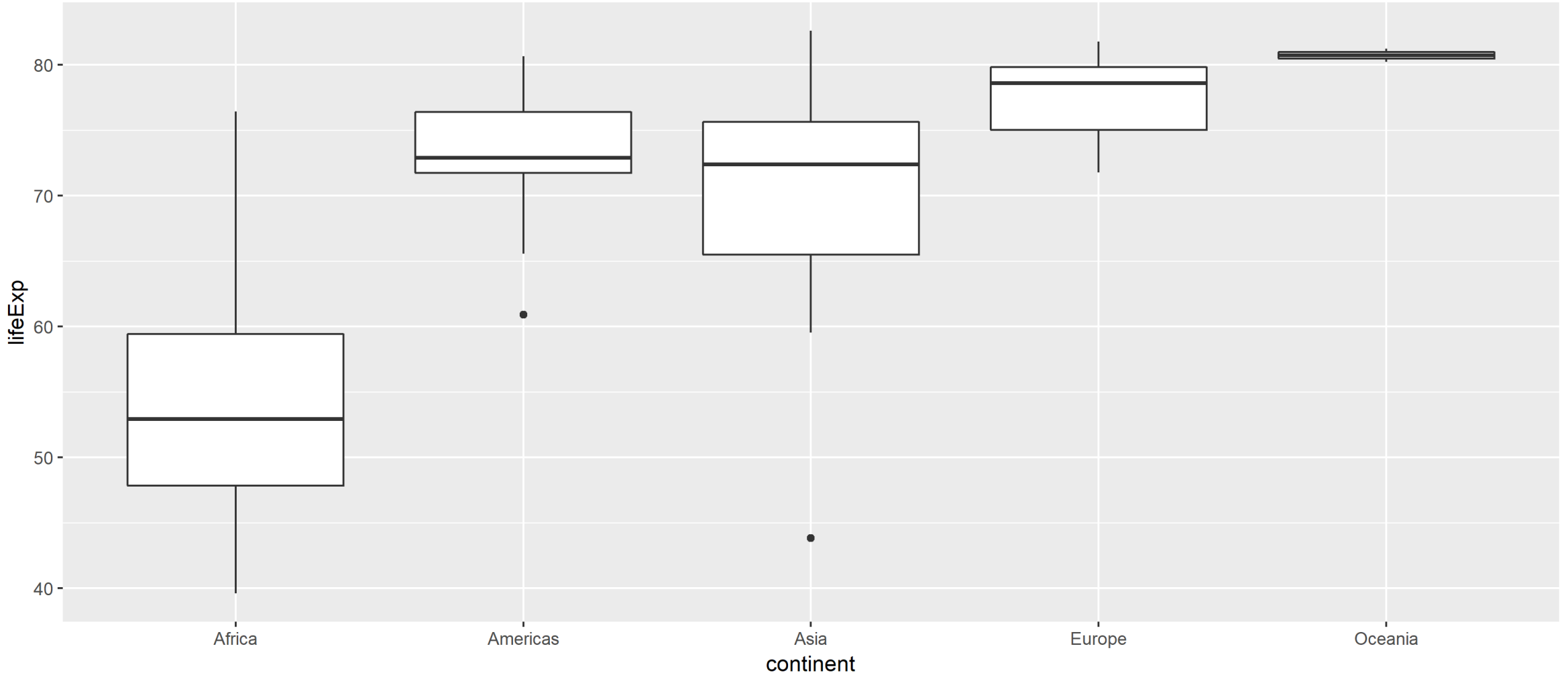
```
ggplot(gap2007) + aes(x=lifeExp, fill=continent) + geom_density(alpha=.2)
```



#box-plot

Single Variable Across Groups

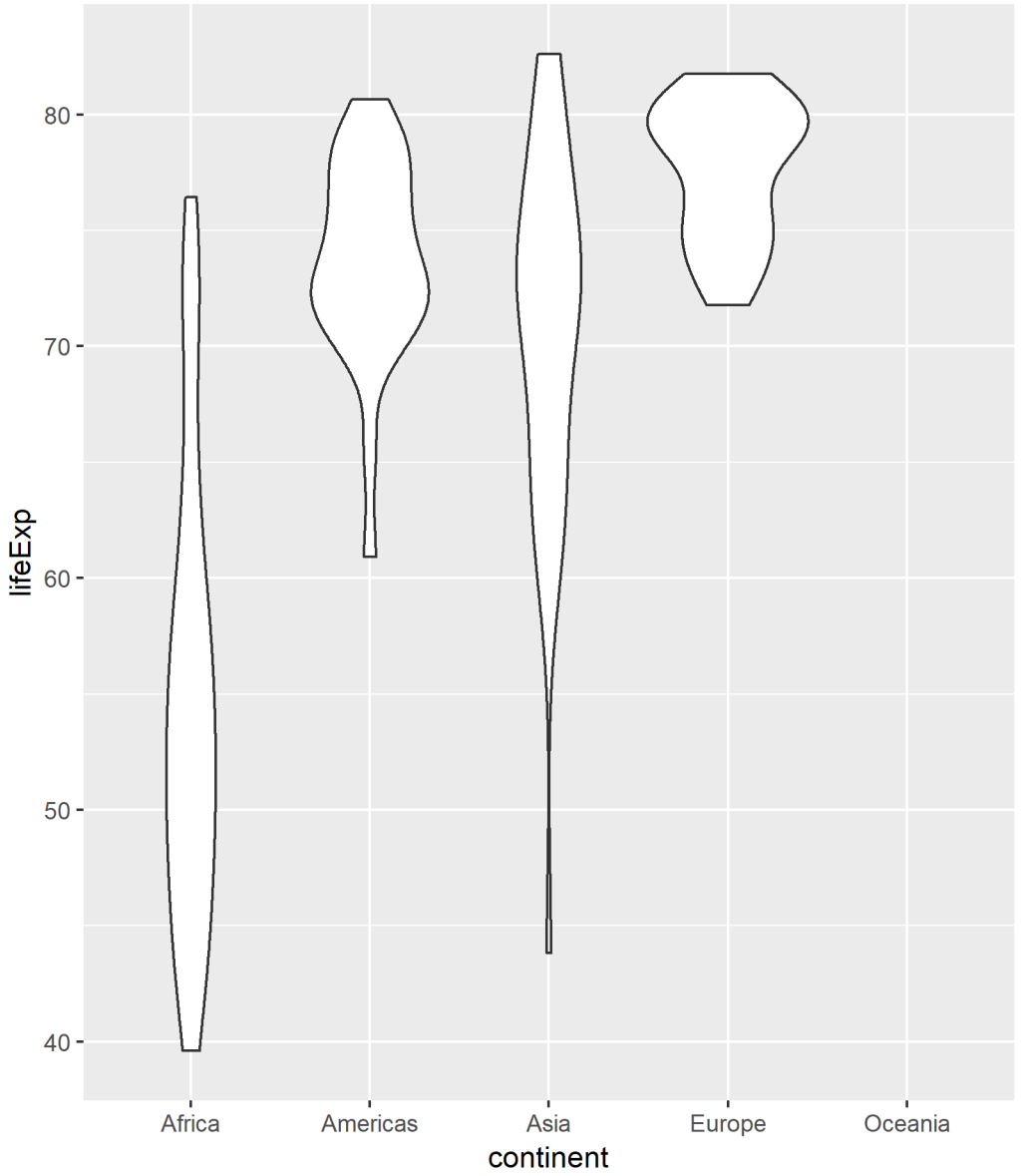
```
ggplot(gap2007) + aes(x=continent, y=lifeExp) + geom_boxplot()
```



#geom-poll

How can you get this plot?

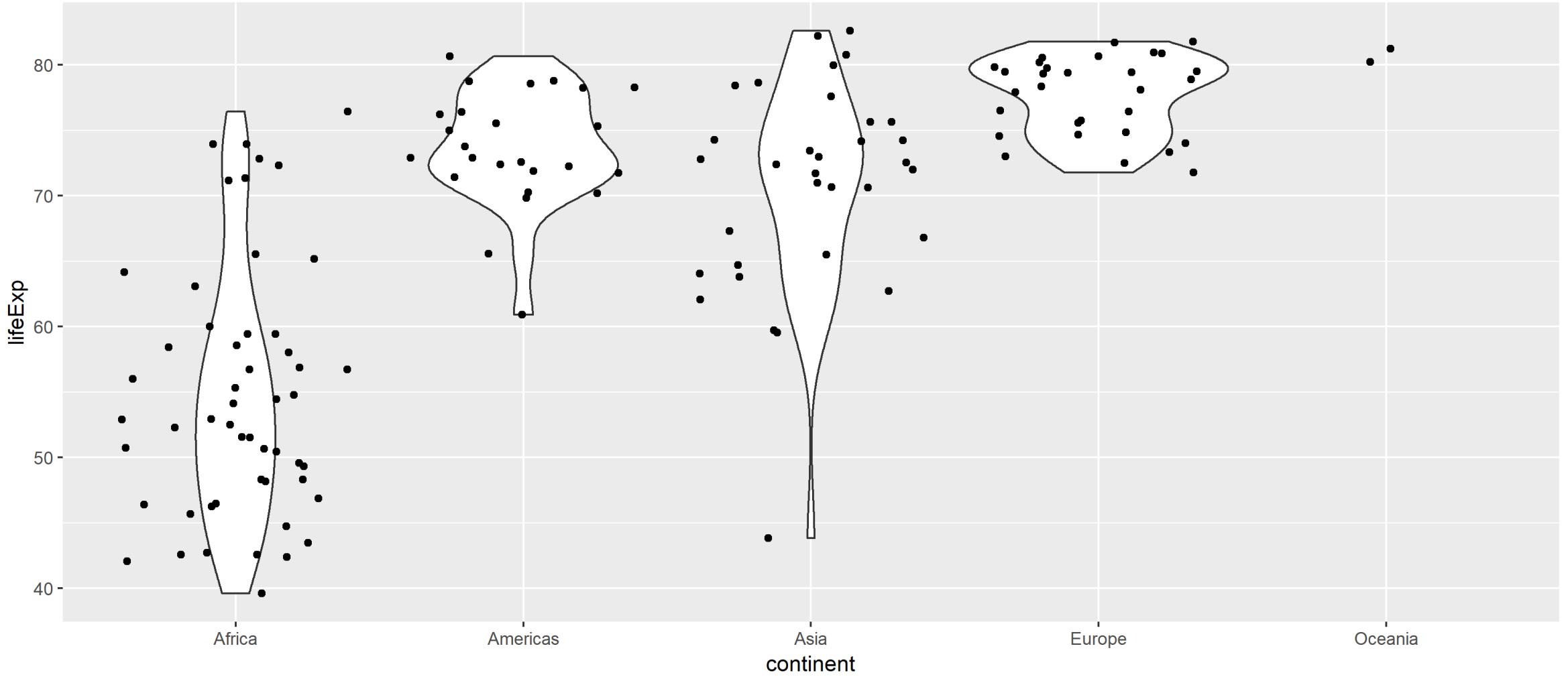
```
ggplot(gap2007) +
  aes(x = continent) +
  aes(y = lifeExp) +
  geom_???()
```



What geometry would give this plot? (check cheat sheet)

Single Variable Across Groups

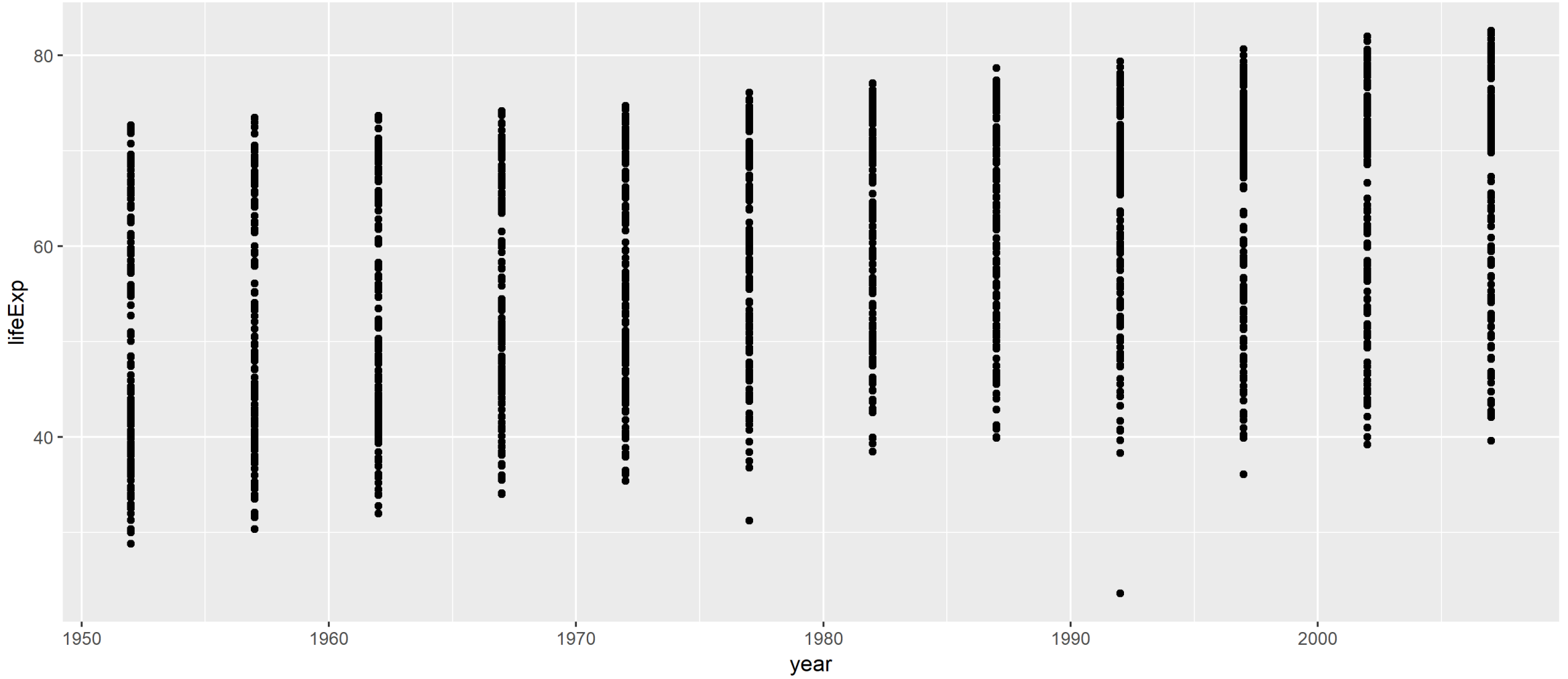
```
ggplot(gap2007) + aes(x=continent, y=lifeExp) + geom_violin() + geom_jitter()
```



#time-points

Change Across Time

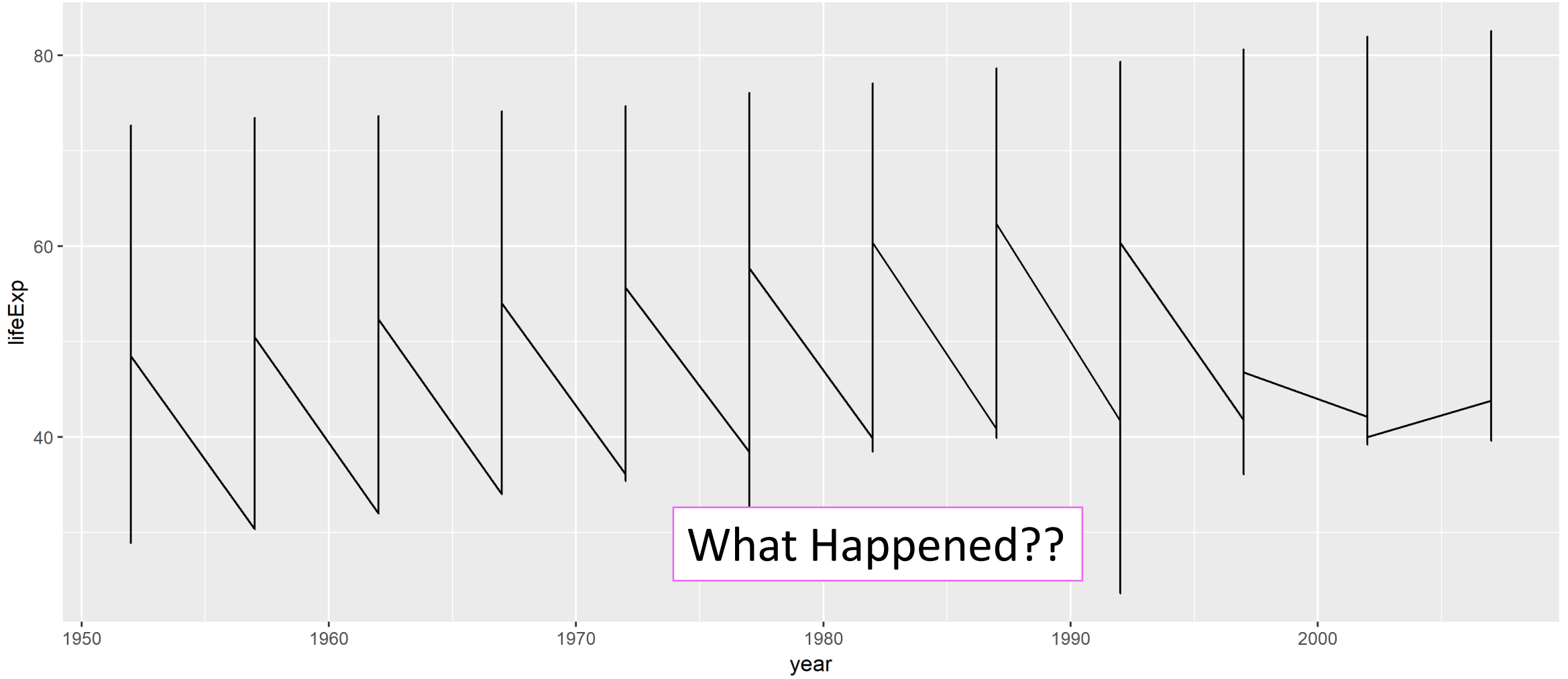
```
ggplot(gapminder) + aes(x=year, y=lifeExp) + geom_point()
```



#time-lines-bad

Change Across Time

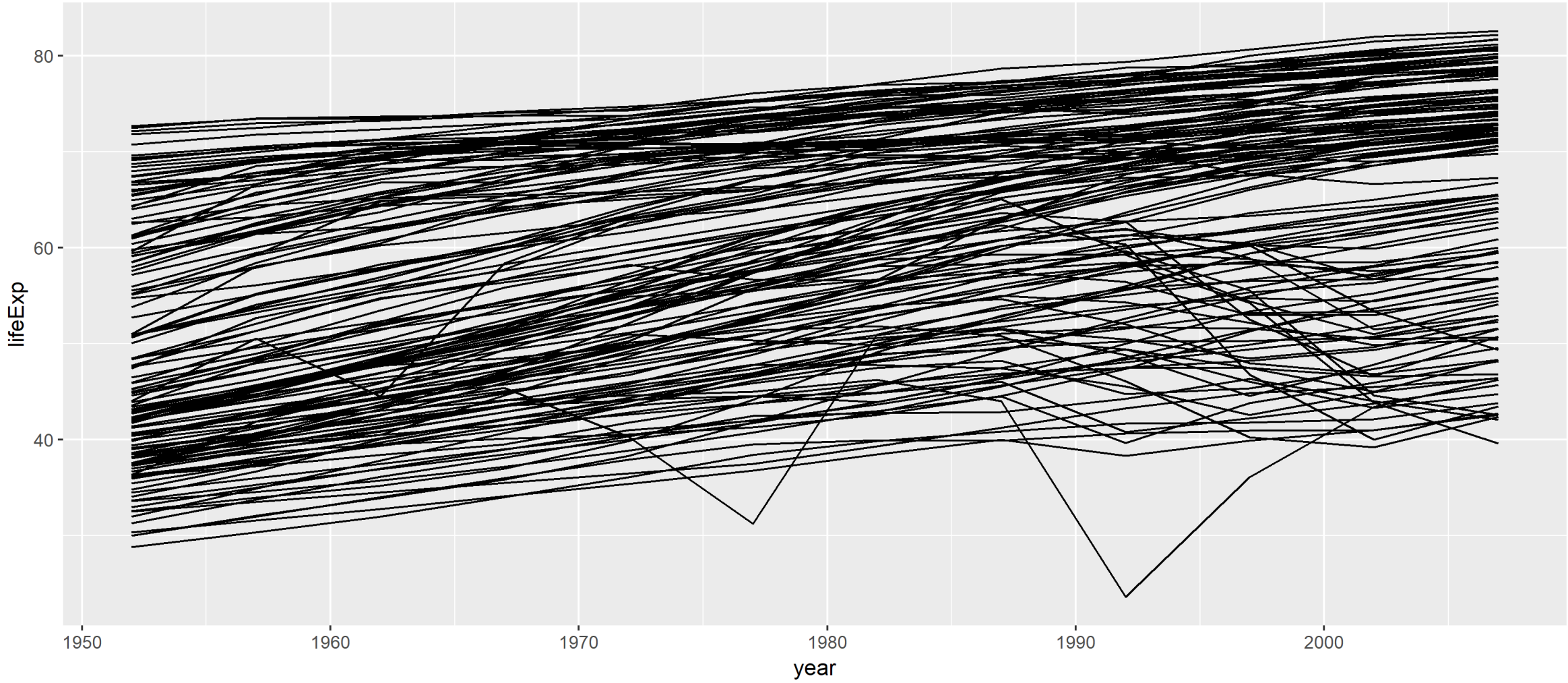
```
ggplot(gapminder) + aes(x=year, y=lifeExp) + geom_line()
```



#time-lines

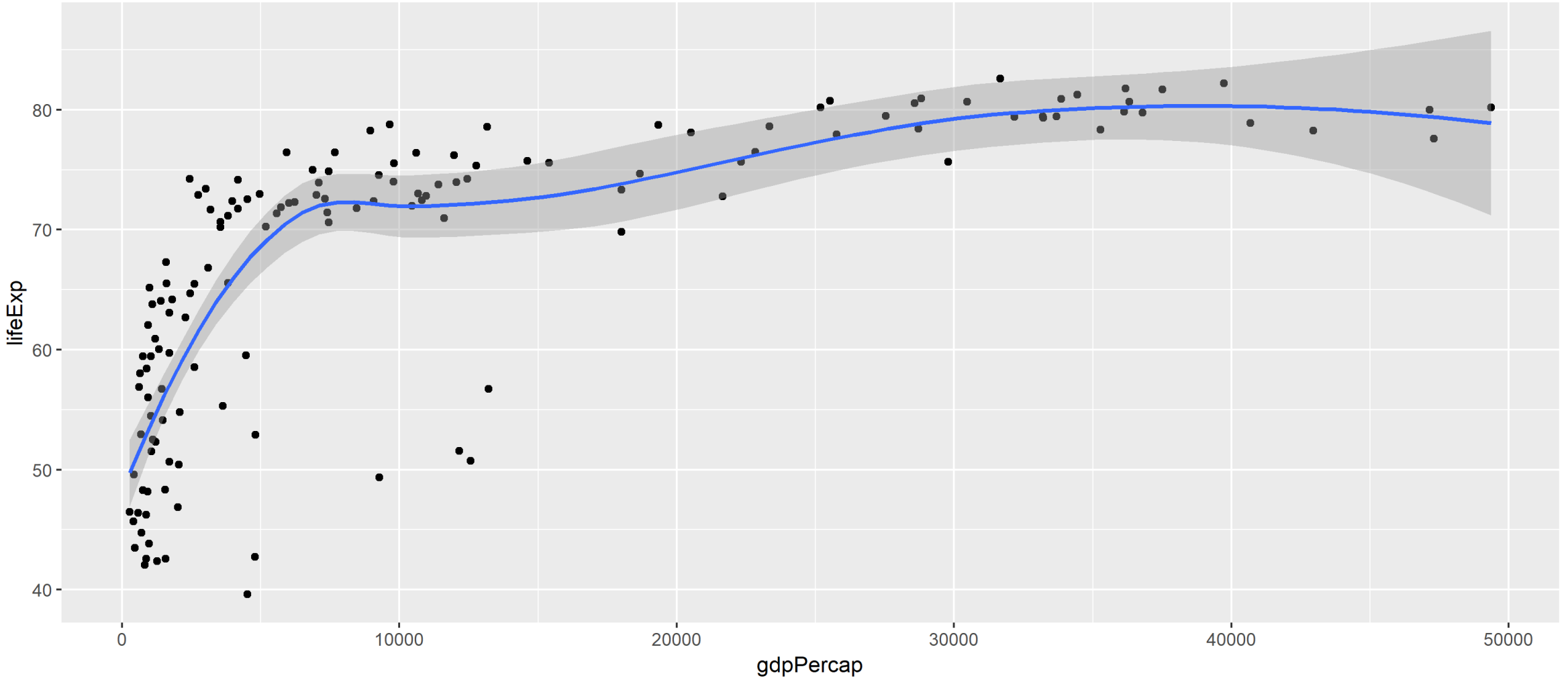
Change Across Time

```
ggplot(gapminder) + aes(x=year, y=lifeExp) + geom_line(aes(group=country))
```



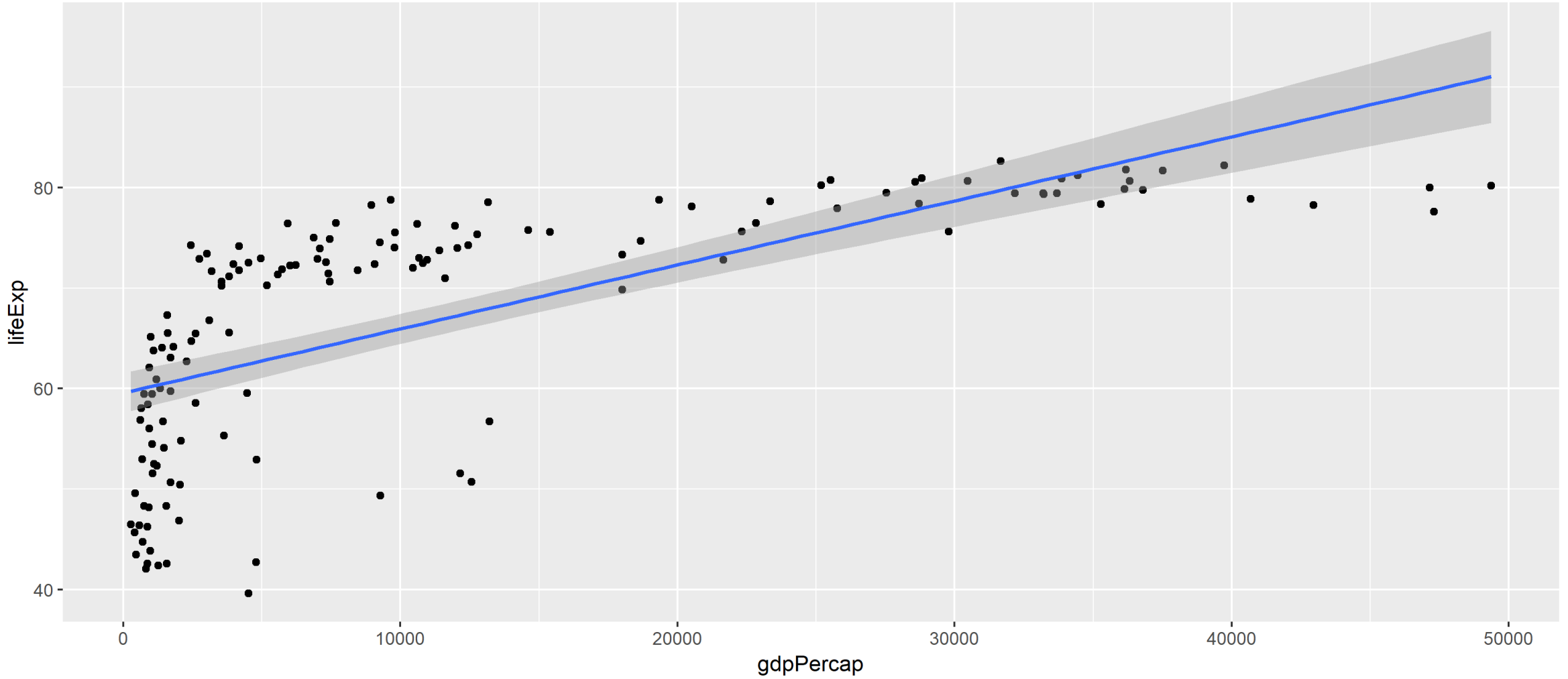
Smoothing Trends

```
ggplot(gap2007) + aes(x=gdpPercap, y=lifeExp) + geom_point() + geom_smooth()
```



Smoothing Trends

```
ggplot(gap2007) + aes(x=gdpPercap, y=lifeExp) + geom_point() + geom_smooth(method="lm")
```

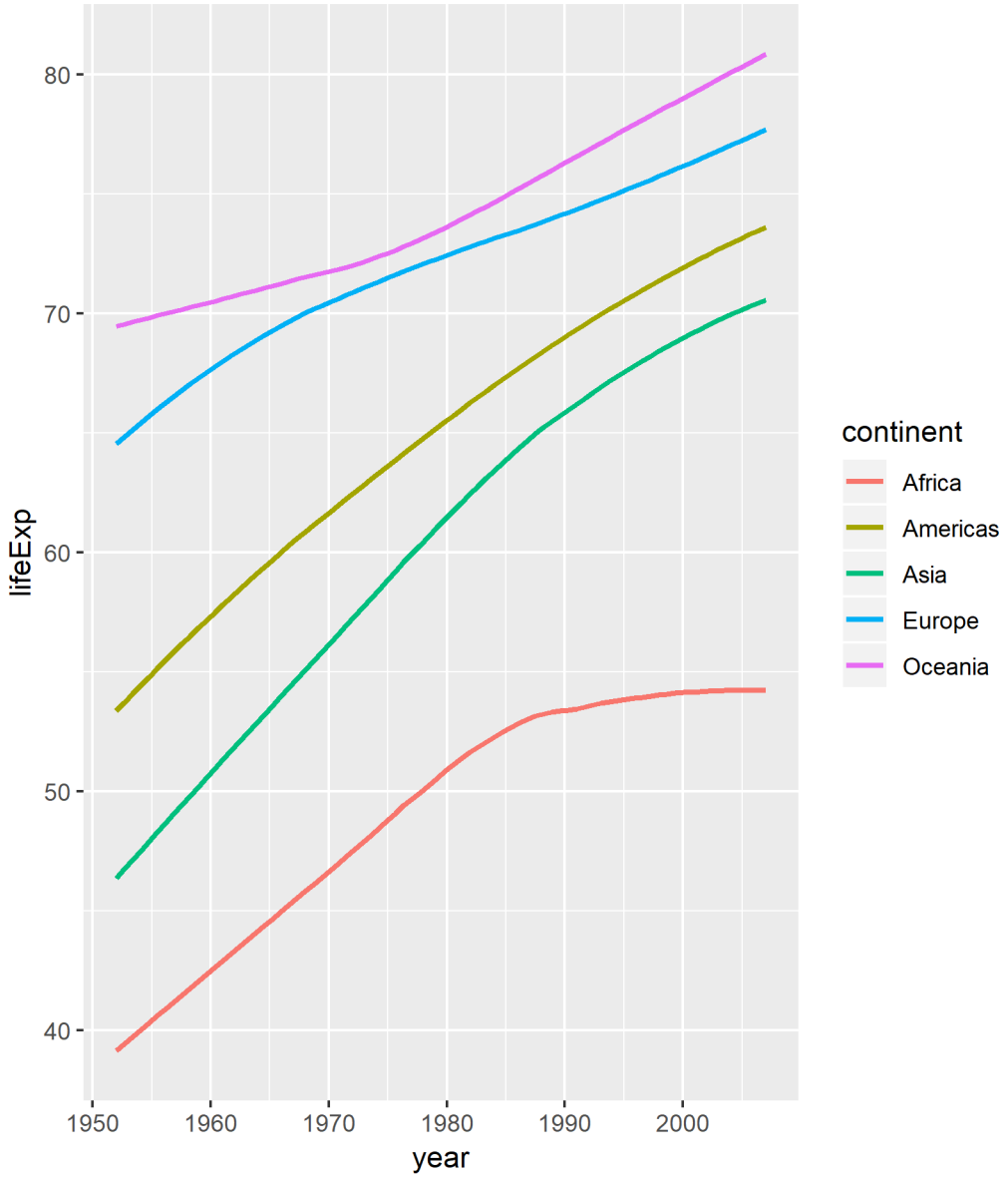


#smooth-poll

Trend per continent?

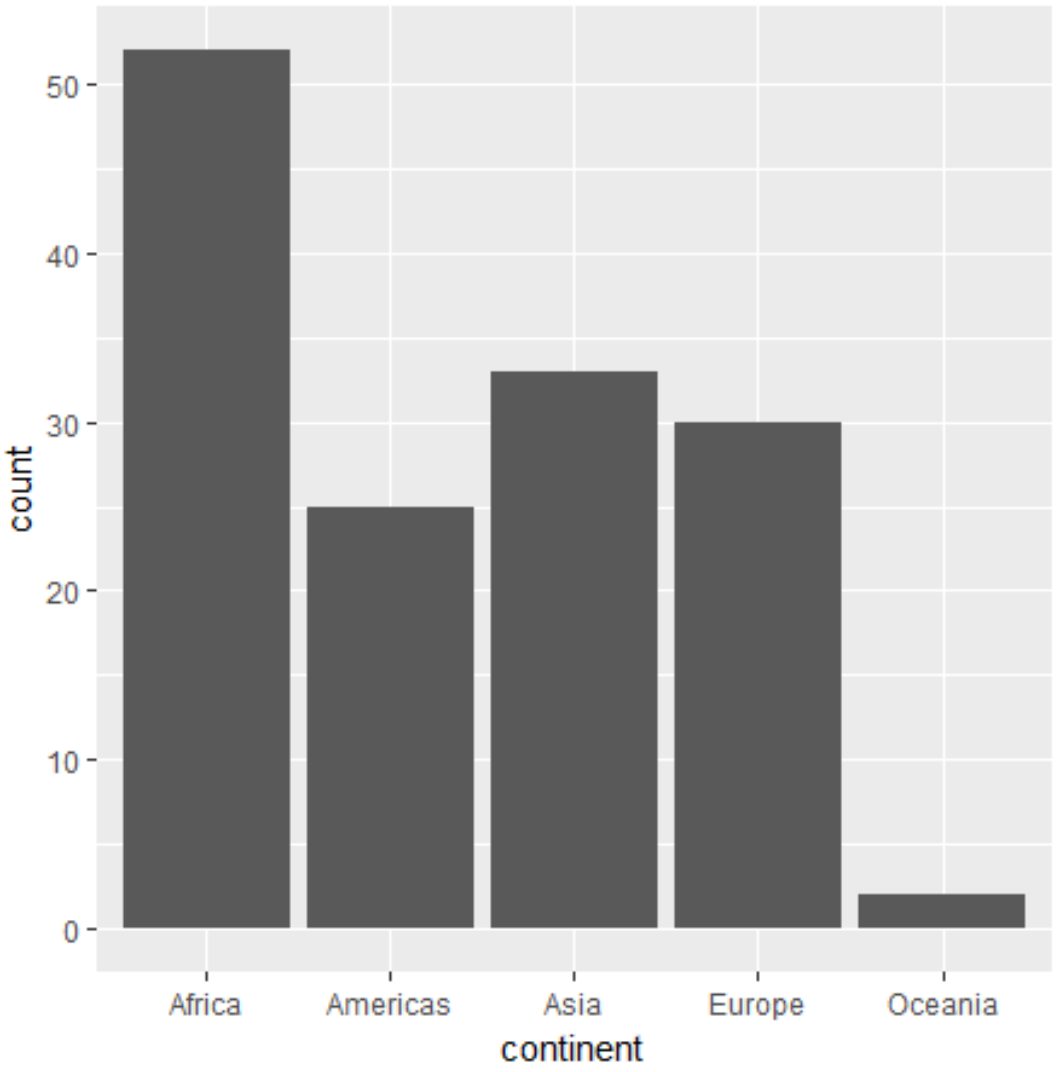
```
ggplot(gapminder) +
  aes(??) +
  geom_??(??)
```

How can we plot a trend line per continent?



Bar charts (for counts)

```
# Plot the number of countries in each
# continent

ggplot(gap2007) +
  aes(x=continent) +
  geom_bar()
```



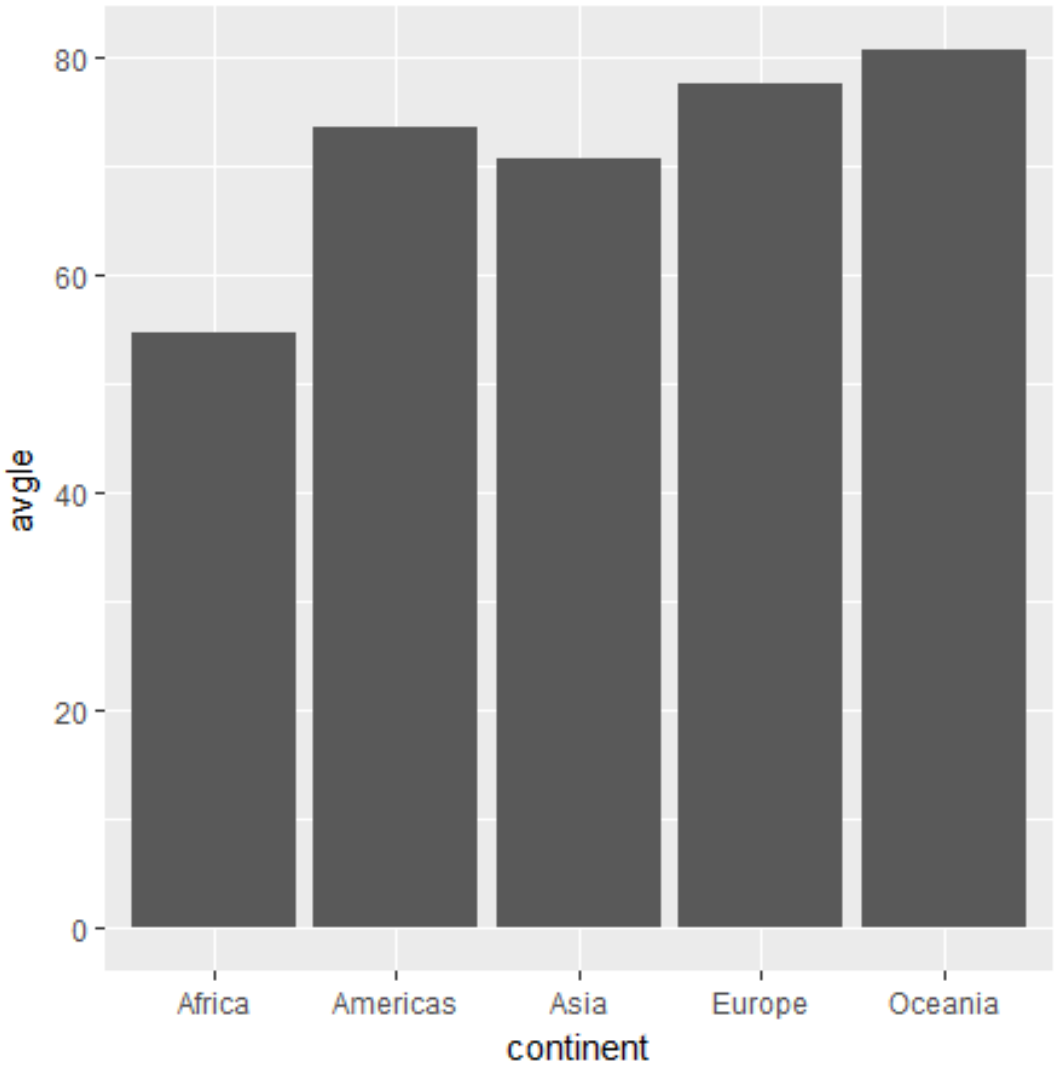
Tidy data

- ggplot works best with tidy data
- Rules of tidy data
  - Each variable in the data set is placed in its own column
  - Each observation is placed in its own row
  - Each value is placed in its own cell
- You data (usually) should be in a single data.frame (or tibble)
- You may need to summarize or transform your data prior to plotting
- Some geoms will do basic summarization for you

#bar-plot-calc

Column charts (for other values)

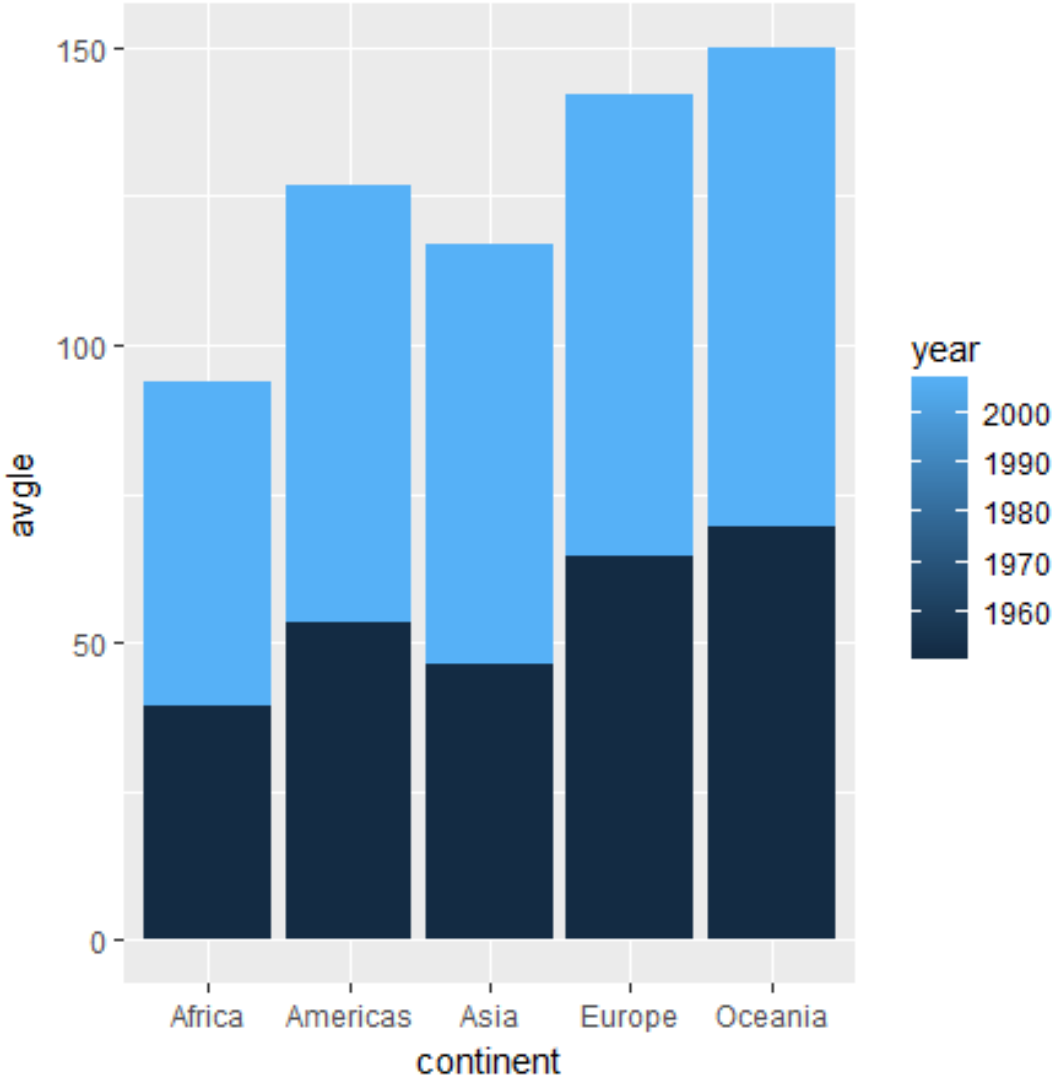
```
# Plot the average life expectancy for
each continent

gapminder %>%
  filter(year==2007) %>%
  group_by(continent) %>%
  summarize(avgle = mean(lifeExp)) %>%
  ggplot(data = .) +
  aes(x=continent, y=avgle) +
  geom_col()
```



#bar-plot-stacked-1

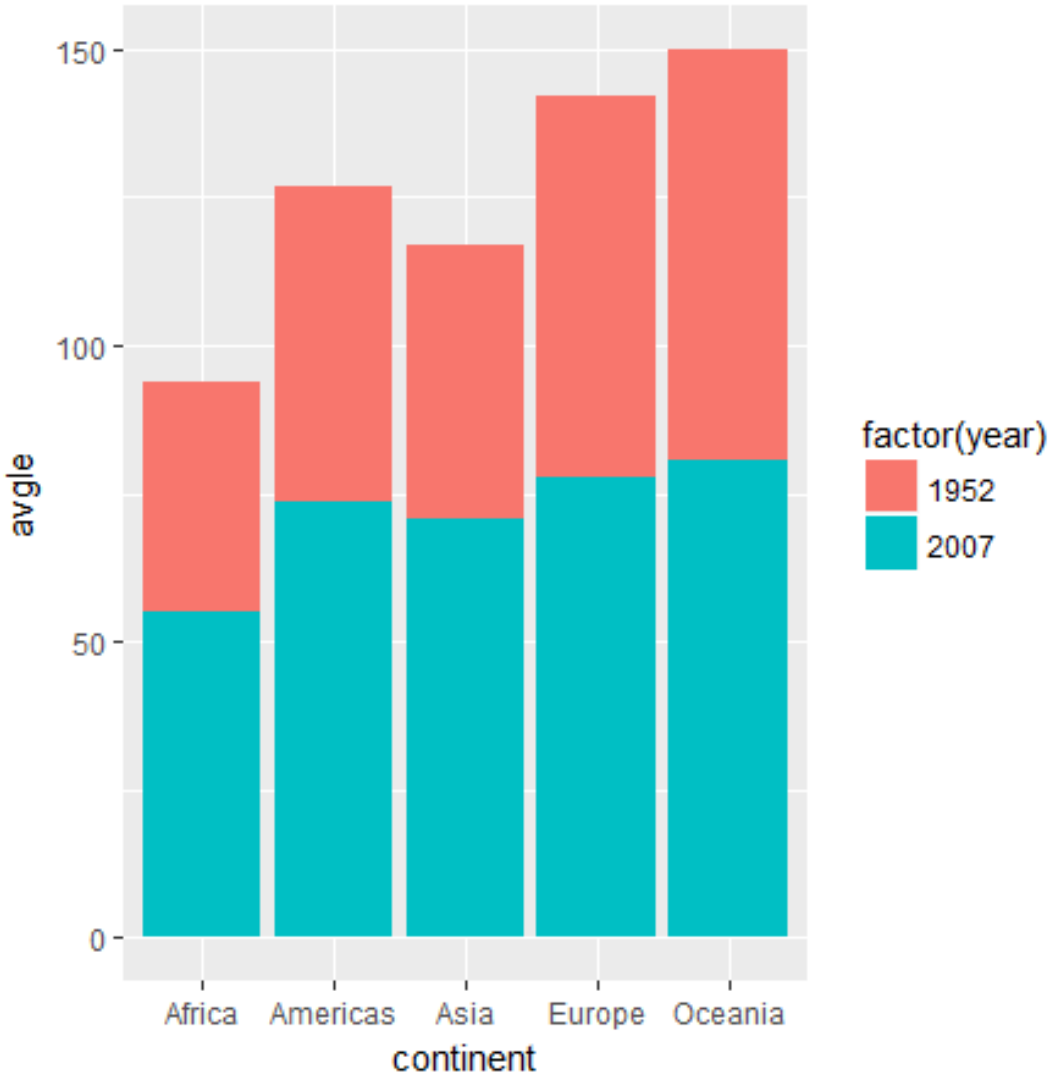
Column charts (for other values)

```
# Plot the average life expectancy for
each continent for 1952 & 2007

gapminder %>%
  filter(year==2007 | year==1952) %>%
  group_by(continent, year) %>%
  summarize(avgle = mean(lifeExp)) %>%
  ggplot(data = .) +
  aes(x=continent, y=avgle) +
  geom_col( aes(fill=year) )
```



Column charts (for other values)

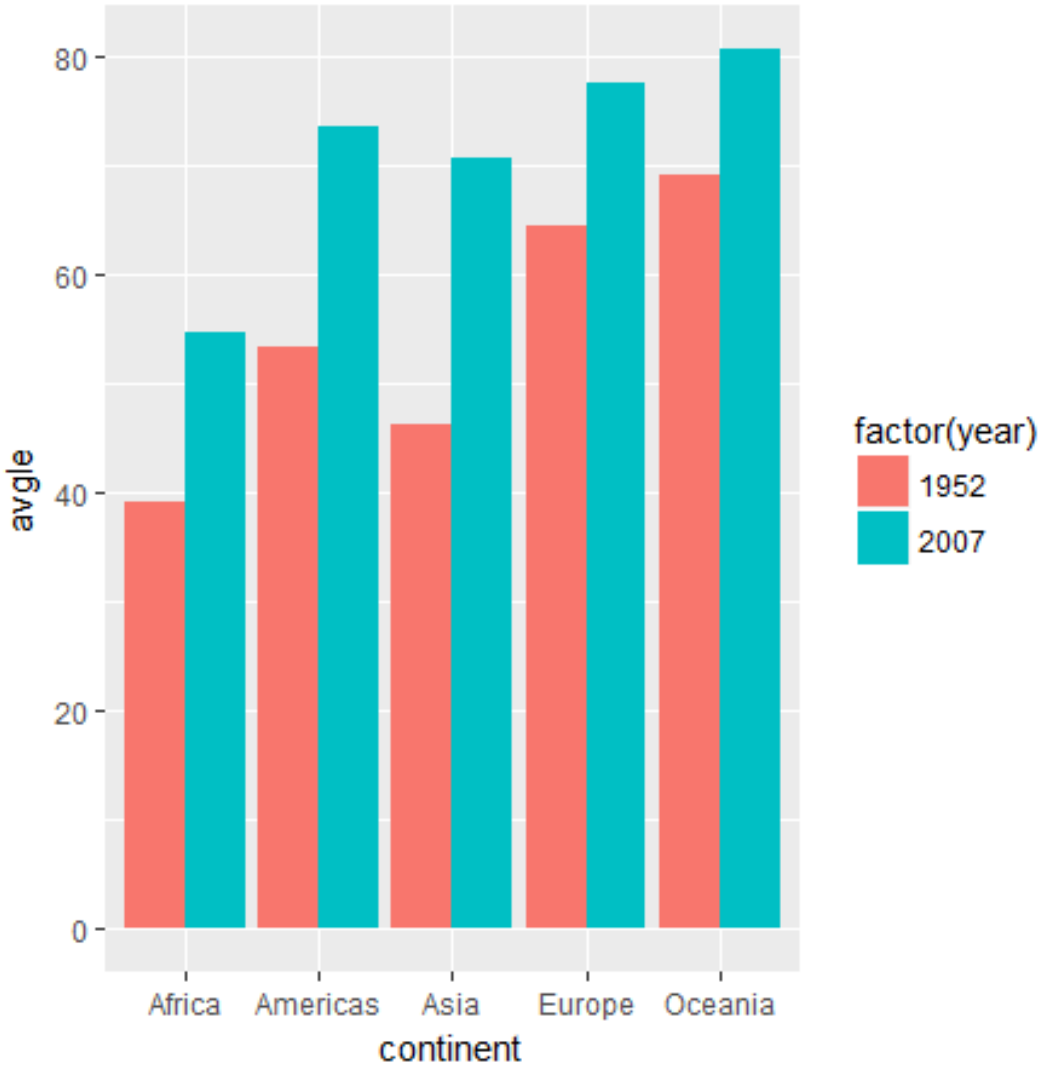
```
# Plot the average life expectancy for
each continent for 1952 & 2007

gapminder %>%
  filter(year==2007 | year==1952) %>%
  group_by(continent, year) %>%
  summarize(avgle = mean(lifeExp)) %>%
  ggplot(data = .) +
  aes(x=continent, y=avgle) +
  geom_col( aes(fill=factor(year)) )
```



#bar-plot-dodge

Column charts (for other values)

```
# Plot the average life expectancy for
each continent for 1952 & 2007

gapminder %>%
  filter(year==2007 | year==1952) %>%
  group_by(continent, year) %>%
  summarize(avgle = mean(lifeExp)) %>%
  ggplot(data = .) +
  aes(x=continent, y=avgle) +
  geom_col( aes(fill=factor(year)) ,
     position="dodge")
```

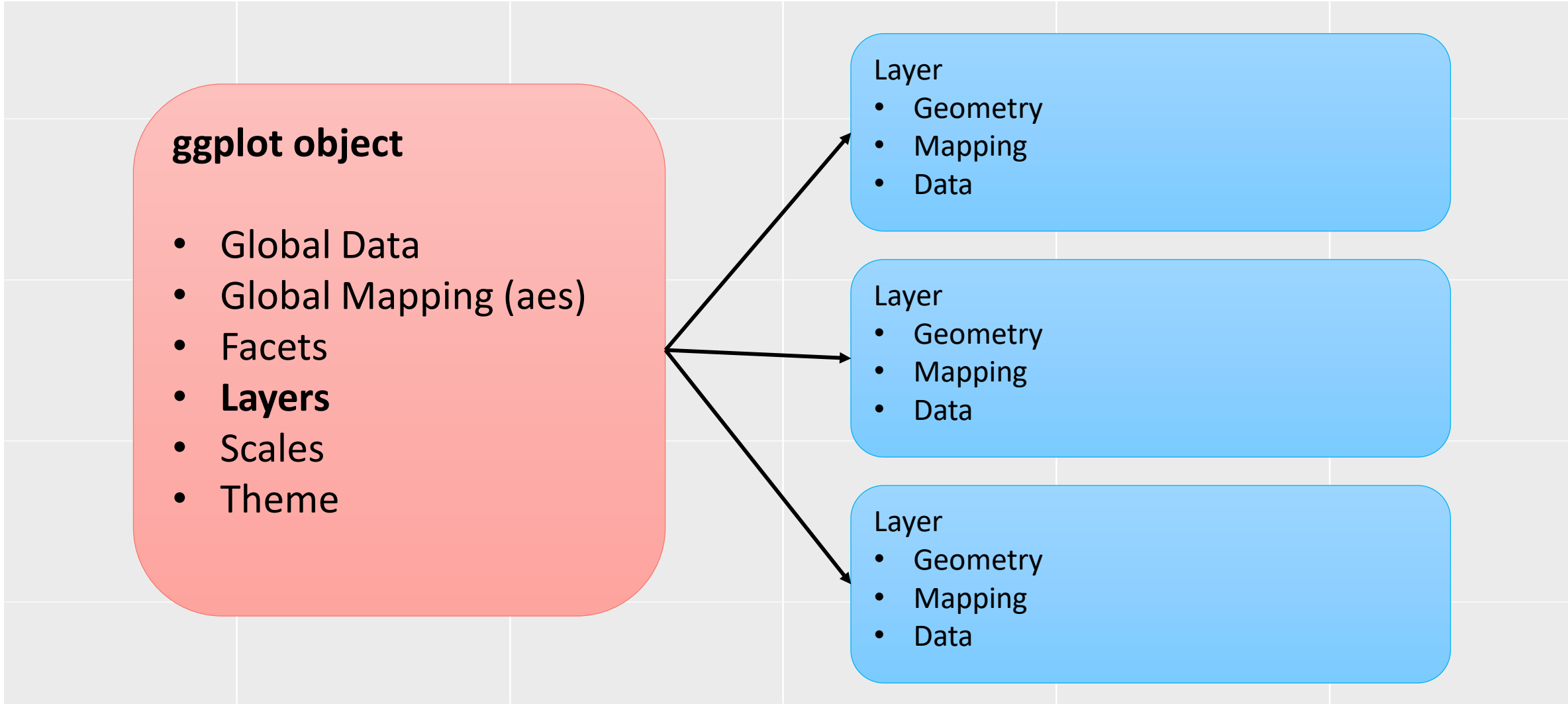


What does ggplot() do?

- The ggplot() function creates a "gg/ggplot" object
- You use (+) to add additional instructions to the object to build your plot (Note: do not use %>% to add layers)
- Can be saved to a variable
- Doesn't actually "draw" the plot, that only happen

```
p <- ggplot(data = gap2007) +
  aes(x = gdpPercap, y = lifeExp) +
  geom_point()
# nothing happens until
p
print(p)
```

Global options vs Layer options



More about aes()

- By default, layer `aes()` values are inherited from `ggplot()`
- Disable inheritance with `geom_<name>(…, inherit.aes=FALSE)`
- You may also add, override, or remove `aes()` values

global mapping	layer mapping	Resulting aesthetics	Operation
aes(x=year, y=pop)	aes(color=continent)	aes(x=year, y= pop, color=continent)	Add
aes(x=year, y=pop)	aes(y=lifeExp)	aes(x=year, y=lifeExp)	Override
aes(x=year, y=pop)	aes(y=NULL)	aes(x=year)	Remove

```
ggplot(mapping = aes(<global>)) +
  aes(<global>) +
  geom_<name>(mapping = aes(<layer>))
```

What is the final mapping?

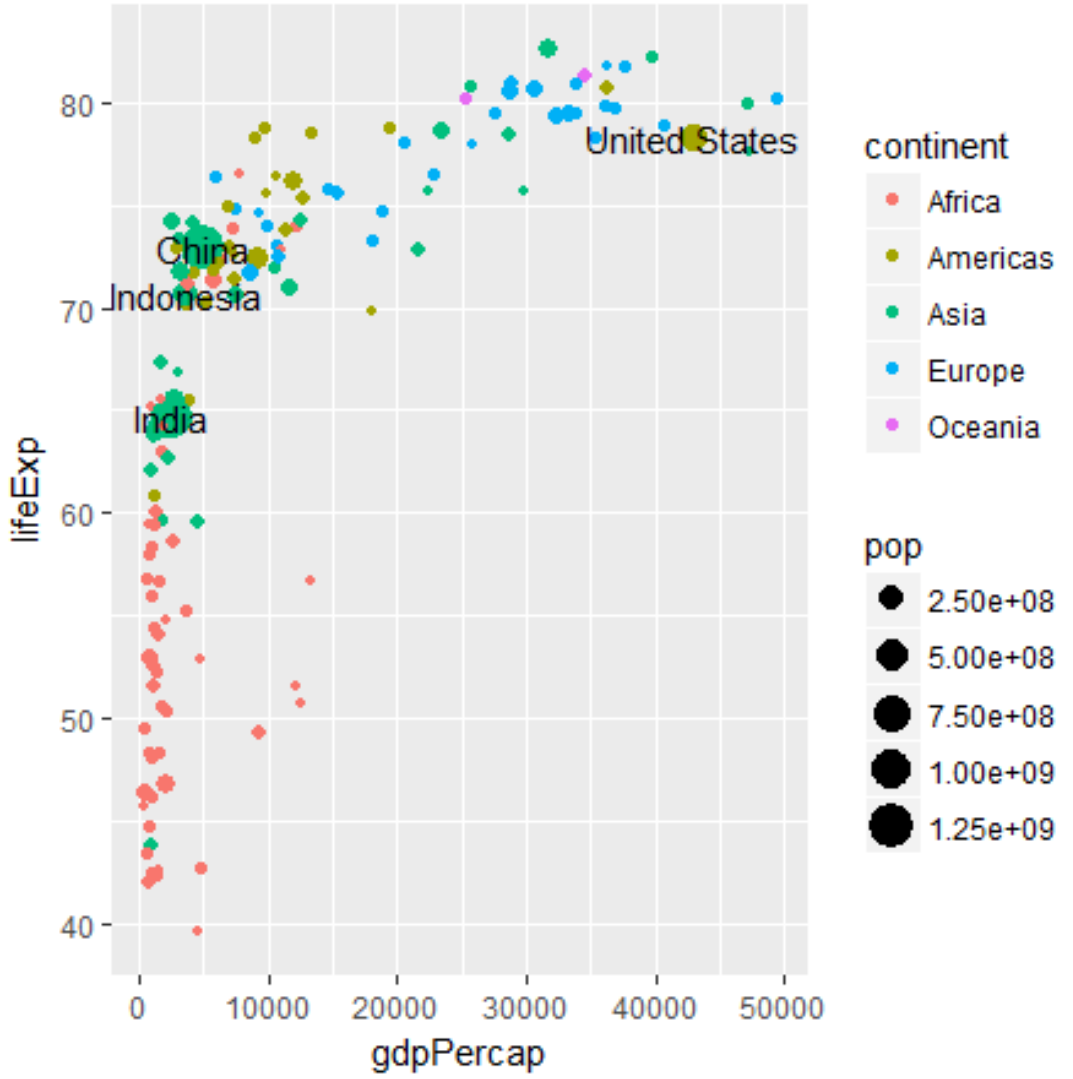
```
ggplot(gapminder, aes(x=year, y=pop)) +
  aes(y=country) +
  geom_point(mapping=aes(y=gdpPercap)) +
  aes(y=lifeExp)
```

A) `aes(x=year, y=pop)`
B) `aes(x=year, y=country)`
C) `aes(x=year, y=gdpPercap)`
D) `aes(x=year, y=lifeExp)`

Which will be the final mapping for the points?

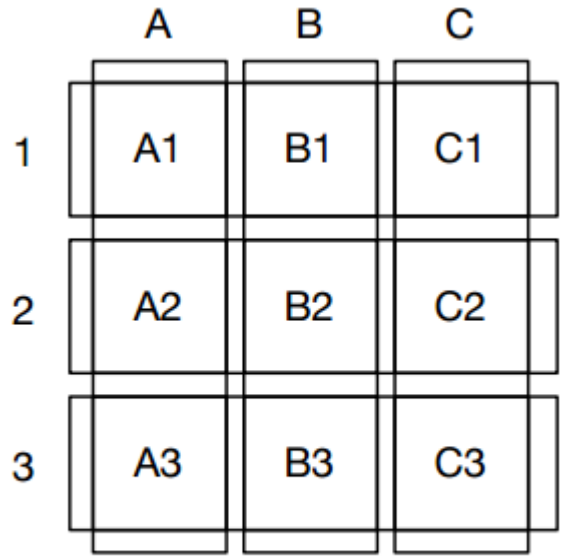
Different data for different layers

```
# Only label large populations
to_label <- gap2007 %>%
  filter(pop > 200000000)

gap2007 %>%
  ggplot(data = .) +
  aes(x=gdpPercap, y=lifeExp) +
  geom_point(aes(size=pop, color=continent)) +
  geom_text(aes(label=country), data=to_label)
```

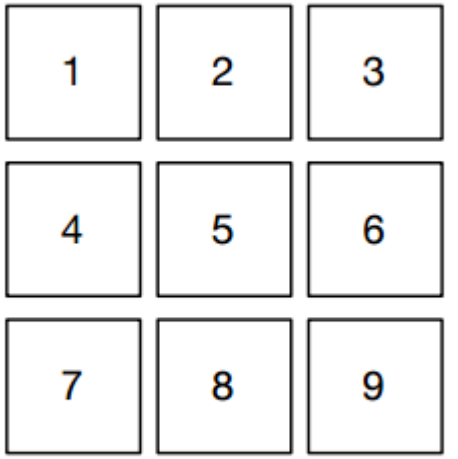


Faceting

- Divide plot into subgroups and draw layers for each set
- Two primary options
  - Grid – up to two variables: one row rows, one for cols
  - Wrap – no panel structure
- Each facet gets all layers
- Each layer's data split on same variables



facet_grid

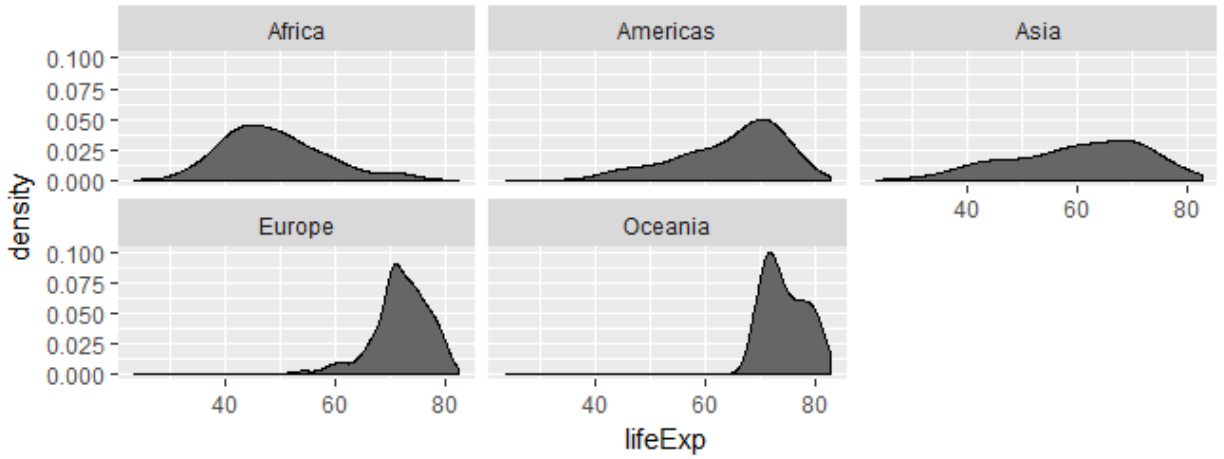


facet_wrap

facet_wrap()

- Requires a list of column names wrapped in `vars()`
- Each combination of columns gets it own panel
- scales= options
  - “fixed” scales same in all
  - “free_x” x range can vary
  - “free_y” y range can vary
  - “free” domain and range can change for each panel
- ncol= number of columns

```
ggplot(gapminder) +
  aes(x=lifeExp) +
  geom_density(fill="grey40") +
  facet_wrap(vars(continent))
```



facet_grid()

- You can specify a list of column names wrapped in `vars()` to, `rows=`, `cols=`, or both
- Share axis across panels
- Results in “rectangular” output

```
p <- gapminder %>%
  mutate(decade=year%/%10*10) %>%
  group_by(continent, decade,
    country) %>%
  select(-year) %>%
  summarize_all(mean) %>%
  ggplot() +
    aes(gdpPercap, lifeExp) +
    geom_point()

p + facet_grid(rows=vars(decade),
               cols=vars(continent))
p + facet_grid(rows=vars(decade))
p + facet_grid(cols=vars(continent))
```

Wrap vs Grid

- Grid
  - rows and columns have meaning
  - each rows/column represents a single level of a discrete variable
  - Labels are at the top of each row/column
- Wrap
  - rows and columns do not have meaning
  - Labels are at the top of each sub plot

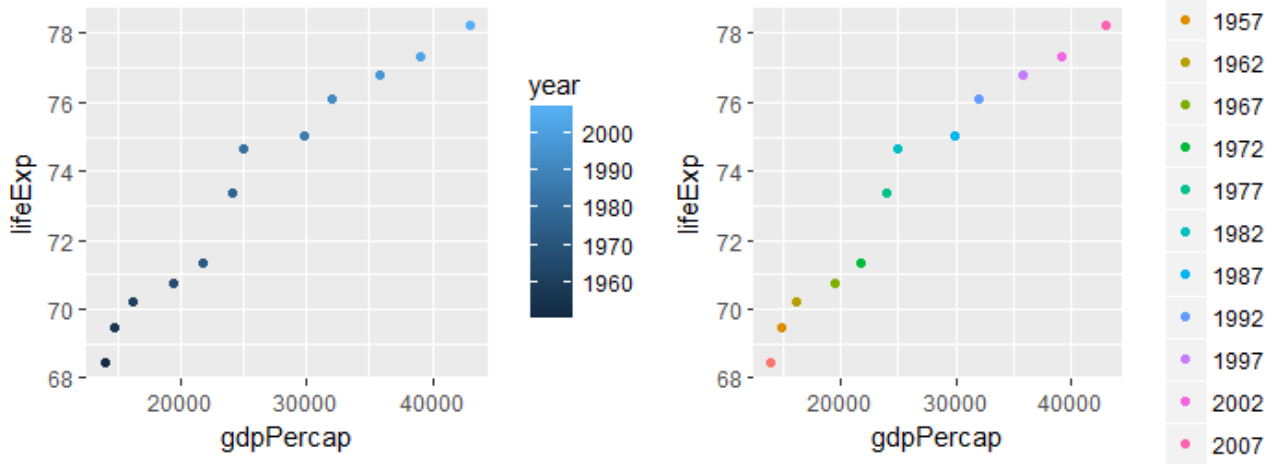
Scales

- Scales describe how raw data values should be converted to aesthetic values
- Default scales are determined by the class of the variables in your data
- Each aesthetic (eg, color, fill, size, shape) can have at most one scale
- Scales can have guides
  - Axes for positions
  - Legends for everything else

Scaling based on data type

- Color is mapped differently depending if year is a numeric or character vector
- Default color scales
  - Numeric: scale_color_continuous()
  - Factor: scale_color_discrete()

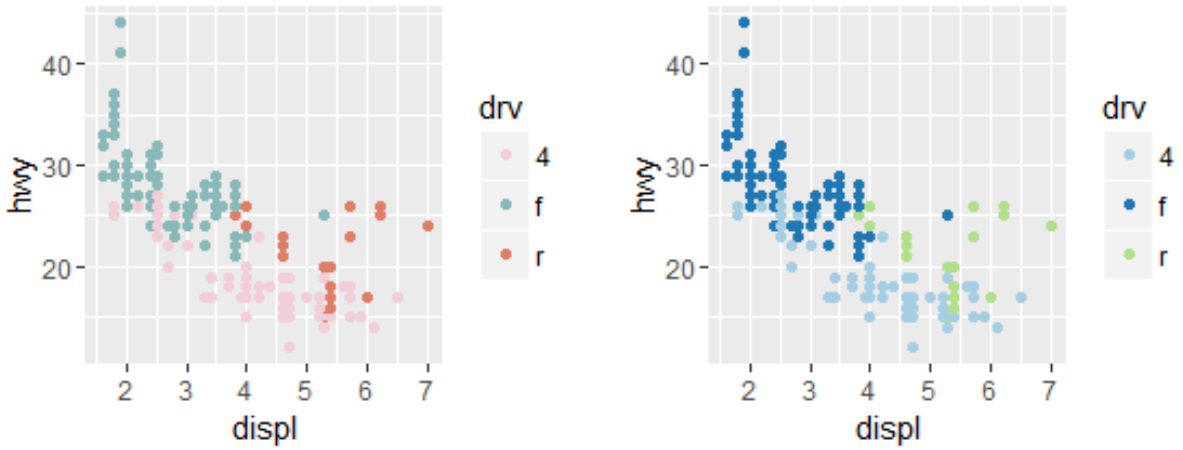
```
p <- gapminder %>%
  filter(country=="United States") %>%
  ggplot(aes(gdpPercap, lifeExp))

# Compare output
p + geom_point(aes(color=year))
p + geom_point(aes(color=factor(year)))
```



Manually setting discrete colors

- You can customize the default color scales
- Or you can create your own manual scale
  - http://colorbrewer2.org/ RColorBrewer::display.brewer.all()
  - Get RGB values from anywhere
    - http://colormind.io/
    - http://color.adobe.com
- scale_color_manual expects
  - Vector of color values=
  - Named vector of color values=
  - Vector of color values= for levels named in breaks=

```
p <- ggplot(mpg, aes(displ, hwy)) +
  geom_point(aes(color = drv))

p + scale_color_manual(values=c(
"4"="#F2CED8", "f"="#88B8B8",
"r"="#DE7E68") )

p + scale_color_brewer(palette="Paired")
```



Expanding scale_color_manual()

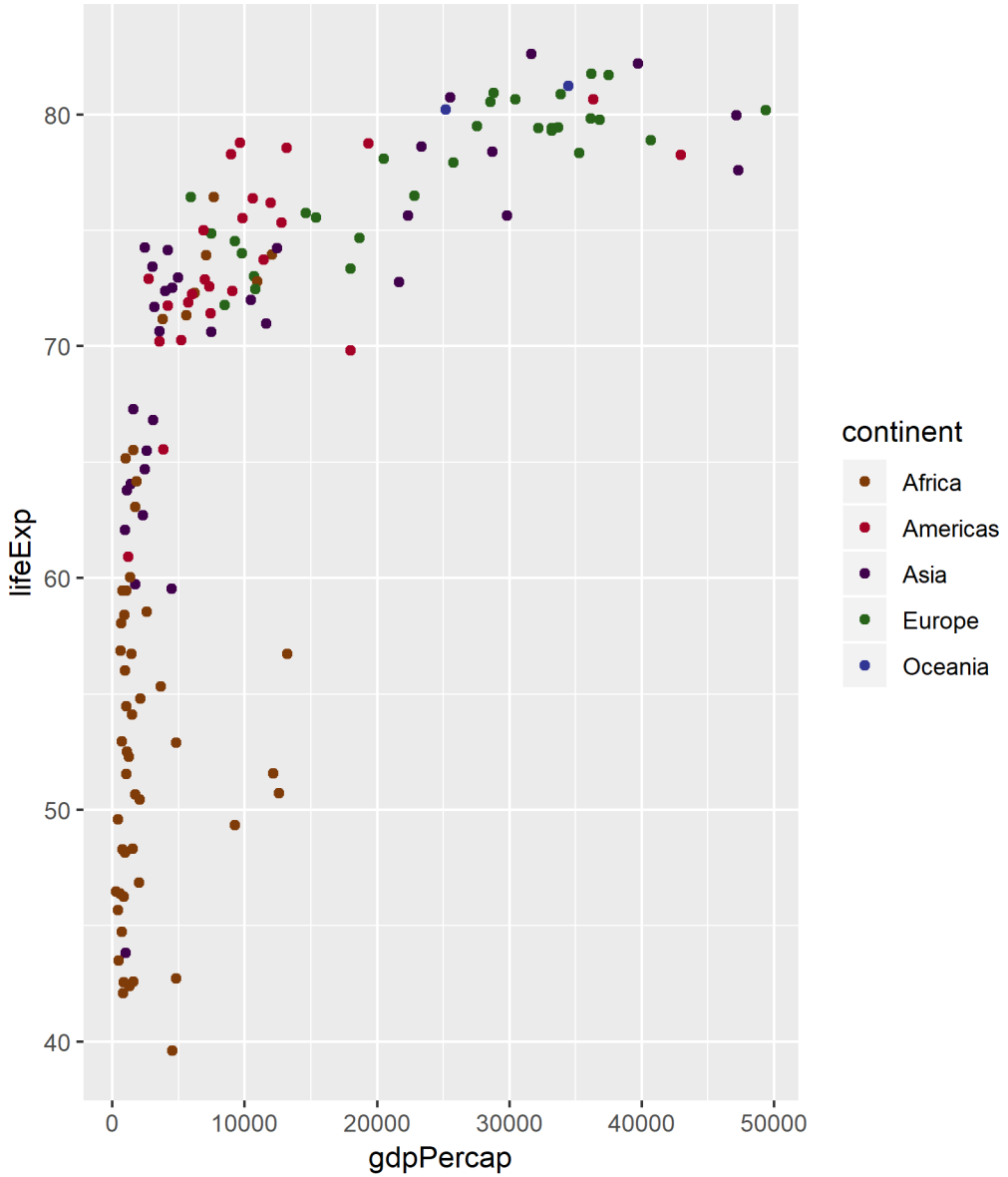
-
```
+ scale_color_manual(
    values=c(
      "4"="#F2CED8",
      "f"="#88B8B8",
      "r"="#DE7E68"))
```
-
```
+ scale_color_manual(
    values=c("#F2CED8", "#88B8B8", "#DE7E68"),
    breaks=c("4","f","r"),
    labels=c("4 wheel", "front", "rear"),
    name="Drive")
```

#scale-color-poll

Use continent colors

```
gap2007 %>%
  ggplot() +
  aes(x=gdpPercap) +
  aes(y=lifeExp) +
  aes(color = ??) +
  geom_point() +
  scale_color_??( ?? )
```

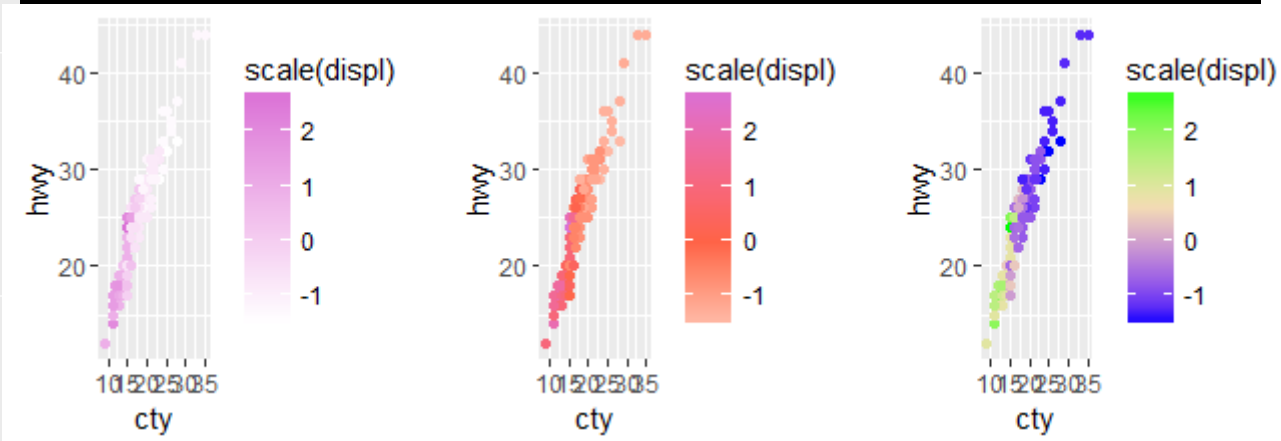
How can you use the built-in vector `continent_colors` to change the colors of the points for each continent?



Manually setting continuous colors

- Continuous values plotted with gradients
- Two color gradient
  - scale_color_gradient()
- Three color divering gradient
  - scale_color_gradient2()
- N-color gradient
  - scale_color_gradientn()
- See all color names in R
  - colors()

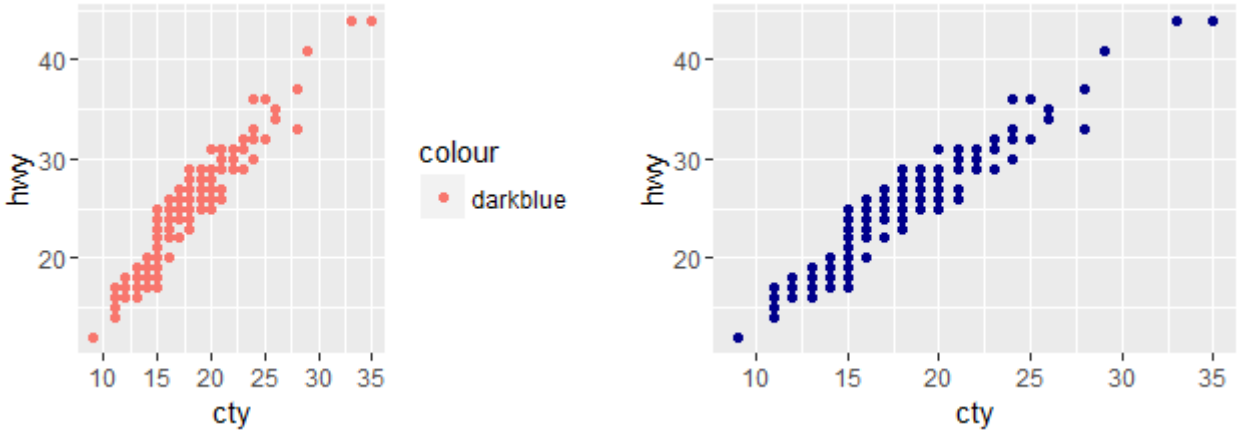
```
p <- ggplot(mpg, aes(cty, hwy)) +
  geom_point(aes(color = scale(displ)))

p + scale_color_gradient(
  low="white", high="orchid")
p + scale_color_gradient2(low="white",
  high="orchid", mid="tomato")
p + scale_color_gradientn(colors=
  c("blue","wheat","green"))
```



Setting vs mapping

- Notice the difference that color= makes inside vs outside the aes() function
- Only things inside an aes() get a legend (only the mappings)
- If you have a column that has color values, use scale_color_identity() to prevent remapping

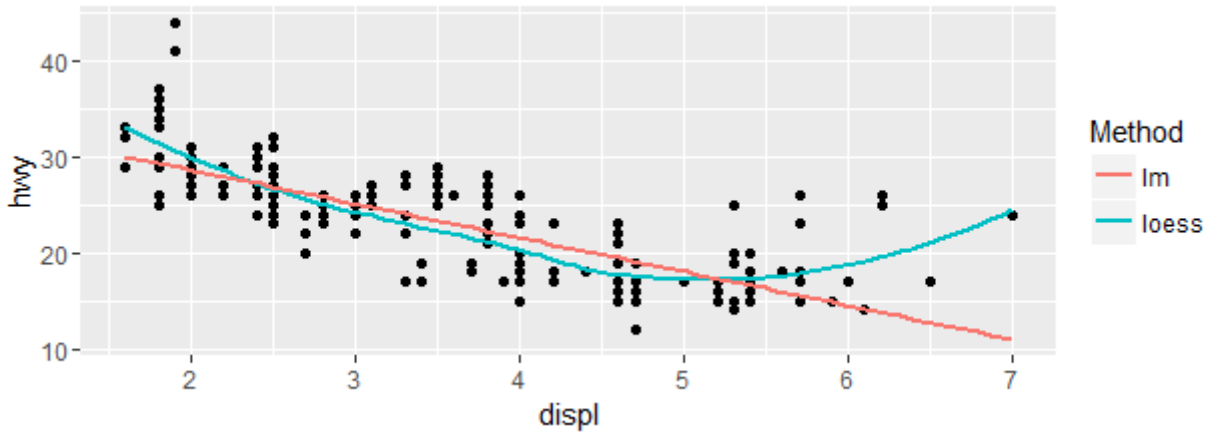
```
# "odd" behavior
ggplot(mpg, aes(cty, hwy)) +
  geom_point(aes(color = "darkblue"))

ggplot(mpg, aes(cty, hwy)) +
  geom_point(color = "darkblue")
```



Literal string mappings

- Specifying a literal mapping can be useful if using multiple layers
- Here we add two layers with different smoothers
- We specify a color= in the aes() so we get a nice legend

```
# mapping a literal value
ggplot(mpg, aes(displ, hwy)) +
  geom_point() +
  geom_smooth(aes(color = "loess"),
    method = "loess", se = FALSE) +
  geom_smooth(aes(color = "lm"),
    method = "lm", se = FALSE) +
  labs(color = "Method")
```



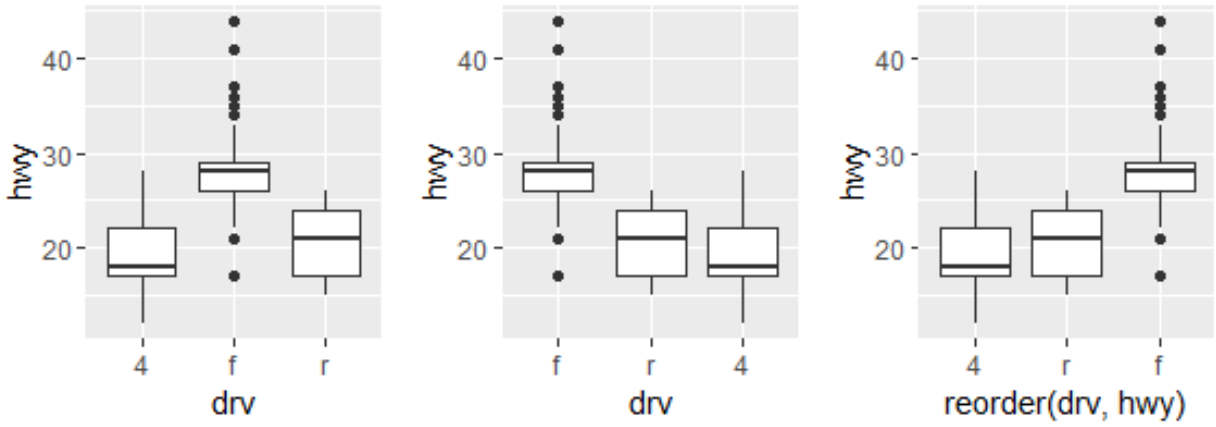
Axes are just scales as well

- You can change transformations of x/y axes via scales
  - scale_x_log10()
  - scale_x_sqrt()
  - scale_x_reverse()
- Can also take finer control over tick marks
  - scale_x_continuous() – numeric values
  - scale_x_datetime() – date/time values
- Control display of factor levels
  - scale_x_discrete()
  - Choose new labels for factor levels

Discrete axes plotting order

- Discrete axes are drawn in the order of the levels() of the corresponding factor.
- You can change that order by changing the axes scale
- Or you can re-order the factor itself (see "?reorder")

```
p <- ggplot(mpg, aes(y=hwy))

# default
p + geom_boxplot(aes(drv))
# use scale
p + geom_boxplot(aes(drv)) +
 scale_x_discrete(limits=c("f","r","4"))
# use data
p + geom_boxplot(aes(reorder(drv, hwy)))
```



Labeling your plot and axes

- You can label your x and y axes
  -
```
+ labs(
    x="X Name",
    y="Y Name",
    title="Plot Title"
  )
```
- You can also include mathematical expressions (see "?plotmath")
  - `+ labs(title=expression(y==alpha+beta*x))`
- Setting values to "" shows no label, setting values to NULL removes space for label as well

```
ggplot(mpg, aes(cyl)) +
geom_bar() +
labs(
  title=expression(y==alpha+beta*x),
  x="Cylinder", y="Count")
```

Formatting your tick labels

- Format 0-1 as percents
  - + scale_y_continuous(labels = scales::percent_format())
- Format as dollar amounts
  - + scale_y_continuous(labels = scales::dollar_format("$"))
- Format in thousands
  - + scale_fill_continuous(labels = scales::unit_format("k", 1e-3))
- You can pass any function as the labels= argument

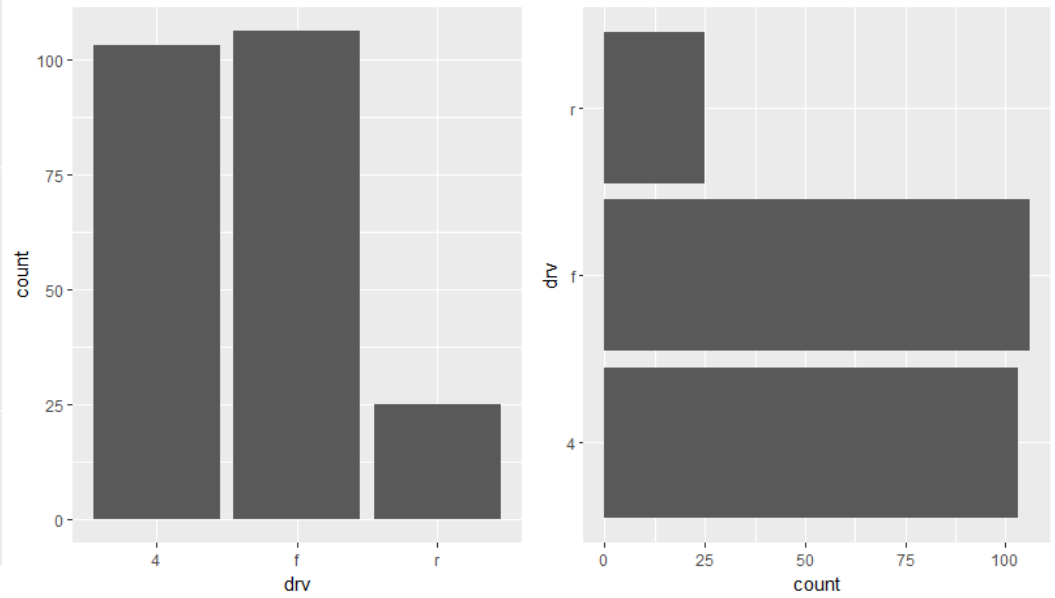
Formatting your tick marks

- You can set where your tick marks fall with scale_x_continuous
  - breaks= vector of values where to draw major tick marks
  - labels = vector of values with what to draw at those tick marks
  - minor_breaks= vector of values where to draw minor (unlabeled) tick marks
  - trans= optional transformation to apply to axis
  - expand= how far to extend axis past observed data
  - limits= lower and upper bound for tick marks
- For datetime axes
  - date_minor_breaks= units like "1 month" or "2 years"

Coordinate Transformations

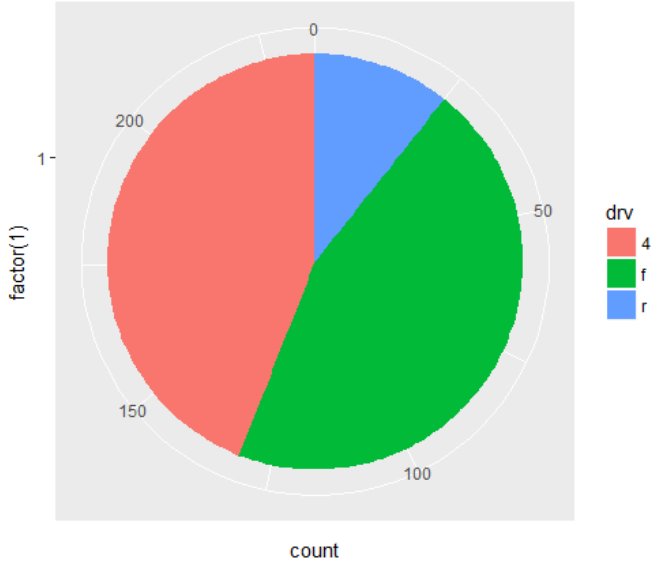
coord_flip(): swap x & y axes

```
p <- ggplot(mpg, aes(drv)) +
    geom_bar()
p
P + coord_flip()
```



coord_polar(): make "pie" charts

```
ggplot(mpg) +
    geom_bar(aes(factor(1),
        fill=drv),width=1) +
    coord_polar(theta="y")
```



Coordinate Transformations

- coord_cartesian(): limit the plotting window
  - xlim= range of x values c(lower, upper)
  - ylim= range of y values c(lower, upper)
  - Differs from changing limits on scales which will subset data
- coord_fixed(): fix the distance ratio for x and y

Setting a theme

- The overall "look" of a plot is set by the theme
- Just call one of the theme functions to see all the values you can customize
- You can create your own theme objects
- The “ggthemes” package has additional themes to try out

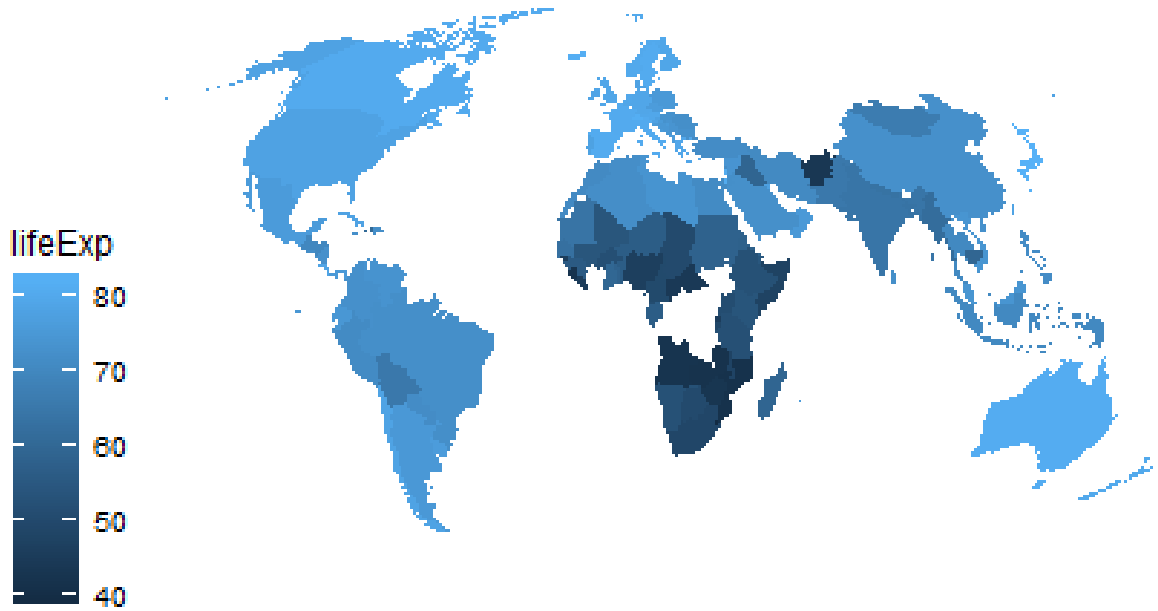
```
p <- ggplot(mpg, aes(cty, hwy, color =
factor(cyl))) +
  geom_jitter() +
  geom_abline(color="grey50", size=2) +
  ggtitle("My Plot!")

#default
p + theme_grey()
# try these
p + theme_bw()
p + theme_linedraw()
p + theme_light()
p + theme_dark()
p + theme_minimal()
```

Customizing a theme

- **Increase font size for presentation slides**
  - `p + theme_grey(base_size=18)`
- You can customize parts of themes
  - `p + theme(plot.title = element_text(color="red", margin=margin(t=20, b=20)))`
  - `p + theme(panel.background = element_rect(fill = "linen"))`
- Set default theme for session
  - `theme_set(theme(…))`
- Read the ggplot2 book or the “theme” help page on the ggplot2 website for more info

Does anybody have a map?

```
mapdata <- map_data("world") %>%
  mutate(region = recode(region,
   USA="United States",
   UK="United Kingdom"))

gap2007 %>%
  ggplot() +
  geom_map(aes(map_id=country,
fill=lifeExp), map=mapdata) +
  expand_limits(x = mapdata$long, y =
mapdata$lat) +
  coord_map(projection = "mollweide",
xlim = c(-180, 180)) +
  ggthemes::theme_map()
```

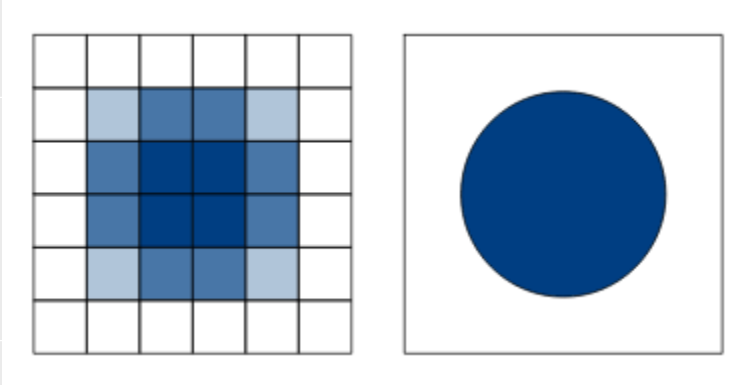


ggsave()

- Once you've made your masterpiece, use ggsave() to save it
- It will create a file in your current working directory (getwd()/setwd())
- Saves last plot printed
- Looks at file name to determine type
  - ```
    ggsave("plot.pdf"); ggsave("plot.eps");
    ggsave("plot.png"); ggsave("plot.jpg");
    ggsave("plot.tiff"); ggsave("plot.svg")
    ```
- Can also pass plot object to save any plot
  - ```
    ggsave("plot.png",
      ggplot(mpg, aes(cty,hwy))+geom_point())
    ```

Vector vs raster

- Two main image format categories
- Vector images
  - Remembers the shapes drawn
  - Infinitely zoom-able
  - pdf, svg, eps,
- Raster/bitmap images
  - Remembers just the pixels of the image
  - Number of pixels depends on the dots per inch (DPI) of your image
- Typically vector is better, but if you have lots of points, raster may be easier to work with



Programming with aesthetics

- The aes() function requires symbols, not variables
- aes_string() allows you to specify columns as characters
- aes() can expand quosures
- Best descriptions of Hadley's POV on this is: vignette("programming", package="dplyr")

```
# won't work
f <- function(y) {
  ggplot(mpg, aes(cty, y)) +
  geom_point()
}
f(hwy)
# works
g <- function(y) {
  ggplot(mpg, aes_string("cty", y)) +
  geom_point()
}
g("hwy")
h <- function(y) {
  ggplot(mpg, aes(cty, !!enquo(y))) +
  geom_point()}
h(hwy)
```

q()