

## R11. ggplot (5)

label, annotate, legend, scale, theme, and table

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



서울과학기술대학교 데이터사이언스학과

# About

Complete the template below to build a graph.

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

required

Not required, sensible defaults supplied

- Rmd 문서에 데이터프레임을 삽입하는 방법을 다룹니다.
  - Part 1. Table presentation in Rmd
- ggplot의 vis를 다듬기 위하여 아래의 5가지 기능을 다룹니다.
  - Part 2. Label
  - Part 3. Annotate
  - Part 4. Legend
  - Part 5. Scale
  - Part 6. Theme

## Part 1. Table presentation

## Dataset *gapminder*

- Gapminder라는 데이터셋은 국가별 경제 수준과 의료 수준 동향을 정리한 데이터를 담고 있다.
- R의 *gapminder* library에서는 아래 변수들을 제공한다.
  - life expectancy
  - GDP per capita
  - population by country

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

## Presentation 1 - no additional styling

```
gapminder_2007_table
```

```
## # A tibble: 3 x 6
```

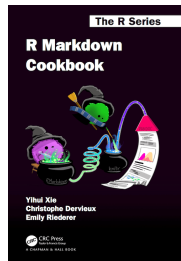
```
##   year country    continent gdpPercap lifeExp      pop
##   <int> <fct>      <fct>      <dbl>   <dbl>   <int>
## 1  2007 Botswana    Africa      12570.    50.7  1639131
## 2  2007 Greece     Europe      27538.    79.5  10706290
## 3  2007 South Africa Africa       9270.    49.3  43997828
```

## Presentation 2 - KableExtra::kable()

```
kableExtra::kable(gapminder_2007_table)
```

year	country	continent	gdpPercap	lifeExp	pop
2007	Botswana	Africa	12569.852	50.728	1639131
2007	Greece	Europe	27538.412	79.483	10706290
2007	South Africa	Africa	9269.658	49.339	43997828

- kable()에 대한 설명 페이지  
<https://bookdown.org/yihui/rmarkdown-cookbook/kable.html>
- 위의 사이트는 Yihui Xie가 직접 저술한 R markdown을 다루고 있는 단행본이며, 무료로 접근이 가능함



## Presentation 3 - `xtable::xtable()`

- 테이블은  $\text{\LaTeX}$  포맷으로 변환하여 제공

```
xtable::xtable(gapminder_2007_table)
```

```
## % latex table generated in R 4.1.0 by xtable 1.8-4 package
## % Sat Nov 12 11:52:18 2022
## \begin{table}[ht]
## \centering
## \begin{tabular}{rrllrrr}
## \hline
## & year & country & continent & gdpPercap & lifeExp & pop \\\
## \hline
## 1 & 2007 & Botswana & Africa & 12569.85 & 50.73 & 1639131 \\\
## 2 & 2007 & Greece & Europe & 27538.41 & 79.48 & 10706290 \\\
## 3 & 2007 & South Africa & Africa & 9269.66 & 49.34 & 43997828 \\\
## \hline
## \end{tabular}
## \end{table}
```



## Part 2. Label

# You must title, labels, and legends

- Make vis as **self-documenting** as possible
- 의미있고 유용한 title, labels, legends를 부여해야 함

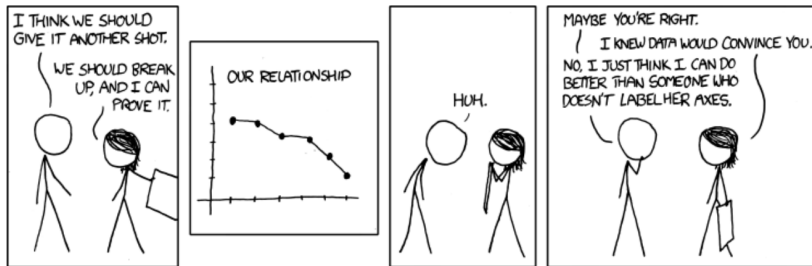
## Label

- Axx와 panes/subwindows들은 모두 label을 가져야 함
- Axx는 적절한 min/max 범위와 tick marks를 가져야 함

## Legend

- 그려지는 모든 것은 legend를 가져야함
- 모든 legend는 header/labels를 가져야함 (if not redundant with main title)
- 이해하기 쉬운 numerical format을 가져야함 (scientific notation을 가급적이면 피해야 함)

## You must title, labels, and legends



[<https://xkcd.com/833/>]

- `labs()` 함수를 이용해서 그래픽을 설명하는 문구를 넣음
  - `title`: 그래프의 주제를 담음
  - `subtitle`: `title`보다 작은 폰트로 `title`에 비해 자세한 내용을 담음
  - `caption`: 그래프 오른쪽아래에 주로 데이터 출처를 명시함
  - `x`, `y`: `axis`에 대한 설명을 넣음
  - `color`, `size`, ...: `aesthetic`에 대해서 생성되는 `legend`에 설명을 넣음

```
fig1 <- ggplot(mpg, aes(cty, hwy, color=drv)) + geom_point() +  
  labs(title = "Efficiency: city vs highway",  
        subtitle = "Highly correlated. fwd is generally more efficient at high way",  
        caption = "Source: R dataset mpg",  
        x = "city road efficiency",  
        y = "high way road efficiency",  
        color = "Driving wheel") +  
  annotate(geom = "label", x = 15, y = 22, label = "Our product")
```

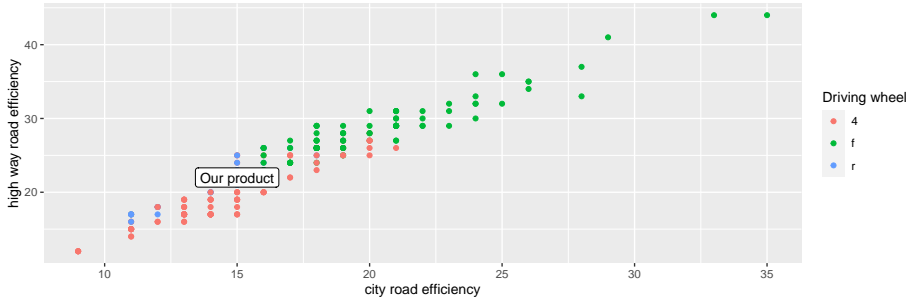
- `annotate()` 함수를 이용해서 임의의 `geom`을 넣을 수 있음.

```
fig1 <- ggplot(mpg, aes(cty, hwy, color=drv)) + geom_point() +  
  labs(title = "Efficiency: city vs highway",  
        subtitle = "Highly correlated. fwd is generally more efficient at high way",  
        caption = "Source: R dataset mpg",  
        x = "city road efficiency",  
        y = "high way road efficiency",  
        color = "Driving wheel") +  
  annotate(geom = "label", x = 15, y = 22, label = "Our product")
```

fig1

Efficiency: city vs highway

Highly correlated. fwd is generally more efficient at high way



Source: R dataset mpg

## Part 3. Annotation

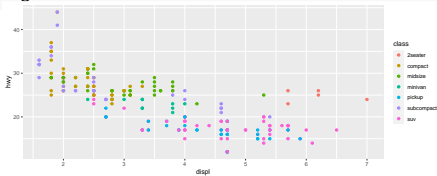
## Annotation

- 그래프의 major component외에도 individual component나 groups of observation에 대해서 label을 추가하는 것은 때로는 매우 유용하다.
- `geom_text()`와 `geom_label()`
  - `geom_point()`와 매우 비슷한 문법을 가지고 있으며
  - Annotation에 매우 유용한 geom이다.

## Case 1 - 각 class의 best 연비를 annotation

1. 아래 그래프에 각 class 가장 연비가 좋은 차의 이름을 표시해보자.

```
fig2 <- ggplot(mpg, aes(displ, hwy)) +
  geom_point(aes(color=class))
fig2
```



2. 우선 각 class별로 가장 연비가 좋은 차를 명시한 데이터셋을 만든다.

```
best_in_class <- mpg %>% group_by(class) %>%
  filter(row_number(desc(hwy)) == 1) %>%
  select(class, model, displ, hwy)
best_in_class
```

```
## # A tibble: 7 x 4
```

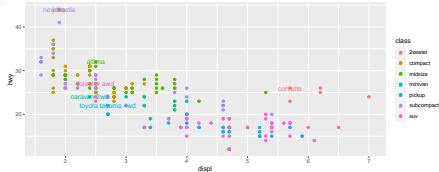
```
## # Groups:   class [7]
```

##	class	model	displ	hwy
##	<chr>	<chr>	<dbl>	<int>
## 1	2seater	corvette	5.7	26
## 2	minivan	caravan 2wd	2.4	24
## 3	midsize	altima	2.5	32
## 4	suv	forester awd	2.5	27
## 5	pickup	toyota tacoma 4wd	2.7	22
## 6	compact	jetta	1.9	44
## 7	subcompact	new beetle	1.9	44



### 3. geom\_text() 함수를 추가한다.

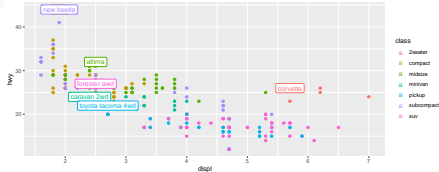
```
fig2 + geom_text(
  data = best_in_class,
  mapping = aes(label=model, color=class))
```



- Issue: Text와 point의 간섭이 있다.

### 4. geom\_label()을 사용한다.

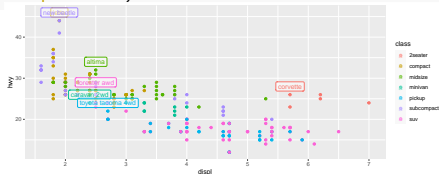
```
fig2 + geom_label(
  data = best_in_class,
  mapping = aes(label=model, color=class))
```



- Issue: 포지션이 겹치면서 point가 사라진다.

## 5. geom\_label()의 위치를 약간 바꾸고, 투명하게 만든다.

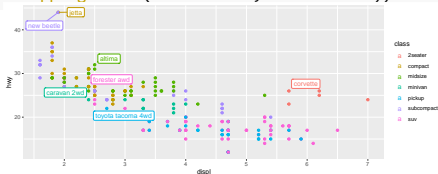
```
fig2 + geom_label(
  data = best_in_class,
  mapping = aes(label=model, color=class),
  nudge_y = 2,
  alpha = 0.5)
```



- Issue: 왼쪽 상단에 new beetle과 jetta가 겹쳐있다.

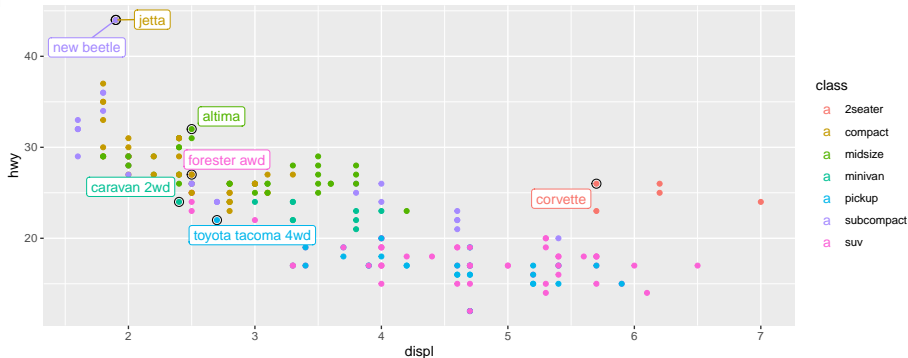
## 6. ggrepel::geom\_label\_repel() 함수를 사용하면 자동으로 overlap이 해결된다.

```
fig2 + ggrepel::geom_label_repel(
  data = best_in_class,
  mapping = aes(label=model, color=class))
```



## 7. Finally, with the circle for the annotated points.

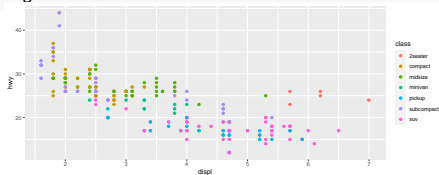
```
ggplot(mpg, aes(displ, hwy)) +  
  geom_point(aes(color=class)) +  
  geom_point(data = best_in_class, size = 3, shape = 1) +  
  ggrepel::geom_label_repel(  
    data = best_in_class,  
    mapping = aes(label=model, color=class))
```



## Case 2 - annotation으로 legend 대체

1. 아래 그래프에서 legend를 없애고  
그래프 위에 직접 표시해보자.

```
fig2 <- ggplot(mpg, aes(displ, hwy)) +  
  geom_point(aes(color=class))  
fig2
```



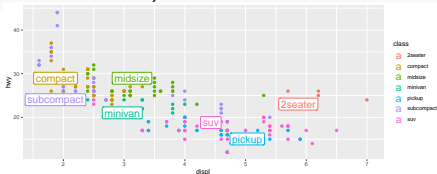
2. Annotate할 포지션을 정할 수 있게  
아래의 데이터셋을 만든다.

```
class_avg <- mpg %>% group_by(class) %>%  
  summarise(displ = median(displ),  
            hwy = median(hwy))  
class_avg
```

```
## # A tibble: 7 x 3  
##   class      displ  hwy  
##   <chr>    <dbl> <dbl>  
## 1 2seater     6.2   25  
## 2 compact     2.2   27  
## 3 midsize     2.8   27  
## 4 minivan     3.3   23  
## 5 pickup     4.7   17  
## 6 subcompact  2.2   26  
## 7 suv        4.65  17.5
```

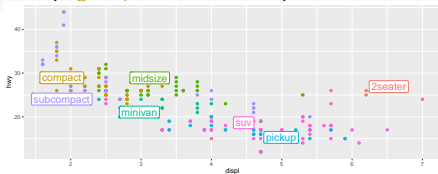
### 3. geom\_label\_repel() 함수 사용

```
fig2 + ggrepel::geom_label_repel(
  data = class_avg,
  mapping = aes(label=class, color=class),
  size = 6,
  label.size = 0)
```



4. 기존 legend를 제거. Legend를 제거하면 그래프에 더 집중할 수 있다.

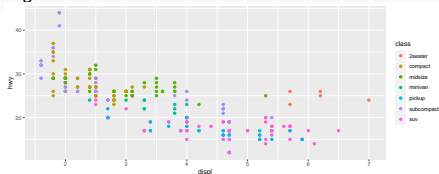
```
fig2 + ggrepel::geom_label_repel(
  data = class_avg,
  mapping = aes(label=class, color=class),
  size = 6,
  label.size = 0) +
theme(legend.position = "none")
```



## Case 3 - Single label을 추가하여 데이터 묘사

1. 아래 그래프의 우상단에 데이터를 묘사하는 문장을 삽입한다.

fig2



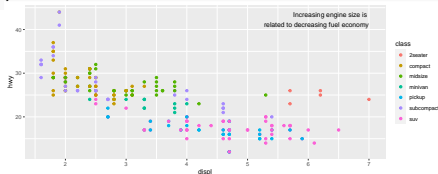
2. label이라는 데이터셋에 포지션과 문장 내용을 입력한다.

```
label <- mpg %>% summarise(
  displ = max(displ),
  hwy = max(hwy),
  label = paste(
    "Increasing engine size is \nrelated to",
    "decreasing fuel economy"
  )
)
label

## # A tibble: 1 x 3
##   displ hwy label
##   <dbl> <int> <chr>
## 1     7   44 "Increasing engine size is \nrelated to"
```

### 3. geom\_text() 사용

```
fig2 + geom_text(
  data = label,
  mapping = aes(label=label),
  vjust = "top",
  hjust = "right"
)
```



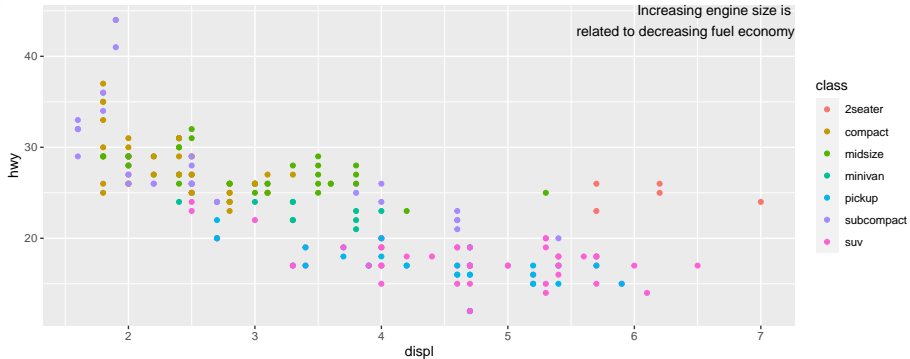
4. Alternatively, 문장을 정확히 우상단 구석에 위치시키려면 position에 강제로 +Inf를 입력할 수 있다. 이 경우에는 아래와 같이 label 데이터셋을 준비한다.

```
label <- tibble(
  displ = Inf,
  hwy = Inf,
  label = paste(
    "Increasing engine size is \nrelated to",
    "decreasing fuel economy"
  )
)
label

## # A tibble: 1 x 3
##   displ hwy label
##   <dbl> <dbl> <chr>
## 1   Inf   Inf "Increasing engine size is \nrelated to"
```

## 5. Text가 확실하게 우상단 코너에 붙는다.

```
fig2 + geom_text(  
  data = label,  
  mapping = aes(label=label),  
  vjust = "top",  
  hjust = "right"  
)
```





- `geom_text()`와 `geom_label()` 외에도 아래의 기능들을 활용할 수 있다.
- `geom_hline()`과 `geom_vline()`은 각각 horizontal과 vertical의 보조선을 그린다.
- `geom_abline()`은 직선의 기울기와 y절편을 지정하여 보조선을 긋는다.
- `geom_rect()`은 4개의 점을 지정해서 직사각형을 그린다.
- `geom_segment()`는 화살표를 그린다.

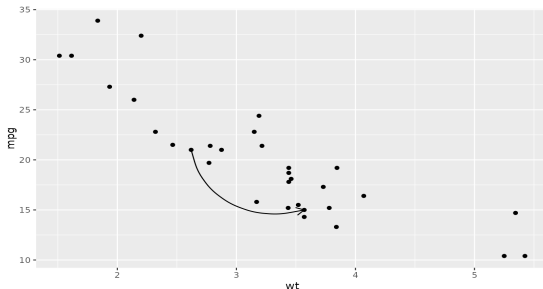


그림 1: Example of geom-segment

## Part 4. Legend

`theme(legend.position=)`을 사용하여 legend의 위치를 바꿀 수 있다.

```
# Choose among c("top", "bottom", "left", "right", "none")
```

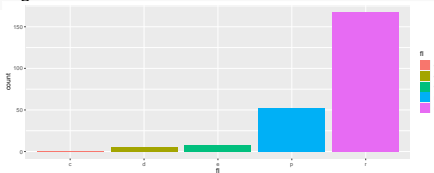
```
fig1 + theme(legend.position = "top")
```



Dataset의 내용을 바꾸지 않고도 legend label을 바꿀 수 있다.

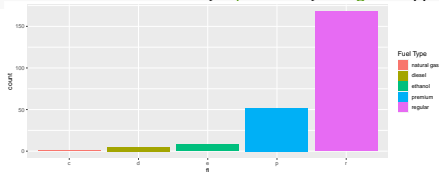
1. Legend의 c, d, e, p, r의 값을 알기 어렵다.

```
fig3 <-  
  ggplot(mpg, aes(x=f1, fill = f1)) + geom_bar()  
fig3
```



2. fill에 해당하는 legend이며, 변수의 특징이 discrete 하므로 scale\_fill\_discrete() 함수를 사용한다.

```
fig3 + scale_fill_discrete(  
  name = "Fuel Type",  
  labels = c("natural gas", "diesel",  
             "ethanol", "premium", "regular"))
```




## Part 5. Scale

# About

- Scale은 data의 값과 aesthetic의 대응에 상세한 configuration을 넣는다.

## From cheatsheet

**Scales** map data values to the visual values of an aesthetic. To change a mapping, add a new scale.



```
(n <- d + geom_bar(aes(fill = fl)))
```

scale\_    aesthetic to adjust    prepackaged scale to use    scale specific arguments

```
n + scale_fill_manual(
  values = c("skyblue", "royalblue", "blue", "navy"),
  limits = c("d", "e", "p", "r"), breaks = c("d", "e", "p", "r"),
  name = "fuel", labels = c("D", "E", "P", "R"))
```

range of values to include in mapping    title to use in legend/axis    labels to use in legend/axis    breaks to use in legend/axis

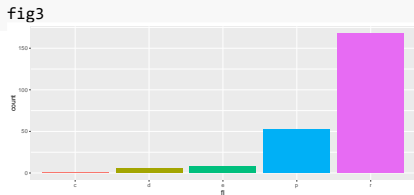
## *scale\_\*\_\**() 함수들

- `scale_*_continuous()`
- `scale_*_discrete()`
- `scale_*_identity()`: 데이터 값 자체를 이용한다.
- `scale_*_manual(values = c())`: 수동적으로 조작한다.
- `scale_*_date(date_labels = "%m/%d", date_breaks = "2 weeks")`: 날짜를 다룬다.
- `scale_*_datetime()`
  - treat data x values as date times.
  - Use same arguments as `scale_x_date()`.
  - See `?strptime` for label formats.

# 1. 표현 범위 지정

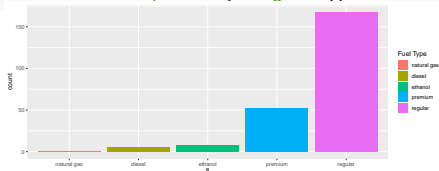
Case 1 - 앞의 그래프에서 x축을 상세하게 설명하고자 한다.

## 1. Original chart



## 2. 변수값을 명확하게 설명

```
fig3 +
  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"))
```

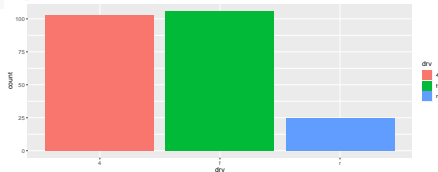




## Case 2 - $f$ 와 $r$ 만 표시되게 차트를 바꾸고 싶다.

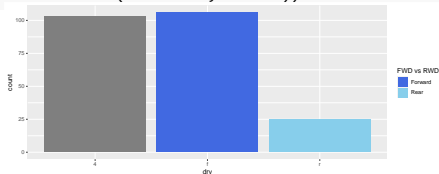
### 1. Original figure

```
fig4 <- ggplot(mpg, aes(x=drv, fill=drv)) +  
  geom_bar()  
fig4
```



### 2. scale\_fill\_manual()을 사용한다.

```
fig4 + 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"))
```

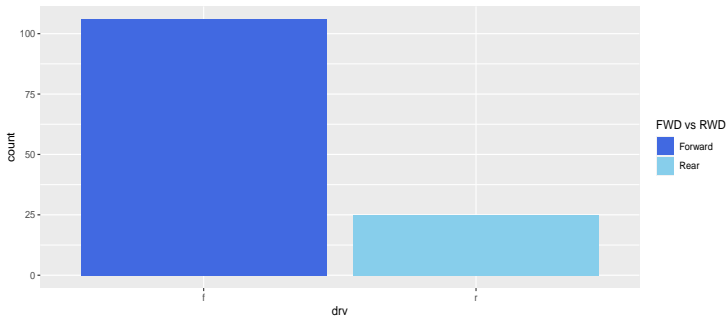


#### ● Issue

- fill에 관련해서는 2개의 값에 대해서만 지정되었지만
- x에 관련해서는 3개의 모든 수치가 표시되었다.

### 3. scale\_x\_discrete()을 사용한다.

```
fig4 + 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")) +  
scale_x_discrete(limits = c("f", "r"))
```

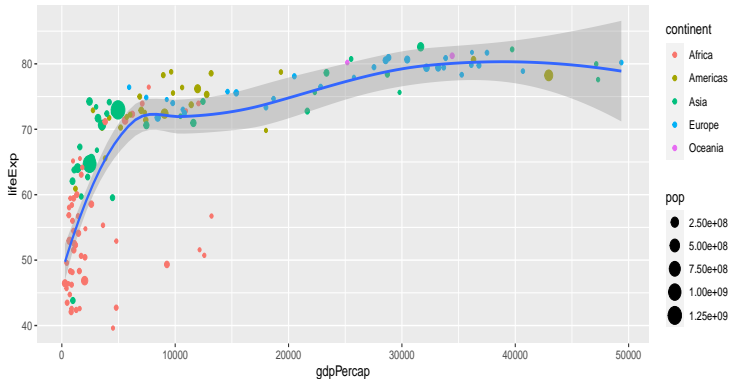


## 2. *scale* for X and Y location

- x 혹은 y aesthetic과 함께 사용
- `scale_x_log10()`: x축을 log10 scale로 변환한다.
- `scale_x_reverse()`: x 축의 방향을 뒤집는다.
- `scale_x_sqrt()`: x축을 square root scale로 변환한다.

## Original figure

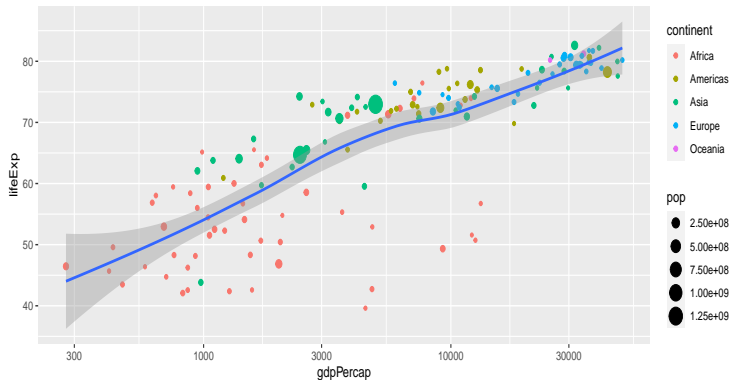
```
fig <- gapminder_2007 %>%  
  ggplot(aes(x=gdpPerCap, y=lifeExp)) +  
  geom_point(aes(size=pop, color=continent)) +  
  geom_smooth()  
fig
```



## `scale_x_log10()`

- x 축을  $\log(x)$  로 변환하여
- concave growth를 linear하게 표현한다.

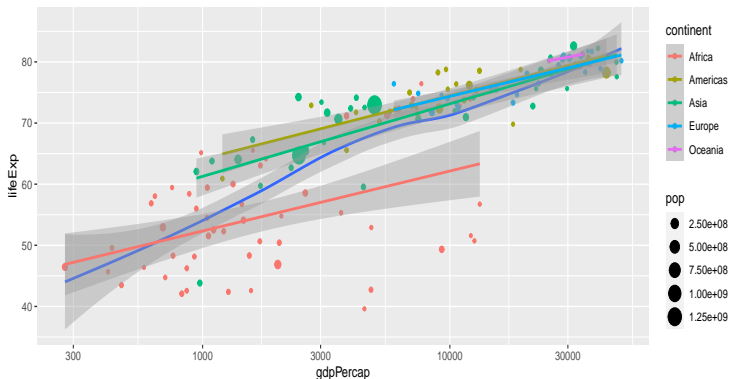
```
fig + scale_x_log10()
```

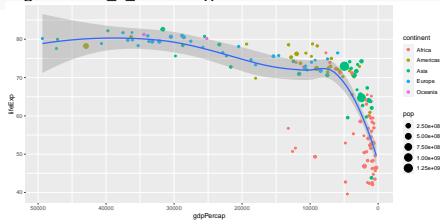
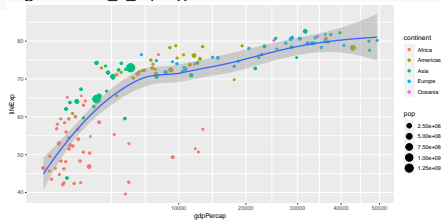


## *geom\_smooth()* with *group* aesthetic

- Continent별로 smoothing curve를 추가하였다.
- 오래 살고 싶으면 어떻게 해야하는가??

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

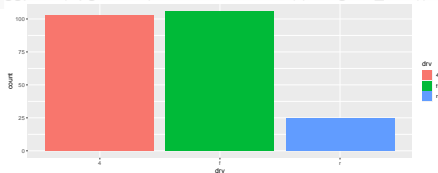


● `scale_x_reverse()``fig + scale_x_reverse()`● `scale_x_sqrt()``fig + scale_x_sqrt()`

### 3. Color scale

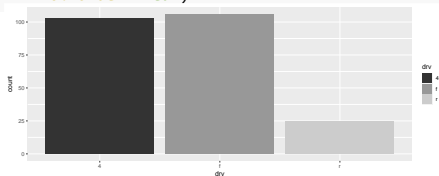
- Original

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



- scale\_fill\_grey()

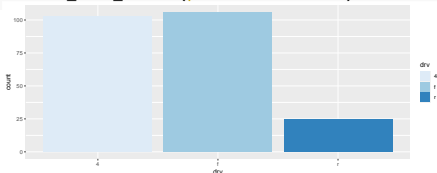
```
ggplot(mpg, aes(x=drv, fill=drv)) + geom_bar() +  
  scale_fill_grey(  
    start = 0.2,  
    end = 0.8,  
    na.value = "red")
```





## ● scale\_fill\_brewer()

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

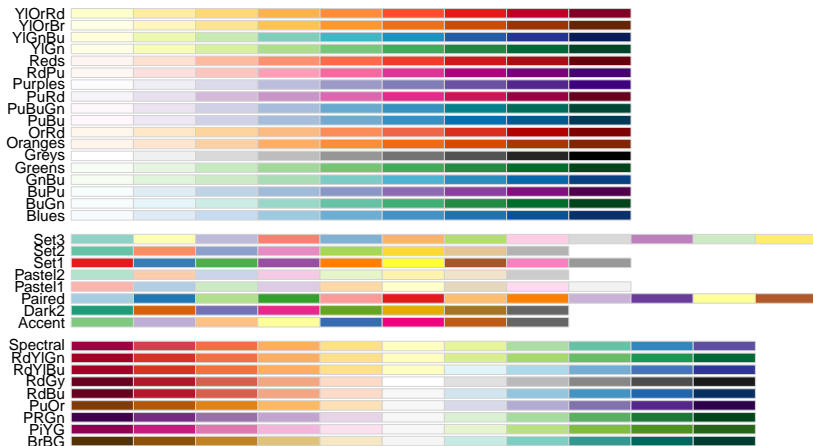


## ● 3개 그림 중에 어느것이 가장 바람직한가?

- drv는 unordered variable
- 두번째 그림의 color lumination은 ordered attribute of vis channel
- 세번째 그림의 color saturation은 ordered attribute of vis channel
- (cf. 무채색의 luminance과 유채색의 saturation은 비슷한 개념이다.)
- 그러므로 색상의 scale은 ordered attribute에 적합하다.

## RColorBrewer에서 제공하는 팔레트들

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

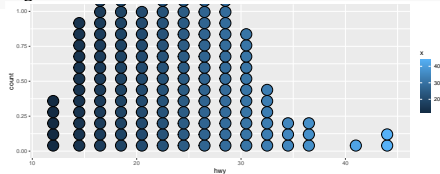


## 4. Color scale on continuous variable

### ● Original

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

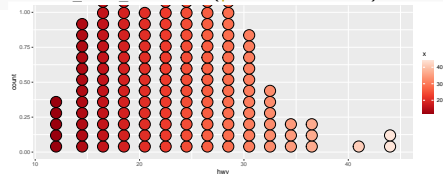
fig



### ● scale\_fill\_distiller()

fig +

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



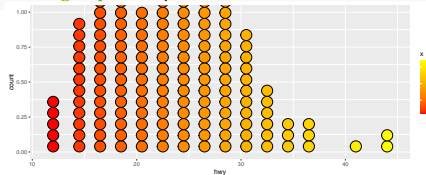
## ● scale\_fill\_gradient()

fig +

```
scale_fill_gradient(
```

```
  low="red",
```

```
  high="yellow")
```



## ● scale\_fill\_gradient2()

fig +

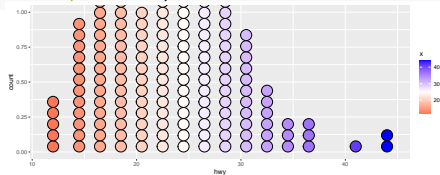
```
scale_fill_gradient2(
```

```
  low="red",
```

```
  high="blue",
```

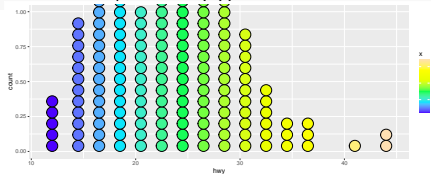
```
  mid = "white",
```

```
  midpoint = 25)
```

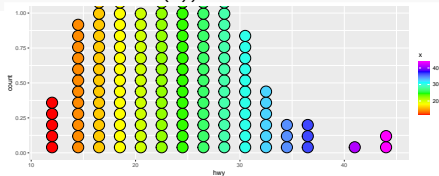


## ● `scale_fill_gradientn()`

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



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



## ● SEE Also:

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

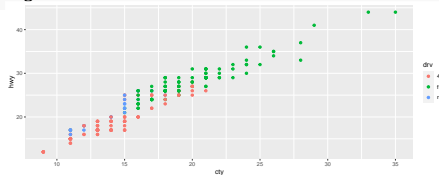
## Part 6. Theme

# About

- Non-data 요소들을 customize할 수 있는 강력한 도구
- titles, labels, fonts, background, gridlines, and legends
- vis들이 consistent하며 customized한 외형을 갖게함

# 1. Style

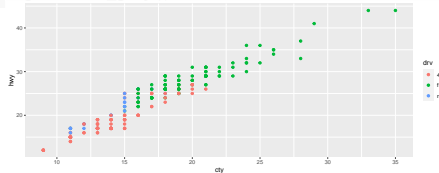
```
fig <- ggplot(mpg, aes(cty,hwy)) +  
  geom_point(aes(color=drv))  
fig
```



## 1. theme\_gray()

- Gray background (default theme)

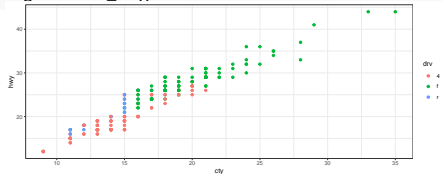
```
fig + theme_gray()
```



## 2. theme\_bw()

- White background with gridlines

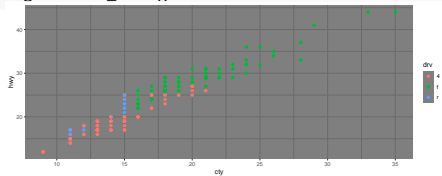
```
fig + theme_bw()
```



## 3. theme\_dark()

- Dark for contrast

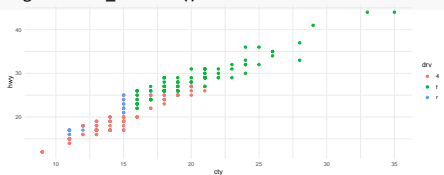
```
fig + theme_dark()
```





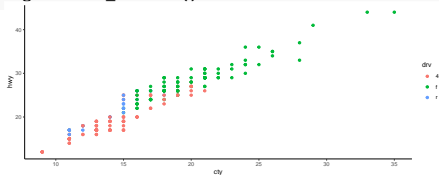
#### 4. theme\_minimal()

```
fig + theme_minimal()
```



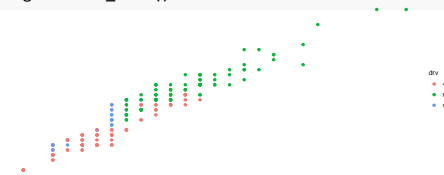
#### 6. theme\_classic()

```
fig + theme_classic()
```



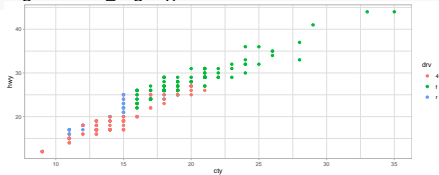
#### 5. theme\_void()

```
fig + theme_void()
```



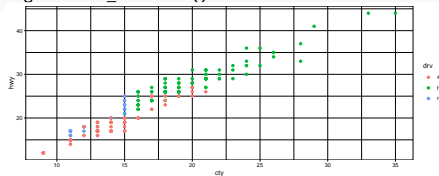
#### 7. theme\_light()

```
fig + theme_light()
```



## 8. theme\_linedraw()

```
fig + theme_linedraw()
```



## 2. Inheritance

- 테마의 요소들은 계층적인 상속을 한다.
- 예를들어 `axis.title.x.bottom` ← `axis.title.x` ← `axis.title` ← `text`의 상속관계를 가진다.
- 모든 text 요소들은 직접/간접적으로 `text`로 부터 상속받는다.
- 모든 line 요소들은 직접/간접적으로 `line`으로 부터 상속받는다.
- 모든 rectangular 요소들은 직접/간접적으로 `rect`으로 부터 상속받는다.
- High-level의 구성요소를 바꾸어서 한번에 여러개 구성요소의 외형을 바꿀수 있다.

## 3. Arguments

### 1) Global

#### Global

Category	Argument	Description
others	...	additional element specifications not part of base <code>ggplot2</code> . If supplied <code>validate</code> needs to be set to <code>FALSE</code> .
	<code>complete</code>	set this to <code>TRUE</code> if this is a complete theme, such as the one returned by <code>theme_grey()</code> . Complete themes behave differently when added to a <code>ggplot</code> object. Also, when setting <code>complete = TRUE</code> all elements will be set to inherit from <code>blank</code> elements.
	<code>validate</code>	<code>TRUE</code> to run <code>validate_element()</code> , <code>FALSE</code> to bypass checks.
	<code>line</code>	all <code>line</code> elements ( <code>element_line()</code> )
	<code>rect</code>	all <code>rect</code> angular elements ( <code>element_rect()</code> )
	<code>text</code>	all <code>text</code> elements ( <code>element_text()</code> )
	<code>title</code>	all <code>title</code> elements: <code>plot</code> , <code>axes</code> , <code>legends</code> ( <code>element_text()</code> ; inherits from <code>text</code> )
	<code>aspect.ratio</code>	aspect ratio of the panel

## 2) panel

### panel

Category	Argument	Description
<code>panel.spacing</code>	<code>panel.spacing</code> , <code>panel.spacing.x</code> , <code>panel.spacing.y</code>	spacing between facet panels (unit). <code>panel.spacing.x</code> & <code>panel.spacing.y</code> inherit from <code>panel.spacing</code> or can be specified separately.
<code>panel.grid</code>	<code>panel.grid</code> , <code>panel.grid.major</code> , <code>panel.grid.minor</code> , <code>panel.grid.major.x</code> , <code>panel.grid.major.y</code> , <code>panel.grid.minor.x</code> , <code>panel.grid.minor.y</code>	grid lines ( <code>element_line()</code> ). Specify major grid lines, or minor grid lines separately (using <code>panel.grid.major</code> or <code>panel.grid.minor</code> ) or individually for each axis (using <code>panel.grid.major.x</code> , <code>panel.grid.minor.x</code> , <code>panel.grid.major.y</code> , <code>panel.grid.minor.y</code> ). Y axis grid lines are horizontal and x axis grid lines are vertical. <code>panel.grid.*.*</code> inherits from <code>panel.grid.*</code> which inherits from <code>panel.grid</code> , which in turn inherits from <code>line</code>
others	<code>panel.background</code>	background of plotting area, drawn underneath plot ( <code>element_rect()</code> ); inherits from <code>rect</code> )
	<code>panel.border</code>	border around plotting area, drawn on top of plot so that it covers tick marks and grid lines. This should be used with <code>fill = NA</code> ( <code>element_rect()</code> ); inherits from <code>rect</code>
	<code>panel.ontop</code>	option to place the panel (background, gridlines) over the data layers (logical). Usually used with a transparent or blank <code>panel.background</code> .

### 3) plot

#### plot

Category	Argument	Description
	<code>plot.background</code>	background of the entire plot ( <code>element_rect()</code> ; inherits from <code>rect</code> )
	<code>plot.title</code>	plot title (text appearance) ( <code>element_text()</code> ; inherits from <code>title</code> ) left-aligned by default
	<code>plot.subtitle</code>	plot subtitle (text appearance) ( <code>element_text()</code> ; inherits from <code>title</code> ) left-aligned by default
	<code>plot.caption</code>	caption below the plot (text appearance) ( <code>element_text()</code> ; inherits from <code>title</code> ) right-aligned by default
	<code>plot.tag</code>	upper-left label to identify a plot (text appearance) ( <code>element_text()</code> ; inherits from <code>title</code> ) left-aligned by default
	<code>plot.tag.position</code>	The position of the tag as a string ( "topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright" ) or a coordinate. If a string, extra space will be added to accommodate the tag.
	<code>plot.margin</code>	margin around entire plot (unit with the sizes of the top, right, bottom, and left margins)

4) *axis***axis**

Category	Argument	Description
<b>axis.title</b>	<code>axis.title</code> , <code>axis.title.x</code> , <code>axis.title.y</code> , <code>axis.title.x.top</code> , <code>axis.title.x.bottom</code> , <code>axis.title.y.left</code> , <code>axis.title.y.right</code>	labels of axes ( <code>element_text()</code> ). Specify all axes' labels ( <code>axis.title</code> ), labels by plane (using <code>axis.title.x</code> or <code>axis.title.y</code> ), or individually for each axis (using <code>axis.title.x.bottom</code> , <code>axis.title.x.top</code> , <code>axis.title.y.left</code> , <code>axis.title.y.right</code> ). <code>axis.title.*</code> inherits from <code>axis.title</code> which inherits from <code>axis.title</code> , which in turn inherits from <code>text</code>
<b>axis.text</b>	<code>axis.text</code> , <code>axis.text.x</code> , <code>axis.text.y</code> , <code>axis.text.x.top</code> , <code>axis.text.x.bottom</code> , <code>axis.text.y.left</code> , <code>axis.text.y.right</code>	tick labels along axes ( <code>element_text()</code> ). Specify all axis tick labels ( <code>axis.text</code> ), tick labels by plane (using <code>axis.text.x</code> or <code>axis.text.y</code> ), or individually for each axis (using <code>axis.text.x.bottom</code> , <code>axis.text.x.top</code> , <code>axis.text.y.left</code> , <code>axis.text.y.right</code> ). <code>axis.text.*</code> inherits from <code>axis.text</code> which inherits from <code>axis.text</code> , which in turn inherits from <code>text</code>
<b>axis.ticks</b>	<code>axis.ticks</code> , <code>axis.ticks.x</code> , <code>axis.ticks.x.top</code> , <code>axis.ticks.x.bottom</code> , <code>axis.ticks.y</code> , <code>axis.ticks.y.left</code> , <code>axis.ticks.y.right</code>	tick marks along axes ( <code>element_line()</code> ). Specify all tick marks ( <code>axis.ticks</code> ), ticks by plane (using <code>axis.ticks.x</code> or <code>axis.ticks.y</code> ), or individually for each axis (using <code>axis.ticks.x.bottom</code> , <code>axis.ticks.x.top</code> , <code>axis.ticks.y.left</code> , <code>axis.ticks.y.right</code> ). <code>axis.ticks.*</code> inherits from <code>axis.ticks</code> which inherits from <code>axis.ticks</code> , which in turn inherits from <code>line</code>
	<code>axis.ticks.length</code>	length of tick marks (unit)
<b>axis.line</b>	<code>axis.line</code> , <code>axis.line.x</code> , <code>axis.line.x.top</code> , <code>axis.line.x.bottom</code> , <code>axis.line.y</code> , <code>axis.line.y.left</code> , <code>axis.line.y.right</code>	lines along axes ( <code>element_line()</code> ). Specify lines along all axes ( <code>axis.line</code> ), lines for each plane (using <code>axis.line.x</code> or <code>axis.line.y</code> ), or individually for each axis (using <code>axis.line.x.bottom</code> , <code>axis.line.x.top</code> , <code>axis.line.y.left</code> , <code>axis.line.y.right</code> ). <code>axis.line.*</code> inherits from <code>axis.line</code> which inherits from <code>axis.line</code> , which in turn inherits from <code>line</code>

## 5) Legend

### legend

Category	Argument	Description
<code>legend.key</code>	<code>legend.key</code>	background underneath legend keys ( <code>element_rect()</code> ); inherits from <code>rect</code> )
	<code>legend.key.size</code> , <code>legend.key.height</code> , <code>legend.key.width</code>	size of legend keys (unit); key background height & width inherit from <code>legend.key.size</code> or can be specified separately
<code>legend.text</code>	<code>legend.text</code>	legend item labels ( <code>element_text()</code> ); inherits from <code>text</code> )
	<code>legend.text.align</code>	alignment of legend labels (number from 0 (left) to 1 (right))
<code>legend.title</code>	<code>legend.title</code>	title of legend ( <code>element_text()</code> ); inherits from <code>title</code> )
	<code>legend.title.align</code>	alignment of legend title (number from 0 (left) to 1 (right))
<code>legend.box</code>	<code>legend.box</code>	arrangement of multiple legends ( "horizontal" or "vertical" )
	<code>legend.box.just</code>	justification of each legend within the overall bounding box, when there are multiple legends ( "top", "bottom", "left", or "right" )
	<code>legend.box.margin</code>	margins around the full legend area, as specified using <code>margin()</code>
	<code>legend.box.background</code>	background of legend area ( <code>element_rect()</code> ); inherits from <code>rect</code> )
	<code>legend.box.spacing</code>	The spacing between the plotting area and the legend box (unit)



(cont'd)

others	<code>legend.background</code>	background of legend ( <code>element_rect()</code> ; inherits from <code>rect</code> )
	<code>legend.margin</code>	the margin around each legend ( <code>margin()</code> )
	<code>legend.spacing</code> , <code>legend.spacing.x</code> , <code>legend.spacing.y</code>	the spacing between legends (unit). <code>legend.spacing.x</code> & <code>legend.spacing.y</code> inherit from <code>legend.spacing</code> or can be specified separately
	<code>legend.position</code>	the position of legends ( "none", "left", "right", "bottom", "top" , or two-element numeric vector)
	<code>legend.direction</code>	layout of items in legends ( "horizontal" or "vertical" )
	<code>legend.justification</code>	anchor point for positioning legend inside plot ( "center" or two-element numeric vector) or the justification according to the plot area when positioned outside the plot

## 6) strip

### strip

Category	Argument	Description
strip.background	strip.background, strip.background.x, strip.background.y	background of facet labels ( <code>element_rect()</code> ); inherits from <code>rect</code> ). Horizontal facet background ( <code>strip.background.x</code> ) & vertical facet background ( <code>strip.background.y</code> ) inherit from <code>strip.background</code> or can be specified separately
strip.text	strip.text, strip.text.x, strip.text.y	facet labels ( <code>element_text()</code> ); inherits from <code>text</code> ). Horizontal facet labels ( <code>strip.text.x</code> ) & vertical facet labels ( <code>strip.text.y</code> ) inherit from <code>strip.text</code> or can be specified separately
strip.switch.pad	strip.switch.pad.grid	space between strips and axes when strips are switched (unit)
	strip.switch.pad.wrap	space between strips and axes when strips are switched (unit)
others	strip.placement	placement of strip with respect to axes, either "inside" or "outside". Only important when axes and strips are on the same side of the plot.

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

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