# Assignment_Mar16

Garrett Bullivant

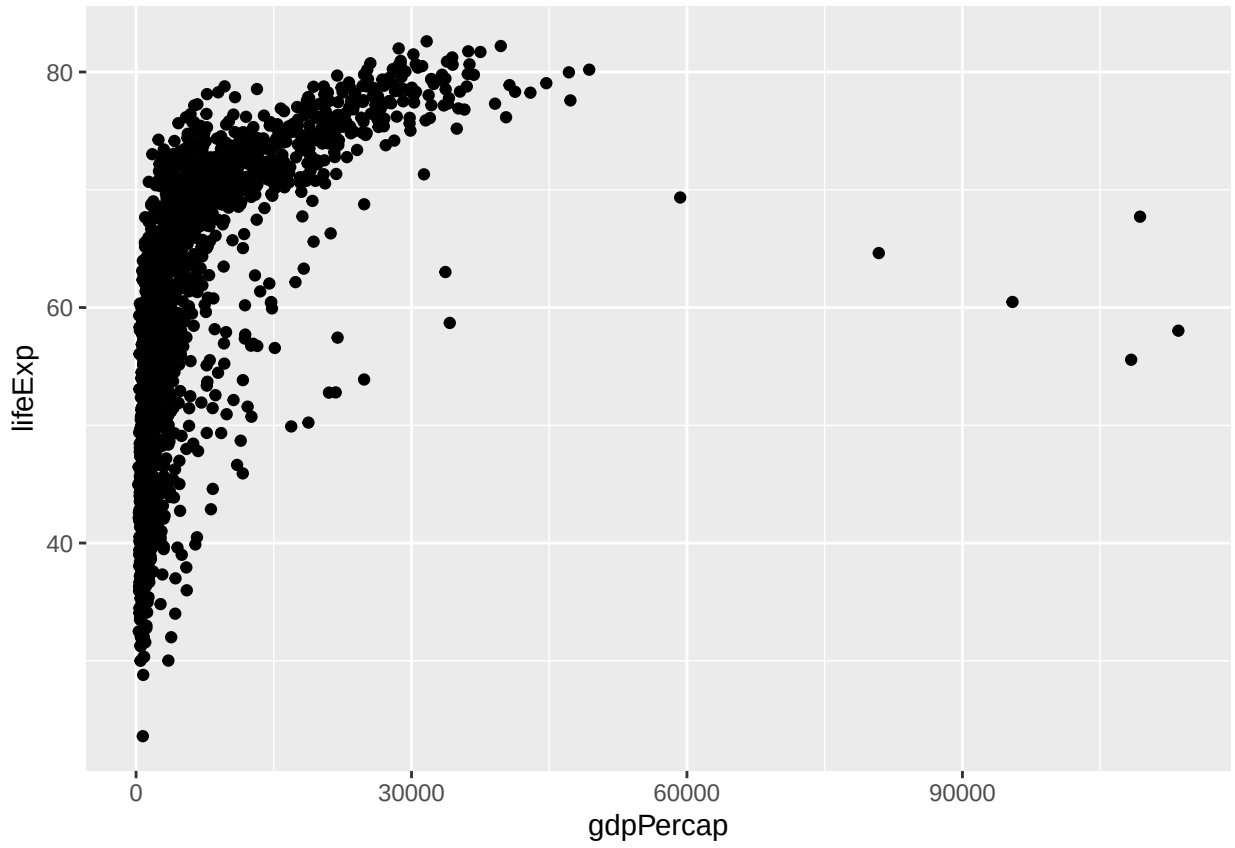
2023-03-16

```
# load relevant libraries
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.1     v purrr   0.3.5
## v tibble  3.1.8     v dplyr   1.1.0
## v tidyr   1.2.1     v stringr 1.5.0
## v readr   2.1.3     v forcats 0.5.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

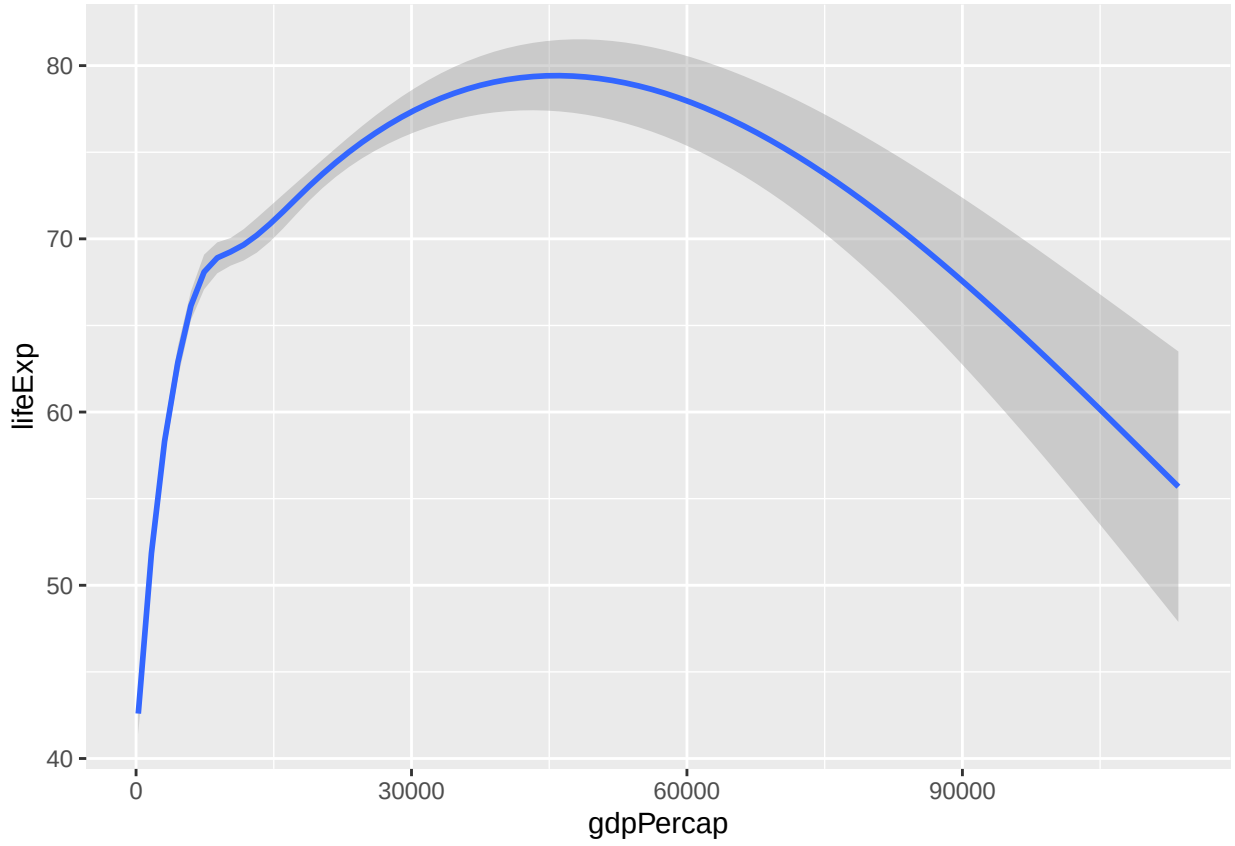
```
library(ggplot2)
library(socviz)
library(gapminder)
```

```
# generate our first ggplot using gapminder dataset and plotting for gdpPercap against lifeExp
# we must first generate our ggplot as an object and then plot it with our geom of choice

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

p + geom_point()
```



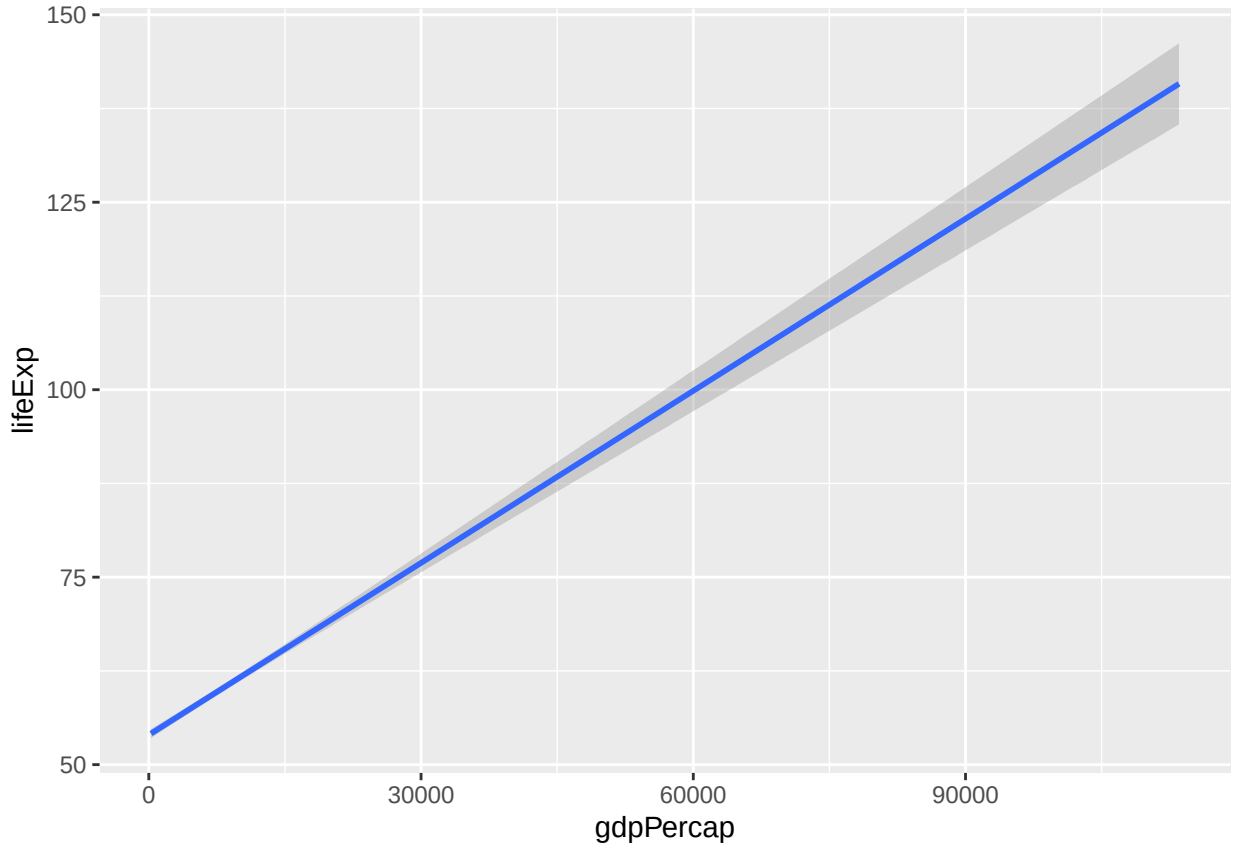

```
# we can also plot our ggplot with a trend line geom
p + geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



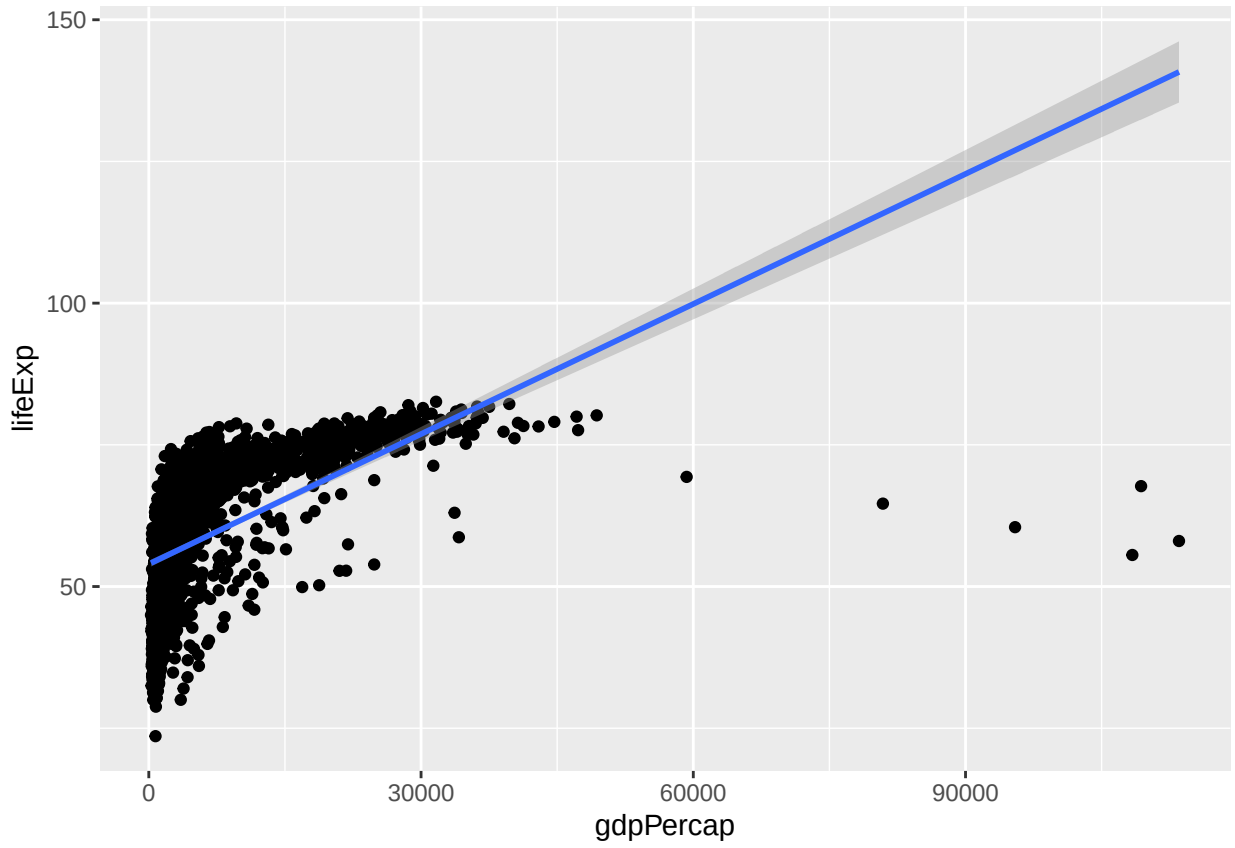

```
# we can make our trend line linear instead of additive
p + geom_smooth(method = "lm")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



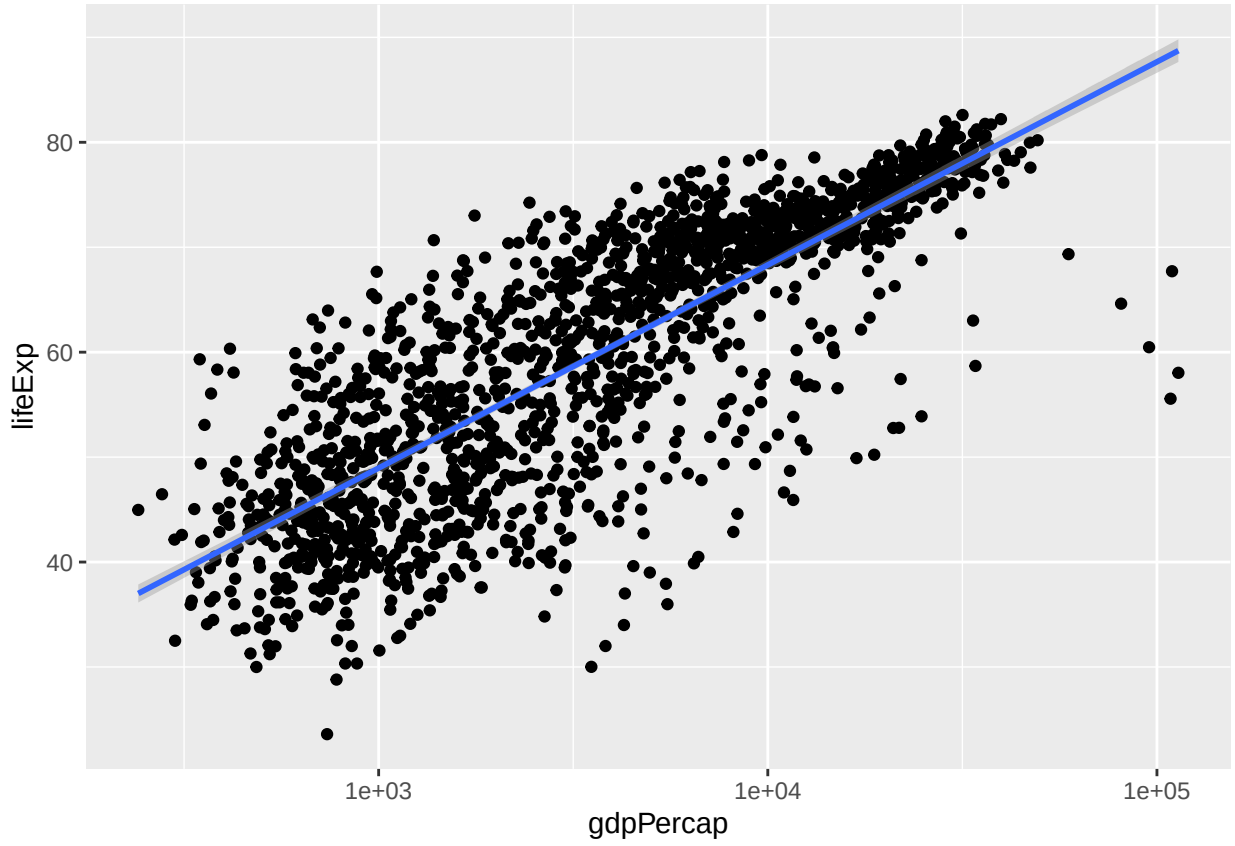

```
# we can have more than one geom. We can include our data points and a trend line
p + geom_point() + geom_smooth(method = "lm")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



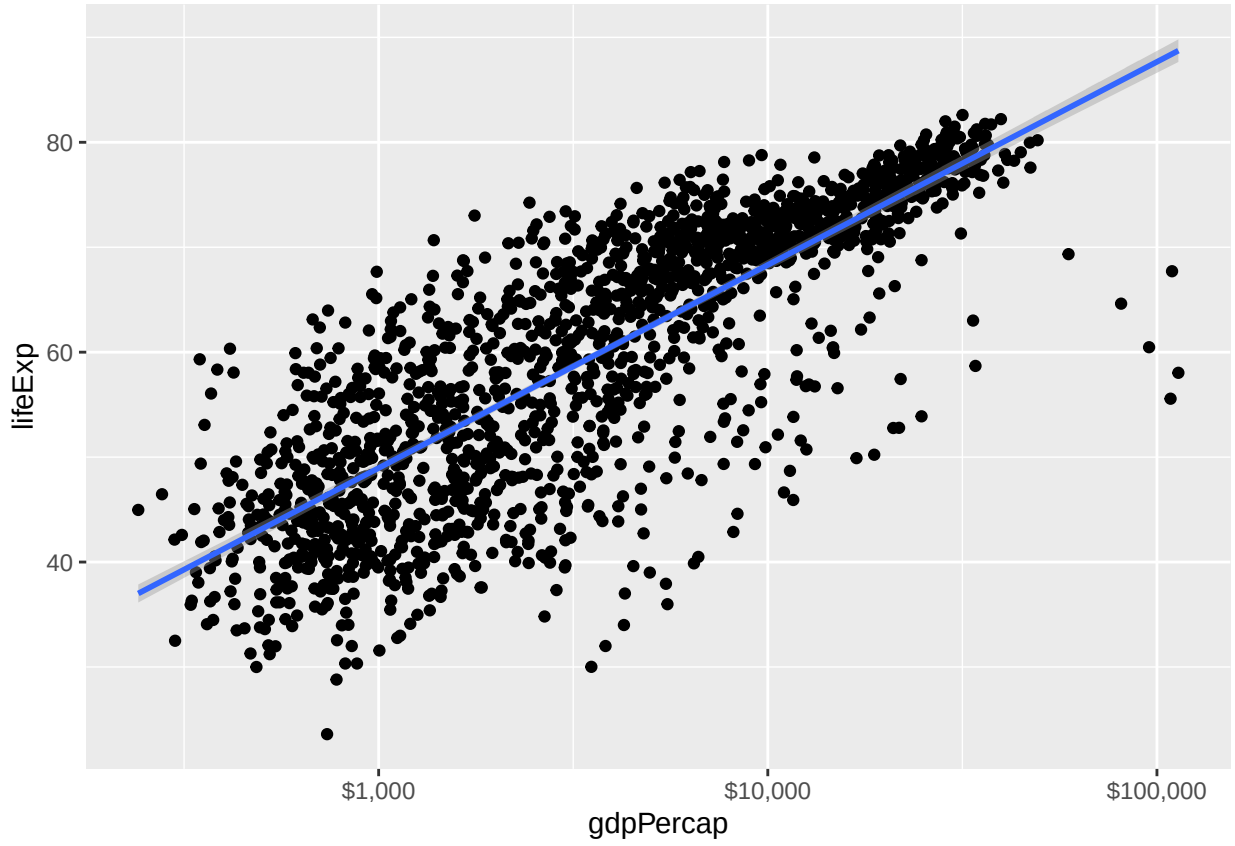

```
# since our data is skewed to the left we can log normalize it for better visualization
p + geom_point() + geom_smooth(method = "lm") + scale_x_log10()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
# since we are working with gdp, we can change our scale to dollars
p + geom_point() + geom_smooth(method = "lm") + scale_x_log10(labels = scales::dollar)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

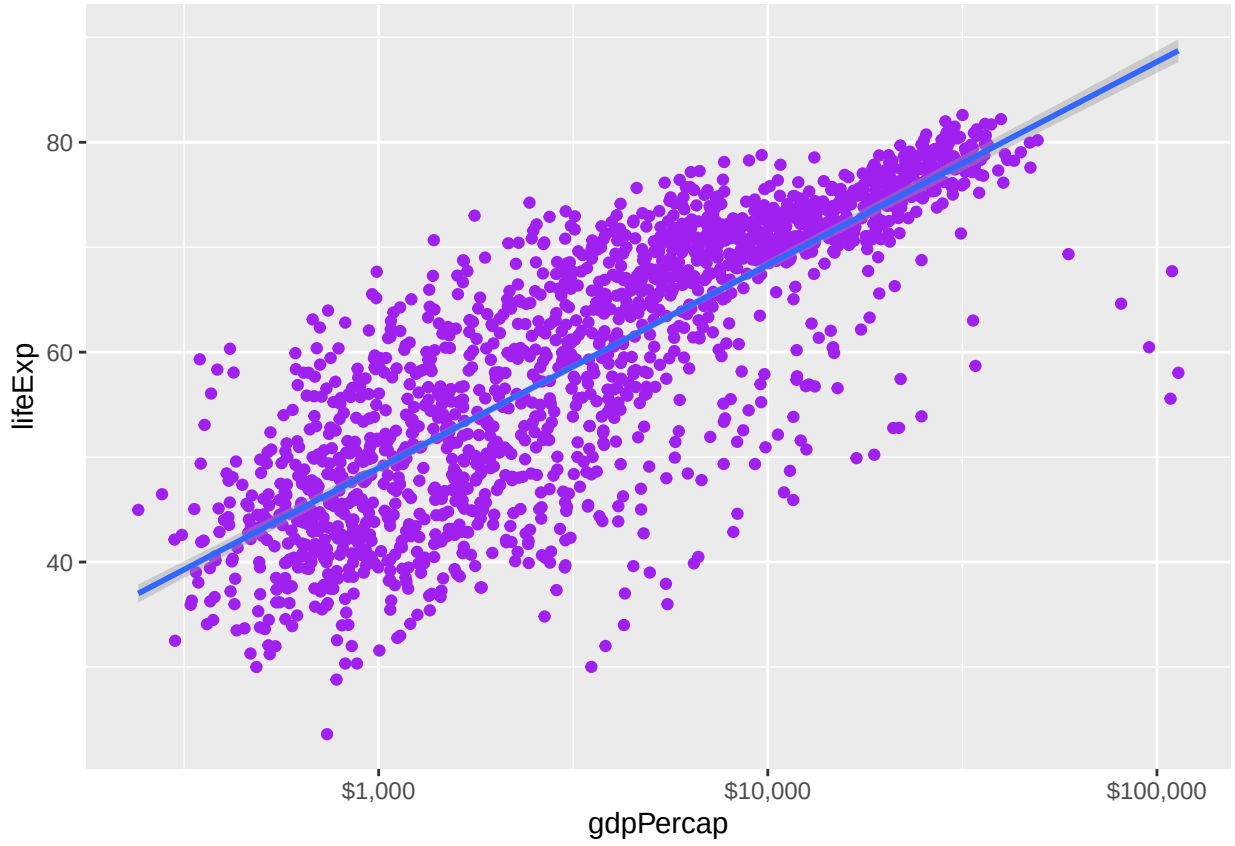



```
# aesthetic mapping allows us to express a variable with a given visual element. Here we can add colour
p1 <- ggplot(data = gapminder,
             mapping = aes(x = gdpPercap, y = lifeExp, colour = continent))


p1 + geom_smooth(method = "lm") + geom_point() + scale_x_log10(labels = scales::dollar)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

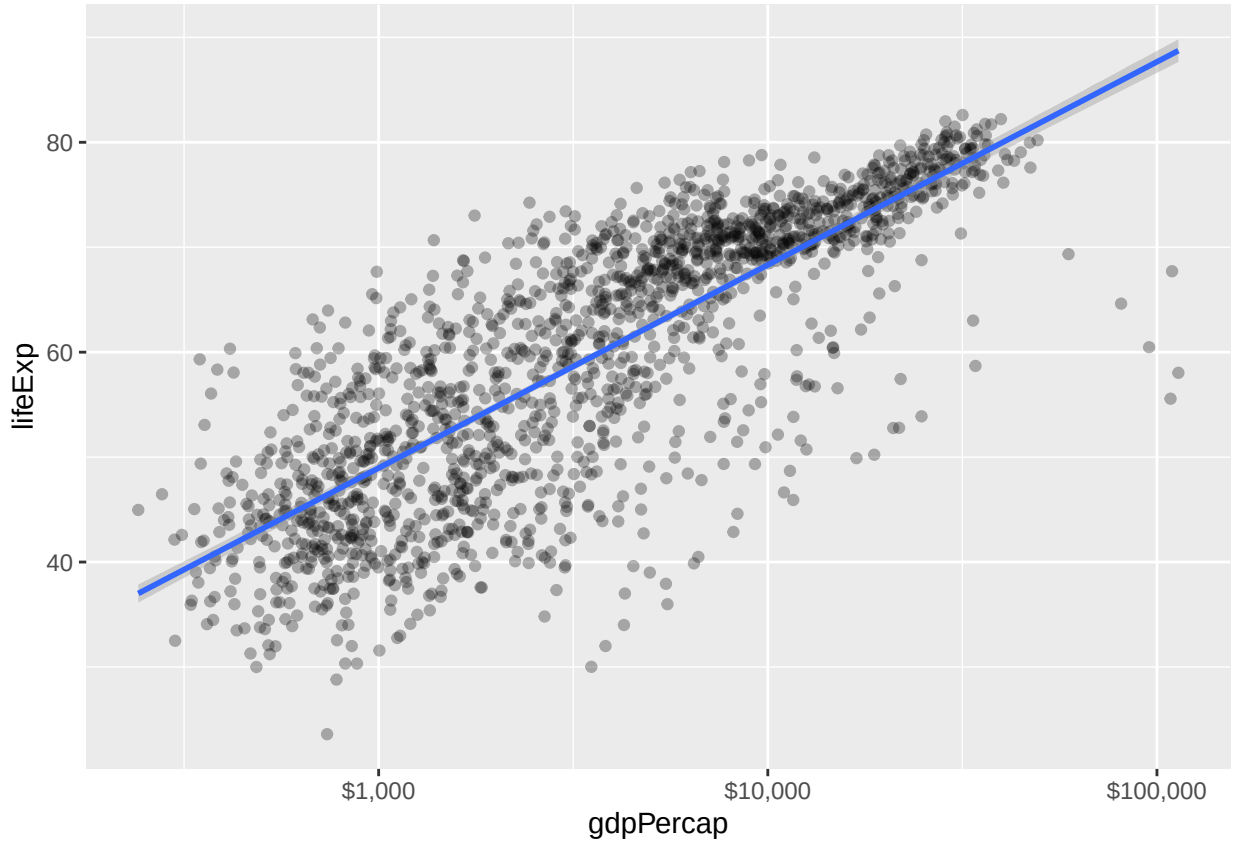

```
# aesthetic setting occurs in the geom() object not the ggplot(). This applies to the entire dataset
p + geom_point(colour = "purple") + geom_smooth(method = "lm") + scale_x_log10(labels = scales::dollar)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```




```
# we can also use aesthetic setting to change the opacity of our datasets points
p + geom_point(alpha = 0.3) + geom_smooth(method = "lm") + scale_x_log10(labels = scales::dollar)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

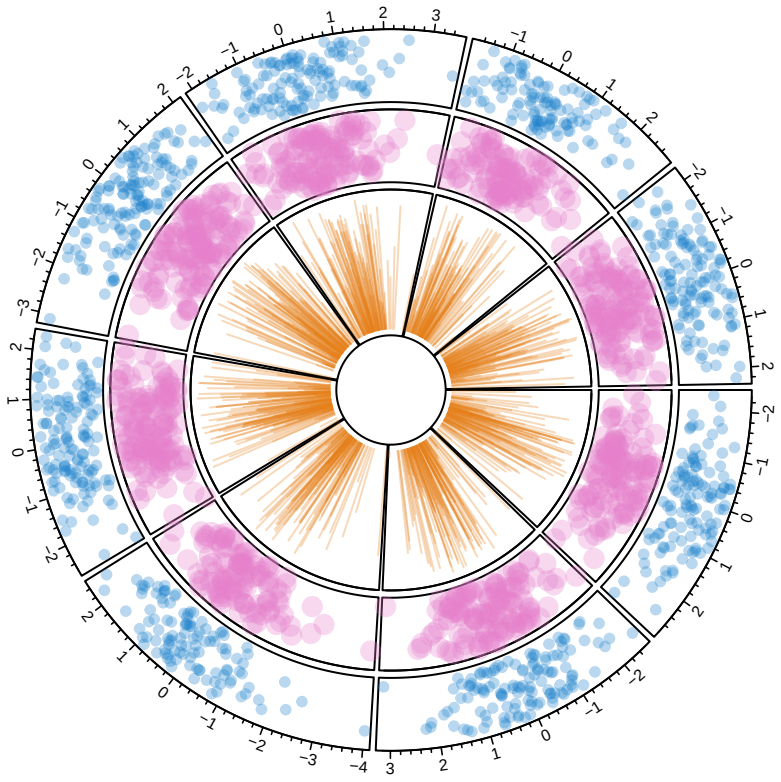



```
# we can use ggsave to save our most recent plot
#ggsave(filename = "sampleimage.png")
```

```
# take code from the R graph gallery to make custom circular chart with multiple tracks
# https://r-graph-gallery.com/227-add-several-tracks.html

#library
library(circlize)
```

```
## ========================================
## circlize version 0.4.15
## CRAN page: https://cran.r-project.org/package=circlize
## Github page: https://github.com/jokergoo/circlize
## Documentation: https://jokergoo.github.io/circlize_book/book/
##
## If you use it in published research, please cite:
## Gu, Z. circlize implements and enhances circular visualization
##   in R. Bioinformatics 2014.
##
## This message can be suppressed by:
##   suppressPackageStartupMessages(library(circlize))
## ========================================
```

```
circos.clear()

#Create data
data = data.frame(
    factor = sample(letters[1:8], 1000, replace = TRUE),
    x = rnorm(1000),
    y = runif(1000)
    )

#Initialize the plot.
par(mar = c(1, 1, 1, 1) )
circos.initialize(factors = data$factor, x = data$x )


# Build the regions of track #1
circos.trackPlotRegion(factors = data$factor, y=data$y, panel.fun = function(x, y) {
    circos.axis(labels.cex=0.5, labels.font=1, lwd=0.8)
    })
# --> Add a scatterplot on it:
circos.trackPoints(data$factor, data$x, data$y, col = rgb(0.1,0.5,0.8,0.3), pch=20)


# Build the regions of track #2:
circlize::circos.trackPlotRegion(factors = data$factor, y=data$y, panel.fun = function(x, y) {
    circos.axis(labels=FALSE, major.tick=FALSE)
    })
# --> Add a scatterplot on it
circos.trackPoints(data$factor, data$x, data$y, col = rgb(0.9,0.5,0.8,0.3), pch=20, cex=2)


# Add track #3 --> don't forget you can custom the height of tracks!
circos.par("track.height" = 0.4)
circos.trackPlotRegion(factors = data$factor, y=data$y, panel.fun = function(x, y) {
    circos.axis(labels=FALSE, major.tick=FALSE)
    })
circos.trackLines(data$factor, data$x, data$y, col = rgb(0.9,0.5,0.1,0.3), pch=20, cex=2, type="h")
```



```
# and continue as long as needed!
```

- It is aesthetically pleasing to look at
- I am not familiar with these kinds of graphs, so I'm not sure whether it accurately represents data
- Furthermore, it is challenging for me to know what the graph is trying to convey
- The lack of labels/legend makes it challenging to interpret