Class 5: Data Viz with ggplot

Jessica Gao (PID16939806)

##Intro to ggplot

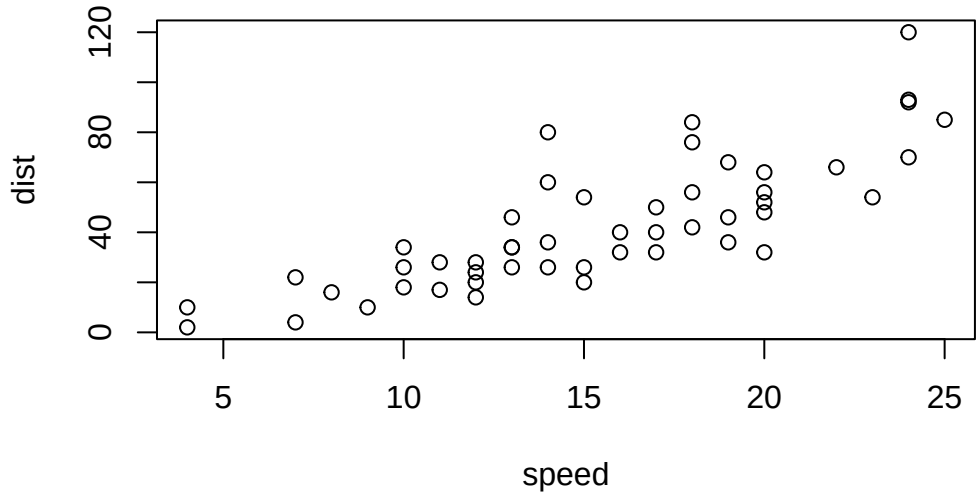
There are many graphics systems in R (ways to make plots and figures). These include "base" R plots. Today we will focus mostly on the **ggplot2** package.

Let's start with a plot of a simple in-built dataset called 'cars'.

```
head(cars, n=10)
```

```
   speed dist
1      4    2
2      4   10
3      7    4
4      7   22
5      8   16
6      9   10
7     10   18
8     10   26
9     10   34
10    11   17
```

```
plot(cars)
```



Let's see how we can make this figure using **ggplot** First I need to install this package on my computer. To install any R package I use the function 'install.packages()'.

I will run 'install.packages("ggplot2") in my R console not this quarto document!

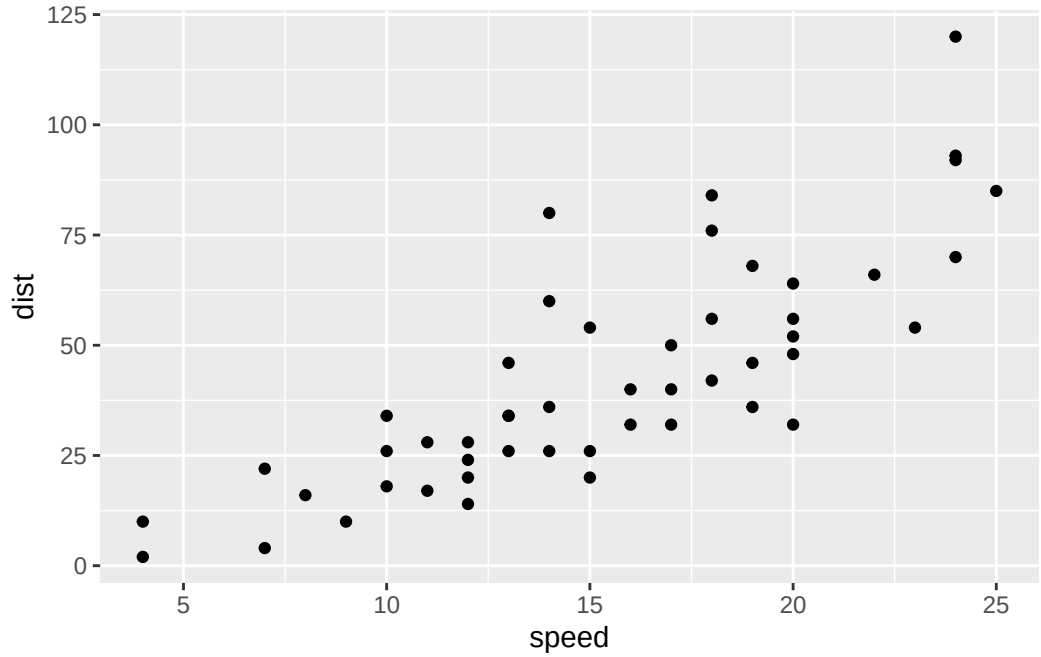
Before I can use any functions from add on packages I need to load the package from my "library()" with the 'library(ggplot2)' call.

```
#install.packages("ggplot2")
library(ggplot2)
ggplot(cars)
```

All ggplot figures have at least 3 things (called layers). These include:

- **data** (the input dataset I want to plot from),
- **aes** (the aesthetic mapping of the data to my plot),
- **geoms** (the geom_point(), geom_line() etc. that I want to draw).

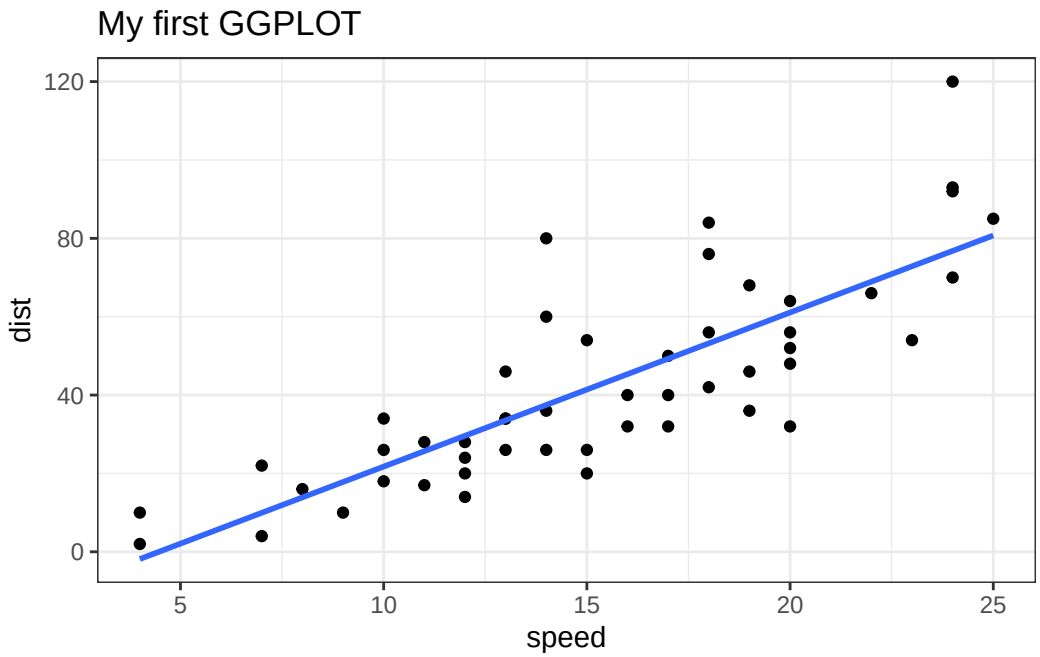
```
ggplot(cars)+
  aes(x=speed,y=dist)+
  geom_point()
```



Let's add a line to show the relationship here:

```
ggplot(cars)+
  aes(x=speed,y=dist)+
  geom_point()+
  geom_smooth(method="lm", se=FALSE)+
  theme_bw()+
  labs(title="My first GGPLOT")
```

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



Q1 Which geometric layer should be used to create scatter plots in ggplot2?

geom_point()

Q2. In your own RStudio can you add a trend line layer to help show the relationship between the plot variables with the geom_smooth() function?

geom_smooth(method="lm")

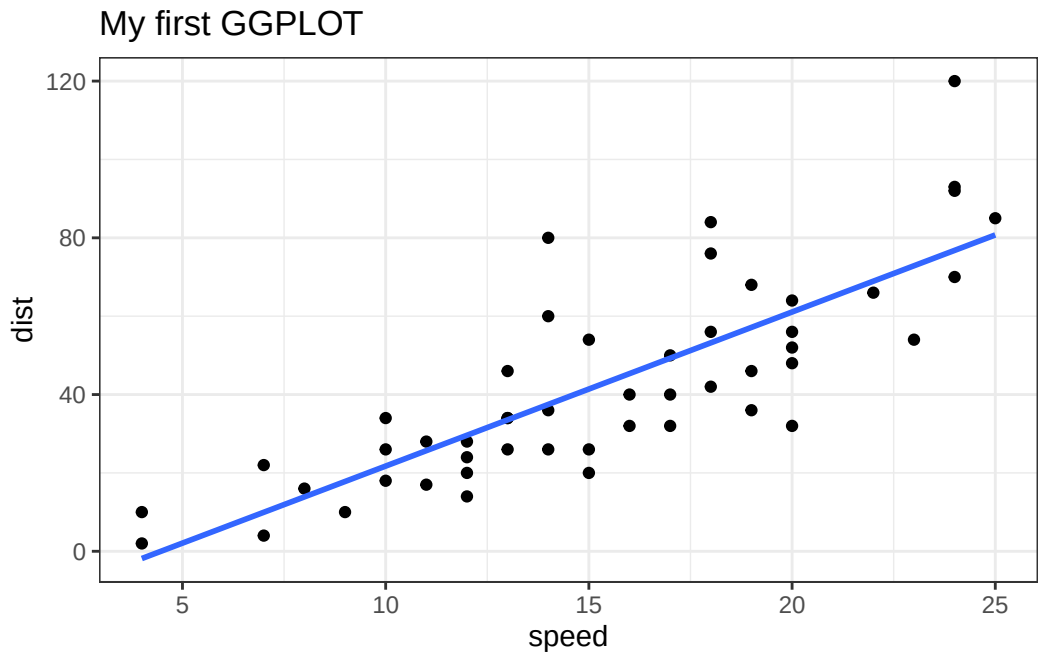
Q3. Argue with geom_smooth() to add a straight line from a linear model without the shaded standard error region?

geom_smooth(method="lm", se=FLASE)

Q4. Can you finish this plot by adding various label annotations with the labs() function and changing the plot look to a more conservative "black & white" theme by adding the theme_bw() function:

```
ggplot(cars)+
  aes(x=speed,y=dist)+
  geom_point()+
  geom_smooth(method="lm", se=FALSE)+
  theme_bw()+
  labs(title="My first GGPLOT")
```

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



##Gene expression figure

The code to read the dataset

```
url <- "https://bioboot.github.io/bimm143_S20/class-material/up_down_expression.txt"
genes <- read.delim(url)
head(genes)
```

```
        Gene Condition1 Condition2      State
1      A4GNT -3.6808610 -3.4401355 unchanging
2       AAAS  4.5479580  4.3864126 unchanging
3      AASDH  3.7190695  3.4787276 unchanging
4       AATF  5.0784720  5.0151916 unchanging
5       AATK  0.4711421  0.5598642 unchanging
6 AB015752.4 -3.6808610 -3.5921390 unchanging
```

> How many genes are in this dataset? Use the nrow() function to find out how many genes are in this dataset. What is your answer?

```
nrow(genes) #tells you how many rows there are
```

```
[1] 5196
```

```
#each row is a gene, so knowing the number of row you know how many genes.
```

> Q. Use the colnames() function and the ncol() function on the genes data frame to find out what the column names are (we will need these later) and how many columns there are. How many columns did you find?

```
colnames(genes)
```

```
[1] "Gene"       "Condition1" "Condition2" "State"
```

```
ncol(genes)
```

```
[1] 4
```

> Q. Use the table() function on the State column of this data.frame to find out how many 'up' regulated genes there are. What is your answer?

```
table(genes$State)
```

```
      down unchanging         up
        72       4997        127
```

127 genes are "up" regulated

> Q Use the table() function on the State column of this data.frame to find out how many 'up' regulated genes there are. What is your answer?

```
round (table(genes$State)/nrow(genes), 2)
```

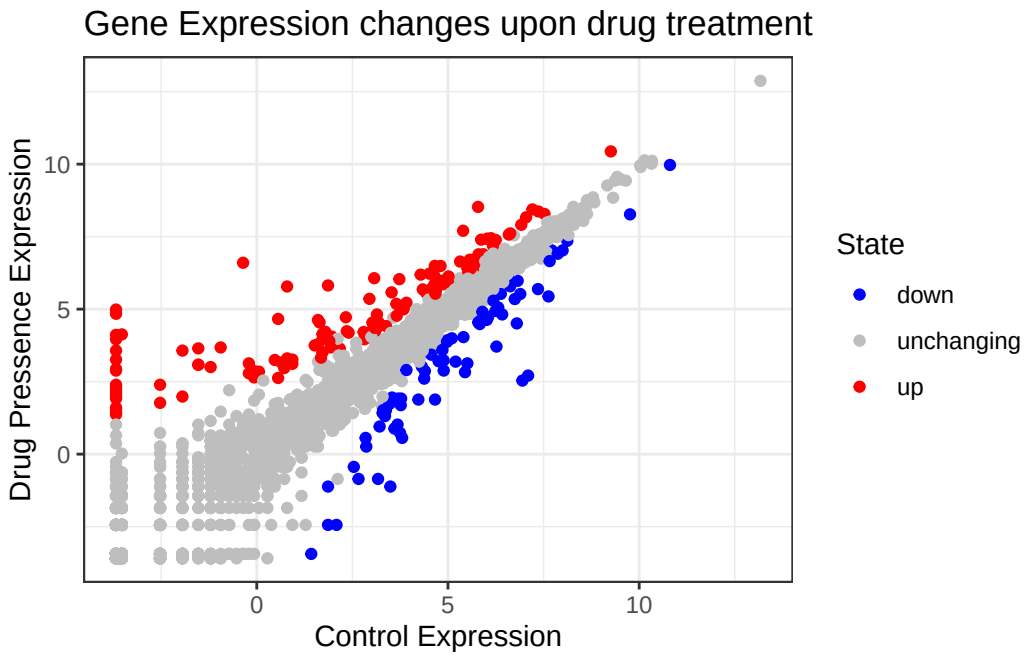
```
      down unchanging         up
      0.01       0.96       0.02
```

```
n.tot <- nrow(genes)
vals <- table(genes$State)

vals.percent <- vals/n.tot*100
round(vals.percent, 2)
```

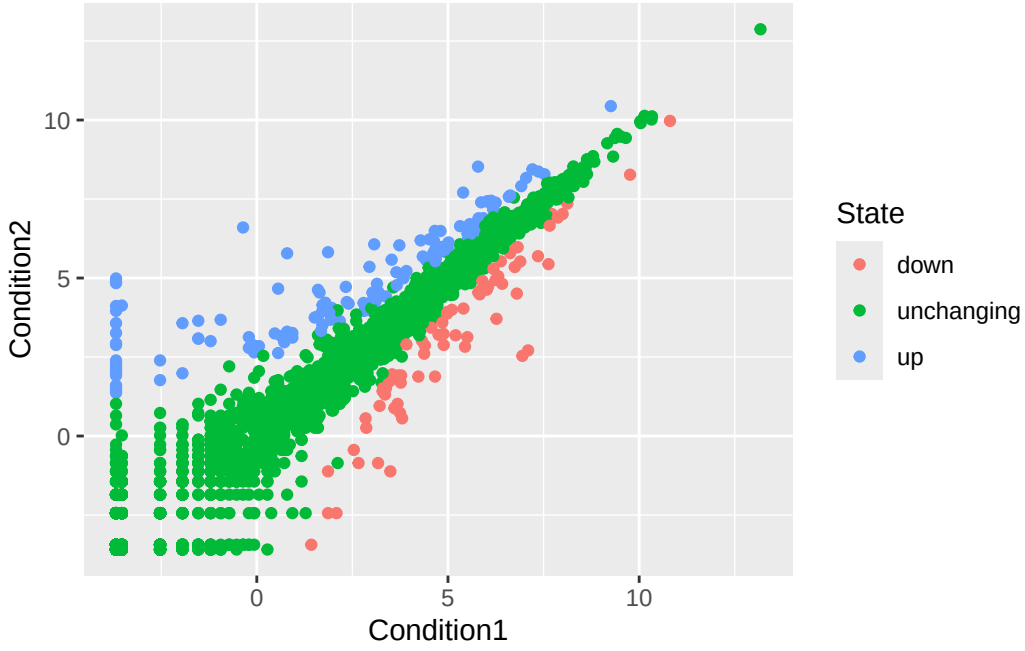
```
      down unchanging         up
      1.39      96.17       2.44
```

A first plot of this dataset

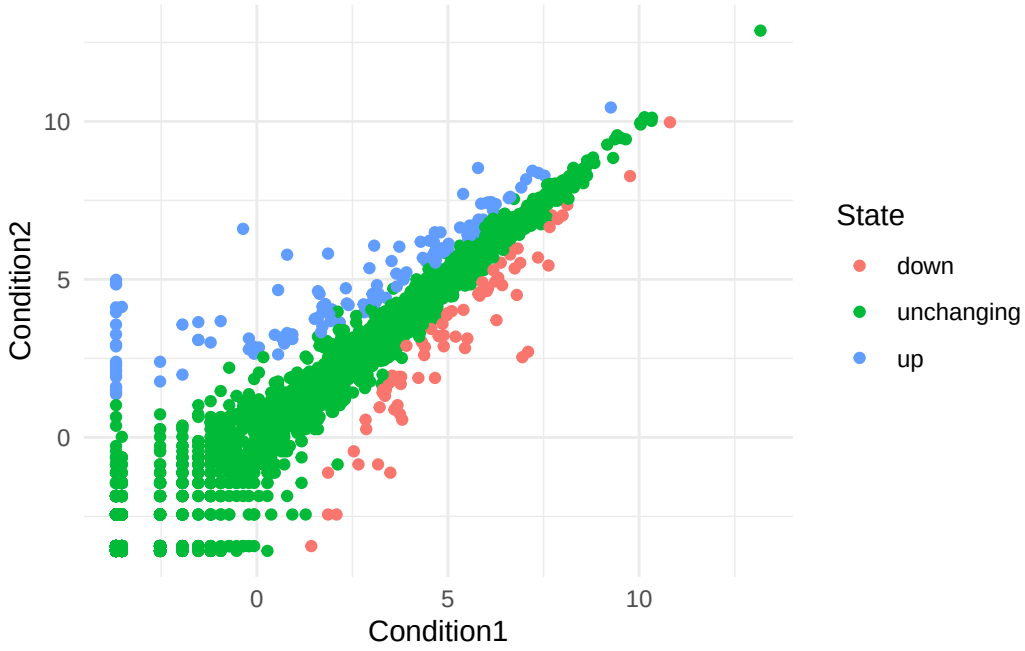
```
ggplot(genes)+
  aes(x=Condition1, y=Condition2, col=State)+
  geom_point()+
  theme_bw()+
  labs(title="Gene Expression changes upon drug treatment",
       x="Control Expression",
       y="Drug Pressence Expression") +
  scale_colour_manual(values=c("blue", "gray","red"))
```



```
p <- ggplot(genes)+ aes(x=Condition1, y=Condition2, col=State) + geom_point()
p
```




```
p + theme_minimal()
```



Section 7: Going Further**

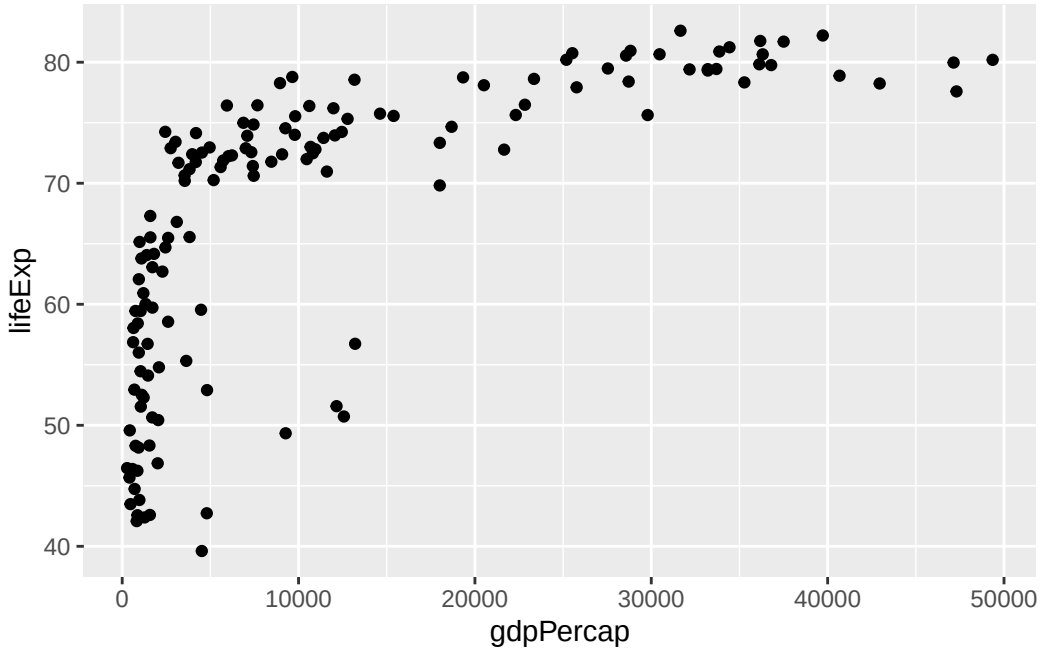
```
#install.packages("gapminder")
library(gapminder)
```

```
#install.packages("dplyr")
library(dplyr)
```

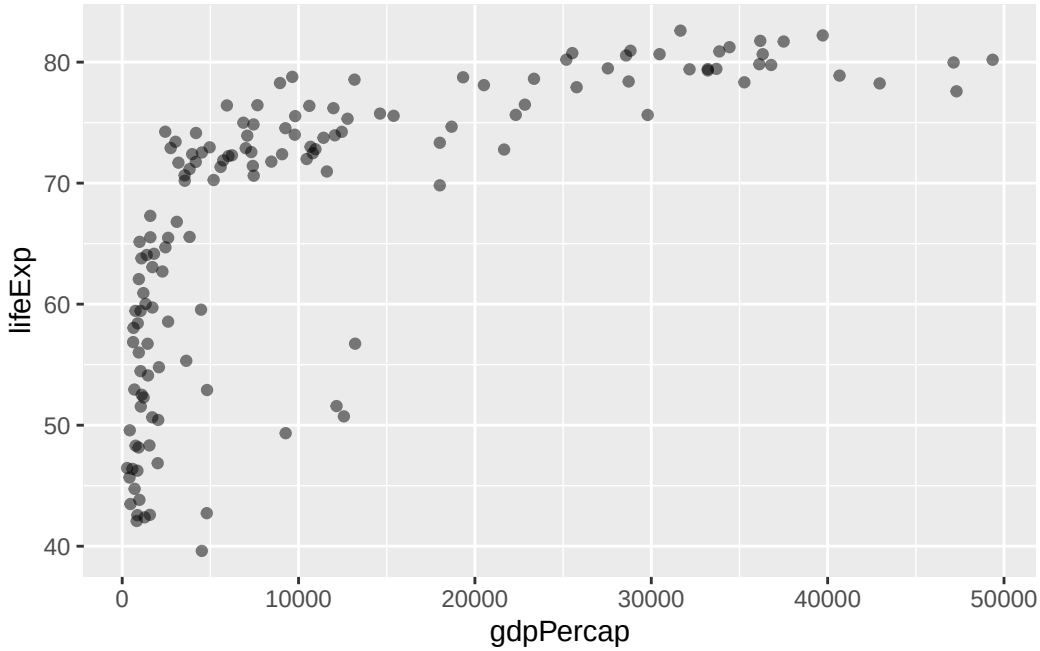
```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

Q.Complete the code below to produce a first basic scater plot of this gapminder_2007 dataset:

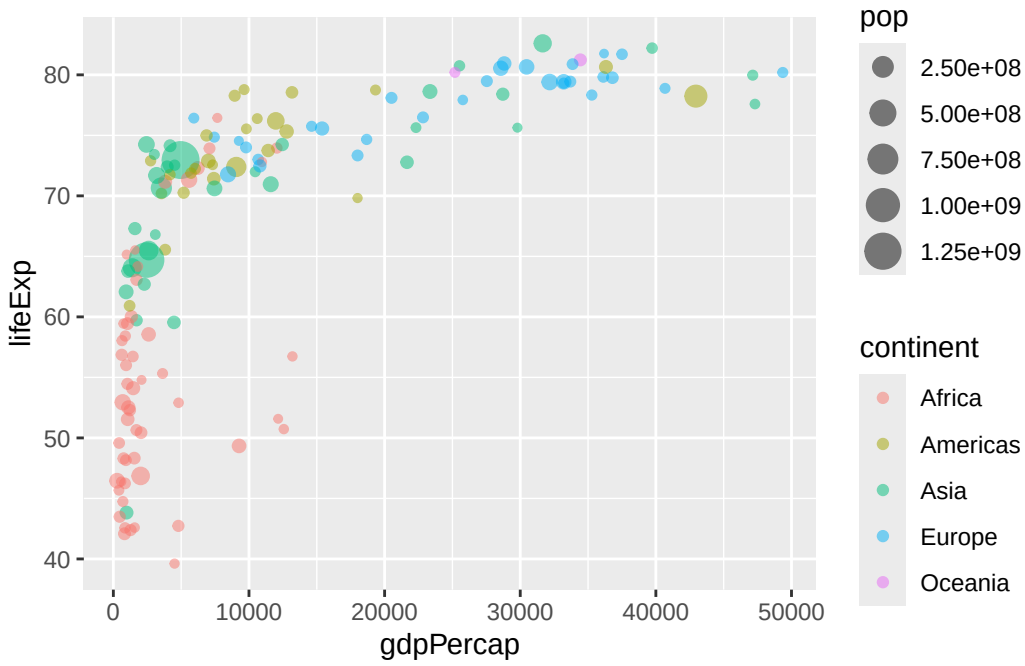
```r
gapminder_2007 <- gapminder %>% filter(year==2007)
ggplot(gapminder_2007) +
  aes(x=gdpPercap, y=lifeExp) +
  geom_point()
```



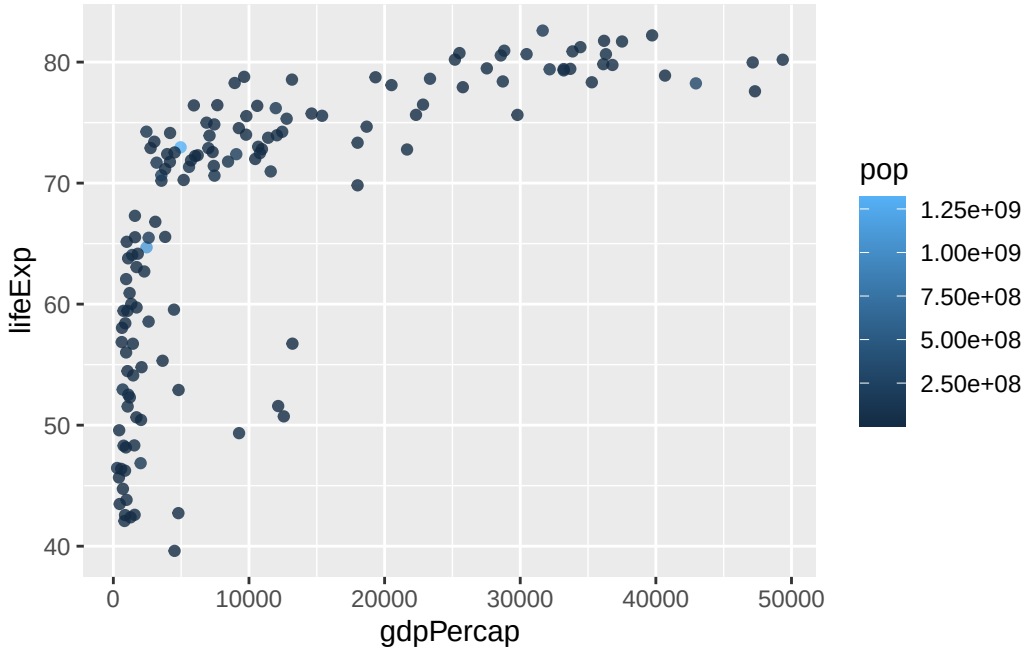

```r
ggplot(gapminder_2007) +
  aes(x=gdpPercap, y=lifeExp) +
  geom_point(alpha=0.5)
```




```
ggplot(gapminder_2007) +
  aes(x=gdpPercap, y=lifeExp, color=continent, size=pop) +
  geom_point(alpha=0.5)
```

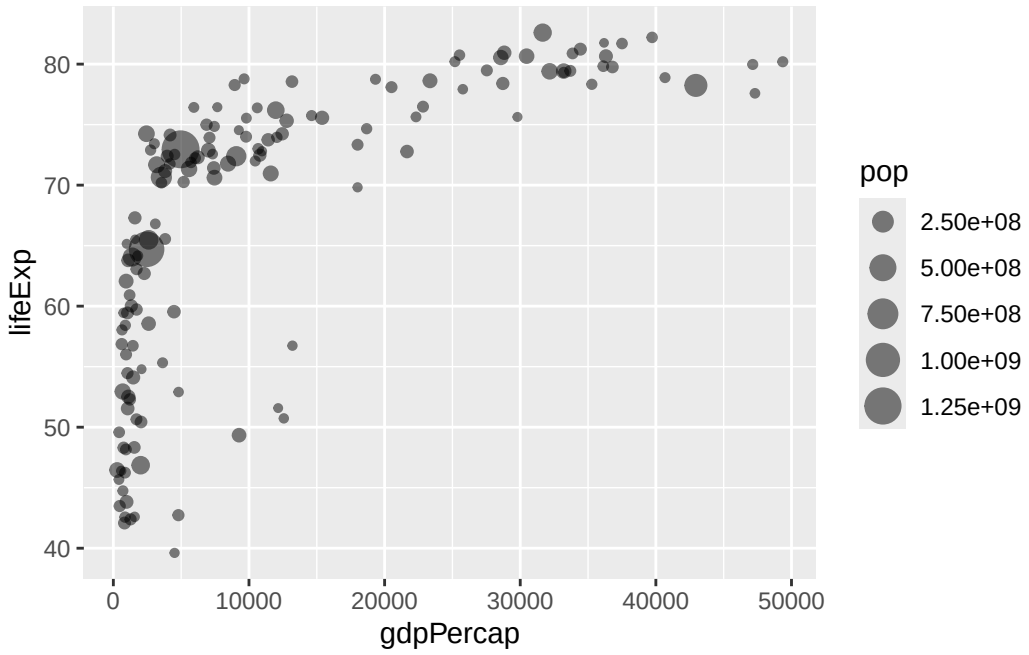


```
ggplot(gapminder_2007) +
  aes(x = gdpPercap, y = lifeExp, color = pop) +
  geom_point(alpha=0.8)
```

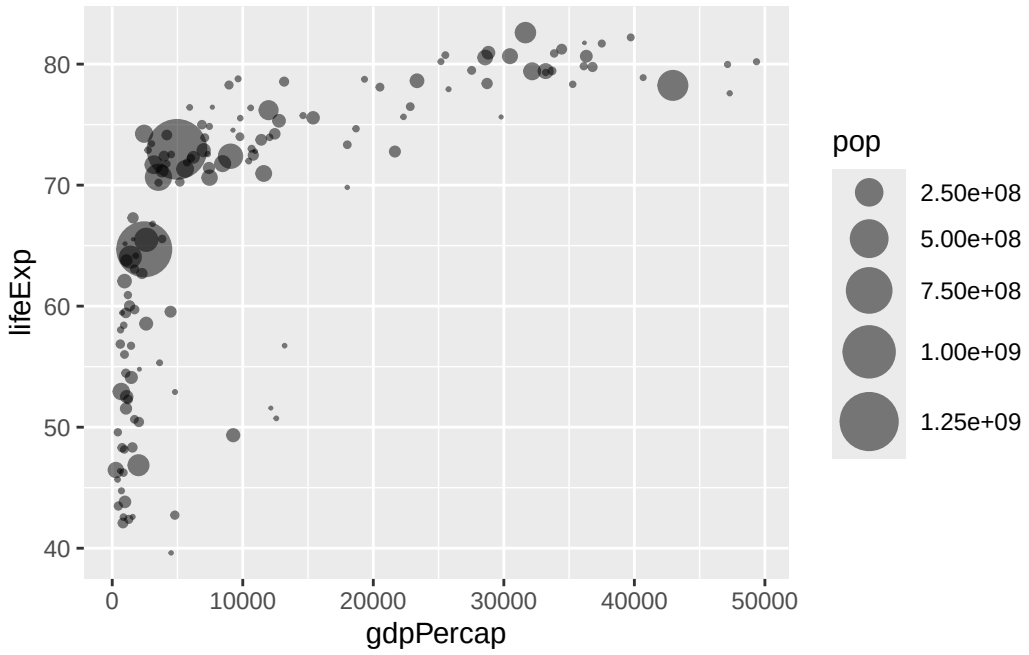


Section 7: Adjusting point size

```
ggplot(gapminder_2007) +
  aes(x = gdpPercap, y = lifeExp, size = pop) +
  geom_point(alpha=0.5)
```

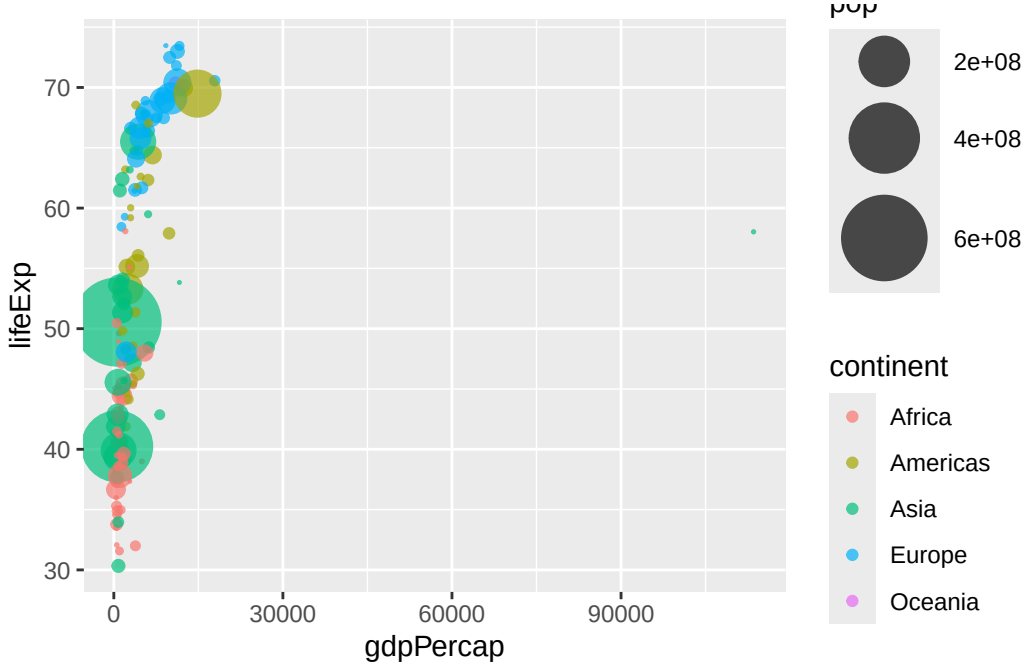



```
ggplot(gapminder_2007) +
  geom_point(aes(x = gdpPercap, y = lifeExp,
                 size = pop), alpha=0.5) +
  scale_size_area(max_size = 10)
```



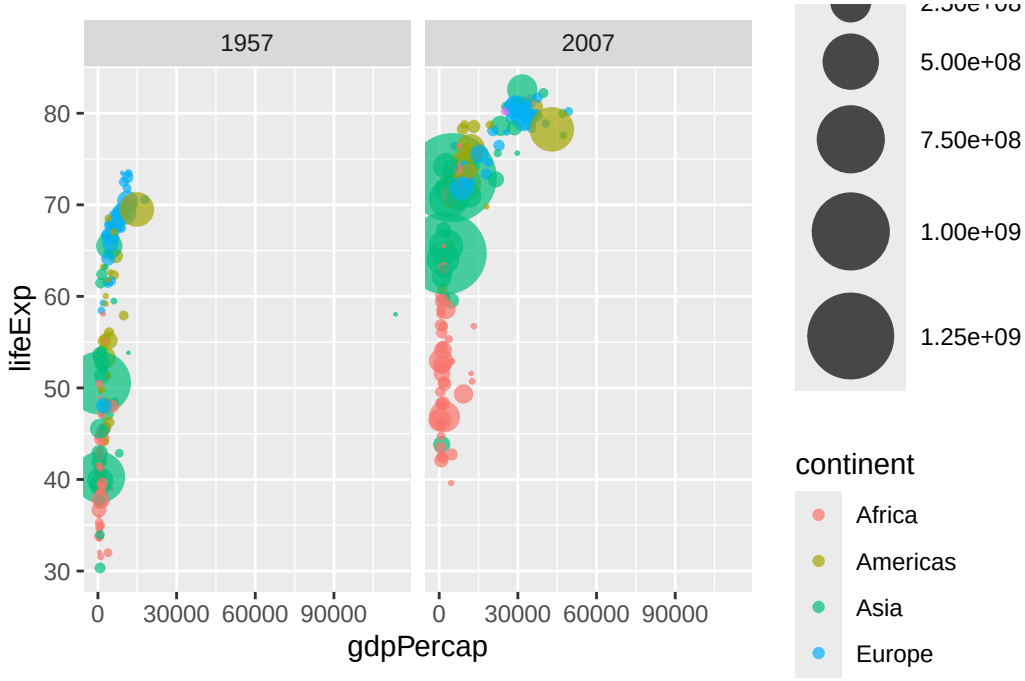
Q. Can you adapt the code you have learned thus far to reproduce our gapminder scatter plot for the year 1957? What do you notice about this plot is it easy to compare with the one for 2007?

```
gapminder_1957 <- gapminder %>% filter(year==1957)
ggplot(gapminder_1957)+
  aes(x=gdpPercap, y=lifeExp, size= pop, color=continent)+
  scale_size_area(max_size=15)+
  geom_point(alpha=0.7)
```



Q. Do the same steps above but include 1957 and 2007 in your input dataset for ggplot(). You should now include the layer facet_wrap(~year) to produce the following plot:

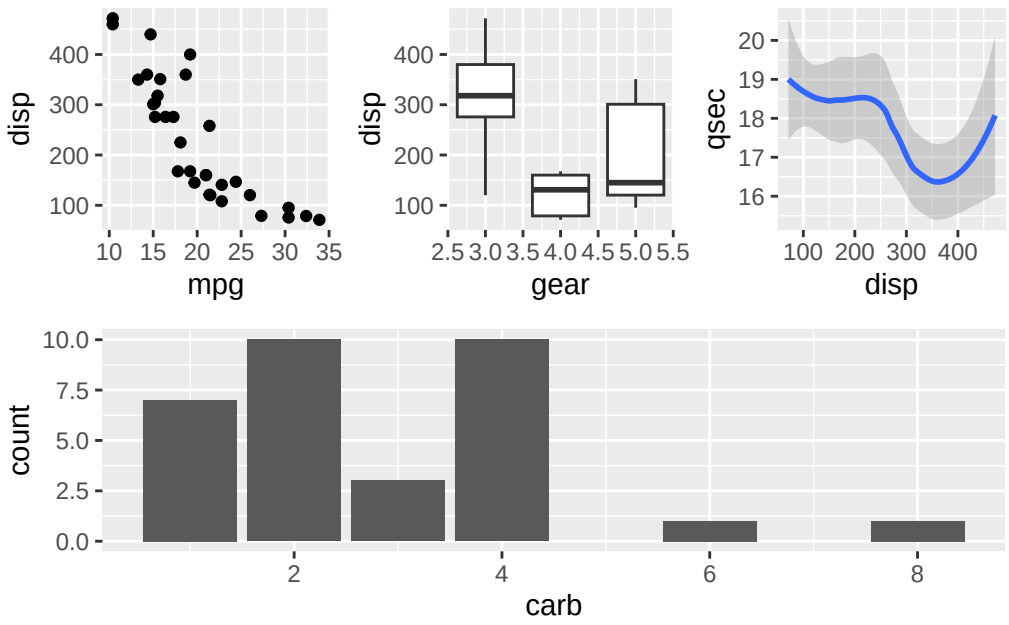
```
gapminder_both_years <- gapminder %>% filter(year==1957 | year==2007)
ggplot(gapminder_both_years)+
  aes(x=gdpPercap, y=lifeExp, size= pop, color=continent)+
  scale_size_area(max_size=15)+
  geom_point(alpha=0.7)+
  facet_wrap(~year)
```



Section 10: Combining plots

```
#install.packages("patchwork")
library(patchwork)

# Setup some example plots
scatter_plot <- ggplot(mtcars) + geom_point(aes(mpg, disp))
box_whisker_plot <- ggplot(mtcars) + geom_boxplot(aes(gear, disp, group = gear))
curve_graph <- ggplot(mtcars) + geom_smooth(aes(disp, qsec))
bar_graph <- ggplot(mtcars) + geom_bar(aes(carb))

# Use patchwork to combine them here:
(scatter_plot | box_whisker_plot | curve_graph) /
      bar_graph
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```




```
sessionInfo()
```

```
R version 4.4.2 (2024-10-31 ucrt)
Platform: x86_64-w64-mingw32/x64
Running under: Windows 11 x64 (build 26100)

Matrix products: default


locale:
[1] LC_COLLATE=English_United States.utf8
[2] LC_CTYPE=English_United States.utf8
[3] LC_MONETARY=English_United States.utf8
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.utf8

time zone: America/Los_Angeles
tzcode source: internal

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base
```

```
other attached packages:
[1] patchwork_1.3.0 dplyr_1.1.4     gapminder_1.0.0 ggplot2_3.5.1

loaded via a namespace (and not attached):
 [1] vctrs_0.6.5       nlme_3.1-166      cli_3.6.3         knitr_1.49
 [5] rlang_1.1.4       xfun_0.50         generics_0.1.3    jsonlite_1.8.9
 [9] labeling_0.4.3    glue_1.8.0        colorspace_2.1-1  htmltools_0.5.8.1
[13] scales_1.3.0      rmarkdown_2.29    grid_4.4.2        evaluate_1.0.3
[17] munsell_0.5.1     tibble_3.2.1      fastmap_1.2.0     yaml_2.3.10
[21] lifecycle_1.0.4   compiler_4.4.2    pkgconfig_2.0.3   mgcv_1.9-1
[25] lattice_0.22-6    farver_2.1.2      digest_0.6.37     R6_2.5.1
[29] tidyselect_1.2.1  splines_4.4.2     pillar_1.10.1     magrittr_2.0.3
[33] Matrix_1.7-1      withr_3.0.2       tools_4.4.2       gtable_0.3.6
```