# Plotting and Visualization in R

*PS239T*

*Spring 2018*

> "Visualizing your data should be the beginning and end of every project."
>
> "Think–what is the story you're trying to tell? This picture **should** be worth a thousand words."

## Plotting in R

There are two major sets of tools for creating plots in R:

- 1. base, which come with all R installations. These are mostly a quick and dirty tool–they're often hard to read and hard to modify, but can be useful if you need to whip up a quick histogram or something similar.
- 2. ggplot2, a stand-alone package. This is going to be our primary tool for examining our data visually. It's very powerful and makes for beautiful visualizations, but the number of options makes for a steeper learning curve at first.

Note that other plotting facilities do exist (notably **lattice**), but base and ggplot2 are by far the most popular.

#### setup environment

```
# remove all objects
rm(list=ls())

# set working directory
setwd(dir="~/YOURPATHHERE/PS239T/05_r-visualization/")
```

#### The dataset

For the following examples, we will using the gapminder dataset we saw last week. Gapminder is a country-year dataset with information on life expectancy, among other things.

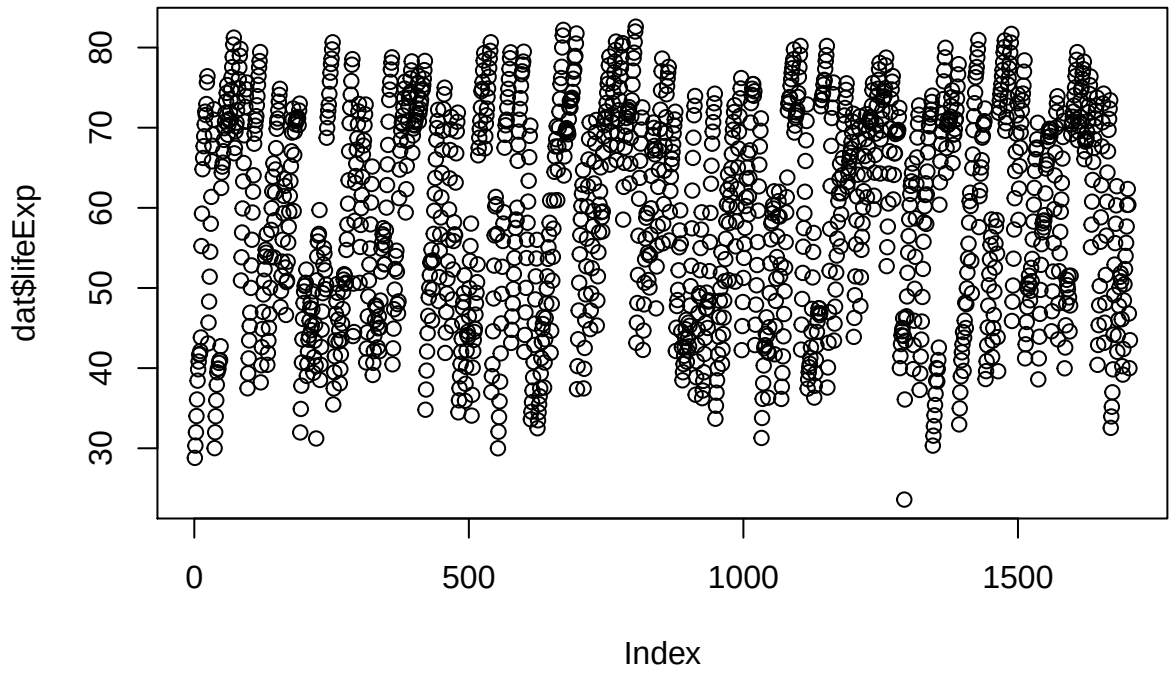
```
dat <- read.csv("data/gapminder-FiveYearData.csv")
```

## 1. R base graphics

- *Minimal* call takes the following form

```
plot(x=)
```

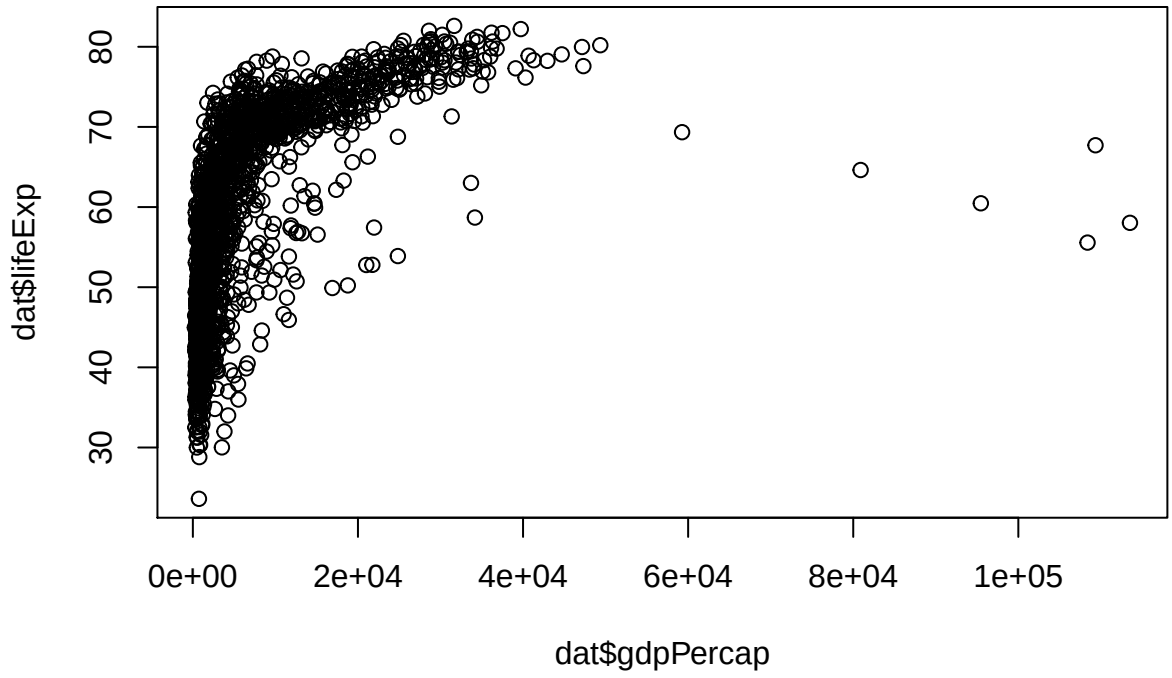
```
# Note that when asked to plot a single vector, R will assume the index positions of each vector elemen
plot(x = dat$lifeExp)
```




- *Basic* call takes the following form

```
plot(x=, y=)
```

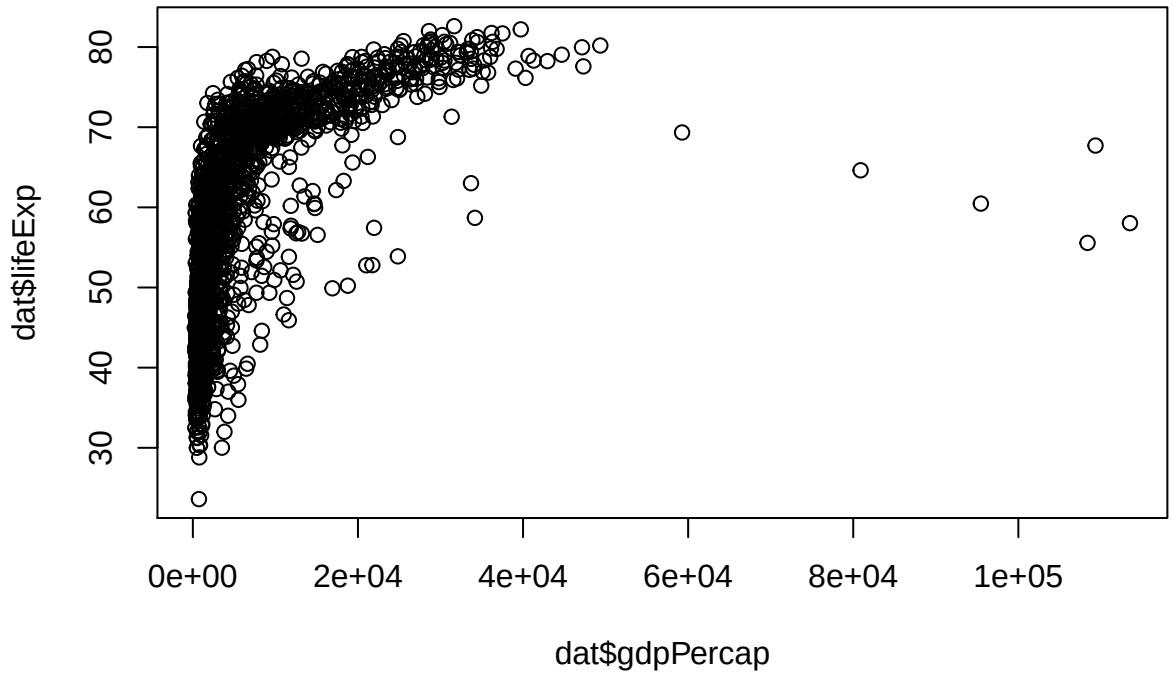
```
plot(x = dat$gdpPercap, y = dat$lifeExp)
```




### 1a. Scatter and Line Plots

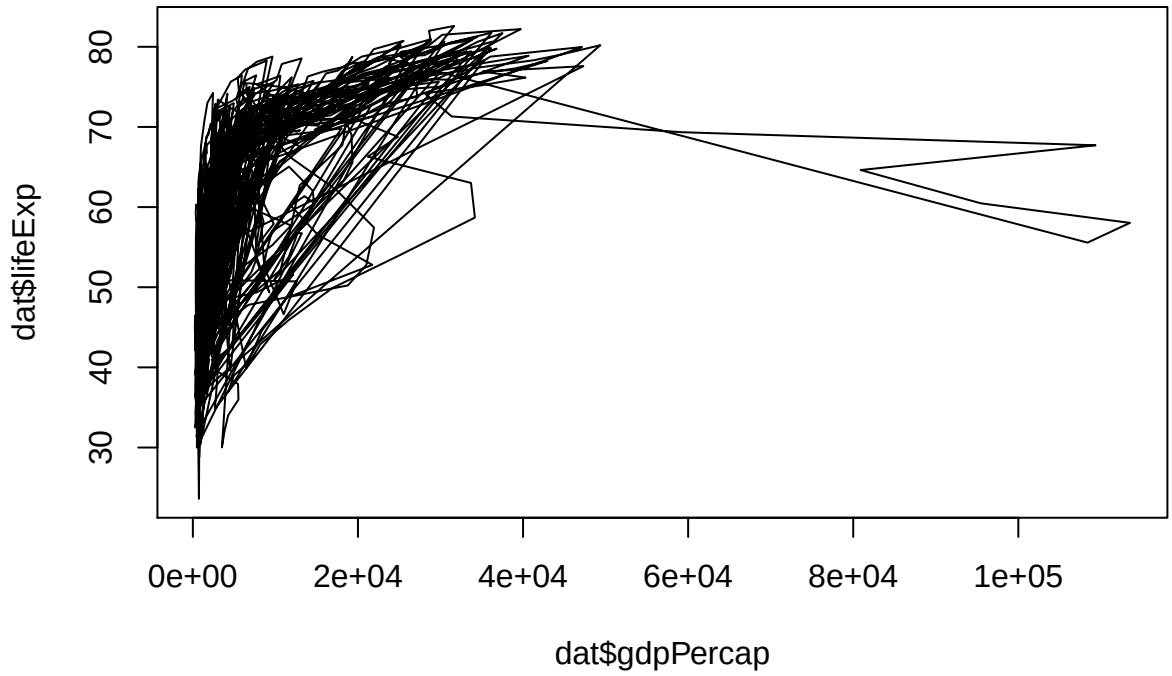
- The "type" argument accepts the following character indicators
- "p" – point/scatter plots (default plotting behavior)

```
plot(x = dat$gdpPercap, y = dat$lifeExp, type="p")
```



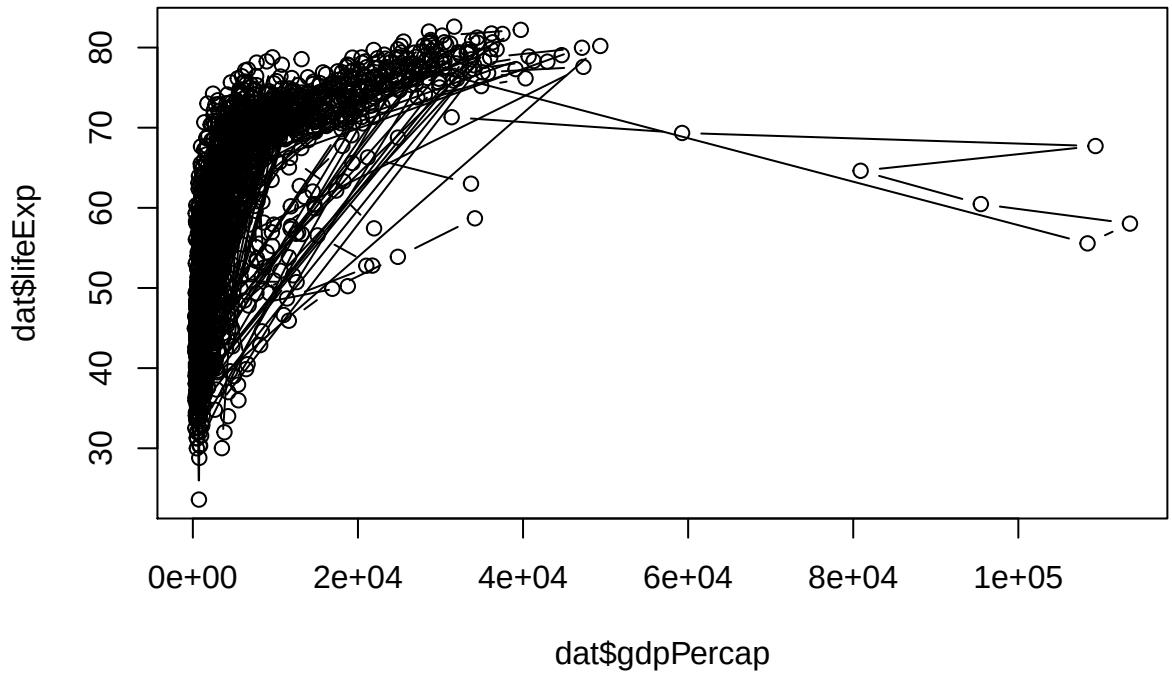

* “l” – line graphs

```
# Note that "line" does not create a smoothing line, just connected points
plot(x = dat$gdpPercap, y = dat$lifeExp, type="l")
```




* “b” – both line and point plots

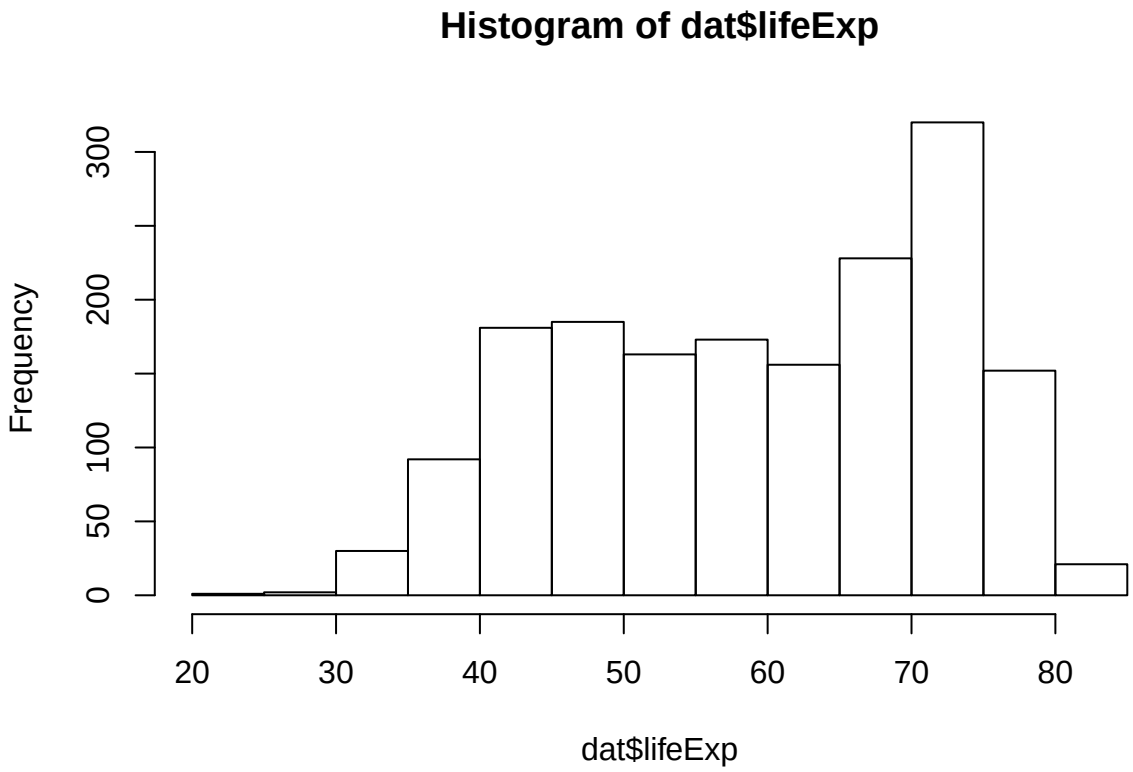
```
plot(x = dat$gdpPercap, y = dat$lifeExp, type="b")
```



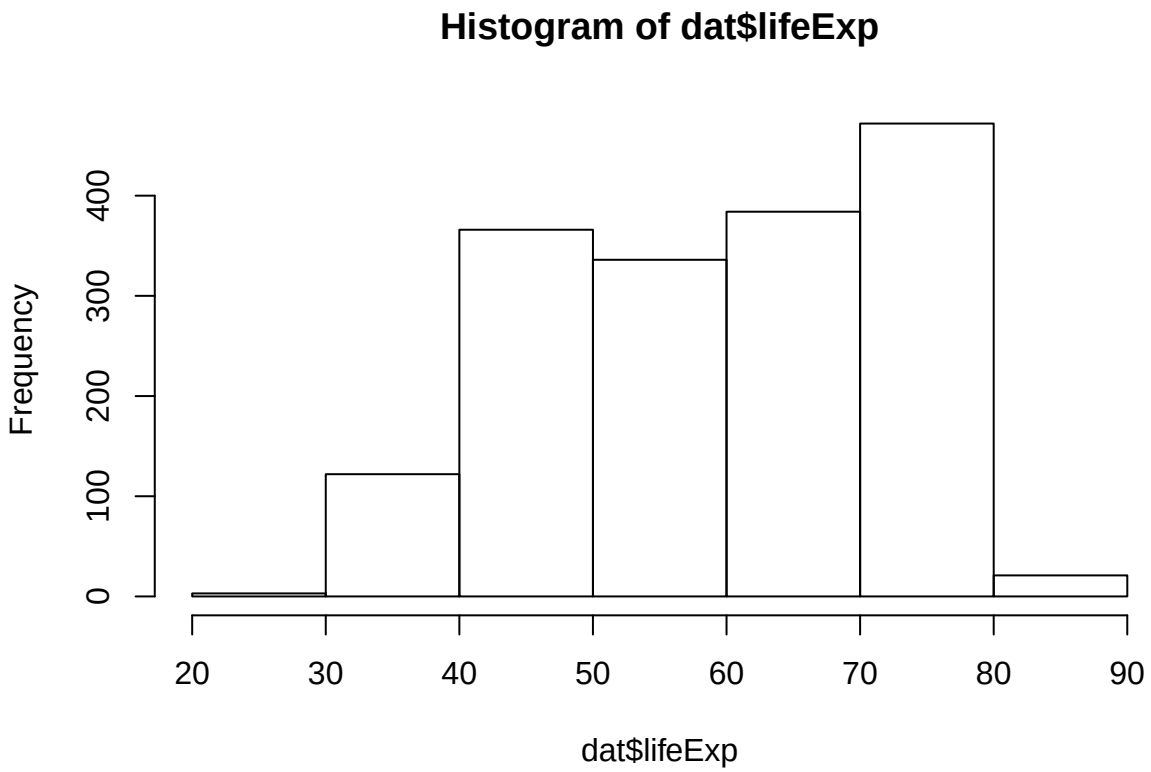

**1b. Histograms and Density Plots**

- Certain plot types require different calls outside of the "type" argument
- Ex) Histograms

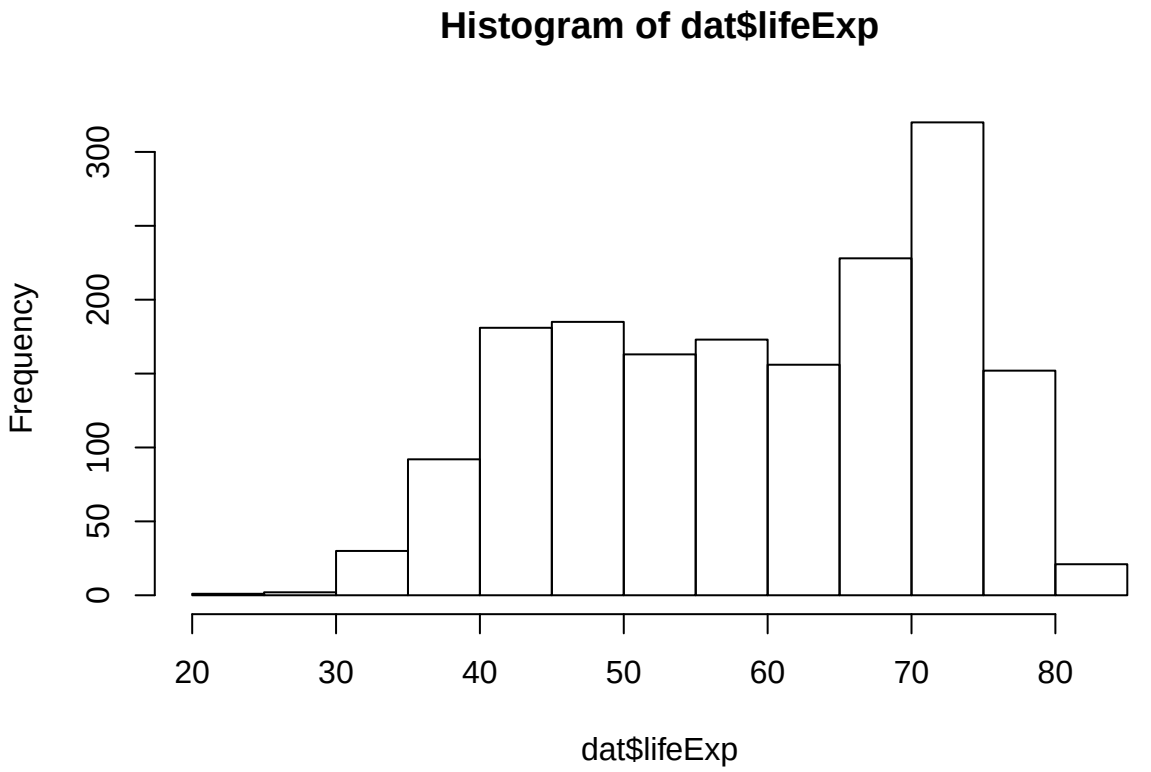
```
hist(x=dat$lifeExp)
```



```r
hist(x=dat$lifeExp, breaks=5)
```




```r
hist(x=dat$lifeExp, breaks=10)
```




- Ex) Density plots

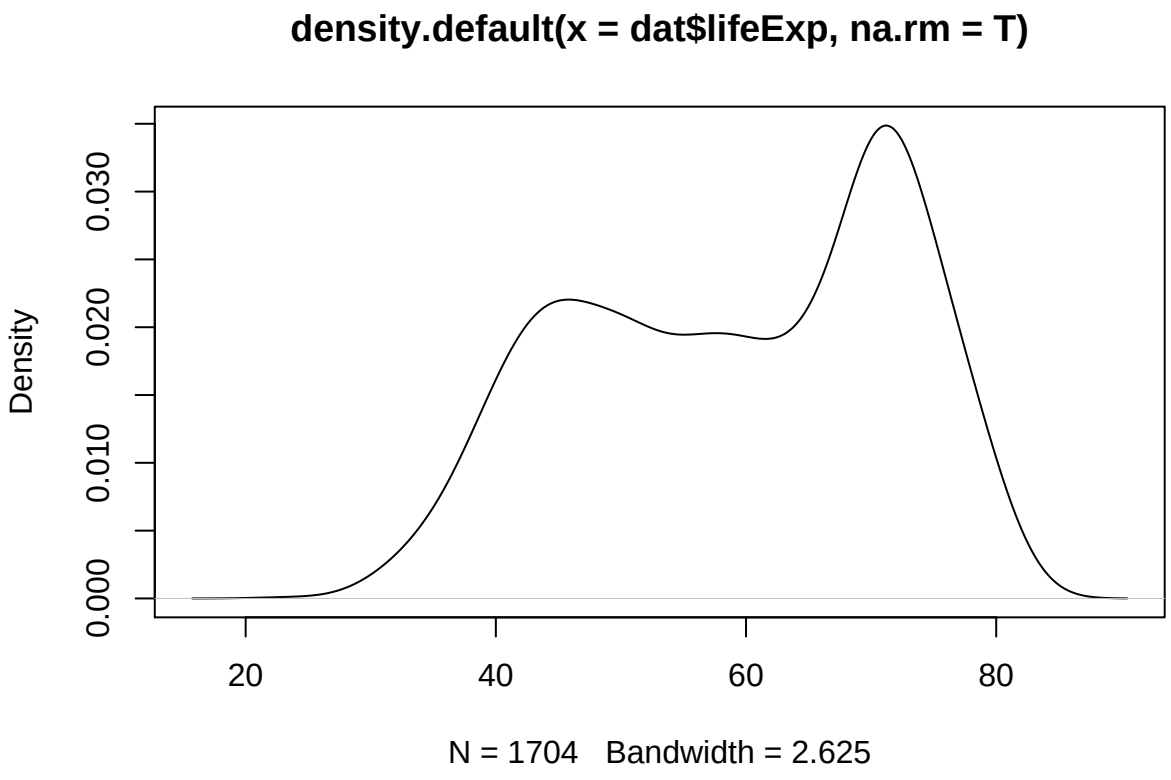
```r
# Create a density object (NOTE: be sure to remove missing values)
age.density<-density(x=dat$lifeExp, na.rm=T)
# Check the class of the object
class(age.density)
```

```
## [1] "density"
```

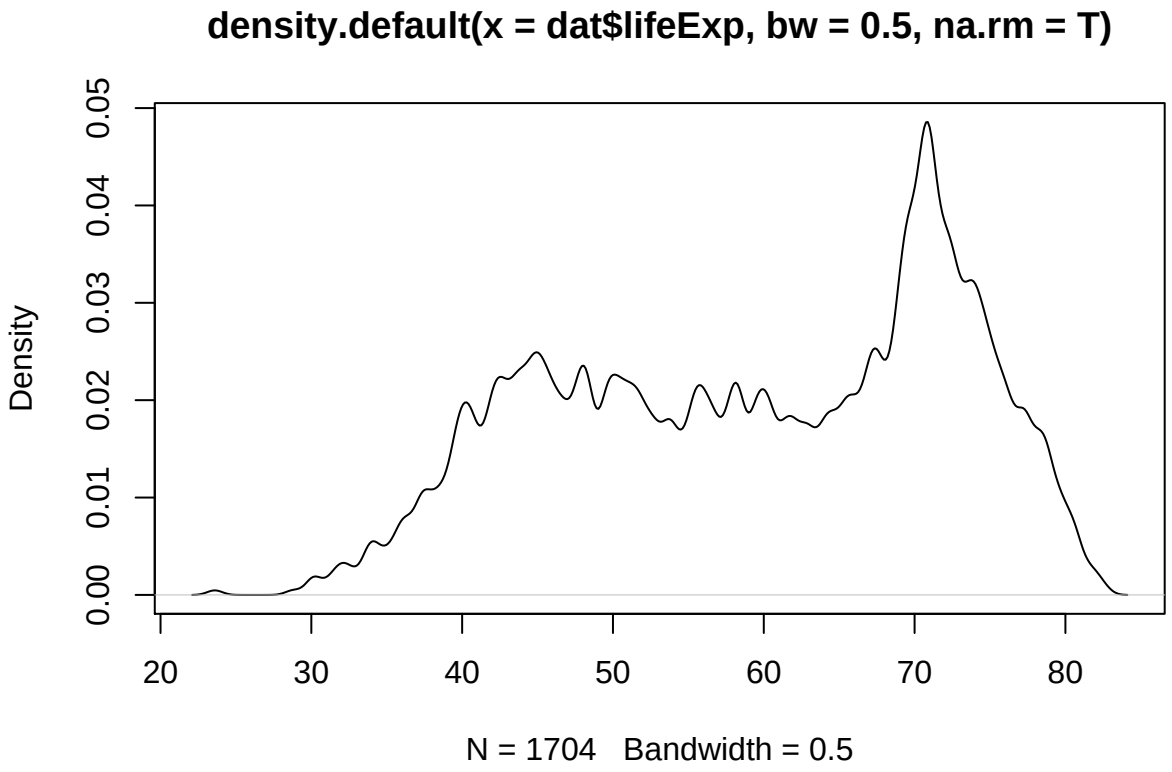
```r
# View the contents of the object
age.density
```

```
##
## Call:
##  density.default(x = dat$lifeExp, na.rm = T)
##
## Data: dat$lifeExp (1704 obs.);   Bandwidth 'bw' = 2.625
##
##        x                y
##  Min.   :15.72   Min.   :0.000001
##  1st Qu.:34.41   1st Qu.:0.001079
##  Median :53.10   Median :0.017034
##  Mean   :53.10   Mean   :0.013364
##  3rd Qu.:71.79   3rd Qu.:0.020987
##  Max.   :90.48   Max.   :0.034870
```

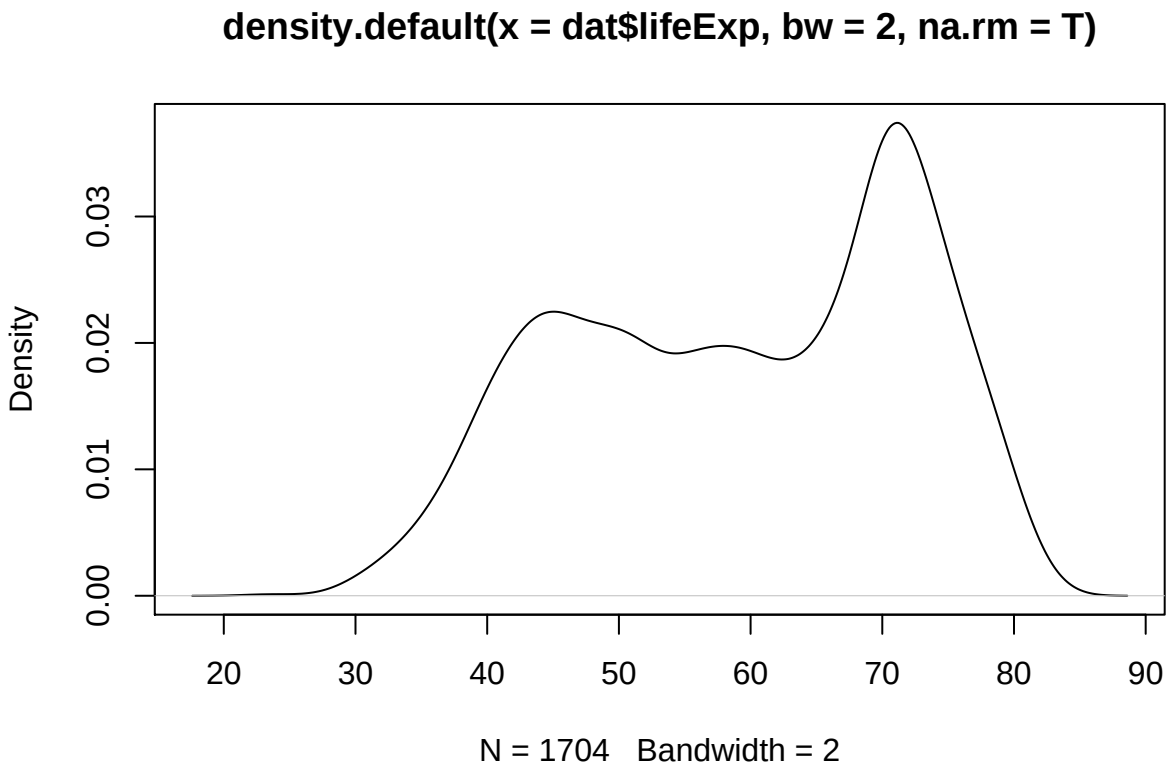
```r
# Plot the density object
plot(x=age.density)
```



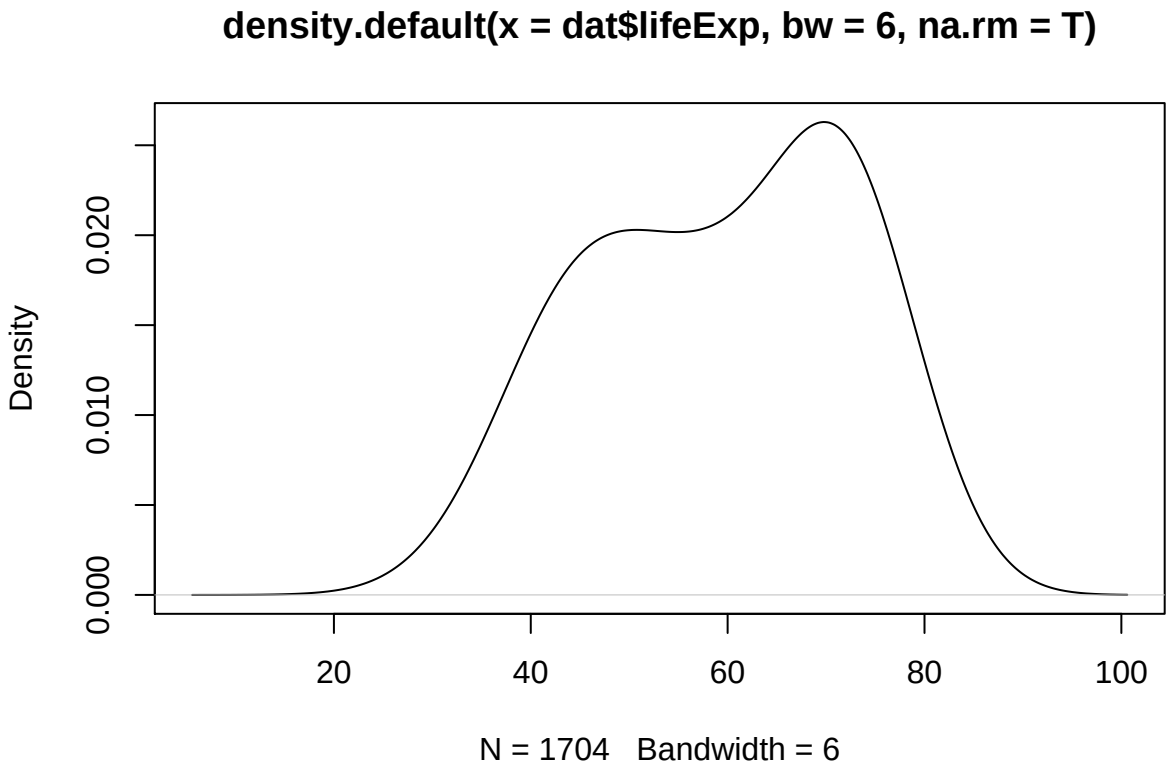

```r
# Plot the density object, bandwidth of 0.5
plot(x=density(x=dat$lifeExp, bw=.5, na.rm=T))
```




```r
# Plot the density object, bandwidth of 2
plot(x=density(x=dat$lifeExp, bw=2, na.rm=T))
```




```r
# Plot the density object, bandwidth of 6
plot(x=density(x=dat$lifeExp, bw=6, na.rm=T))
```




**1c. Labels**

- Basic call with popular labeling arguments

```
plot(x=, y=, type="", xlab="", ylab="", main="")
```

- From the previous example

```
plot(x = dat$gdpPercap, y = dat$lifeExp, type="p", xlab="GDP per cap", ylab="Life Expectancy", main="Li
```


### 1d. Axis and size scaling

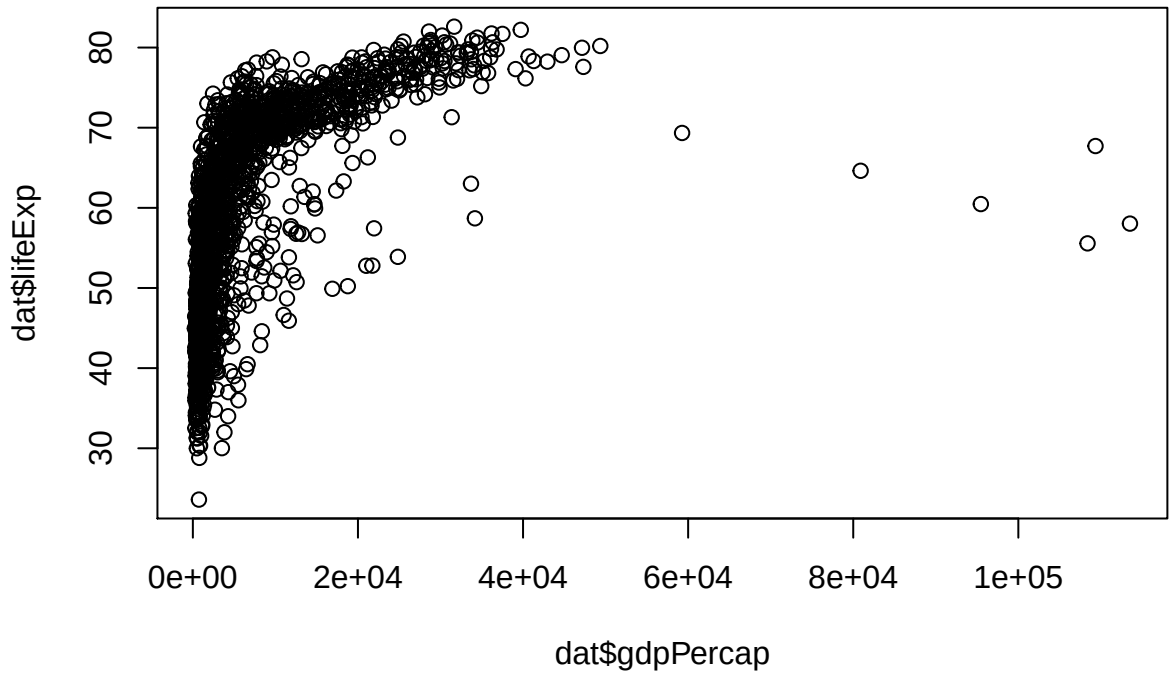
Currently it's hard to see the relationship between the points due to some strong outliers in GDP per capita. We can change the scale of units on the x axis using scaling arguments.

- Basic call with popular scaling arguments

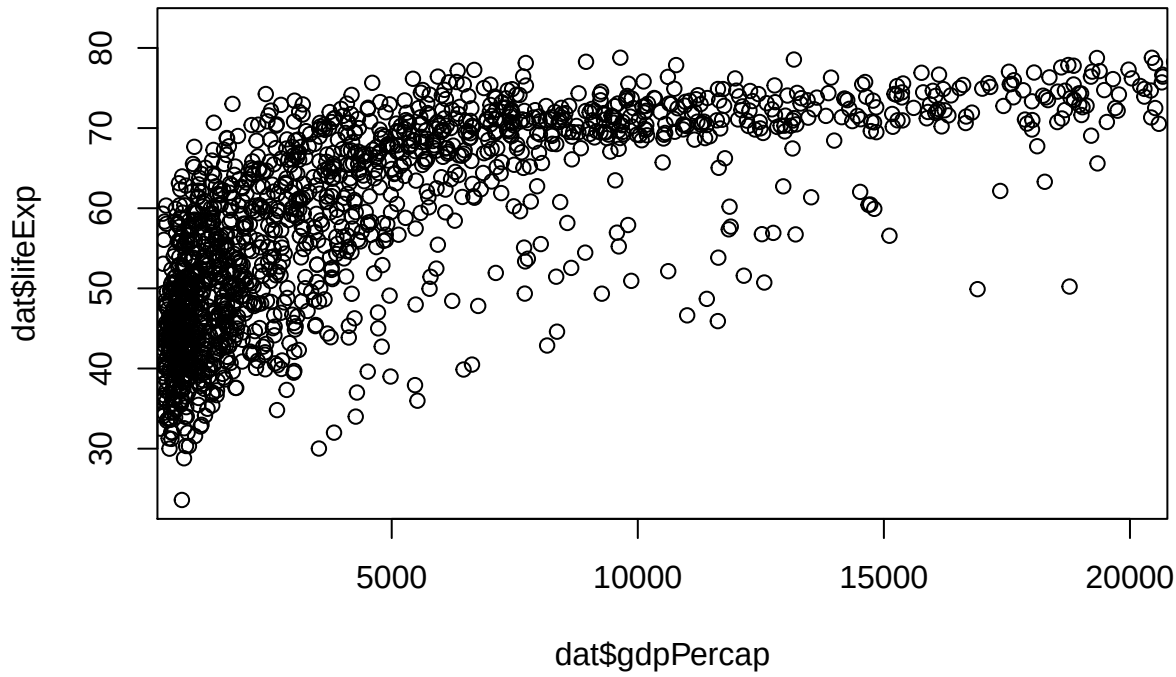
```
plot(x=, y=, type="", xlim=, ylim=, cex=)
```

- From the previous example

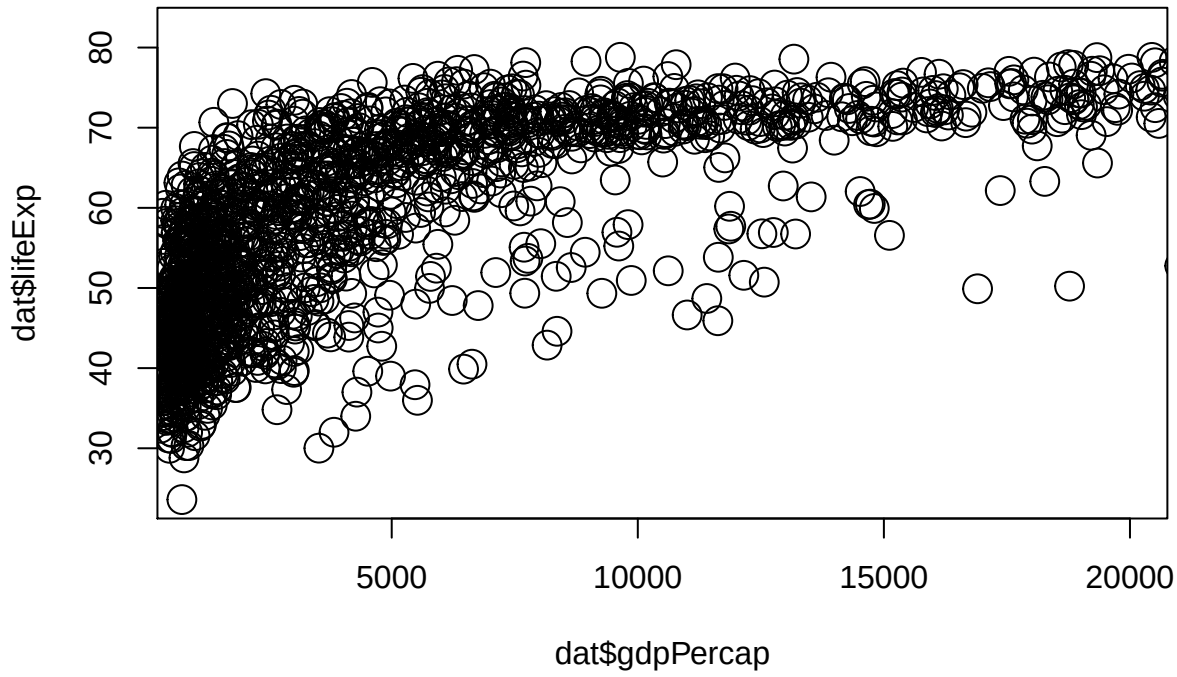
```
# Create a basic plot
plot(x = dat$gdpPercap, y = dat$lifeExp, type="p")
```



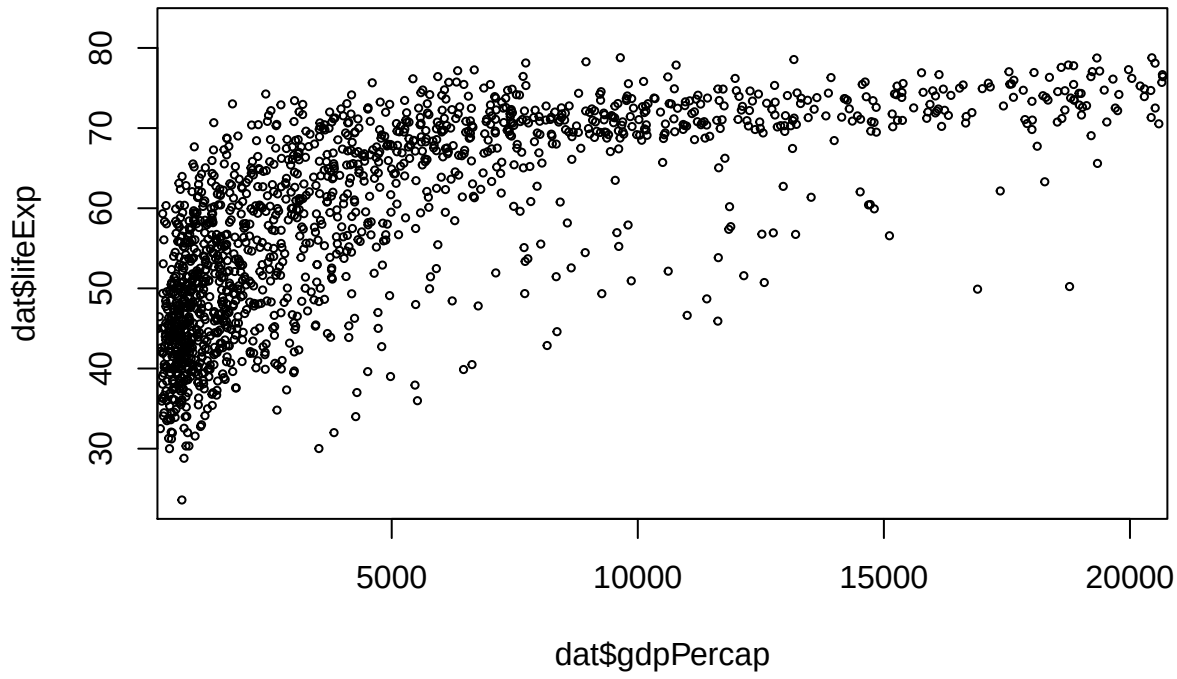

```
# Limit gdp (x-axis) to between 1,000 and 20,000
plot(x = dat$gdpPercap, y = dat$lifeExp, xlim = c(1000,20000))
```




```
# Limit gdp (x-axis) to between 1,000 and 20,000, increase point size to 2
plot(x = dat$gdpPercap, y = dat$lifeExp, xlim = c(1000,20000), cex=2)
```




```
# Limit gdp (x-axis) to between 1,000 and 20,000, decrease point size to 0.5
plot(x = dat$gdpPercap, y = dat$lifeExp, xlim = c(1000,20000), cex=0.5)
```




### 1e. Graphical parameters

- Basic call with popular scaling arguments

```
plot(x=, y=, type="", col="", pch=, lty=, lwd=)
```

- Colors

```r
colors() # View all elements of the color vector
```

```
##   [1] "white"                "aliceblue"            "antiquewhite"
##   [4] "antiquewhite1"        "antiquewhite2"        "antiquewhite3"
##   [7] "antiquewhite4"        "aquamarine"           "aquamarine1"
##  [10] "aquamarine2"          "aquamarine3"          "aquamarine4"
##  [13] "azure"                "azure1"               "azure2"
##  [16] "azure3"               "azure4"               "beige"
##  [19] "bisque"               "bisque1"              "bisque2"
##  [22] "bisque3"              "bisque4"              "black"
##  [25] "blanchedalmond"       "blue"                 "blue1"
##  [28] "blue2"                "blue3"                "blue4"
##  [31] "blueviolet"           "brown"                "brown1"
##  [34] "brown2"               "brown3"               "brown4"
##  [37] "burlywood"            "burlywood1"           "burlywood2"
##  [40] "burlywood3"           "burlywood4"           "cadetblue"
##  [43] "cadetblue1"           "cadetblue2"           "cadetblue3"
##  [46] "cadetblue4"           "chartreuse"           "chartreuse1"
##  [49] "chartreuse2"          "chartreuse3"          "chartreuse4"
##  [52] "chocolate"            "chocolate1"           "chocolate2"
##  [55] "chocolate3"           "chocolate4"           "coral"
##  [58] "coral1"               "coral2"               "coral3"
##  [61] "coral4"               "cornflowerblue"       "cornsilk"
##  [64] "cornsilk1"            "cornsilk2"            "cornsilk3"
##  [67] "cornsilk4"            "cyan"                 "cyan1"
##  [70] "cyan2"                "cyan3"                "cyan4"
##  [73] "darkblue"             "darkcyan"             "darkgoldenrod"
##  [76] "darkgoldenrod1"       "darkgoldenrod2"       "darkgoldenrod3"
##  [79] "darkgoldenrod4"       "darkgray"             "darkgreen"
##  [82] "darkgrey"             "darkkhaki"            "darkmagenta"
##  [85] "darkolivegreen"       "darkolivegreen1"      "darkolivegreen2"
##  [88] "darkolivegreen3"      "darkolivegreen4"      "darkorange"
##  [91] "darkorange1"          "darkorange2"          "darkorange3"
##  [94] "darkorange4"          "darkorchid"           "darkorchid1"
##  [97] "darkorchid2"          "darkorchid3"          "darkorchid4"
## [100] "darkred"              "darksalmon"           "darkseagreen"
## [103] "darkseagreen1"        "darkseagreen2"        "darkseagreen3"
## [106] "darkseagreen4"        "darkslateblue"        "darkslategray"
## [109] "darkslategray1"       "darkslategray2"       "darkslategray3"
## [112] "darkslategray4"       "darkslategrey"        "darkturquoise"
## [115] "darkviolet"           "deeppink"             "deeppink1"
## [118] "deeppink2"            "deeppink3"            "deeppink4"
## [121] "deepskyblue"          "deepskyblue1"         "deepskyblue2"
## [124] "deepskyblue3"         "deepskyblue4"         "dimgray"
## [127] "dimgrey"              "dodgerblue"           "dodgerblue1"
## [130] "dodgerblue2"          "dodgerblue3"          "dodgerblue4"
## [133] "firebrick"            "firebrick1"           "firebrick2"
## [136] "firebrick3"           "firebrick4"           "floralwhite"
## [139] "forestgreen"          "gainsboro"            "ghostwhite"
## [142] "gold"                 "gold1"                "gold2"
## [145] "gold3"                "gold4"                "goldenrod"
## [148] "goldenrod1"           "goldenrod2"           "goldenrod3"
## [151] "goldenrod4"           "gray"                 "gray0"
## [154] "gray1"                "gray2"                "gray3"
```

```
## [157] "gray4"              "gray5"              "gray6"
## [160] "gray7"              "gray8"              "gray9"
## [163] "gray10"             "gray11"             "gray12"
## [166] "gray13"             "gray14"             "gray15"
## [169] "gray16"             "gray17"             "gray18"
## [172] "gray19"             "gray20"             "gray21"
## [175] "gray22"             "gray23"             "gray24"
## [178] "gray25"             "gray26"             "gray27"
## [181] "gray28"             "gray29"             "gray30"
## [184] "gray31"             "gray32"             "gray33"
## [187] "gray34"             "gray35"             "gray36"
## [190] "gray37"             "gray38"             "gray39"
## [193] "gray40"             "gray41"             "gray42"
## [196] "gray43"             "gray44"             "gray45"
## [199] "gray46"             "gray47"             "gray48"
## [202] "gray49"             "gray50"             "gray51"
## [205] "gray52"             "gray53"             "gray54"
## [208] "gray55"             "gray56"             "gray57"
## [211] "gray58"             "gray59"             "gray60"
## [214] "gray61"             "gray62"             "gray63"
## [217] "gray64"             "gray65"             "gray66"
## [220] "gray67"             "gray68"             "gray69"
## [223] "gray70"             "gray71"             "gray72"
## [226] "gray73"             "gray74"             "gray75"
## [229] "gray76"             "gray77"             "gray78"
## [232] "gray79"             "gray80"             "gray81"
## [235] "gray82"             "gray83"             "gray84"
## [238] "gray85"             "gray86"             "gray87"
## [241] "gray88"             "gray89"             "gray90"
## [244] "gray91"             "gray92"             "gray93"
## [247] "gray94"             "gray95"             "gray96"
## [250] "gray97"             "gray98"             "gray99"
## [253] "gray100"            "green"              "green1"
## [256] "green2"             "green3"             "green4"
## [259] "greenyellow"        "grey"               "grey0"
## [262] "grey1"              "grey2"              "grey3"
## [265] "grey4"              "grey5"              "grey6"
## [268] "grey7"              "grey8"              "grey9"
## [271] "grey10"             "grey11"             "grey12"
## [274] "grey13"             "grey14"             "grey15"
## [277] "grey16"             "grey17"             "grey18"
## [280] "grey19"             "grey20"             "grey21"
## [283] "grey22"             "grey23"             "grey24"
## [286] "grey25"             "grey26"             "grey27"
## [289] "grey28"             "grey29"             "grey30"
## [292] "grey31"             "grey32"             "grey33"
## [295] "grey34"             "grey35"             "grey36"
## [298] "grey37"             "grey38"             "grey39"
## [301] "grey40"             "grey41"             "grey42"
## [304] "grey43"             "grey44"             "grey45"
## [307] "grey46"             "grey47"             "grey48"
## [310] "grey49"             "grey50"             "grey51"
## [313] "grey52"             "grey53"             "grey54"
## [316] "grey55"             "grey56"             "grey57"
```

```
## [319] "grey58"               "grey59"               "grey60"
## [322] "grey61"               "grey62"               "grey63"
## [325] "grey64"               "grey65"               "grey66"
## [328] "grey67"               "grey68"               "grey69"
## [331] "grey70"               "grey71"               "grey72"
## [334] "grey73"               "grey74"               "grey75"
## [337] "grey76"               "grey77"               "grey78"
## [340] "grey79"               "grey80"               "grey81"
## [343] "grey82"               "grey83"               "grey84"
## [346] "grey85"               "grey86"               "grey87"
## [349] "grey88"               "grey89"               "grey90"
## [352] "grey91"               "grey92"               "grey93"
## [355] "grey94"               "grey95"               "grey96"
## [358] "grey97"               "grey98"               "grey99"
## [361] "grey100"              "honeydew"             "honeydew1"
## [364] "honeydew2"            "honeydew3"            "honeydew4"
## [367] "hotpink"              "hotpink1"             "hotpink2"
## [370] "hotpink3"             "hotpink4"             "indianred"
## [373] "indianred1"           "indianred2"           "indianred3"
## [376] "indianred4"           "ivory"                "ivory1"
## [379] "ivory2"               "ivory3"               "ivory4"
## [382] "khaki"                "khaki1"               "khaki2"
## [385] "khaki3"               "khaki4"               "lavender"
## [388] "lavenderblush"        "lavenderblush1"       "lavenderblush2"
## [391] "lavenderblush3"       "lavenderblush4"       "lawngreen"
## [394] "lemonchiffon"         "lemonchiffon1"        "lemonchiffon2"
## [397] "lemonchiffon3"        "lemonchiffon4"        "lightblue"
## [400] "lightblue1"           "lightblue2"           "lightblue3"
## [403] "lightblue4"           "lightcoral"           "lightcyan"
## [406] "lightcyan1"           "lightcyan2"           "lightcyan3"
## [409] "lightcyan4"           "lightgoldenrod"       "lightgoldenrod1"
## [412] "lightgoldenrod2"      "lightgoldenrod3"      "lightgoldenrod4"
## [415] "lightgoldenrodyellow" "lightgray"            "lightgreen"
## [418] "lightgrey"            "lightpink"            "lightpink1"
## [421] "lightpink2"           "lightpink3"           "lightpink4"
## [424] "lightsalmon"          "lightsalmon1"         "lightsalmon2"
## [427] "lightsalmon3"         "lightsalmon4"         "lightseagreen"
## [430] "lightskyblue"         "lightskyblue1"        "lightskyblue2"
## [433] "lightskyblue3"        "lightskyblue4"        "lightslateblue"
## [436] "lightslategray"       "lightslategrey"       "lightsteelblue"
## [439] "lightsteelblue1"      "lightsteelblue2"      "lightsteelblue3"
## [442] "lightsteelblue4"      "lightyellow"          "lightyellow1"
## [445] "lightyellow2"         "lightyellow3"         "lightyellow4"
## [448] "limegreen"            "linen"                "magenta"
## [451] "magenta1"             "magenta2"             "magenta3"
## [454] "magenta4"             "maroon"               "maroon1"
## [457] "maroon2"              "maroon3"              "maroon4"
## [460] "mediumaquamarine"     "mediumblue"           "mediumorchid"
## [463] "mediumorchid1"        "mediumorchid2"        "mediumorchid3"
## [466] "mediumorchid4"        "mediumpurple"         "mediumpurple1"
## [469] "mediumpurple2"        "mediumpurple3"        "mediumpurple4"
## [472] "mediumseagreen"       "mediumslateblue"      "mediumspringgreen"
## [475] "mediumturquoise"      "mediumvioletred"      "midnightblue"
## [478] "mintcream"            "mistyrose"            "mistyrose1"
```

```
## [481] "mistyrose2"          "mistyrose3"          "mistyrose4"
## [484] "moccasin"            "navajowhite"         "navajowhite1"
## [487] "navajowhite2"        "navajowhite3"        "navajowhite4"
## [490] "navy"                "navyblue"            "oldlace"
## [493] "olivedrab"           "olivedrab1"          "olivedrab2"
## [496] "olivedrab3"          "olivedrab4"          "orange"
## [499] "orange1"             "orange2"             "orange3"
## [502] "orange4"             "orangered"           "orangered1"
## [505] "orangered2"          "orangered3"          "orangered4"
## [508] "orchid"              "orchid1"             "orchid2"
## [511] "orchid3"             "orchid4"             "palegoldenrod"
## [514] "palegreen"           "palegreen1"          "palegreen2"
## [517] "palegreen3"          "palegreen4"          "paleturquoise"
## [520] "paleturquoise1"      "paleturquoise2"      "paleturquoise3"
## [523] "paleturquoise4"      "palevioletred"       "palevioletred1"
## [526] "palevioletred2"      "palevioletred3"      "palevioletred4"
## [529] "papayawhip"          "peachpuff"           "peachpuff1"
## [532] "peachpuff2"          "peachpuff3"          "peachpuff4"
## [535] "peru"                "pink"                "pink1"
## [538] "pink2"               "pink3"               "pink4"
## [541] "plum"                "plum1"               "plum2"
## [544] "plum3"               "plum4"               "powderblue"
## [547] "purple"              "purple1"             "purple2"
## [550] "purple3"             "purple4"             "red"
## [553] "red1"                "red2"                "red3"
## [556] "red4"                "rosybrown"           "rosybrown1"
## [559] "rosybrown2"          "rosybrown3"          "rosybrown4"
## [562] "royalblue"           "royalblue1"          "royalblue2"
## [565] "royalblue3"          "royalblue4"          "saddlebrown"
## [568] "salmon"              "salmon1"             "salmon2"
## [571] "salmon3"             "salmon4"             "sandybrown"
## [574] "seagreen"            "seagreen1"           "seagreen2"
## [577] "seagreen3"           "seagreen4"           "seashell"
## [580] "seashell1"           "seashell2"           "seashell3"
## [583] "seashell4"           "sienna"              "sienna1"
## [586] "sienna2"             "sienna3"             "sienna4"
## [589] "skyblue"             "skyblue1"            "skyblue2"
## [592] "skyblue3"            "skyblue4"            "slateblue"
## [595] "slateblue1"          "slateblue2"          "slateblue3"
## [598] "slateblue4"          "slategray"           "slategray1"
## [601] "slategray2"          "slategray3"          "slategray4"
## [604] "slategrey"           "snow"                "snow1"
## [607] "snow2"               "snow3"               "snow4"
## [610] "springgreen"         "springgreen1"        "springgreen2"
## [613] "springgreen3"        "springgreen4"        "steelblue"
## [616] "steelblue1"          "steelblue2"          "steelblue3"
## [619] "steelblue4"          "tan"                 "tan1"
## [622] "tan2"                "tan3"                "tan4"
## [625] "thistle"             "thistle1"            "thistle2"
## [628] "thistle3"            "thistle4"            "tomato"
## [631] "tomato1"             "tomato2"             "tomato3"
## [634] "tomato4"             "turquoise"           "turquoise1"
## [637] "turquoise2"          "turquoise3"          "turquoise4"
## [640] "violet"              "violetred"           "violetred1"
```

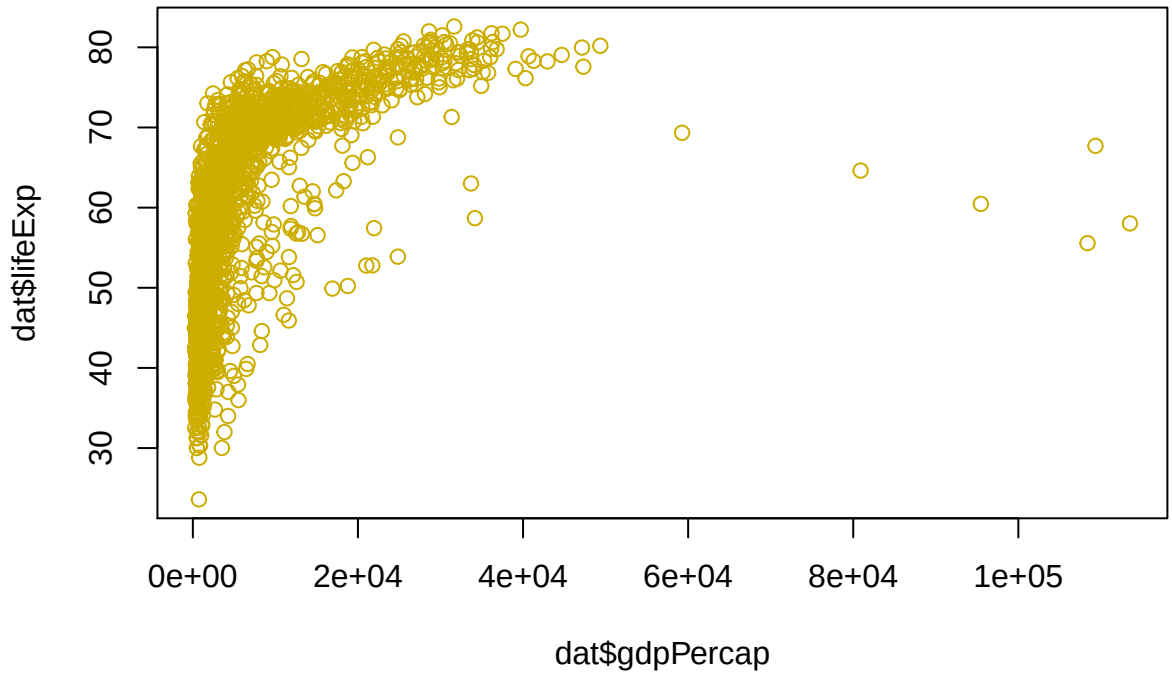
```
## [643] "violetred2"           "violetred3"           "violetred4"
## [646] "wheat"                "wheat1"               "wheat2"
## [649] "wheat3"               "wheat4"               "whitesmoke"
## [652] "yellow"               "yellow1"              "yellow2"
## [655] "yellow3"              "yellow4"              "yellowgreen"
```

```
colors()[179] # View specific element of the color vector
```

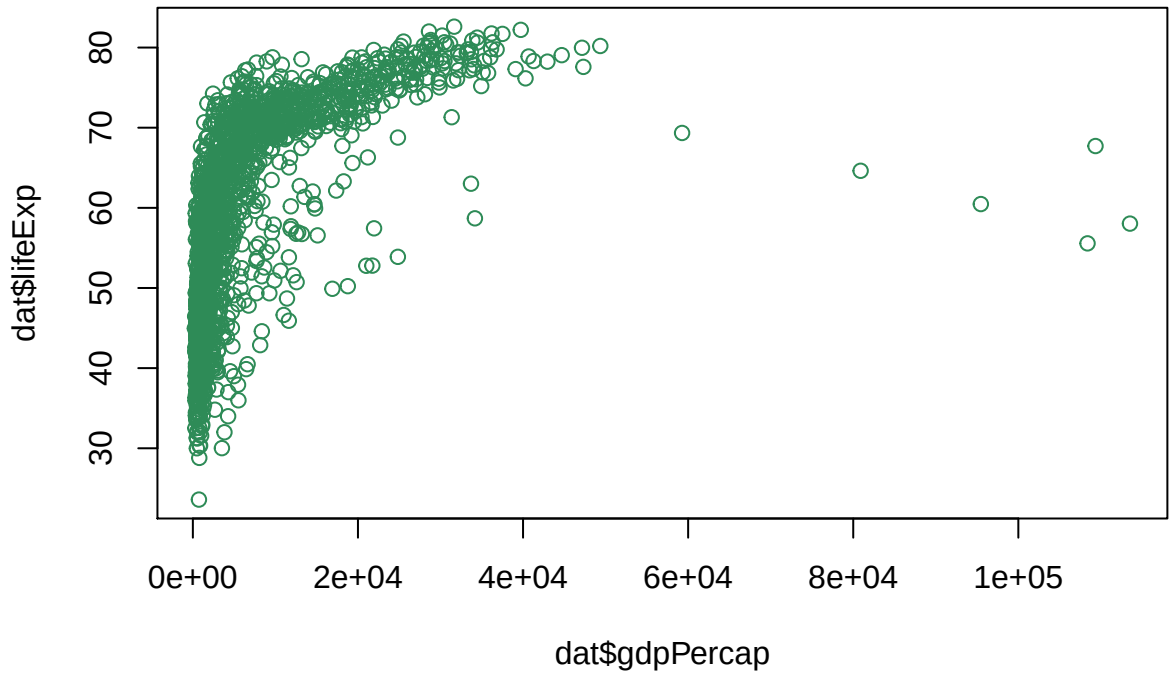
```
## [1] "gray26"
```

Another option: R Color Infographic

```
plot(x = dat$gdpPercap, y = dat$lifeExp, type="p", col=colors()[145]) # or col="gold3"
```



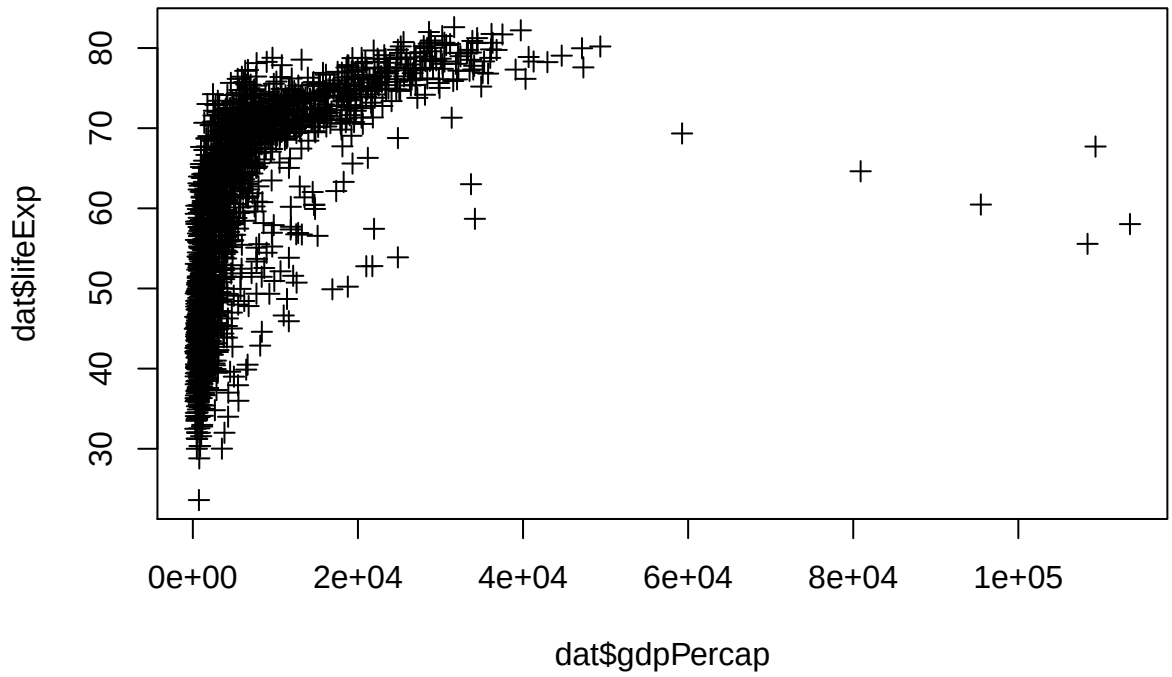

```
plot(x = dat$gdpPercap, y = dat$lifeExp, type="p", col="seagreen4") # or col=colors()[578]
```



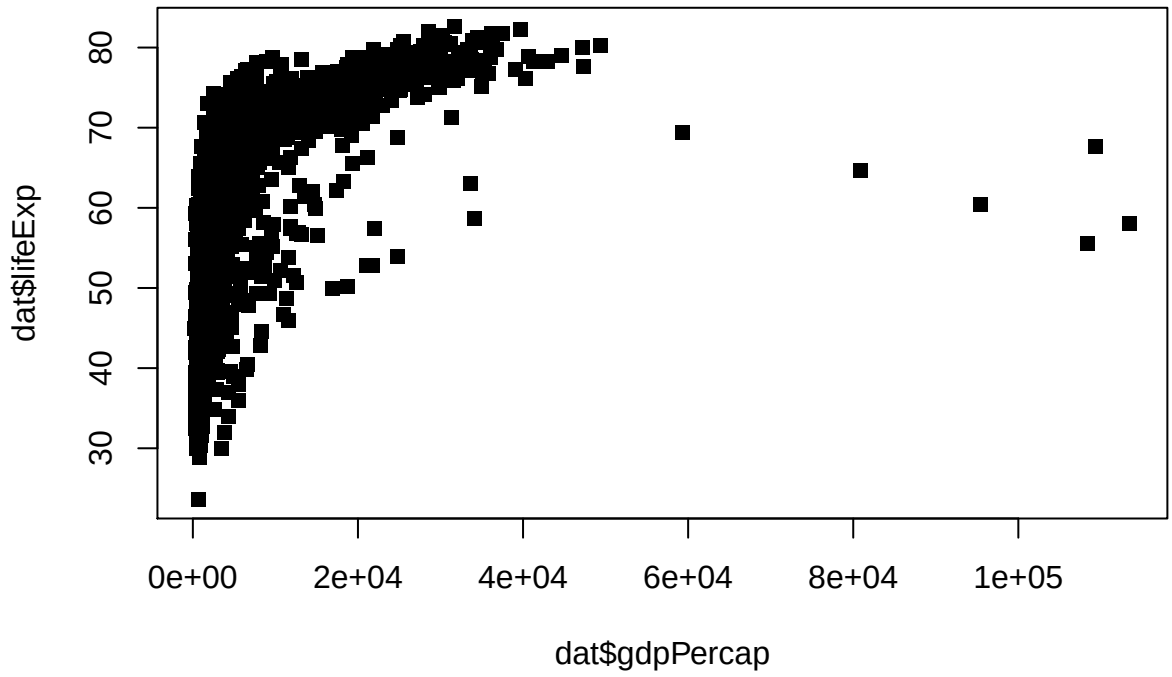

- Point Styles and Widths

A Good Reference

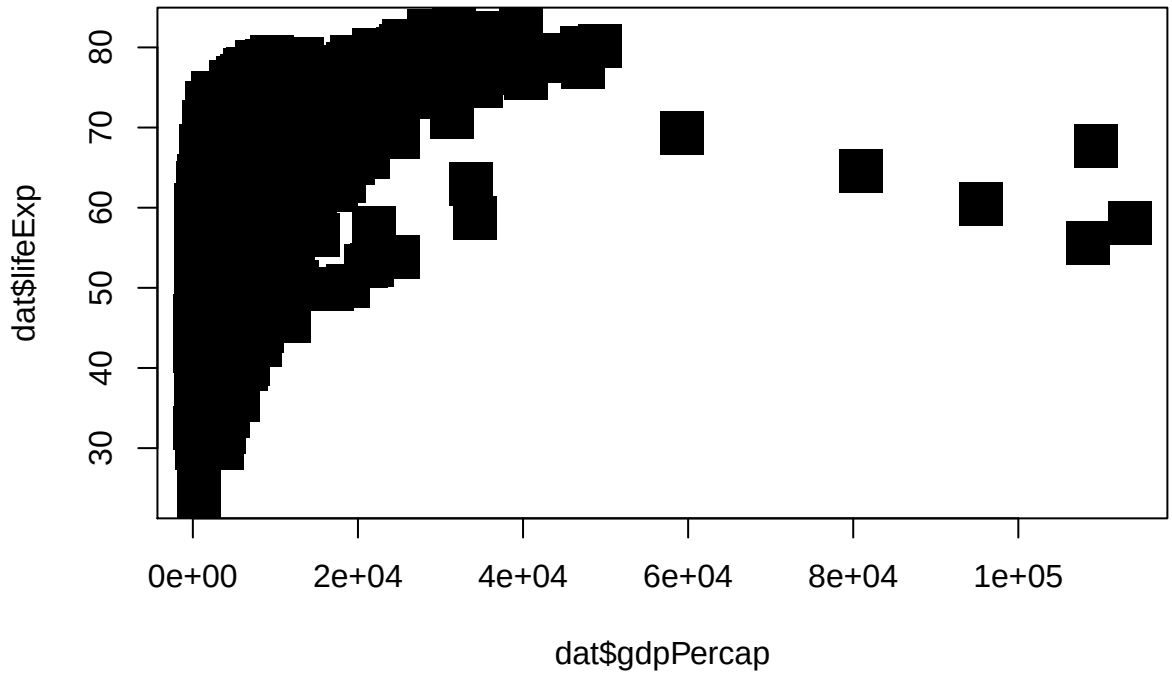
```
# Change point style to crosses
plot(x = dat$gdpPercap, y = dat$lifeExp, type="p", pch=3)
```



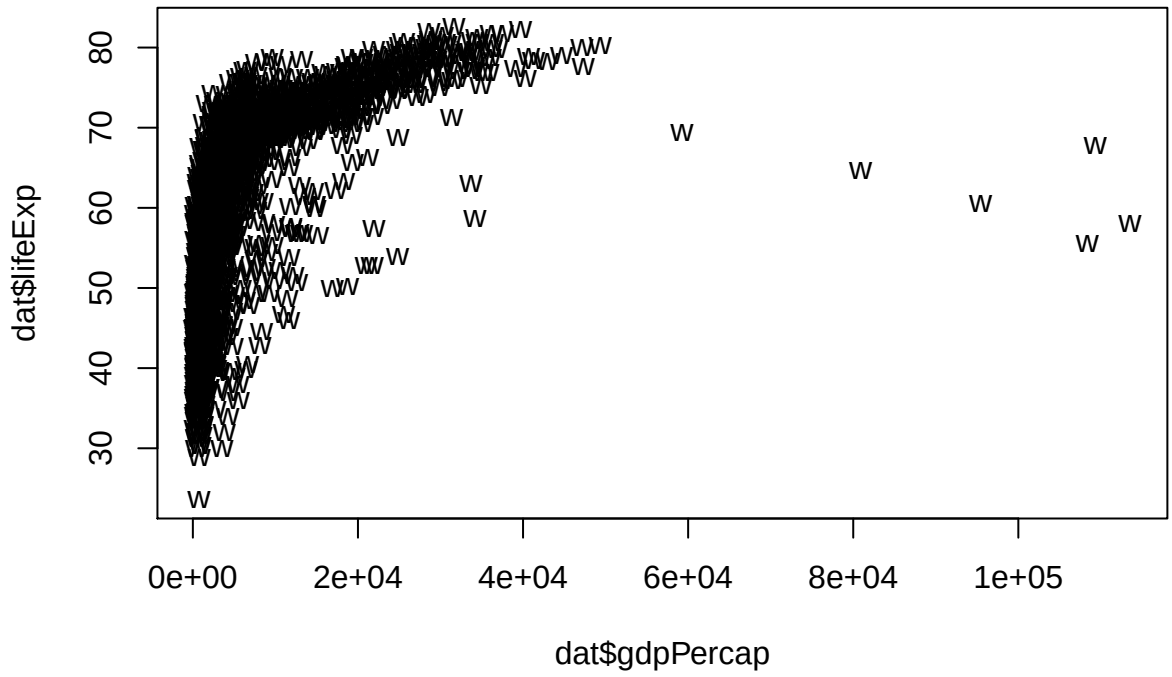

```
# Change point style to filled squares
plot(x = dat$gdpPercap, y = dat$lifeExp, type="p",pch=15)
```



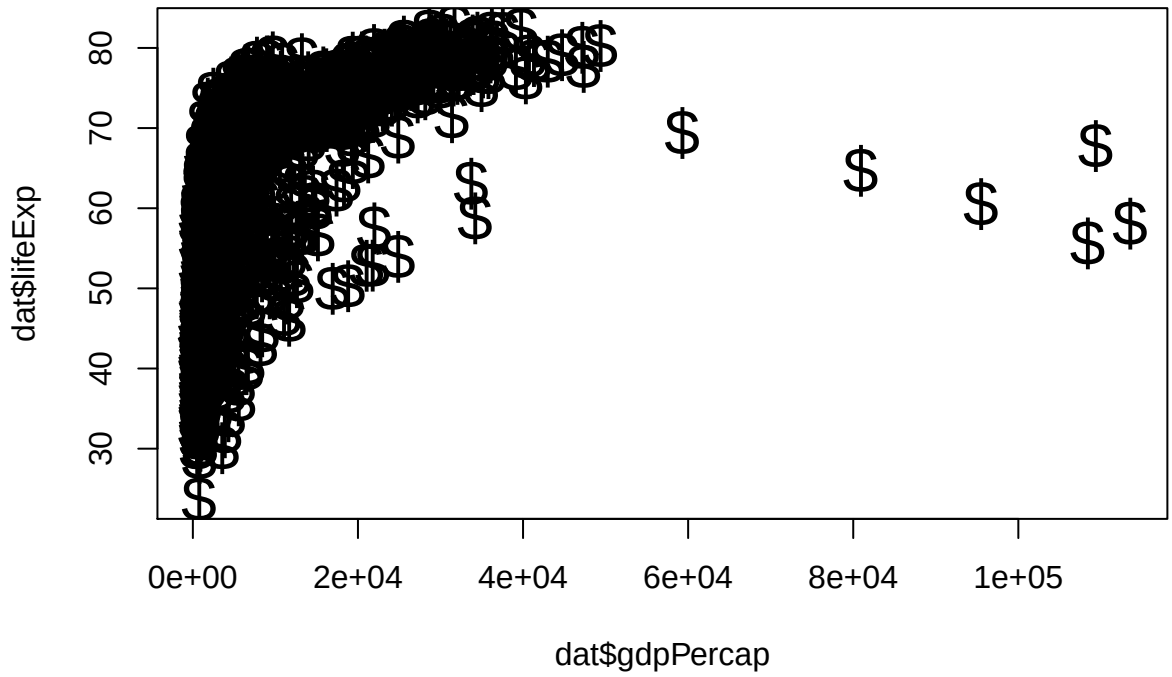

```r
# Change point style to filled squares and increase point size to 3
plot(x = dat$gdpPercap, y = dat$lifeExp, type="p",pch=15, cex=3)
```




```r
# Change point style to "w"
plot(x = dat$gdpPercap, y = dat$lifeExp, type="p", pch="w")
```

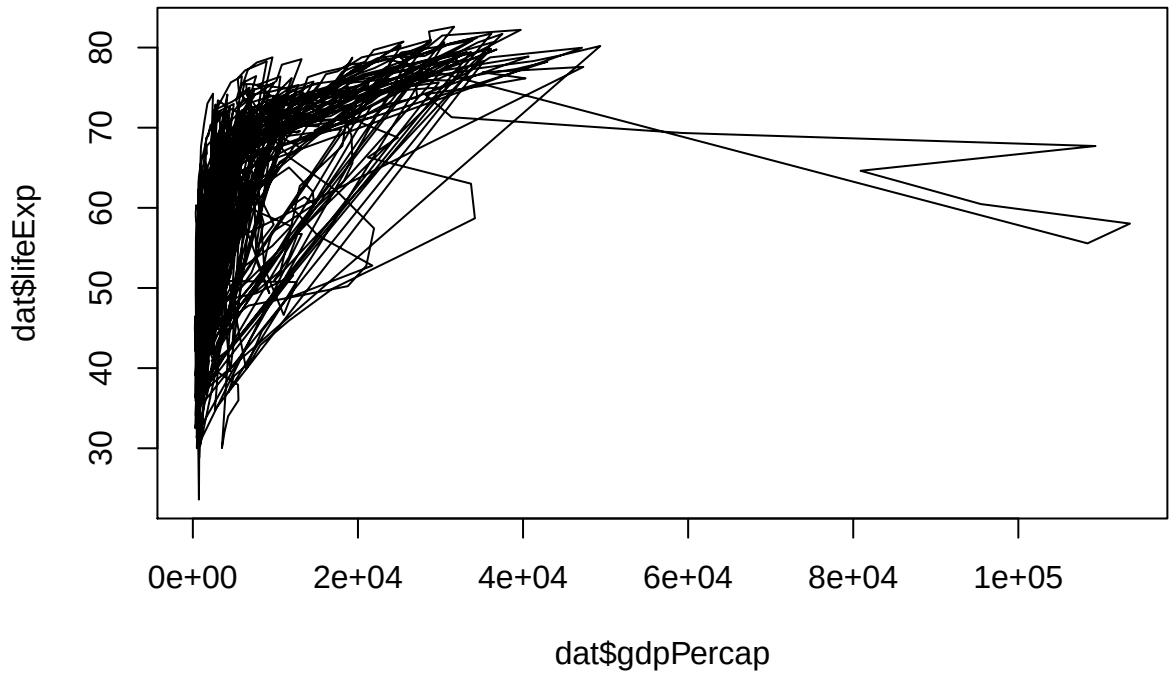



```
# Change point style to "$" and increase point size to 2
plot(x = dat$gdpPercap, y = dat$lifeExp, type="p",pch="$", cex=2)
```

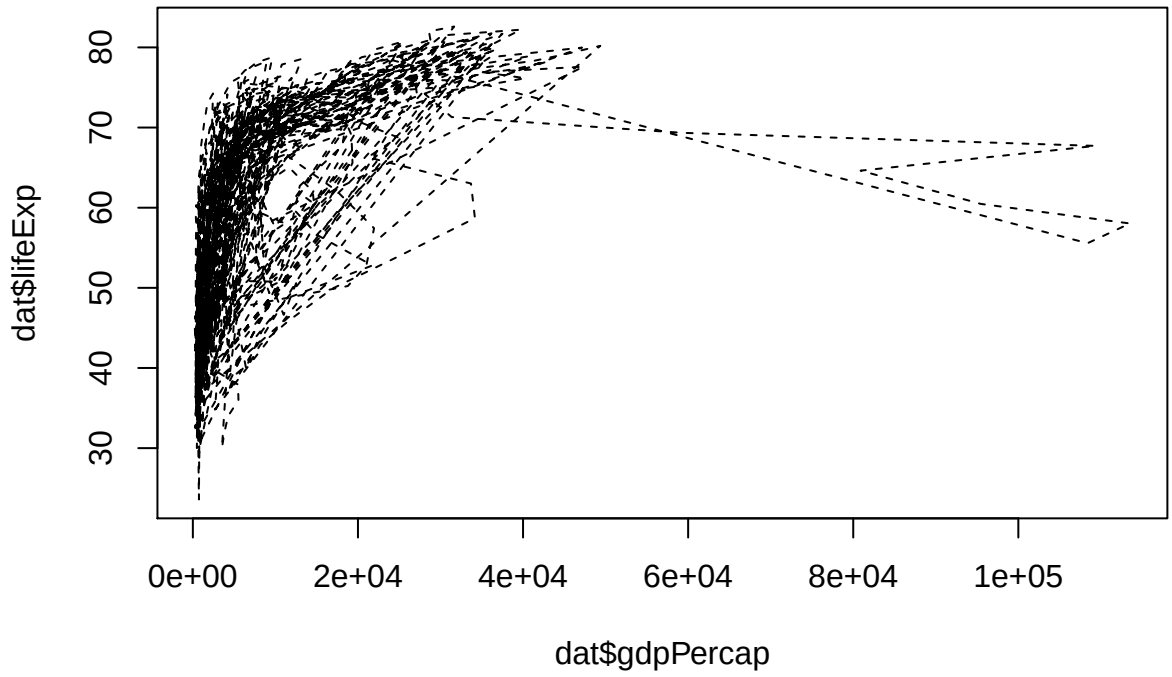



- Line Styles and Widths

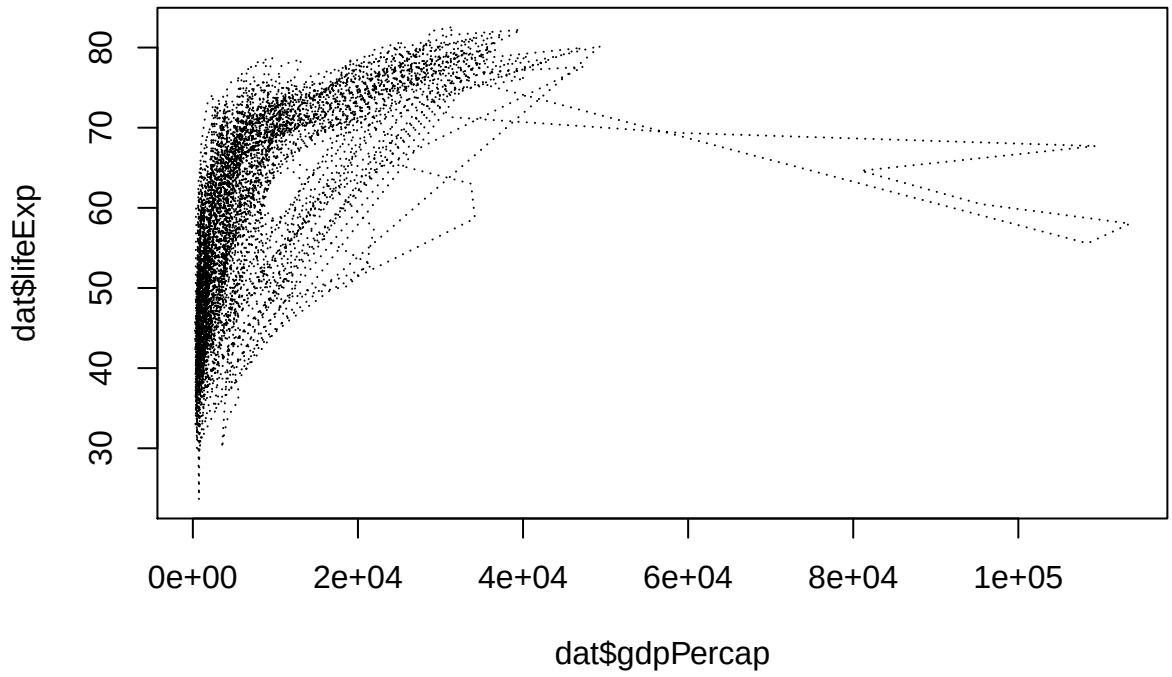
```
# Line plot with solid line
plot(x = dat$gdpPercap, y = dat$lifeExp, type="l", lty=1)
```



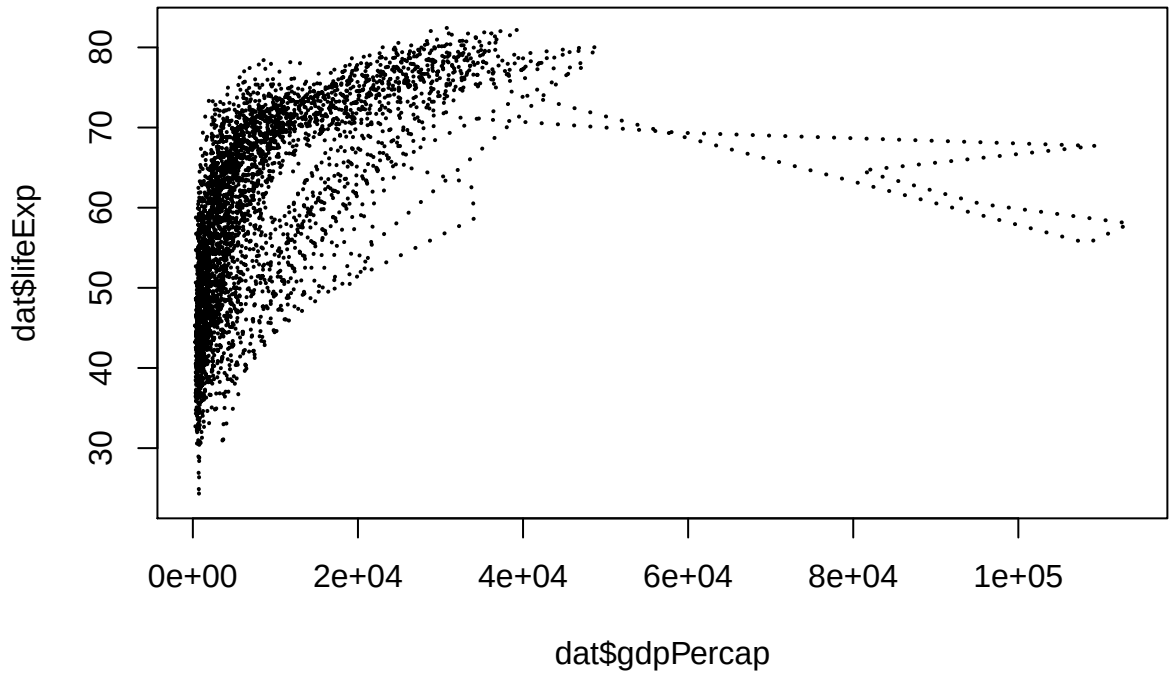

```
# Line plot with medium dashed line
plot(x = dat$gdpPercap, y = dat$lifeExp, type="l", lty=2)
```



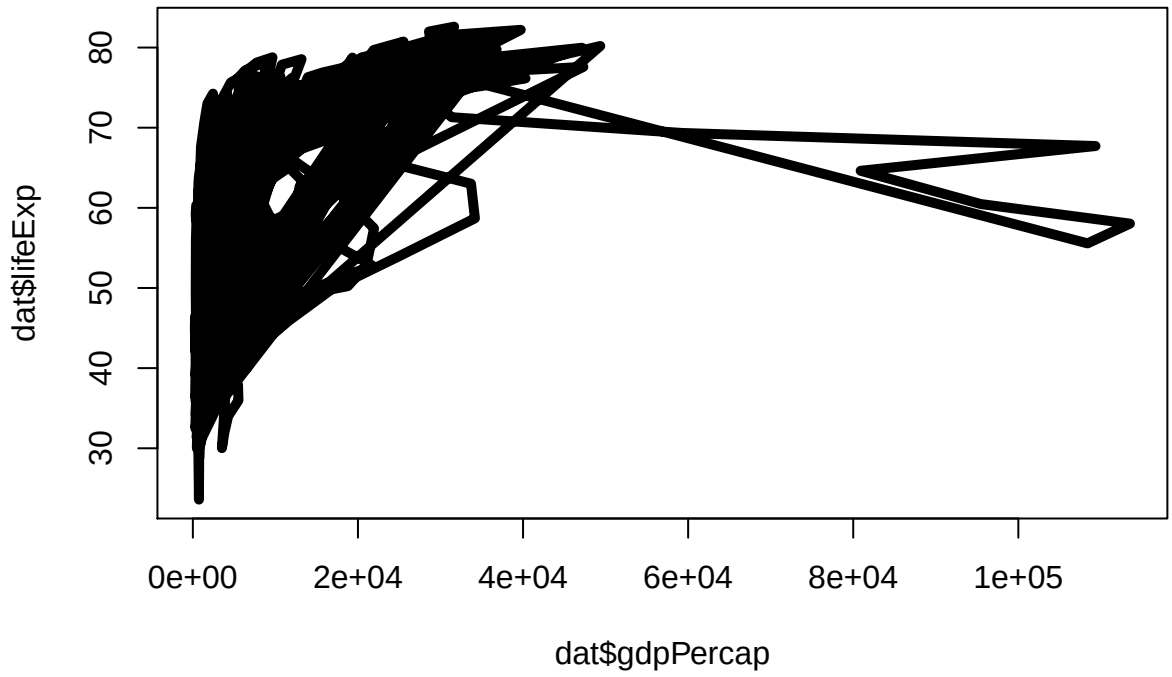

```
# Line plot with short dashed line
plot(x = dat$gdpPercap, y = dat$lifeExp, type="l", lty=3)
```



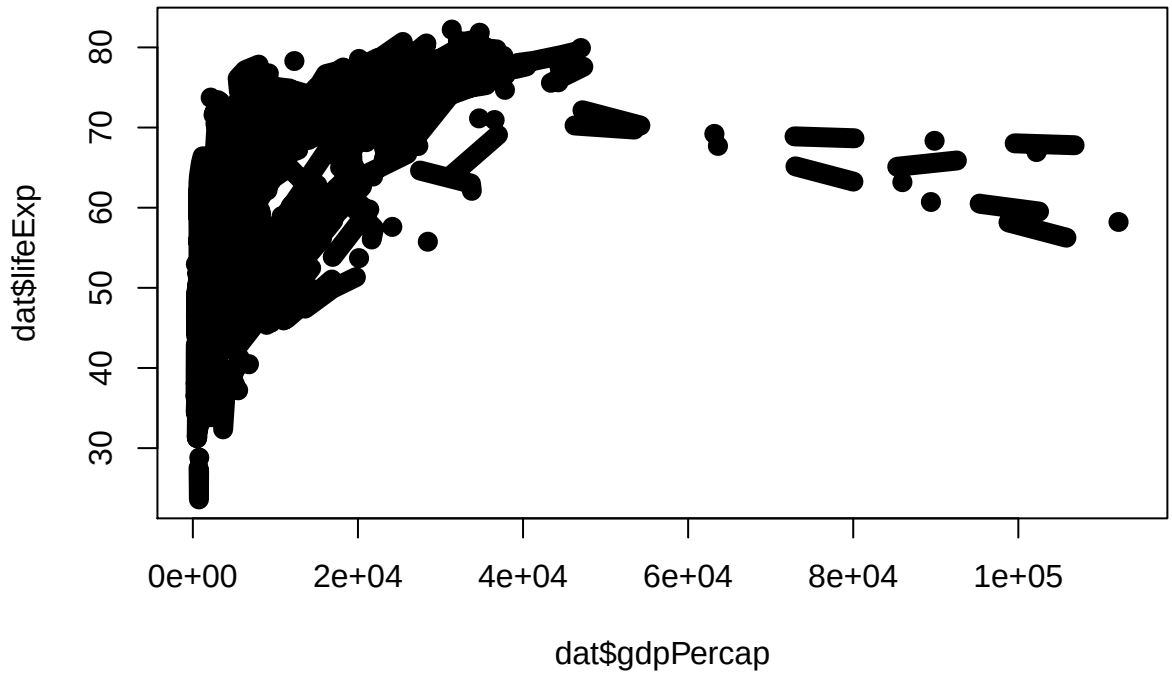

```
# Change line width to 2
plot(x = dat$gdpPercap, y = dat$lifeExp, type="l", lty=3, lwd=2)
```




```
# Change line width to 5
plot(x = dat$gdpPercap, y = dat$lifeExp, type="l",  lwd=5)
```




```
# Change line width to 10 and use dash-dot
plot(x = dat$gdpPercap, y = dat$lifeExp, type="l",  lty=4, lwd=10)
```

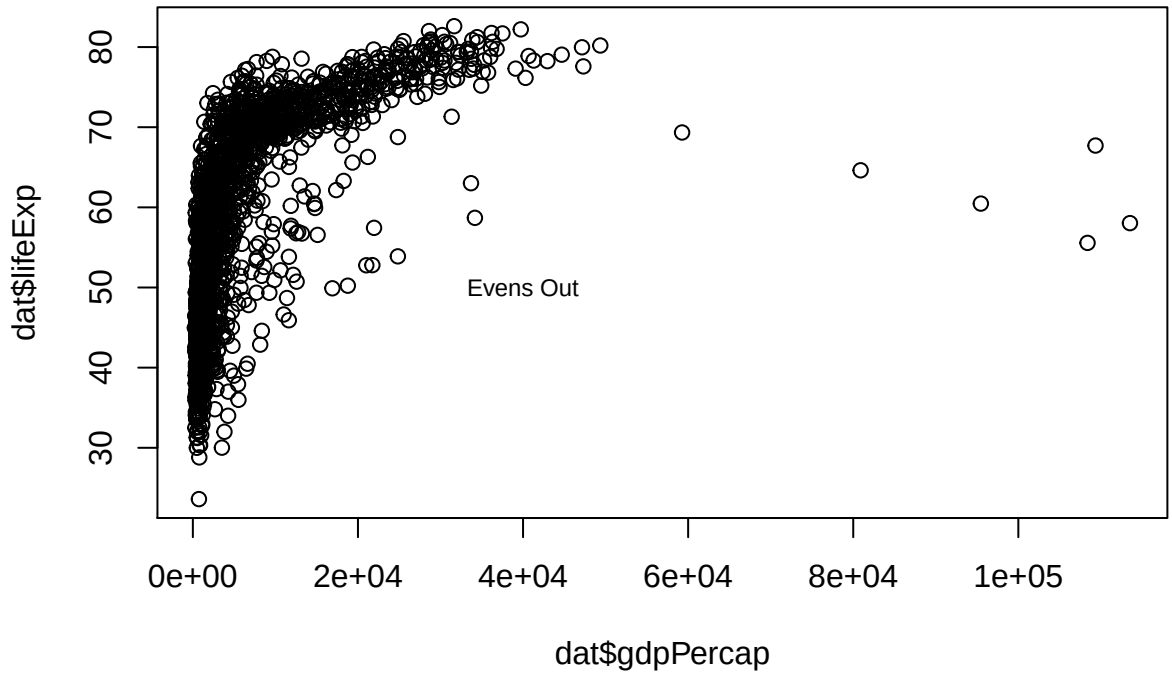



**1f. Annotations, reference lines, and legends]**

- Text

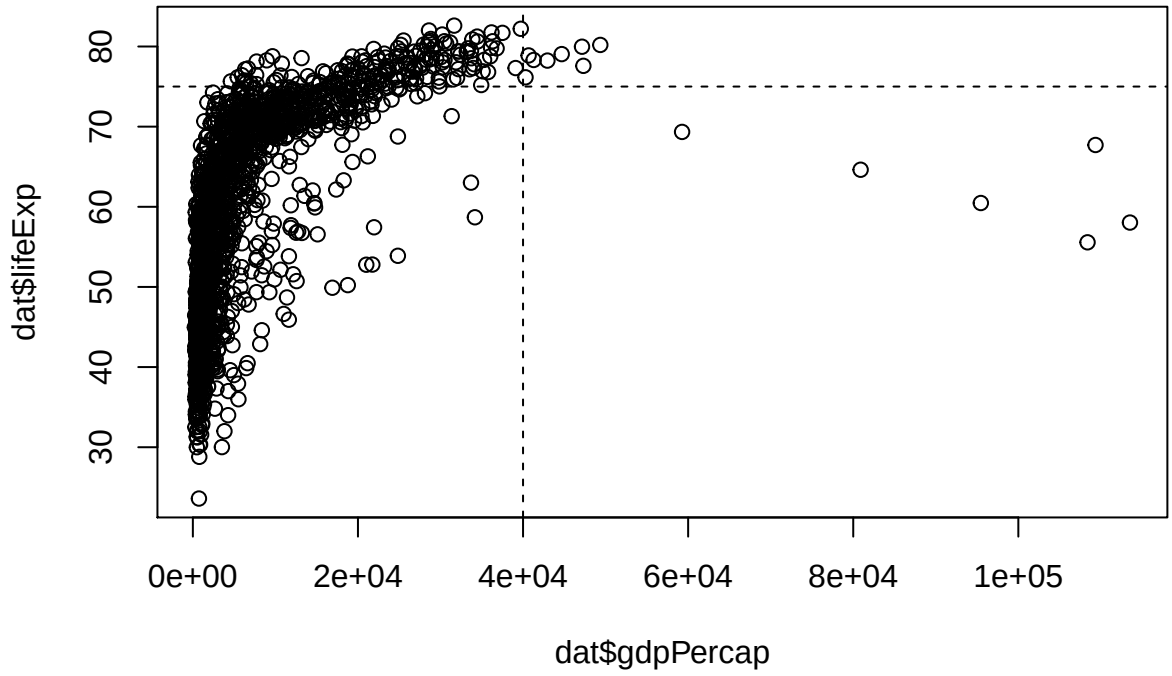
```
# plot the line first
plot(x = dat$gdpPercap, y = dat$lifeExp, type="p")
# now add the label
```

```
text(x=40000, y=50, labels="Evens Out", cex = .75)
```




- Reference Lines

```
# plot the line
plot(x = dat$gdpPercap, y = dat$lifeExp, type="p")
# now the guides
abline(v=40000, h=75, lty=2)
```




So, plot gets us the basics–but it's VERY basic. As we said above, it's usually not much use except as a very rough sketch of our data, like we might want to generate during exploratory data analysis. For more than the

basics, we need ggplot.

# 2. ggplot2

### 2a. Why ggplot?

- More elegant & compact code than with base graphics
- More aesthetically pleasing defaults than lattice
- Very powerful for exploratory data analysis
- Follows a grammar, just like any language.
- It defines basic components that make up a sentence. In this case, the grammar defines components in a plot.
- *G*rammar of *g*raphics (gg) originally coined by Lee Wilkinson

### 2b. Grammar

The general call for ggplot2 looks like this:

```
ggplot(data=, aes(x=, y=), color=, size=,) + geom_type1()+geom_type2()
```

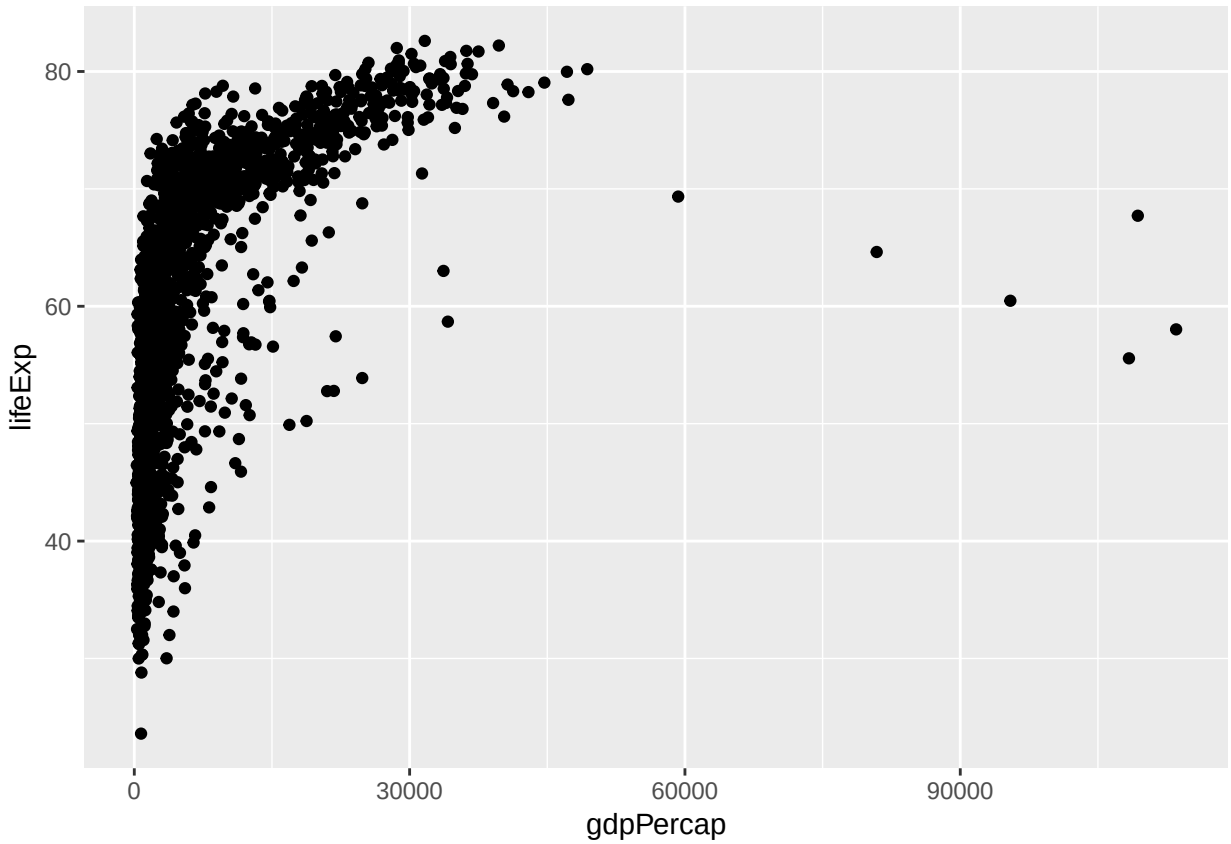
The *grammar* involves some basic components:

1. **Data**: a data frame that you pass in as an argument.
2. **Aes**thetics: How your data are represented visually, aka "mapping". Which variables are shown on x, y axes, as well as color, size, shape, etc.
3. **Geom**etry: The geometric objects in a plot. points, lines, polygons, etc.

The key to understanding ggplot2 is thinking about a figure in layers: just like you might do in an image editing program like Photoshop, Illustrator, or Inkscape. Each layer gets added on top of the previous one, so you can "stack" additional layers of information as needed.

Let's look at an example:

```
library(ggplot2)
ggplot(data = dat, aes(x = gdpPercap, y = lifeExp)) +
  geom_point()
```



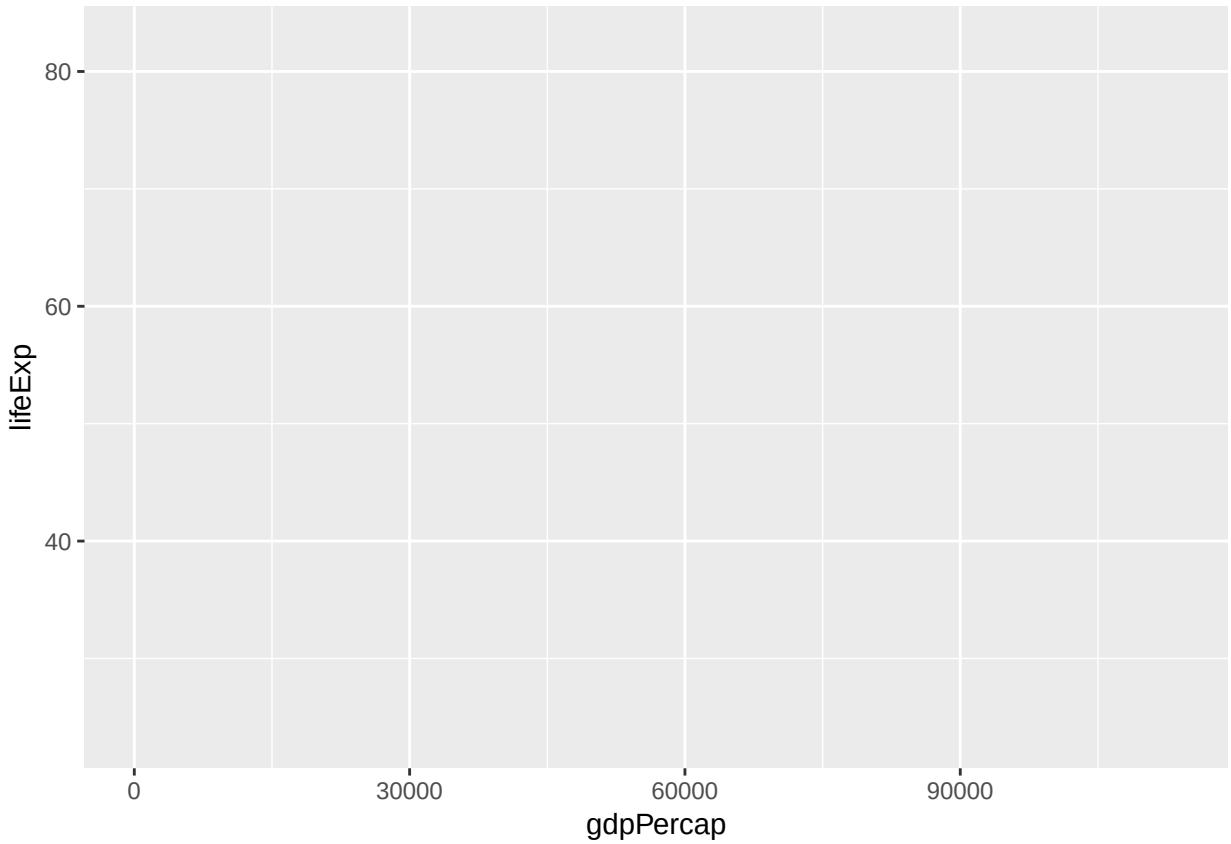

So the first thing we do is call the `ggplot` function. This function lets R know that we're creating a new plot, and any of the arguments we give the ggplot function are the global options for the plot: they apply to all layers on the plot.

Here, we've passed in two arguments to `ggplot`. First, we tell ggplot what `data` we want to show on our figure, in this example the `gapminder` data we read in earlier.

For the second argument we passed in the `aes` function, which tells `ggplot` how variables in the data map to aesthetic properties of the figure, in this case the x and y locations. Here we told ggplot we want to plot the `lifeExp` column of the gapminder data frame on the x-axis, and the `gdpPercap` column on the y-axis. Notice that we didn't need to explicitly pass `aes` these columns (e.g. `x = gapminder[, "lifeExp""])`, this is because ggplot is smart enough to know to look in the data for that column!

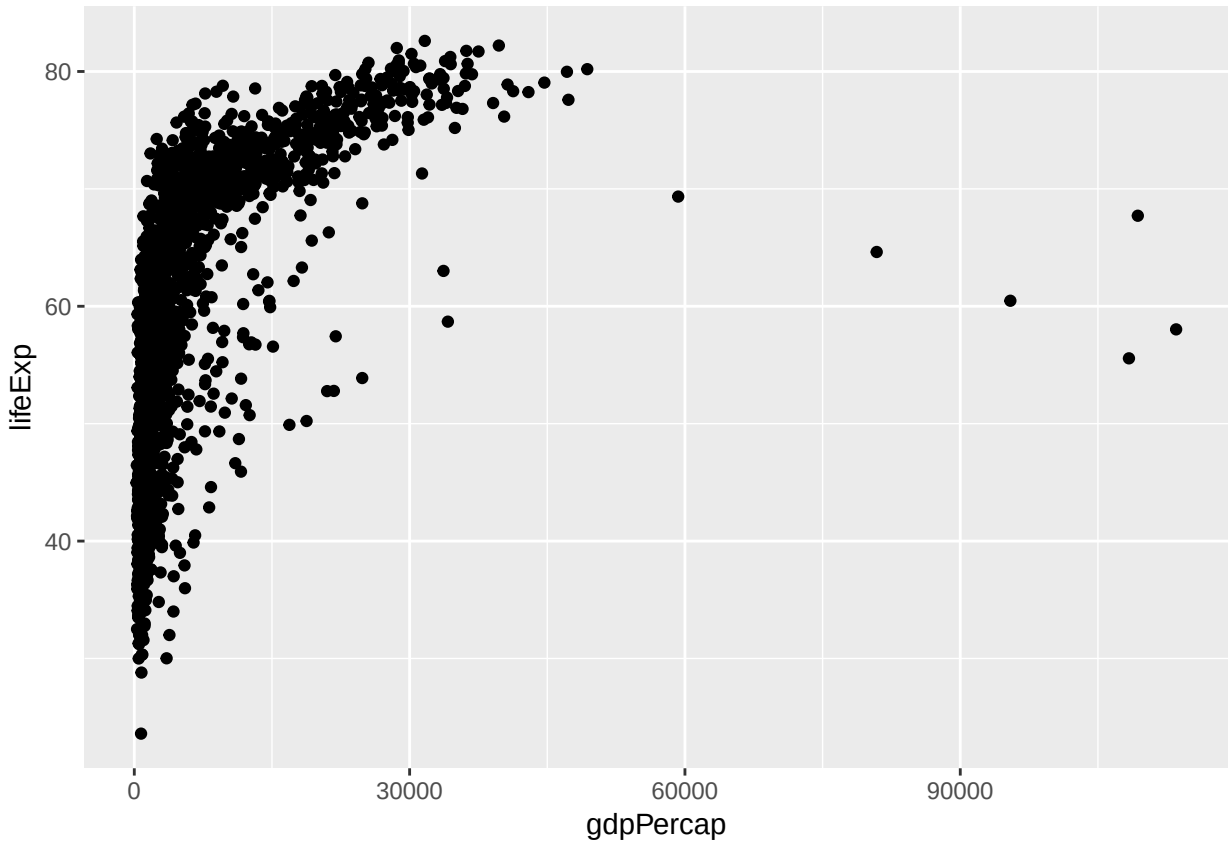
By itself, the call to ggplot isn't enough to draw a figure:

```
ggplot(data = dat, aes(x = gdpPercap, y = lifeExp))
```

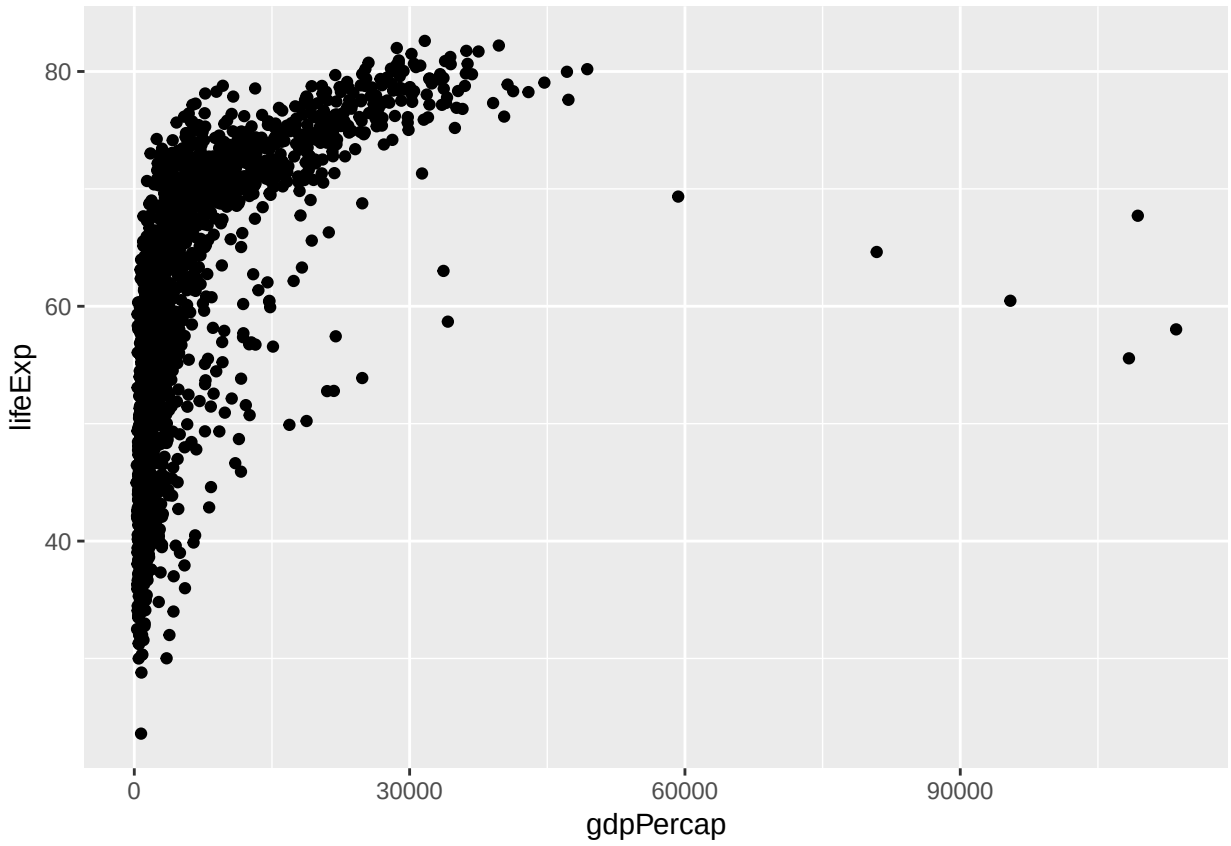



We need to tell `ggplot` **how** we want to visually represent the data, which we do by adding a new `geom` layer. In our example, we used `geom_point`, which tells ggplot we want to visually represent the relationship between x and y as a scatterplot of points:

```
ggplot(data = dat, aes(x = gdpPercap, y = lifeExp)) + geom_point()
```




```
# same as
my_plot <- ggplot(data = dat, aes(x = gdpPercap, y = lifeExp))
my_plot + geom_point()
```




**Challenge 1**

Modify the example so that the figure visualise how life expectancy has changed over time:

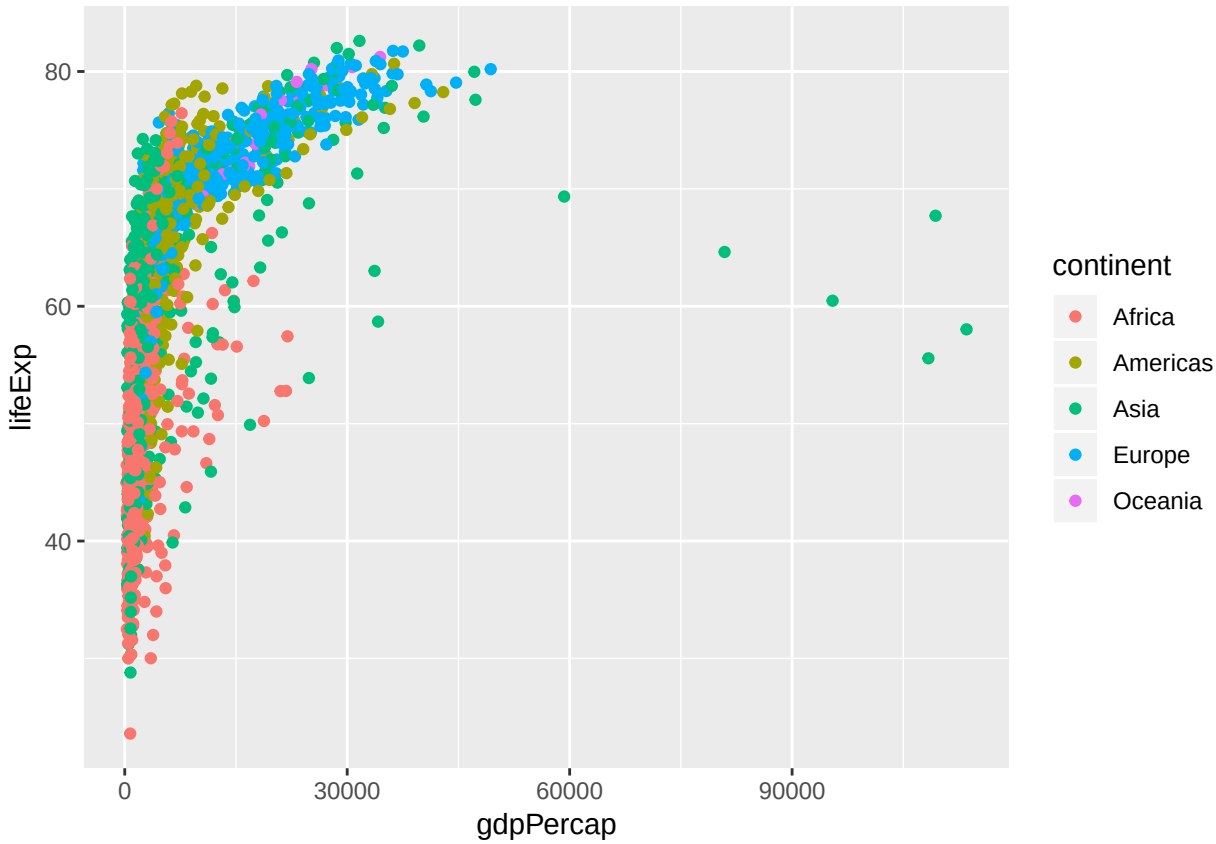
Hint: the gapminder dataset has a column called "year"", which should appear on the x-axis.

```
# YOUR CODE HERE
```

### 2c. Anatomy of `aes`

In the previous examples and challenge we've used the `aes` function to tell the scatterplot geom about the **x** and **y** locations of each point. Another aesthetic property we can modify is the point **color**.

```
ggplot(data = dat, aes(x = gdpPercap, y = lifeExp, color=continent)) + geom_point()
```




In base plotting, we have to specify particular properties, like `color="red"` or `size=10`, which is a bit limiting if we have to do every modification by hand! Inside ggplot's `aes()` function, however, these arguments are passed entire variables, whose values will then be displayed using different realizations of that aesthetic.

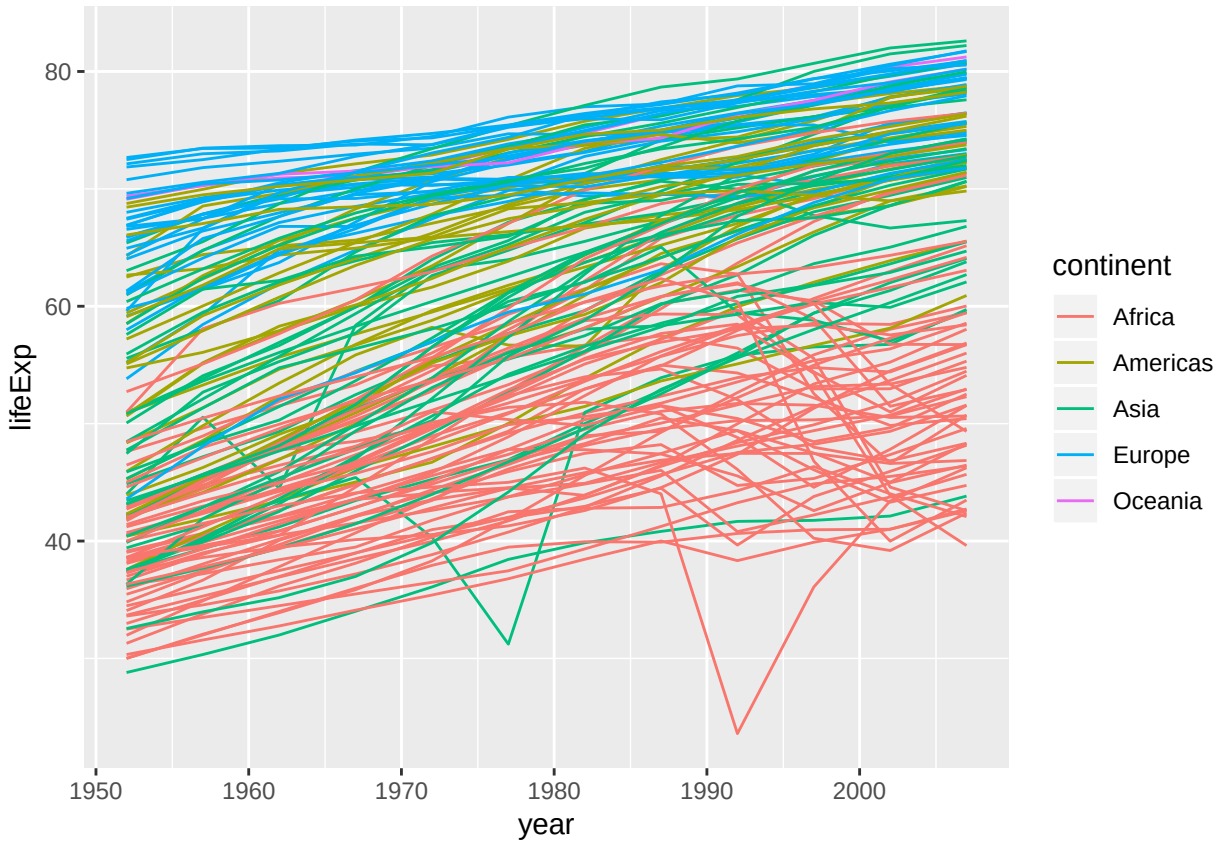
**Color** isn't the only aesthetic argument we can set to display variation in the data. We can also vary by shape, size, etc. Try playing around with the options in the cell below.

```
ggplot(data=, aes(x=, y=, by =, color=, linetype=, shape=, size=))
```

### 2d. Layers

In the previous challenge, you plotted lifExp over time. Using a scatterplot probably isn't the best for visualising change over time. Instead, let's tell ggplot to visualise the data as a line plot:

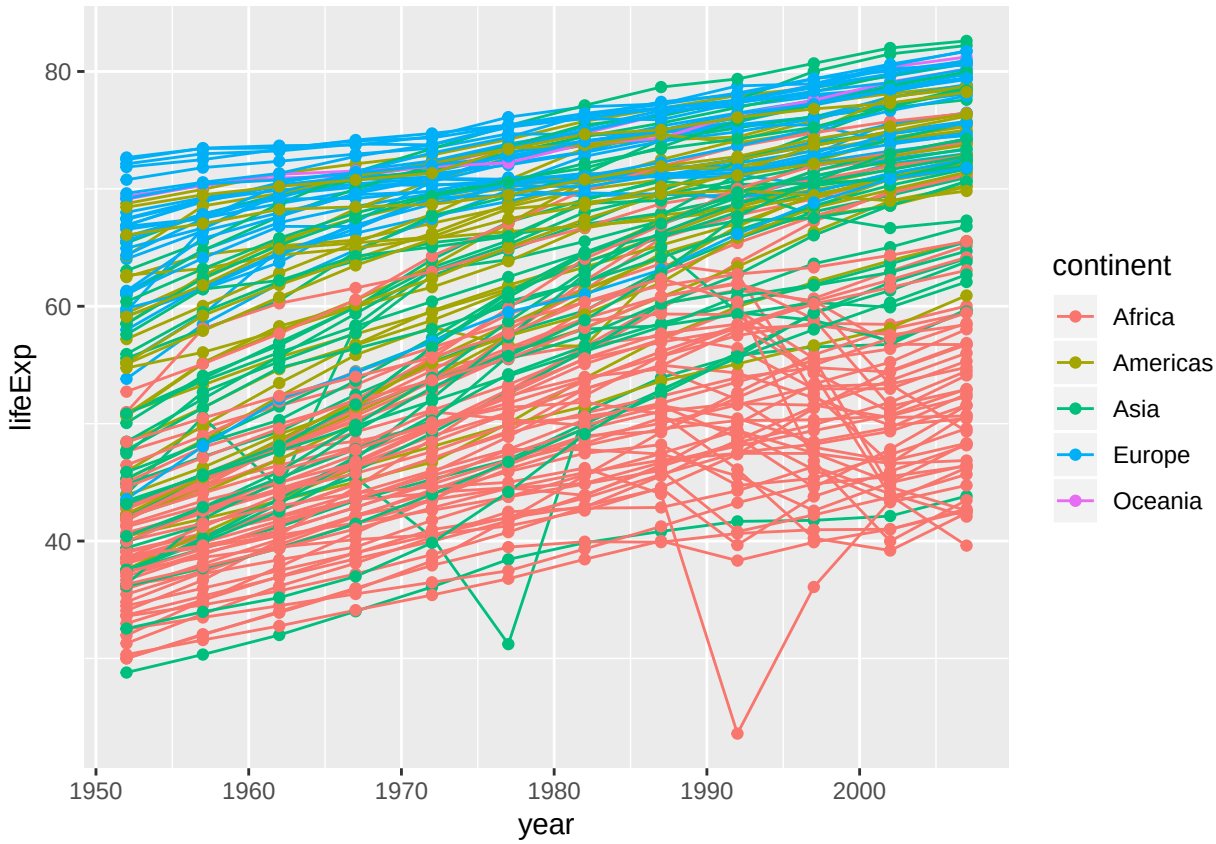
```
ggplot(data = dat, aes(x=year, y=lifeExp, by=country, color=continent)) + geom_line()
```




Instead of adding a `geom_point` layer, we've added a `geom_line` layer. We've added the **by** aesthetic, which tells ggplot to draw a line for each country.

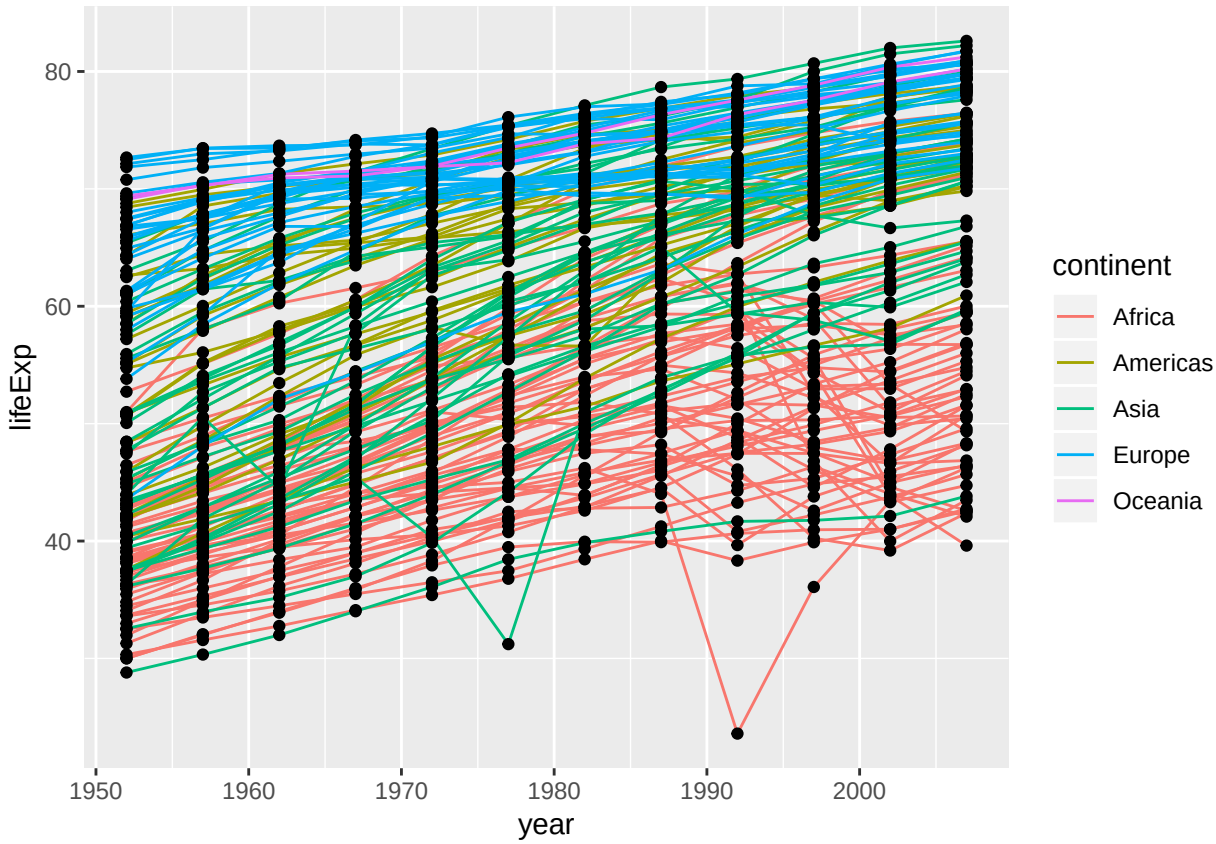
But what if we want to visualise both lines and points on the plot? We can simply add another layer to the plot:

```
ggplot(data = dat, aes(x=year, y=lifeExp, by=country, color=continent)) + geom_line() + geom_point()
```




It's important to note that each layer is drawn on top of the previous layer. In this example, the points have been drawn on top of the lines. Here's a demonstration:

```
ggplot(data = dat, aes(x=year, y=lifeExp, by=country)) + geom_line(aes(color=continent)) + geom_point()
```




In this example, the aesthetic mapping of **color** has been moved from the global plot options in ggplot to the `geom_line` layer so it no longer applies to the points. Now we can clearly see that the points are drawn on top of the lines.

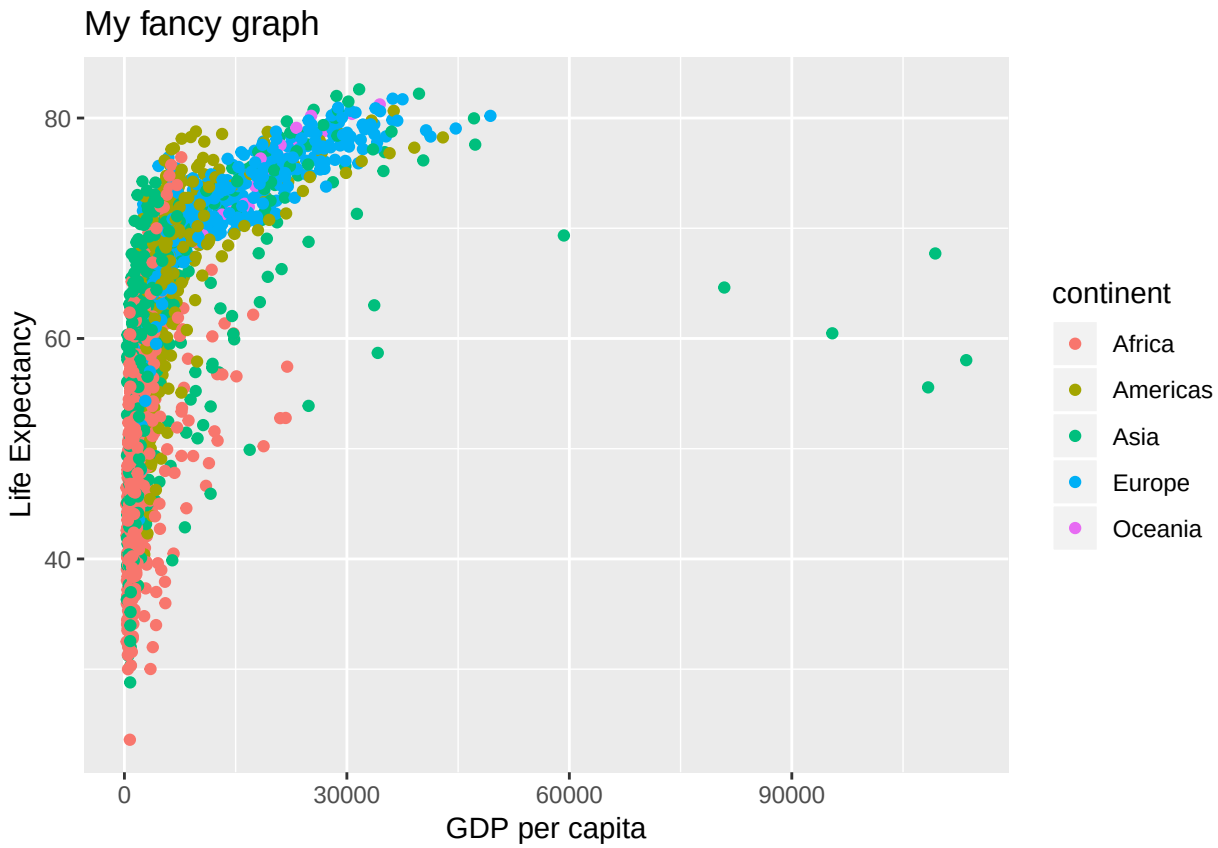
### Challenge 2

Switch the order of the point and line layers from the previous example. What happened?

### 2e. Labels

Labels are considered to be their own layers in ggplot.

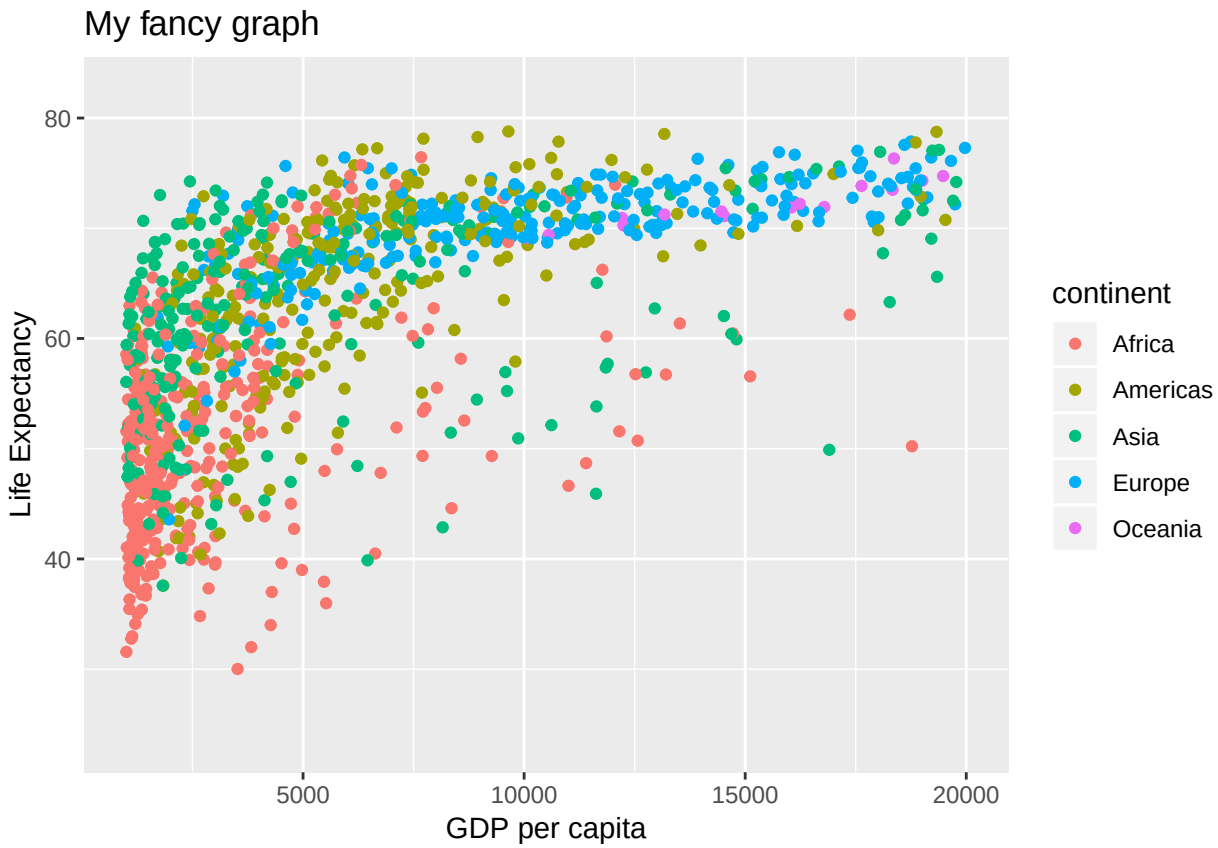
```
# add x and y axis labels
ggplot(data = dat, aes(x = gdpPercap, y = lifeExp, color=continent)) + geom_point() + xlab("GDP per cap
```




So are scales:

```
# limit x axis from 1,000 to 20,000
ggplot(data = dat, aes(x = gdpPercap, y = lifeExp, color=continent)) + geom_point() + xlab("GDP per cap
```

```
## Warning: Removed 515 rows containing missing values (geom_point).
```

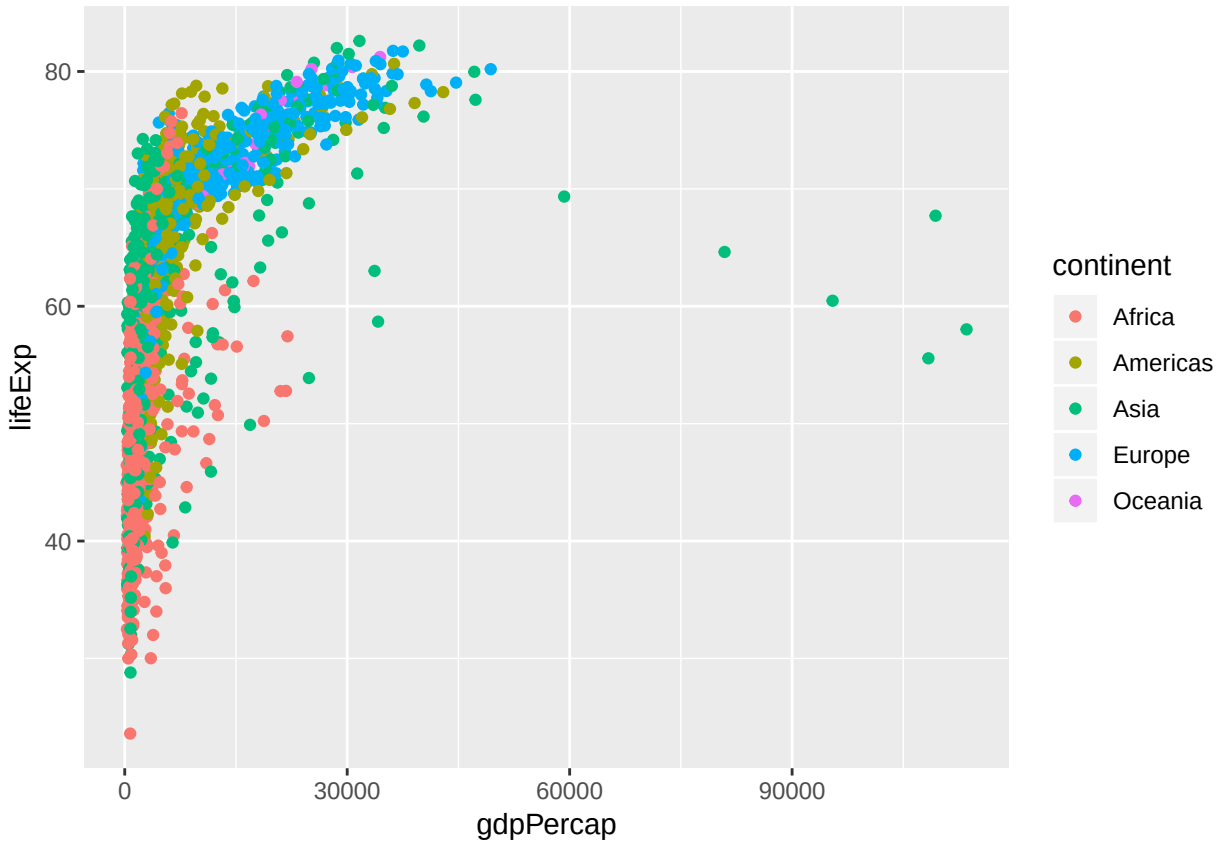



Note that we get a warning message that some of the data has been dropped due to the new limits we imposed.

### 2f. Transformations and Stats

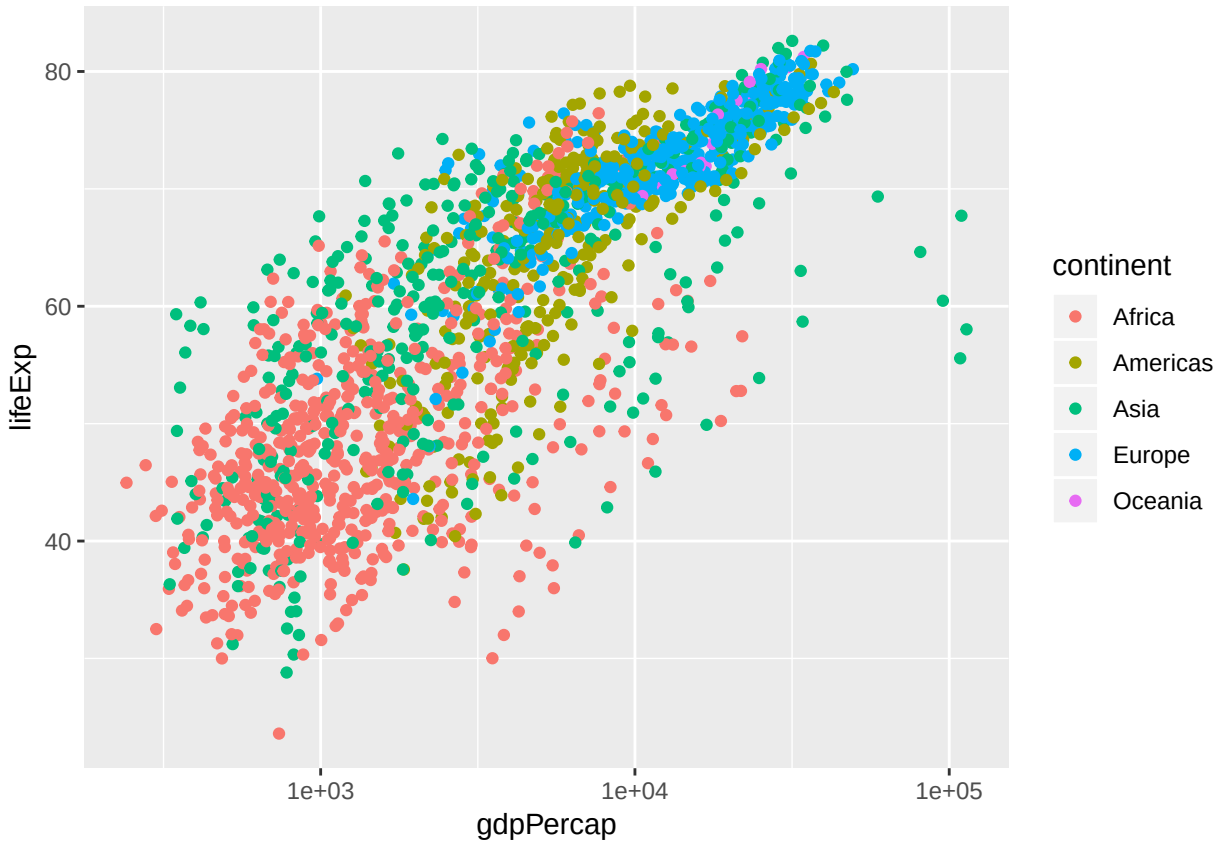
ggplot also makes it easy to overlay statistical models over the data. To demonstrate we'll go back to an earlier example:

```
ggplot(data = dat, aes(x = gdpPercap, y = lifeExp, color=continent)) + geom_point()
```




We can change the scale of units on the x axis using the `scale` functions. These control the mapping between the data values and visual values of an aesthetic. This is nice because we don't have to apply the transformations we might want for graphing on our real data:

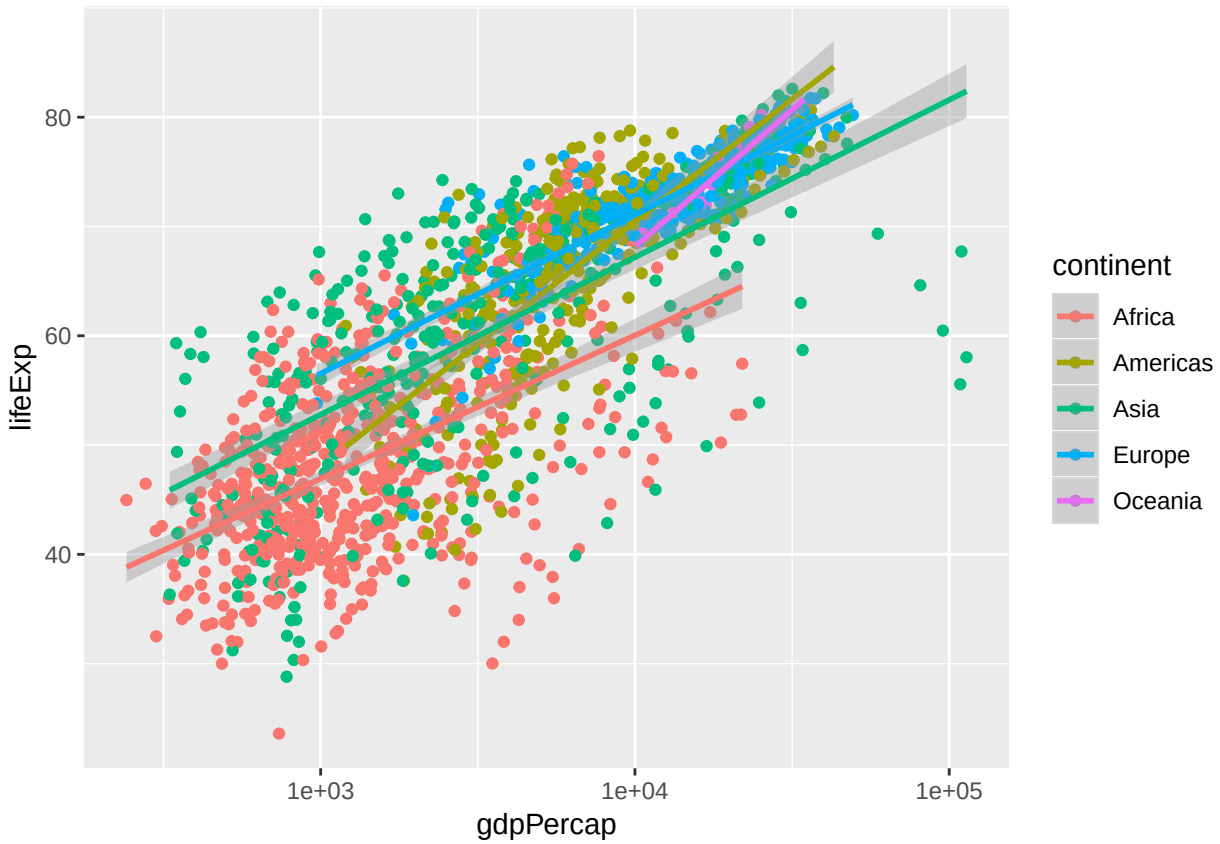
```
ggplot(data = dat, aes(x = gdpPercap, y = lifeExp, color=continent)) + geom_point() + scale_x_log10()
```




The `log10` function applied a transformation to the values of the `gdpPercap` column before rendering them on the plot, so that each multiple of 10 now only corresponds to an increase in 1 on the transformed scale, e.g. a GDP per capita of 1,000 is now 3 on the y axis, a value of 10,000 corresponds to 4 on the x axis and so on. This makes it easier to visualise the spread of data on the x-axis.

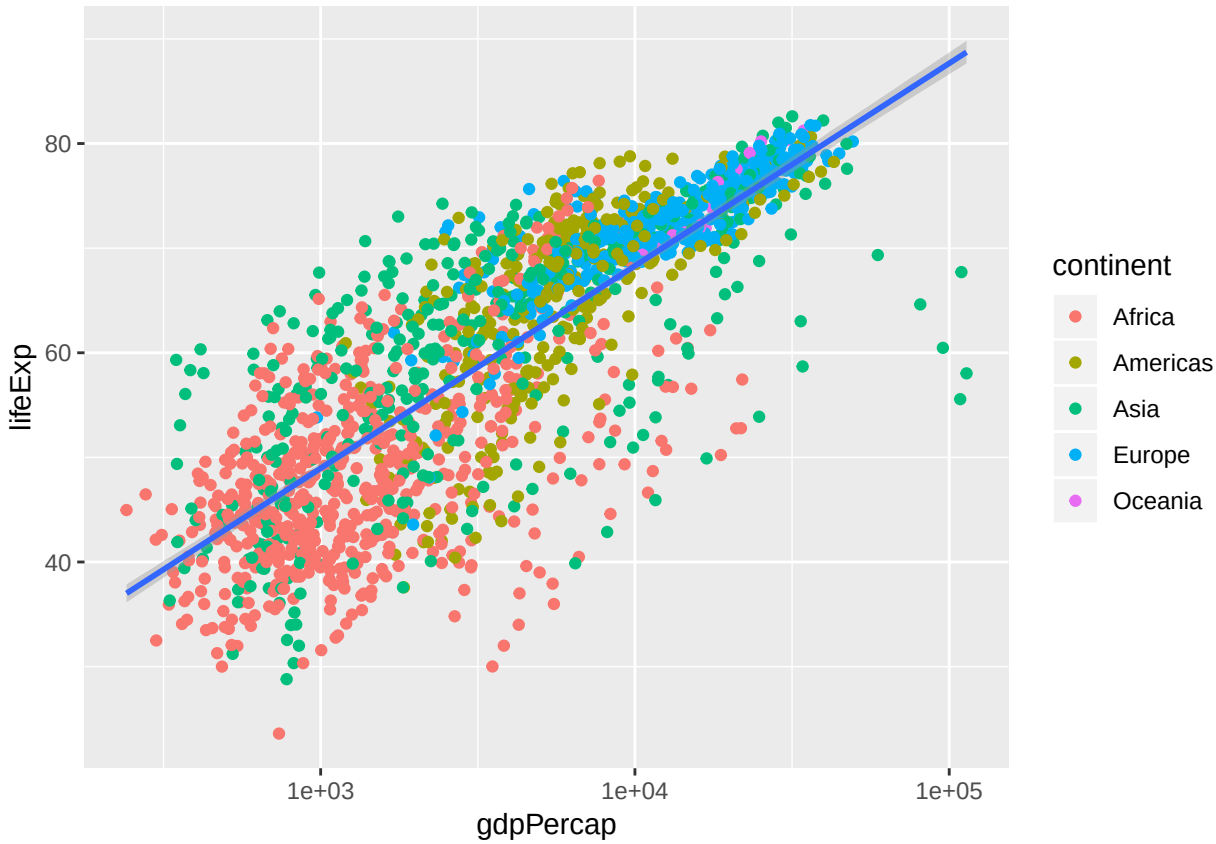
We can fit a simple relationship to the data by adding another layer, `stat_smooth` (in many cases, but not all, `geom_smooth` and `stat_smooth` are interchangeable):

```
ggplot(data = dat, aes(x = gdpPercap, y = lifeExp, color=continent)) + geom_point() + scale_x_log10() +
```




Note that we currently have 5 lines, one for each region, because the **color** option is the global `aes` function..
But if we move it, we get different restuls:

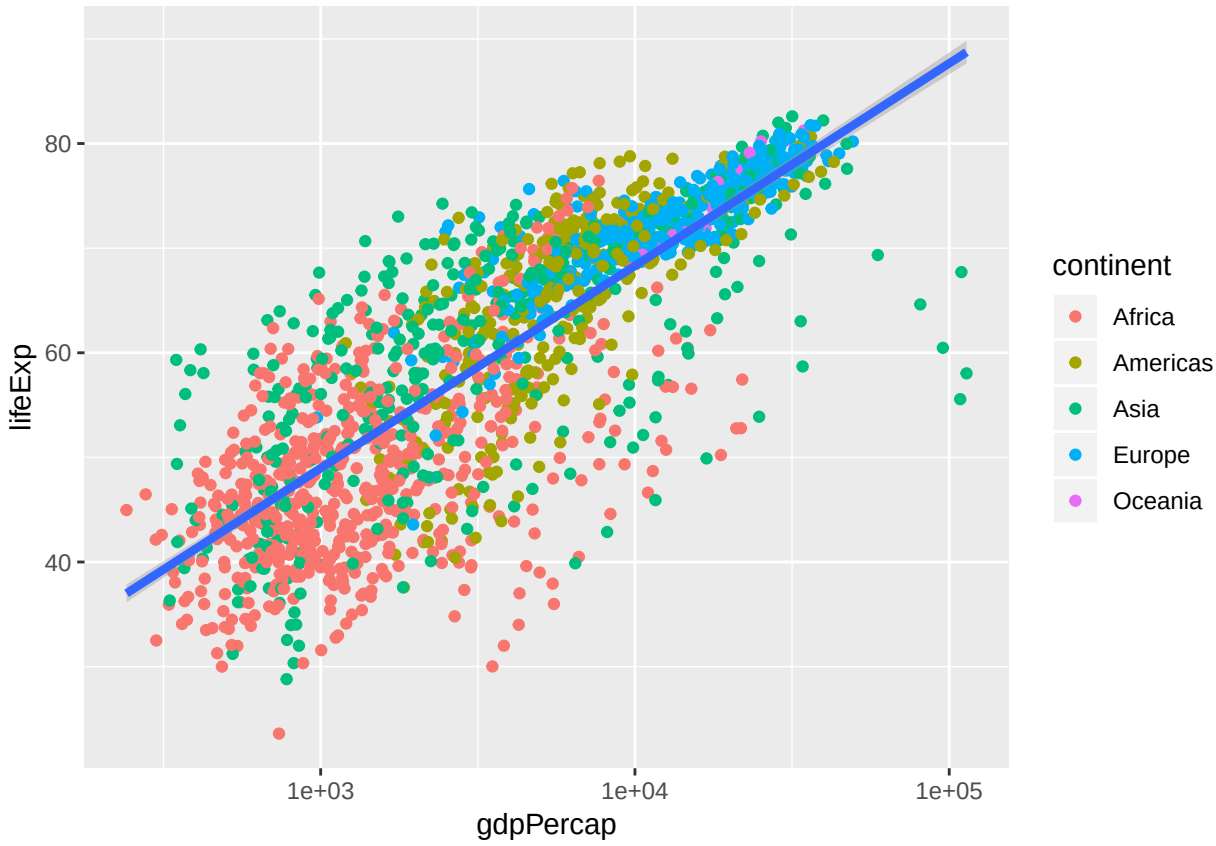
```
ggplot(data = dat, aes(x = gdpPercap, y = lifeExp)) + geom_point(aes(color=continent)) + scale_x_log10()
```




Now the `stat_smooth` operation is only acting on the x and y specified within `aes`. This tells us that what we specify as our aesthetic also affects future layers. Here, the **color** aesthetic is only applied to the mapping of `geom_point`, not the line generated by `stat_smooth`.

As you might expect, we can set other properties within each additional layer as well. Here, we can make the line thicker by setting the **size** aesthetic in the geom_smooth layer:

```
ggplot(data = dat, aes(x = gdpPercap, y = lifeExp)) + geom_point(aes(color=continent)) + scale_x_log10(
```




**Challenge 3**

Modify the color and size of the points on the point layer in the previous example so that they are fixed (i.e. not reflective of continent).

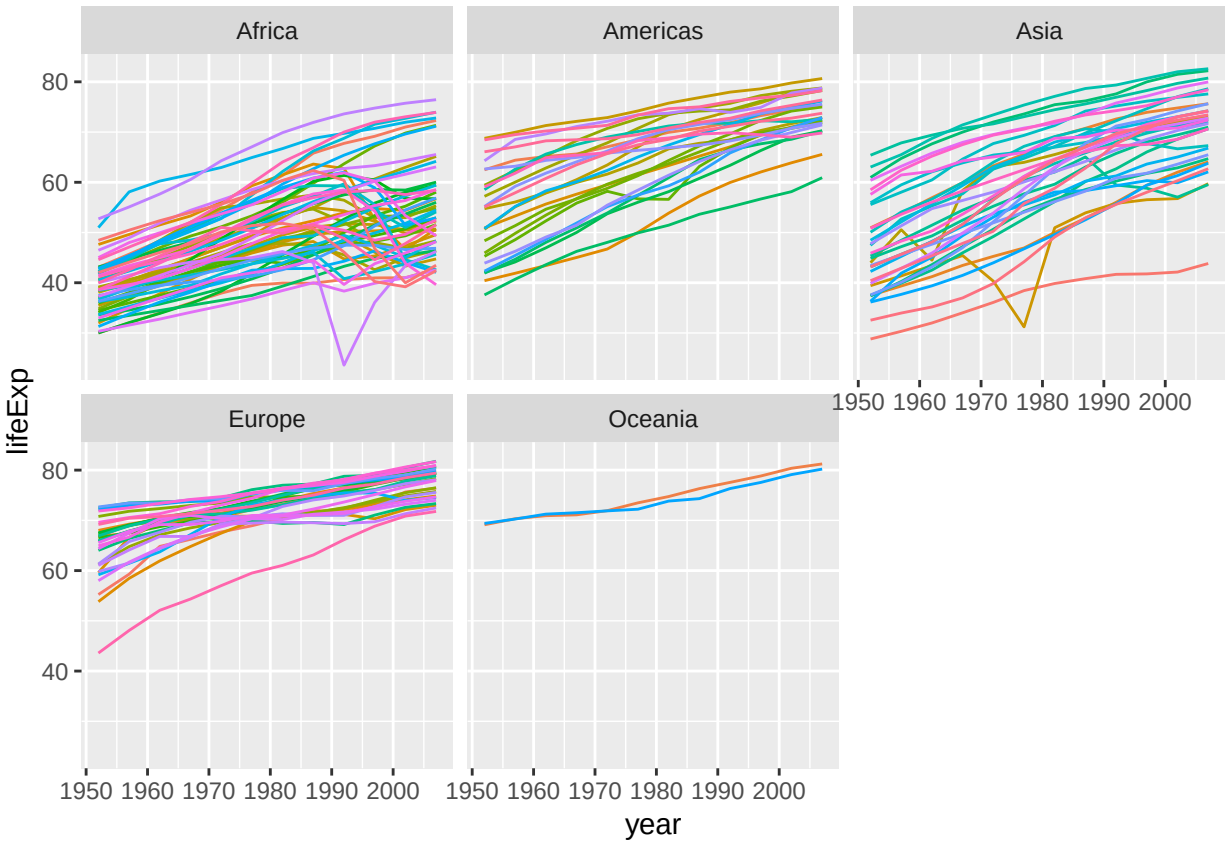
Hint: do not use the first `aes` function.

```
# YOUR CODE HERE
```

### 2g. Facets

Earlier we visualised the change in life expectancy over time across all countries in one plot. Alternatively, we can split this out over multiple panels by adding a layer of **facet** panels:

```
ggplot(data = dat, aes(x = year, y = lifeExp, color=country)) +
  geom_line() + facet_wrap( ~ continent) + theme(legend.position="none")
```




```
# I've removed the legend here so that the charts themselves will be visible.
```

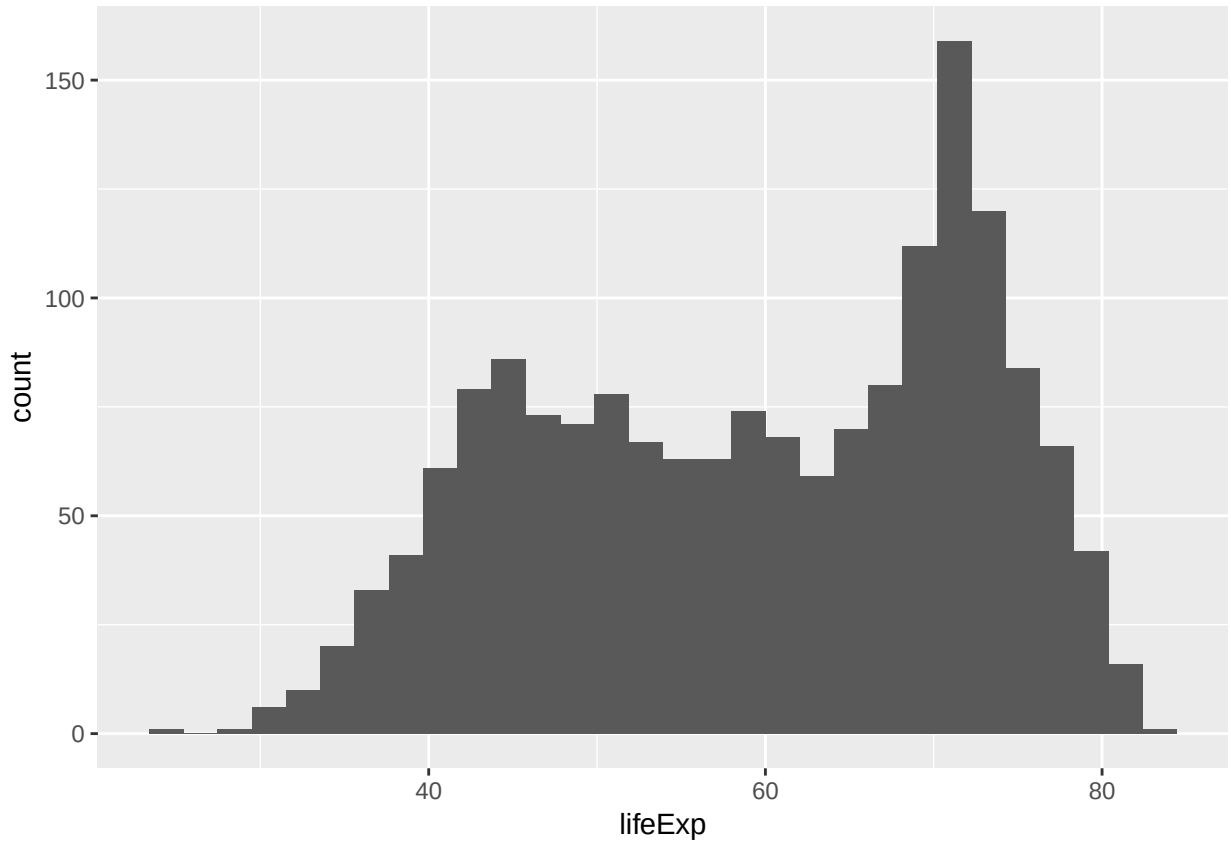
### 2h. Putting it all together

This is just a taste of what you can do with ggplot2. RStudio provides a really useful cheat sheet of the different layers available, as do the authors of the R Graphic cookbook, and more extensive documentation is available on the ggplot2 website. Finally, if you have no idea how to change something, a quick google search will usually send you to a relevant question and answer on Stack Overflow with reusable code to modify!

#### bar plots

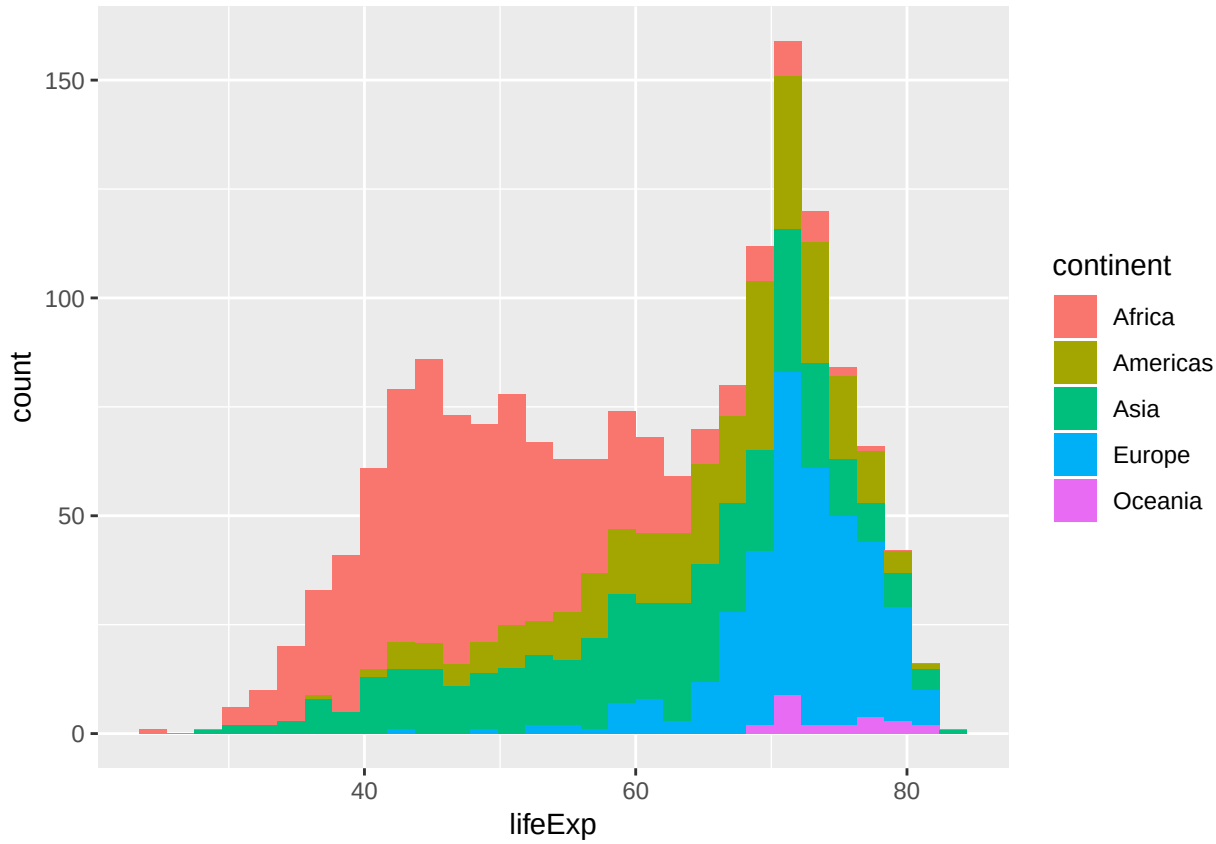
```
# count of lifeExp bins
ggplot(data = dat, aes(x = lifeExp)) + geom_bar(stat="bin")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



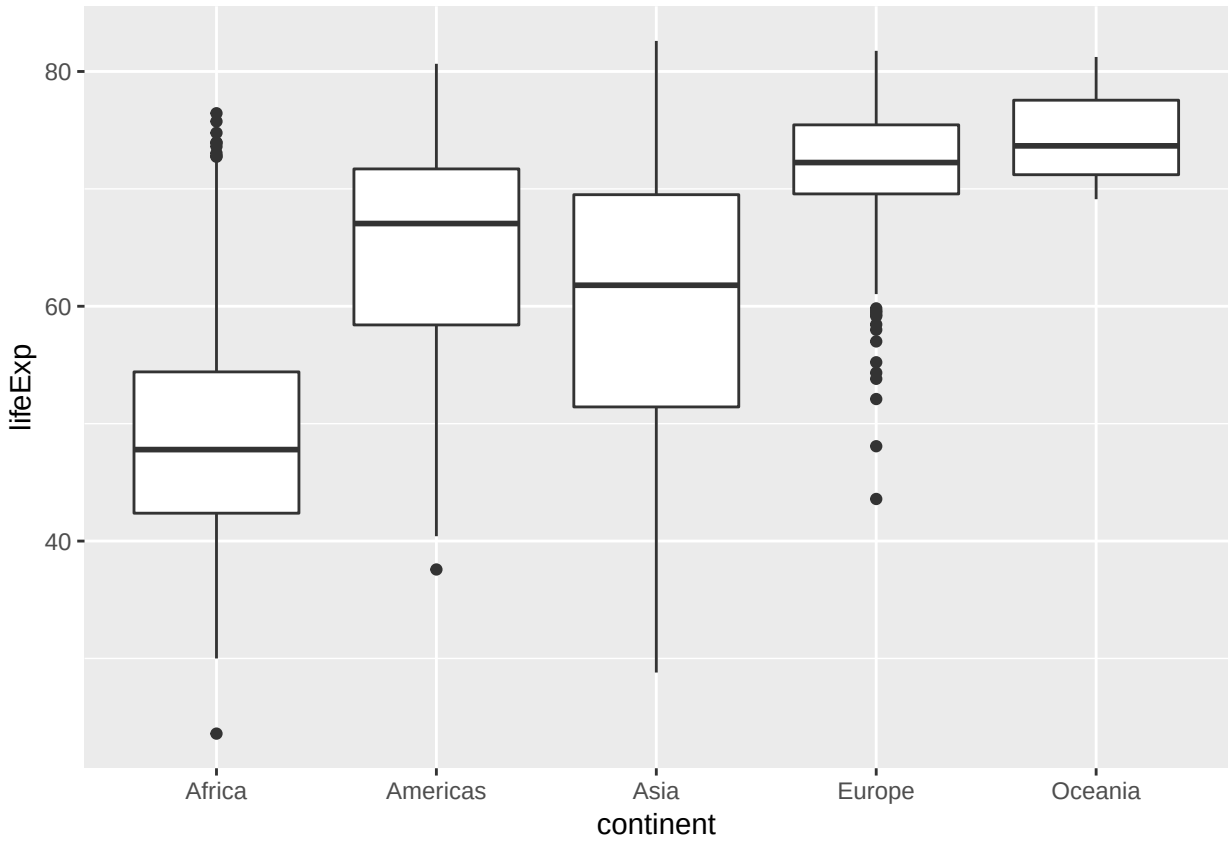

```
# with color representing regions
ggplot(data = dat, aes(x = lifeExp, fill = continent)) + geom_bar(stat="bin")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




**box plots**

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




**Challenge 4**

Create a density plot of GDP per capita, filled by continent.

Advanced: - Transform the x axis to better visualise the data spread. - Add a facet layer to panel the density plots by year.

```
# YOUR CODE HERE.
```

###. Exporting

Two basic image types

1) **Raster/Bitmap** (.png, .jpeg)

Every pixel of a plot contains its own separate coding; not so great if you want to resize the image. In other words, if you expect to do resizing, or the chart might be displayed at a very different size than what you see on your screen (e.g., a conference presentation), you probably don't want these formats.

```
jpeg(filename="example.png", width=, height=)
plot(x,y)
dev.off()
```

2) **Vector** (.pdf, .ps)

Every element of a plot is encoded with a function that gives its coding conditional on several factors; great for resizing (presentations, etc.).

```
pdf(filename="example.pdf", width=, height=)
plot(x,y)
dev.off()
```

**Exporting with ggplot**

```
# Assume we saved our plot is an object called example.plot
ggsave(filename="example.pdf", plot=example.plot, scale=, width=, height=)
```