L11. *ggplot* (5)

More functionality

Sim, Min Kyu, Ph.D., mksim@seoultech.ac.kr

SEOULTECH 서울과학기술대학교 데이터사이언스학과

1. I. Labels
2. II. Legend
3. III. Scale
4. IV. Theme

Overview

```
ggplot(data=<DATA>) +
  <GEOM_FUNCTION> (
    mapping = aes(<MAPPINGS>),
    stat = <STAT>,
    position = <POSITION>) +
  <COORDINATE_FUNCTION> +
  <FACET_FUNCTION> +
  <SCALE_FUNCTION> +
  <THEME_FUNCTION>
```

1. Labels
2. Legend
3. Scale: Scales map data values to the visual values of an aesthetic. To change a mapping, add a new scale.
4. Theme

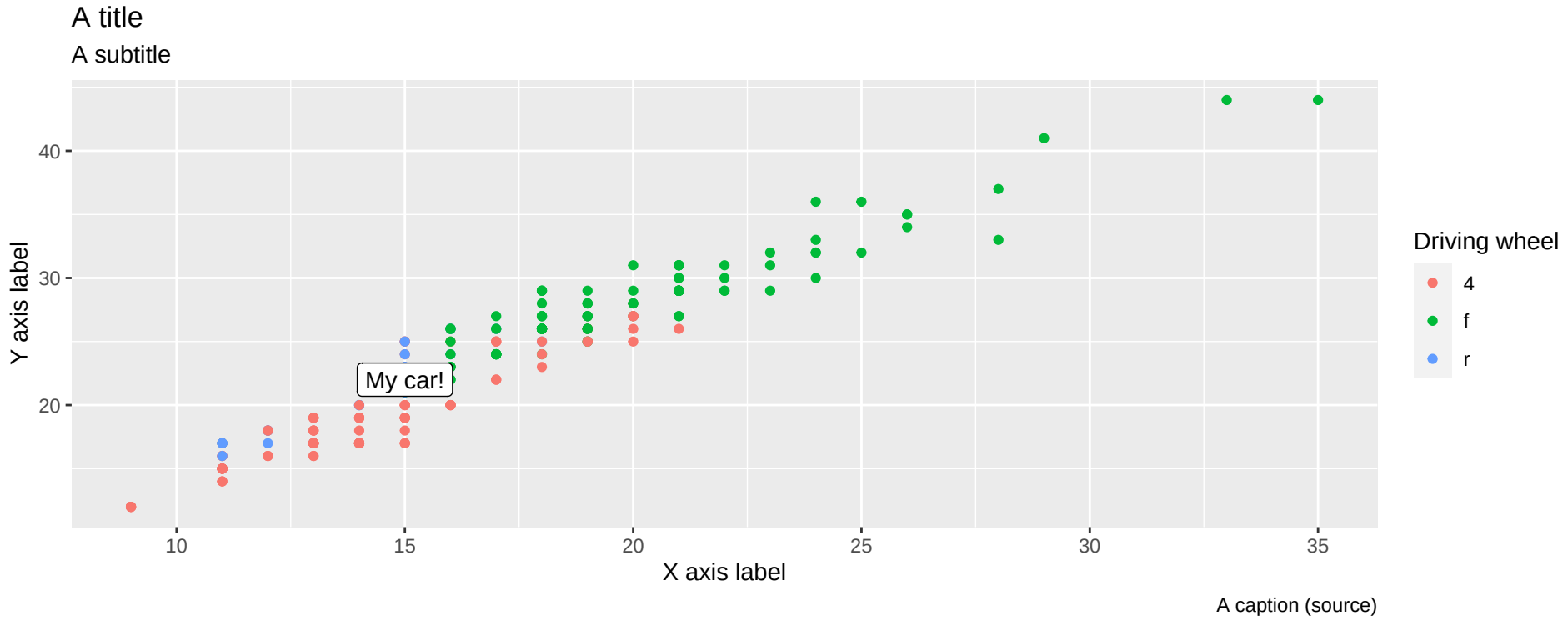
Section 1

I. Labels

- 각 Aesthetics(`x`,`y`,`color`,`size`…)에 대해 label을 넣음
- 전체 그래픽(Title, Subtitle, Caption)에 대해 label을 넣음
- `annotate()` 함수를 이용해 임의의 geometric object를 추가

```
fig <- ggplot(mpg, aes(cty, hwy, color=drv)) + geom_point() +
  labs(x = "X axis label",
       y = "Y axis label",
       color = "Driving wheel",
       title ="A title",
       subtitle = "A subtitle",
       caption = "A caption (source)") +
  annotate(geom = "label", x = 15, y = 22, label = "My car!")
```

```
fig
```

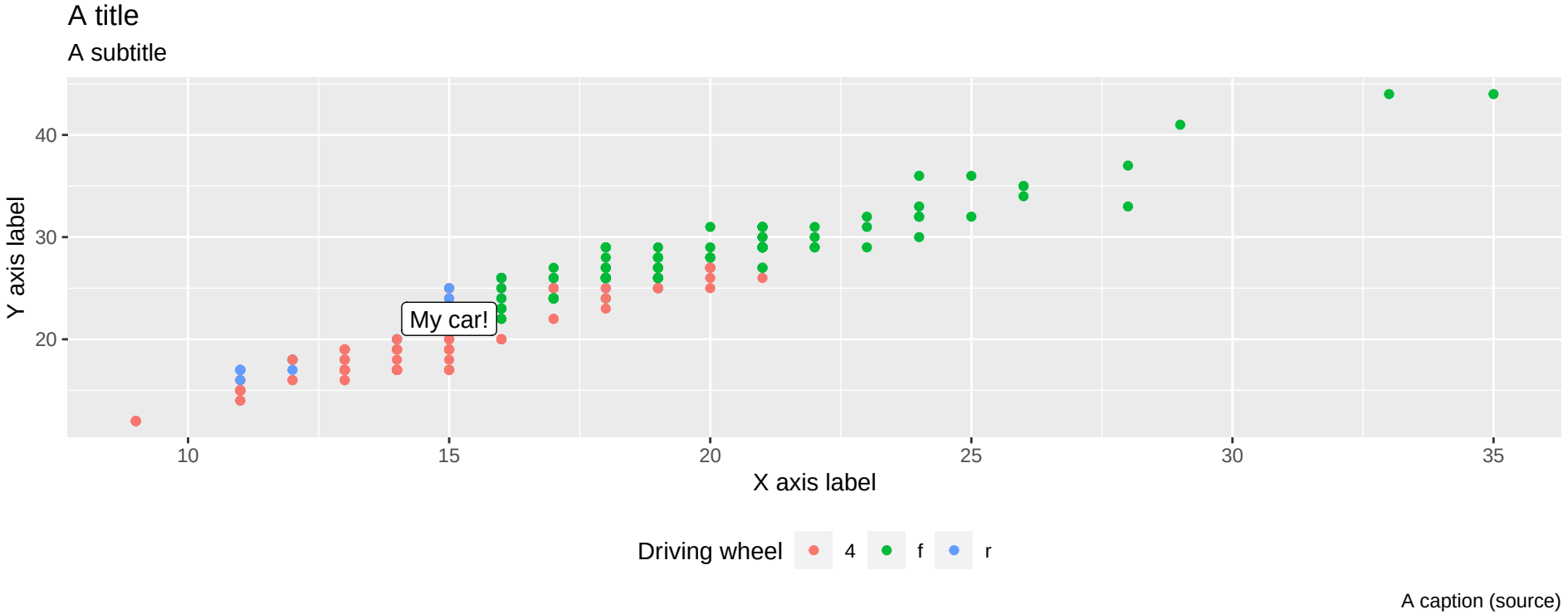

A title
A subtitle
Y axis label
X axis label
My car!
A caption (source)
Driving wheel
4
f
r
40
30
20
10
15
20
25
30
35

Section 2

II. Legend

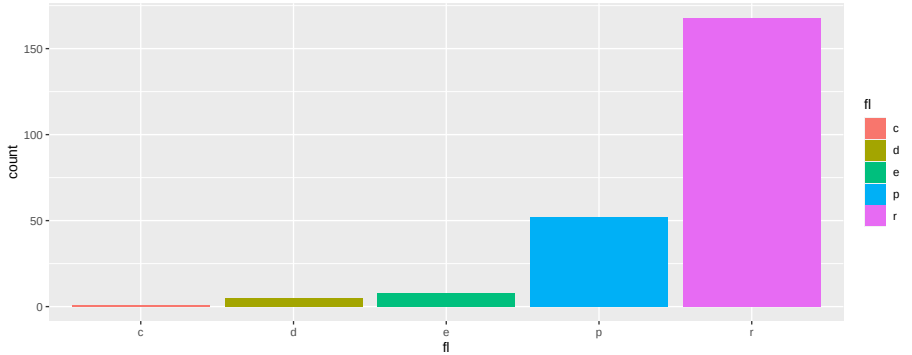
theme(legend.position=): move legends around

```
# Choose among c("top", "bottom", "left", "right")
fig + theme(legend.position = "bottom")
```

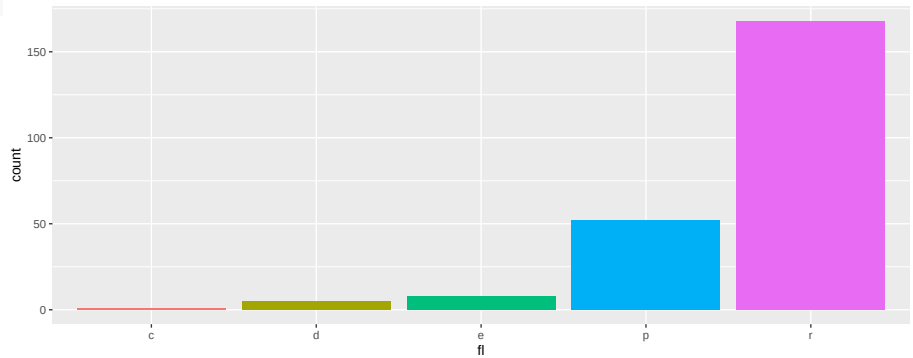


guides(AES = "none"): remove the legend of the *AES*

```
ggplot(mpg, aes(x=fl, fill = fl)) + geom_bar()
```

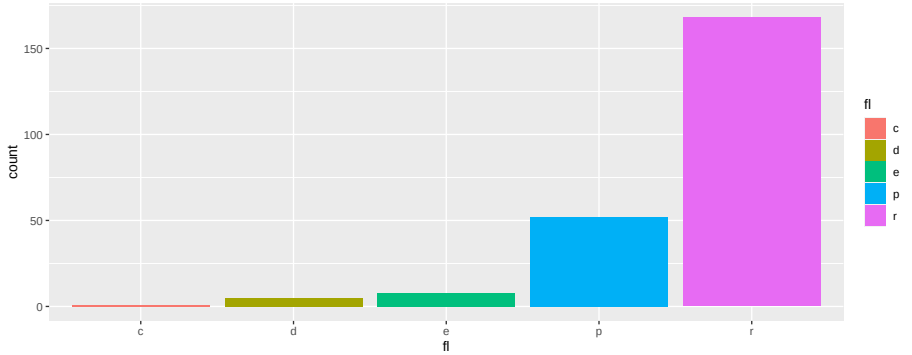



```
ggplot(mpg, aes(x=fl, fill = fl)) + geom_bar() +
  guides(fill = "none")
```



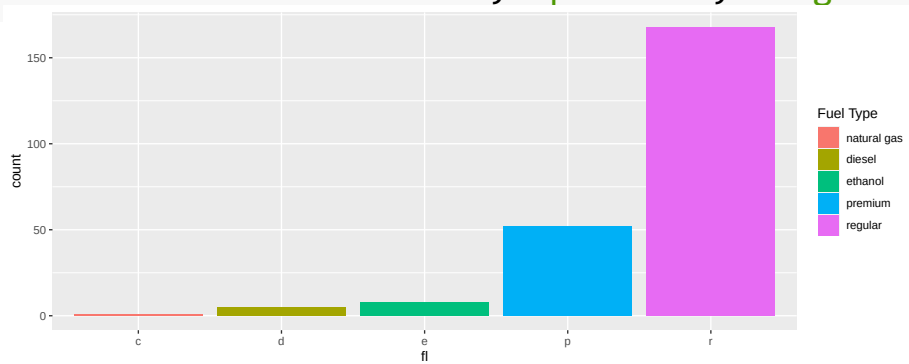
Customizing legend labels

```
ggplot(mpg, aes(x=fl, fill = fl)) + geom_bar()
```



- scale_AES_type()
  - name=<legend name>
  - label=<each label>
- AES
- type
- name
- label

```
ggplot(mpg, aes(x=fl, fill = fl)) + geom_bar() +
  scale_fill_discrete(
    name = "Fuel Type",
    labels = c("natural gas", "diesel",
               "ethanol", "premium", "regular"))
```



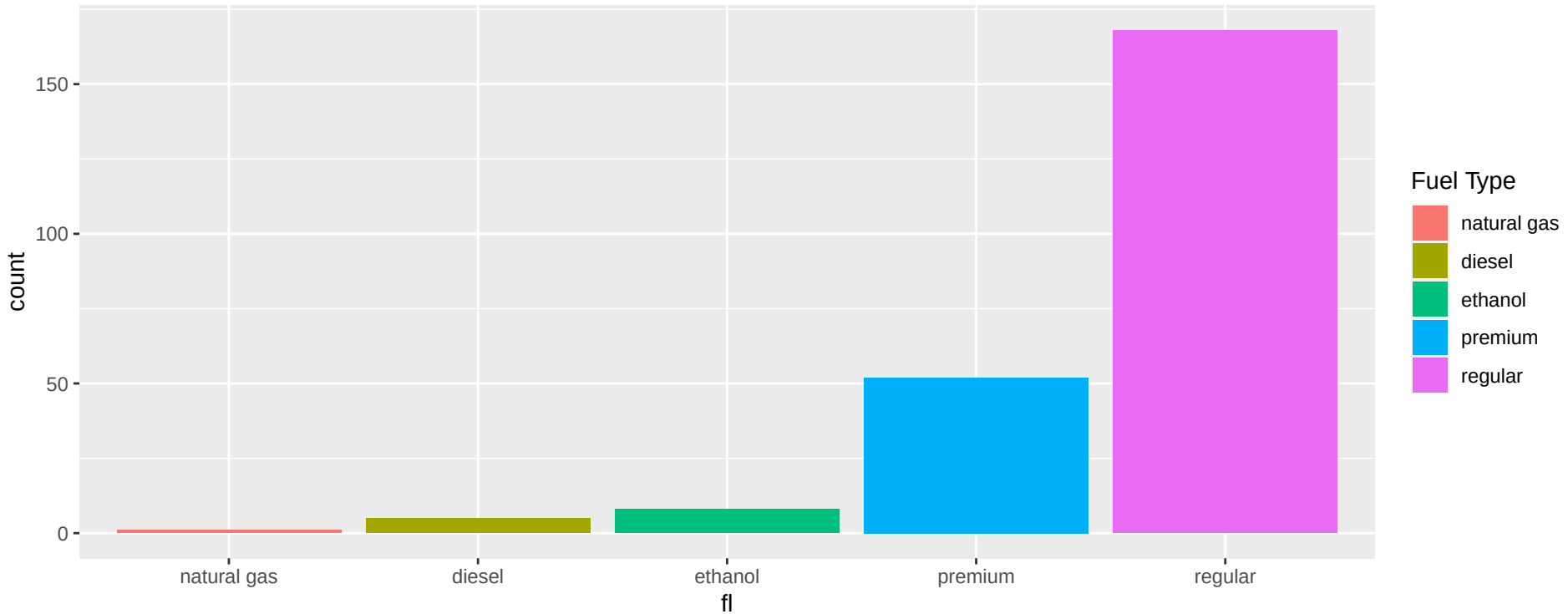
Section 3

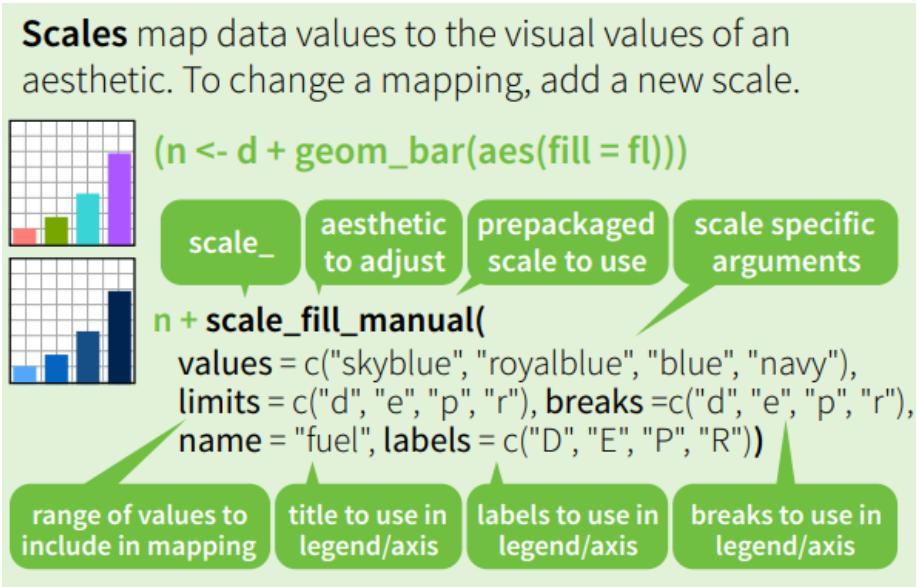
III. Scale

- 데이터 값을 aesthetic에 매핑하는 역할을 함
- Scales map data values to the visual values of an aesthetic. To change a mapping, add a new scale.

```
fig <- ggplot(mpg, aes(x=fl, fill = fl)) + geom_bar() +
  scale_fill_discrete(
    name = "Fuel Type",
    labels = c("natural gas", "diesel", "ethanol", "premium", "regular")) +
  scale_x_discrete(
    labels = c("natural gas", "diesel", "ethanol", "premium", "regular"))
```

```
fig
```


150
100
50
0
count
natural gas
diesel
ethanol
premium
regular
fl
Fuel Type
natural gas
diesel
ethanol
premium
regular




```
fig1 <- ggplot(mpg, aes(x=drv, fill=drv)) +
  geom_bar()
```

```
fig2 <- fig1 +
  scale_fill_manual(
    values = c("royalblue", "skyblue"),
    limits = c("f", "r"), # you may filter
    breaks = c("f", "r"),
    name = "FWD vs RWD",
    labels = c("Forward", "Rear"))
```

fig1

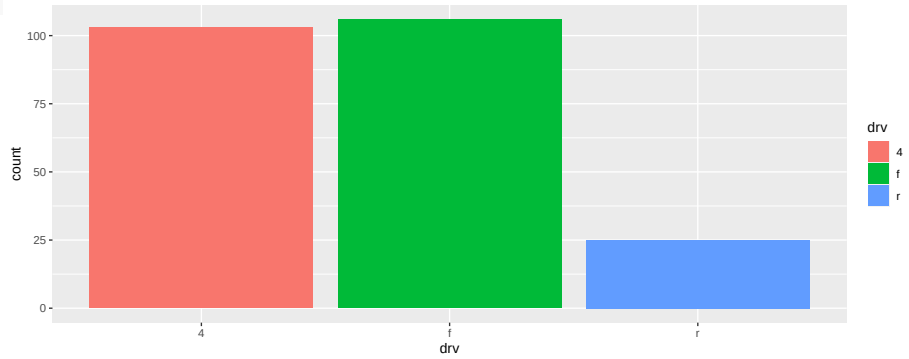


fig2

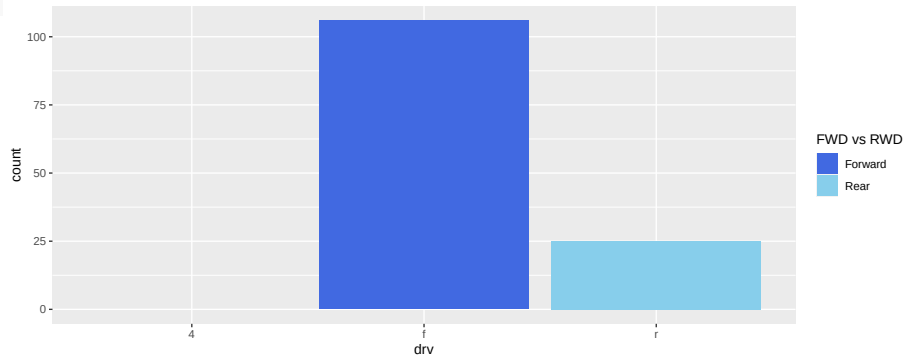
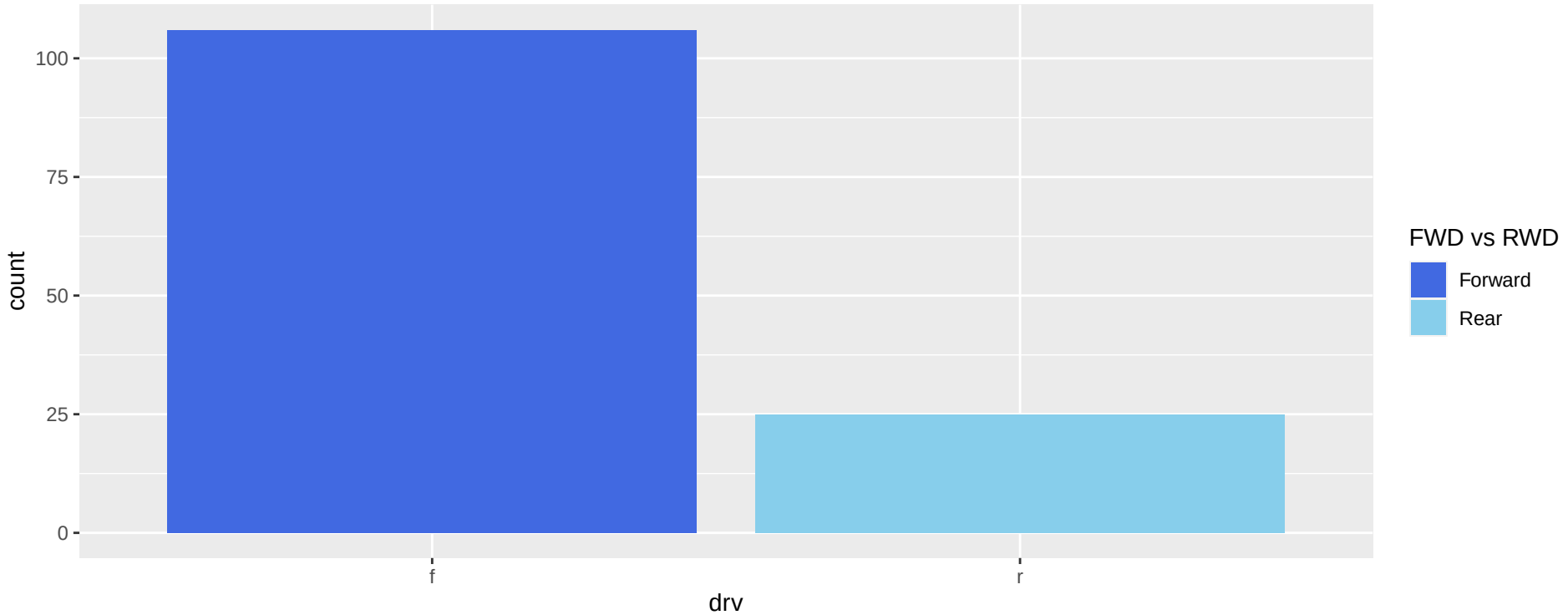


fig2에서 '4'는 왜 안 없어지고 빈칸일까?

General Purpose scales (use with most aesthetics)

- `fig2`에서 '4'는 왜 안 없어지고 빈칸일까?
- Aesthetic `x`에서 `limits`를 걸어주면 된다!

```
fig2 + scale_x_discrete(limits = c("f", "r"))
```



- `scale_*_continuous()`
  - map cont' values to visual ones
- `scale_*_discrete()`
  - map discrete values to visual ones
- `scale_*_identity()`
  - use data values as visual ones
- `scale_*_manual(values = c())`
  - map discrete values to manually chosen visual ones
- `scale_*_date(date_labels = "%m/%d", date_breaks = "2 weeks")`
  - treat data values as dates.
- `scale_*_datetime()`
  - treat data `x` values as date times.
  - Use same arguments as scale_x_date().
  - See `?strptime` for label formats.

Dataset *gapminder*

- Excerpt of the Gapminder data on
- life expectancy, GDP per capita, and population by country.

```
set.seed(123) # fix random seed
library(gapminder)
gapminder_2007 <- gapminder %>%
  filter(year==2007) %>%
  select(year, country, continent, gdpPercap, lifeExp, pop)
gapminder_2007_table <- gapminder_2007 %>% sample_n(3)
```

Rmarkdown tabular – a momentary digression

```
gapminder_2007_table
```

```
## # A tibble: 3 x 6
##    year country           continent gdpPercap lifeExp      pop
##   <int> <fct>             <fct>         <dbl>   <dbl>    <int>
## 1  2007 Equatorial Guinea Africa       12154.    51.6   551201
## 2  2007 Serbia            Europe        9787.    74.0 10150265
## 3  2007 Iceland           Europe       36181.    81.8   301931
```

kable() for quick tabular

```
library(kableExtra)
kable(gapminder_2007_table)
```

year	country	continent	gdpPercap	lifeExp	pop
2007	Equatorial Guinea	Africa	12154.090	51.579	551201
2007	Serbia	Europe	9786.535	74.002	10150265
2007	Iceland	Europe	36180.789	81.757	301931

xtable() for generating latex source

```
library(xtable)
xtable(gapminder_2007_table)
```

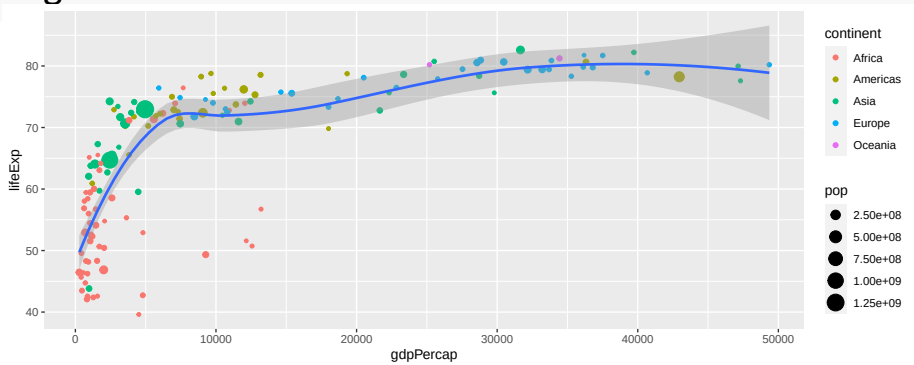
```
## % latex table generated in R 3.5.3 by xtable 1.8-4 package
## % Sat Apr 11 13:44:57 2020
## \begin{table}[ht]
## \centering
## \begin{tabular}{rrllrrr}
##   \hline
##  & year & country & continent & gdpPercap & lifeExp & pop \\
##   \hline
## 1 & 2007 & Equatorial Guinea & Africa & 12154.09 & 51.58 & 551201 \\
##   2 & 2007 & Serbia & Europe & 9786.53 & 74.00 & 10150265 \\
##   3 & 2007 & Iceland & Europe & 36180.79 & 81.76 & 301931 \\
##    \hline
## \end{tabular}
## \end{table}
```

	year	country	continent	gdpPercap	lifeExp	pop
1	2007	Equatorial Guinea	Africa	12154.09	51.58	551201
2	2007	Serbia	Europe	9786.53	74.00	10150265
3	2007	Iceland	Europe	36180.79	81.76	301931

X and Y location scales (use with *x* or *y* aesthetics)

- `scale_x_log10()`: Plot `x` on log10 scale
- `scale_x_reverse()`: Reverse direction of `x` axis
- `scale_x_sqrt()`: Plot `x` on square root scale

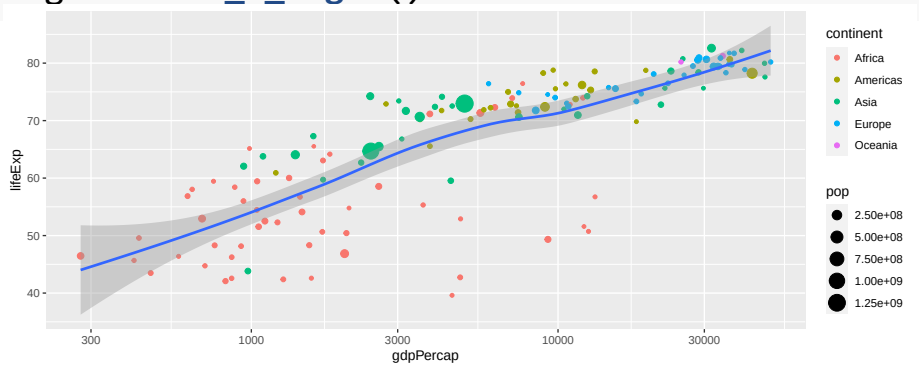
```
fig <- gapminder_2007 %>%
  ggplot(aes(x=gdpPercap, y=lifeExp)) +
  geom_point(aes(size=pop, color=continent)) +
  geom_smooth()
fig
```



`scale_x_log10()`

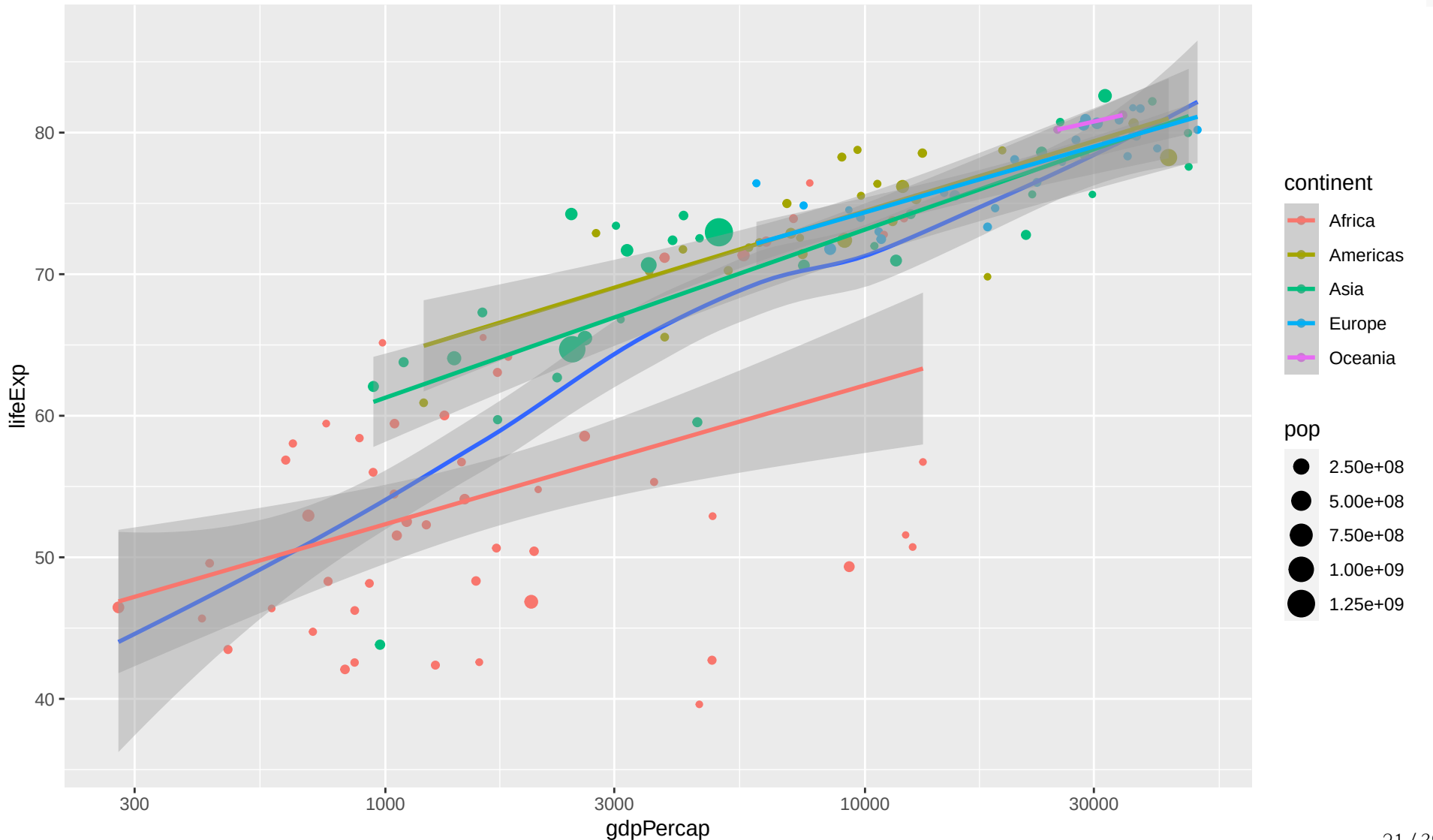
- `x` 축을 `log(x)`로 변환하여
- exponential growth를 visualize 한다.

```
fig + scale_x_log10()
```

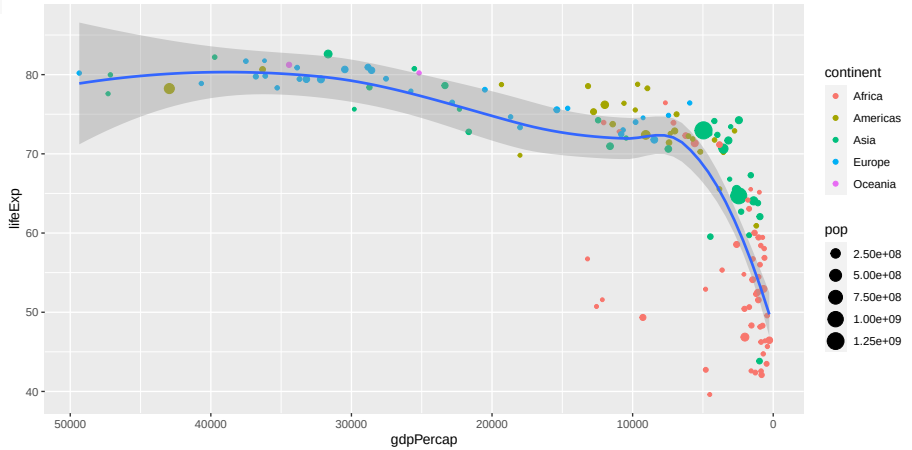


오래 살고 싶으면 어떻게 해야하는가??

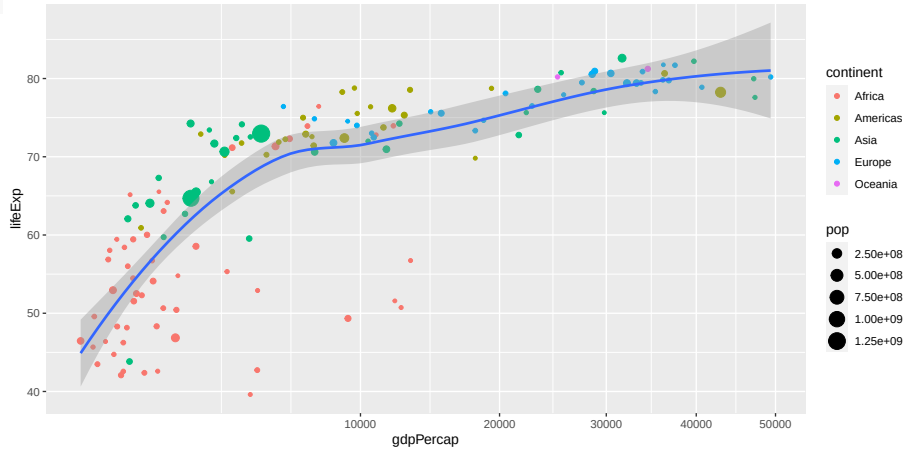
```
fig +
  scale_x_log10() +
  geom_smooth(aes(group=continent, color=continent), method="lm")
```



```
fig + scale_x_reverse()
```

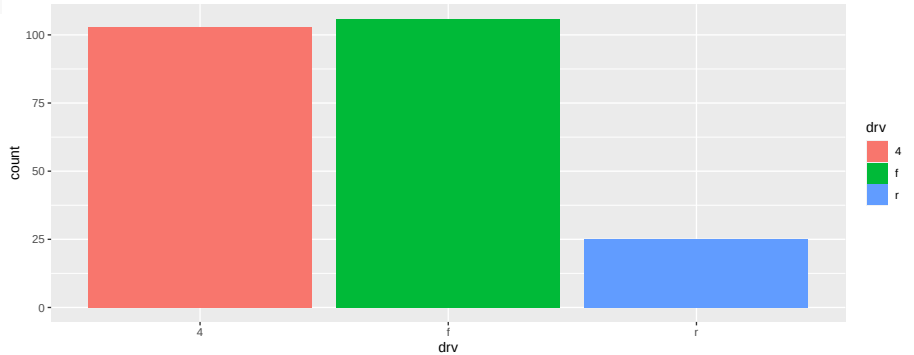



```
fig + scale_x_sqrt()
```



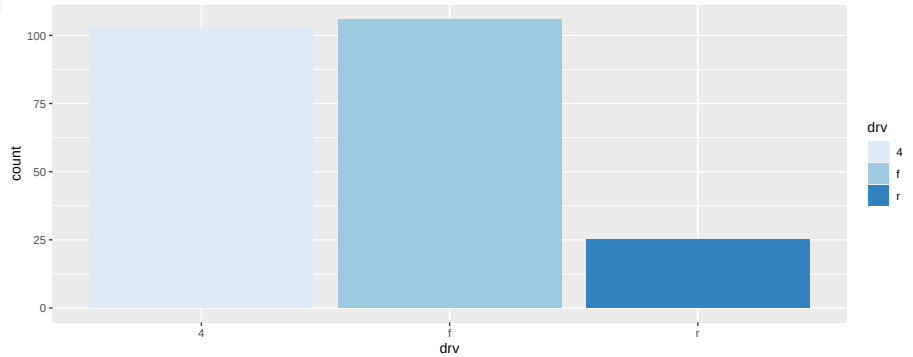
Color and fill scales (Discrete)

```
ggplot(mpg, aes(x=drv, fill=drv)) + geom_bar()
```



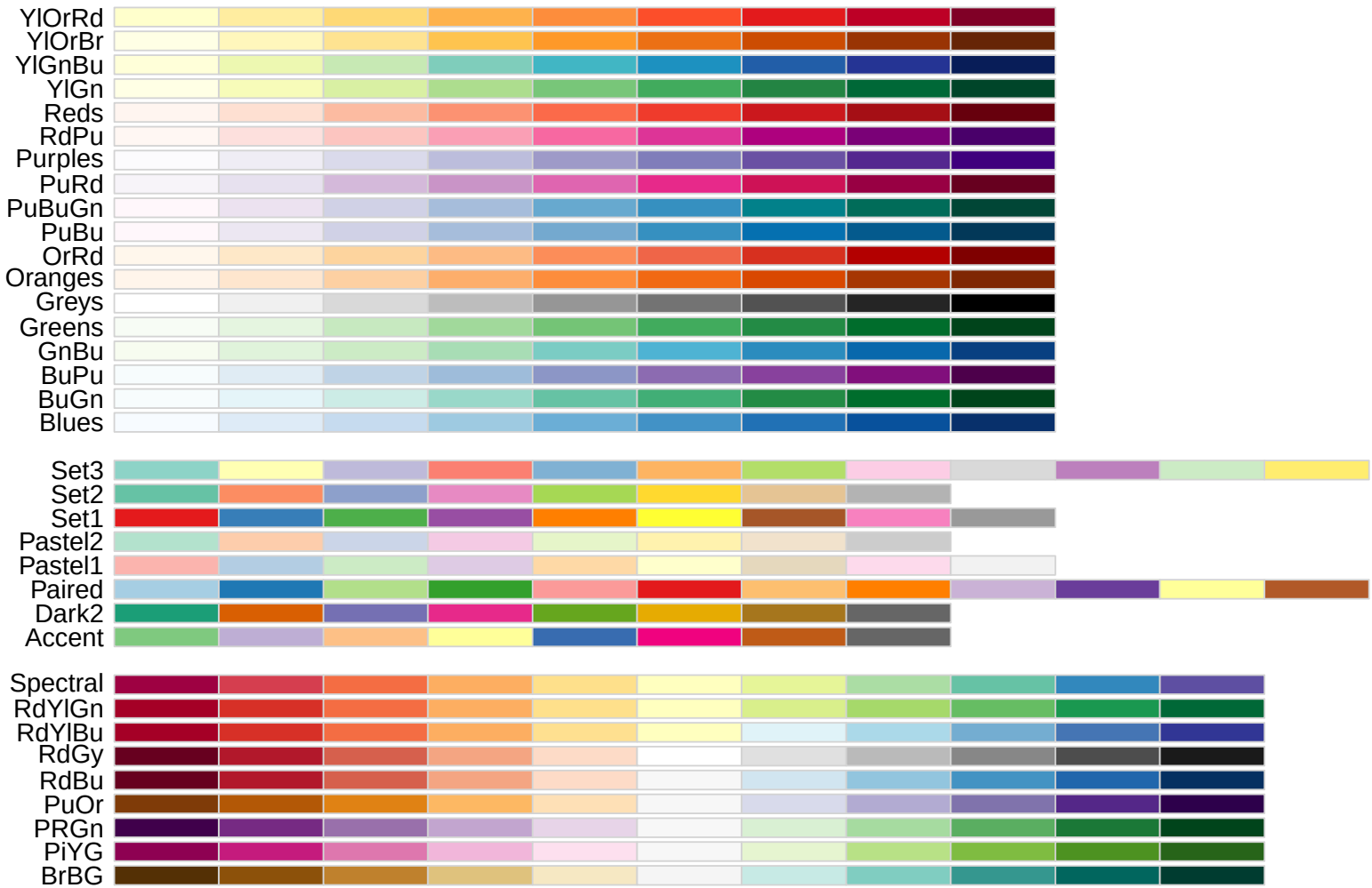
scale_fill_brewer()

```
library(RColorBrewer)
ggplot(mpg, aes(x=drv, fill=drv)) + geom_bar() +
  scale_fill_brewer(palette = "Blues")
```

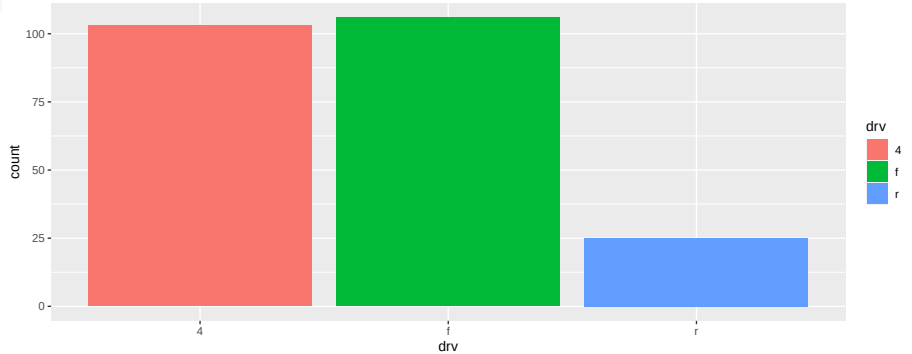


For more palette choices.

```
RColorBrewer::display.brewer.all()
```

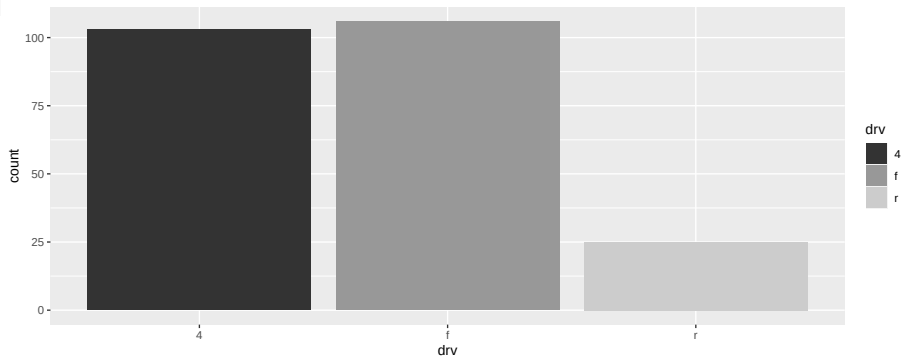


```
ggplot(mpg, aes(x=drv, fill=drv)) + geom_bar()
```



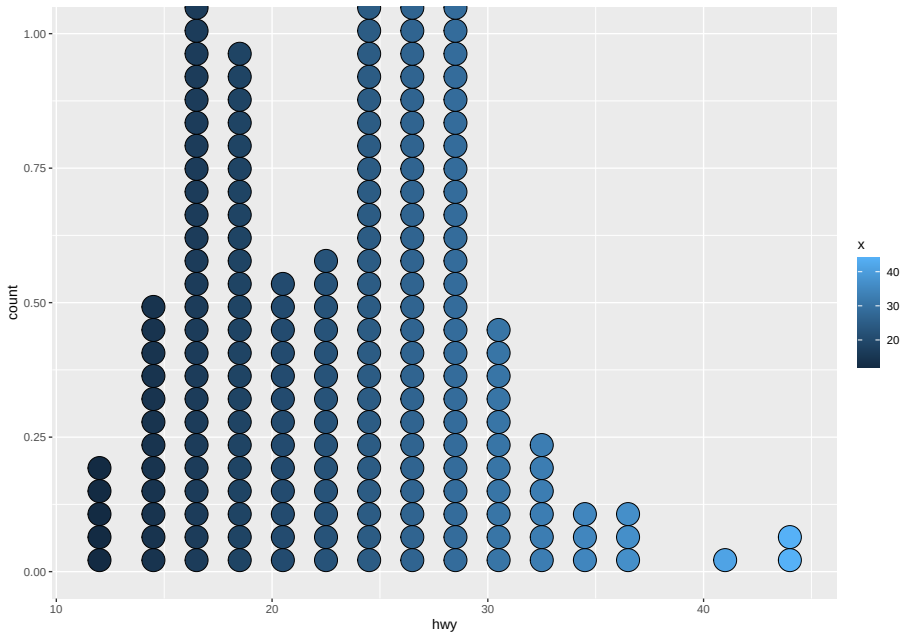
scale_fill_grey()

```
ggplot(mpg, aes(x=drv, fill=drv)) + geom_bar() +
  scale_fill_grey(
    start = 0.2,
    end = 0.8,
    na.value= "red")
```



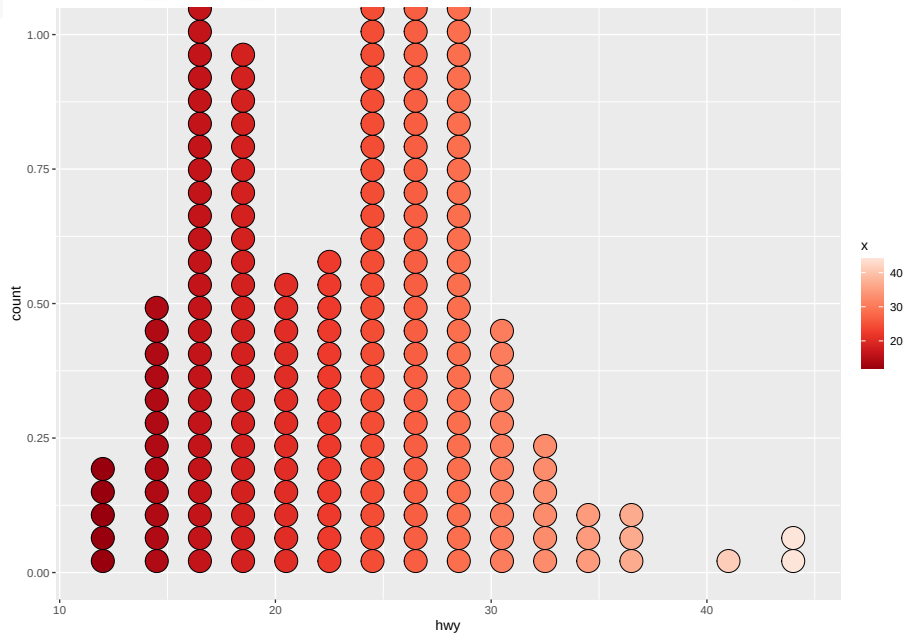
Color and fill scales (Continuous)

```
fig <- ggplot(mpg, aes(x=hwy, fill=..x..)) +
  geom_dotplot()
fig
```



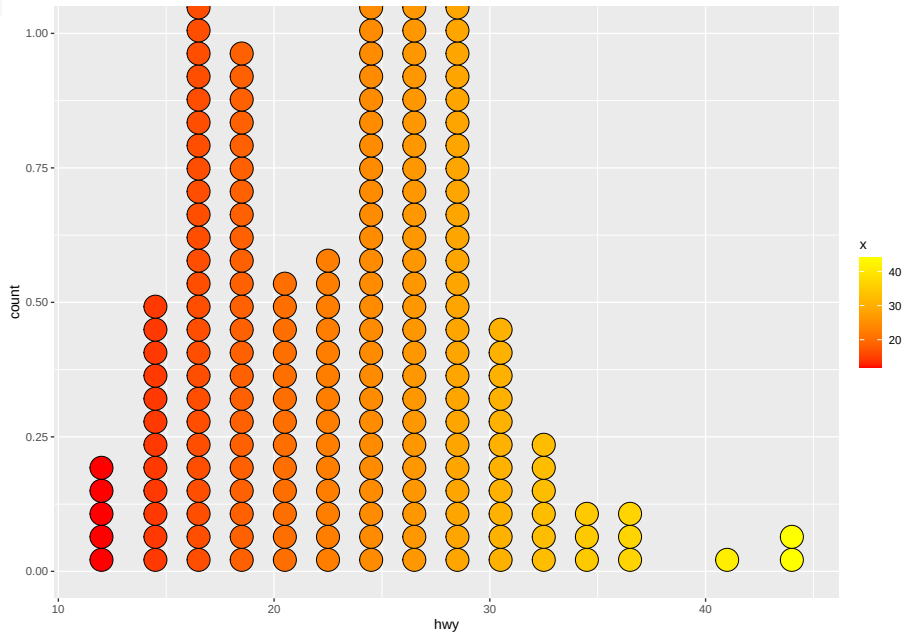
scale_fill_distiller()

```
fig +
  scale_fill_distiller(palette = "Reds")
```



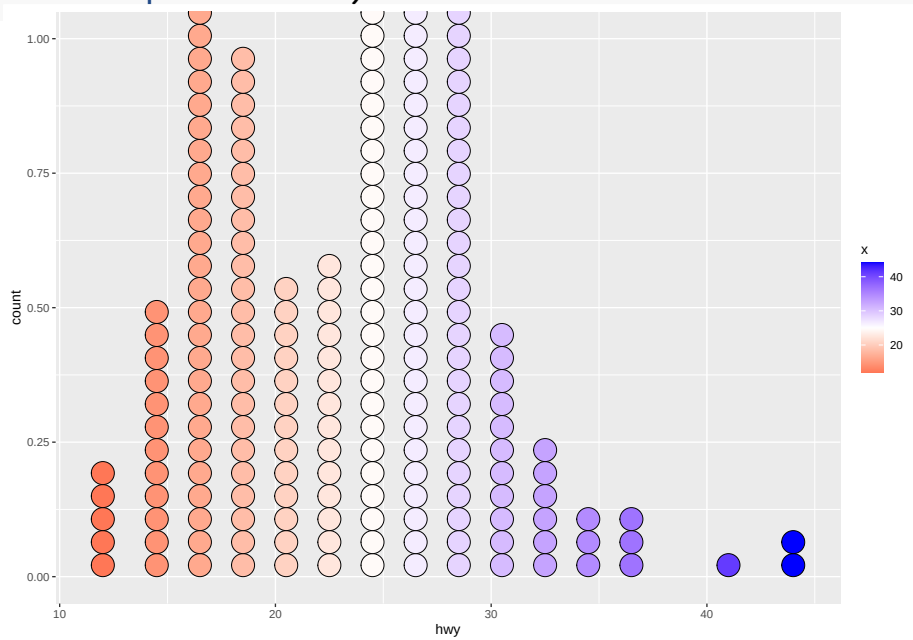
scale_fill_gradient()

```
fig +
  scale_fill_gradient(
    low="red",
    high="yellow")
```



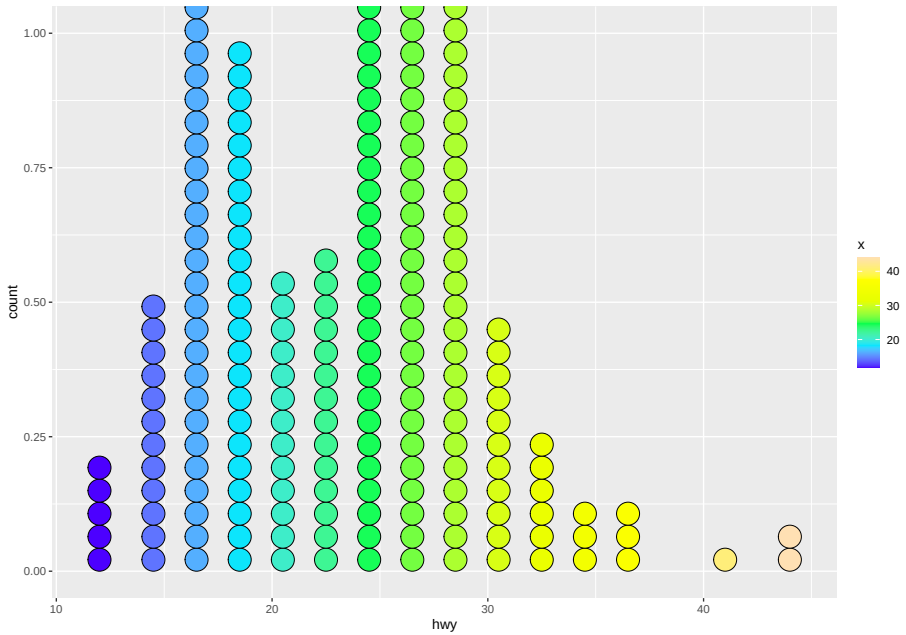
scale_fill_gradient2()

```
fig +
  scale_fill_gradient2(
    low="red",
    high="blue",
    mid = "white",
    midpoint = 25)
```



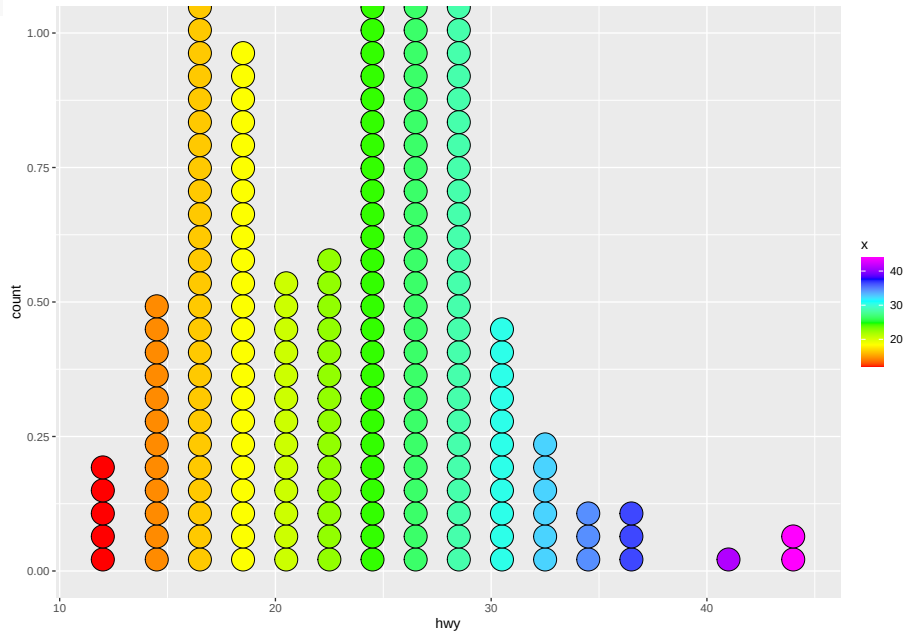
scale_fill_gradientn()

```
fig + scale_fill_gradientn(
  colours=topo.colors(6))
```



- SEE Also:
  - heat.colors()
  - terrain.colors()
  - cm.colors()
  - RColorBrewer::brewer.pal()

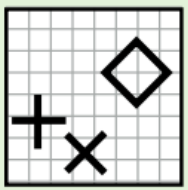
```
fig + scale_fill_gradientn(
  colours=rainbow(6))
```



Shape and size scales

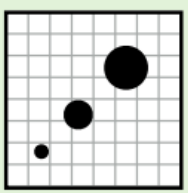
Shape and size scales

```
p <- e + geom_point(aes(shape = fl, size = cyl))
```



```
p + scale_shape() + scale_size()
p + scale_shape_manual(values = c(3:7))
```

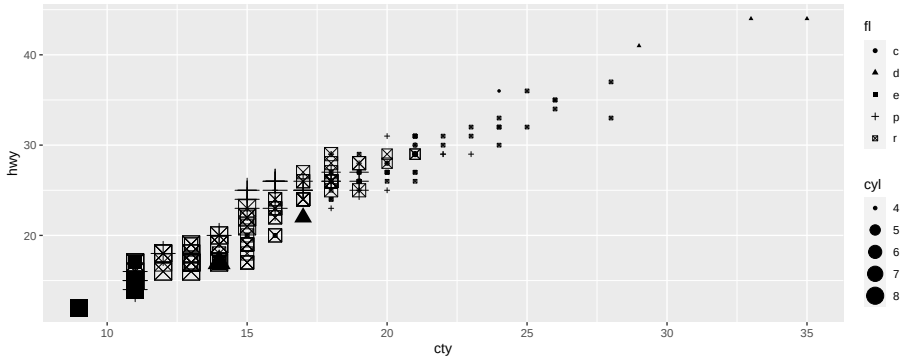
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25



```
p + scale_radius(range = c(1,6))
p + scale_size_area(max_size = 6)
```

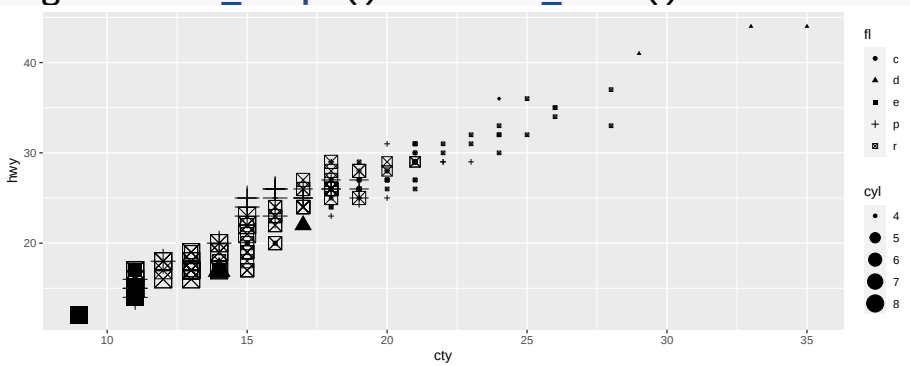
Maps to radius of circle, or area

```
fig <- ggplot(mpg, aes(cty, hwy)) +
  geom_point(aes(shape = fl, size = cyl))
fig
```



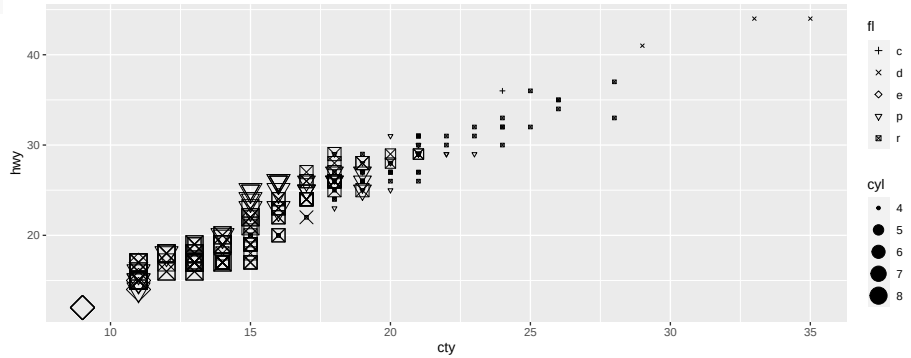
`scale_shape()` + `scale_size()` are implicitly applied.

```
fig + scale_shape() + scale_size()
```



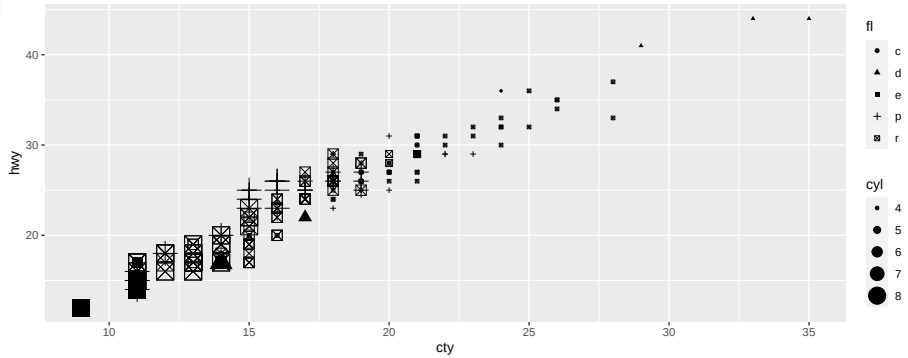
`scale_shape()` overriden

```
fig + scale_shape_manual(values = c(3:7))
```

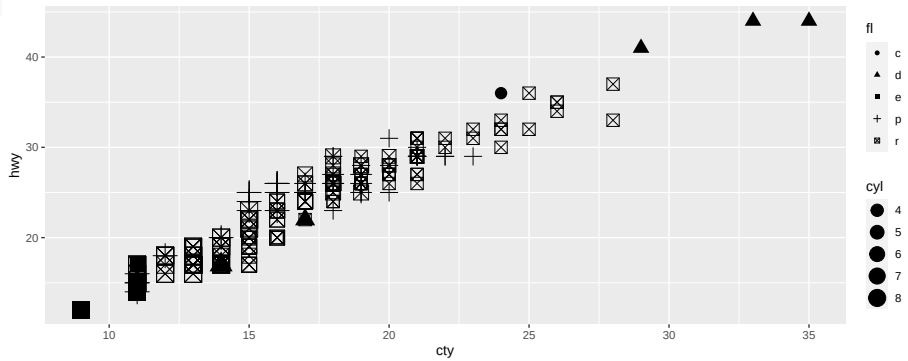


`scale_size()` overriden

```
fig + scale_radius(range = c(1,6))
```




```
fig + scale_size_area(max_size = 6)
```



Section 4

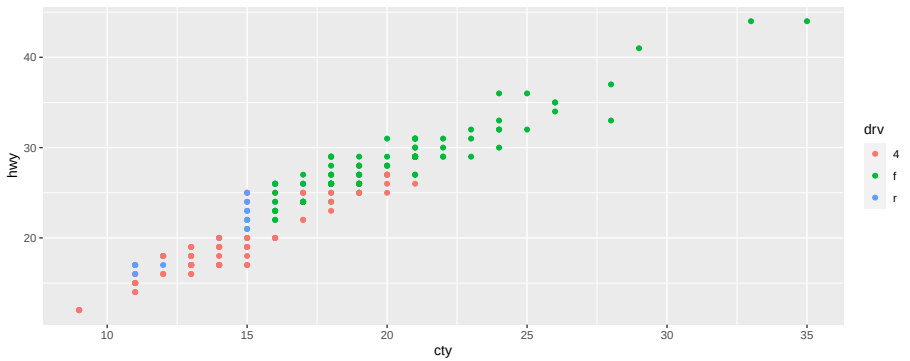
IV. Theme

Theme

- A powerful way to customize the non−data components of your plots
- i.e. titles, labels, fonts, background, gridlines, and legends.
- Allows plots to have a consistent customized look.

theme_STYLE() – Broadly applied

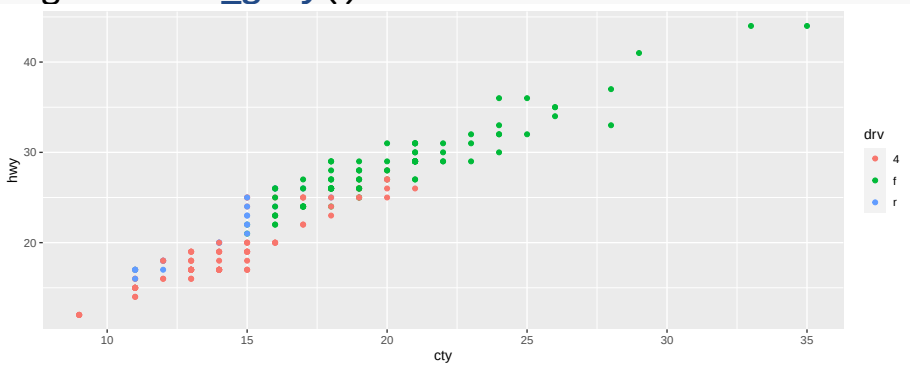
```
fig <- ggplot(mpg, aes(cty,hwy)) +
  geom_point(aes(color=drv))
fig
```



theme_gray()

- Grey background (default theme)

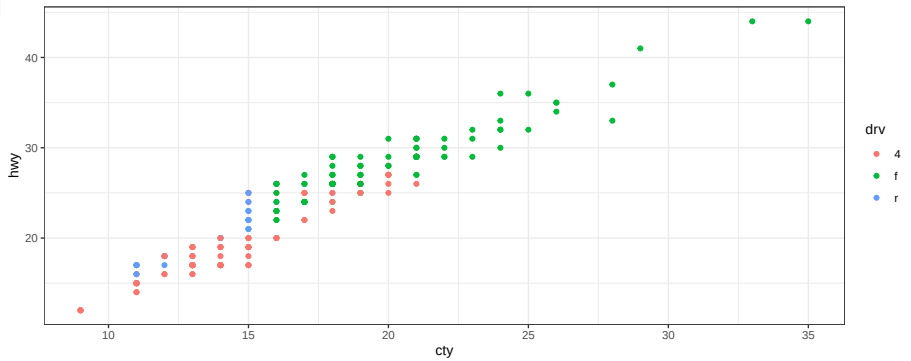
```
fig + theme_gray()
```



theme_bw()

- White background with grid lines

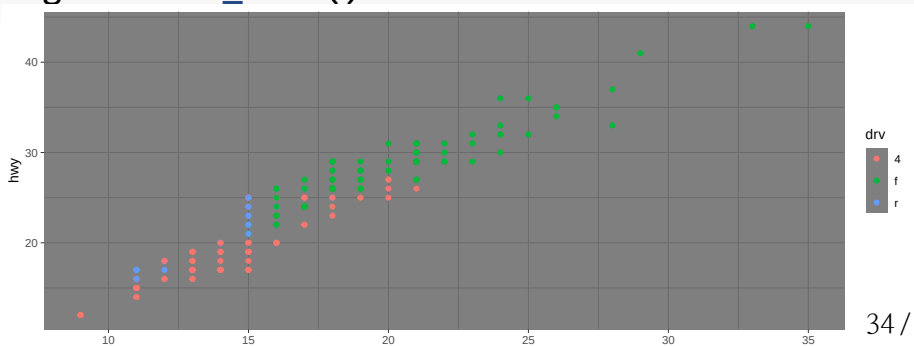
```
fig + theme_bw()
```



theme_dark()

- Dark for contrast

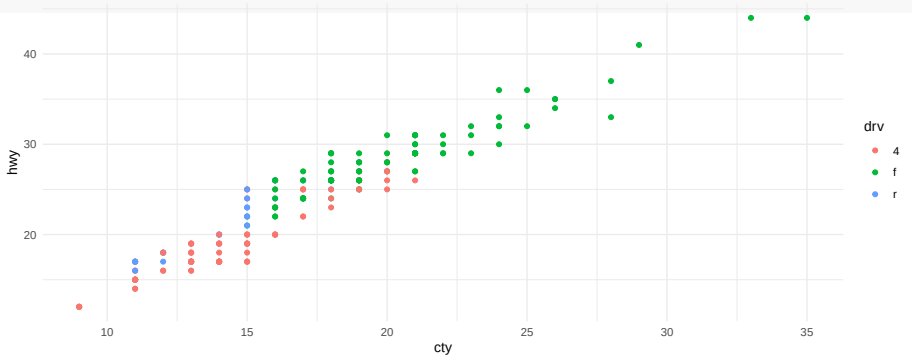
```
fig + theme_dark()
```



theme_minimal()

- Minimal themes

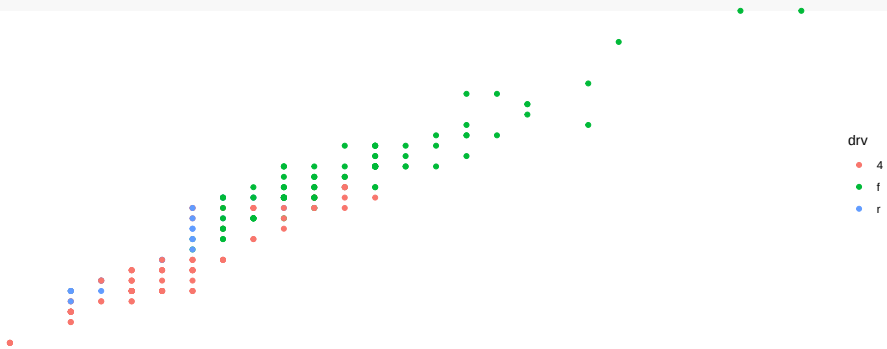
```
fig + theme_minimal()
```



theme_void()

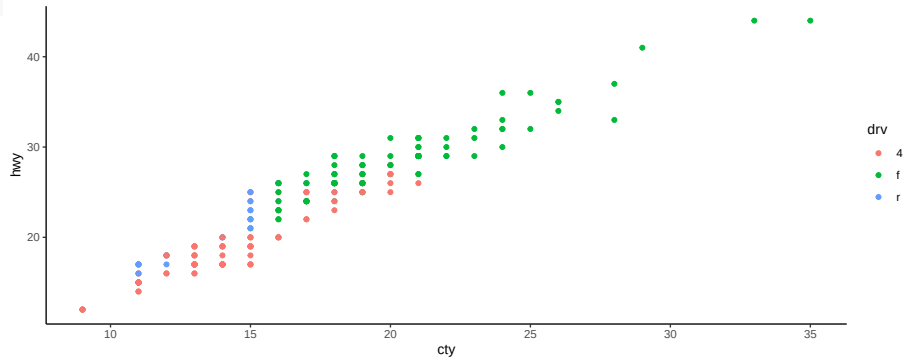
- Empty theme

```
fig + theme_void()
```



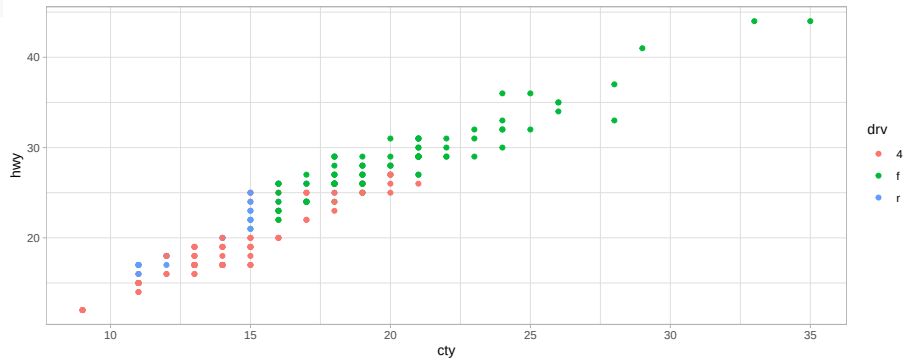
theme_classic()

```
fig + theme_classic()
```

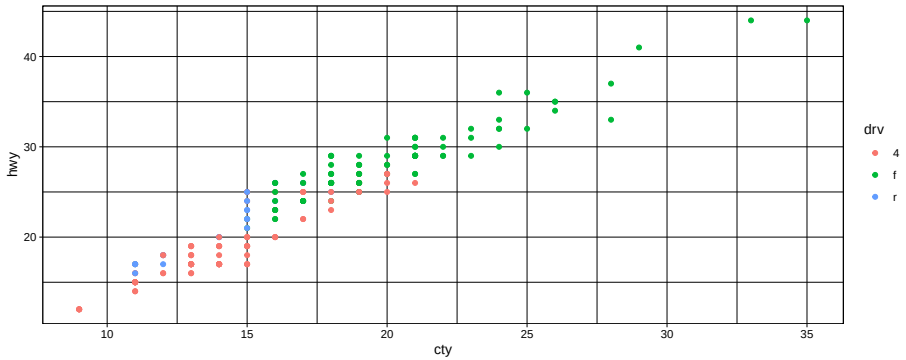


theme_light()

```
fig + theme_light()
```



```
fig + theme_linedraw()
```



Inheritance

- Theme elements inherit properties from other theme elements heirarchically.
  - For example, `axis.title.x.bottom` inherits from `axis.title.x` which inherits from `axis.title`, which in turn inherits from `text`.
  - All text elements inherit directly or indirectly from text;
  - all lines inherit from line, and
  - all rectangular objects inherit from rect.
- This means that you can modify the appearance of multiple elements by setting a single high−level component.

Arguments

https://github.com/aceMKSim/teaching/blob/master/Data%20Visualization/MISC/theme.md

```
"Tantum videmus quantum scimus."
```

```
## [1] "Tantum videmus quantum scimus."
```