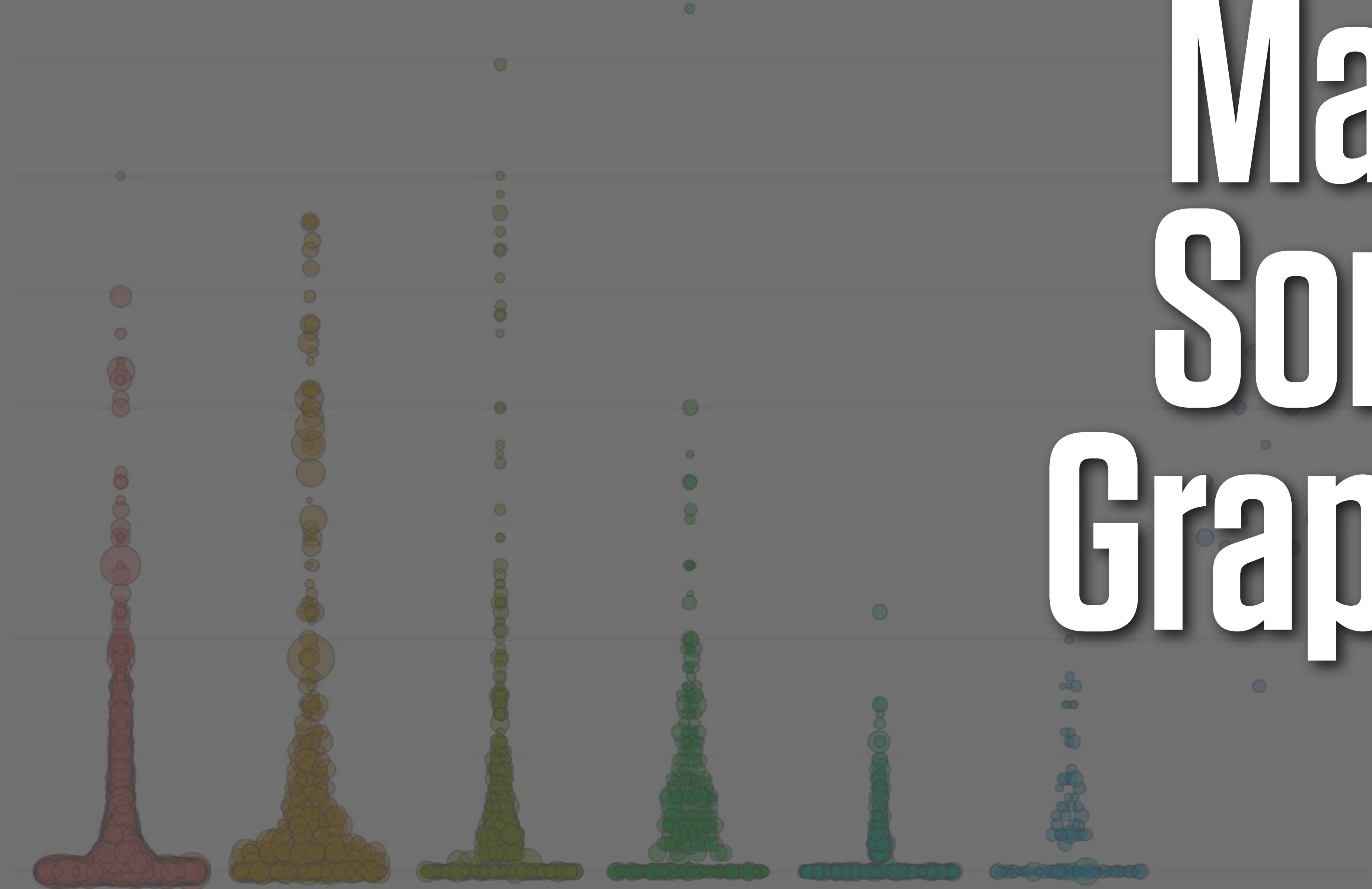


Make Some Graphs





ggplot wants you
to feed it **TIDY DATA**

FEED ME

gdp	lifexp	pop	continent
340	65	31	Euro
227	51	200	Amer
909	81	80	Euro
126	40	20	Asia

country	year	cases	population
Afghanistan	1999	745	10357071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	21166	128028583

variables

Grolemund & Wickham (2017)

country	year	cases	population
Afghanistan	1995	740	19507000
Afghanistan	2000	2000	20000000
Burkina Faso	1995	57767	17200000
Burkina Faso	2000	86400	17400100
China	1995	212200	127291024
China	2000	215700	128042030

observations

country	year	cases	population
Afghanistan	1990	745	19981071
Afghanistan	2000	2606	20599360
Brasil	1990	37737	172000362
Brasil	2000	80488	174504898
China	1990	212258	1272919272
China	2000	213766	1280420583

values

Table A-1. Years of School Completed by People 25 Years and Over, by Age and Sex: Selected Years 1940 to 2016

(Numbers in thousands. Noninstitutionalized population except where otherwise specified.)

Age, sex, and years	Total	Years of School Completed						
		Elementary		High school		College		Median
		0 to 4 years	5 to 8 years	1 to 3 years	4 years	1 to 3 years	4 years or more	

25 YEARS AND OLDER

Male

2016	103,372	1,183	3,513	7,144	30,780	26,468	34,283	(NA)
2015	101,887	1,243	3,669	7,278	30,997	25,778	32,923	(NA)
2014	100,592	1,184	3,761	7,403	30,718	25,430	32,095	(NA)
2013	99,305	1,127	3,836	7,314	30,014	25,283	31,731	(NA)
2012	98,119	1,237	3,879	7,388	30,216	24,632	30,766	(NA)
2011	97,220	1,234	3,883	7,443	30,370	24,319	29,971	(NA)
2010	96,325	1,279	3,931	7,705	30,682	23,570	29,158	(NA)
2009	95,518	1,372	4,027	7,754	30,025	23,634	28,706	(NA)
2008	94,470	1,310	4,136	7,853	29,491	23,247	28,433	(NA)
2007	93,421	1,458	4,249	8,294	29,604	22,219	27,596	(NA)
2006	92,233	1,472	4,395	7,940	29,380	22,136	26,910	(NA)
2005	90,899	1,505	4,402	7,787	29,151	21,794	26,259	(NA)



readxl part of the tidyverse

library(readxl)

State																	
A	B	C	D	E	F	G	H	I	J	K	L	M	N	P	Q		
State	CD#	2018 Cook PVI Score	2018 Winner	Party	Dem Votes	GOP Votes	Other Votes	Dem %	GOP %	Other %	Dem Margin	2016 Clinton Margin	Swing vs. 2016 Prez	Raw Votes vs. 2016	Final?		
New House Breakdown: 235D, 199R, 1 Not Certified				D	60,619,428	50,896,244	1,978,795	53.4%	44.8%	1.7%	8.6%	2.1%	6.5%	83.3%			
Compiled by: David Wasserman & Ally Flinn, Cook Political Report. @Redistrict/@CookPolitical. <i>Italics</i> denotes freshman, Bold denotes party change.																	
Alabama	1	R+15	Bradley Byrne	R	89,226	153,228	163	36.8%	63.2%	0.1%	-26.4%	-29.2%	2.8%	79.3%	x		
Alabama	2	R+16	Martha Roby	R	86,931	138,879	420	38.4%	61.4%	0.2%	-23.0%	-31.7%	8.7%	78.7%	x		
Alabama	3	R+16	Mike Rogers	R	83,996	147,770	149	36.2%	63.7%	0.1%	-27.5%	-33.0%	5.5%	79.6%	x		
Alabama	4	R+30	Robert Aderholt	R	46,492	184,255	222	20.1%	79.8%	0.1%	-59.6%	-62.5%	2.9%	78.9%	x		
Alabama	5	R+18	Mo Brooks	R	101,388	159,063	222	38.9%	61.0%	0.1%	-22.1%	-32.9%	10.8%	82.8%	x		
Alabama	6	R+26	Gary Palmer	R	85,644	192,542	142	30.8%	69.2%	0.1%	-38.4%	-43.8%	5.4%	82.8%	x		
Alabama	7	D+20	Terri Sewell	D	185,010	0	4,153	97.8%	0.0%	2.2%	97.8%	41.2%	N/A	64.2%	x		
Alaska	AL	R+9	Don Young	R	131,199	149,779	1,188	46.5%	53.1%	0.4%	-6.6%	-14.7%	8.1%	88.6%	x		
Arizona	1	R+2	Tom O'Halleran	D	143,240	122,784	65	53.8%	46.1%	0.0%	7.7%	-1.1%	8.8%	92.0%	x		
Arizona	2	R+1	<i>Ann Kirkpatrick</i>	D	161,000	133,102	50	54.7%	45.2%	0.0%	9.5%	4.8%	4.7%	91.5%	x		
Arizona	3	D+13	Raul Grijalva	D	114,650	64,868	0	63.9%	36.1%	0.0%	27.7%	29.5%	-1.8%	84.8%	x		
Arizona	4	R+21	Paul Gosar	R	84,521	188,842	3,672	30.5%	68.2%	1.3%	-37.7%	-39.4%	1.7%	91.1%	x		
Arizona	5	R+15	Andy Biggs	R	127,027	186,037	0	40.6%	59.4%	0.0%	-18.8%	-20.5%	1.7%	91.7%	x		
Arizona	6	R+9	David Schweikert	R	140,559	173,140	0	44.8%	55.2%	0.0%	-10.4%	-9.8%	-0.6%	91.2%	x		
Arizona	7	D+23	Ruben Gallego	D	113,044	301	18,706	85.6%	0.2%	14.2%	85.4%	48.3%	N/A	79.0%	x		
Arizona	8	R+13	Debbie Lesko	R	135,569	168,835	13	44.5%	55.5%	0.0%	-10.9%	-20.8%	9.9%	91.5%	x		
Arizona	9	D+4	<i>Greg Stanton</i>	D	159,583	101,662	0	61.1%	38.9%	0.0%	22.2%	15.9%	6.3%	90.0%	x		
Arkansas	1	R+17	Rick Crawford	R	57,907	138,757	4,581	28.8%	68.9%	2.3%	-40.2%	-34.8%	-5.4%	77.2%	x		
Arkansas	2	R+7	French Hill	R	116,135	132,125	5,193	45.8%	52.1%	2.0%	-6.3%	-10.7%	4.4%	82.6%	x		
Arkansas	3	R+19	Steve Womack	R	74,952	148,717	6,039	32.6%	64.7%	2.6%	-32.1%	-31.4%	-0.7%	78.6%	x		
Arkansas	4	R+17	Bruce Westerman	R	63,984	136,740	4,168	31.2%	66.7%	2.0%	-35.5%	-32.8%	-2.7%	75.7%	x		
California	1	R+11	Doug LaMalfa	R	131,506	160,006	0	45.1%	54.9%	0.0%	-9.8%	-19.4%	9.6%	91.6%			
California	2	D+22	Jared Huffman	D	243,051	72,541	0	77.0%	23.0%	0.0%	54.0%	45.2%	8.8%	90.5%			
California	3	D+5	John Garamendi	D	132,983	96,106	0	58.0%	42.0%	0.0%	16.1%	12.5%	3.6%	86.8%			
California	4	R+10	<i>Tom McClintock</i>	R	156,253	184,401	0	45.9%	54.1%	0.0%	-8.3%	-14.5%	6.2%	94.6%			
California	5	D+21	Mike Thompson	D	203,012	0	53,836	79.0%	0.0%	21.0%	79.0%	44.6%	N/A	83.8%			
California	6	D+21	Doris Matsui	D	201,939	0	0	100.0%	0.0%	0.0%	100.0%	44.0%	N/A	81.4%			
California	7	D+3	Ami Bera	D	155,016	126,601	0	55.0%	45.0%	0.0%	10.1%	11.2%	-1.1%	91.0%			
California	8	R+9	Paul Cook	R	0	170,785	0	0.0%	100.0%	0.0%	-100.0%	-15.1%	N/A	73.3%			
California	9	D+8	Jerry McNerney	D	113,240	87,263	0	56.5%	43.5%	0.0%	13.0%	18.2%	-5.2%	82.4%			

Table A-1. Years of School Completed by People 25 Years and Over, by Age and Sex: Selected Years 1940 to 2016

(Numbers in thousands. Noninstitutionalized population except where otherwise specified.)

Age, sex, and years	Total	Years of School Completed						
		Elementary		High school		College		Median
		0 to 4 years	5 to 8 years	1 to 3 years	4 years	1 to 3 years	4 years or more	

25 YEARS AND OLDER

Male

2016	103,372	1,183	3,513	7,144	30,780	26,468	34,283	(NA)
2015	101,887	1,243	3,669	7,278	30,997	25,778	32,923	(NA)
2014	100,592	1,184	3,761	7,403	30,718	25,430	32,095	(NA)
2013	99,305	1,127	3,836	7,314	30,014	25,283	31,731	(NA)
2012	98,119	1,237	3,879	7,388	30,216	24,632	30,766	(NA)
2011	97,220	1,234	3,883	7,443	30,370	24,319	29,971	(NA)
2010	96,325	1,279	3,931	7,705	30,682	23,570	29,158	(NA)
2009	95,518	1,372	4,027	7,754	30,025	23,634	28,706	(NA)
2008	94,470	1,310	4,136	7,853	29,491	23,247	28,433	(NA)
2007	93,421	1,458	4,249	8,294	29,604	22,219	27,596	(NA)
2006	92,233	1,472	4,395	7,940	29,380	22,136	26,910	(NA)
2005	90,899	1,505	4,402	7,787	29,151	21,794	26,259	(NA)



readxl part of the tidyverse

library(readxl)

```
edu
```

```
## # A tibble: 366 x 11
##   age   sex   year total elem4 elem8   hs3   hs4 coll3 coll4 median
##   <chr> <chr> <int> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 25-34 Male   2016 21845    116    468  1427  6386  6015  7432    NA
## 2 25-34 Male   2015 21427    166    488  1584  6198  5920  7071    NA
## 3 25-34 Male   2014 21217    151    512  1611  6323  5910  6710    NA
## 4 25-34 Male   2013 20816    161    582  1747  6058  5749  6519    NA
## 5 25-34 Male   2012 20464    161    579  1707  6127  5619  6270    NA
## 6 25-34 Male   2011 20985    190    657  1791  6444  5750  6151    NA
## 7 25-34 Male   2010 20689    186    641  1866  6458  5587  5951    NA
## 8 25-34 Male   2009 20440    184    695  1806  6495  5508  5752    NA
## 9 25-34 Male   2008 20210    172    714  1874  6356  5277  5816    NA
## 10 25-34 Male  2007 20024    246    757  1930  6361  5137  5593   NA
## # ... with 356 more rows
```

```
edu %>%  
  pivot_longer(cols = elem4:coll4, names_to = "educ")
```

```
## # A tibble: 2,196 x 7  
##   age   sex   year total median educ  value  
##   <chr> <chr> <int> <int>  <dbl> <chr> <dbl>  
## 1 25-34 Male   2016 21845     NA elem4    116  
## 2 25-34 Male   2016 21845     NA elem8    468  
## 3 25-34 Male   2016 21845     NA hs3     1427  
## 4 25-34 Male   2016 21845     NA hs4     6386  
## 5 25-34 Male   2016 21845     NA coll3   6015  
## 6 25-34 Male   2016 21845     NA coll4   7432  
## 7 25-34 Male   2015 21427     NA elem4    166  
## 8 25-34 Male   2015 21427     NA elem8    488  
## 9 25-34 Male   2015 21427     NA hs3     1584  
## 10 25-34 Male  2015 21427     NA hs4     6198  
## # ... with 2,186 more rows
```

```
gapminder %>%
  select(country, continent, year, lifeExp) %>%
  pivot_wider(names_from = year, values_from = lifeExp)

## # A tibble: 142 x 14
##   country continent `1952` `1957` `1962` `1967` `1972` `1977` `1982` `1987`
##   <fct>    <fct>     <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 Afghan... Asia        28.8    30.3    32.0    34.0    36.1    38.4    39.9    40.8
## 2 Albania Europe      55.2    59.3    64.8    66.2    67.7    68.9    70.4    72
## 3 Algeria Africa      43.1    45.7    48.3    51.4    54.5    58.0    61.4    65.8
## 4 Angola Africa       30.0    32.0    34       36.0    37.9    39.5    39.9    39.9
## 5 Argent... Americas   62.5    64.4    65.1    65.6    67.1    68.5    69.9    70.8
## 6 Austra... Oceania    69.1    70.3    70.9    71.1    71.9    73.5    74.7    76.3
## 7 Austria Europe       66.8    67.5    69.5    70.1    70.6    72.2    73.2    74.9
## 8 Bahrain Asia         50.9    53.8    56.9    59.9    63.3    65.6    69.1    70.8
## 9 Bangla... Asia        37.5    39.3    41.2    43.5    45.3    46.9    50.0    52.8
## 10 Belgium Europe      68      69.2    70.2    70.9    71.4    72.8    73.9    75.4
## # ... with 132 more rows, and 4 more variables: `1992` <dbl>, `1997` <dbl>,
## #   `2002` <dbl>, `2007` <dbl>
```

GETTING YOUR OWN DATA INTO R

```
my_data <- read_csv(file = "data/organdonation.csv")
```

Field delimiter is ,

```
read_csv2(file = "data/my_csv_file.csv")
```

Field delimiter is ;

```
read_dta(file = "data/my_stata_file.dta")
```

```
read_spss(file = "data/my_spss_file.sav")
```

```
read_sas(data_file = "<NAME>,"  
catalog_file = "<NAME>")
```

```
read_table(file = "<NAME>")
```

Structured but not delimited

From a local File Path

```
organs <- read_csv(file = "data/organdonation.csv")
```

From a remote URL

```
url <- "https://cdn.rawgit.com/kjhealy/viz-organdata/  
master/organdonation.csv"
```

```
organs <- read_csv(file = url)
```

England and Wales, Total Population, Death rates (period 1x1), Last modified: 02 Apr 2018; Methods Protocol: v6 (2017)

Year	Age	Female	Male	Total
1841	0	0.136067	0.169189	0.152777
1841	1	0.059577	0.063208	0.061386
1841	2	0.036406	0.036976	0.036689
1841	3	0.024913	0.026055	0.025480
1841	4	0.018457	0.019089	0.018772
1841	5	0.013967	0.014279	0.014123
1841	6	0.010870	0.011210	0.011040
1841	7	0.008591	0.008985	0.008788
1841	8	0.006860	0.007246	0.007053
1841	9	0.005772	0.006050	0.005911
1841	10	0.005303	0.005382	0.005343
1841	11	0.005114	0.005002	0.005057
1841	12	0.005145	0.004856	0.004999
1841	13	0.005455	0.004955	0.005202
1841	105	0.576967	1.121840	0.700375
1841	106	0.677711	6.000000	0.795287
1841	107	0.900000	.	0.900000
1841	108	1.388430	.	1.388430
1841	109	.	.	.
1841	110+	.	.	.
1842	0	0.148491	0.184007	0.166481
1842	1	0.063038	0.066596	0.064818
1842	2	0.035203	0.035854	0.035527

```
engmort <- read_table(file = "data/mortality.txt",
                      skip = 2, na = ".")
```

HOW ggplot WORKS

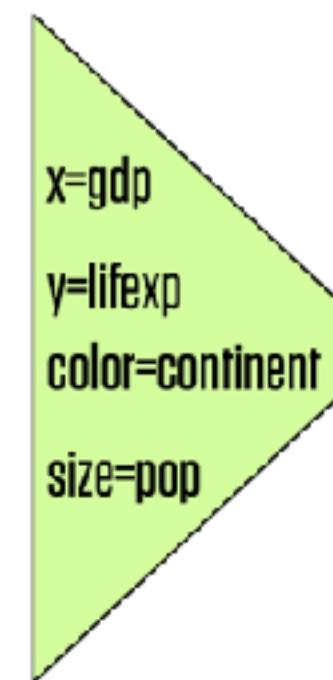
ggplot's FLOW OF ACTION

1. Tidy Data

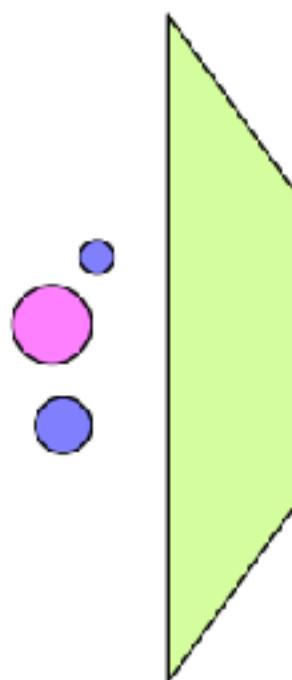
gdp	lifexp	pop	continent
340	65	31	Euro
227	51	200	Amer
909	81	80	Euro
126	40	20	Asia

```
ggplot(data = gapminder, mapping =  
       aes(x = gdp,  
             y = lifespan,  
             color = continent,  
             size = pop))
```

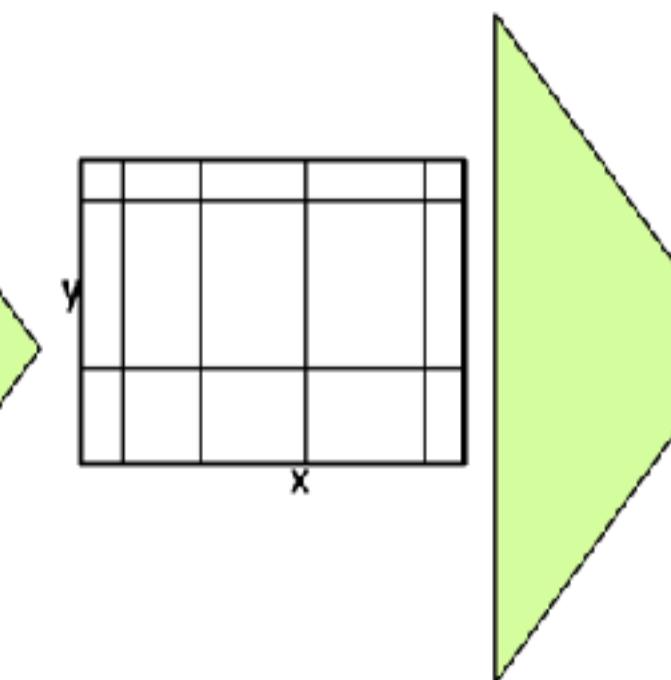
2. Mapping



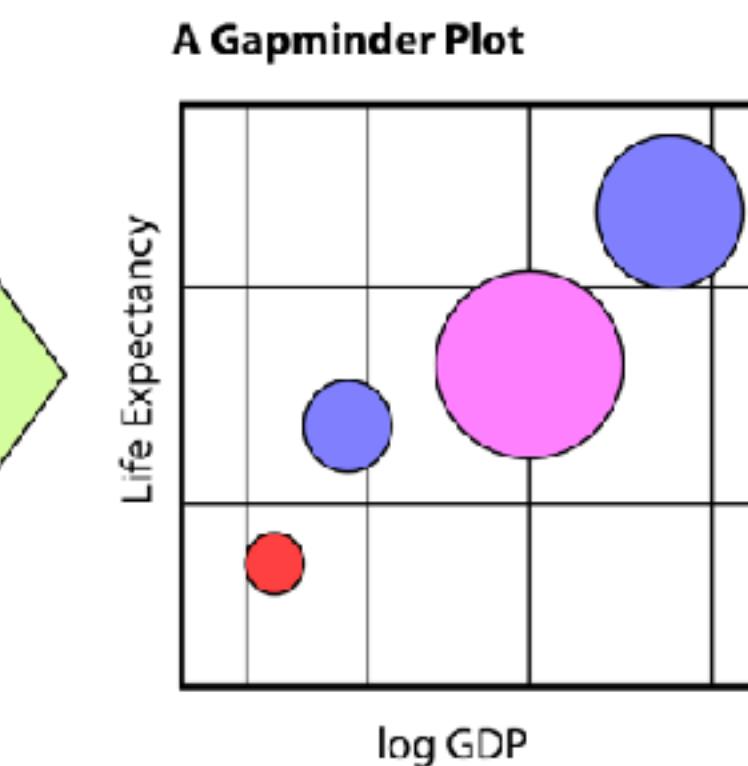
3. Geom



4. Co-ordinates, Scales

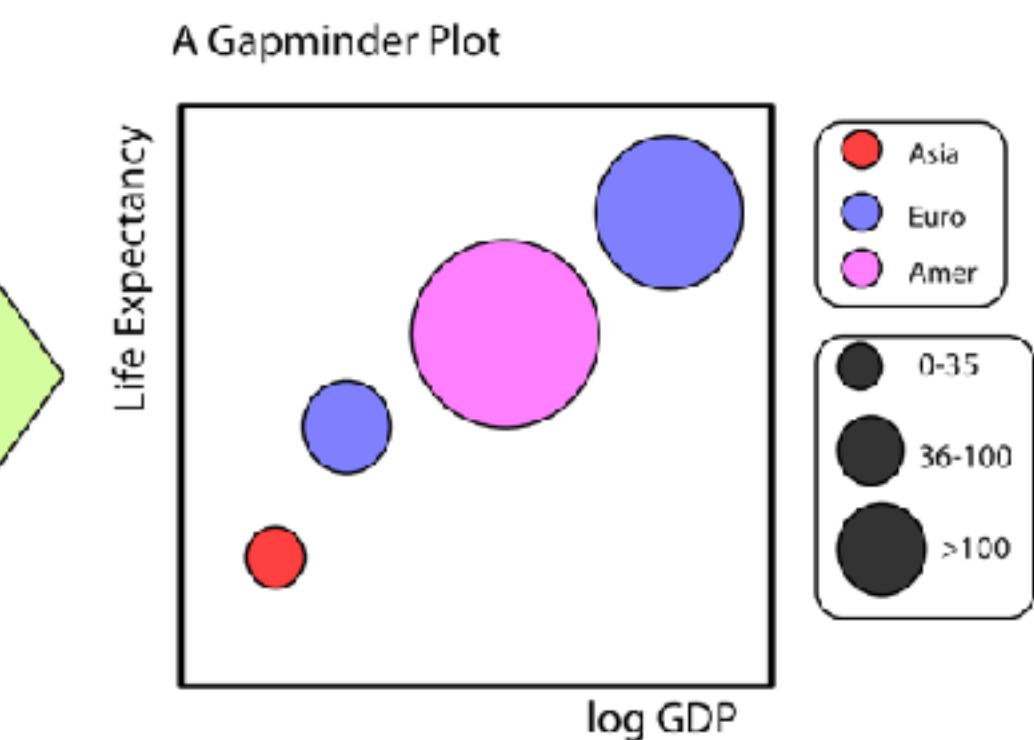


5. Labels & Guides



```
labs()  
guides()
```

6. Themes

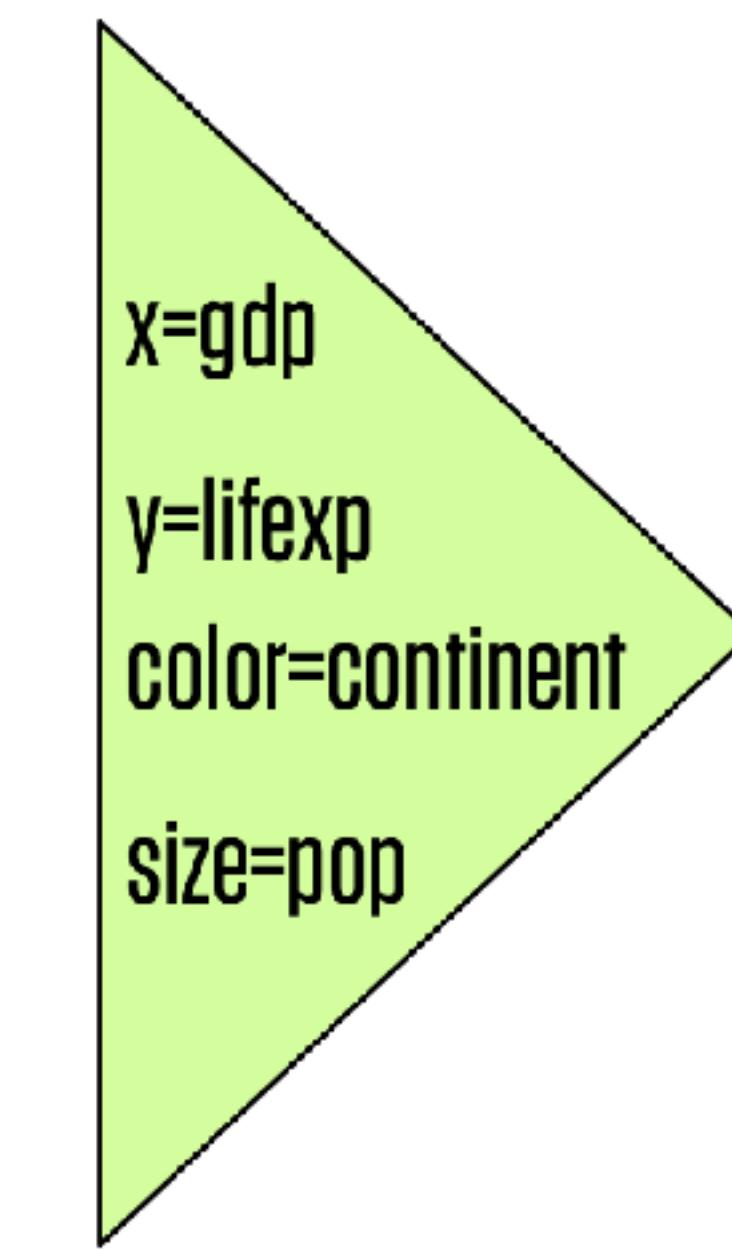


```
theme_minimal()
```

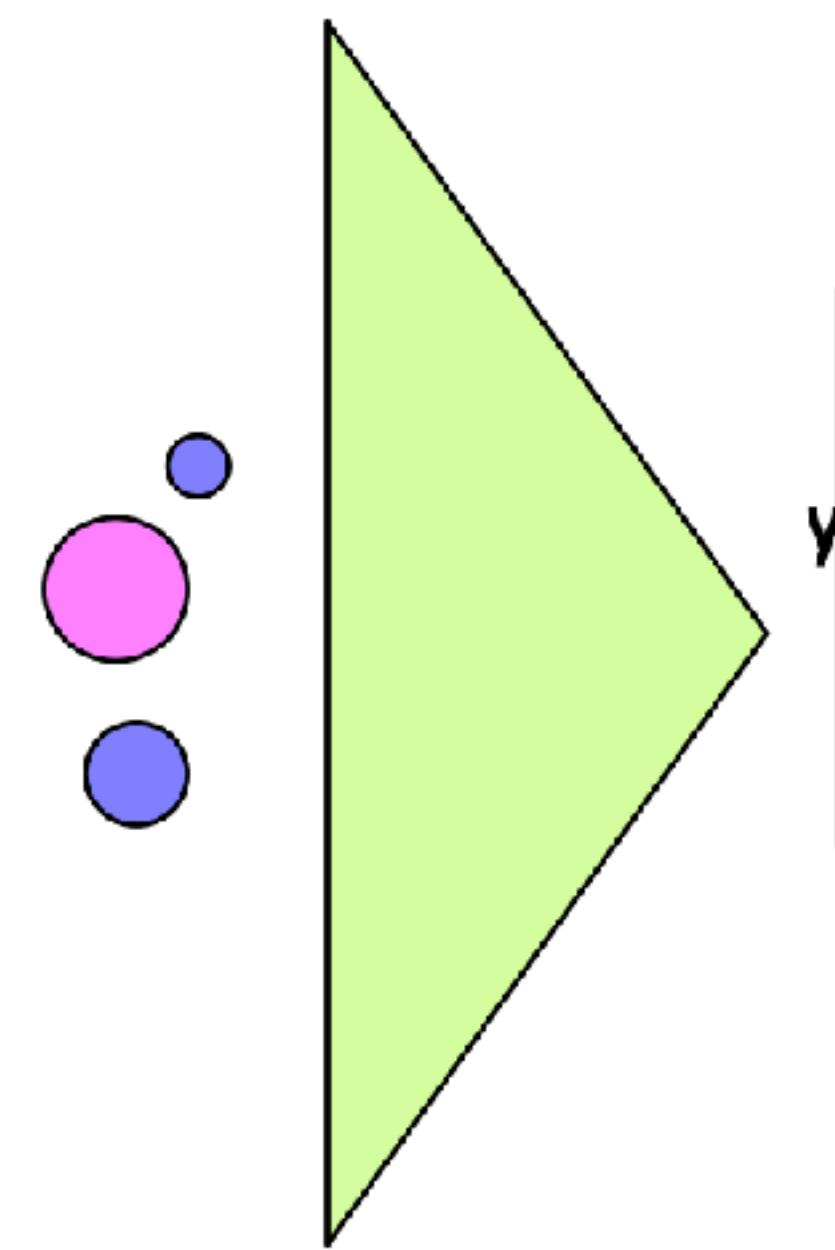
1. Tidy Data

gdp	lifexp	pop	continent
340	65	31	Euro
227	51	200	Amer
909	81	80	Euro
126	40	20	Asia

2. Mapping



3. Geom



4. Co-Ordinates Scales

```
ggplot(data = gapminder, mapping =  
aes(x = gdp,  
y = lifespan,  
color = continent,  
size = pop))
```

geom_point()

coord_cartes
scale_x_log

2. Mapping

	continent
0	Euro
1	Amer
2	Euro
3	Asia

```
x=gdp  
y=lifexp  
color=continent  
size=pop
```

3. Geom

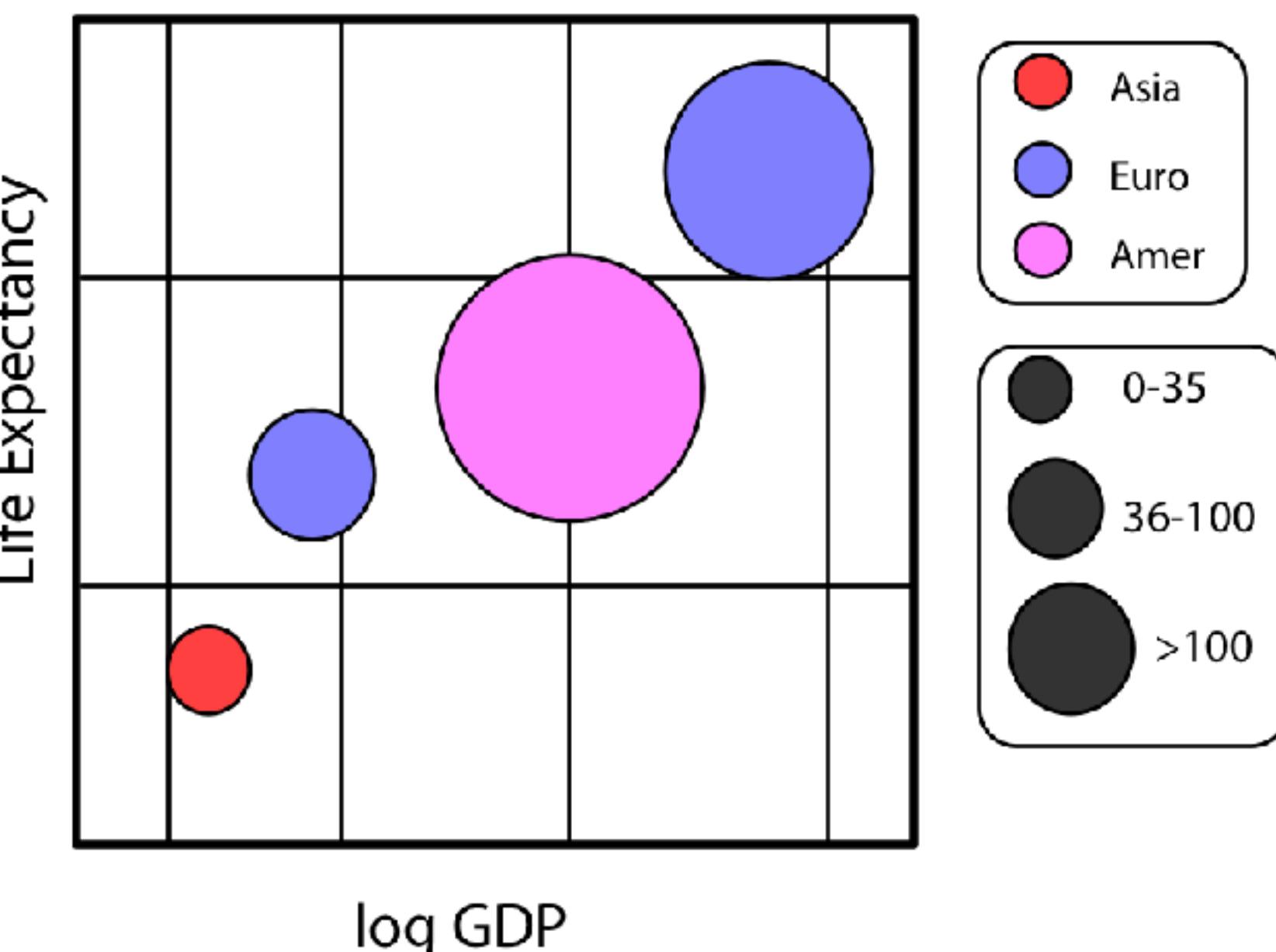
```
geom_point()
```

4. Co-ordinates, Scales

```
coord_cartesian()  
scale_x_log10()
```

5. Labels & Guides

A Gapminder Plot

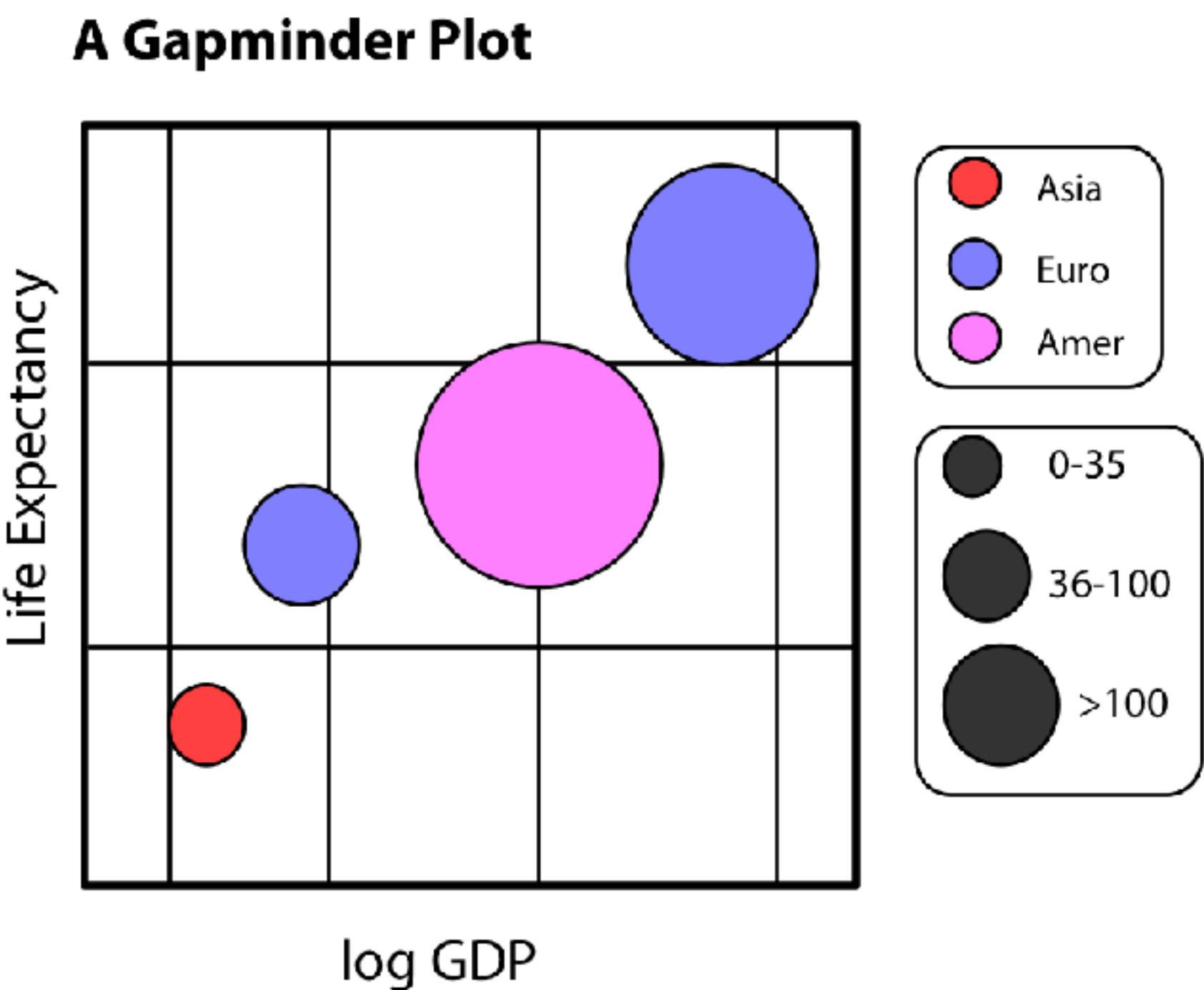


```
a = gapminder, mapping =  
aes(x = gdp,  
y = lifespan,  
color = continent,  
size = pop))
```

```
labs()  
guides()
```

ates,

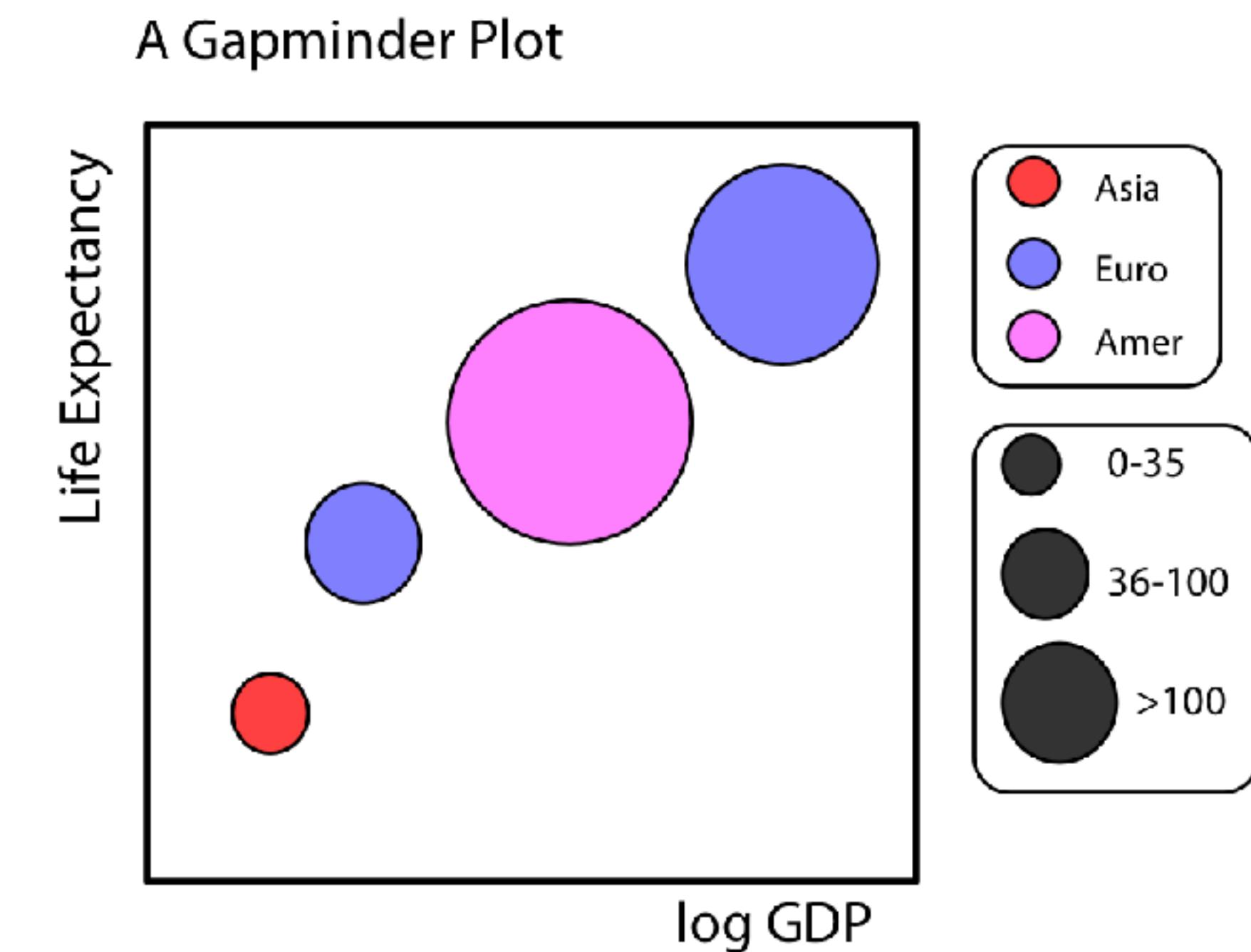
5. Labels & Guides



labs()

guides()

6. Themes



theme_minimal()

1. Tidy Data

gdp	lifexp	pop	continent
340	65	31	Euro
227	51	200	Amer
909	81	80	Euro
126	40	20	Asia

2. Mapping

```
x=gdp  
y=lifexp  
color=continent  
size=pop
```

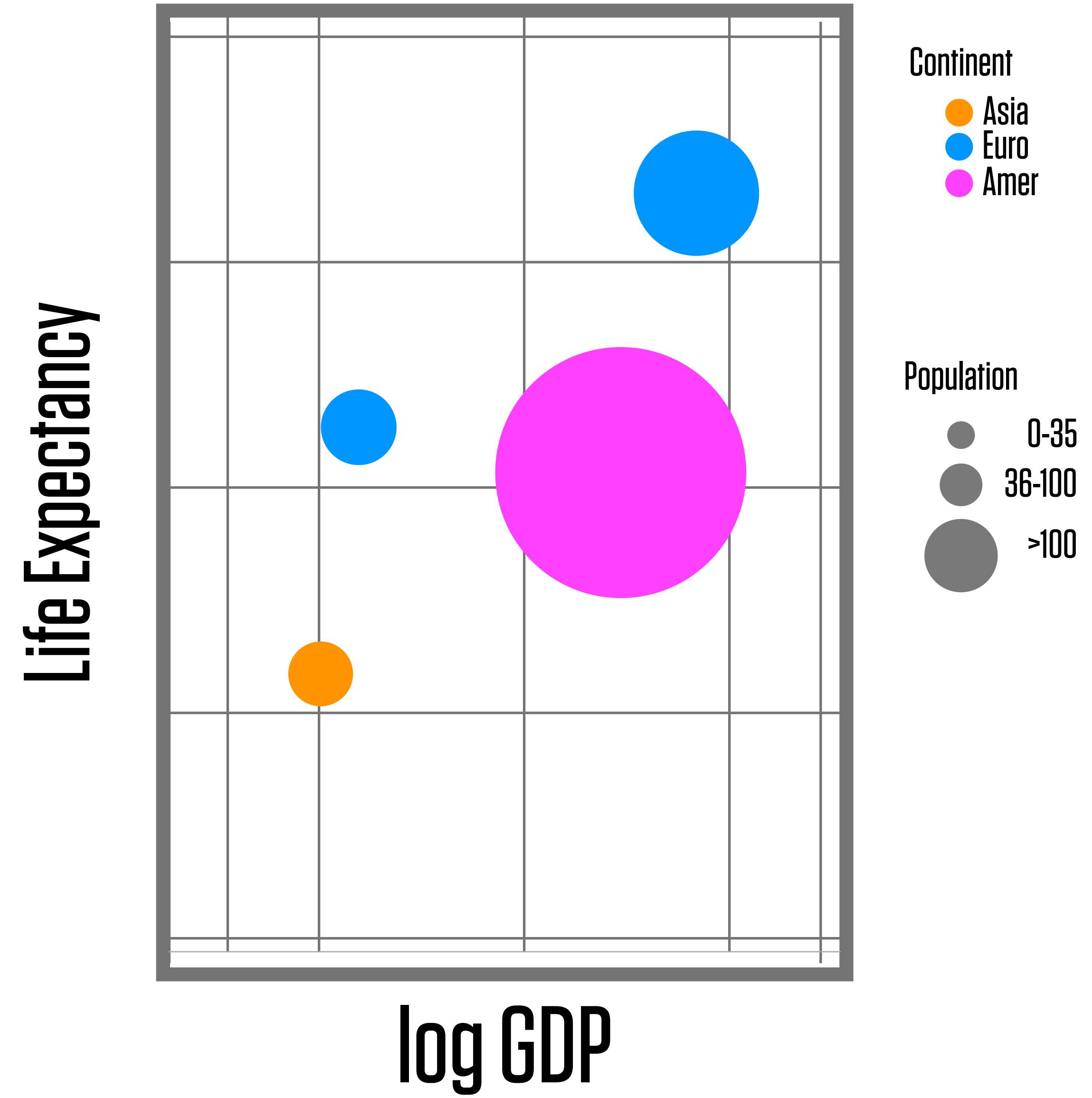
3. Geom

```
geom_point()
```

```
ggplot(data = gapminder, mapping =  
aes(x = gdp,
```

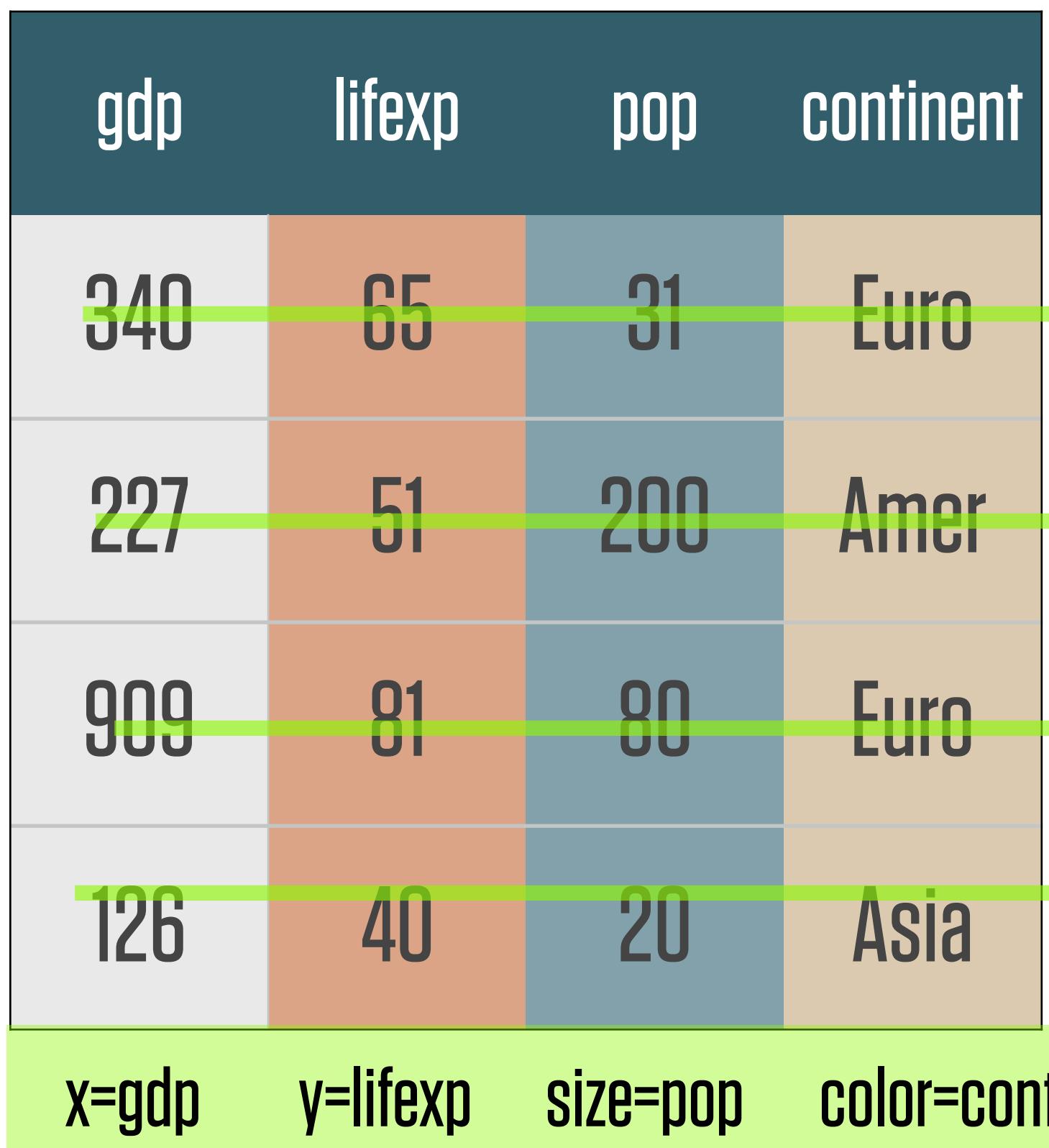
gdp	lifexp	pop	continent
340	65	31	Euro
227	51	200	Amer
909	81	80	Euro
126	40	20	Asia

A Gapminder Plot



1. Tidy Data

`ggplot(data = gapminder)`



2. Mapping

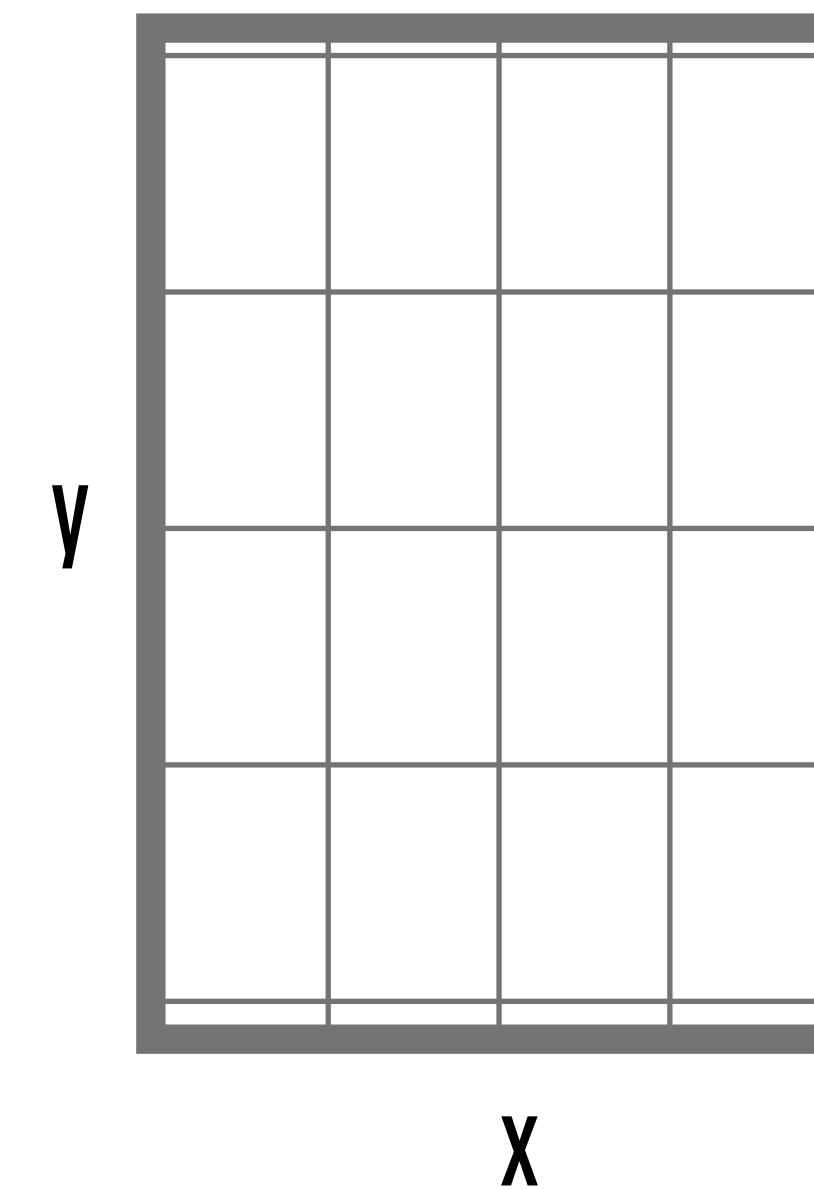
`ggplot(mapping = aes(x = ...))`

3. Geom

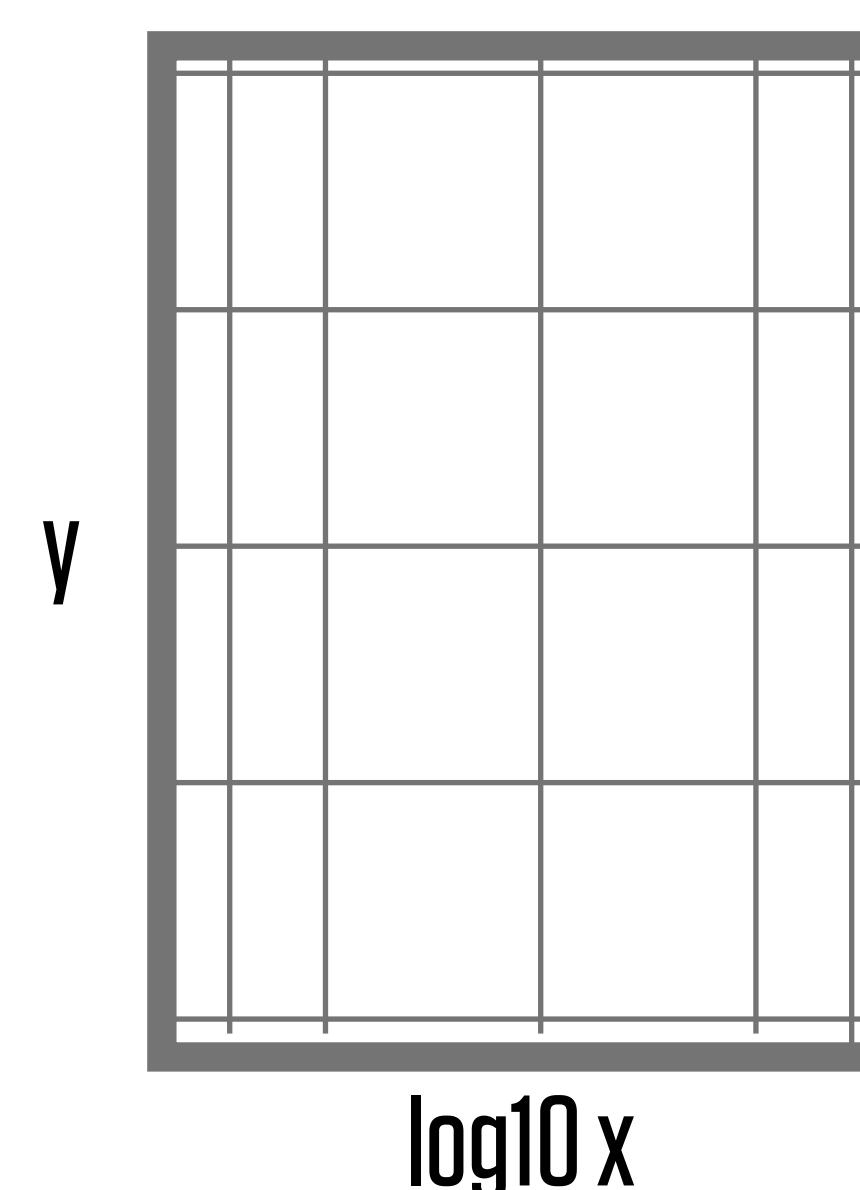
`geom_point()`

Required

4. Coordinate System



5. Scales



6. Labels & Guides

A Gapminder Plot

- Asia
 - Euro
 - Amer
- 0-35
 - 36-100
 - >100

Life Expectancy

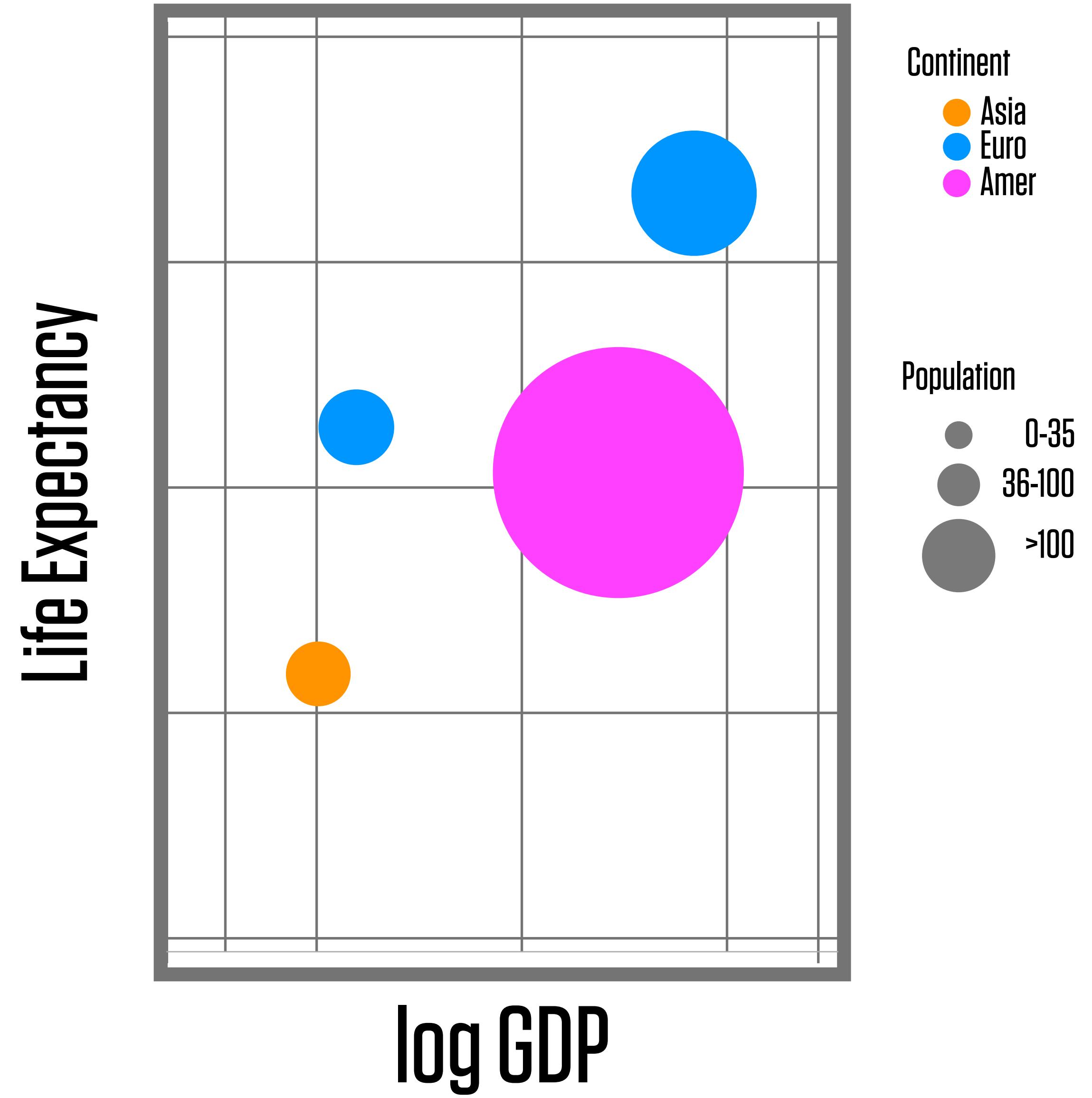
Continent

Population

log GDP

These steps are initially implicit

A Gapminder Plot



PIECE BY PIECE

```
head(gapminder)
```

```
## # A tibble: 6 × 6
##       country continent   year lifeExp      pop gdpPercap
##       <fctr>     <fctr> <int>    <dbl>    <int>     <dbl>
## 1 Afghanistan      Asia  1952  28.801  8425333  779.4453
## 2 Afghanistan      Asia  1957  30.332  9240934  820.8530
## 3 Afghanistan      Asia  1962  31.997 10267083  853.1007
## 4 Afghanistan      Asia  1967  34.020 11537966  836.1971
## 5 Afghanistan      Asia  1972  36.088 13079460  739.9811
## 6 Afghanistan      Asia  1977  38.438 14880372  786.1134
```

```
dim(gapminder)
## [1] 1704      6
```

```
p <- ggplot(data = gapminder)
```

Create a ggplot object
Data is gapminder table

```
p <- ggplot(data = gapminder,  
             mapping = aes(x = gdpPercap,  
                            y = lifeExp))
```

mapping: tell ggplot the variables you want represented by elements of the plot

- The mapping = `aes(...)` instruction **links**

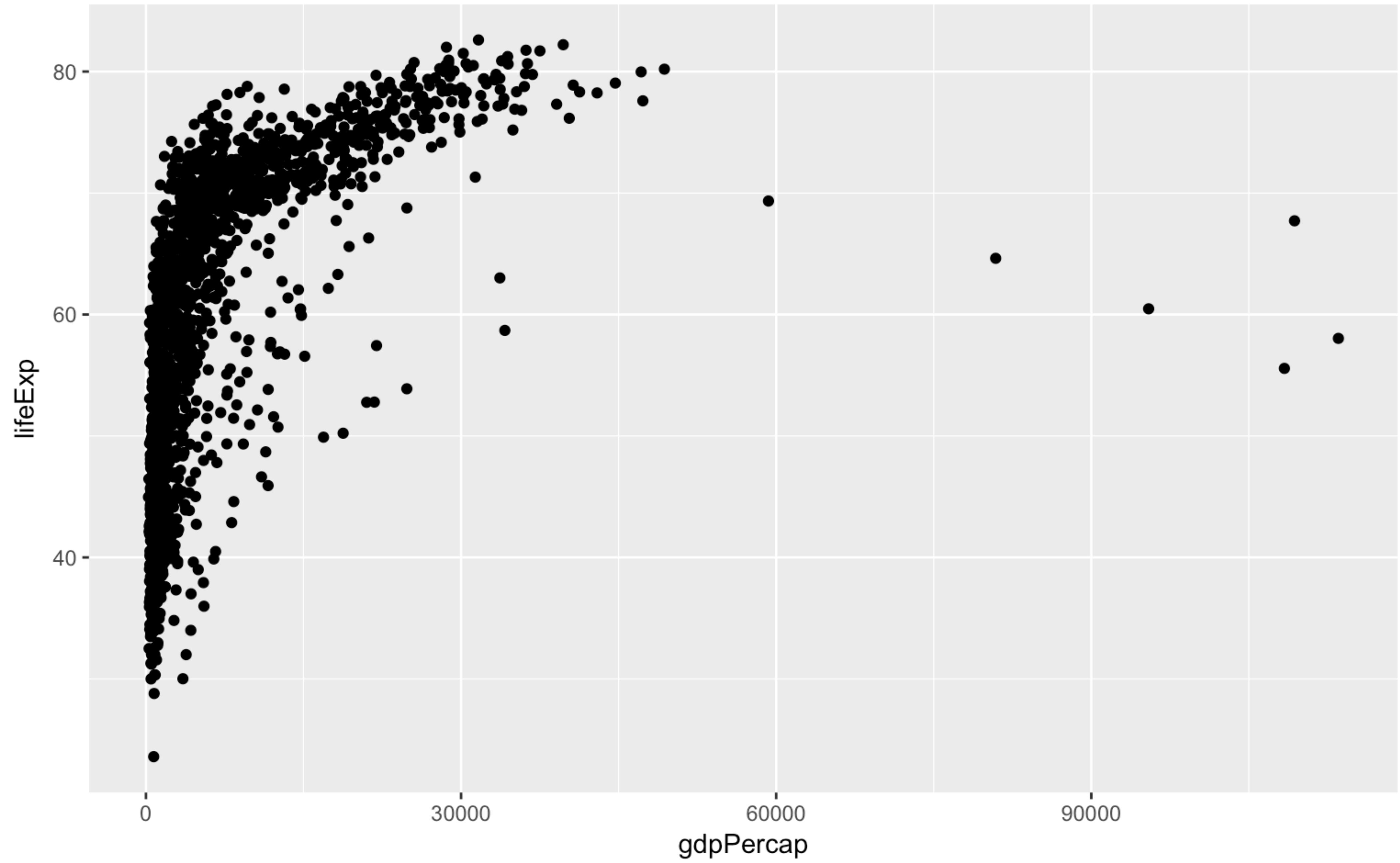
variables to things you will see on the plot.

- The `x` and `y` values are the most obvious ones.
- Other aesthetic mappings can include, e.g.,
color, shape, and size.

Mappings do not directly specify the particular, e.g., colors, shapes, or line styles that will appear on the plot. Rather they establish **which variables in the data will be represented by which visible elements of the plot.**

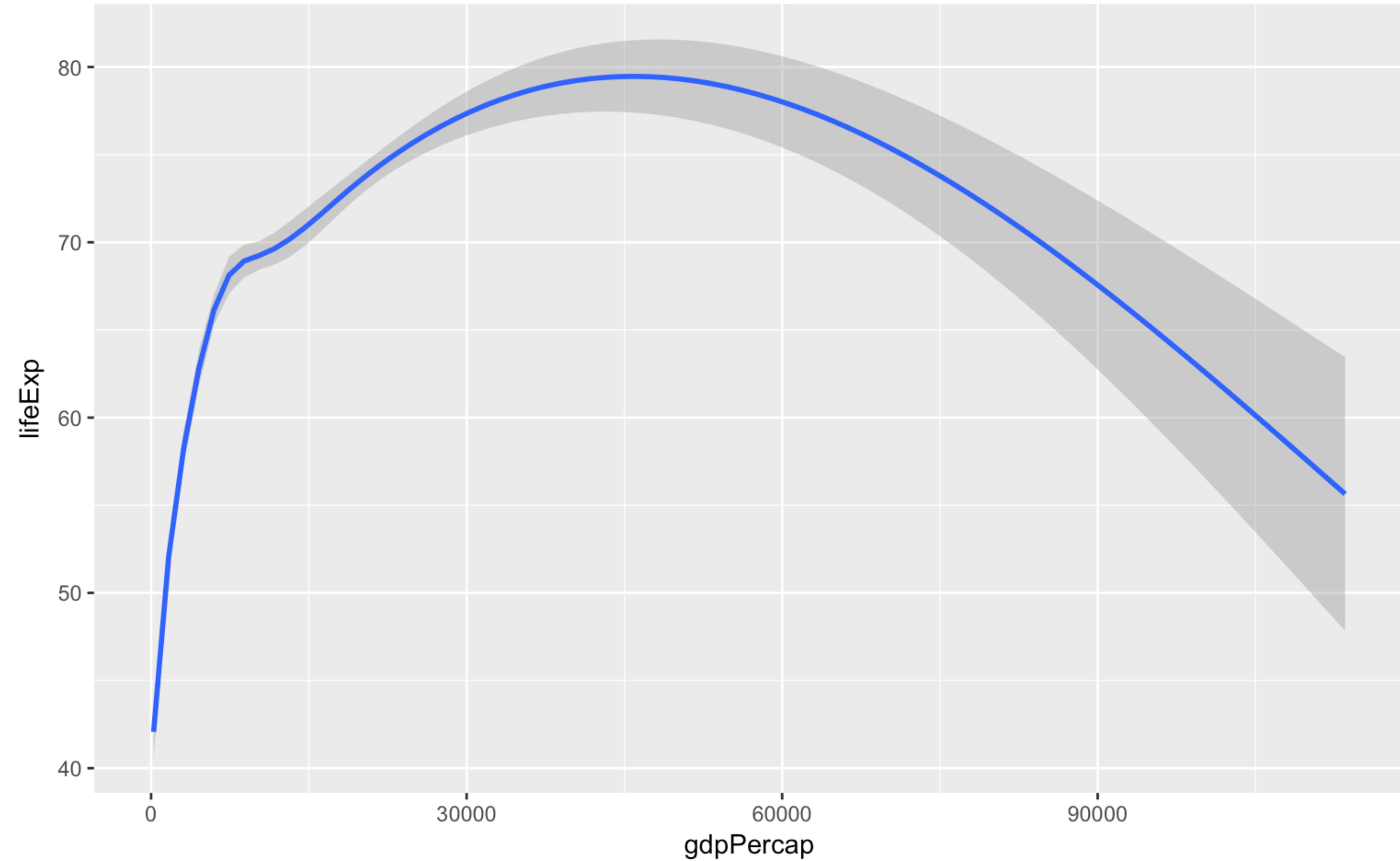
```
p + geom_point()
```

Add a geom layer
to the plot



```
p + geom_smooth()
```

Try a different geom

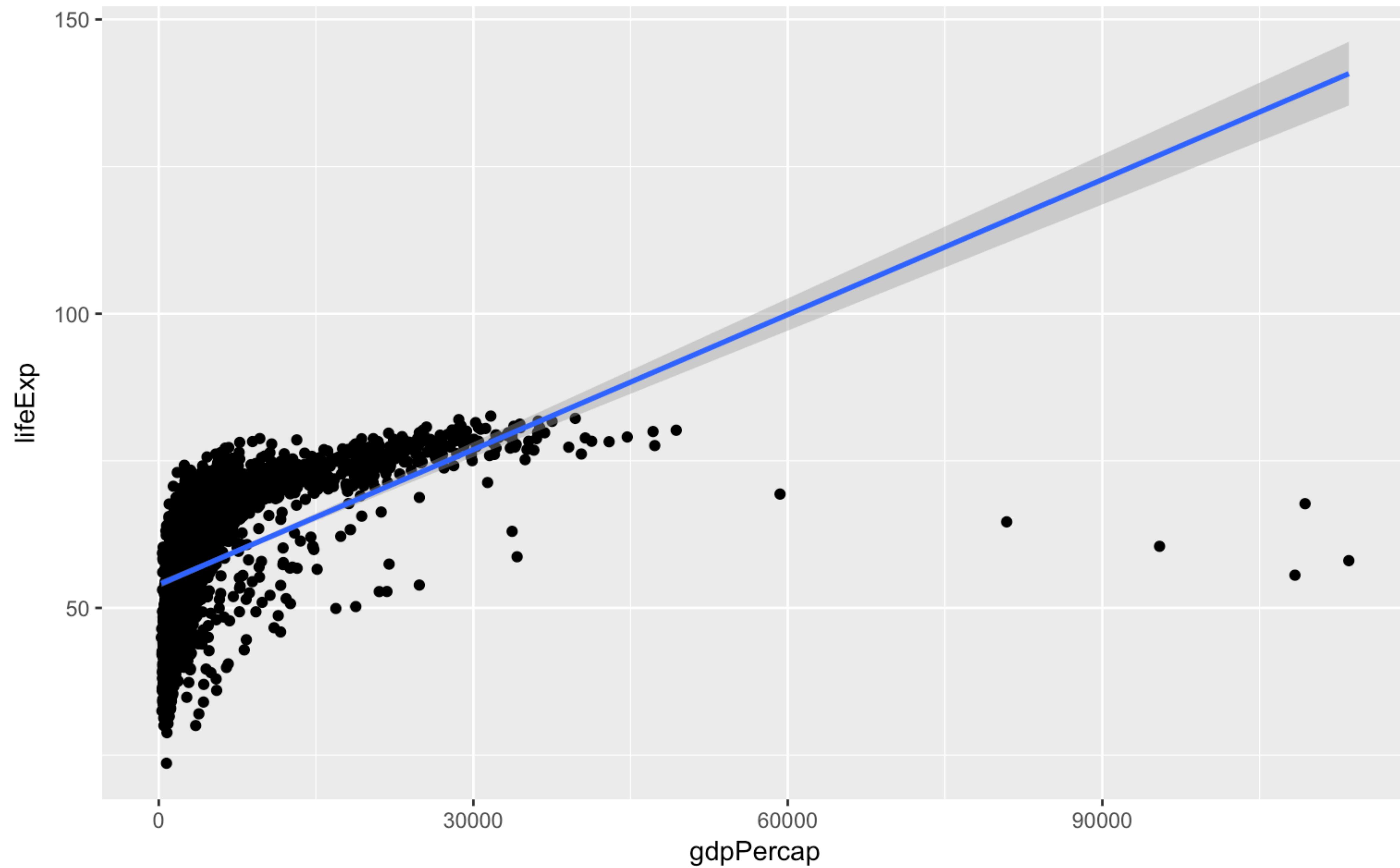


```
p + geom_point() +  
  geom_smooth() +  
  scale_x_log10(labels = scales::dollar)
```

This process is additive

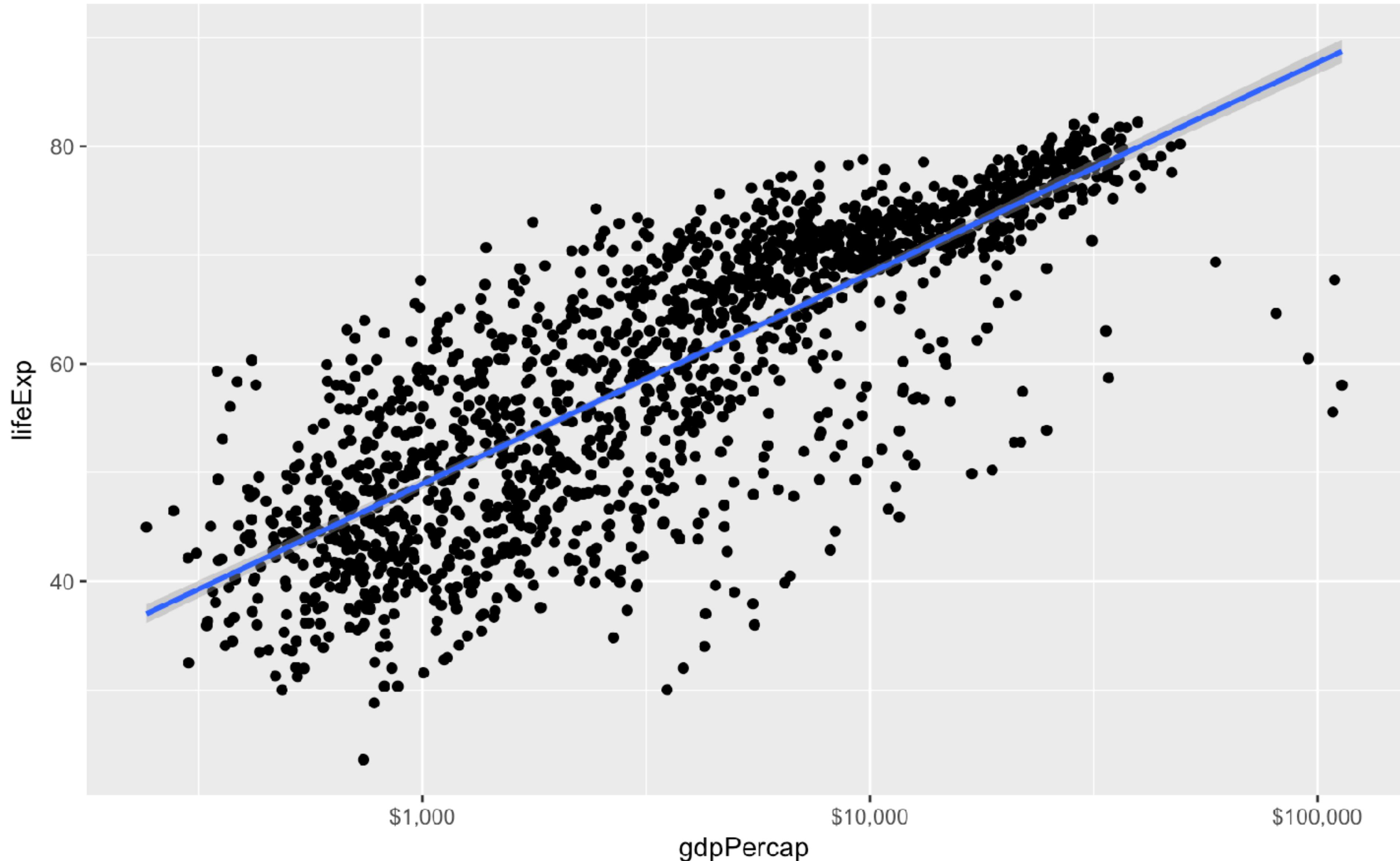
```
p + geom_point() +  
  geom_smooth(method = "lm")
```

Every geom is a function.
Remember, functions take arguments.



```
p <- ggplot(data = gapminder,  
             mapping = aes(x = gdpPercap,  
                           y = lifeExp))  
p + geom_point() +  
  geom_smooth(method = "lm") +  
  scale_x_log10(label = scales::dollar)
```

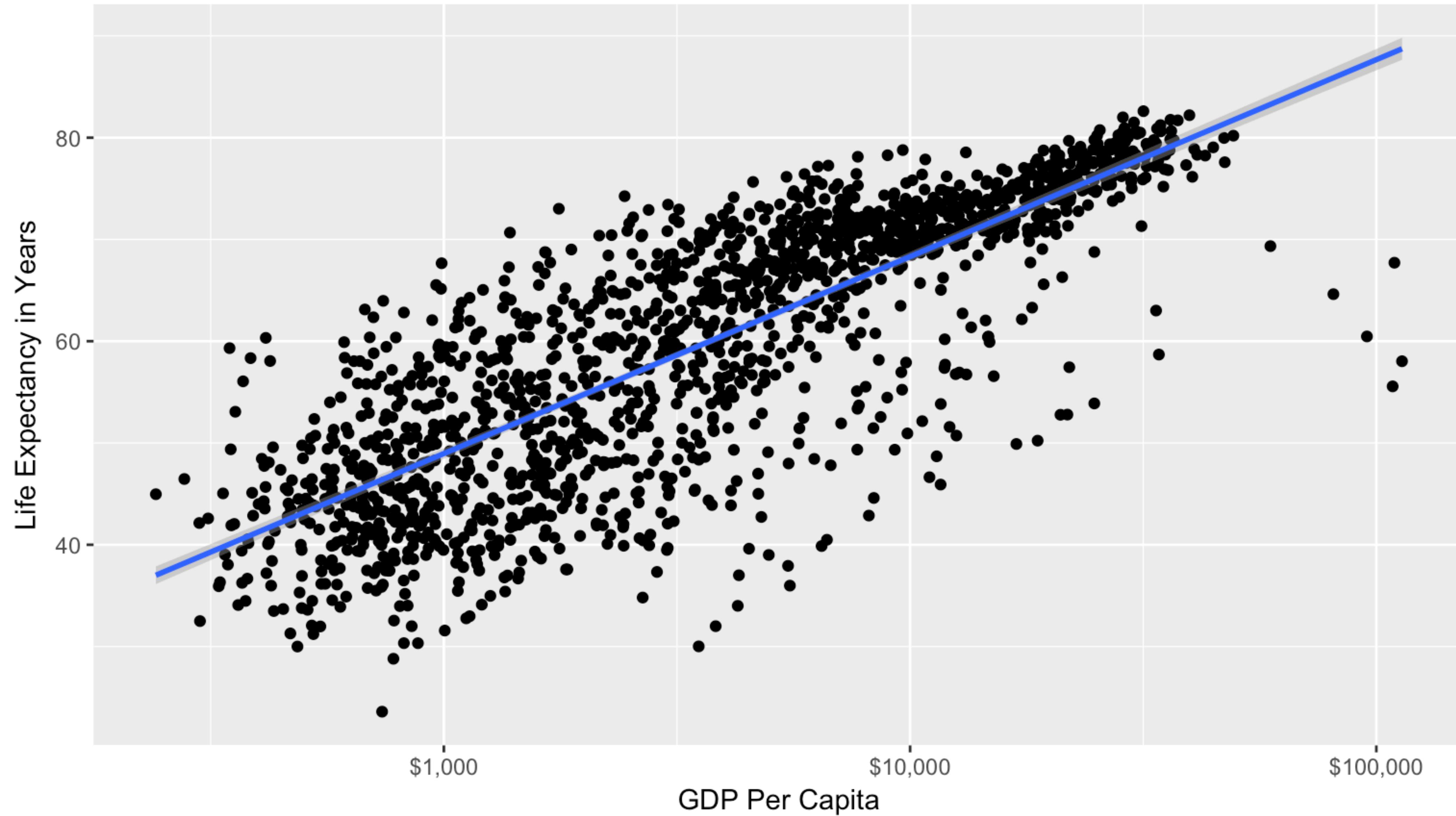
Keep Layering



```
p + geom_point() +
  geom_smooth(method = "gam") +
  scale_x_log10(labels = scales::dollar) +
  labs(x = "GDP Per Capita",
       y = "Life Expectancy in Years",
       title = "Economic Growth and Life Expectancy",
       subtitle = "Data points are country-years",
       caption = "Data source: Gapminder")
```

Economic Growth and Life Expectancy

Data points are country-years



**MAPPING
vs SETTING
AESTHETICS**

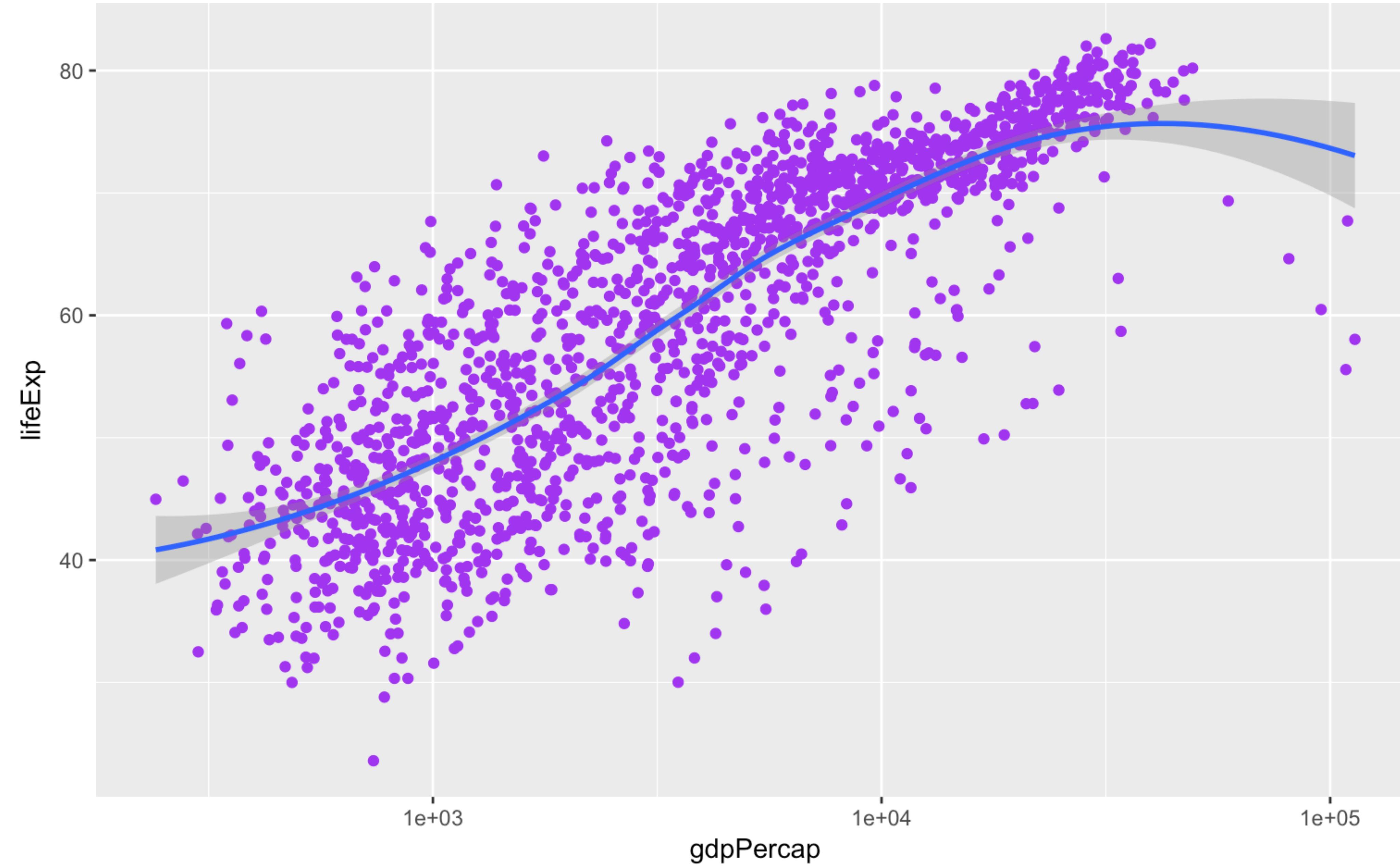
```
p <- ggplot(data = gapminder,  
             mapping = aes(x = gdpPercap,  
                            y = lifeExp,  
                            color = "purple"))
```

```
p + geom_point() +  
  geom_smooth(method = "loess") +  
  scale_x_log10()
```



What has gone wrong here?

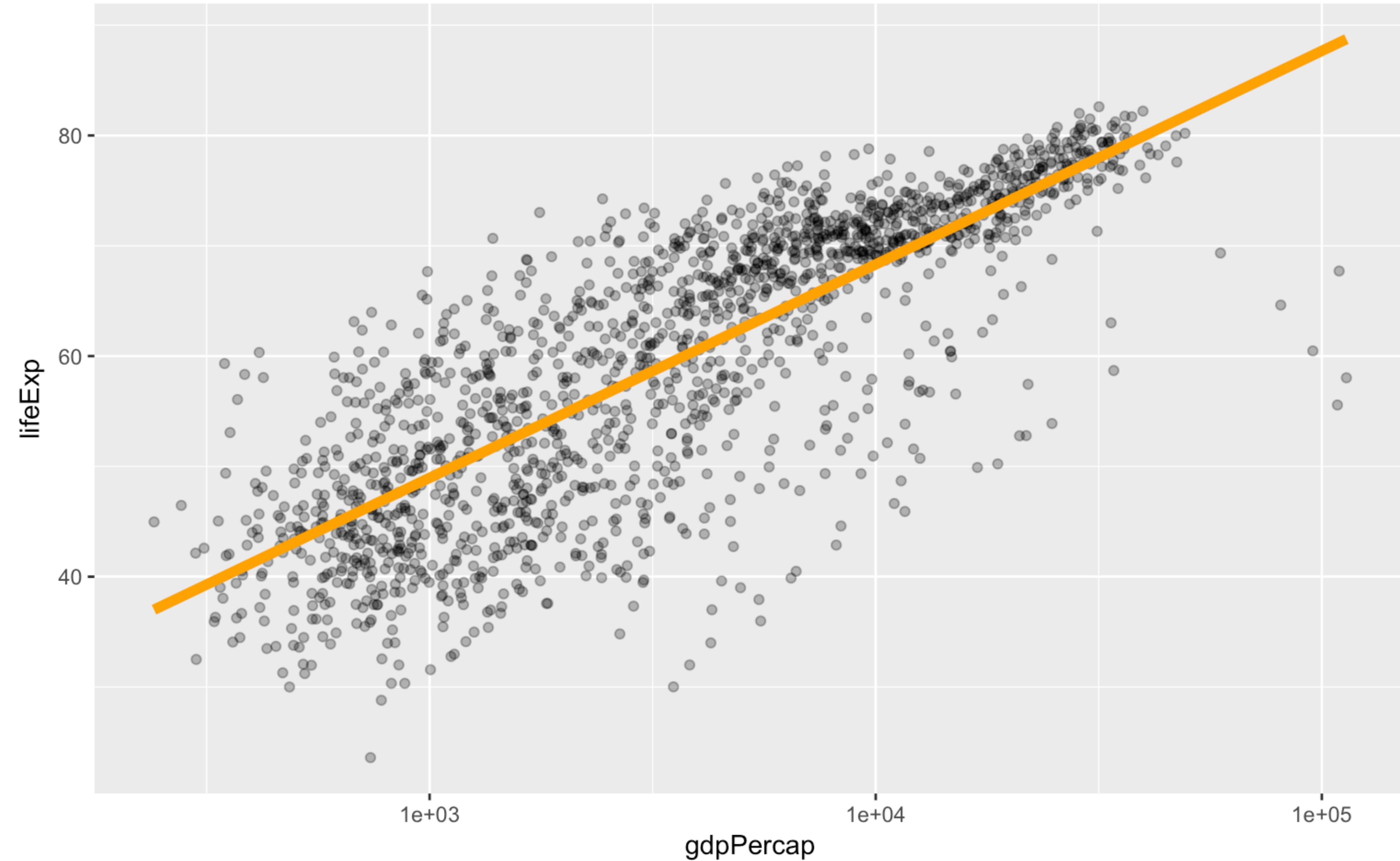
```
p <- ggplot(data = gapminder,  
             mapping = aes(x = gdpPercap,  
                           y = lifeExp)) +  
  geom_point(color = "purple") +  
  geom_smooth(method = "loess") +  
  scale_x_log10()
```



geom functions can take many different arguments

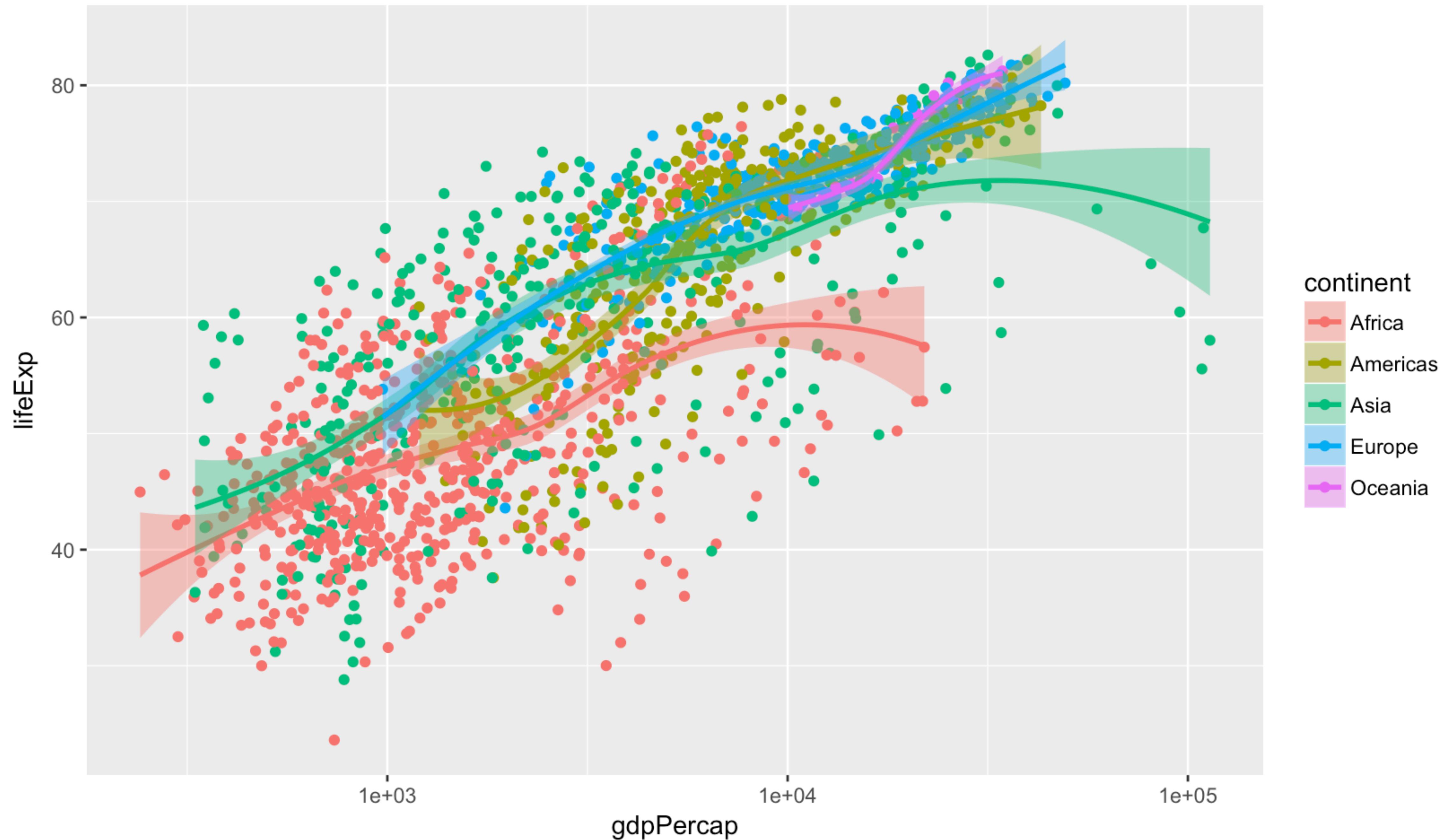
```
p <- ggplot(data = gapminder,  
             mapping = aes(x = gdpPercap,  
                            y = lifeExp))  
  
p + geom_point(alpha = 0.3) +  
  geom_smooth(color = "orange",  
              se = FALSE, size = 2, method = "lm") +  
  scale_x_log10()
```

Here, some elements are **mapped to variables**, while others are **set to values**



```
p <- ggplot(data = gapminder,  
             mapping = aes(x = gdpPercap,  
                            y = lifeExp,  
                            color = continent,  
                            fill = continent))
```

```
p + geom_point() +  
  geom_smooth(method = "loess") +  
  scale_x_log10()
```



MAP or SET
AESTHETICS
per geom

```
p <- ggplot(data = gapminder,  
             mapping = aes(x = gdpPercap,  
                           y = lifeExp))  
p + geom_point(mapping = aes(color = continent)) +  
  geom_smooth(method = "loess") +  
  scale_x_log10()
```

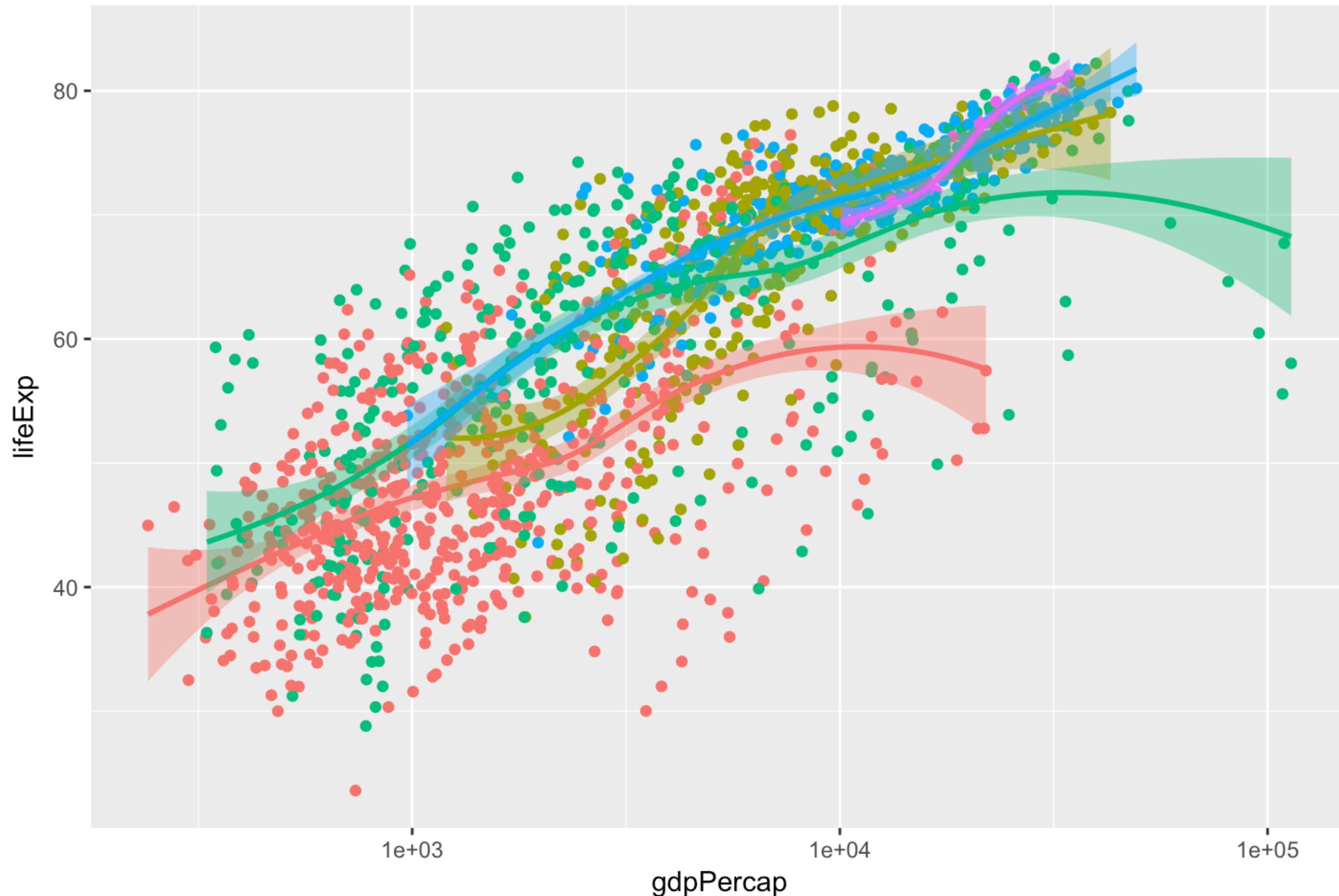


PAY CLOSE ATTENTION
TO WHICH SCALES AND
GUIDES ARE DRAWN,
AND WHY THAT HAPPENS

```

p <- ggplot(data = gapminder,
             mapping =
               aes(x = gdpPercap, y = lifeExp,
                   color = continent, fill = continent))
p + geom_point() +
  geom_smooth(method = "loess") +
  scale_x_log10()

```



continent

- Africa
- Americas
- Asia
- Europe
- Oceania

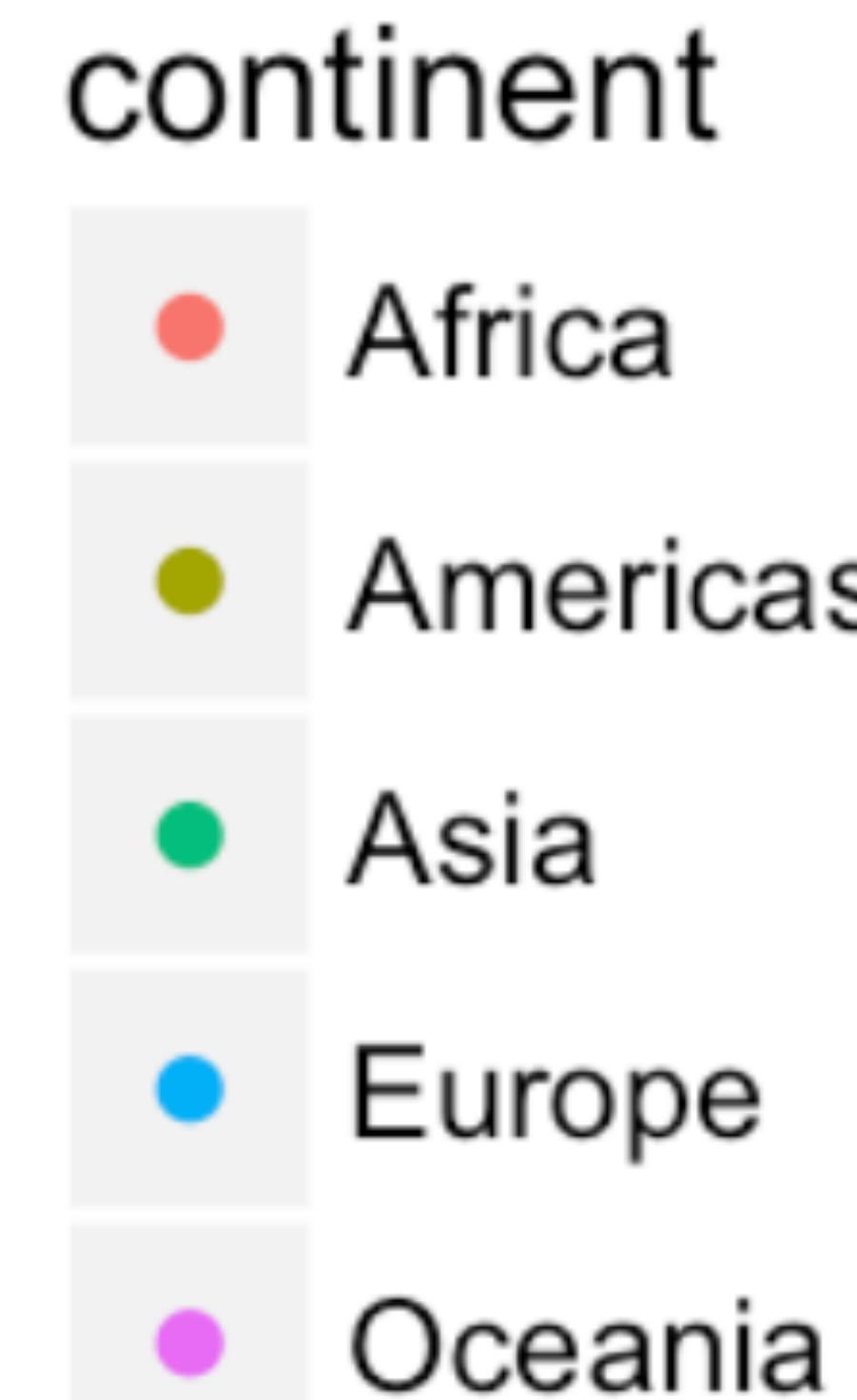
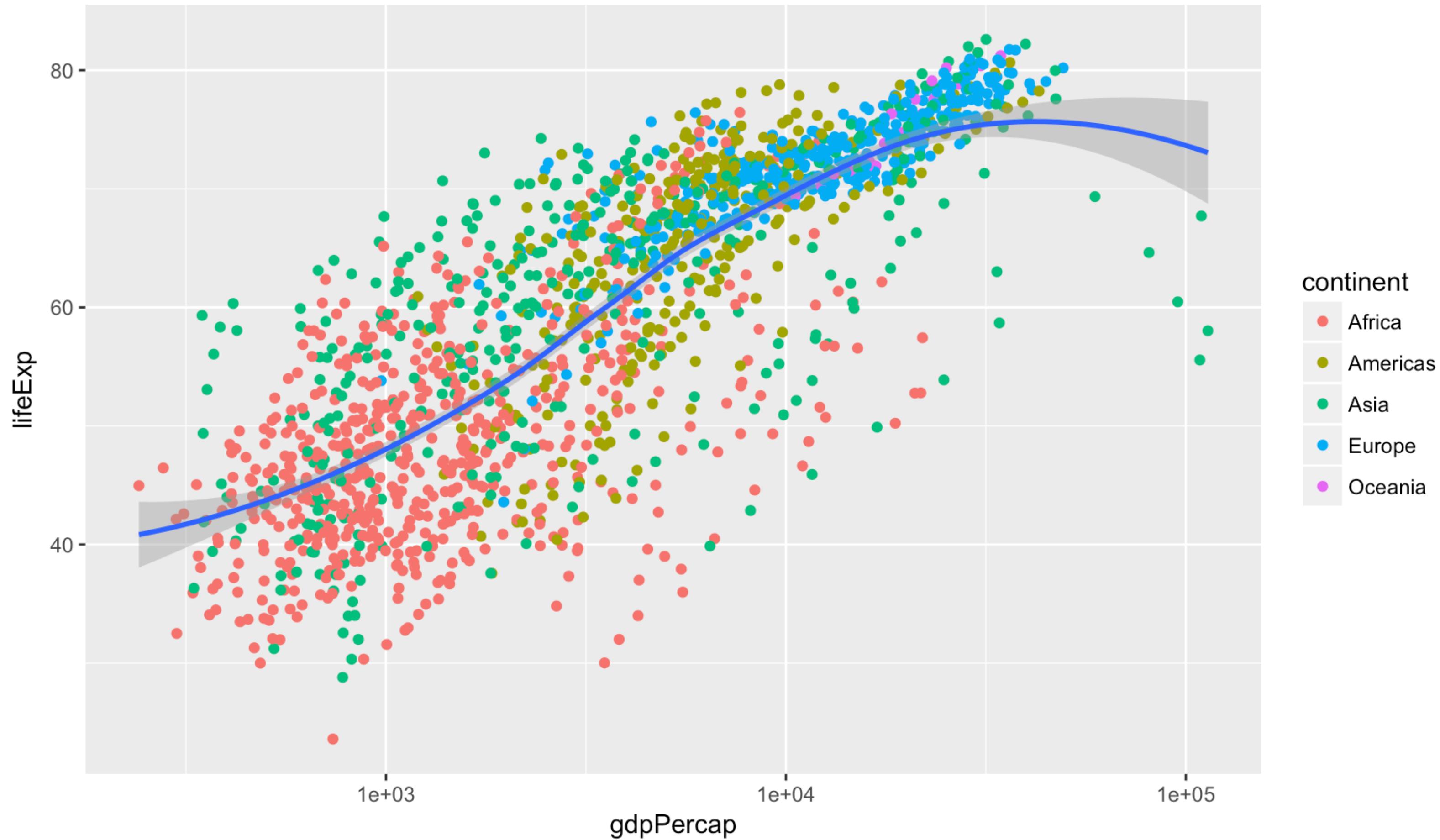
continent

- Africa
- Americas
- Asia
- Europe
- Oceania

```

p <- ggplot(data = gapminder,
             mapping =
               aes(x = gdpPercap, y = lifeExp))
p + geom_point(mapping = aes(color = continent)) +
  geom_smooth(method = "loess") +
  scale_x_log10()

```



**REMEMBER:
EVERY MAPPED
VARIABLE HAS A SCALE**

Saving Your Work

ggsave()

ggsave("figures/my_figure.png")

ggsave("my_figure.pdf")

**ggsave("my_figure.pdf",
plot = p5,
scale = 1.2)**

**ggsave("figures/my-figure.pdf",
plot = p5,
width = 8,
height = 5)**

With ggsave

With pdf() or other graphics devices

```
pdf(file = "plot.pdf", height = 5in,  
     width = 5in)
```

Open device ...

```
print(p4)
```

```
dev.off()
```



... and close when done

Getting Help

The name of the function, and the library it is in.

mean {base}

R Documentation

Arithmetic Mean

What it does.

Generic function for the (trimmed) arithmetic mean.

Usage

```
mean(x, ...)  
  
## Default S3 method:  
mean(x, trim = 0, na.rm = FALSE, ...)
```

The function's name, and in the parentheses the named arguments it expects, in the order it expects them. If an argument has a default value, it is shown. Arguments without default values (e.g. `x`) must be provided by you.

Arguments

- `x` An R object. Currently there are methods for numeric/logical vectors and [date](#), [date-time](#) and [time interval](#) objects. Complex vectors are allowed for `trim = 0`, only.
- `trim` the fraction (0 to 0.5) of observations to be trimmed from each end of `x` before the mean is computed. Values of `trim` outside that range are taken as the nearest endpoint.
- `na.rm` a logical value indicating whether `NA` values should be stripped before the computation proceeds.
- `...` further arguments passed to or from other methods.

The ellipsis allows other arguments to be passed to and from the function.

Value

If `trim` is zero (the default), the arithmetic mean of the values in `x` is computed, as a numeric or complex vector of length one. If `x` is not logical (coerced to numeric), numeric (including integer) or complex, `NA_real_` is returned, with a warning.

If `trim` is non-zero, a symmetrically trimmed mean is computed with a fraction of `trim` observations deleted from each end before the mean is computed.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

[weighted.mean](#), [mean.POSIXct](#), [colMeans](#) for row and column means.

Other related functions

Examples

```
x <- c(0:10, 50)  
xm <- mean(x)  
c(xm, mean(x, trim = 0.10))
```

Self-contained examples that you can run at the console. These may use built-in datasets or other R functions.

The name of the function, and the library it is in.

What it does.

More details on each

mean {base}

Description

Generic function for the (trimmed) arithmetic mean.

Usage

mean(x, ...)

Default S3 method:

mean(x, trim = 0, na.rm = FALSE, ...)

Arguments

R Documentation

Arithmetic Mean

Arithmetic Mean

Description

Generic function for the (trimmed) arithmetic mean.

Usage

```
mean(x, ...)
```

```
## Default S3 method:
```

```
mean(x, trim = 0, na.rm = FALSE, ...)
```

Arguments

x An R object. Currently there are methods for numeric/logical vectors and [date](#), [date-time](#) and [time interval](#) objects. Complex vectors are allowed for `trim = 0`, only.

trim the fraction (0 to 0.5) of observations to be trimmed from each end of **x** before the mean is computed. Values of `trim` outside that range are taken as the nearest endpoint.

na.rm a logical value indicating whether NA values should be stripped before the computation proceeds.

The function's name, and in the parentheses the named arguments it expects, in the order it expects them. If an argument has a default value, it is shown. Arguments without default values (e.g. `x`) must be provided by you.

What it does.

More details on each named argument. This will tell you what class of thing each argument has to be—an object, a number, a data frame, a logical value, etc.

What the function returns—i.e., the result of whatever operation or calculation it performs. This can be

Description

Generic function for the (trimmed) arithmetic mean.

Usage

```
mean(x, ...)  
  
## Default S3 method:  
mean(x, trim = 0, na.rm = FALSE, ...)
```

Arguments

- x An R object. Currently there are methods for numeric/logical vectors and [date](#), [date-time](#) and [time interval](#) objects. Complex vectors are allowed for `trim = 0`, only.
- trim the fraction (0 to 0.5) of observations to be trimmed from each end of x before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.
- na.rm a logical value indicating whether NA values should be stripped before the computation proceeds.
- ... further arguments passed to or from other methods.

Value

If `trim` is zero (the default), the arithmetic mean of the values in `x` is computed, as a numeric or complex vector of length one. If `x` is not logical (coerced to numeric), numeric (including integer) or complex, `NA_real_` is returned, with a warning.

If `trim` is non-zero, a symmetrically trimmed mean is computed with a fraction of `trim` observations

The function's name, and in the parentheses the named arguments it expects, in the order it expects them. an argument has a default value, it is shown. Arguments without default values (e.g. `x`) must be provided by you

The ellipsis allows other arguments to be passed to and from the function.

has to be—an object, a number, a data frame, a logical value, etc.

What the function returns—i.e., the result of whatever operation or calculation it performs. This can be a single number, as here, or a multi-part object such as a list, a data frame, a plot, or a model.

trim the fraction (0 to 0.5) of observations to be trimmed from each end of x before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.
na.rm a logical value indicating whether NA values should be stripped before the computation proceeds.
... further arguments passed to or from other methods.

The ellipsis allows other arguments to be passed to and from the function.

Value

If `trim` is zero (the default), the arithmetic mean of the values in x is computed, as a numeric or complex vector of length one. If x is not logical (coerced to numeric), numeric (including integer) or complex, `NA_real_` is returned, with a warning.

If `trim` is non-zero, a symmetrically trimmed mean is computed with a fraction of `trim` observations deleted from each end before the mean is computed.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

[weighted.mean](#), [mean.POSIXct](#), [colMeans](#) for row and column means.

Other related functions

Examples

```
x <- c(0:10, 50)
xm <- mean(x)
c(xm, mean(x, trim = 0.10))
```

Self-contained examples that you can run at the console. These may use built-in datasets or other R functions.

If `trim` is non-zero, a symmetrically trimmed mean is computed with a fraction of `trim` observations deleted from each end before the mean is computed.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

[weighted.mean](#), [mean.POSIXct](#), [colMeans](#) for row and column means.

Other related functions

Examples

```
x <- c(0:10, 50)
xm <- mean(x)
c(xm, mean(x, trim = 0.10))
```

Self-contained examples that you can run at the console. These may use built-in datasets or other R functions.

[Package *base* version 3.4.3 [Index](#)]

Visit the package's Index page to look for Demos and Vignettes detailing how it works.