

EDA and Data Visualization

2020 년 봄 학기

2020 년 02 월 28 일

https://www.github.com/KU-BIG/KUBIG_2019_Spring

1. EDA (Exploratory Data Analysis)

“탐색적 데이터 분석 (EDA)은 우리가 존재한다고 믿는 것들은 물론이고 존재하지 않는다고 믿는 것들을 발견하려는 태도, 유연성, 그리고 자발성이다.”

-Schut Rachel (Doing Data Science 의 저자)

1.1. EDA 정의

수집한 데이터를 다양한 각도에서 관찰하고 이해하는 과정. 데이터를 분석하기 전에 그래프나 통계적인 방법으로 자료를 직관적으로 바라보는 과정

1.2. EDA 필요성

첫째, 데이터의 분포 및 값을 검토함으로써 데이터가 표현하는 현상을 더 잘 이해하고, 데이터에 대한 잠재적인 문제를 발견할 수 있다. 이를 통해, 본격적인 분석에 들어가기에 앞서 수집의사를 결정할 수 있다.

둘째, 다양한 각도에서 살펴보는 과정을 통해 문제 정의 단계에서 미처 발생하지 못했을 다양한 패턴을 발견하고, 이를 바탕으로 기존의 가설을 수정하거나 새로운 가설을 세울 수 있다.

1.3. EDA 특성

①저항성(Resistance to outliers, missing data, or miscoded data)

- 이상치, 결측치, 입력 오류에 영향을 받지 않는 tool 을 사용한다

②잔차의 해석(Residual is a off-value from the main stream)

-왜 잔차가 존재하는지에 대해 탐색하는 작업

③자료의 재표현(Data Re-expression)

-자료의 여러 가지 성질을 나타낼 수 있는 다양한 형태로 표현 (로그 사용)

④자료의 현시성(Graphic presentation)

-시각화를 통해 데이터 의미를 직관적으로 전달

1.4. EDA 과정

기본적인 출발점은 문제 정의 단계에서 세웠던 연구 질문과 가설을 바탕으로 분석 계획을 세우는 것이다. 분석 계획에는 어떤 속성 및 속성 간의 관계를 집중적으로 관찰해야 할지, 이를 위한 최적의 방법은 무엇인지가 포함되어야 한다.

①분석의 목적과 변수가 무엇이 있는지 확인. 개별 변수의 이름이나 설명을 가지는지 확인.

②데이터를 전체적으로 살펴보기 : 데이터에 문제가 없는지 확인. head 나 tail 부분을 확인, 추가적으로 다양한 탐색. (이상치, 결측치 등을 확인하는 과정)

③데이터의 개별 속성값을 관찰 : 각 속성값이 예측한 범위와 분포를 갖는지 확인. 만약 그렇지 않다면, 이유가 무엇인지를 확인해 본다.

④속성 간의 관계에 초점을 맞추어, 개별 속성 관찰에서 찾아내지 못했던 패턴을 발견한다.

(상관관계, 시각화 등)

1.5. EDA 주의점

많은 사람들이 EDA 관련하여 실수하는 부분은 실제로 무엇을 연구하는지도 잘 모르면서 그들의 방법론을 “탐구적”이라고 주장하는 것이다. 이렇게 EDA 라는 이름으로 열악한 연구가 종종 진행된다. EDA 가 사전 결정된 가설 테스트를 필요로 하지 않는 것은 사실이나, 연구 질문이나 변수가 없거나 유효한 p-value 를 얻을 때까지 모든 테스트를 시도하는 것을 정당화 하지는 않는다. It is true that EDA does not require a pre-determined hypothesis to be tested, but it doesn't justify the absence of research questions and ill-defined variables or trying every test until obtaining a significant p value (p-hacking) (Jebb, Parrigon, & Woo, 2017).

탐색적 데이터 분석을 설립한 John Tukey 는 EDA 를 탐정 업무와 비유하며, 데이터의 타당한 스토리가 나타날 때까지 가능한 한 많은 방법으로 데이터를 탐색하는 것이라고 하였다. 형사는 아무 정보를 모두 수집하는 게 아니라, 사전의 중심 문제와 관련된 증거와 단서를 수집하는 것이다.

1.6. EDA 요소 (네 가지 방법)

Velleman and Hoaglin (1981)은 EDA 의 네 가지 기본 요소를 다음과 같이 설명하였다.

- Data visualization
- Residual analysis
- Data transformation or re-expression
- Resistance procedures

특히 Data Visualization (데이터 시각화)는 데이터의 특징을 쉽고 간편하게 관찰하여 EDA 의 과정에 꼭 필요한 요소라고 할수 있다. 이 자료에서는 Data visualization 에 관하여서 자세히 다룰 것이다. 나머지 EDA 방법은 <https://www.creative-wisdom.com/teaching/WBI/EDA.shtml> 에서 확인할 수 있다.

2. 데이터 시각화 (Data Visualization)

“A picture is worth a thousand words.” 그림은 천 단어의 가치가 있다. 데이터 사이언스 또는 빅데이터에서의 시각화는 그보다 몇 백 만개의 데이터를 한 그림으로 나타낼 수 있다.

데이터 시각화는 중요한 만큼, 잘못되거나 과장된 인식을 제공할 수 있을 만큼 오해의 요소가 있기 때문에 데이터 사이언티스트는 조심해야 한다.

(데이터 시각화 원칙: <https://rafalab.github.io/dsbook/data-visualization-principles.html> 참고)

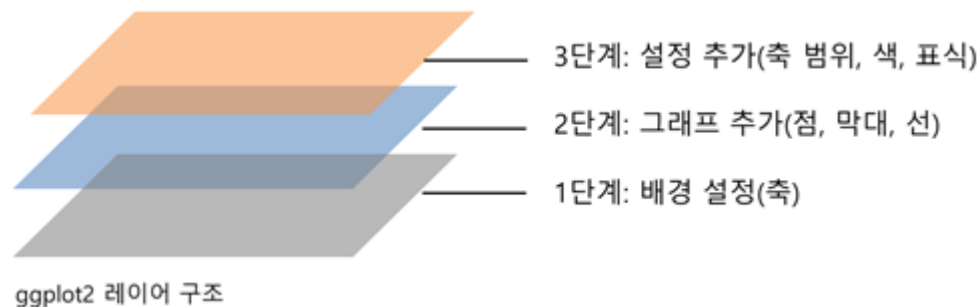
데이터 시각화의 중요성에 대해서는 간추리고, 데이터 시각화를 위한 코딩 방법론에 대하여 자세히 설명하고자 한다.

3. R 의 시각화 패키지 – ggplot2

R 에는 시각화를 구현할 수 있는 다양한 패키지들이 있지만, ggplot2 는 그 중에서 가장 널리 사용되고 활용도가 높은 패키지이다. 또한 ggplot2 를 이용하면 쉽고 짧은 문법으로 아름다운 그래프를 만들 수가 있다. 따라서 R 에서는 ggplot2 패키지를 활용하는 방법에 대해 다룰 것이다.

3.1 ggplot2 의 문법

ggplot2 의 문법은 레이어 구조로 되어있다. 배경을 만들고, 그 위에 그래프 형태를 그리고, 마지막으로 축 범위, 색, 표식 등 설정을 추가하는 순서로 그래프를 만든다.



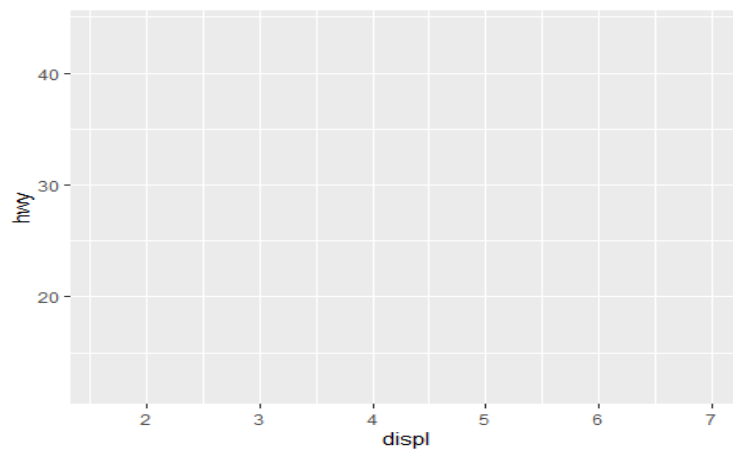
ggplot2 패키지의 코드를 작성할 때, 우리는 + 기호를 이용하여 각 요소를 추가해나갈 수 있다. 또한, 꼭 해야 하는 것은 아니지만 일반적으로 + 기호 뒤에서 Enter 키를 눌러 줄을 바꾸면 가독성 있는 코드를 작성할 수가 있다.

그러면 이제 레이어 구조의 순서에 따라 각각 산점도, 막대그래프, 선 그래프, 상자 그림을 그려보려 한다. 데이터는 ggplot2 패키지 안에 내장되어있는 mpg 데이터와 economics 데이터를 이용할 것이다.

3.2 산점도(Scatter Plot) 만들기

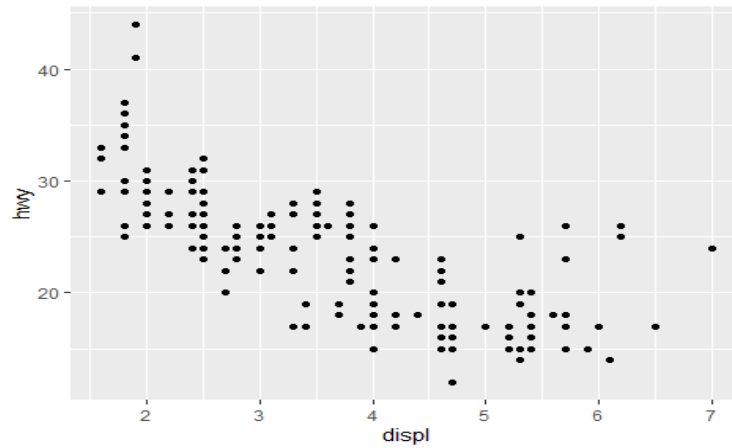
1 단계 배경 설정: 가장 먼저 그래프를 그릴 배경을 만들어준다. data 에 그래프를 그리는 데 사용할 데이터를 지정해주고, aes 에는 x 축과 y 축에 사용할 변수를 지정하면 배경이 만들어진다. x 축에는 mpg 데이터의 displ(배기량), y 축에는 hwy(고속도로 연비)를 각각 변수로 지정해 줄 것이다.

```
ggplot(data = mpg, aes(x = displ, y = hwy))
```



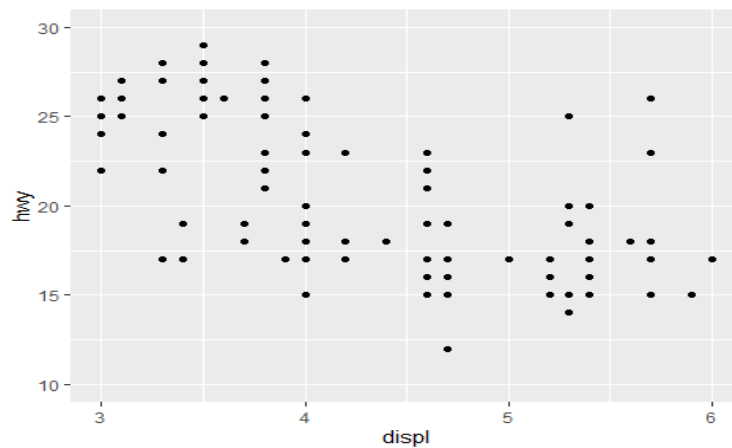
2 단계 그래프 추가: 배경을 만들었으니 그 위에 그래프를 그릴 차례이다. 앞서 말했듯이 + 기호를 이용해 그래프 유형을 지정하는 함수를 추가하면 된다. 산점도를 그리는 함수는 geom_point 이므로, 1 단계에서 완성한 코드 뒤에 '+ geom_point()'를 추가해준다.

```
ggplot(data=mpg, aes(x=displ, y=hwy)) + geom_point()
```



3 단계 설정 추가: + 기호를 이용하면 그래프 설정을 변경하는 코드를 추가할 수가 있다. 여기서서는 축의 범위를 변경해보려 한다. xlim()과 ylim()을 이용하면 축이 시작되는 값과 끝나는 값을 설정해줄 수가 있다. x 축의 범위는 3~6, y 축의 범위는 10~30 로 지정해주는 코드는 다음과 같다

```
ggplot(data = mpg, aes(x = displ, y = hwy)) +  
  geom_point() +  
  xlim(3, 6) +  
  ylim(10, 30)
```



3.3 막대그래프(Bar chart) 만들기

막대그래프의 경우 평균 막대 그래프와 빈도 막대 그래프, 두 가지를 그려볼 것이다. 요약표를 이용하는지에 따라 그래프를 만드는 절차와 함수가 다르다. 요약표는 `geom_col()`, 원자료는 `geom_bar()`을 사용해 막대 그래프를 만든다.

3.3.1 평균 막대 그래프

mpg 데이터를 이용해 drv(구동방식)별 평균 hwy(고속도로 연비) 막대그래프를 그려 볼 것이다. 평균 막대 그래프는 데이터를 요약한 평균표를 먼저 만든 후 이 평균표를 이용해 만든다. 다음과 같은 작업을 통해 평균 hwy 값을 나타내는 새로운 변수인 `mean_hwy` 을 만들고, `drv` 와 `mean_hwy` 을 변수로 갖는 새로운 데이터 프레임 `df_mpg` 를 만들 수 있다. 요약표를 완성해준 뒤, 앞서 언급한 것처럼 레이어 순서대로 차근차근 그래프를 만들어주면 된다.

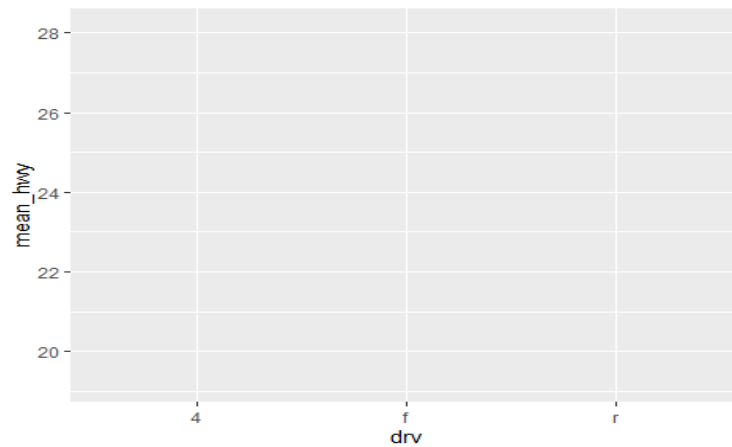
```
df_mpg <- mpg %>%
  group_by(drv) %>%
  summarise(mean_hwy = mean(hwy))
```

```
df_mpg
```

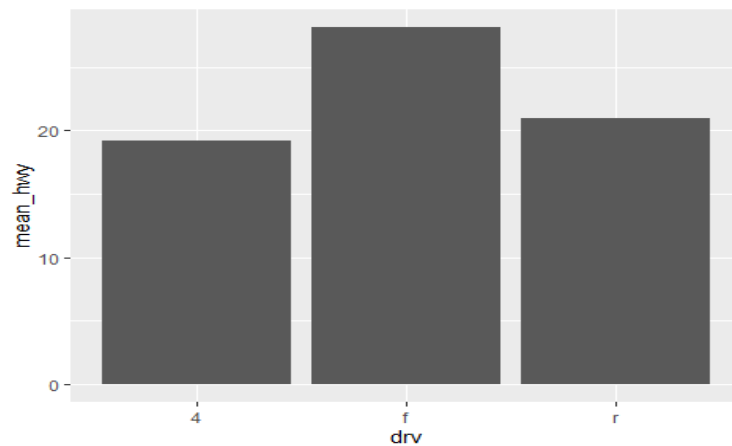
```
## # A tibble: 3 x 2
##   drv   mean_hwy
##   <chr>   <dbl>
## 1 4      19.2
## 2 f      28.2
## 3 r      21
```

1 단계 배경 설정: 이번에는 사용하는 데이터가 `df_mpg`, x 축에 지정할 변수는 `drv`, 그리고 y 축에 지정할 변수는 `mean_hwy` 가 된다.


```
ggplot(data = df_mpg, aes(x = drv, y = mean_hwy))
```



2 단계 그래프 추가: 요약표를 이용하여 막대 그래프를 만들 때는 `geom_col()`을 사용한다. 따라서 + 기호를 통해 `geom_col()`을 추가해준다.

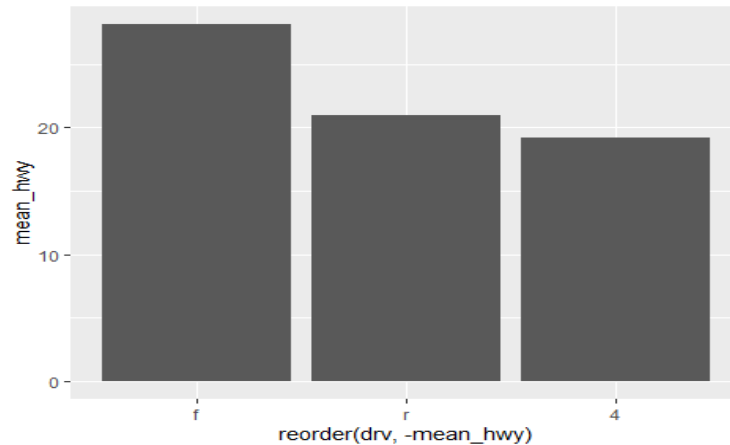


3 단계 설정 추가: 설정 추가는 필수적인 부분은 아니니 생략하도록 하겠다.

막대는 기본적으로 범주의 알파벳 순서로 정렬이 된다. 만약 막대를 값의 크기 순으로 정렬하고 싶다면 `reorder()`를 사용하면 된다. `reorder()`에 x 축 변수와 정렬 기준으로 삼을 변수를 지정해주면 된다. 정렬 기준 변수 앞에 - 기호를 붙이면 내림차순으로 정렬된다. x 축 변수 `drv`를

mean_hwy 값에 따라 내림차순으로 정렬해줄 것이기 때문에, `reorder(drv, -mean_hwy)`을 사용하도록 하겠다. 그 결과는 다음과 같다.

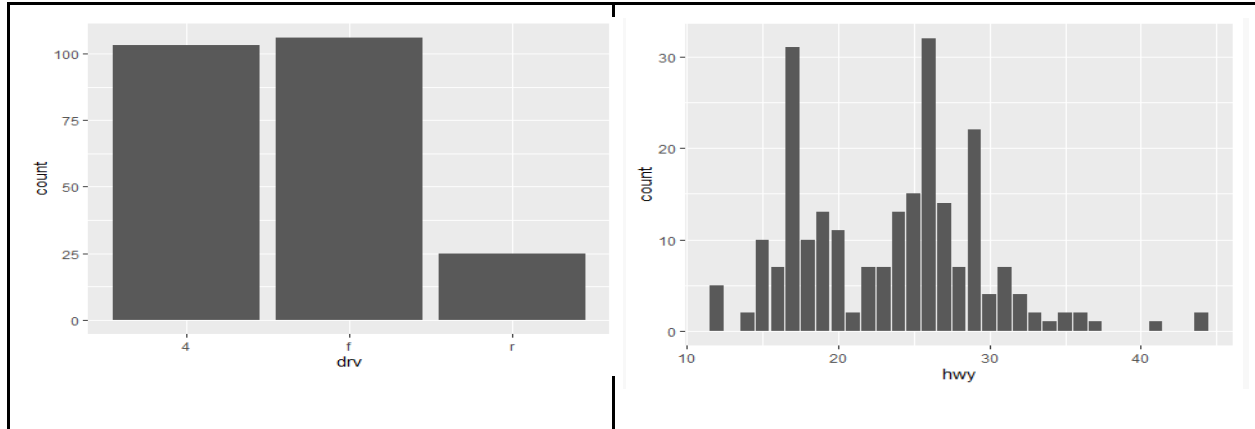
```
ggplot(data = df_mpg, aes(x = reorder(drv, -mean_hwy), y=mean_hwy)) +  
  geom_col()
```



3.3.2 빈도 막대 그래프

빈도 막대 그래프는 값의 개수(빈도)로 막대의 길이를 표현한 그래프이다. 빈도 막대 그래프를 만들려면 y 축 없이 x 축만 지정하고, `geom_col()` 대신 `geom_bar()`을 사용하면 된다.

<p><drv 변수 항목별 빈도 막대 그래프></p> <pre>ggplot(data = mpg, aes(x = drv)) + geom_bar()</pre>	<p><hwy 변수의 분포를 나타낸 빈도 막대 그래프></p> <pre>ggplot(data = mpg, aes(x = hwy)) + geom_bar()</pre>
--	---

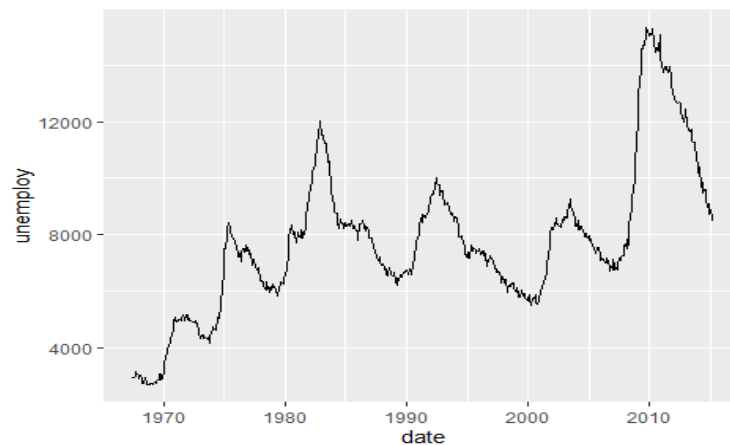


3.4 선 그래프(Line Chart) 만들기

선 그래프를 이용하면, 시간에 따라 달라지는 데이터를 표현하기가 용이하다. economics 데이터를 이용해 시간에 따라 실업자 수가 어떻게 변하는지, 시계열 그래프로 나타내 보려 한다.

이 경우, data 에는 economics, x 축에는 시간을 의미하는 date, y 축에는 실업자 수를 의미하는 unemploy 를 지정해 주어야 한다. 또한 선 그래프를 그리는 함수는 geom_line()이므로, geom_line()를 추가해주면 된다. 그러면 다음과 같은 그래프가 완성이 된다.

```
ggplot(data = economics, aes(x = date, y = unemploy)) + geom_line()
```

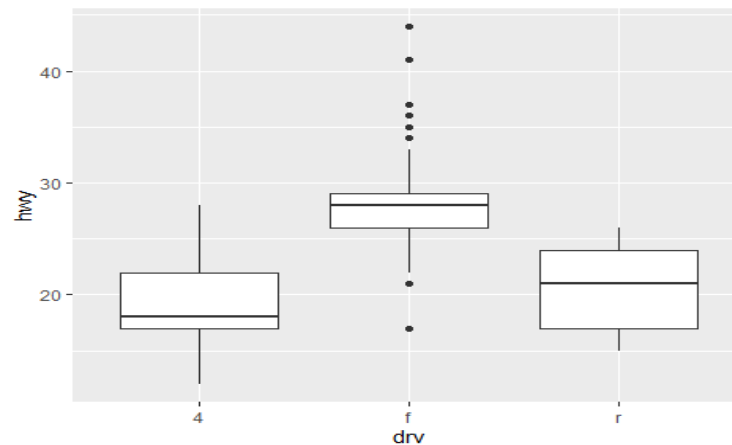


3.5 상자그림(Box Plot) 만들기

상자 그림은 데이터의 분포를 직사각형 상자 모양으로 표현한 그래프이다. 따라서 상자 그림을 보면 데이터의 분포를 알 수 있기 때문에 데이터의 특징을 좀 더 자세히 이해할 수 있다. mpg 데이터의 drv(구동 방식)별 hwy(고속도로 연비)를 상자그림으로 표현해 볼 것이다.

이 경우 data 에는 mpg, x 축에는 drv, y 축에는 hwy 를 지정해준다. 그리고 뒤에 geom_boxplot()을 추가해주면 상자 그림으로 표현할 수가 있다.

```
ggplot(data = mpg, aes(x = drv, y = hwy)) + geom_boxplot()
```

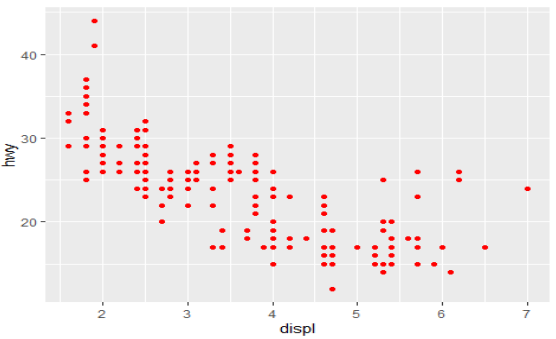
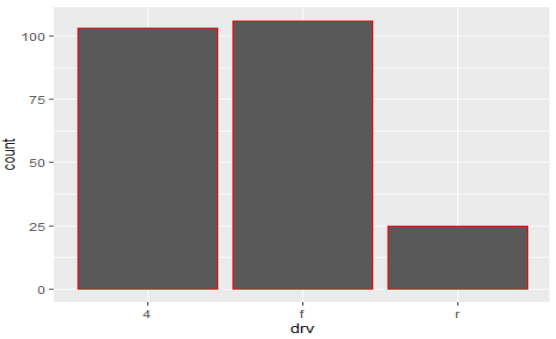
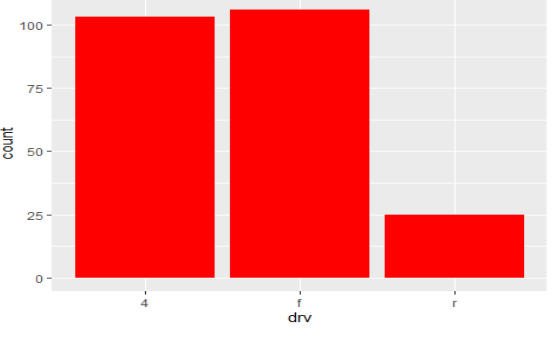


3.6 기타 옵션

지금까지 기본적인 그래프를 그리는 방법에 대해 살펴보았다. 이렇게만 해도 어느 정도의 간단한 시각화는 구현할 수 있지만, 다양한 옵션을 추가적으로 사용하면 더 다채롭고 풍성한 그래프를 그릴 수 있다. 그 추가적인 방법들을 몇 가지만 간단히 소개해보겠다. 이 외 더 자세한 내용은 R 스튜디오의 치트 시트(Cheat Sheet)를 참고하길 바란다.

3.6.1 그래프에 색 입히기

그래프에 색을 입히기 위해서는 추가하려는 그래프의 함수에 color 지정을 해주면 된다. 예를 들어 빨간색 점이 찍히는 산점도를 그리고 싶을 때는 `geom_point(color = "red")`을 추가해주면 된다. 그러나 막대 그래프나 상자그래프 등에 있어서는 주의해야 점이 하나 있는데, `color = "red"`를 해주게 되면, 아래와 같이 겉의 테두리만 빨간색인 결과가 나오게 된다는 점이다. 막대가 전부 빨간색이길 원한다면 `color`가 아닌 `fill = "red"`를 입력해 주어야 한다.

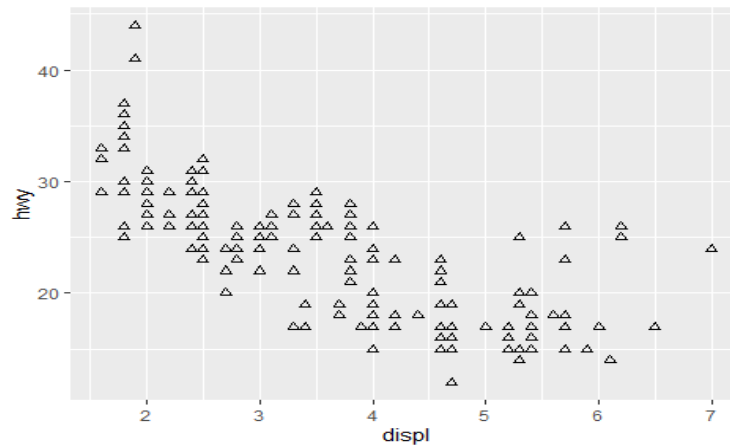
<pre>ggplot(data=mpg, aes(x=displ, y=hwy)) + geom_point(color = "red")</pre>	 <p>A scatter plot showing the relationship between engine displacement (displ) on the x-axis and highway mileage (hwy) on the y-axis. The points are colored red. The x-axis ranges from approximately 1.6 to 7.0, and the y-axis ranges from approximately 12 to 44.</p>								
<pre>ggplot(data = mpg, aes(x = driv)) + geom_bar(color = "red")</pre>	 <p>A bar chart showing the count of vehicles for each drive type (driv). The x-axis categories are 4, f, and r. The y-axis represents the count, ranging from 0 to 100. The bars have red outlines and dark gray fills.</p> <table border="1"> <thead> <tr> <th>driv</th> <th>count</th> </tr> </thead> <tbody> <tr> <td>4</td> <td>104</td> </tr> <tr> <td>f</td> <td>105</td> </tr> <tr> <td>r</td> <td>25</td> </tr> </tbody> </table>	driv	count	4	104	f	105	r	25
driv	count								
4	104								
f	105								
r	25								
<pre>ggplot(data = mpg, aes(x = driv)) + geom_bar(fill = "red")</pre>	 <p>A bar chart showing the count of vehicles for each drive type (driv). The x-axis categories are 4, f, and r. The y-axis represents the count, ranging from 0 to 100. The bars are solid red.</p> <table border="1"> <thead> <tr> <th>driv</th> <th>count</th> </tr> </thead> <tbody> <tr> <td>4</td> <td>104</td> </tr> <tr> <td>f</td> <td>105</td> </tr> <tr> <td>r</td> <td>25</td> </tr> </tbody> </table>	driv	count	4	104	f	105	r	25
driv	count								
4	104								
f	105								
r	25								

3.6.2 기호의 모양 설정하기

그래프 함수 안에 `pch` 옵션을 지정해주면, 기호의 모양을 따로 설정해 줄 수가 있다. 예를 들어 속이 빈 세모 모양을 사용하고 싶다면, `pch = 2` 을 추가해주면 된다. `pch` 옵션의 번호와 모양은 다음과 같다.

0	1	2	3	4	
□	○	△	+	×	
5	6	7	8	9	
◇	▽	⊠	✱	⬡	
10	11	12	13	14	
⊕	⊗	田	⊗	⊠	
15	16	17	18	19	
■	●	▲	◆	●	
20	21	22	23	24	25
●	●	■	◆	▲	▼

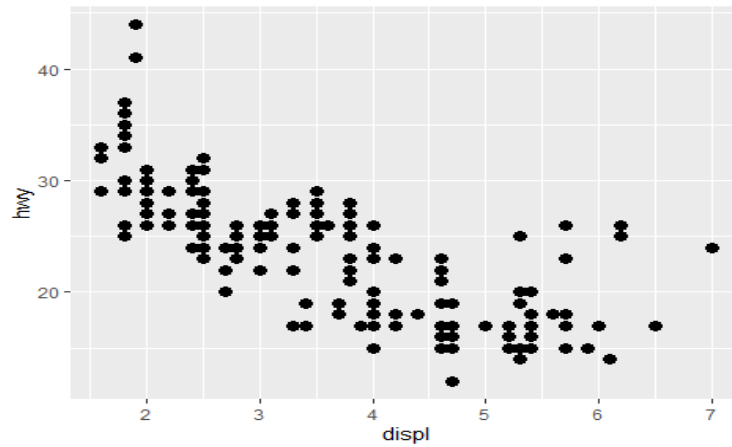
```
ggplot(data=mpg, aes(x=displ, y=hwy)) +  
  geom_point(pch = 2)
```



3.6.3 기호의 사이즈 설정하기

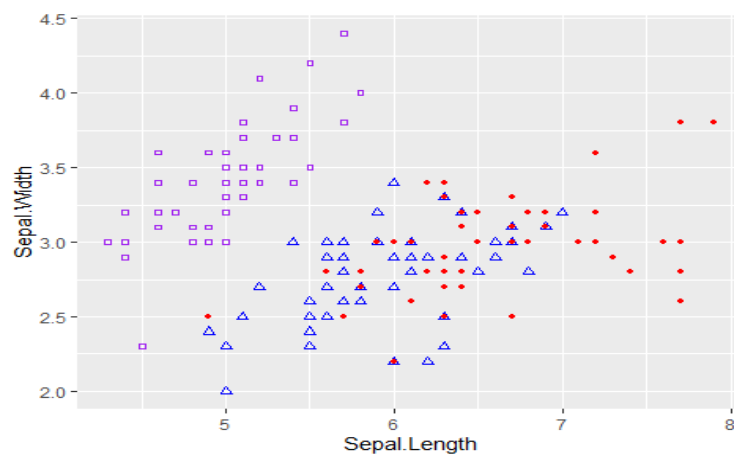
기호의 사이즈 역시 설정해줄 수가 있다. 그래프 함수 안에 `size = 숫자` 를 추가해주면, 그 숫자만큼의 사이즈로 기호가 나타난다.

```
ggplot(data=mpg, aes(x=displ, y=hwy)) + geom_point(size = 3)
```



이 때, 옵션을 단일하게 지정하지 않고 그룹별 옵션을 설정할 수도 있다. 아래의 그래프는 iris 데이터에서 Species에 따라 기호의 색상, 모양, 크기를 다르게 나타낸 결과이다.

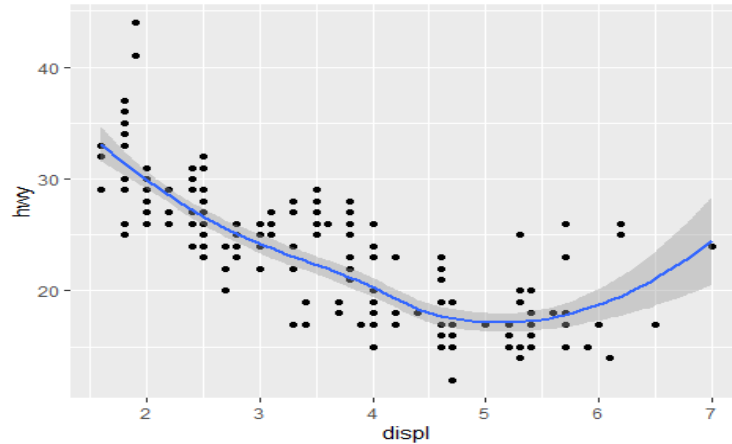
```
ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width)) +  
  geom_point(color = c("purple", "blue", "red")[iris$Species],  
            pch = c(0, 2, 20)[iris$Species],  
            size = c(1, 1.5, 2)[iris$Species])
```



3.6.4 복수의 그래프 겹쳐 그리기

+ 기호를 통해 연결해주면, 복수의 geom 함수를 동시에 그릴 수가 있다.

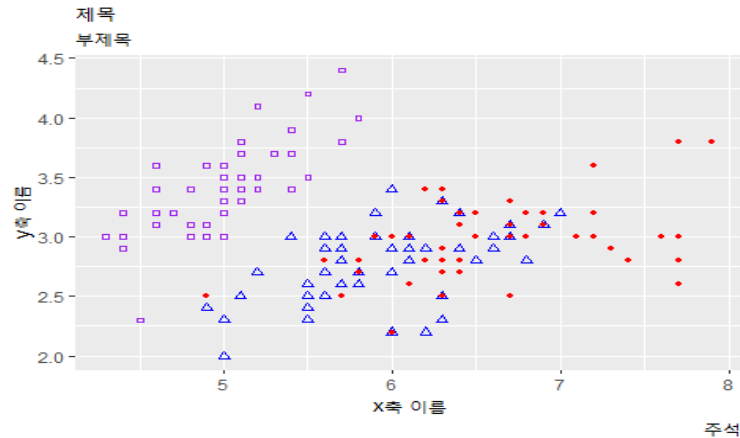
```
ggplot(data = mpg, aes(x = displ, y = hwy)) +  
  geom_point() +  
  geom_smooth()
```



3.6.5 라벨링하기

labs()을 이용하면 그래프에 라벨링을 해줄 수가 있다.

```
ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width)) +  
  geom_point(color = c("purple", "blue", "red")[iris$Species],  
    pch = c(0, 2, 20)[iris$Species],  
    size = c(1, 1.5, 2)[iris$Species]) +  
  labs(title = "제목", subtitle = "부제목", caption = "주석",  
    x = "x 축 이름", y = "y 축 이름")
```

3.6.6 변수의 level 별로 sub 그래프 그리기

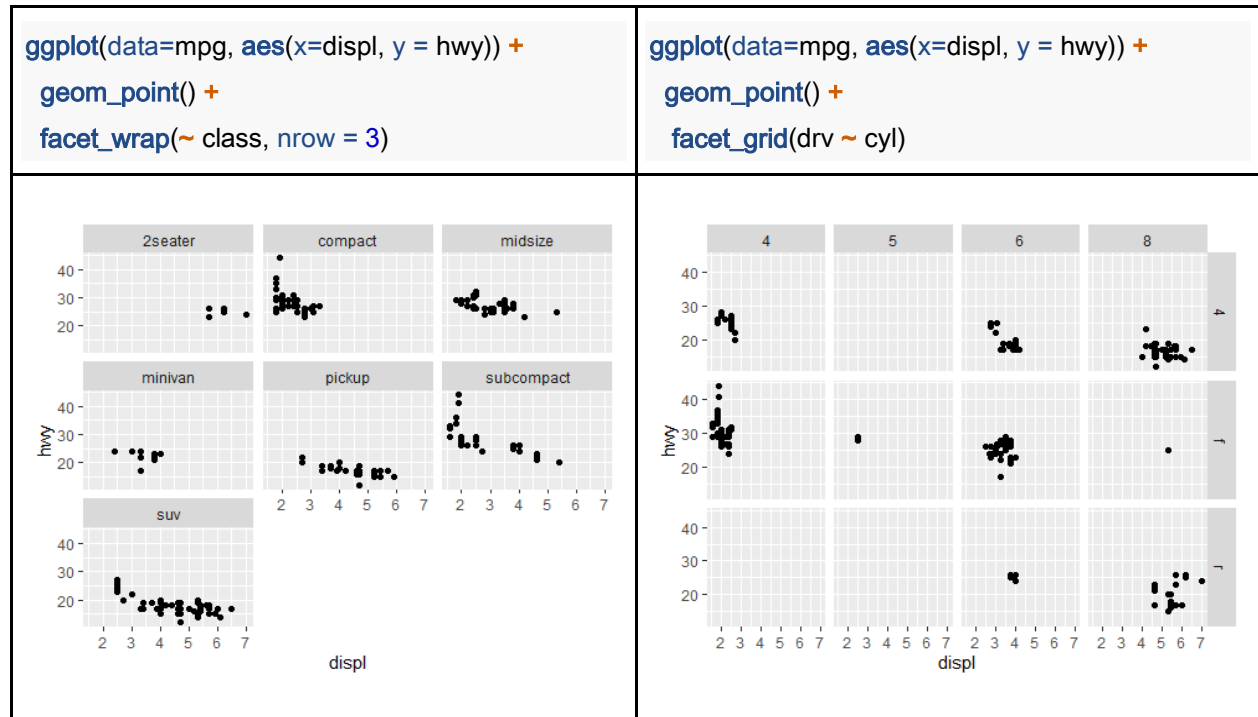
facet 함수를 이용하면 변수의 level 별로 sub 그래프를 각기 다른 패널에 그릴 수가 있다. facet 함수는 `facet_wrap()`와 `facet_grid()` 함수로 구분될 수 있다.

-`facet_wrap()`

`facet_wrap()` 함수 안에 ~ “변수명” 을 입력해주면, 물결 무늬 오른쪽에 기재되는 변수의 level 순서대로 sub 그래프가 그려진다. 이 때, sub 그래프들은 1 차원(한쪽 방향: 왼쪽에서 오른쪽)으로만 그려진다. 또한 `nrow` 와 `ncol` 을 이용해 sub 그래프가 그려지는 행 및 열의 수를 지정할 수가 있다. 예를 들어 `nrow=3` 을 설정해주면, 아래와 같이 3 개 행으로 그래프들이 그려진다.

-`facet_grid()`

`facet_grid()` 함수 안에 “변수명” ~ “변수명” 을 입력해주면, 물결 무늬 좌/우 변수를 각각 행/열로 나누어 2 차원으로 sub 그래프들을 그려준다. 다만, `facet_wrap()` 함수와는 달리 `nrow` 및 `ncol` 옵션을 적용할 수가 없다.



4. Python 의 시각화 패키지 – Matplotlib

Matplotlib 은 파이썬에서 데이터를 차트나 플롯으로 그려주는 패키지로서, 가장 많이 사용되는 데이터 시각화 패키지 중 하나이다. 이번 장에서는 Matplotlib 를 이용하여 기본적인 그래프를 그려보는 내용을 다룰 것이다. 더 다양한 예제나 내용이 궁금하다면, Matplotlib 갤러리 웹사이트를 참고하기 바란다. (<http://matplotlib.org/gallery.html>)

4.1 Matplotlib 불러오기

Matplotlib 를 사용하기 위해서는 먼저 `matplotlib.pyplot` 를 import 해야한다. `pyplot` 을 다른 이름으로 사용할 수 있지만, 통상적으로 `plt` 이라는 표현을 사용한다.

```
import matplotlib
import matplotlib.pyplot as plt
```

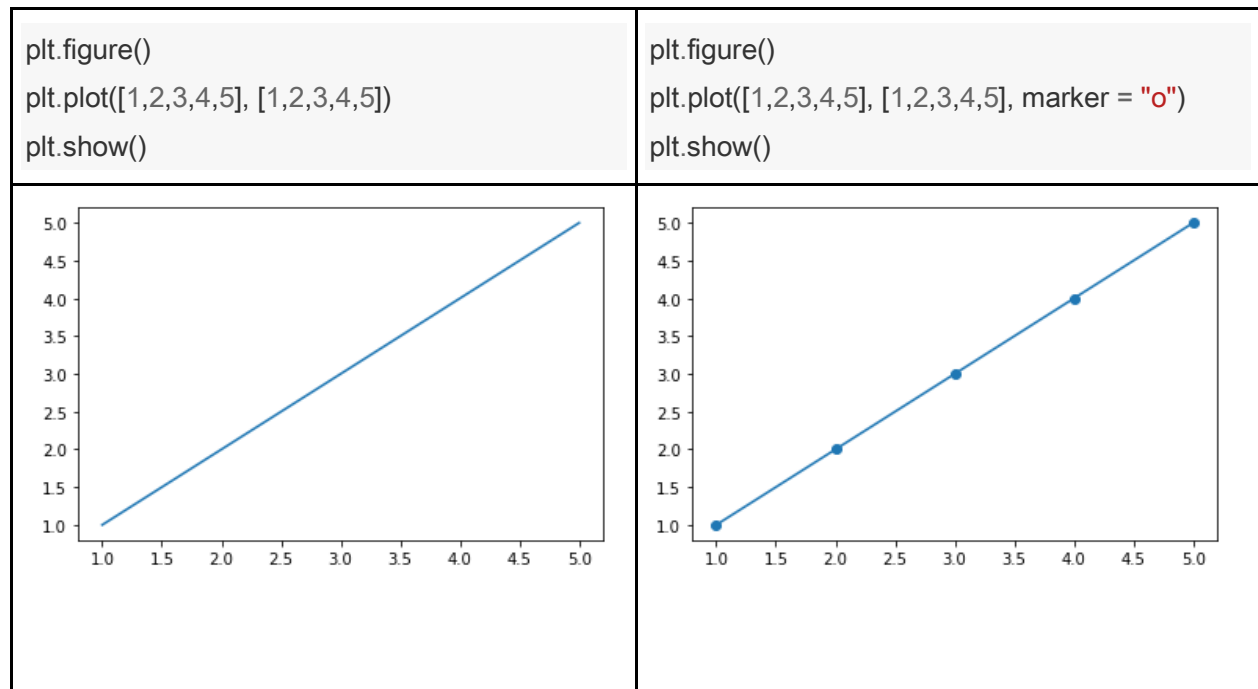
4.2 Matplotlib 의 기본 형태

기본적인 형태는 다음과 같다. plt.figure()은 figure, 즉 그래프를 표현할 액자를 먼저 만드는 것이고, plt.show 는 figure 를 출력하는 것이다. 이 때, figure 에 대한 특별한 설정을 해주지 않는다면, plt.figure()은 생략해도 무방하다.

```
plt.figure()
plt.show()
<Figure size 432x288 with 0 Axes>
```

4.3 선 그래프(Line Chart) 만들기

(1,1) , (2,2), (3,3), (4,4), (5,5) 의 데이터를 넣어서 그래프를 그려 볼 것이다. 선 그래프를 그리는 함수는 plt.plot ()이며, plt.plot([x 축 데이터], [y 축 데이터])의 형태로 사용할 수 있다.



그래프에서 찍은 점을 표현하고 싶다면, marker 인자를 추가해주면 된다. marker 는 선 그래프와 산점도에서 사용되는데, 기호에 따라 다양한 모양을 사용할 수 있다. 사용 가능한 marker 의 표시방법과 그 종류는 다음과 같다.

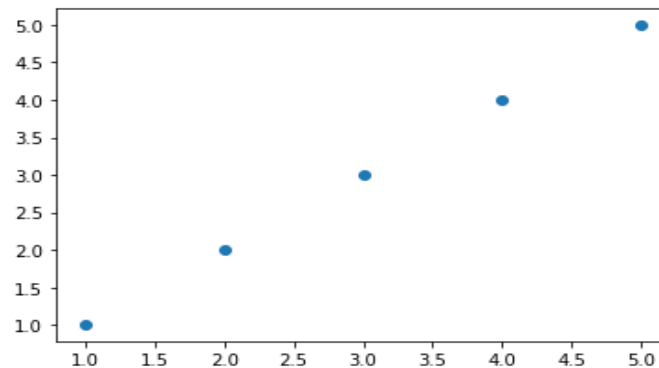
marker	symbol	marker	symbol
"."	.	"8"	8
","	,	"s"	s
"o"	o	"p"	p
"v"	v	"P"	P
"^"	^	"x"	x
"<"	<	"h"	h
">"	>	"H"	H
"1"	1	"+"	+
"2"	2	"x"	x
"3"	3	"X"	X
"4"	4	"D"	D
" "		"d"	d

https://matplotlib.org/api/markers_api.html

4.3 산점도(Scatter Plot) 만들기

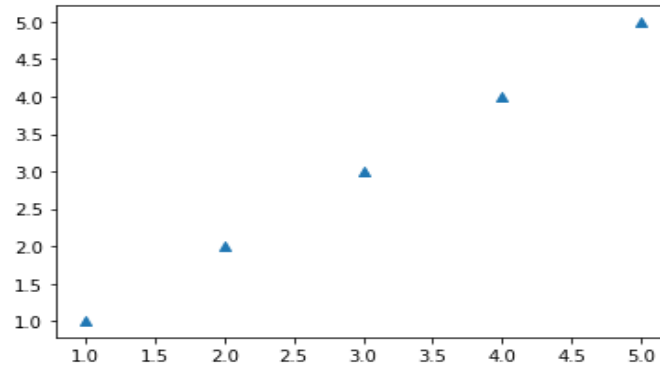
산점도를 그리는 함수는 plt.scatter()이며, plt.scatter([x 축 데이터], [y 축 데이터])의 형태로 사용한다

```
plt.figure()
plt.scatter([1,2,3,4,5], [1,2,3,4,5])
plt.show()
```



산점도 또한, marker 인자를 통해, 그래프에 나타나는 기호의 모양을 원하는 대로 설정할 수 있다.

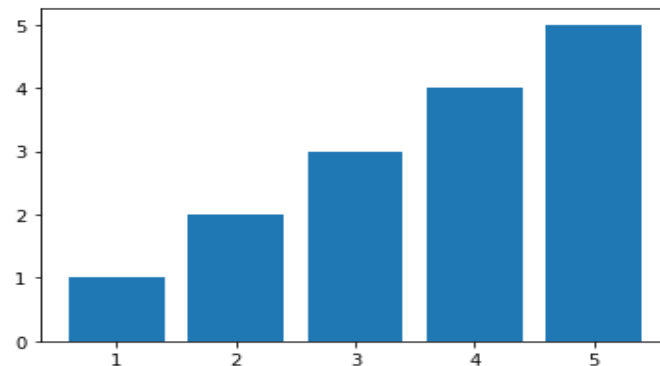
```
plt.figure()
plt.scatter([1,2,3,4,5], [1,2,3,4,5], marker = "^")
plt.show()
```



4.4 막대 그래프(Bar Chart) 만들기

막대 그래프를 그리는 함수는 `plt.bar()`이며, `plt.bar([x 축 데이터], [y 축 데이터])`의 형태로 사용한다. 각 x 축 데이터 지점에 y 축 데이터의 길이의 막대가 그려진다.

```
plt.figure()
plt.bar([1,2,3,4,5], [1,2,3,4,5])
plt.show()
```



4.5 . 파이 그래프(Pie Chart) 만들기

파이 그래프를 그리는 함수는 `plt.pie()`이며, `plt.pie([비율 데이터])`의 형태로 사용한다. 다음과 같이 입력하면, 데이터 비율이 1:2:3 인 파이 그래프가 그려진다.

```
plt.figure()  
plt.pie([1,2,3])  
plt.show()
```



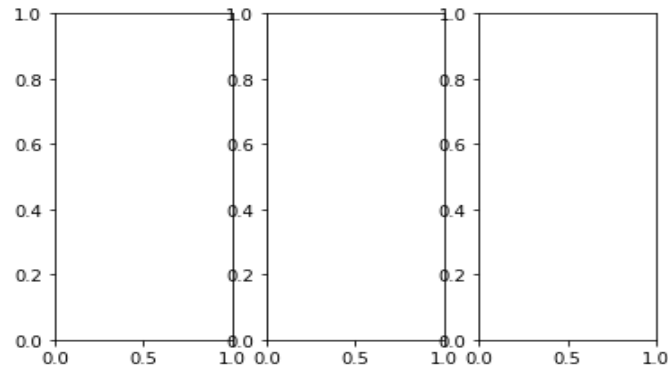
4.6 . 기타 옵션

정말 기본적인 그래프를 그리는 방법들에 대해 앞서 살펴보았다면, 더 풍성한 그래프를 완성하기 위한 추가적인 방법들을 간단히 소개하려 한다.

4.6.1 여러 개의 그래프 그리기

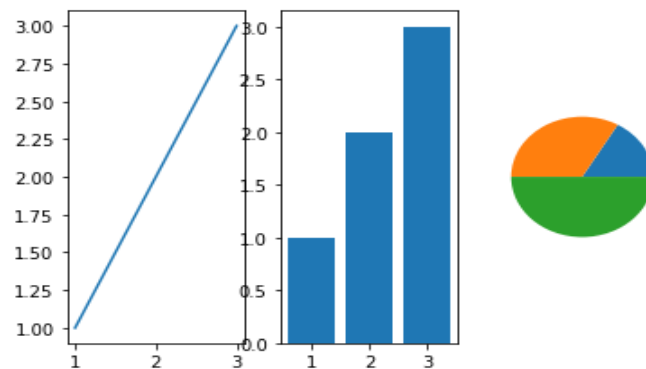
`plt.subplots()` 함수를 이용하면, 한 번에 여러 개의 그래프를 그릴 수가 있다. 이 함수는 `plt.subplots(행, 열)`의 형태로 사용된다. `plt.subplots()`는 2 개의 변수를 반환하는데, 하나는 전체 액자인 `figure`에 대한 변수고, 다른 하나는 액자 내 여러 개의 액자에 대한 리스트다. 따라서 `plt.subplots(1,3)`을 적용하면, 1 행 3 열로 액자들이 그려진다.

```
fig, ax = plt.subplots(1, 3)
```



각 액자에 그래프를 그리려면, `plt.plot()`이 아닌, `ax[i].plot()`을 사용해야 한다. `ax` 는 리스트로, `ax[0]`, `ax[1]` 등 에 각 액자를 하나씩 담고 있다.

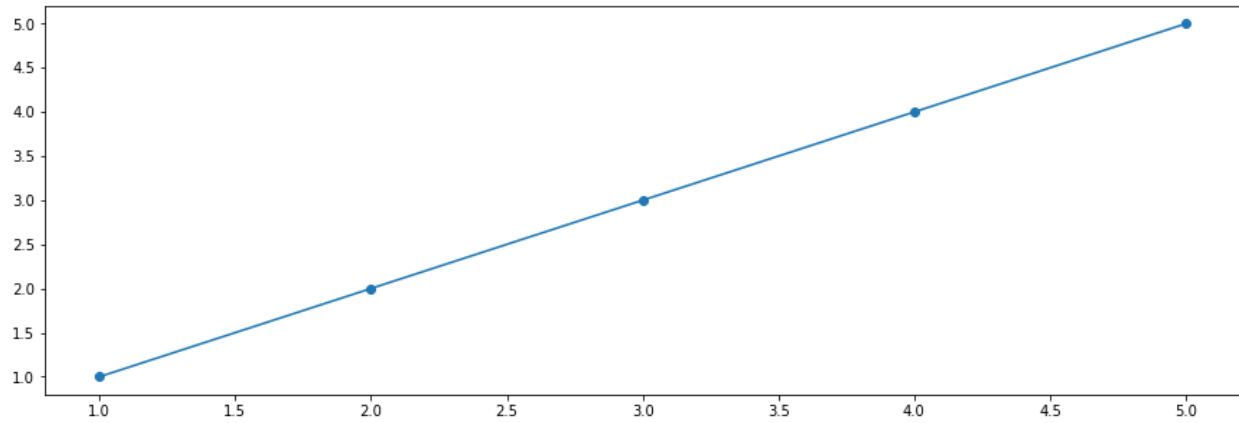
```
fig, ax = plt.subplots(1, 3)
ax[0].plot([1,2,3],[1,2,3])
ax[1].bar([1,2,3],[1,2,3])
ax[2].pie([1,2,3])
plt.show()
```



4.6.2 그래프 크기 변경

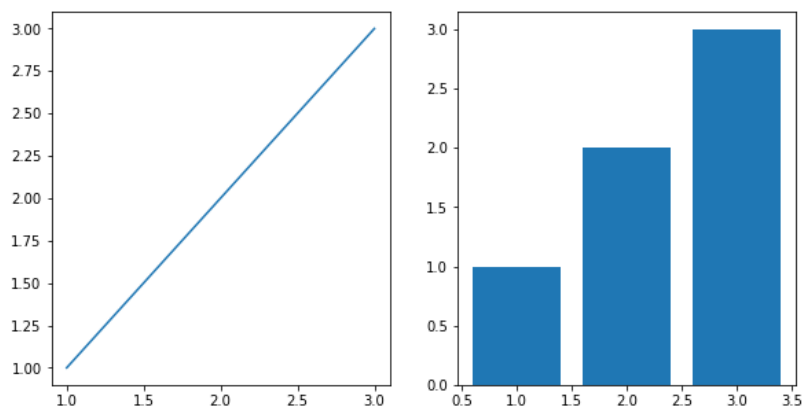
그래프 크기를 변경하려면, `plt.figure()` 함수 안에 `figsize = (가로 길이, 세로 길이)` 인자를 추가해주면 된다. 가로가 15 세로가 5 인 액자를 만들고 싶다면 다음과 같이 하면 된다.

```
plt.figure(figsize = (15, 5))
plt.plot([1,2,3,4,5], [1,2,3,4,5], marker = "o")
plt.show()
```



이는 여러 개의 그래프를 그릴 때도 해당이 된다. `plt.subplots()` 함수 안에 `figsize` 인자를 추가해주면, 그래프의 크기를 설정해줄 수 있다.

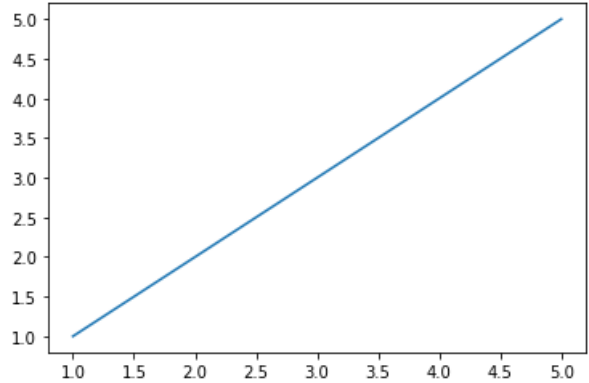
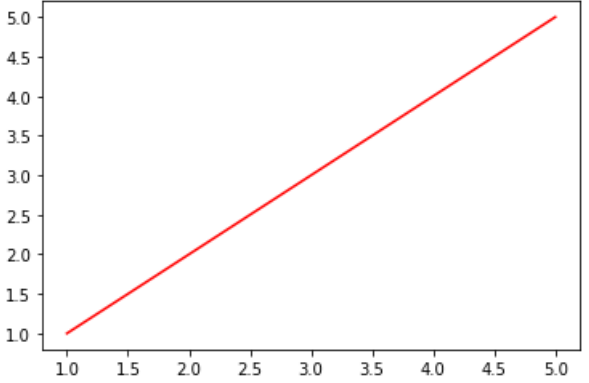
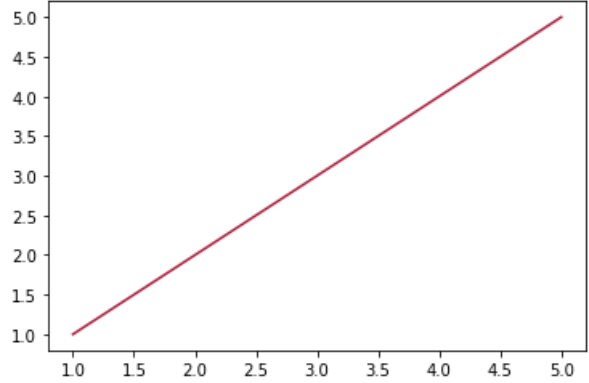
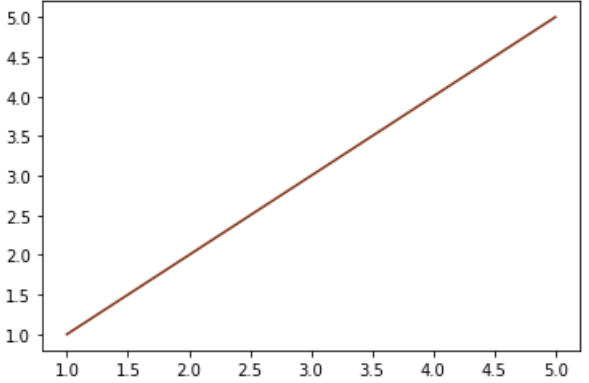
```
fig, ax = plt.subplots(1, 3, figsize = (15, 5))
ax[0].plot([1,2,3],[1,2,3])
ax[1].bar([1,2,3],[1,2,3])
ax[2].pie([1,2,3])
plt.show()
```



4.6.3 그래프에 색깔 입히기

그리려는 그래프의 함수 안에 color 인자를 추가해주면, 그래프에 원하는 색을 입힐 수 있다.

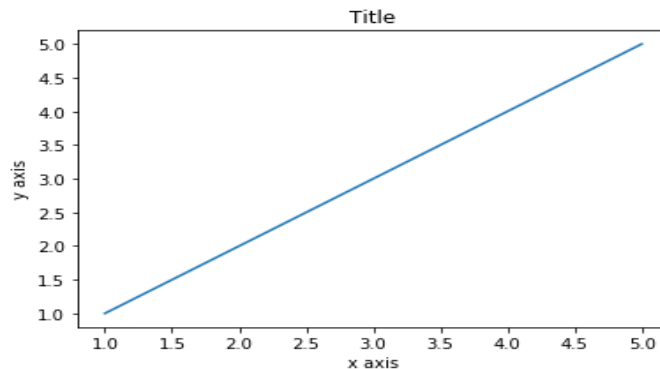
color 인자의 값은 다음과 같은 방법으로 설정해 줄 수 있다

<p>-matplotlib 에 정의된 색상 사용</p> <pre>plt.figure() plt.plot([1,2,3,4,5], [1,2,3,4,5], color = "C0") plt.show()</pre> 	<p>-색 이름을 직접 입력</p> <pre>plt.figure() plt.plot([1,2,3,4,5], [1,2,3,4,5], color = "red") plt.show()</pre> 
<p>-16 진수 표기법 사용</p> <pre>plt.figure() plt.plot([1,2,3,4,5], [1,2,3,4,5], color = "#aa1f3c") plt.show()</pre> 	<p>-0~1 로 정규화된 RGB 사용</p> <pre>plt.figure() plt.plot([1,2,3,4,5], [1,2,3,4,5], color = (0.5, 0.2, 0.1)) plt.show()</pre> 

4.6.4 라벨링 하기

그래프 제목은 `plt.title()`, 각 축의 이름은 `plt.xlabel()`, `plt.ylabel()`을 이용해 설정해 줄 수 있다. 제목은 "Title", x 축의 이름은 "x axis", y 축의 이름은 "y axis"을 지정해주면, 다음과 같은 결과가 나온다.

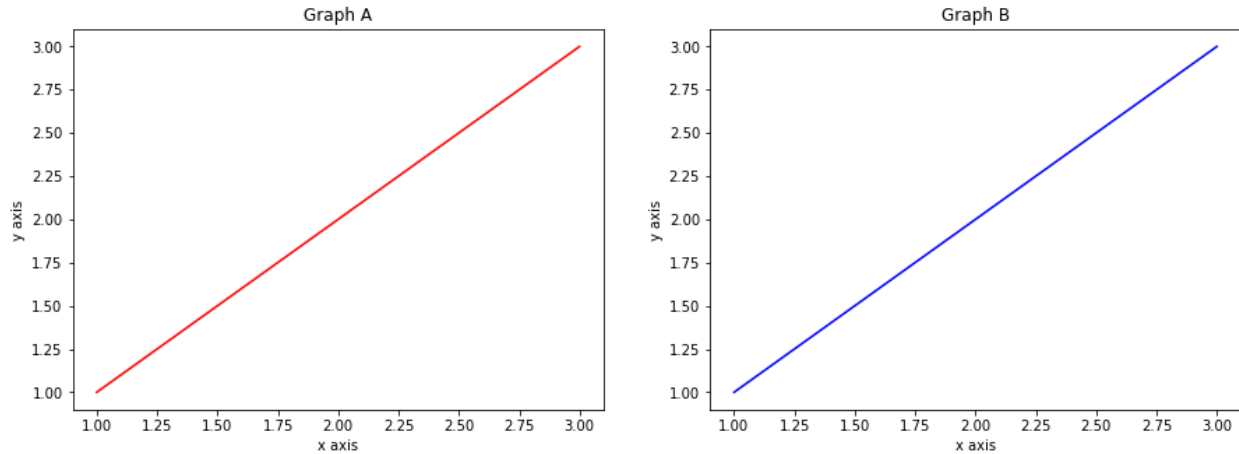
```
plt.figure()
plt.plot([1,2,3,4,5], [1,2,3,4,5])
plt.title("Title")
plt.xlabel("x axis")
plt.ylabel("y axis")
plt.show()
```



다음과 같이 여러 개의 그래프에 각각 설정할 수도 있다.

```
fig, ax = plt.subplots(1, 2, figsize = (15, 5))
ax[0].plot([1,2,3],[1,2,3], color = "red")
ax[0].set_title("Graph A")
ax[0].set_xlabel("x axis")
ax[0].set_ylabel("y axis")

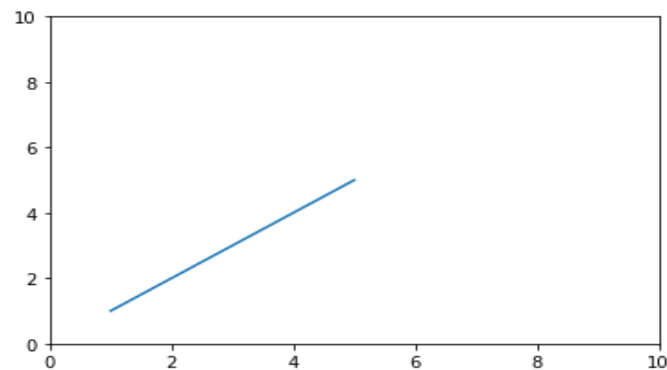
ax[1].plot([1,2,3],[1,2,3], color = "blue")
ax[1].set_title("Graph B")
ax[1].set_xlabel("x axis")
ax[1].set_ylabel("y axis")
plt.show()
```



4.6.5 축의 범위 바꾸기

`plt.xlim()`과 `plt.ylim()`을 이용하면, 각 축의 범위를 직접 설정할 수 있다. `plt.xlim([보여줄 최솟값, 보여줄 최댓값])`의 형태로 사용하면 된다.

```
plt.figure()
plt.plot([1,2,3,4,5], [1,2,3,4,5])
plt.xlim([0, 10])
plt.ylim([0, 10])
plt.show()
```

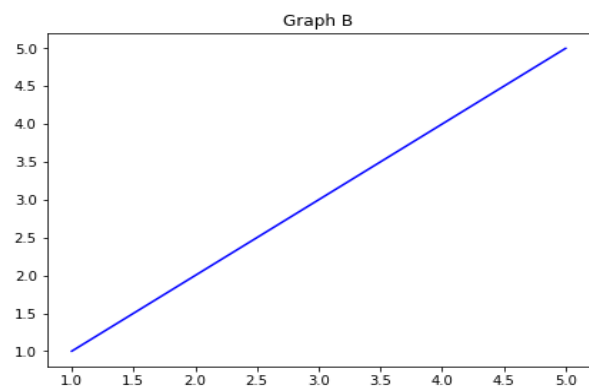
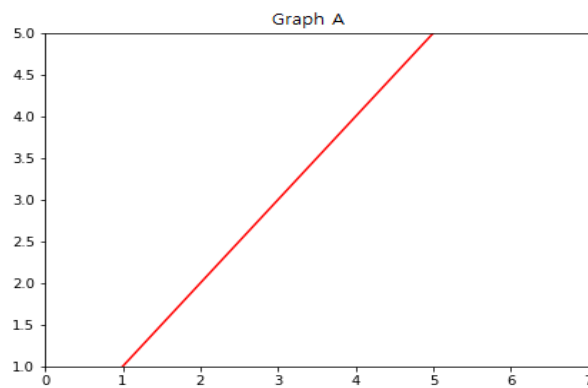


이 또한 여러 개의 그래프에 각각 설정해줄 수 있다.

```
ax[0].plot([1,2,3,4,5],[1,2,3,4,5], color = "red")
ax[1].set_title("Graph A")
ax[0].set_xlim([0,10])
ax[0].set_ylim([0,10])

ax[1].plot([1,2,3,4,5],[1,2,3,4,5], color = "blue")
ax[1].set_title("Graph B")
ax[0].set_xlim([0,7])
ax[0].set_ylim([1,5])

plt.show()
```



4.6.6 범례 표시

범례를 표시하려면, `plt.plot`의 인자로 `label`을 설정해주고, `plt.legend()`을 적어주면 된다. 예를 들어, “Graph A”라고 범례를 표시해주고 싶다면, 다음과 같이 하면 된다.

```
plt.figure()
plt.plot([1,2,3,4,5], [1,2,3,4,5], label = "Graph A")
plt.legend()
plt.show()
```

