

# R을 활용한 Data Visualization

엑셈아카데미 <http://exem-academy.com/>

온사이트 교육 문의 [edu@ex-em.com](mailto:edu@ex-em.com)

엑셈 연구콘텐츠팀 | 김찬경

# 엑셈 연구팀 소개 (edu@ex-em.com)



## - 권건우 상무

- MIS Ph.D.
- 前 삼성 SDS DBA, TA
- 現 엑셈 아카데미 상무
- 現 DB 기술연구소 소장
- DB Deep Internals 강의 (Oracle, PostgreSQL, MySQL, Aurora)
- 엑셈아카데미 블록체인 인사이드 강의
- 엑셈아카데미 인공지능 인사이드 강의
- < Oracle, PostgreSQL, MySQL Core Architecture 1 > 책 집필
- < Oracle, PostgreSQL, MySQL Core Architecture 2 > 책 집필



## - 이근오 팀장

- 삼성카드/삼성생명/ING생명 차세대 PJT 참여
- 現 DB 기술연구소 팀장
- 엑셈아카데미 DB Deep Internals 강의 (Oracle, PostgreSQL, MySQL, Aurora)
- 엑셈아카데미 블록체인 인터널 강의
- 엑셈아카데미 인공지능 인터널 강의
- < Oracle, PostgreSQL, MySQL Core Architecture 1 > 책 집필
- < Oracle, PostgreSQL, MySQL Core Architecture 2 > 책 집필



## - 장서연 연구원

- 現 엑셈 DB기술연구소 선임 연구원
- 엑셈아카데미 DB Deep Internals 강의 (Oracle, PostgreSQL, MySQL, Aurora)
- 엑셈아카데미 오로라 첫걸음 강의
- 엑셈아카데미 블록체인 강의
- 엑셈아카데미 왕초보 SQL 강의



## - 이대덕 연구원

- 現 DB 기술연구소 선임 연구원
- 엑셈아카데미 DB Deep Internals 강의 (Oracle, PostgreSQL, MySQL, Aurora)
- 엑셈아카데미 블록체인 인터널 강의
- < Oracle, PostgreSQL, MySQL Core Architecture 1 > 책 집필
- < Oracle, PostgreSQL, MySQL Core Architecture 2 > 책 집필

# 엑셈 연구팀 소개 (edu@ex-em.com)

## - 김정수 연구원



- 現 엑셈 연구콘텐츠팀 연구원
- Nike, 세노비스, 스와치그룹 BI프로젝트 참여
- 엑셈아카데미 머신러닝을 위한 파이썬 라이브러리 살펴보기 강의

## - 장호형 연구원



- 現 엑셈 DB기술연구소 선임 연구원
- 코웨이 DB암호화 프로젝트 참여
- 인천청라지구 U-City 구축사업 참여
- 마곡 U-City 구축사업 참여
- 통일부 영상편지 공정관리 시스템 구축사업 참여
- 환경부 생활환경 안전정보 제공 시스템 구축사업 참여
- 서울시설공단 도로순찰관리시스템 구축사업 참여
- 제주 스마트그리드 구축사업 참여

## - 문민음 연구원



- 現 엑셈 아카데미 연구콘텐츠팀 객원연구원
- 통계학 석사과정 재학중
- 시계열 예측분석 연구실 소속
- 학부 통계학, 통계학실험 강의조교
- 엑셈아카데미 R을 활용한 머신러닝 강의

## - 권기범 연구원



- 現 엑셈아카데미 연구콘텐츠팀 연구원
- 엑셈 대학생 아카데미 머신러닝 캠프 운영 조교
- 엑셈 대학생 아카데미 머신러닝 캠프 선형대수 기초 강의

# 엑셈 연구팀 소개 (edu@ex-em.com)



## - 김찬경 연구원

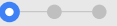
- 現 엑셈 아카데미 연구콘텐츠팀 연구원
  - 엑셈 아카데미
- "R찬 세미나, 제대로 R아보자!" 강의
- 엑셈 아카데미
- "R을 활용한 데이터 시각화" 강의



# 목 차

1. 역사를 통해 배우는 시각화의 중요성
2. 시각화의 정의 및 분류
3. 시각화 차트 선택과 유형
4. R 그래픽 함수를 알아보자
5. ggplot2 패키지를 알아보자
6. ggmap 패키지를 알아보자
7. 시각화 실전예제 해보기





# 1. 역사를 통해 배우는 시각화의 중요성



# 나이팅게일

영국의 간호사, 병원 · 의료제도의 개혁자

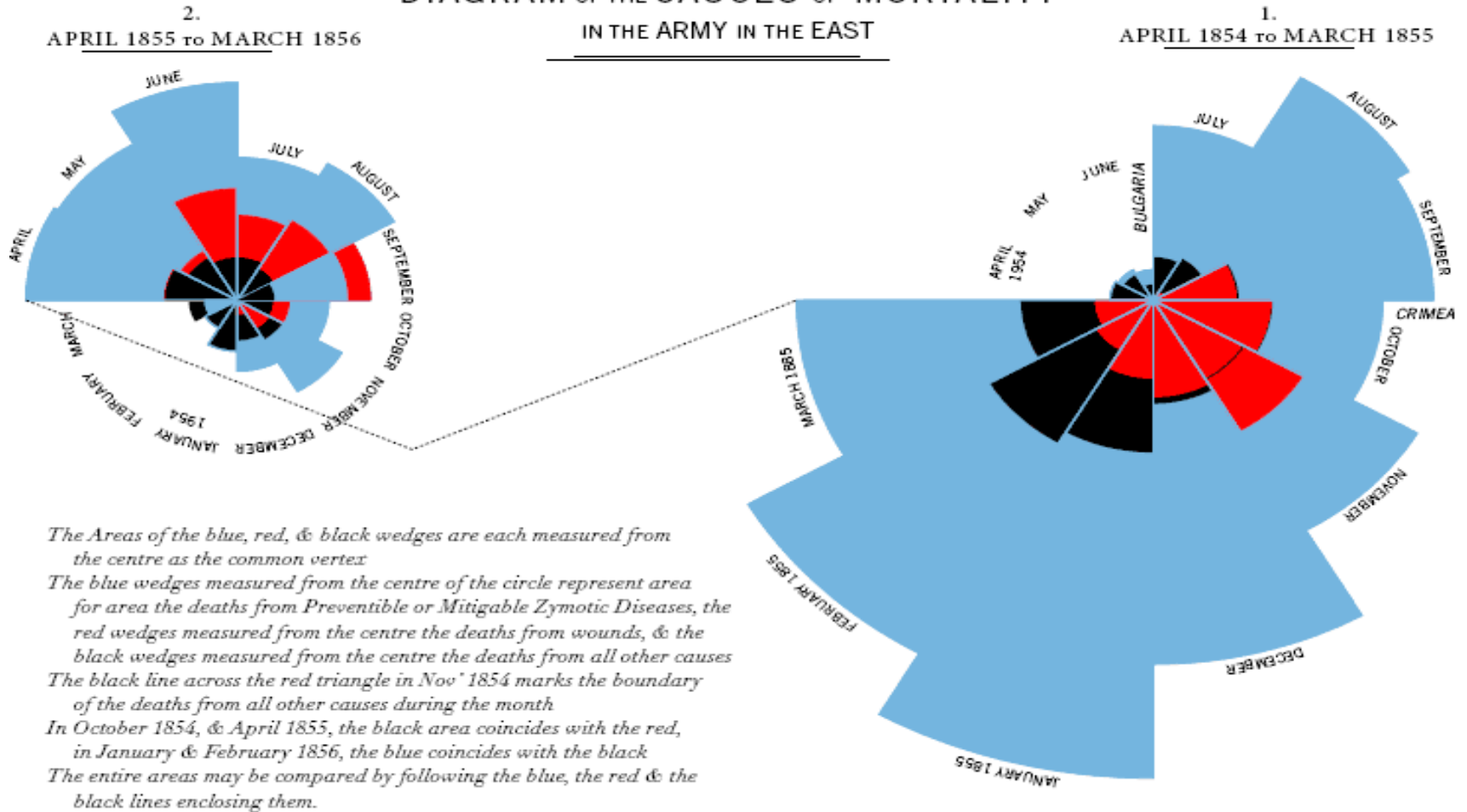




간호사가 된 후 '크림전쟁(1853-56)' 발발

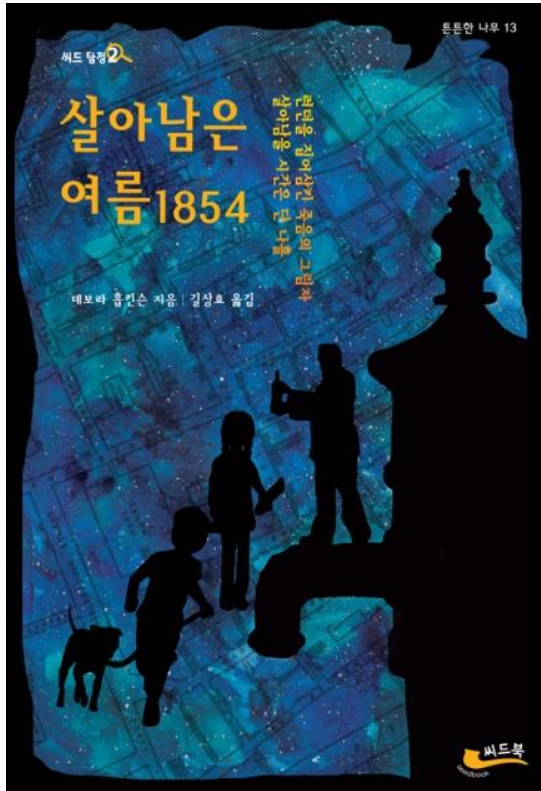
## 로즈다이어그램

### DIAGRAM OF THE CAUSES OF MORTALITY IN THE ARMY IN THE EAST



42 % -> 2%로 사망률 감소





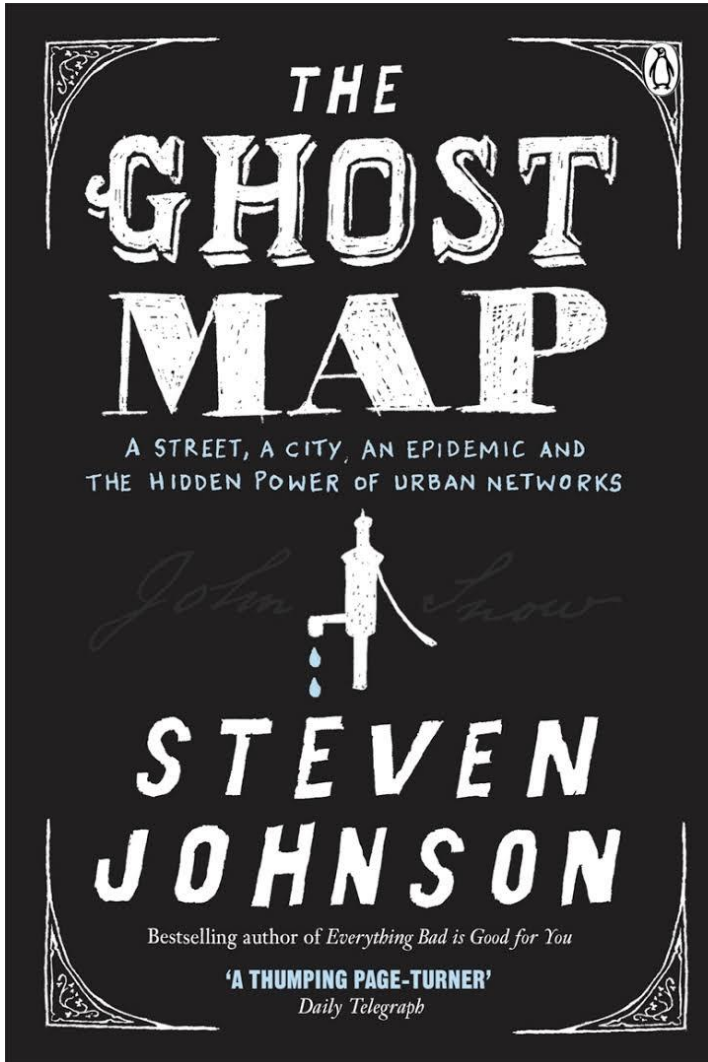
살아남은 여름 1854



그 해 늦여름 616명의 목숨을 앗아간 콜레라 사태의  
진행 과정을 재구성한 일종의 역사 소설



브로드 길 펌프 주변에 분포된 콜레라 사망자 - 스노우 박사

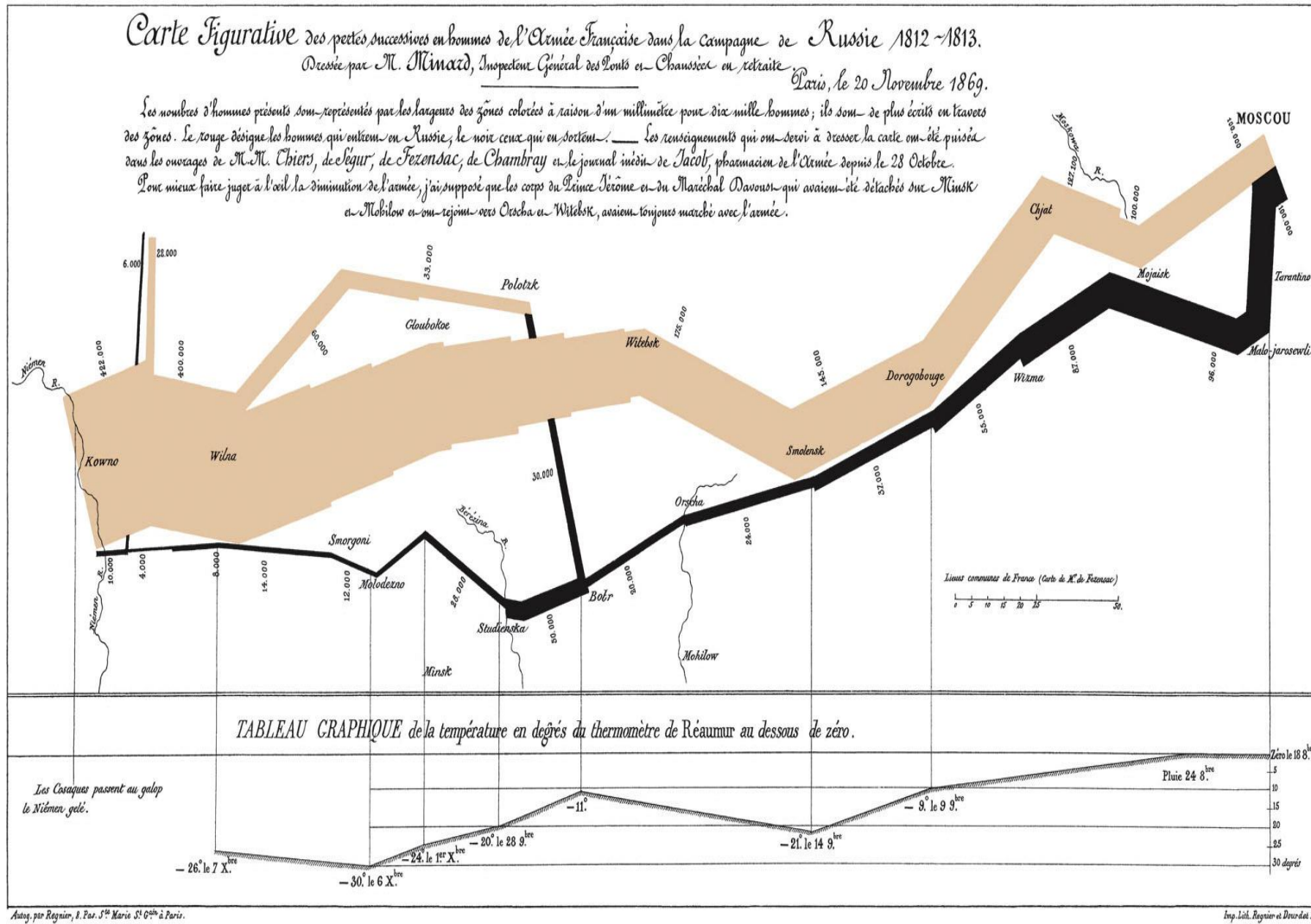


브로드 길 펌프 손잡이가 철거된 9월 8일.

과학에 근거해 시민 보호 조치를 내림으로  
공중 보건에의 새 지평을 연

‘역사적 전환점’

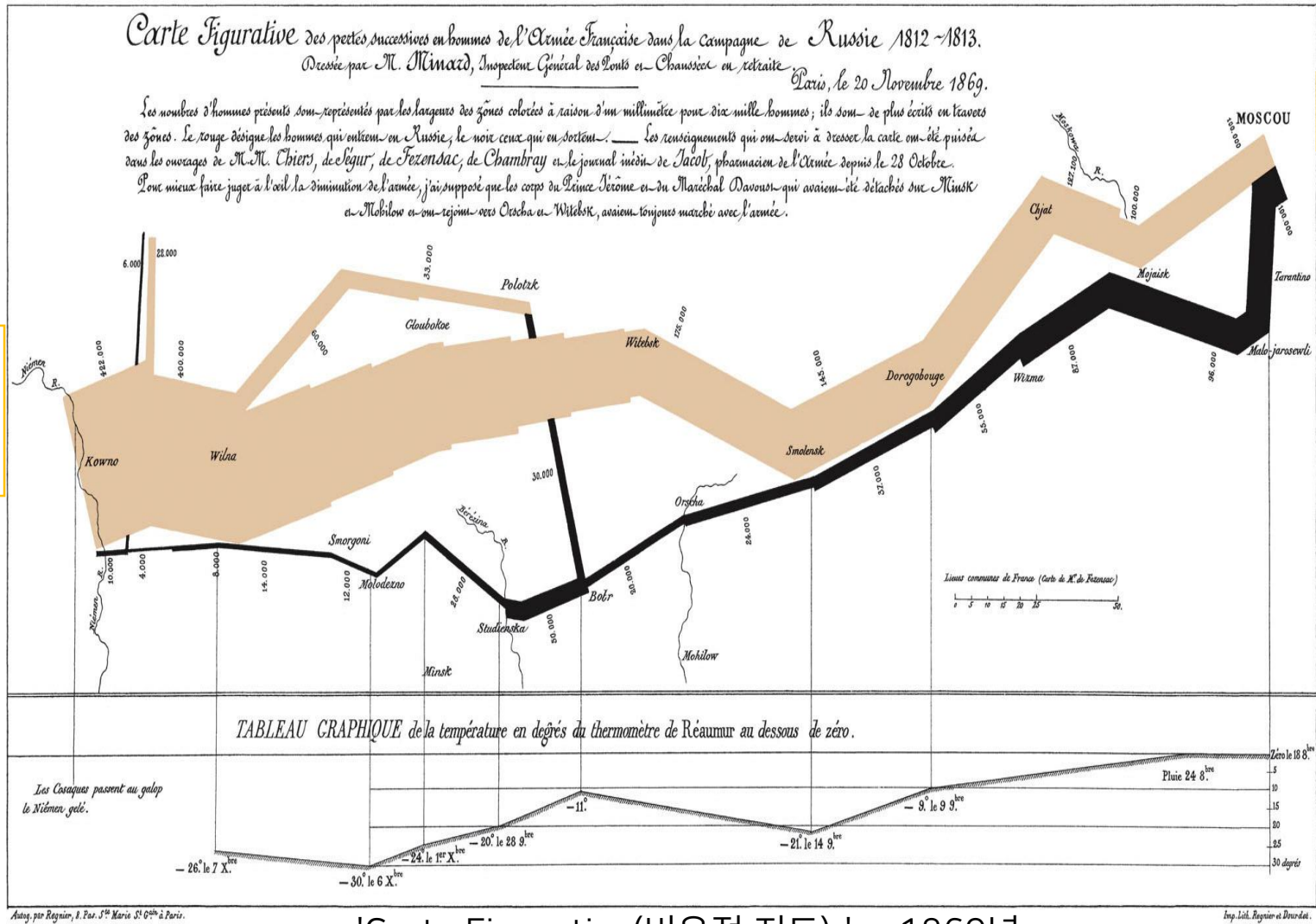
‘죽음의 지도(the ghost map)’ - 스티븐 존슨



\* 모스크바

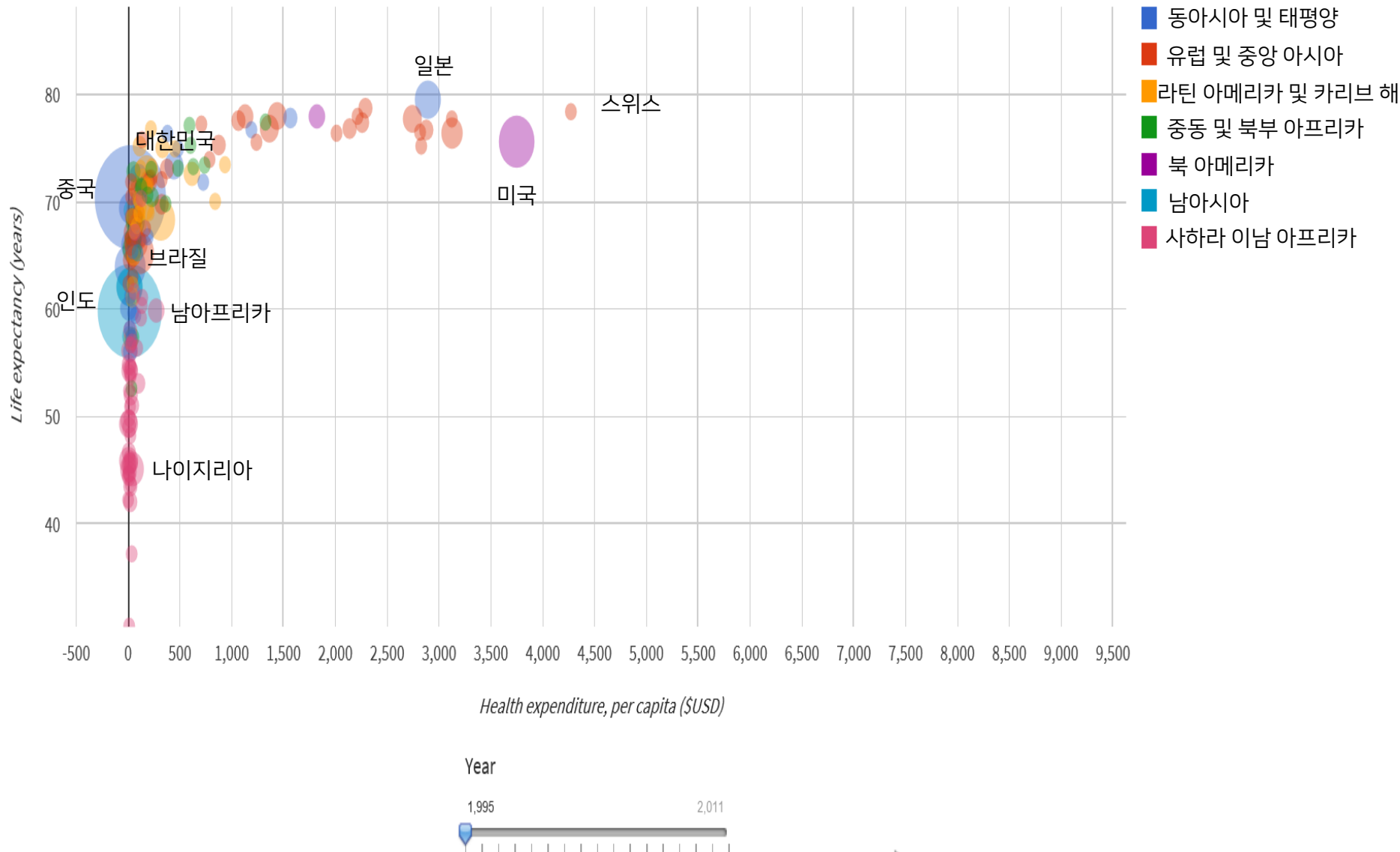
\* 후퇴 당시의 기온과 날짜 (오른쪽에서 왼쪽 순서)

\* 폴란드와 러시아의 국경쪽 네만강

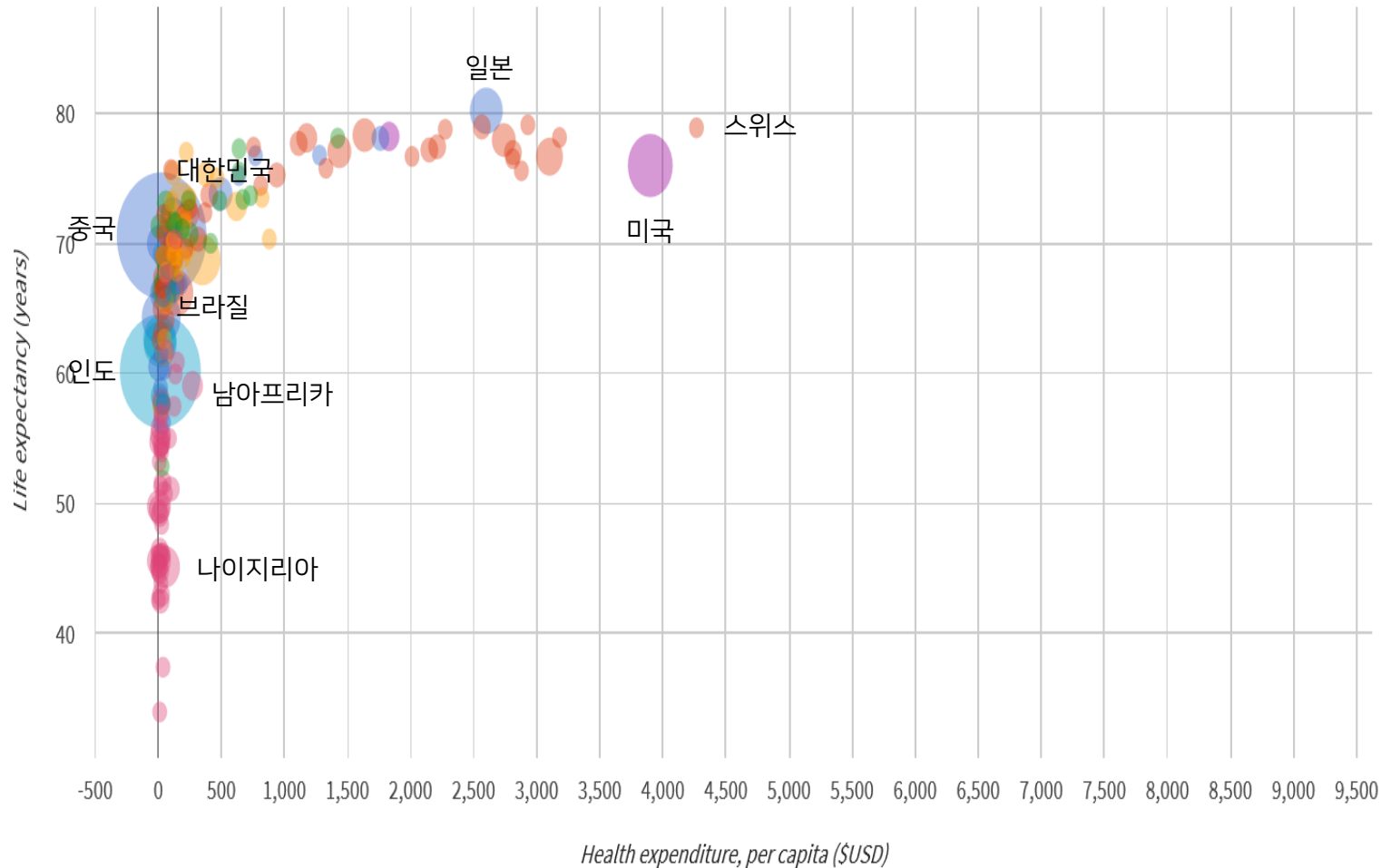


'Carte Figurative(비유적 지도)' - 1869년  
 프랑스의 도시공학자 샤를 미나르(Charles Minard)

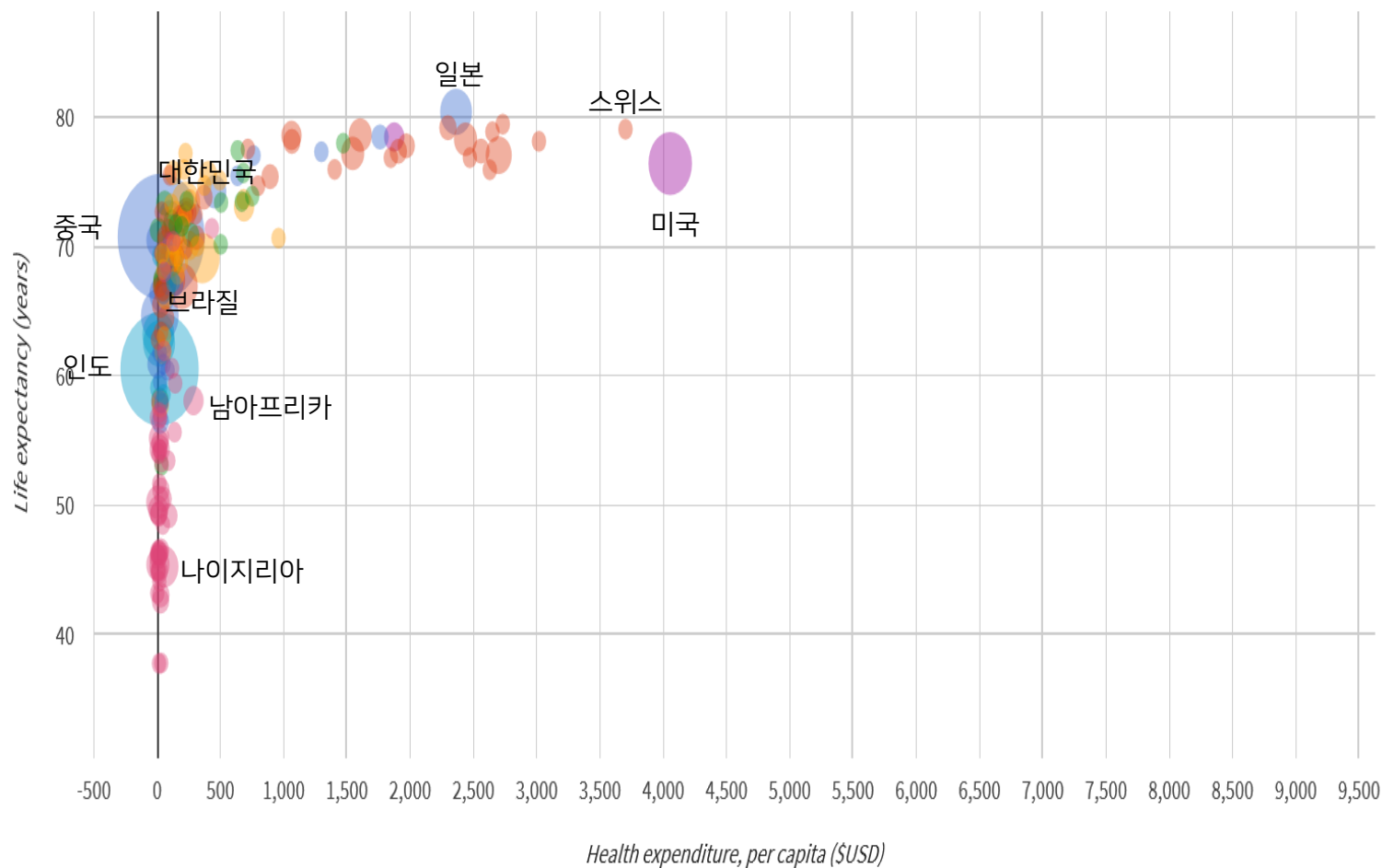
## 건강 지출 vs 기대 수명, 1995



## 건강 지출 vs 기대 수명, 1996



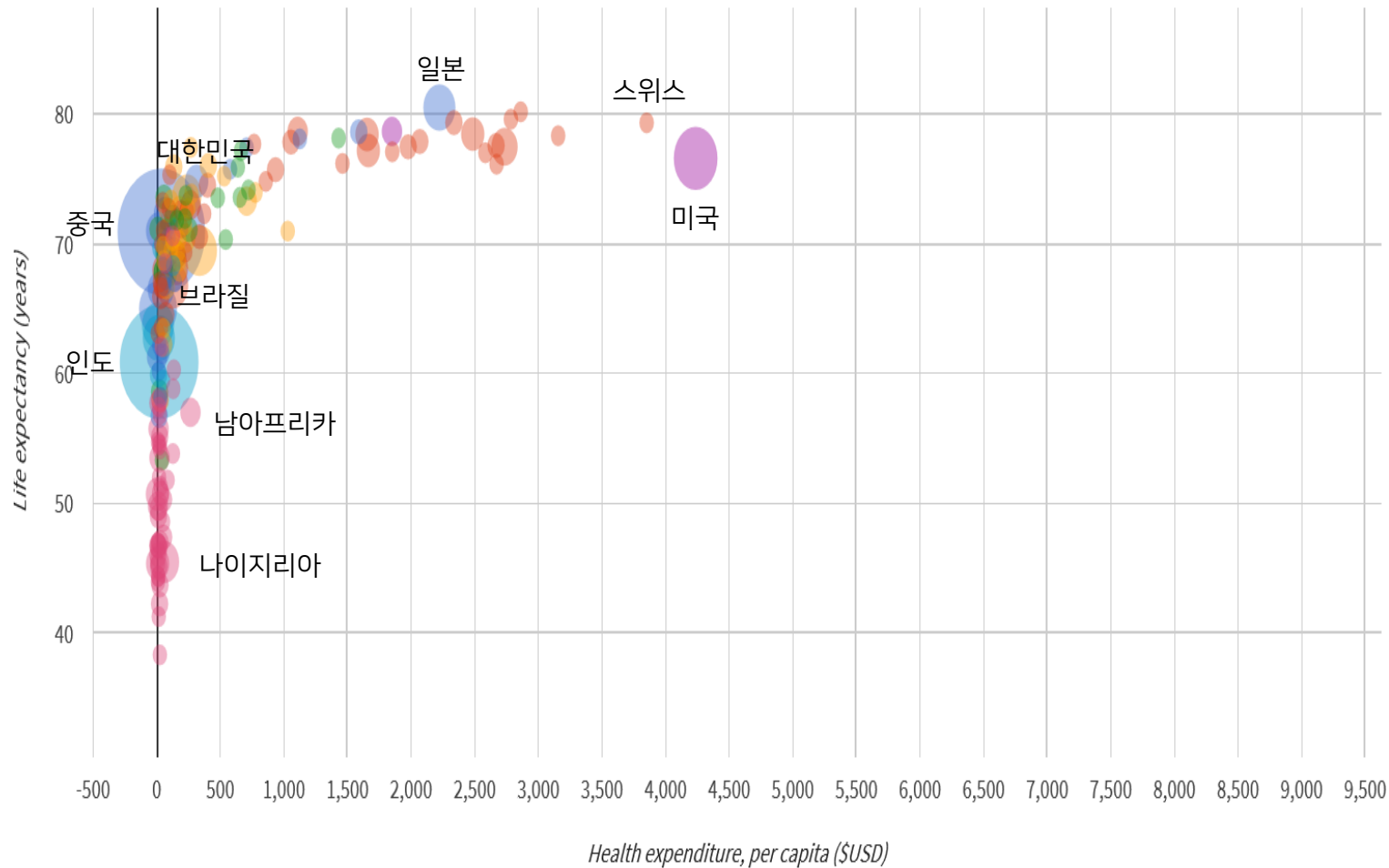
## 건강 지출 vs 기대 수명, 1997



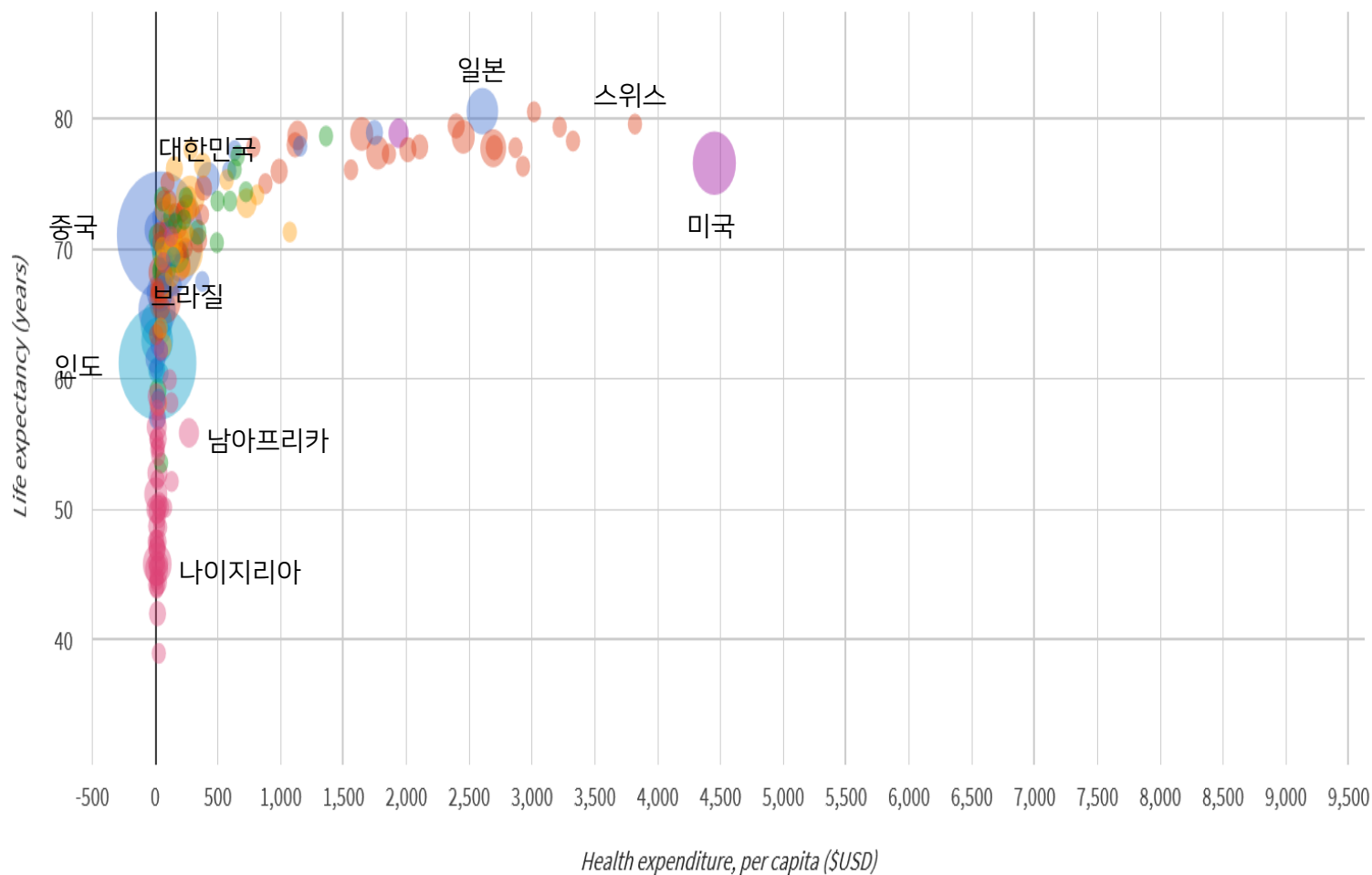
Year



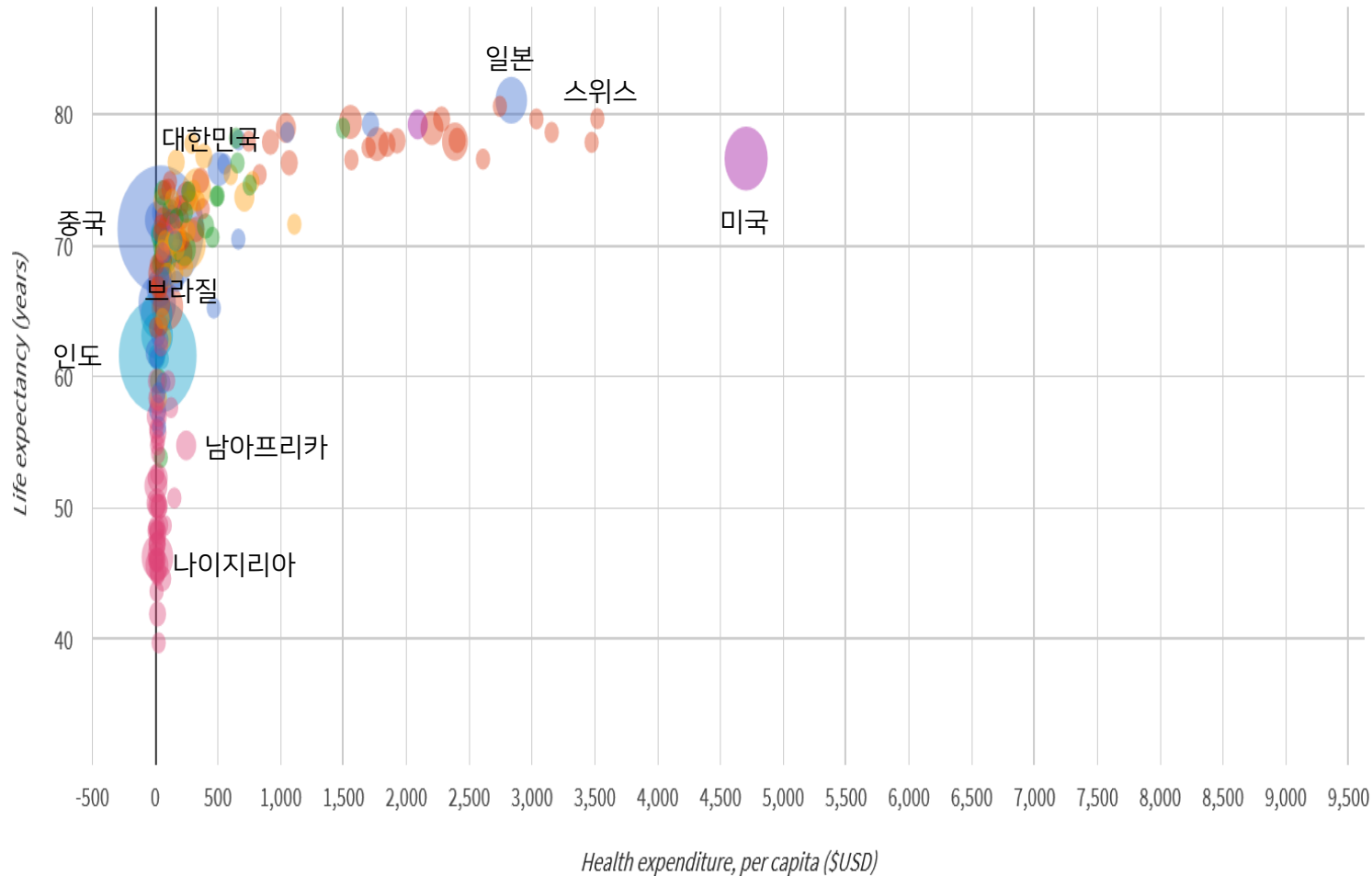
## 건강 지출 vs 기대 수명, 1998



## 건강 지출 vs 기대 수명, 1999



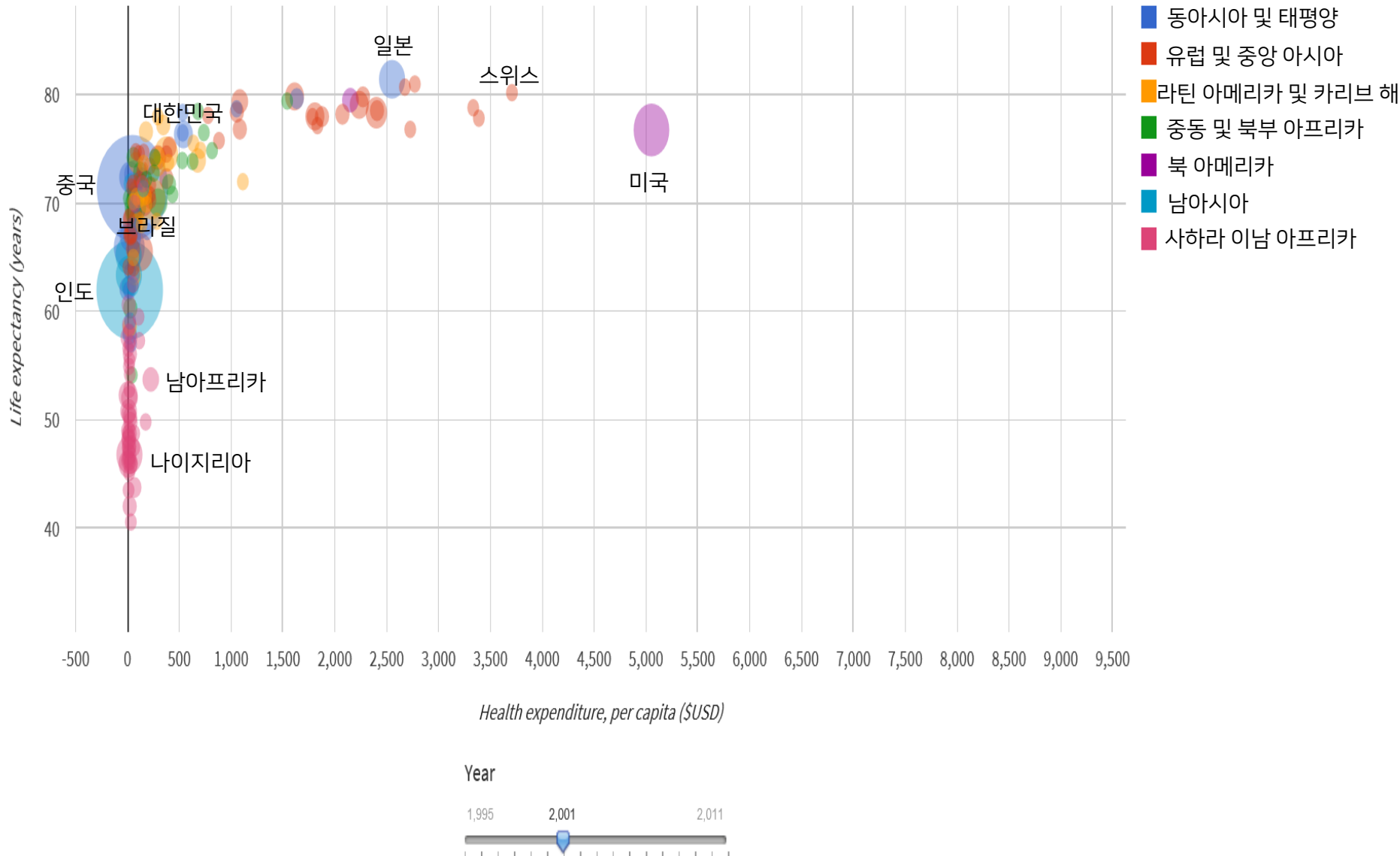
## 건강 지출 vs 기대 수명, 2000



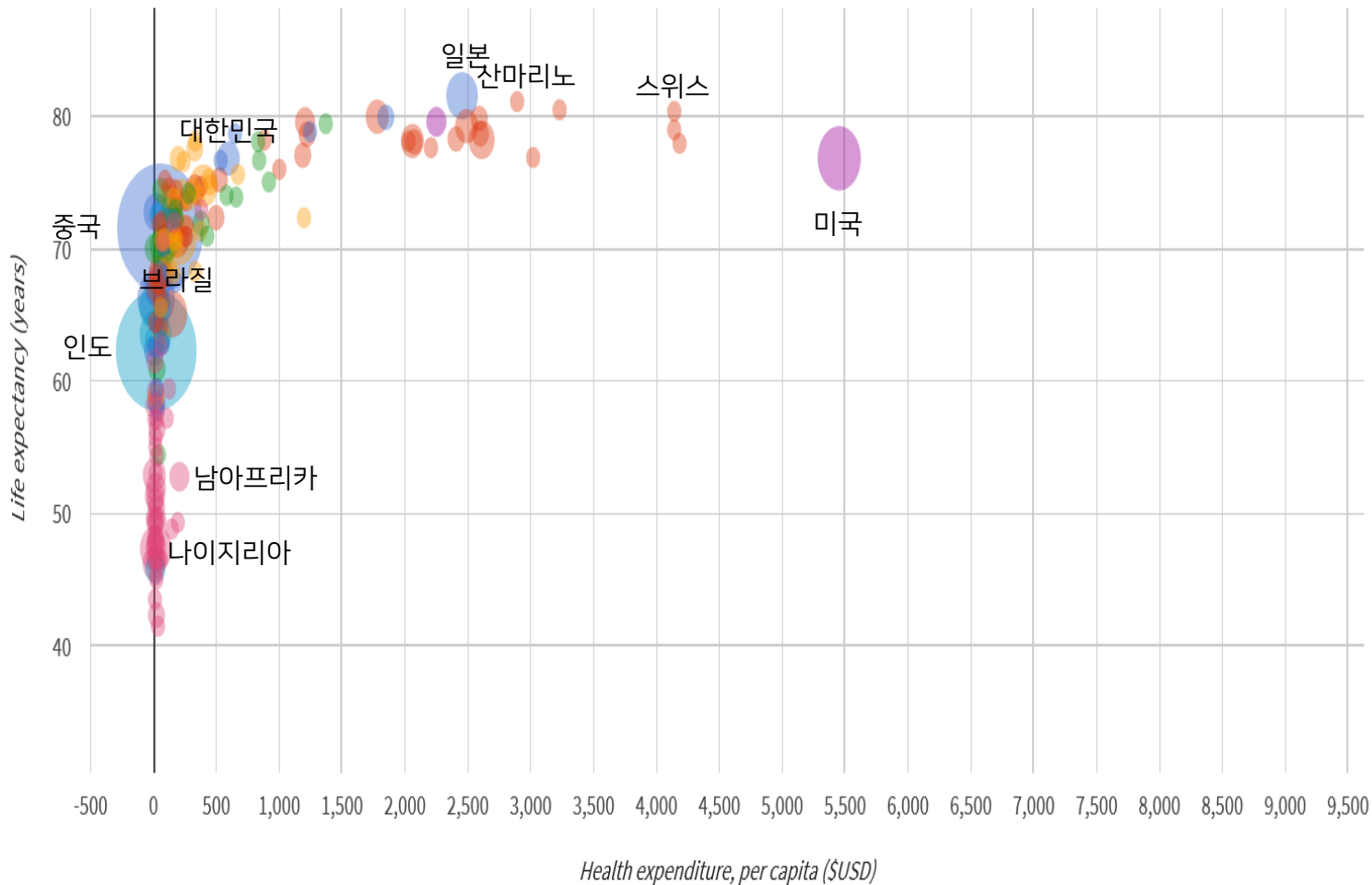
Year



## 건강 지출 vs 기대 수명, 2001



## 건강 지출 vs 기대 수명, 2002

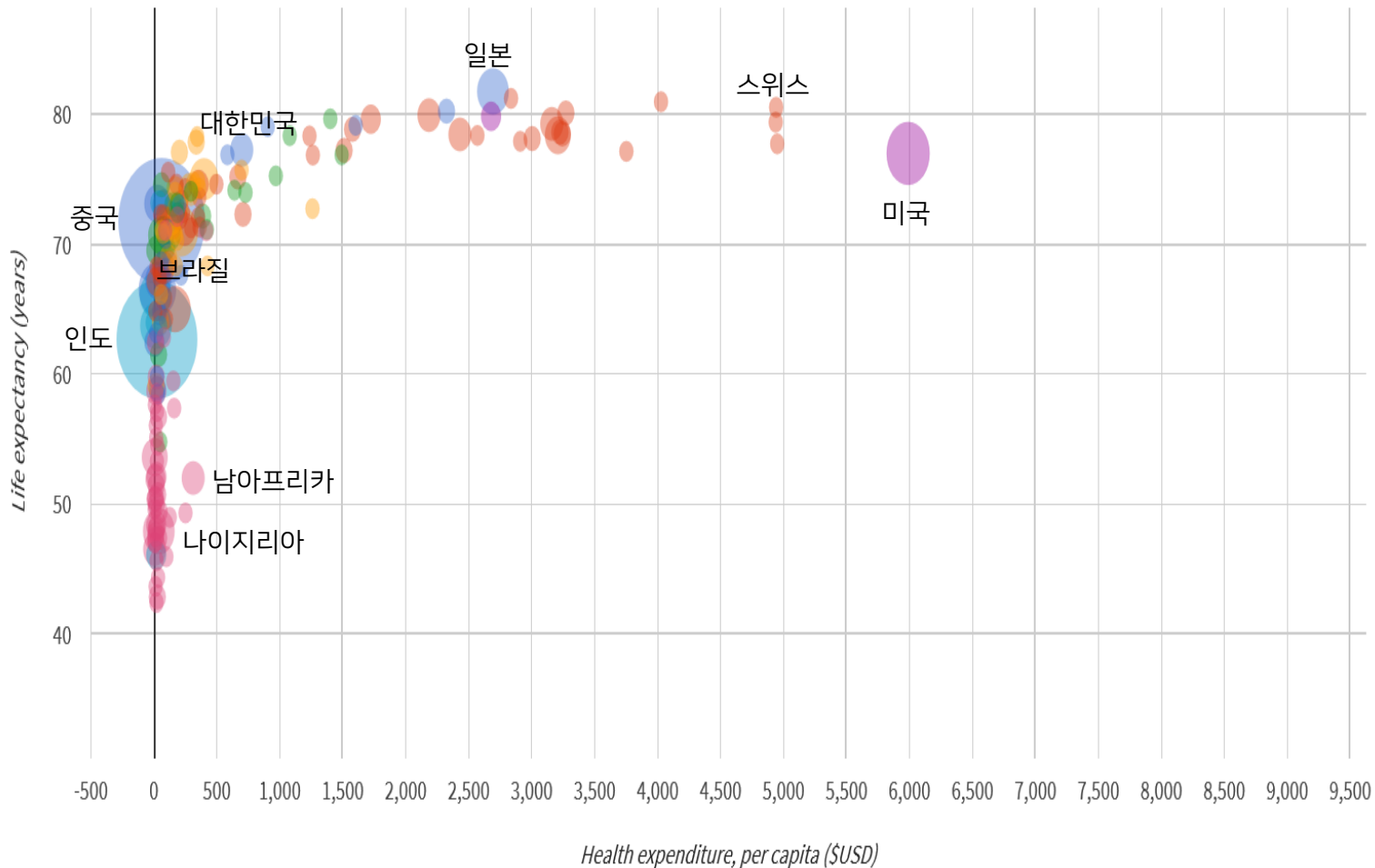


Year

1,995 2,002 2,011



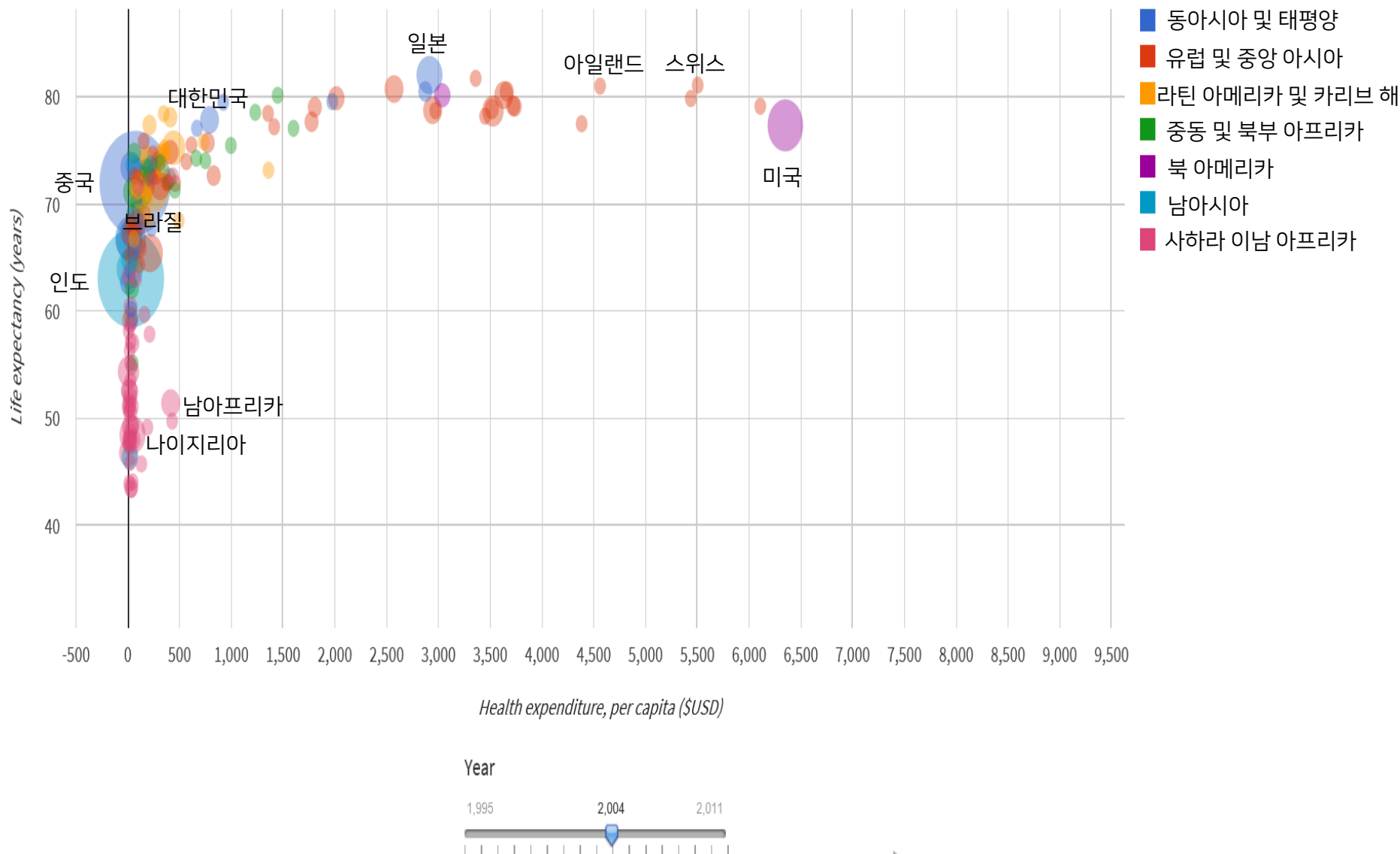
## 건강 지출 vs 기대 수명, 2003



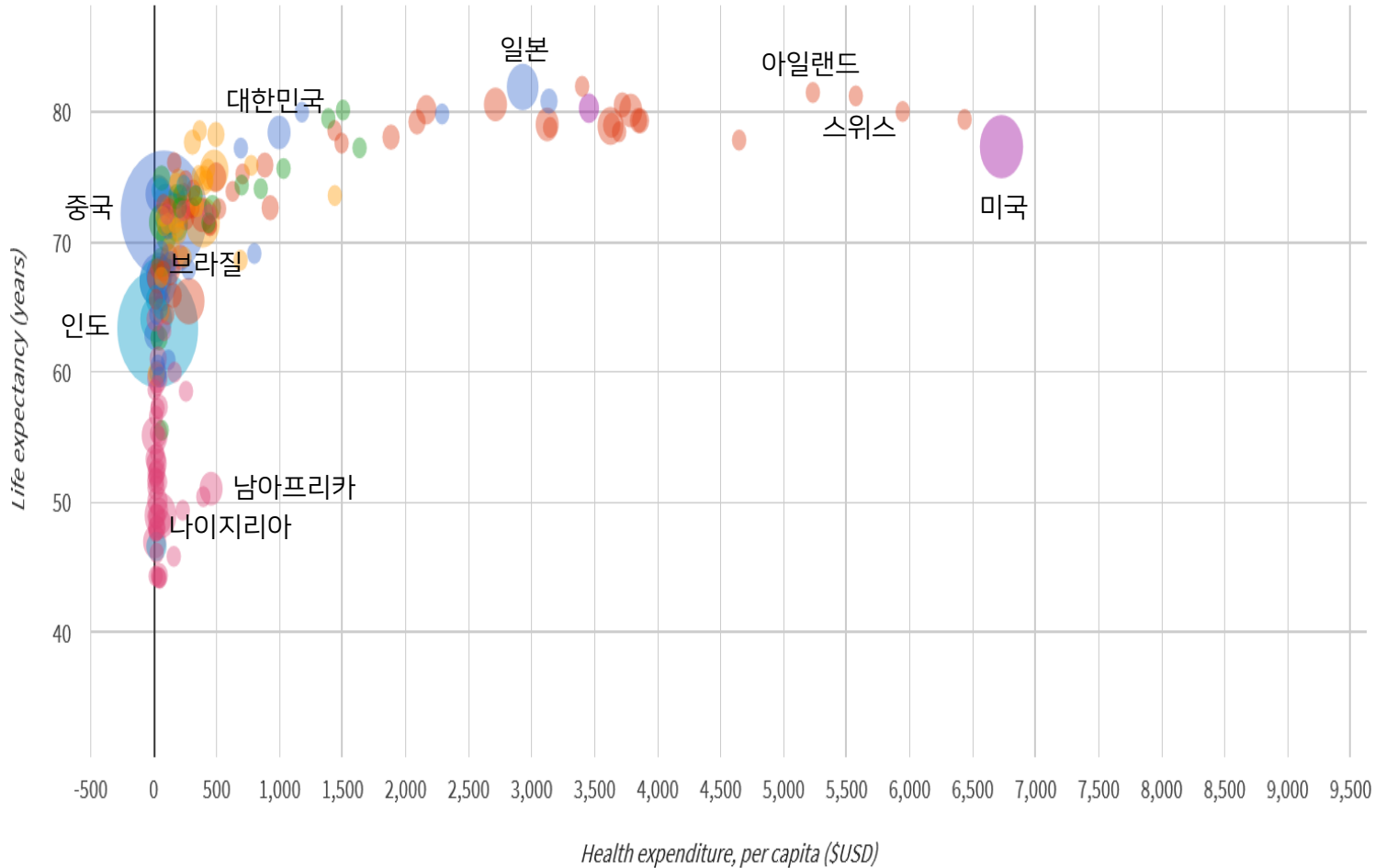
Year



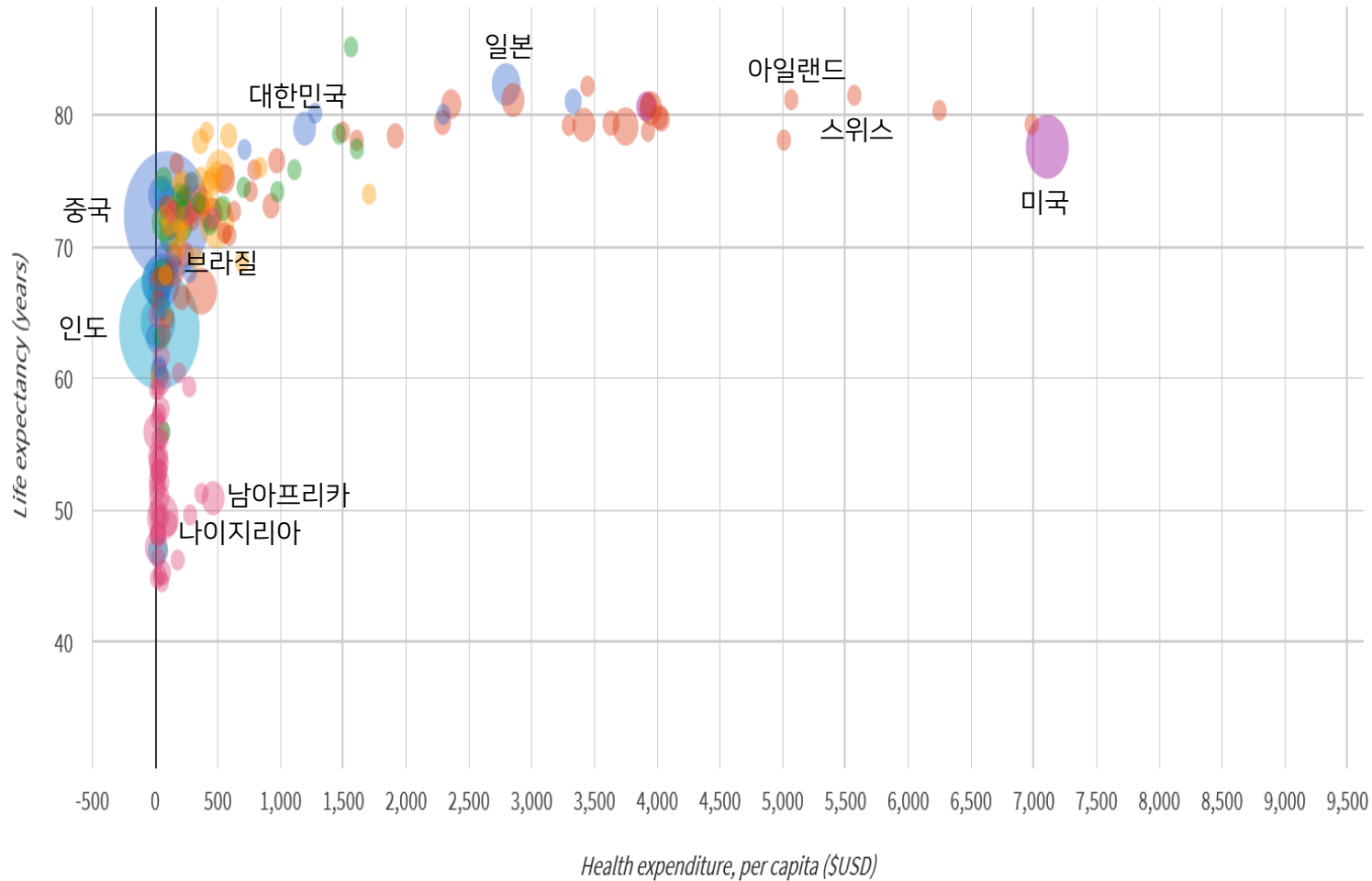
## 건강 지출 vs 기대 수명, 2004



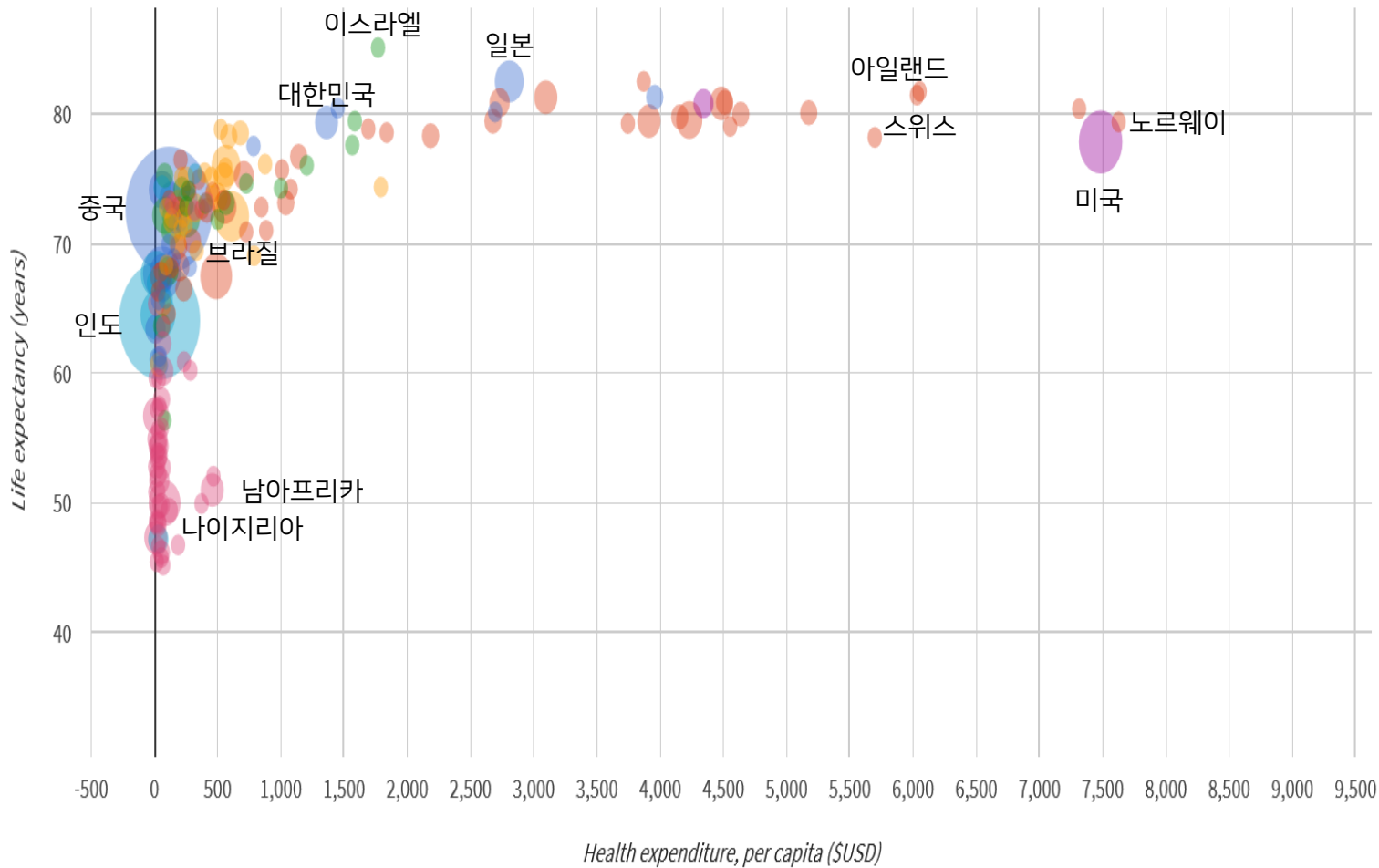
## 건강 지출 vs 기대 수명, 2005



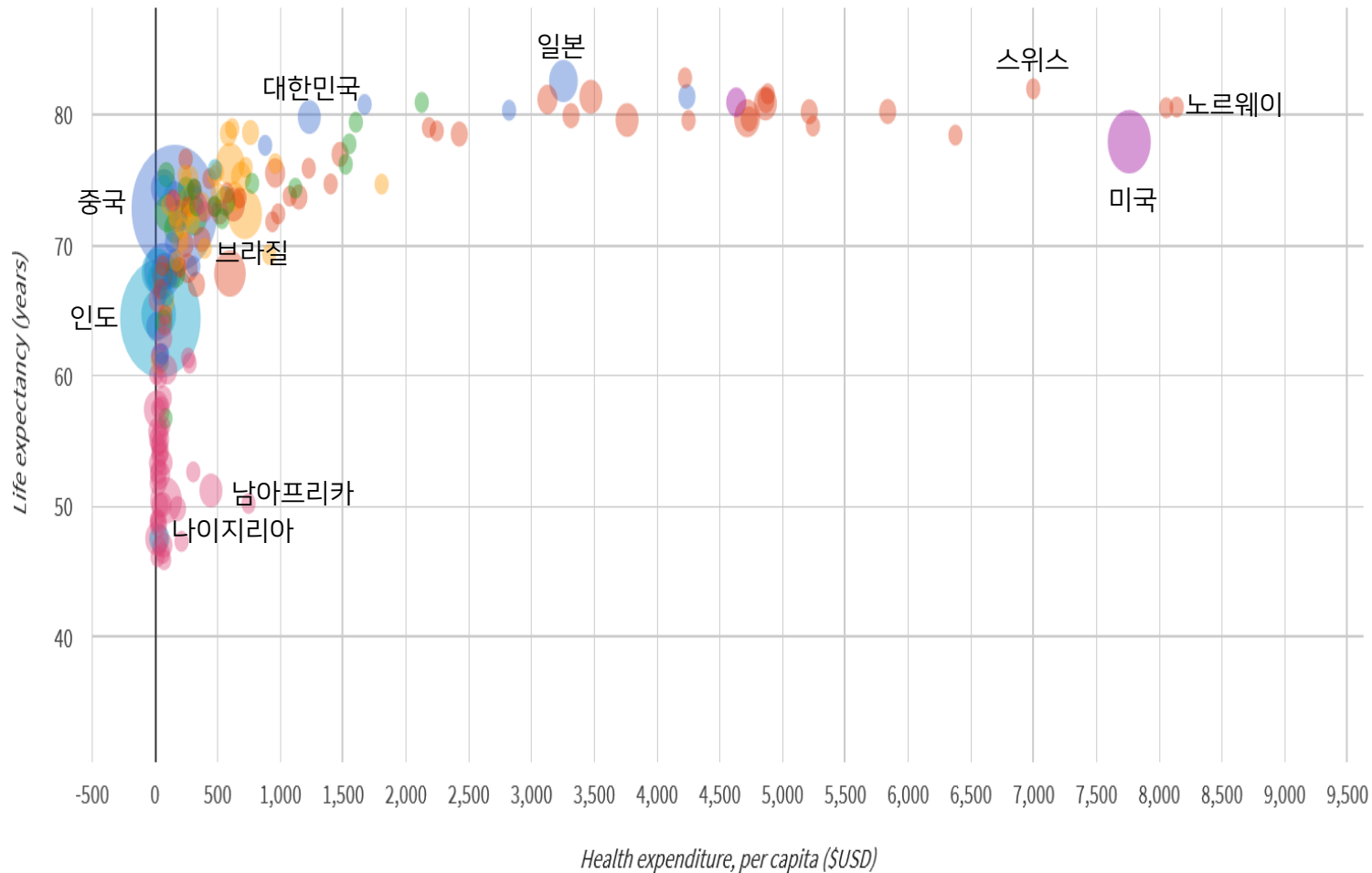
## 건강 지출 vs 기대 수명, 2006



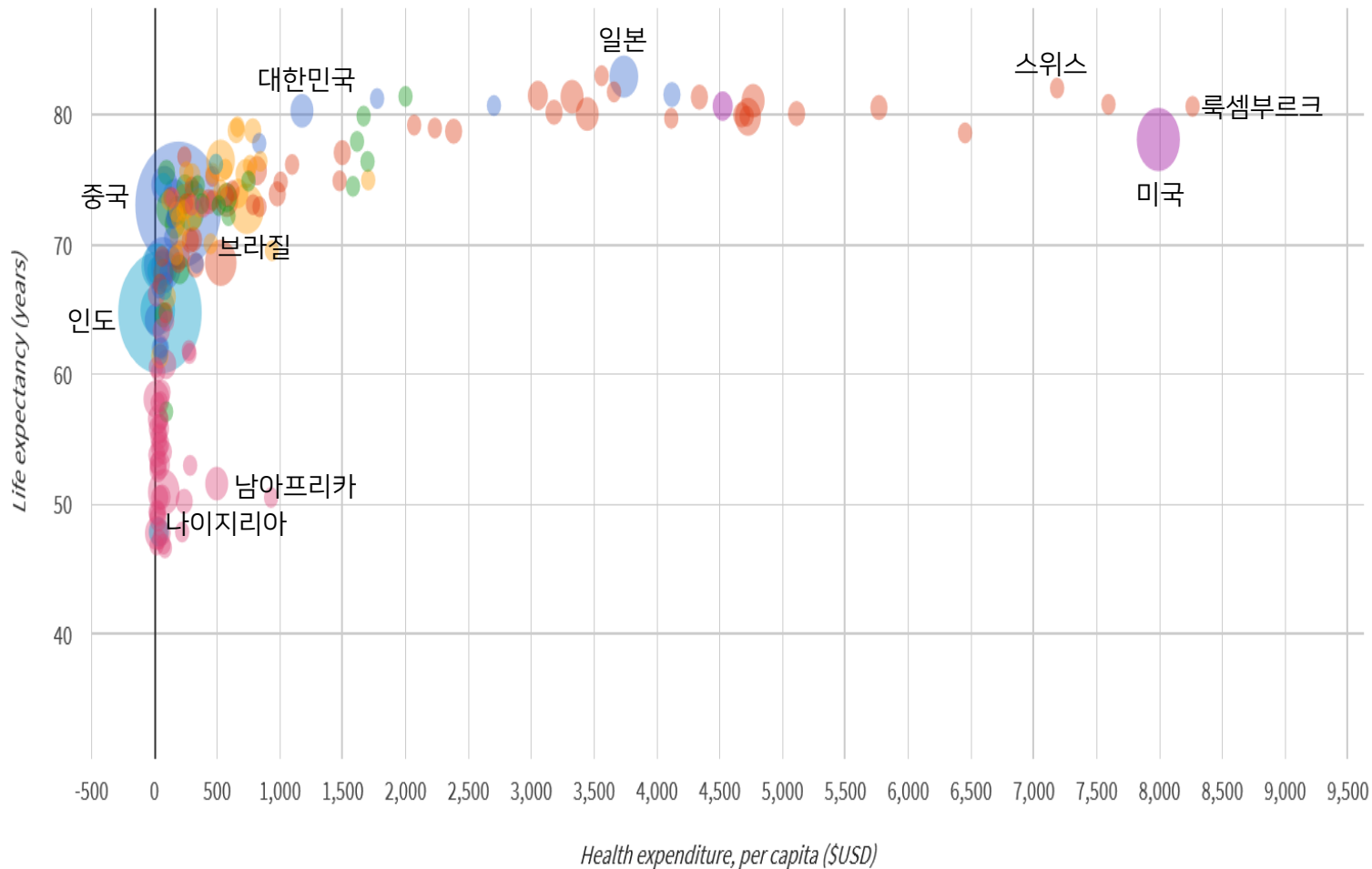
## 건강 지출 vs 기대 수명, 2007



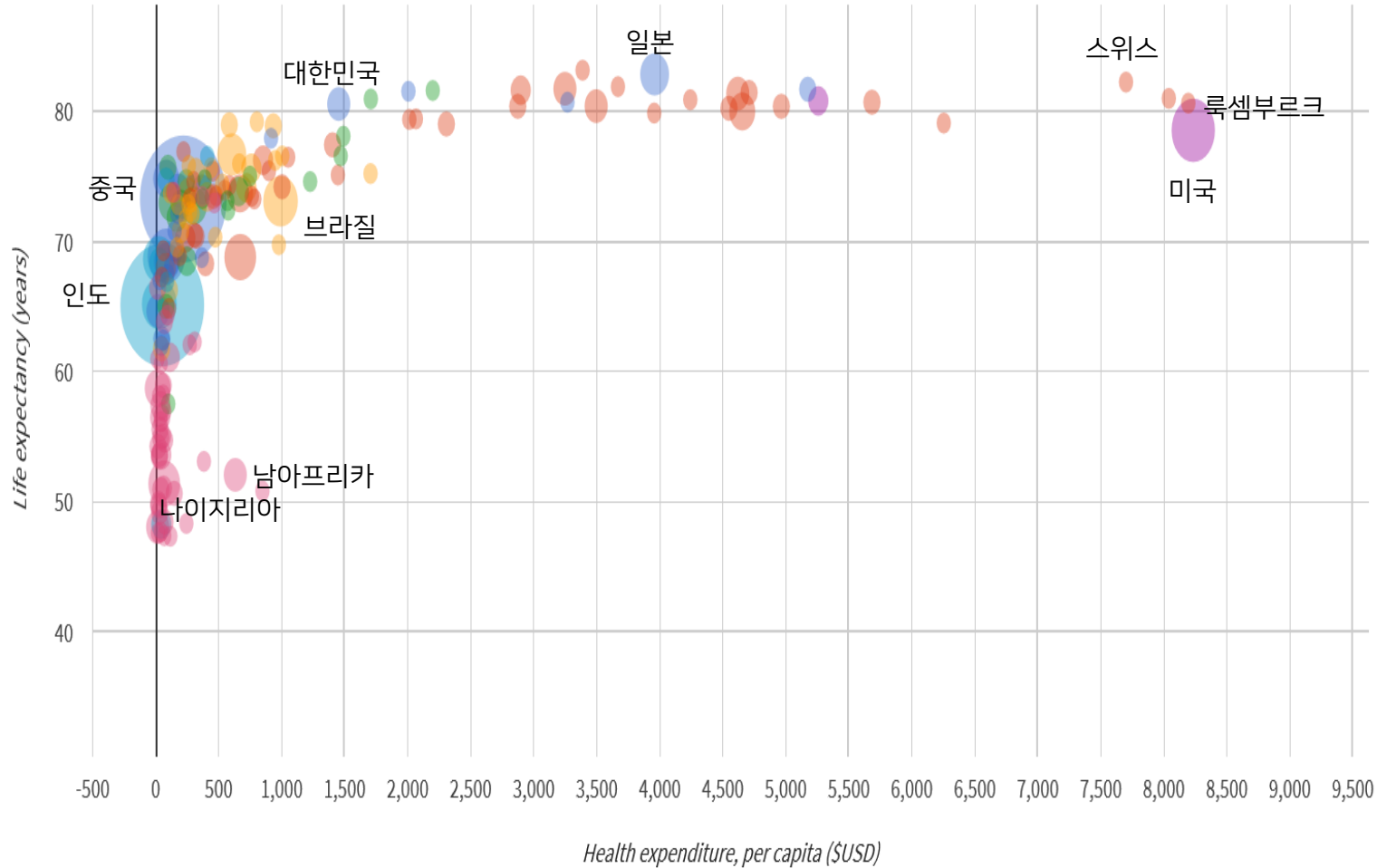
## 건강 지출 vs 기대 수명, 2008



## 건강 지출 vs 기대 수명, 2009



건강 지출 vs 기대 수명, 2010

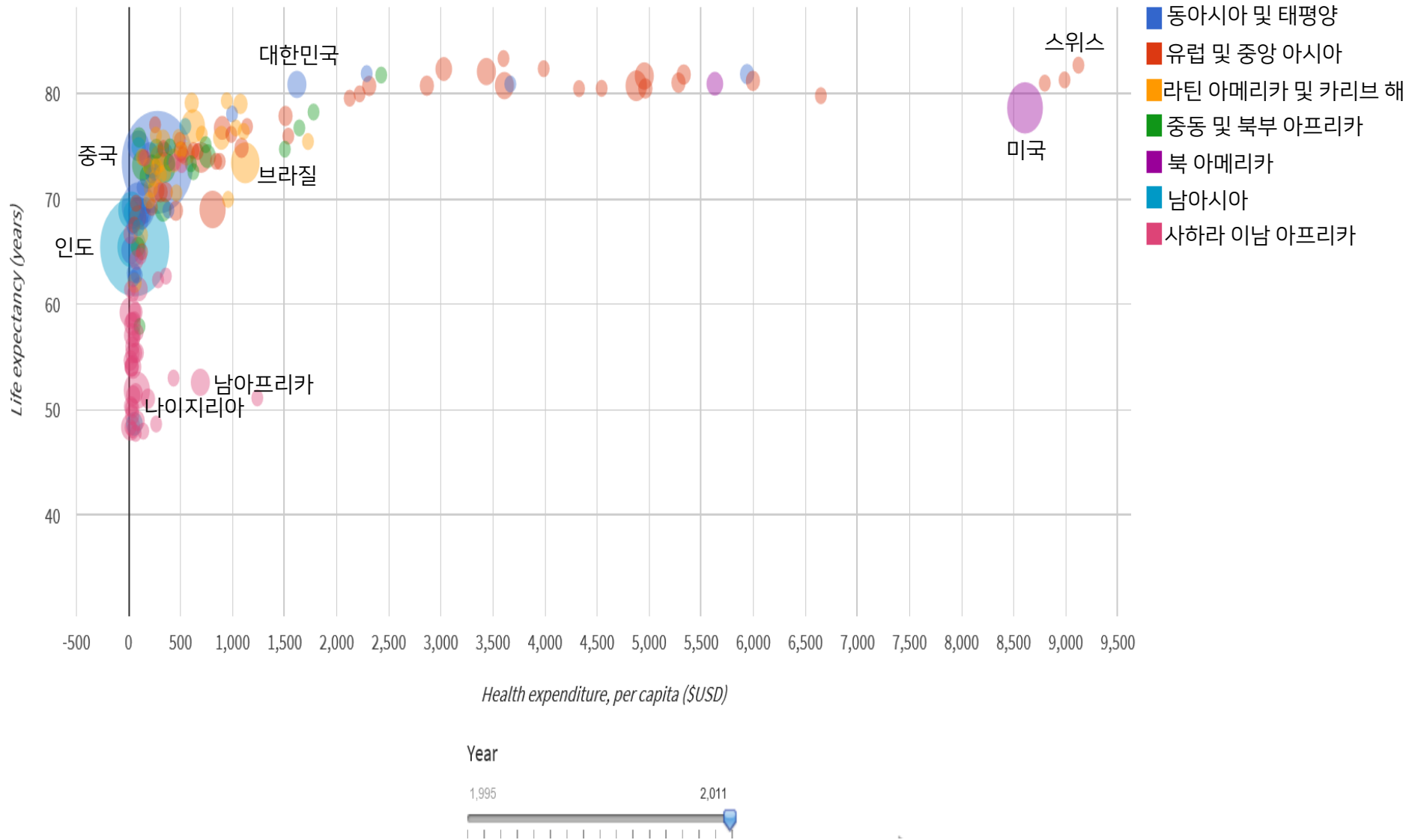


Year

1,995

2,010

## 건강 지출 vs 기대 수명, 2011





## 🔍 데이터 시각화가 중요한 이유

1. 데이터 분석에 대한 전문성이 없을지라도 시각화 차트를 통해 누구나 데이터를 직관적으로 이해할 수 있고, 데이터 기반의 의사결정을 할 수 있다.



<P&G - '비즈니스 스피어' 사례>

## 🔍 데이터 시각화가 중요한 이유



<P&G - '비즈니스 스피어' 사례, 출처: Fusion Brew>

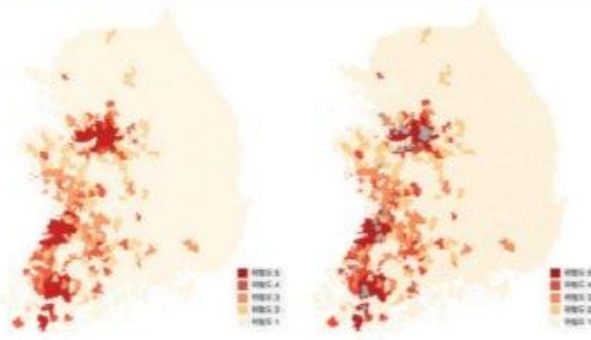
## 데이터 시각화가 중요한 이유



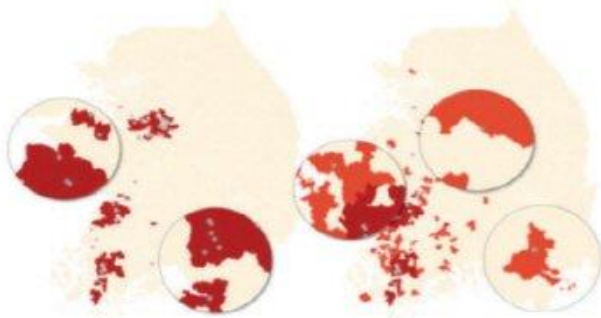
<KT 빅데이터센터 - 'AI 발병 원인 분석' 사례>

# AI 발병 위험지역 추정 시뮬레이션 (2014년 2월 대상)

전국을 위험정도에 따라 5단계로 분류 → 선제적 방역대상 제시



AI 발병농가의 약 83%가 고위험 추정지역(전국영토의 약 4%) 내에서 발생



• 자료 : KT, 빅데이터를 활용한 조류 인플루엔자 확산 대응 성과보고회 발표자료, 2014.12

## 조류인플루엔자(AI) 확산 방지에 동참해주세요

+ AI는 이렇게 확산됩니다



병에 걸린 새의 배설물



배설물이 묻은 신발



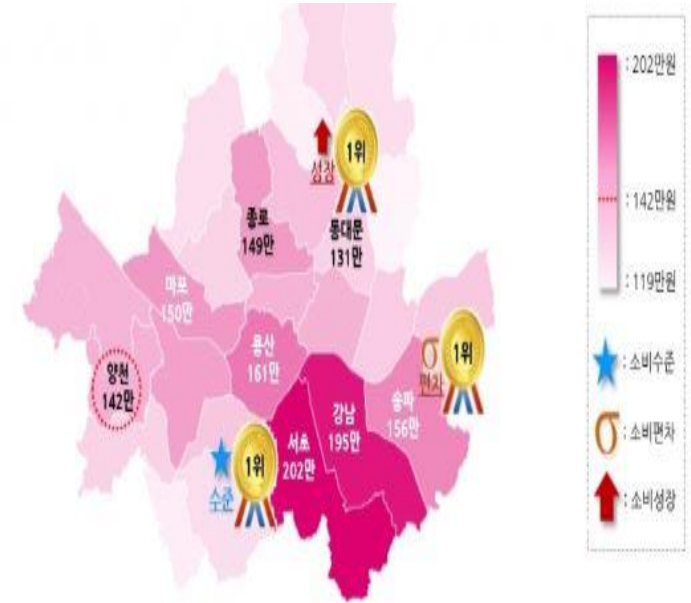
배설물이 묻은 자동차 바퀴



## <KT 빅데이터센터 - 'AI 발병 원인 분석' 사례>

## 데이터 시각화가 중요한 이유

2. 데이터 시각화는 데이터 인사이트를 도출하기 위한 방법론으로서 큰 역할을 한다.



<b>"소비 수준이 제일 높은 지역은?"</b>		<b>"평균대비 편차가 제일 큰 지역은?"</b>		<b>"연평균 성장률이 제일 높은 지역은?"</b>	
서초구 : 202만원 (소비지수*2: 141)	강남구 : 195만원 (소비지수 : 136) 용산구 : 161만원 (소비지수 : 113)	송파구 : 5.37	영등포구 : 3.11 성동구 : 2.87	동대문구 : 6.6%	양천구 : 6.5% 강북구 : 6.1%

<신한은행 - '서울시 생활 금융지도' 사례>

## 🔍 데이터 시각화가 중요한 이유

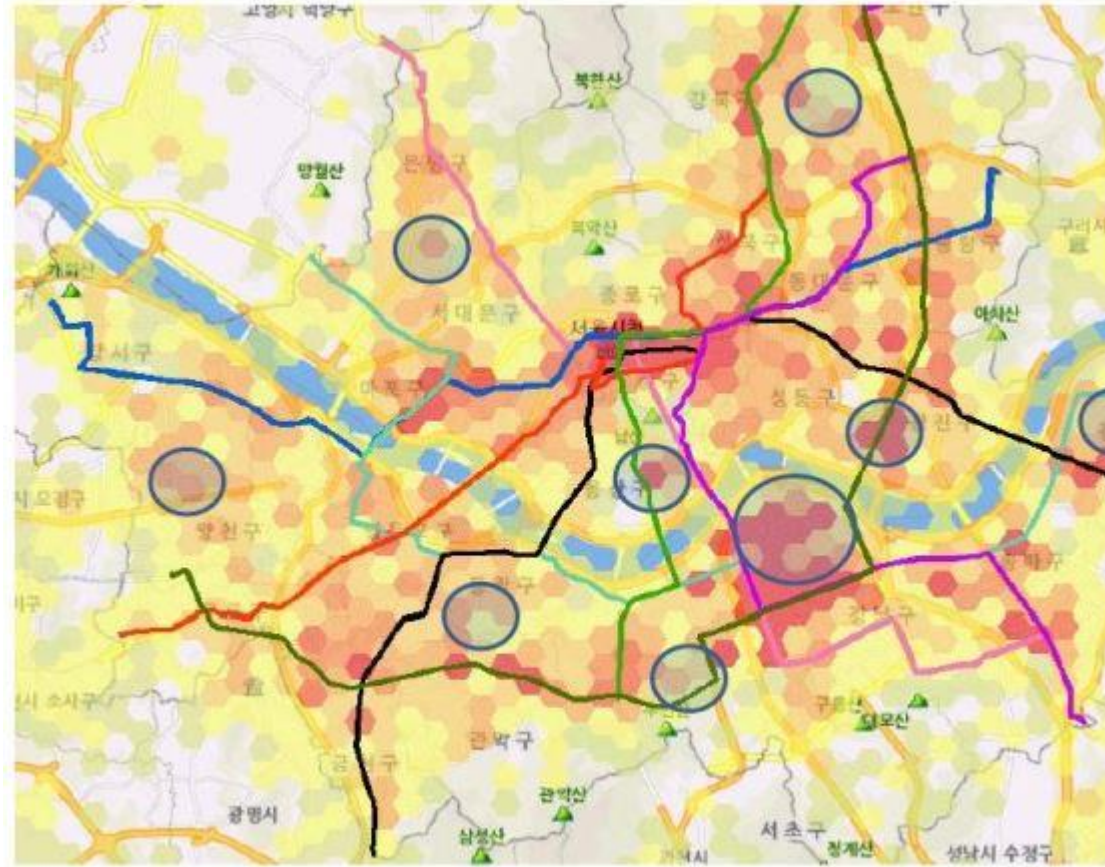
3. 데이터 시각화는 데이터 속에 숨겨진 의미를 찾는데 효과적인 방법이다.



출처 : EBS <소프트웨어 세상을 변화시키다>

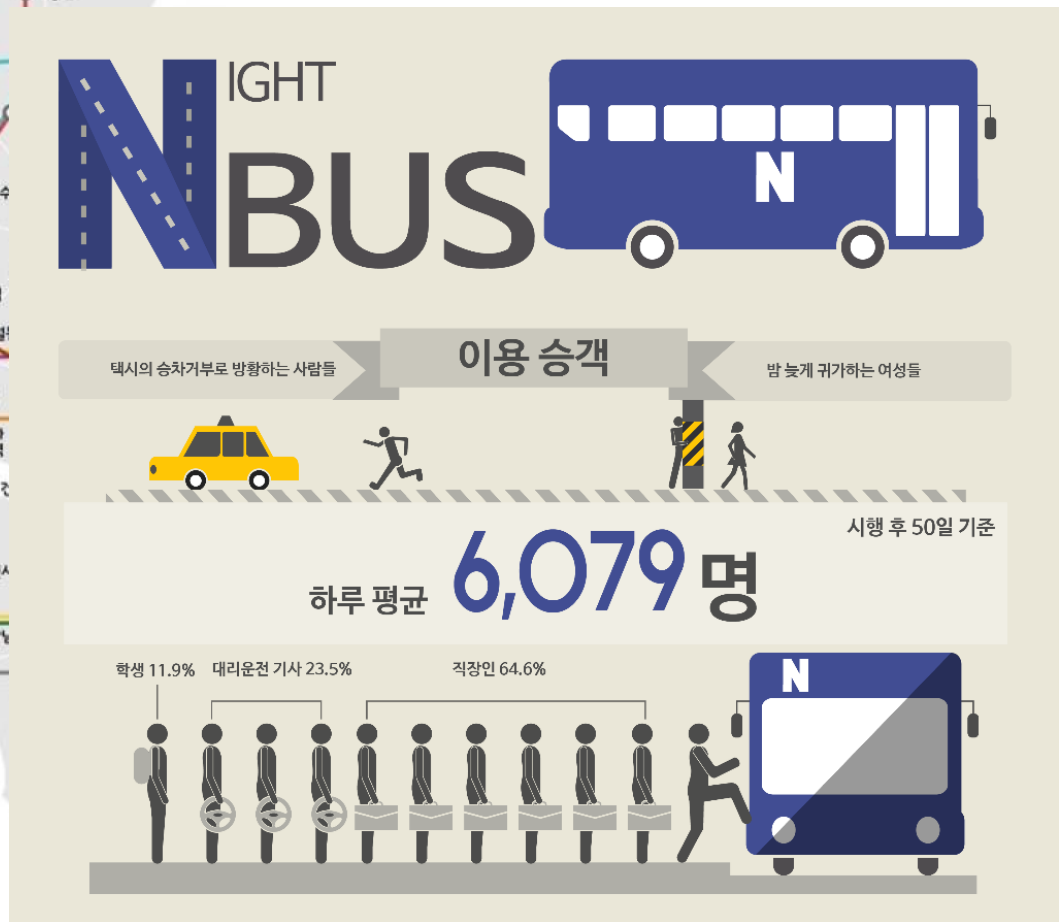
<서울시 심야버스 노선 개발 사례>

## 🔍 데이터 시각화가 중요한 이유



<서울시 심야버스 노선 개발 사례>

## 데이터 시각화가 중요한 이유



## <서울시 심야버스 노선 개발 사례>

## 2. 시각화의 정의 및 분류

## 시각화에 대한 잘못된 인식

일반적으로 사람들은 데이터 시각화를 단순히 차트의 형태로만 표현한 것 이라고 생각

수치 등의 데이터

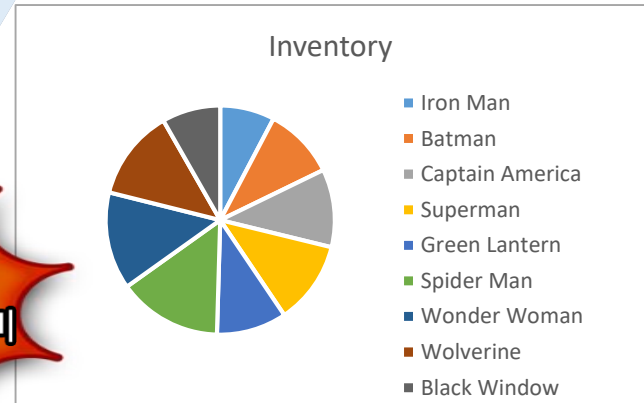
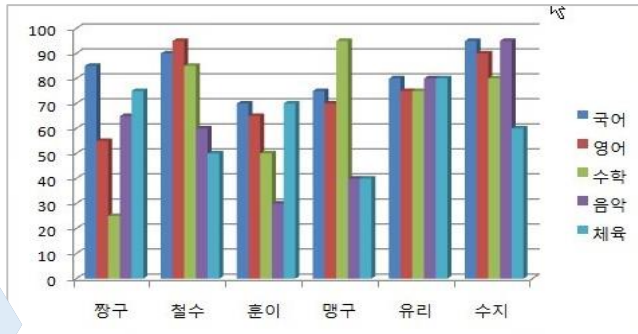
열1	국어	영어	수학	음악	체육
짱구	85	55	25	65	75
철수	90	95	85	60	50
훈이	70	65	50	30	70
맹구	75	70	95	40	40
유리	80	75	75	80	80
수지	95	90	80	95	60

Costumes	Inventory
Iron Man	42
Batman	55
Captain America	60
Superman	64
Green Lantern	54
Spider Man	80
Wonder Woman	75
Wolverine	70
Black Window	45

수치를 차트의  
형태로 표현

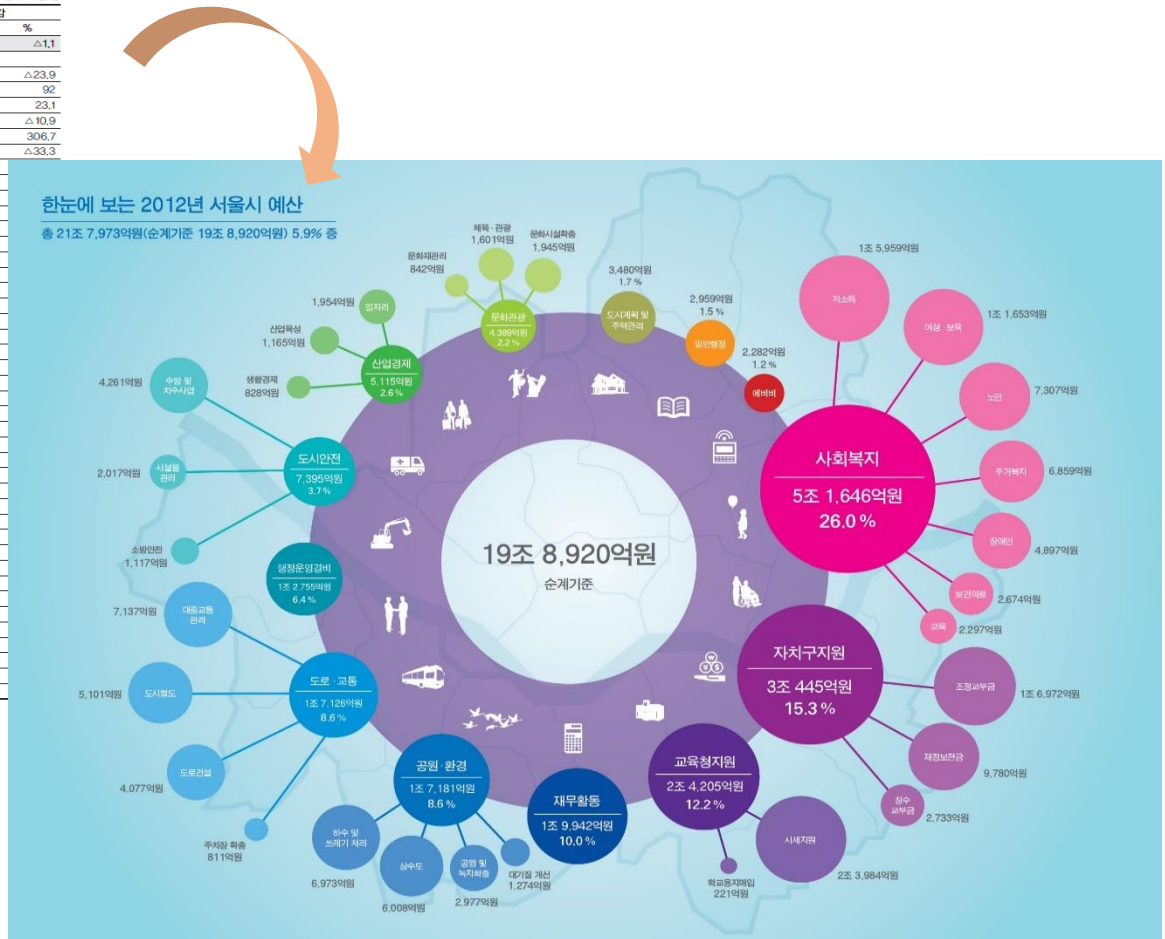
일반적으로 사람들이  
알고있는 시각화의 의미

차트의 형태로 표현



☑ 데이터 시각화란 예쁘게, 멋있게 만들어 주는 것을 떠나 데이터를 쉽게 이해할 수 있도록 돕는 것

구분	'17예산		'18예산(안)		증감	%
	본예산	추경(안)	요구	조정(안)		
<b>총계</b>	<b>461,247</b>	<b>461,247</b>	<b>528,800</b>	<b>456,381</b>	<b>4,866</b>	<b>△1.1</b>
<b>[일반회계]</b>	<b>202,810</b>	<b>202,810</b>	<b>286,534</b>	<b>237,655</b>		
임상연구인프라조성(R&D)	48,300	48,300	37,055	36,736	△11,564	△23.9
국가합선신약개발(R&D)	7,619	7,619	14,632	14,632	7,013	92
연·중생선원 특약육성(R&D)	24,375	24,375	30,015	30,015	5,640	23.1
포스트게놈다부처산신약육성을 위한연구제시사업(R&D)	12,611	12,611	11,234	11,234	△1,377	△10.9
국가전략포도록특화(R&D)	3,478	3,478	14,146	14,146	10,668	306.7
심혈관계질환치료의국가산출원시스템기술개발(R&D)	700	700	1,500	467	△233	△33.3
연구자주도 질병국책 연구사업(R&D)	-	-	13,900	9,300	9,300	
공직차질국책연구연구 지원사업(R&D)	-	-	7,800	3,755	3,755	
국가전략포도록특화(R&D)	-	-	13,900	9,800	9,800	
발생치유전·진상기술개발(R&D)	11,000	11,000	11,000	11,000		
안보지능화이오로봇융합활용기술개발(R&D)	-	-	2,800	2,800	2,800	
한반보건산업진흥원 운영(R&D)	21,636	21,636	22,865	22,304	668	
보건산업정책지원관리	57	57	107	57	-	-
한도화산업육성	517	517	517	-	-	-
보건의료빅데이터플랫폼구축사업	-	-	23,976	11,468	11,468	
첨단의료복합단지(안)기술구축(R&D)	7,394	7,394	7,394	7,322	△72	
통합의학센터건립지원	4,000	4,000	2,000	1,000	△3,000	
통합의료연구지원사업(R&D)	2,667	2,667	1,667	1,667	△1,000	
제대산업육성 지원	9,822	9,822	10,822	9,826	4	
의료기기산업경쟁력강화	4,190	4,190	5,320	4,640	450	
바이오헬스기술비즈니스생태계조성	2,986	2,986	9,086	3,586	600	
글로벌바이오융합성장인프라구축	10,009	10,009	7,609	6,779	△3,230	
보건산업정책국립기관경비(총액)	71	71	73	70	△3	
보건산업정책국립기관경비(비총액)	163	163	168	145	△23	
해외환자유치지원	19,568	19,568	24,068	13,178	△6,390	
의료시스템수출지원	11,972	11,972	12,972	11,531	△441	
해외의료사업지원기관경비(총액)	61	61	63	56	△10	
해외의료사업지원기관경비(비총액)	131	131	135	121	△15	
<b>[지역발전특별회계]</b>	<b>56,798</b>	<b>56,798</b>	<b>58,774</b>	<b>37,834</b>	<b>△18,964</b>	
지역의료발전기금 조성(경제)	56,798	56,798	58,774	37,834	△18,964	
<b>[공인간접회계]</b>	<b>201,639</b>	<b>201,639</b>	<b>183,502</b>	<b>180,892</b>	<b>△2,747</b>	
지방연구특화기술개발(R&D)	66,019	66,019	43,950	41,340	△24,679	
산도형특성화연구사업(R&D)	10,500	10,500	8,021	8,021	△2,479	
감염병연구대응기술개발(R&D)	28,211	28,211	28,481	28,481	270	
의료기기기술개발(R&D)	74,944	74,944	73,944	73,944	△1,000	
의료기기기술개발(R&D)	21,965	21,965	29,106	29,106	7,141	
심혈관계질환치료의국가산출원시스템기술개발(R&D)	81,192	81,192	78,067	66,019	△15,173	
산도형특성화연구사업(R&D)	11,500	11,500	10,500	10,500	△1,000	
감염병연구대응기술개발(R&D)	27,318	27,318	27,911	28,211	893	
첨단의료기술개발(R&D)	73,654	73,654	70,711	74,244	590	
의료기기기술개발(R&D)	21,628	21,628	19,465	21,965	337	



## 🔍 분석에서 시각화의 목적

- ✓ 데이터 중에서도 **의미 있는 정보**를 시각화하여 **인사이트 도출**



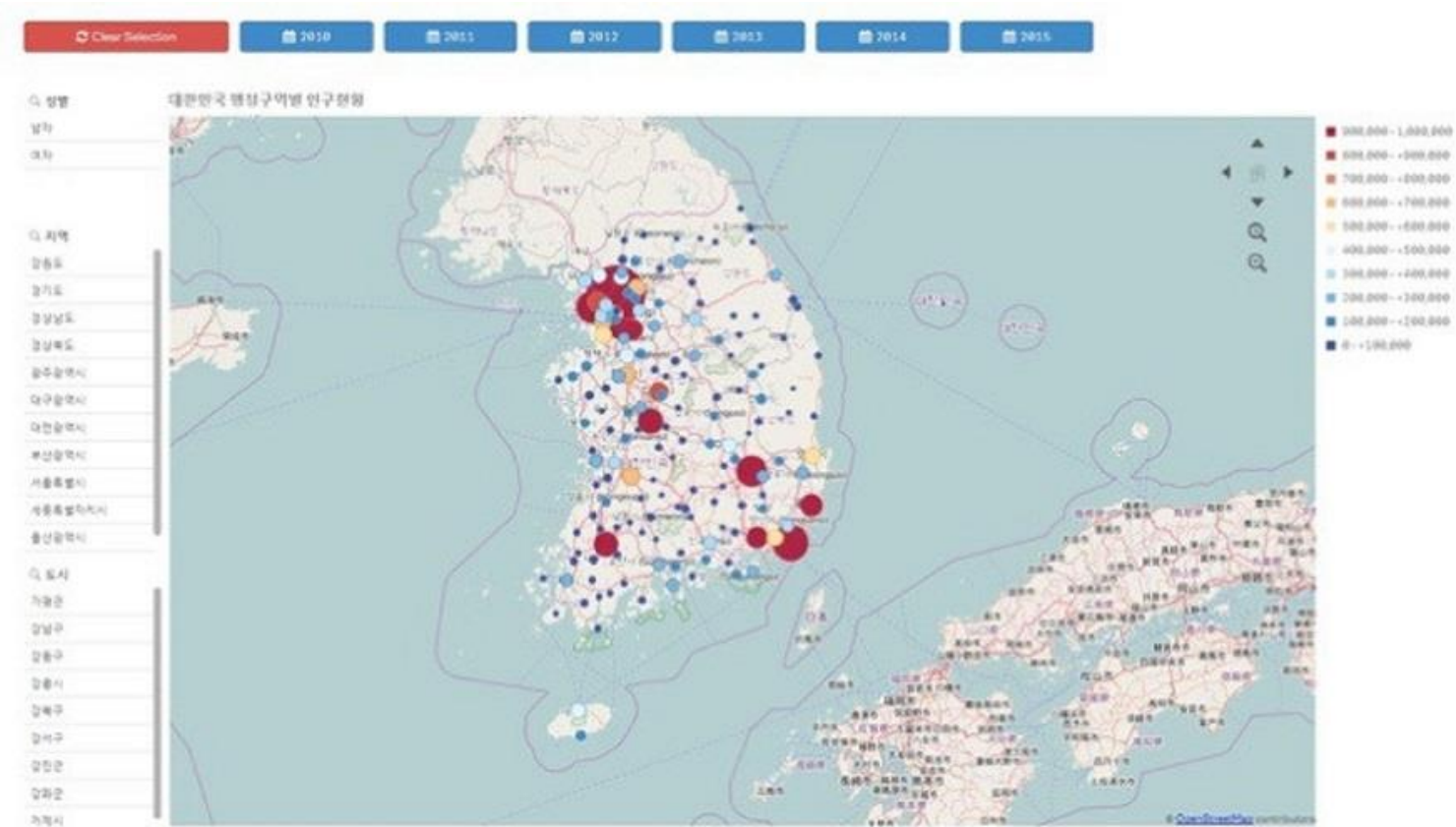
시각화

## 인사이트 도출



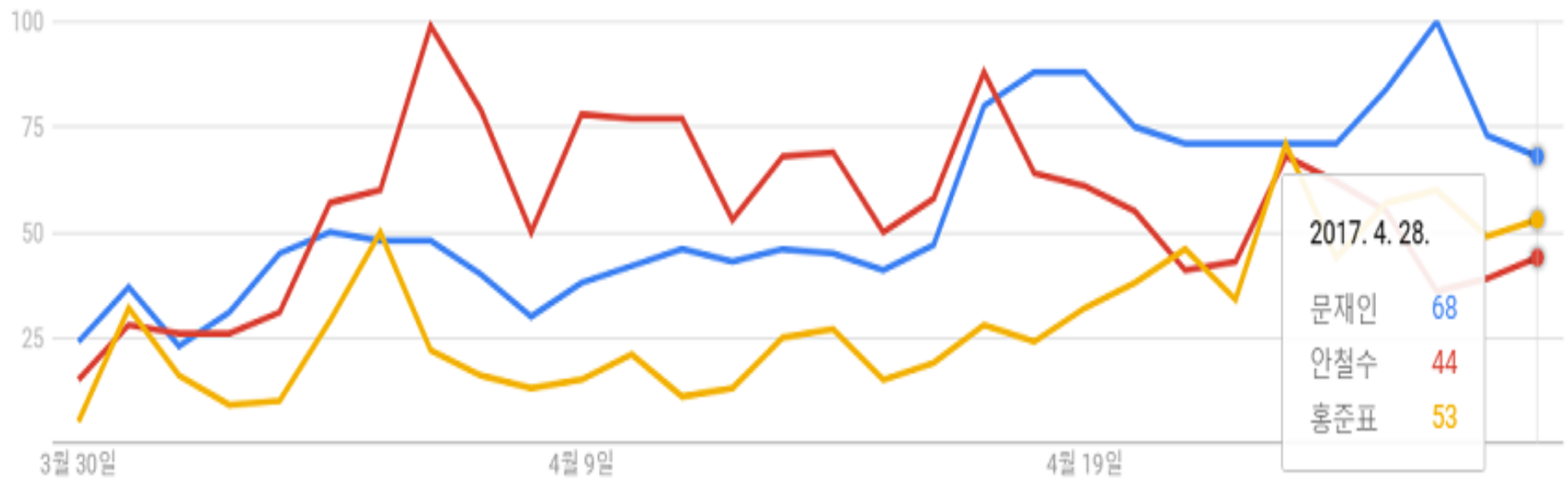
## 시각화의 분류 - ① 정보 시각화

- ✓ 정보 시각화는 대규모 수량, 비수량 데이터를 색채, 통계(도표, 그래프 등) 등을 활용하여 시각적으로 표현하는 것을 의미한다.



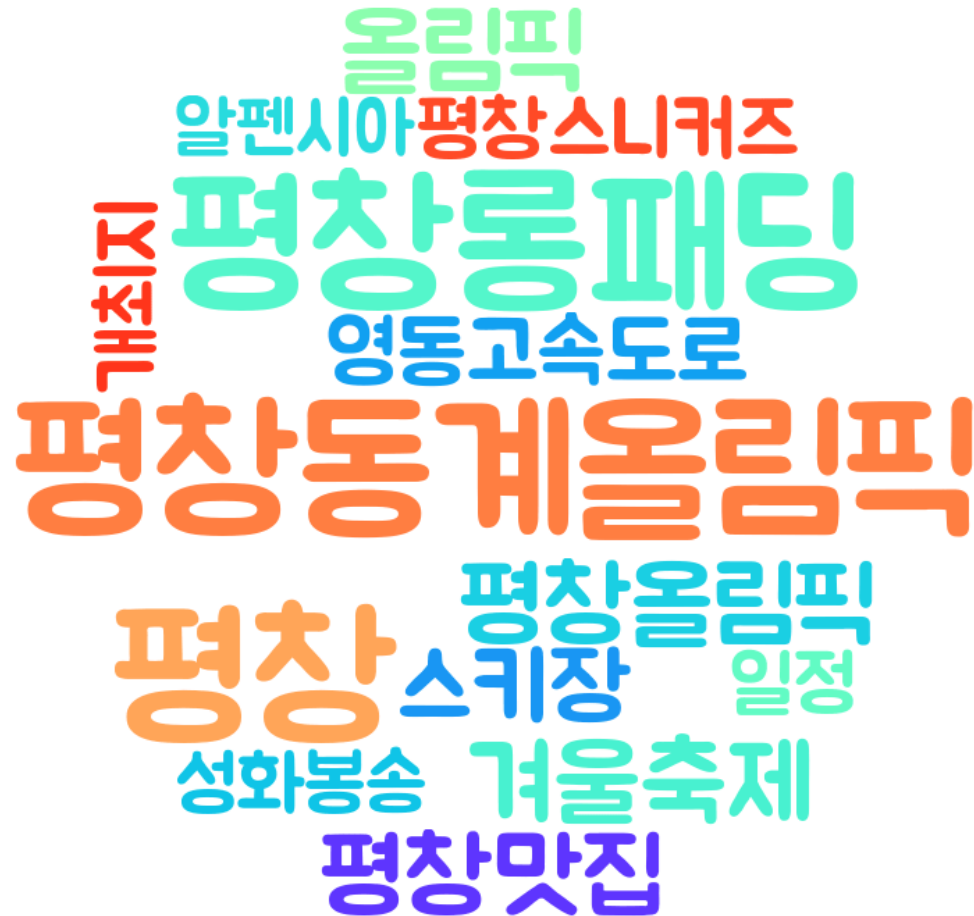
<대한민국 행정구역 별 인구 현황>

## <구글 트렌드 - 검색량 추이>



## <정보 시각화 예 - 라인 차트>

<네이버 키워드 - 평창동계올림픽 연관 키워드>



Made In Wordcloud.kr

<정보 시각화 예 - 워드 클라우드>

## ② 인포그래픽(Infographics)

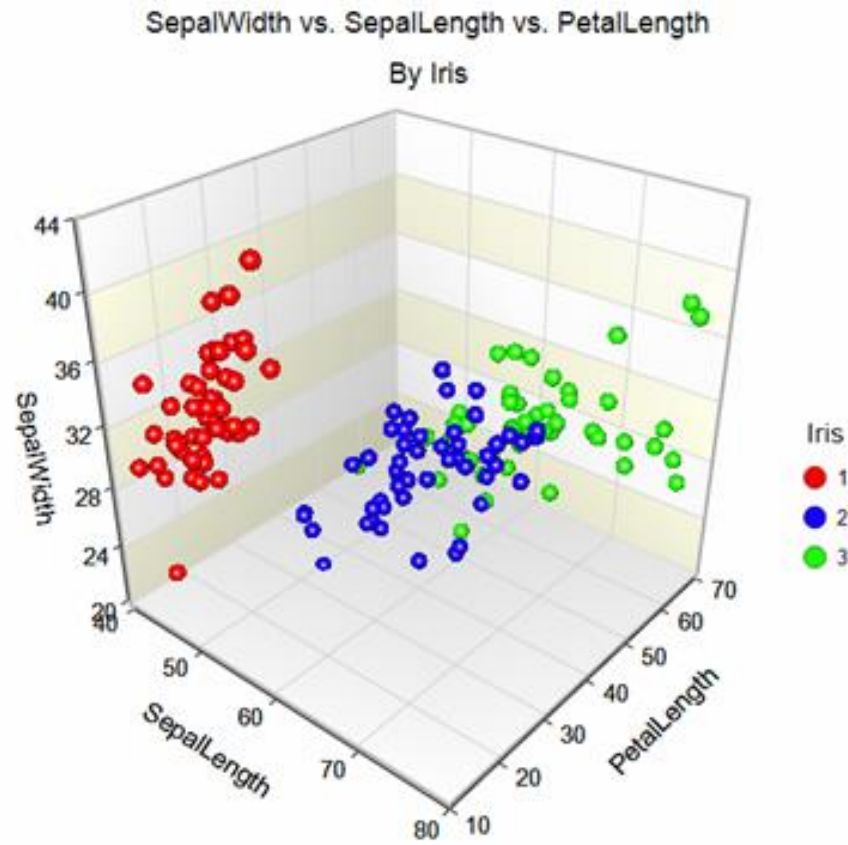
- ✓ 인포메이션과 그래픽의 합성어로, 복잡한 수치나 글로 표현되어 있는 다량의 정보를 차트, 지도, 다이어그램, 로고, 일러스트레이션 등을 활용하여 한눈에 파악할 수 있도록 하는 디자인





### ③ 과학적 시각화

- ✓ 실험결과나 시뮬레이션 데이터 등 복잡한 데이터를 쉽게 탐색할 수 있도록 3차원 그래픽 기술 등을 활용하여 시각화 하는 기술



<과학적 시각화 예 - 3D 그래프>

🔍 시각화 분석 도구



✓ R의 장점은 오픈소스이며 데이터 시각화 성능이 좋다는 것이다.

# 3. 시각화 차트 선택과 유형



 데이터 유형에 가장 적합한 차트는?

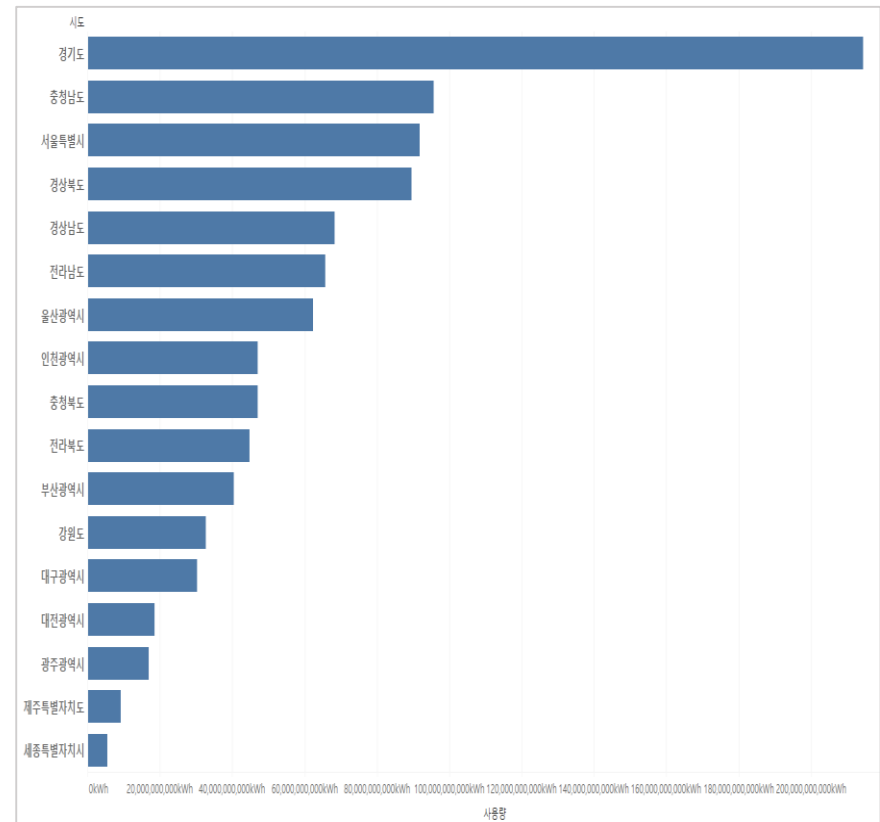
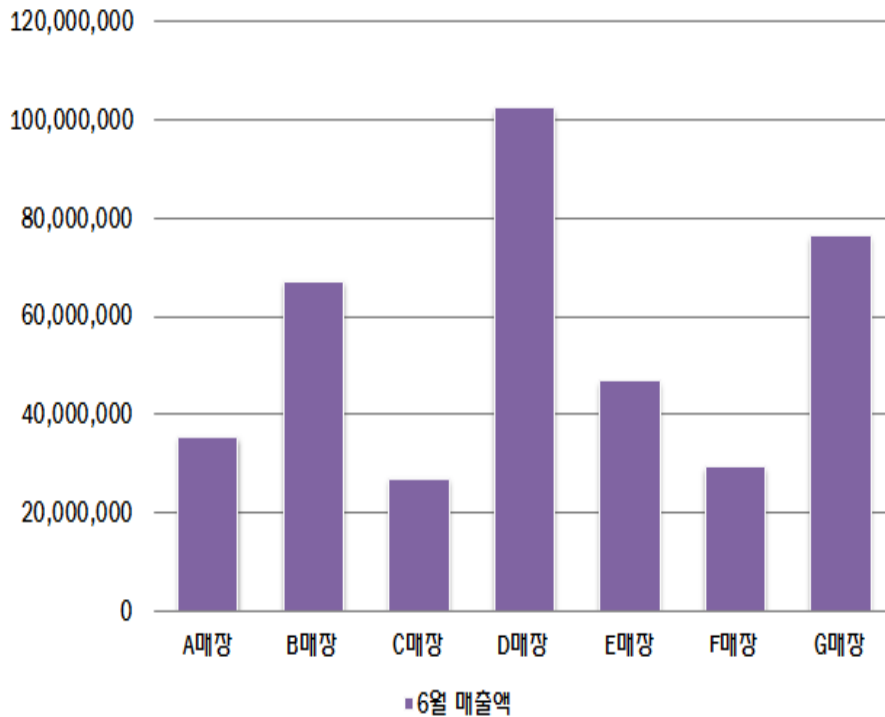




## 막대 차트

- ✓ 데이터 내 항목 별로 많고 적음을 비교할 때 사용  
Ex)매장 별 매출액 비교, 지역별 전력 사용량

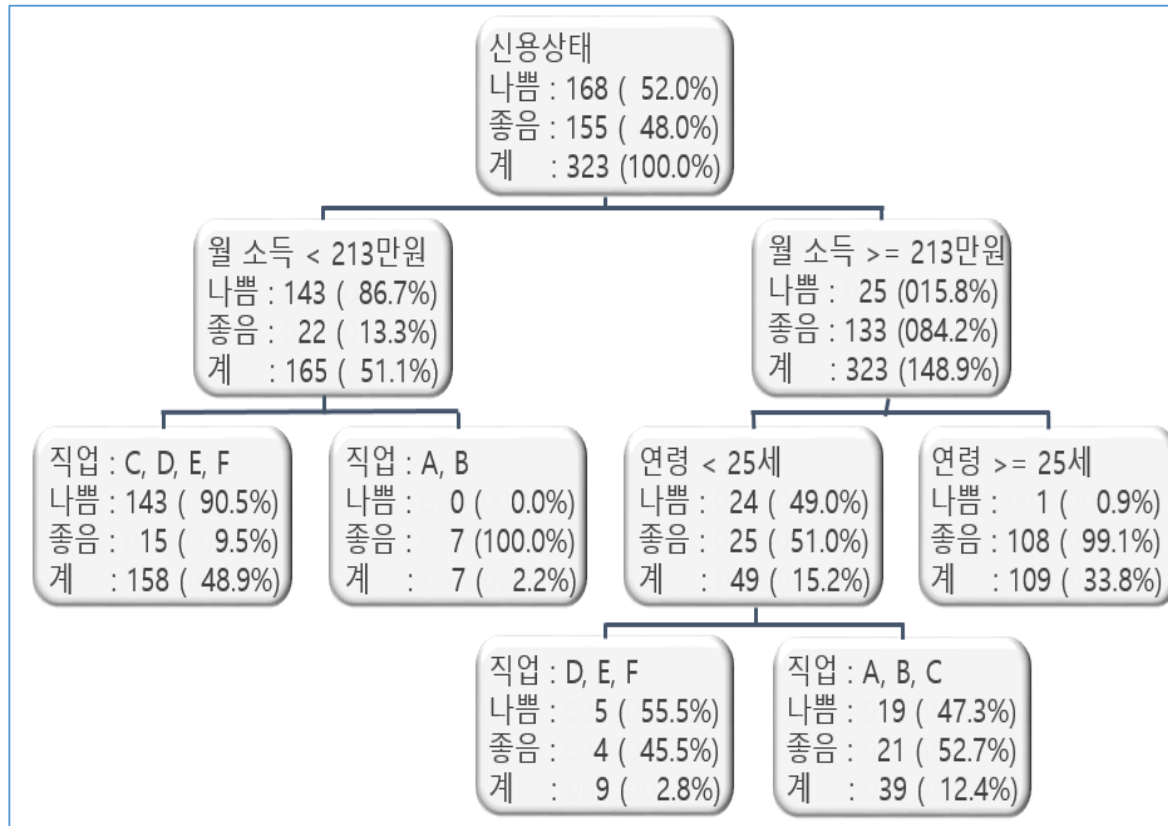
6월 매출액





## 트리 차트

- ✓ 데이터를 나무구조로 도표화하여 분류와 예측을 수행할 때 사용  
Ex) 대출승인 의사결정 자동화 사례



## 라인 차트

- ✓ 일반적으로 시간 경과에 따른 데이터의 트렌드를 볼 때 사용

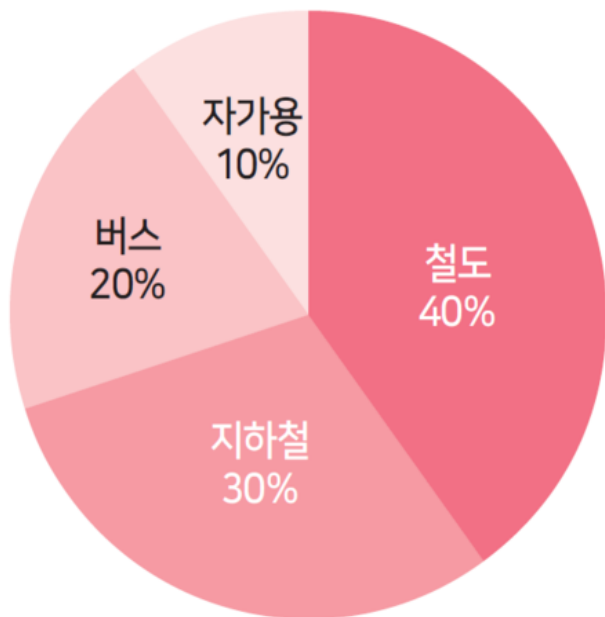


## 🔍 파이 차트

- ✓ 원을 백분율 기준으로 나누어 데이터에 대한 **상대적인 비율을 표시**할 때 사용  
Ex) 설문조사의 응답 카테고리

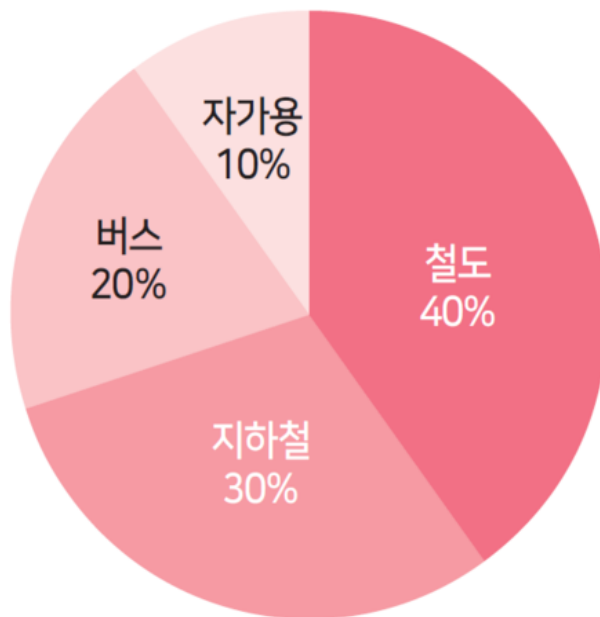
Don't

철도 이용자 40%로 비중 가장 높아



Do

철도 이용자 40%로 비율 가장 높아



### Ex) 미국의 주별 살인과 범죄 발생 빈도



## 히트맵

- ✓ 데이터의 상대 밀도 또는 높고 낮음을 색상으로 표시할 때 사용  
Ex) 운동선수의 활동량 분석, 전국 공시지가 현황

### Heatmap

#### Tottenham Hotspur

- ☐ Hugo Lloris
- ☐ Kyle Walker
- ☐ Toby Alderweireld
- ☐ Jan Vertonghen
- ☐ Ben Davies
- ☒ Son Heung-Min
- ☐ Erik Lamela
- ☐ Eric Dier
- ☐ Dele Alli
- ☐ Nacer Chadli
- ☐ Harry Kane
- ☐ Substitutes
- ☐ Clinton N'Jie
- ☐ Christian Eriksen
- ☐ Tom Carroll

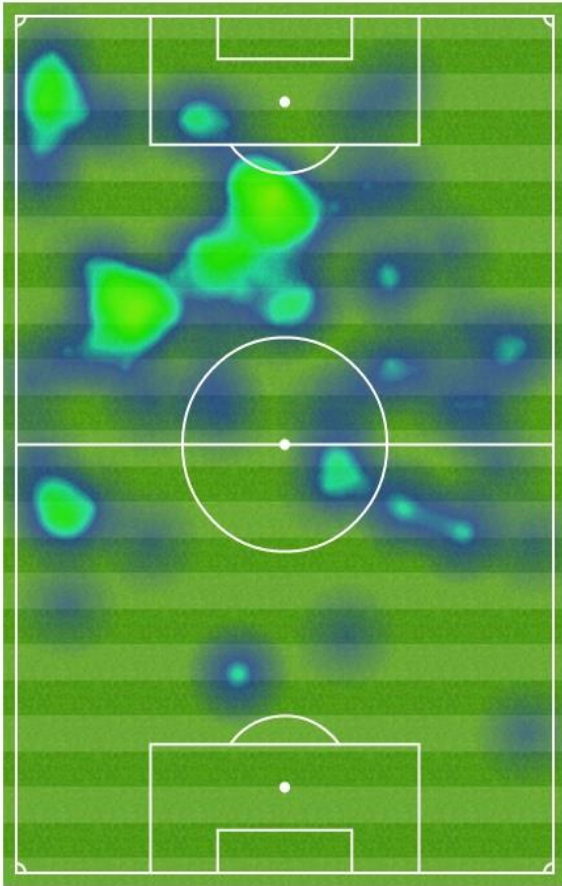
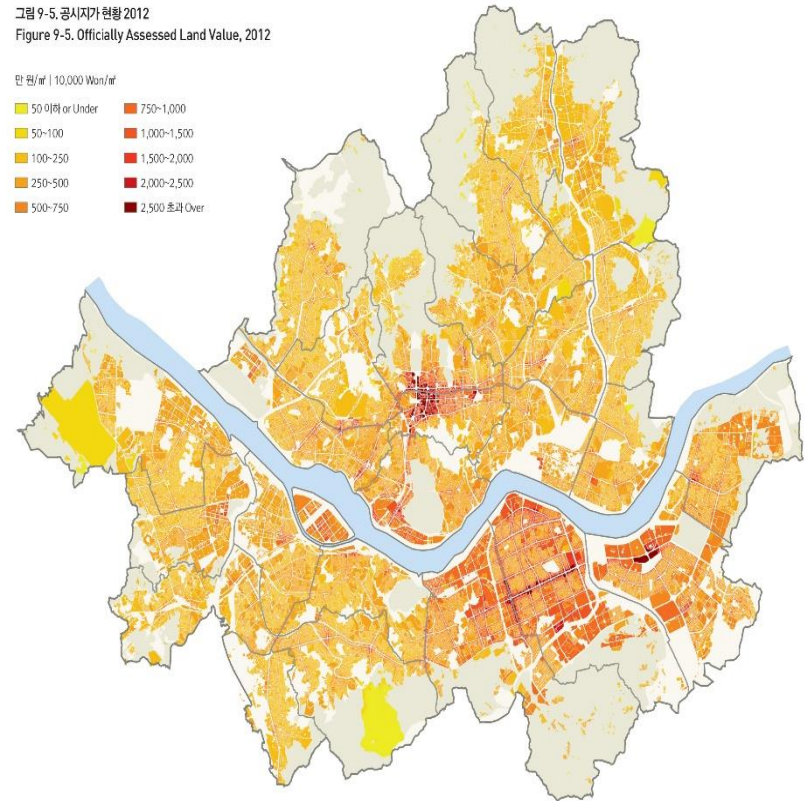


그림 9-5. 공시지가 현황 2012  
Figure 9-5. Officially Assessed Land Value, 2012

단 원/㎡ | 10,000 Won/㎡

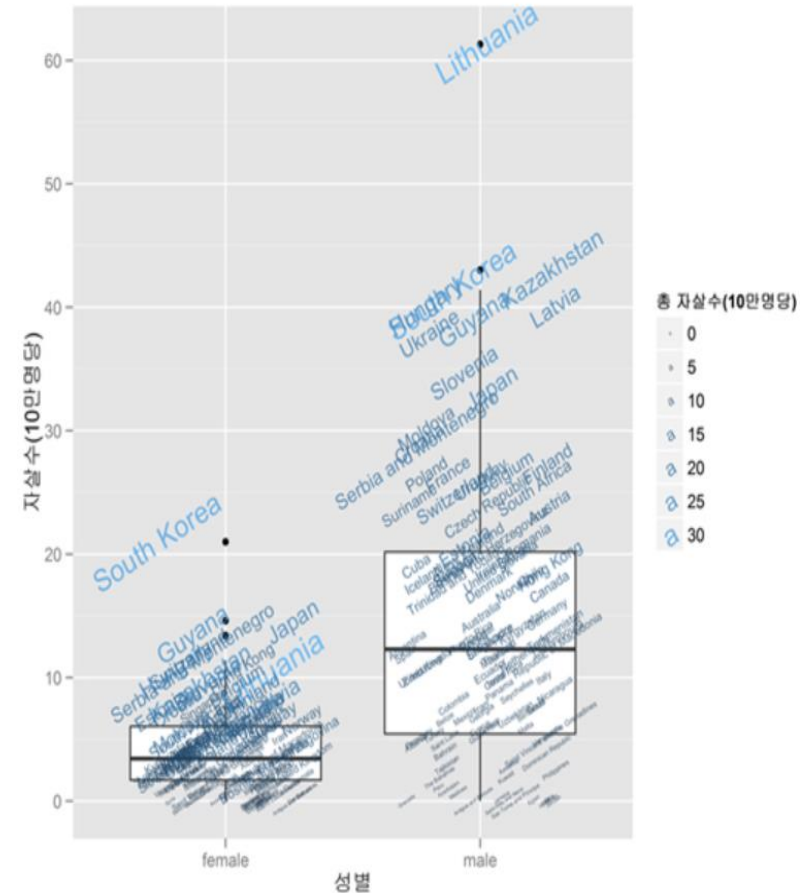
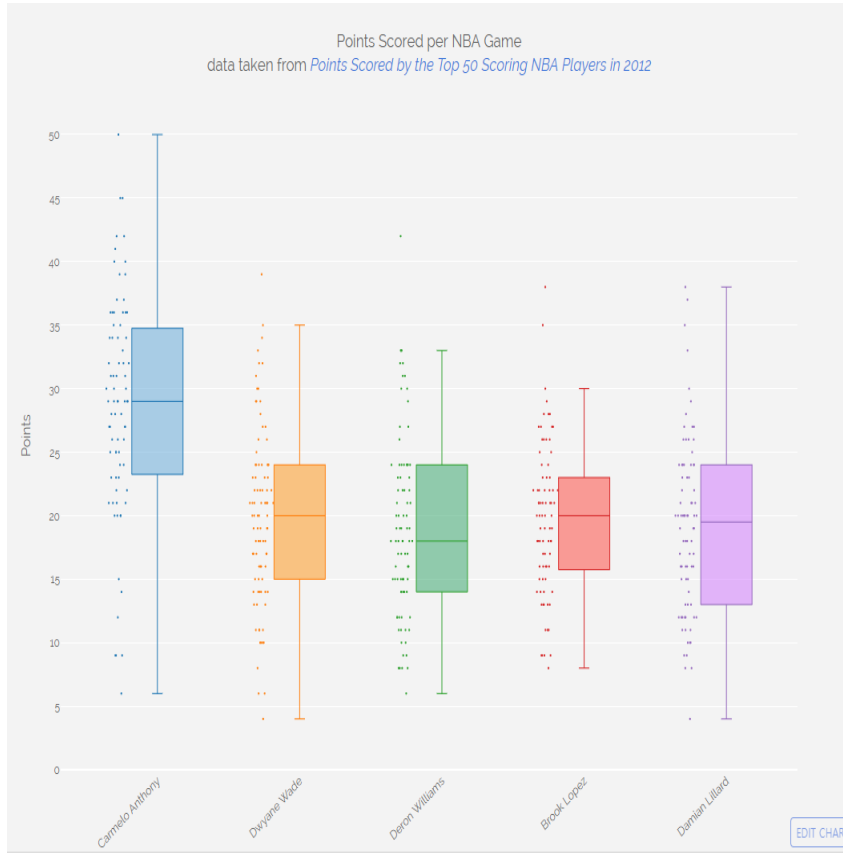
- |  |  |
|--|--|
| <span style="color: yellow;">■</span> 50 이하 or Under | <span style="color: red;">■</span> 750~1,000         |
| <span style="color: orange;">■</span> 50~100         | <span style="color: darkred;">■</span> 1,000~1,500   |
| <span style="color: gold;">■</span> 100~250          | <span style="color: darkred;">■</span> 1,500~2,000   |
| <span style="color: brown;">■</span> 250~500         | <span style="color: darkred;">■</span> 2,000~2,500   |
| <span style="color: darkbrown;">■</span> 500~750     | <span style="color: darkred;">■</span> 2,500 초과 Over |





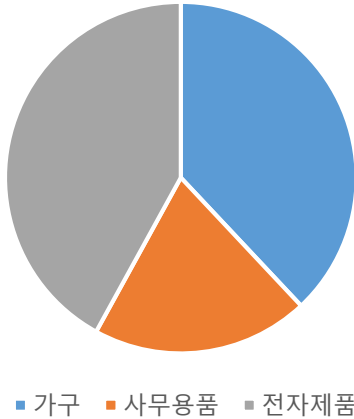
## 박스플롯

- ✓ 데이터 집합의 **범위와 중앙값을 빠르게 확인**하고 통계적으로 **이상치를 파악**할 때 사용  
Ex) 한 눈에 데이터의 범위를 파악해야 하는 경우, 데이터가 한 쪽으로 치우친 것을 확인하는 경우

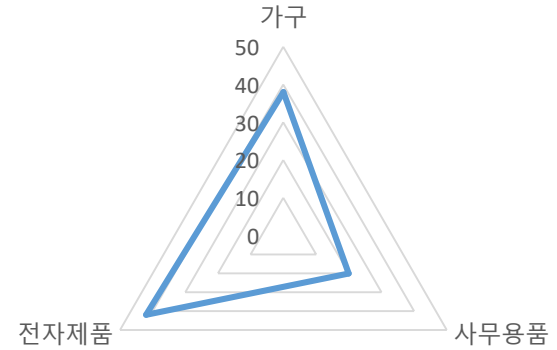


## 다음의 차트 중 어떤 차트가 더 제품 대분류별 비교가 쉽나요?

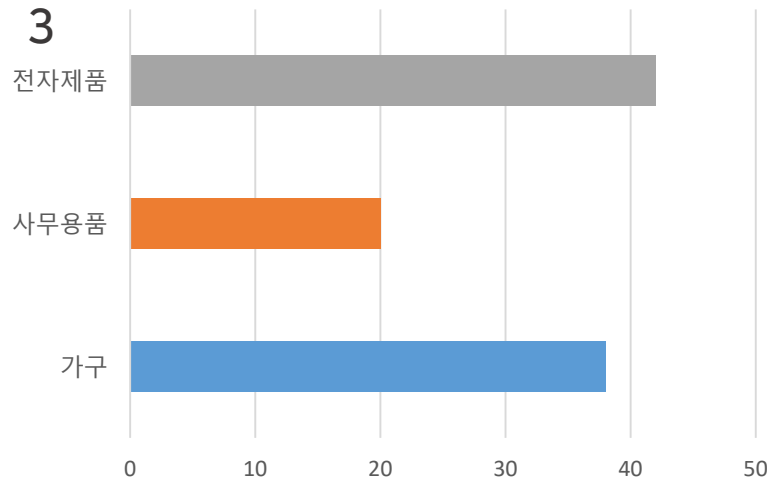
1



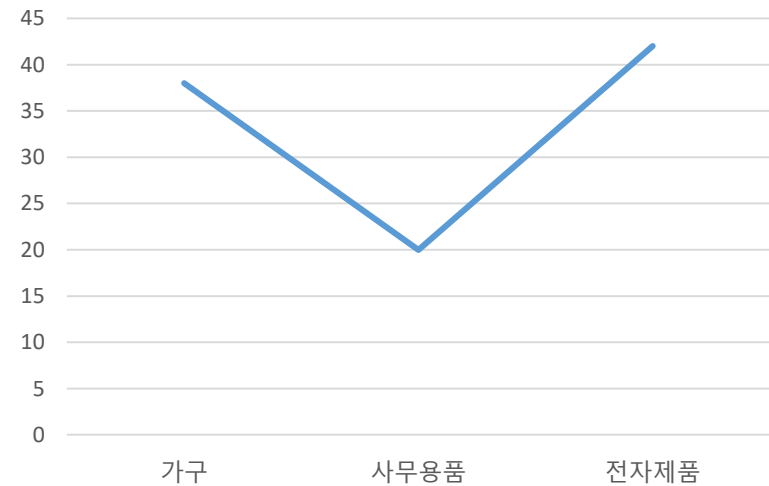
2



3

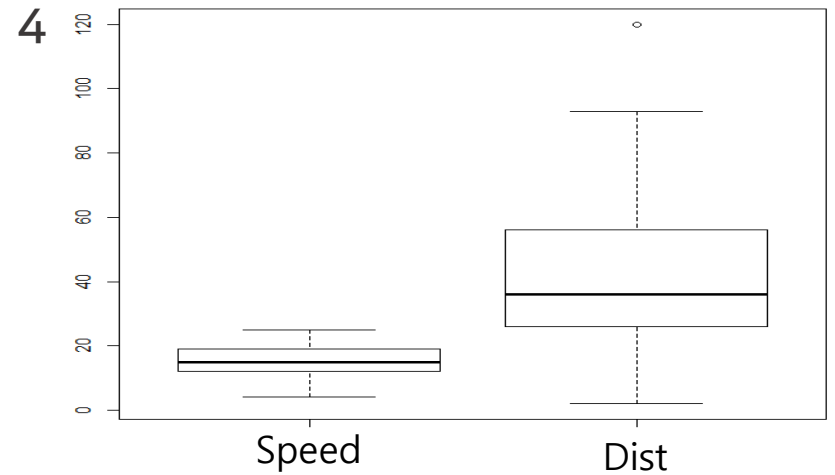
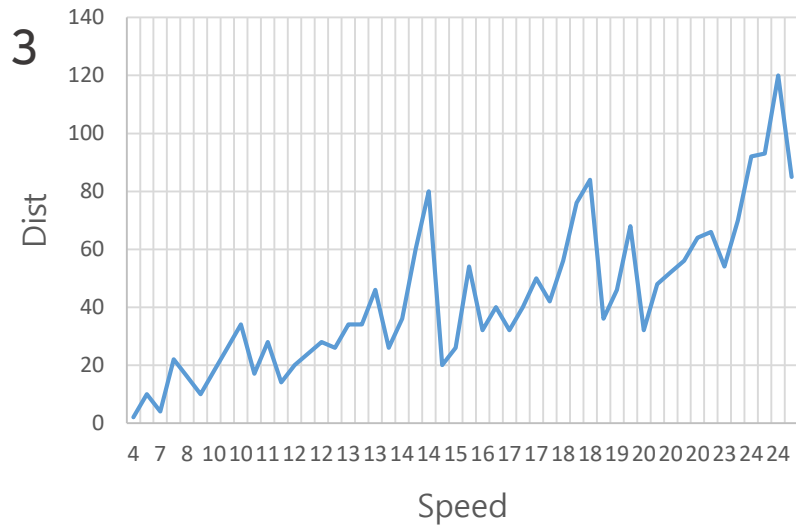
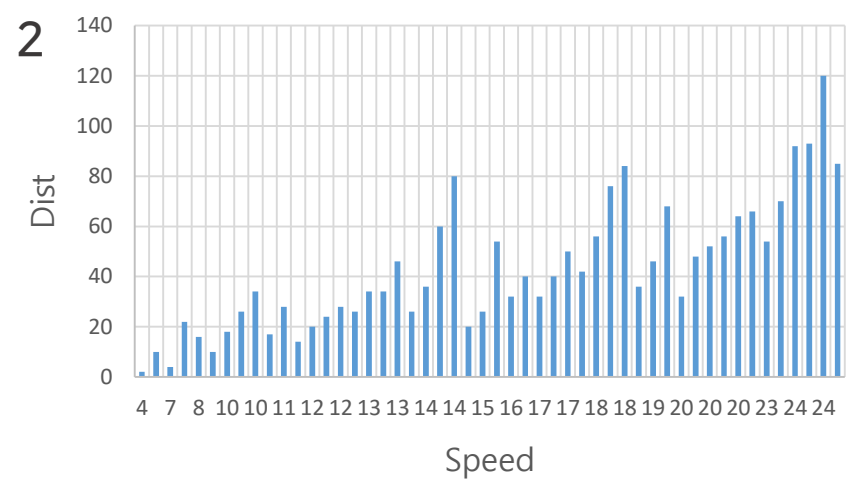
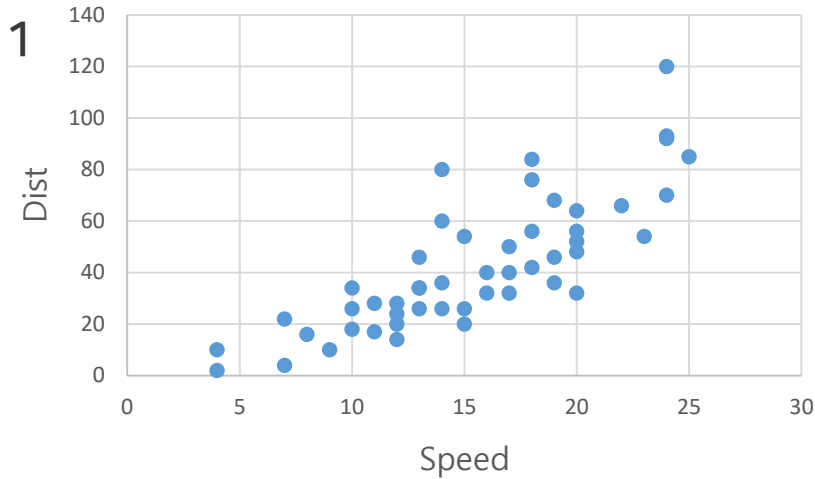


4



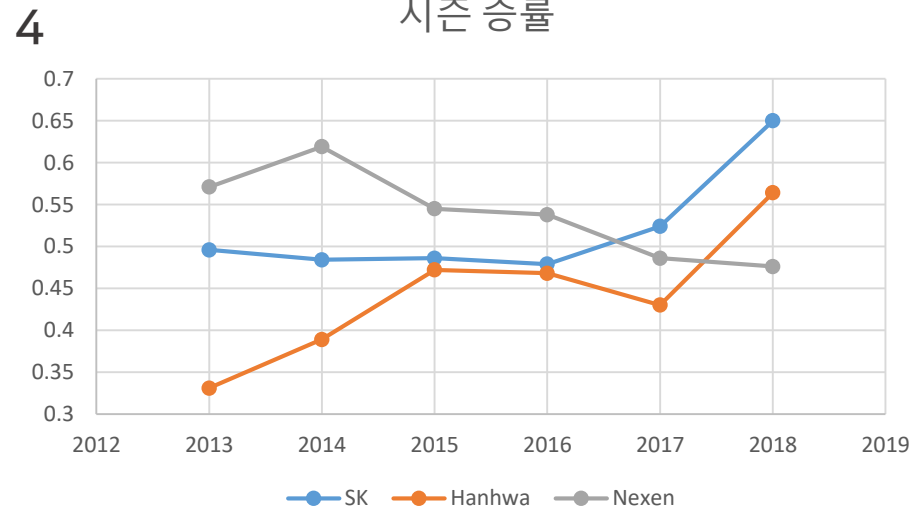
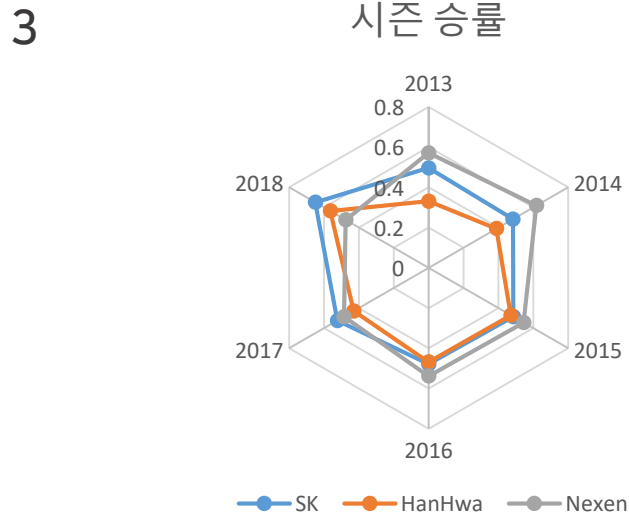
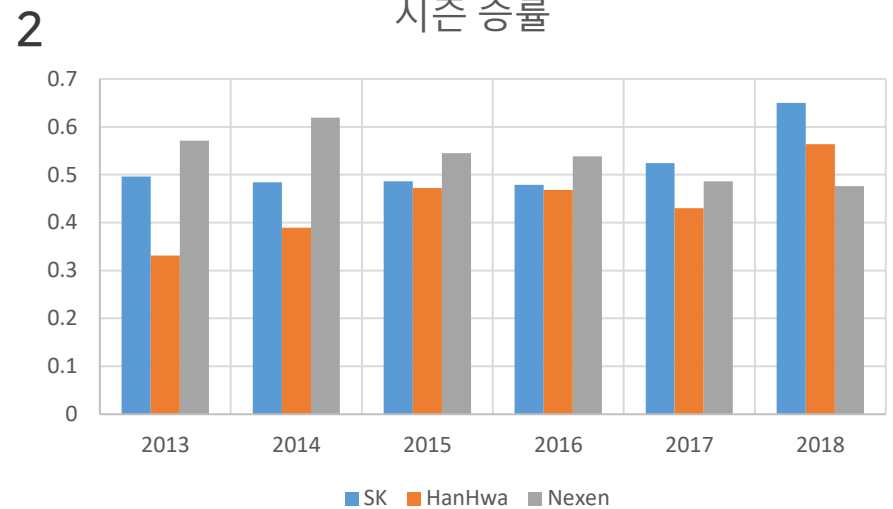
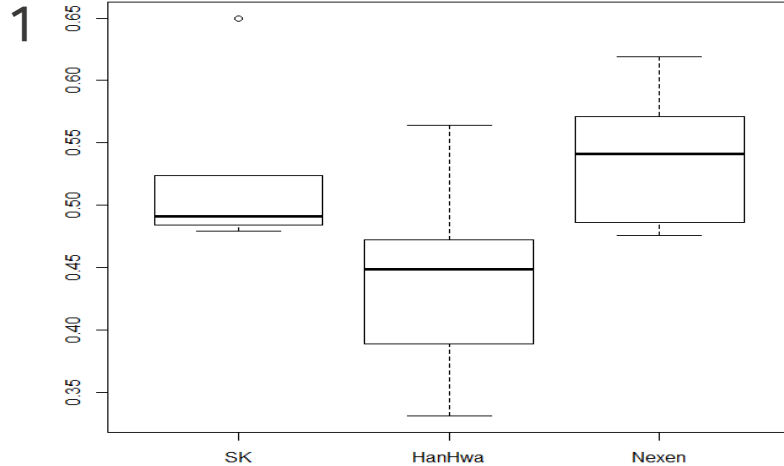
제품 대분류별 매출의 합계를 나타내는 그래프

다음의 차트 중 어떤 차트가 속도와 자동차 제동거리 관계 비교가 쉽나요?



속도에 따른 자동차 제동거리를 나타내는 그래프

## 다음의 차트 중 어떤 차트가 프로야구팀 시즌 별 승률 분석이 쉽나요?

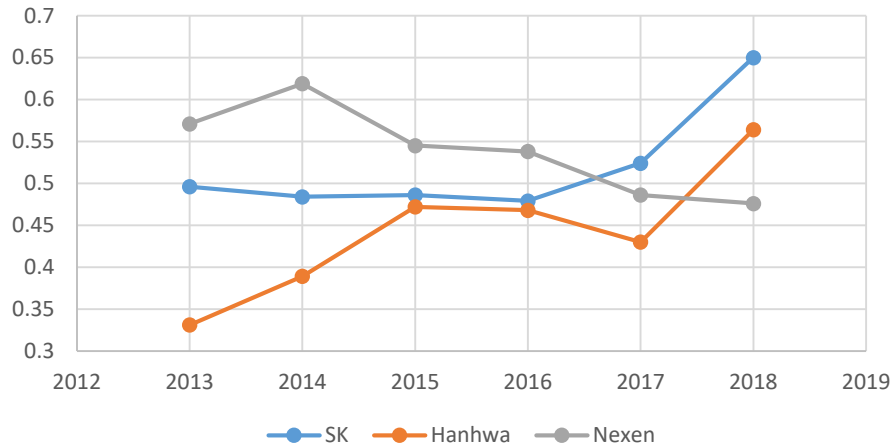


프로야구팀들의 시즌 별 승률을 나타내는 그래프

## 다음의 차트 중 어떤 차트가 프로야구팀 시즌 별 승률 분석이 쉽나요?

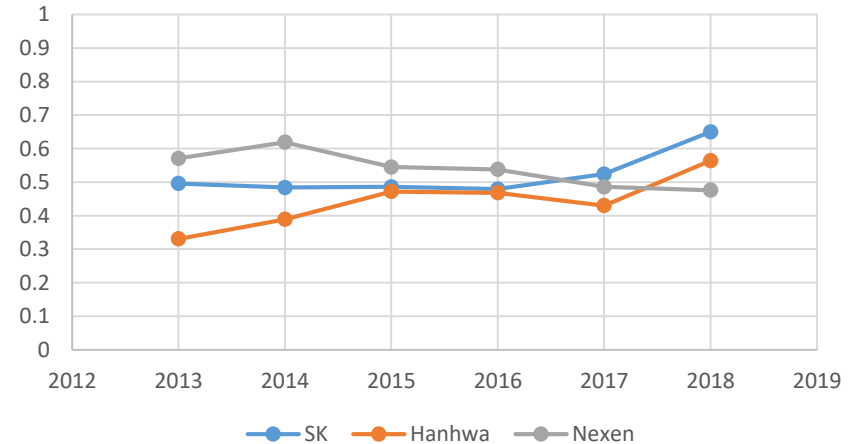
1

시즌 승률



2

시즌 승률



프로야구팀들의 시즌 별 승률을 나타내는 그래프



## 4. R 그래픽 함수



## 고수준 그래픽 함수(high level graphic functions)

- ✓ 그래픽 장치 위에 산점도나 막대그래프같이 독립된 시각화 표현을 하는 함수로 R에서 독립적으로 그래픽 장치를 활성화할 수 있다.

함수	기능
plot()	산점도 출력
barplot()	막대 차트 출력
pie()	파이 차트 출력
matplot()	다중 산점도 출력

## plot() 함수

✓ 데이터를 x-y 평면 상에 출력하는 함수

### 사용법

```
plot(x, y, type = 'type value', main='title', col=color)
```

# type : plot의 형태로 점, 선 등을 선택할 수 있다.

# main: 그래프의 제목 설정

# col : 그래프의 색상

### Type 옵션

p : 점(points), l : 선(lines), b : 점과 선(both points and lines), c : b옵션에서 점이 빠진 모습,

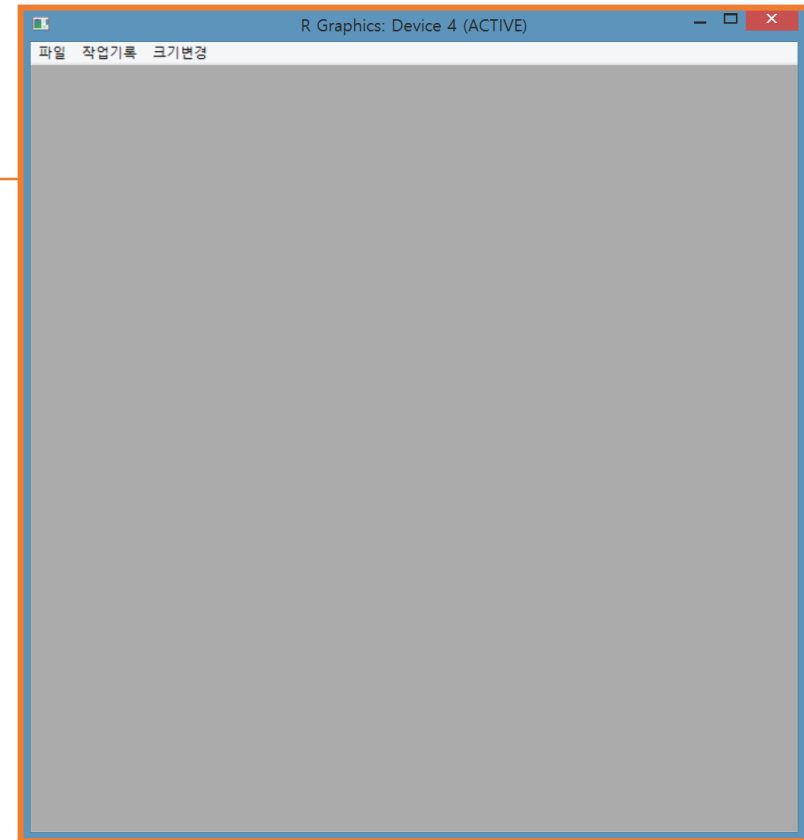
o : 겹친 점과 선(overplotted), h : 수직선 (high density), s : 수평선 우선의 계단 모양 (steps),

S : 수직선 우선의 계단 모양 (steps), n : 출력하지 않음 (no plotting)

```
# R graphic Device 실행  
x11()
```

```
# 2 X 3으로 화면 분할하기  
par(mfrow = c(2,3))
```

```
# type 옵션을 다르게 하면서 plot() 실행하기  
plot(0:6, 0:6, main="default")  
plot(0:6, 0:6, type="b", main="type = \"b\"")  
plot(0:6, 0:6, type="c", main="type = \"c\"")  
plot(0:6, 0:6, type="o", main="type = \"o\"")  
plot(0:6, 0:6, type="s", main="type = \"s\"")  
plot(0:6, 0:6, type="S", main="type = \"S\"")
```



```
# R graphic Device 실행  
x11()
```

```
# 2 X 3으로 화면 분할하기  
par(mfrow = c(2,3))
```

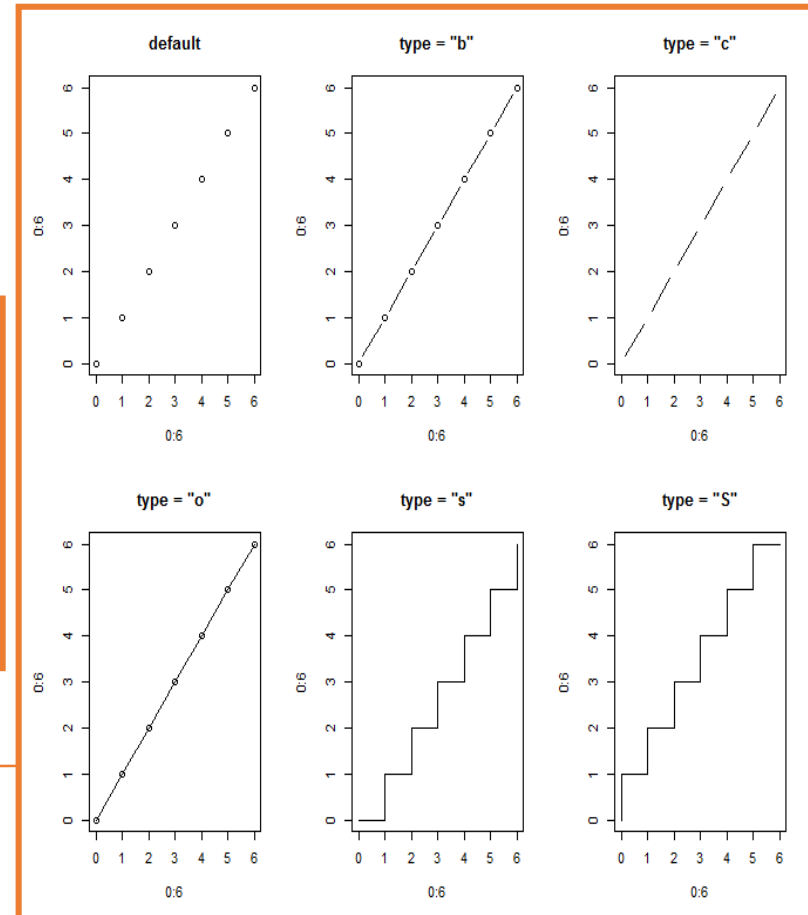
```
# type 옵션을 다르게 하면서 plot() 실행하기  
plot(0:6, 0:6, main="default")  
plot(0:6, 0:6, type="b", main="type = \"b\")  
plot(0:6, 0:6, type="c", main="type = \"c\")  
plot(0:6, 0:6, type="o", main="type = \"o\")  
plot(0:6, 0:6, type="s", main="type = \"s\")  
plot(0:6, 0:6, type="S", main="type = \"S\")
```

Graphic Device 화면을 2X3 화면으로 분할한다. 아래에서 실행하게 될 6개의 plot을 한번에 비교하기 위함이다.

```
# R graphic Device 실행  
x11()
```

```
# 2 X 3으로 화면 분할하기  
par(mfrow = c(2,3))
```

```
# type 옵션을 다르게 하면서 plot() 실행하기  
plot(0:6, 0:6, main="default")  
plot(0:6, 0:6, type="b", main="type = \"b\\\"")  
plot(0:6, 0:6, type="c", main="type = \"c\\\"")  
plot(0:6, 0:6, type="o", main="type = \"o\\\"")  
plot(0:6, 0:6, type="s", main="type = \"s\\\"")  
plot(0:6, 0:6, type="S", main="type = \"S\\\"")
```



## barplot() 함수

- ✓ 범주형 데이터의 수준별 도수(frequency)의 크기를 막대로 나타내는 그래프

### 사용법

`barplot(height, width = 1, beside = FALSE, main='title', col=NULL, ...)`

# height 인수 : 막대그래프로 표현할 데이터의 객체로 벡터나 행렬 입력 가능

# beside 인수 :

TRUE이면서 height가 행렬일 경우 -> 각 막대(Bar)가 행의 크기를 나타내는 막대 그래프

FALSE이면서 height가 행렬일 경우 -> 각 막대(Bar)가 열을 나타내는 막대 그래프

# col : 그래프의 색상

```
par(mfrow=c(1,1))
```

```
# 막대 그래프로 나타낼 객체 만들기
```

```
x <- c(38, 52, 24, 8, 3)
```

```
barplot(x)
```

```
# x 변수 이름 설정하기
```

```
names(x) <- c("Excellent", "Very Good", "Good", "Fair", "Poor")
```

```
barplot(x)
```

```
# 객체 만들기 2
```

```
y <- scan()
```

```
# Console창에 직접 입력
```

```
# 1 2 3 3 4 3 4 1 5 3 3 3 2 4 4
```

```
# 3 4 3 5 3 1 2 3 3 4 4 3 2 3 4
```

```
barplot(y)
```

```
# table 함수를 사용하여 범주화하기
```

```
barplot(table(y), xlab = "Beverage", ylab = "Frequency")
```

```
# 비율로 나타내기
```

```
barplot(table(y)/length(y), xlab = "Beverage", ylab = "Proportion")
```

```
# 객체 만들기 3
```

```
sales <- c(45, 44, 46)
```

```
names(sales) <- c("Park", "Kim", "Lee")
```

```
barplot(sales, main = "Sales", ylab = "Thousands")
```

```
# 범위 조절하기
```

```
barplot(sales, main = "Sales", ylab = "Thousands", ylim = c(42,46), xpd = FALSE)
```

화면 partition(분할)을 1 X 1 로 설정합니다.

```
par(mfrow=c(1,1))  
# 막대그래프로 나타낼 객체 만들기  
x <- c(38, 52, 24, 8, 3)  
barplot(x)
```

x : barplot으로 표현할 객체  
barplot(x) : 막대그래프 표현

```
# x 변수 이름 설정하기  
names(x) <- c("Excellent", "Very Good", "Good", "Fair", "Poor")  
barplot(x)
```

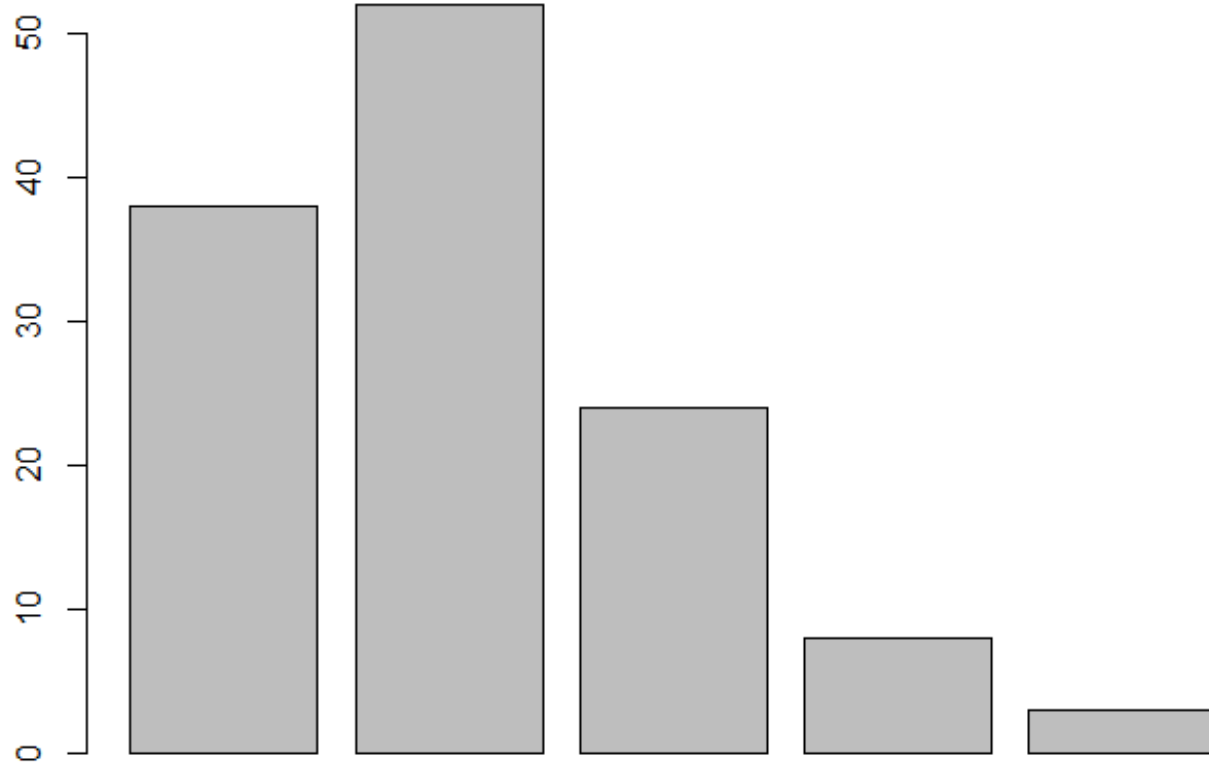
```
# 객체만들기 2  
y <- scan()  
# Console창에 직접 입력  
# 1 2 3 3 4 3 4 1 5 3 3 3 2 4 4  
# 3 4 3 5 3 1 2 3 3 4 4 3 2 3 4  
barplot(y)
```

```
# table 함수를 사용하여 범주화하기  
barplot(table(y), xlab = "Beverage", ylab = "Frequency")
```

```
# 비율로 나타내기  
barplot(table(y)/length(y), xlab = "Beverage", ylab = "Proportion")
```

```
# 객체만들기 3  
sales <- c(45, 44, 46)  
names(sales) <- c("Park", "Kim", "Lee")  
barplot(sales, main = "Sales", ylab = "Thousands")
```

```
# 범위 조절하기  
barplot(sales, main = "Sales", ylab = "Thousands", ylim = c(42,46), xpd = FALSE)
```



```
par(mfrow=c(1,1))  
# 막대 그래프로 나타낼 객체 만들기  
x <- c(38, 52, 24, 8, 3)  
barplot(x)
```

```
# x 변수 이름 설정하기  
names(x) <- c("Excellent", "Very Good", "Good", "Fair", "Poor")  
barplot(x)
```

```
# 객체 만들기 2  
y <- scan()  
# Console창에 직접 입력  
# 1 2 3 3 4 3 4 1 5 3 3 3 2 4 4  
# 3 4 3 5 3 1 2 3 3 4 4 3 2 3 4  
barplot(y)
```

```
# table 함수를 사용하여 범주화하기  
barplot(table(y), xlab = "Beverage", ylab = "Frequency")
```

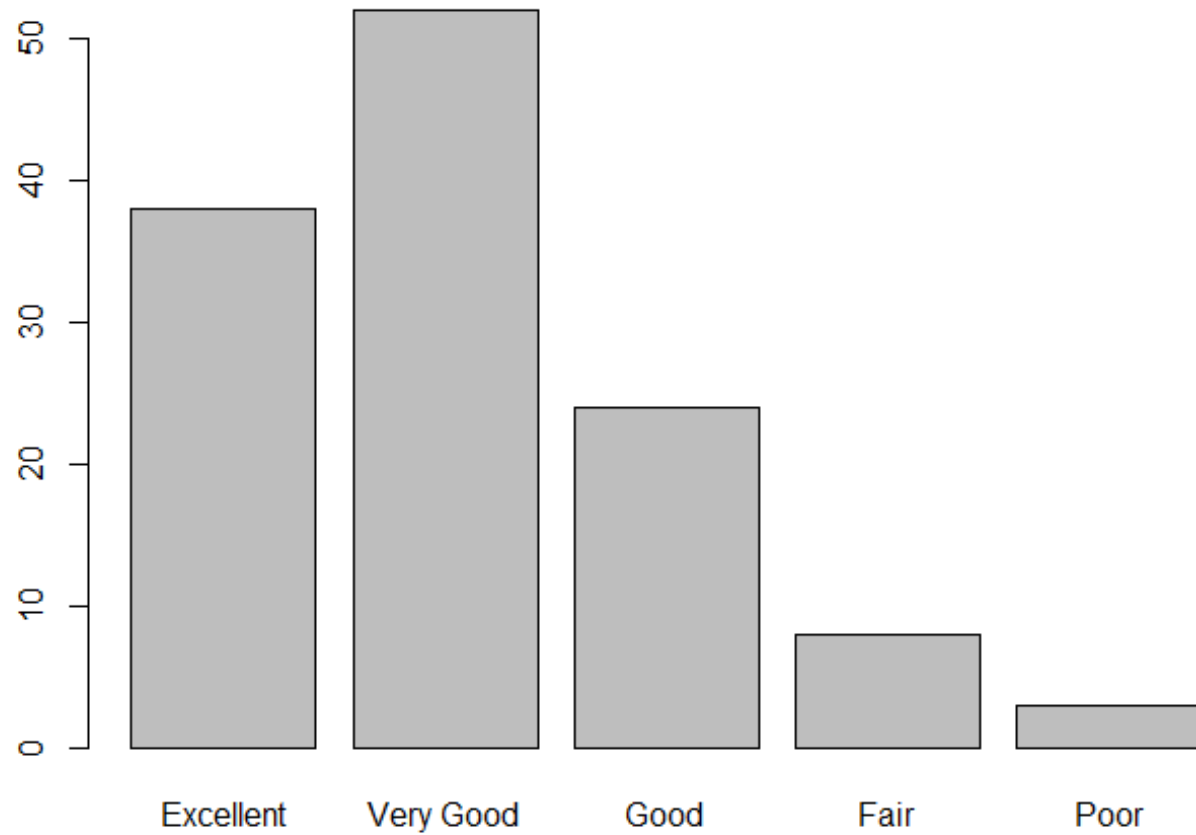
```
# 비율로 나타내기  
barplot(table(y)/length(y), xlab = "Beverage", ylab = "Proportion")
```

```
# 객체 만들기 3  
sales <- c(45, 44, 46)  
names(sales) <- c("Park", "Kim", "Lee")  
barplot(sales, main = "Sales", ylab = "Thousands")
```

```
# 범위 조절하기  
barplot(sales, main = "Sales", ylab = "Thousands", ylim = c(42,46), xpd = FALSE)
```

names(x) : x 객체의 각 데이터에 이름을 설정합니다.

barplot(x) : 막대그래프 표현



```
par(mfrow=c(1,1))
# 막대 그래프로 나타낼 객체 만들기
x <- c(38, 52, 24, 8, 3)
barplot(x)
```

```
# x 변수 이름 설정하기
names(x) <- c("Excellent", "Very Good", "Good", "Fair", "Poor")
barplot(x)
```

```
# 객체 만들기 2
y <- scan()
# Console창에 직접 입력
# 1 2 3 3 4 3 4 1 5 3 3 3 2 4 4
# 3 4 3 5 3 1 2 3 3 4 4 3 2 3 4
barplot(y)
```

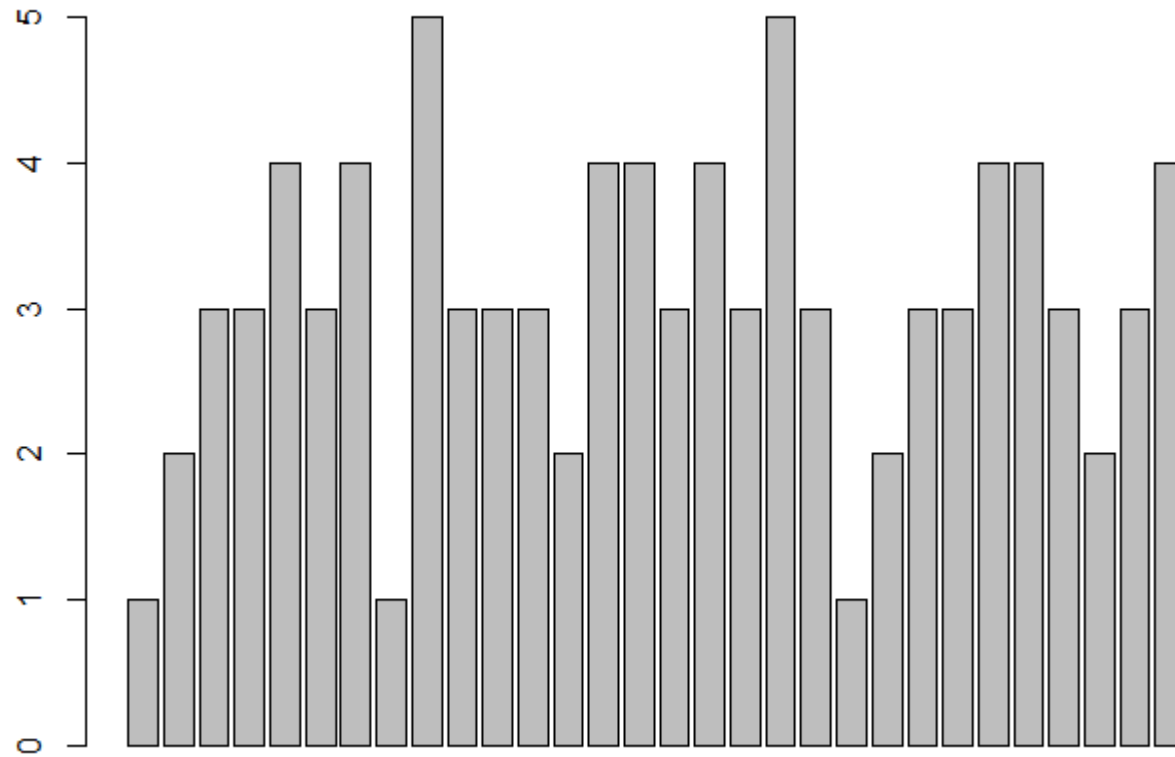
```
# table 함수를 사용하여 범주화하기
barplot(table(y), xlab = "Beverage", ylab = "Frequency")
# 비율로 나타내기
barplot(table(y)/length(y), xlab = "Beverage", ylab = "Proportion")
```

```
# 객체 만들기 3
sales <- c(45, 44, 46)
names(sales) <- c("Park", "Kim", "Lee")
barplot(sales, main = "Sales", ylab = "Thousands")
```

```
# 범위 조절하기
barplot(sales, main = "Sales", ylab = "Thousands", ylim = c(42,46), xpd = FALSE)
```

scan(): 데이터를 콘솔창에서 직접 입력할 수 있다.

barplot(y) : 막대그래프 표현



```
par(mfrow=c(1,1))  
# 막대 그래프로 나타낼 객체 만들기  
x <- c(38, 52, 24, 8, 3)  
barplot(x)
```

```
# x 변수 이름 설정하기  
names(x) <- c("Excellent", "Very Good", "Good", "Fair", "Poor")  
barplot(x)
```

```
# 객체 만들기 2  
y <- scan()  
# Console창에 직접 입력  
# 1 2 3 3 4 3 4 1 5 3 3 3 2 4 4  
# 3 4 3 5 3 1 2 3 3 4 4 3 2 3 4  
barplot(y)
```

```
# table 함수를 사용하여 범주화하기  
barplot(table(y), xlab = "Beverage", ylab = "Frequency")
```

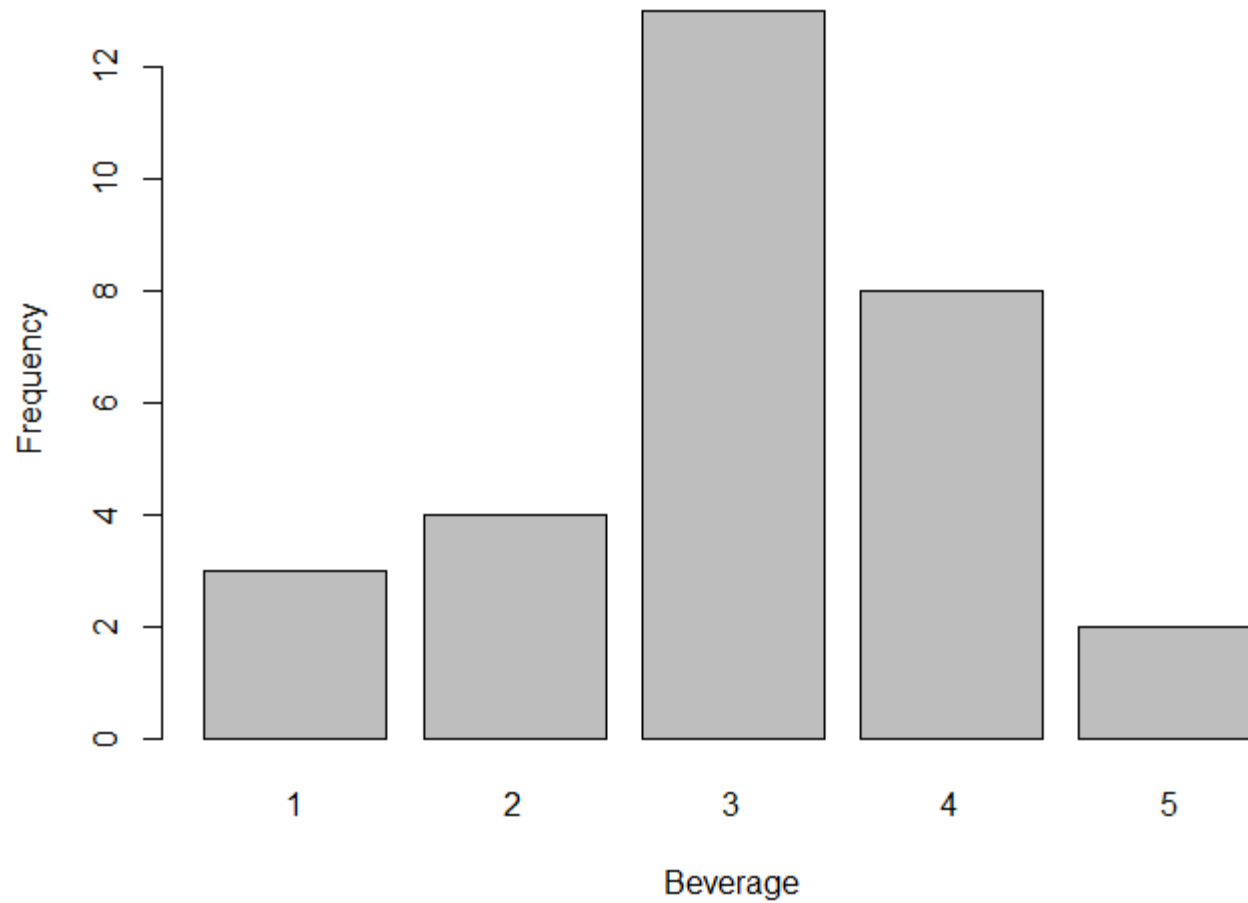
```
# 비율로 나타내기  
barplot(table(y)/length(y), xlab = "Beverage", ylab = "Proportion")
```

```
# 객체 만들기 3  
sales <- c(45, 44, 46)  
names(sales) <- c("Park", "Kim", "Lee")  
barplot(sales, main = "Sales", ylab = "Thousands")
```

```
# 범위 조절하기  
barplot(sales, main = "Sales", ylab = "Thousands", ylim = c(42,46), xpd = FALSE)
```

table(y) : 데이터의 도수를 표현

xlab, ylab : 축 이름 설정



```
par(mfrow=c(1,1))  
# 막대 그래프로 나타낼 객체 만들기  
x <- c(38, 52, 24, 8, 3)  
barplot(x)  
  
# x 변수 이름 설정하기  
names(x) <- c("Excellent", "Very Good", "Good", "Fair", "Poor")  
barplot(x)
```

```
# 객체 만들기 2  
y <- scan()  
# Console창에 직접 입력  
# 1 2 3 3 4 3 4 1 5 3 3 3 2 4 4  
# 3 4 3 5 3 1 2 3 3 4 4 3 2 3 4  
barplot(y)
```

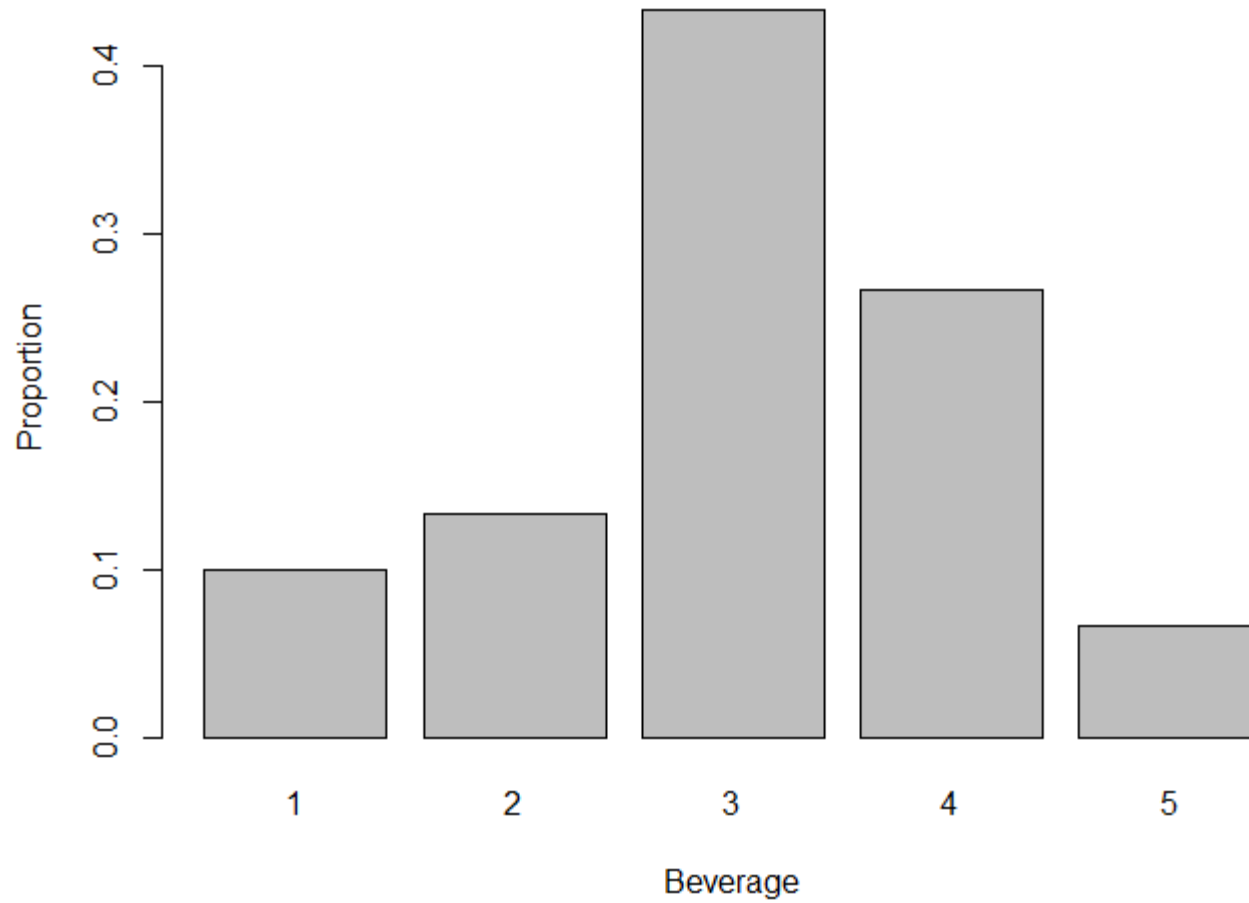
```
# table 함수를 사용하여 범주화하기  
barplot(table(y), xlab = "Beverage", ylab = "Frequency")
```

```
# 비율로 나타내기  
barplot(table(y)/length(y), xlab = "Beverage", ylab = "Proportion")
```

```
# 객체 만들기 3  
sales <- c(45, 44, 46)  
names(sales) <- c("Park", "Kim", "Lee")  
barplot(sales, main = "Sales", ylab = "Thousands")
```

```
# 범위 조절하기  
barplot(sales, main = "Sales", ylab = "Thousands", ylim = c(42,46), xpd = FALSE)
```

table(y) : 데이터의 도수를 표현  
length(y) : 데이터의 갯수(길이)



```
par(mfrow=c(1,1))
# 막대 그래프로 나타낼 객체 만들기
x <- c(38, 52, 24, 8, 3)
barplot(x)

# x 변수 이름 설정하기
names(x) <- c("Excellent", "Very Good", "Good", "Fair", "Poor")
barplot(x)

# 객체 만들기 2
y <- scan()
# Console창에 직접 입력
# 1 2 3 3 4 3 4 1 5 3 3 3 2 4 4
# 3 4 3 5 3 1 2 3 3 4 4 3 2 3 4
barplot(y)

# table 함수를 사용하여 범주화하기
barplot(table(y), xlab = "Beverage", ylab = "Frequency")

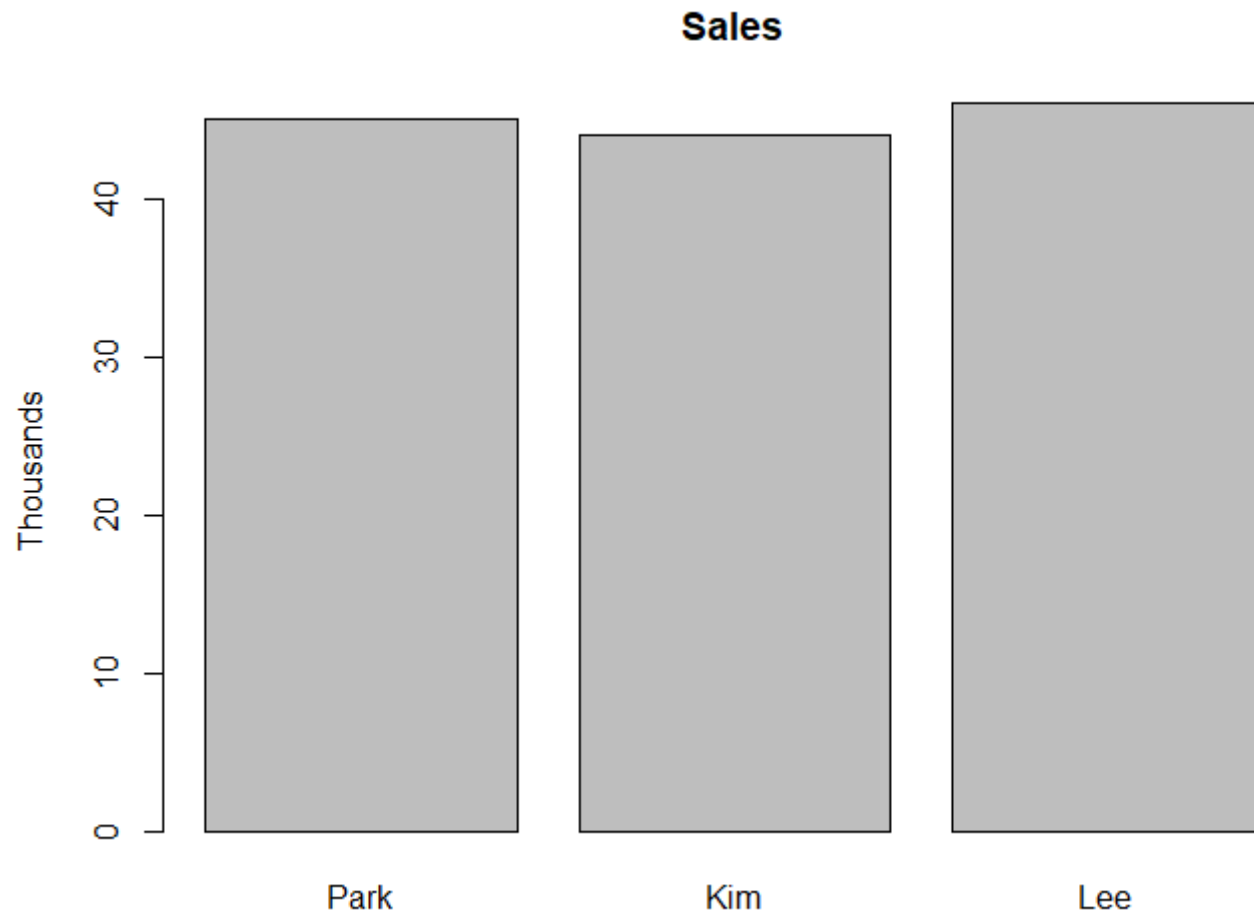
# 비율로 나타내기
barplot(table(y)/length(y), xlab = "Beverage", ylab = "Proportion")

# 객체 만들기 3
sales <- c(45, 44, 46)
names(sales) <- c("Park", "Kim", "Lee")
barplot(sales, main = "Sales", ylab = "Thousands")

# 범위 조절하기
barplot(sales, main = "Sales", ylab = "Thousands", ylim = c(42,46), xpd = FALSE)
```

sales: 데이터 객체

names(sales) : 데이터의 이름 설정



```

par(mfrow=c(1,1))
# 막대 그래프로 나타낼 객체 만들기
x <- c(38, 52, 24, 8, 3)
barplot(x)

# x 변수 이름 설정하기
names(x) <- c("Excellent", "Very Good", "Good", "Fair", "Poor")
barplot(x)

# 객체 만들기 2
y <- scan()
# Console창에 직접 입력
# 1 2 3 3 4 3 4 1 5 3 3 3 2 4 4
# 3 4 3 5 3 1 2 3 3 4 4 3 2 3 4
barplot(y)

# table 함수를 사용하여 범주화하기
barplot(table(y), xlab = "Beverage", ylab = "Frequency")

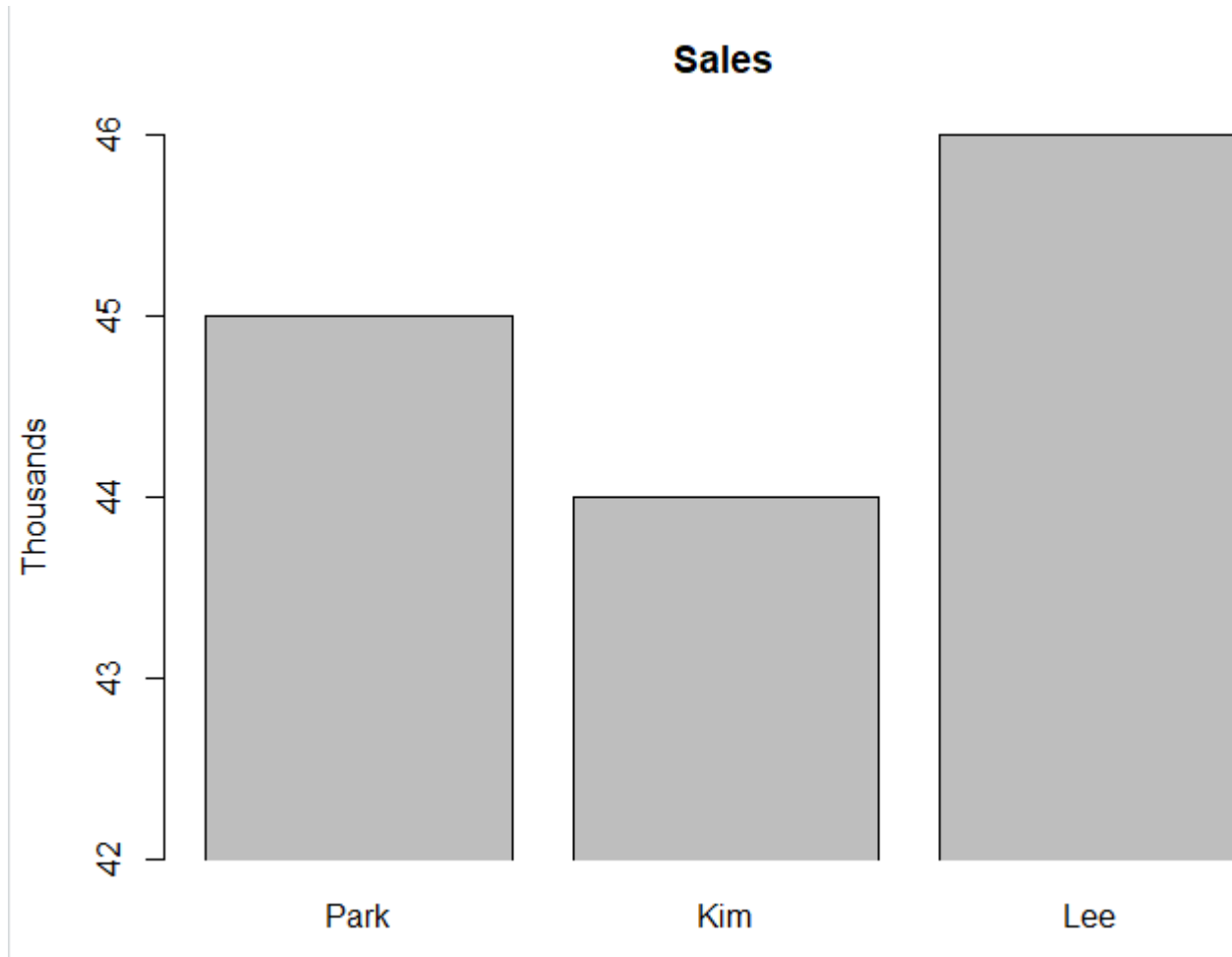
# 비율로 나타내기
barplot(table(y)/length(y), xlab = "Beverage", ylab = "Frequency")

# 객체 만들기 3
sales <- c(45, 44, 46)
names(sales) <- c("Park", "Kim", "Lee")
barplot(sales, main = "Sales", ylab = "Thousands")

# 범위 조절하기
barplot(sales, main = "Sales", ylab = "Thousands", ylim = c(42,46), xpd = FALSE)

```

ylim = c(42,46) : y 축 범위 설정  
xpd=FALSE : 막대의 벗어남 허용여부





## pie() 함수

데이터를 파이 차트(원 그래프)로 출력하는 함수

### 사용법

```
pie(x, labels = names(x), radius = 0.8, clockwise = FALSE, init.angle =  
if(clockwise) 90 else 0, density = NULL, angle = 45, col = NULL, ...)
```

# x : 음수나 0이 아닌 숫자형 벡터

# labels : 기본값으로 x 벡터의 이름이 사용, 새롭게 지정 가능

# radius : 파이의 반지름

# init.angle : 파이 차트가 시작되는 각도(clockwise가 TRUE면 90도 아니면 0도 )

# density : 파이 내부의 빗금을 표시하는 밀도

# angle : 파이 내부의 빗금 표시하는 기울기

# col : 파이 내부의 색상

## read.table() : txt파일 불러오기

```
# 데이터 불러오기
score <- read.table("score.txt", header=T)
```

```
# 파이 차트 그리기
pie(score$국어, labels = paste(score$성명, "-", score$국어), col = rainbow(10), clockwise = TRUE)
```

```
# 두줄로 된 label 나태내기
pie(score$국어, labels = paste(score$성명, "\n", "(,score$국어,")"), col = rainbow(10), clockwise = TRUE)
```

```
## googlevis 패키지 사용하기
install.packages("googlevis")
library(googlevis)
```

```
# 색상 설정
buildcolors <- function(color_count){
  colors <- rainbow(color_count)
  colors <- substring(colors, 1, 7)
  colors <- paste(colors, collapse=" ", sep=" ")
  colors <- paste("'", colors, "'", sep=" ")
  colors <- paste("[", colors, "]", sep=" ")
  return(colors)
}
```

```
cols <- buildcolors(10)
```

```
# 파이 차트 구현하기
# 인터넷 익스플로러에서 실행
pie <- gvisPieChart(data.frame(score$성명, score$국어),
  option = list(width=600, height=600, title="국어 성적",
    colors= cols, piesliceText="label", pieHole=0.5), chartid="donut")
```

```
plot(pie)
```

```
# 3D 파이 차트 만들기
```

```
pie2 <- gvisPieChart(data.frame(score$성명, score$국어),
  option=list(width=600, height=600, title="국어 성적",
    colors=cols, piesliceText="value", pieHole=0.5, is3D=TRUE), chartid="donut")
```

```
plot(pie2)
```

	학번	성명	국어	수학	영어
1	1	길동	8	7	2
2	2	철수	6	5	7
3	3	영수	9	4	9
4	4	미자	4	5	7
5	5	동화	5	8	9
6	6	은미	5	9	6
7	7	인수	4	7	10
8	8	영희	10	10	9
9	9	미정	7	5	4
10	10	효선	7	7	5

▲	학번 ▲	성명 ▲	국어 ▲	수학 ▲	영어 ▲
1	1	길동	8	7	2
2	2	철수	6	5	7
3	3	영수	9	4	9
4	4	미자	4	5	7
5	5	동화	5	8	9
6	6	은미	5	9	6
7	7	인수	4	7	10
8	8	영희	10	10	9
9	9	미정	7	5	4
10	10	효선	7	7	5

```
# 데이터 불러오기
score <- read.table("score.txt", header=T)

# 파이 차트 그리기
pie(score$국어, labels = paste(score$성명, "-", score$국어), col = rainbow(10), clockwise = TRUE)

# 두줄로 된 label 나태내기
pie(score$국어, labels = paste(score$성명, "\n", "(" ,score$국어, ")"), col = rainbow(10), clockwise = TRUE)

## googlevis 패키지 사용하기
install.packages("googlevis")
library(googlevis)

# 색상 설정
buildcolors <- function(color_count){
  colors <- rainbow(color_count)
  colors <- substring(colors, 1, 7)
  colors <- paste(colors, collapse=" ", sep=" ")
  colors <- paste(" ", colors, " ", sep=" ")
  colors <- paste("[", colors, "]", sep=" ")
  return(colors)
}

cols <- buildcolors(10)

# 파이 차트 구현하기
# 인터넷 익스플로러에서 실행
pie <- gvisPieChart(data.frame(score$성명, score$국어),
  option = list(width=600, height=600, title="국어 성적",
    colors= cols, piesliceText="label", pieHole=0.5), chartid="donut")

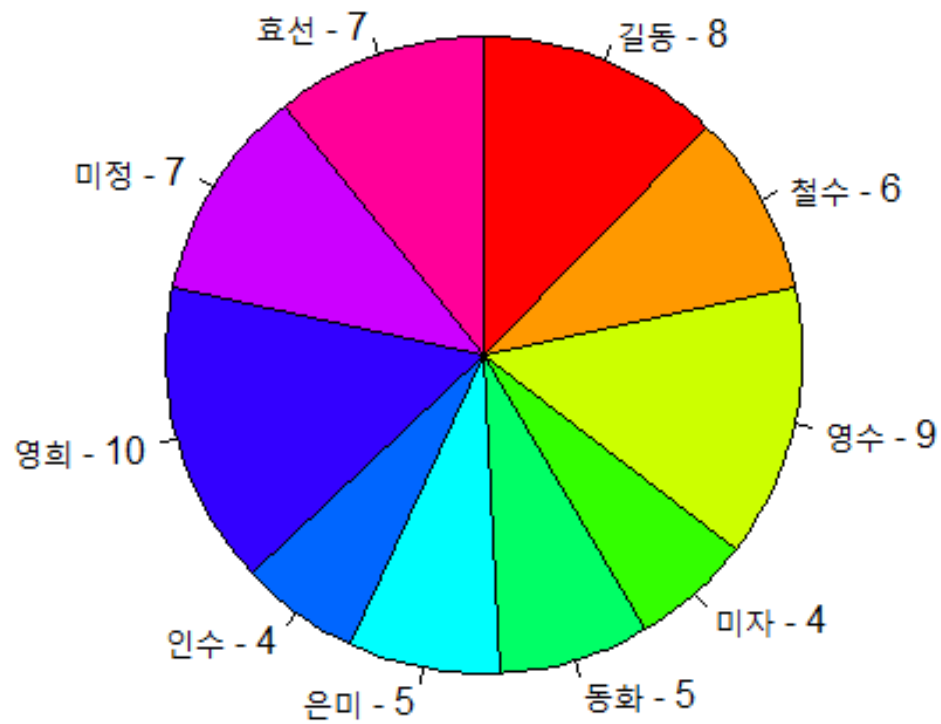
plot(pie)

# 3D 파이 차트 만들기
pie2 <- gvisPieChart(data.frame(score$성명, score$국어),
  option=list(width=600, height=600, title="국어 성적",
    colors=cols, piesliceText="value", pieHole=0.5, is3D=TRUE), chartid="donut")

plot(pie2)
```

paste() : 문자열을 붙이는 함수

clockwise = TRUE : 파이 차트 시작각도를 90도로 설정



```
# 데이터 불러오기
score <- read.table("score.txt", header=T)

# 파이 차트 그리기
pie(score$국어, labels = paste(score$성명, "-", score$국어), col = rainbow(10), clockwise = TRUE)

# 두 줄로 된 label 나태내기
pie(score$국어, labels = paste(score$성명, "\n", "(" ,score$국어, ")"), col = rainbow(10), clockwise = TRUE)

## googlevis 패키지 사용하기
install.packages("googlevis")
library(googlevis)

# 색상 설정
buildcolors <- function(color_count){
  colors <- rainbow(color_count)
  colors <- substring(colors, 1, 7)
  colors <- paste(colors, collapse=" ", sep=" ")
  colors <- paste(" ", colors, " ", sep=" ")
  colors <- paste("[", colors, "]", sep=" ")
  return(colors)
}

cols <- buildcolors(10)

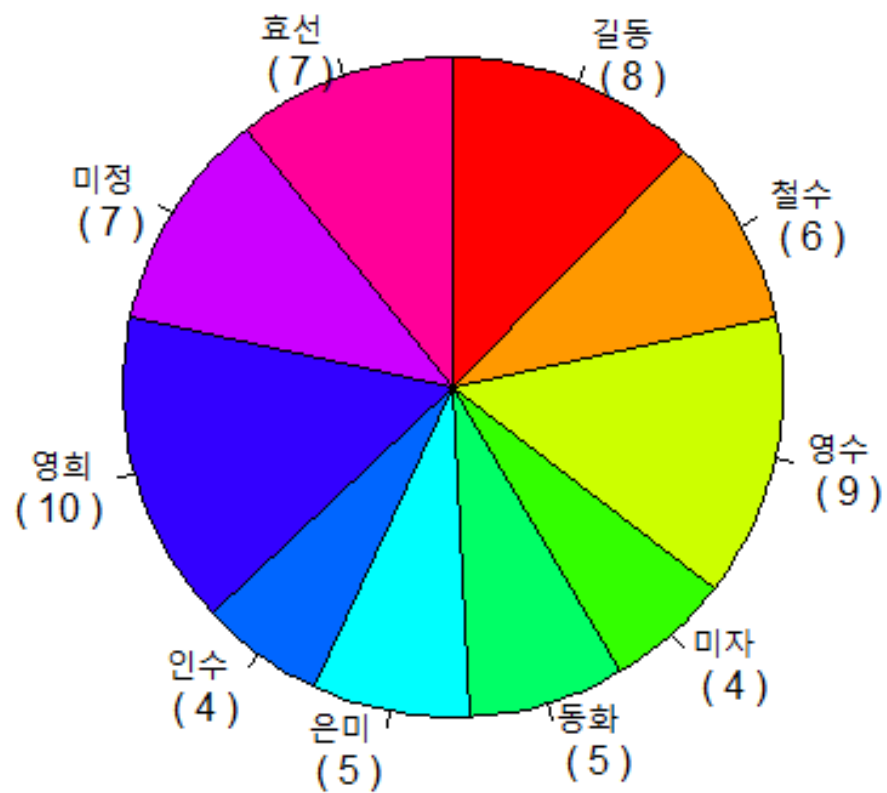
# 파이 차트 구현하기
# 인터넷 익스플로러에서 실행
pie <- gvisPieChart(data.frame(score$성명, score$국어),
  option = list(width=600, height=600, title="국어 성적",
    colors= cols, piesliceText="label", pieHole=0.5), chartid="donut")

plot(pie)

# 3D 파이 차트 만들기
pie2 <- gvisPieChart(data.frame(score$성명, score$국어),
  option=list(width=600, height=600, title="국어 성적",
    colors=cols, piesliceText="value", pieHole=0.5, is3D=TRUE), chartid="donut")

plot(pie2)
```

\n : 줄바꿈



```
# 데이터 불러오기
score <- read.table("score.txt", header=T)

# 파이차트 그리기
pie(score$국어, labels = paste(score$성명, "-", score$국어), col = rainbow(10), clockwise = TRUE)

# 두줄로 된 label 나태내기
pie(score$국어, labels = paste(score$성명, "\n", "(" ,score$국어, ")"), col = rainbow(10), clockwise = TRUE)

## googlevis 패키지 사용하기
install.packages("googlevis")
library(googlevis)

# 색상 설정
buildcolors <- function(color_count){
  colors <- rainbow(color_count)
  colors <- substring(colors, 1, 7)
  colors <- paste(colors, collapse=" ", sep=" ")
  colors <- paste("'", colors, "'", sep=" ")
  colors <- paste("[", colors, "]", sep=" ")
  return(colors)
}

cols <- buildcolors(10)

# 파이 차트 구현하기
# 인터넷 익스플로러에서 실행
pie <- gvisPieChart(data.frame(score$성명, score$국어),
  option = list(width=600, height=600, title="국어 성적",
    colors= cols, piesliceText="label", pieHole=0.5), chartid="donut")

plot(pie)

# 3D 파이 차트 만들기
pie2 <- gvisPieChart(data.frame(score$성명, score$국어),
  option=list(width=600, height=600, title="국어 성적",
    colors=cols, piesliceText="value", pieHole=0.5, is3D=TRUE), chartid="donut")

plot(pie2)
```

googleVis : R과 Google Chart를 연동하는 패키지

```
# 데이터 불러오기
score <- read.table("score.txt", header=T)

# 파이 차트 그리기
pie(score$국어, labels = paste(score$성명, "-", score$국어), col = rainbow(10), clockwise = TRUE)

# 두 줄로 된 label 나태내기
pie(score$국어, labels = paste(score$성명, "\n", "(" ,score$국어, ")"), col = rainbow(10), clockwise = TRUE)

## googlevis 패키지 사용하기
install.packages("googlevis")
library(googlevis)

# 색상 설정
buildcolors <- function(color_count){
  colors <- rainbow(color_count)
  colors <- substring(colors, 1, 7)
  colors <- paste(colors, collapse=" ", sep=" ")
  colors <- paste("(", colors, ")", sep=" ")
  colors <- paste("[", colors, "]", sep=" ")
  return(colors)
}

cols <- buildcolors(10)

# 파이 차트 구현하기
# 인터넷 익스플로러에서 실행
pie <- gvisPieChart(data.frame(score$성명, score$국어),
  option = list(width=600, height=600, title="국어 성적",
    colors= cols, piesliceText="label", pieHole=0.5), chartid="donut")

plot(pie)

# 3D 파이 차트 만들기
pie2 <- gvisPieChart(data.frame(score$성명, score$국어),
  option=list(width=600, height=600, title="국어 성적",
    colors=cols, piesliceText="value", pieHole=0.5, is3D=TRUE), chartid="donut")

plot(pie2)
```

색상 설정

```
# 데이터 불러오기
score <- read.table("score.txt", header=T)

# 파이차트 그리기
pie(score$국어, labels = paste(score$성명, "-", score$국어), col = rainbow(10), clockwise = TRUE)

# 두줄로 된 label 나태내기
pie(score$국어, labels = paste(score$성명, "\n", "(" ,score$국어, ")"), col = rainbow(10), clockwise = TRUE)

## googlevis 패키지 사용하기
install.packages("googlevis")
library(googlevis)

# 색상 설정
buildcolors <- function(color_count){
  colors <- rainbow(color_count)
  colors <- substring(colors, 1, 7)
  colors <- paste(colors, collapse=" ", "")
  colors <- paste("'", colors, "'", sep="")
  colors <- paste("[", colors, "]", sep="")
  return(colors)
}

cols <- buildcolors(10)

# 파이 차트 구현하기
# 인터넷 익스플로어에서 실행
pie <- gvisPieChart(data.frame(score$성명, score$국어),
                    option = list(width=600, height=600, title="국어 성적",
                                   colors= cols, piesliceText="label", pieHole=0.5), chartid="donut")
plot(pie)

# 3D 파이 차트 만들기
pie2 <- gvisPieChart(data.frame(score$성명, score$국어),
                    option=list(width=600, height=600, title="국어 성적",
                                   colors=cols, piesliceText="value", pieHole=0.5, is3D=TRUE), chartid="donut")
plot(pie2)
```

한글 인코딩 문제로 인터넷 익스플로어  
환경에서 실행 권장

## 국어성적



```
# 데이터 불러오기
score <- read.table("score.txt", header=T)

# 파이 차트 그리기
pie(score$국어, labels = paste(score$성명, "-", score$국어), col = rainbow(10), clockwise = TRUE)

# 두 줄로 된 label 나태내기
pie(score$국어, labels = paste(score$성명, "\n", "(" ,score$국어, ")"), col = rainbow(10), clockwise = TRUE)

## googlevis 패키지 사용하기
install.packages("googlevis")
library(googlevis)

# 색상 설정
buildcolors <- function(color_count){
  colors <- rainbow(color_count)
  colors <- substring(colors, 1, 7)
  colors <- paste(colors, collapse="',"")
  colors <- paste(""," ", colors, ""," ", sep="")
  colors <- paste("[", colors, "]", sep="")
  return(colors)
}
```

is3D = TRUE : 파이 차트를 3D 그래프로 표현

```
cols <- buildcolors(10)

# 파이 차트 구현하기
# 인터넷 익스플로러에서 실행
pie <- gvisPieChart(data.frame(score$성명, score$국어),
  option = list(width=600, height=600, title="국어 성적",
    colors= cols, piesliceText="label", pieHole=0.5), chartid="donut")

plot(pie)
```

```
# 3D 파이 차트 만들기
pie2 <- gvisPieChart(data.frame(score$성명, score$국어),
  option=list(width=600, height=600, title="국어 성적",
    colors=cols, piesliceText="value", pieHole=0.5, is3D=TRUE), chartid="donut")

plot(pie2)
```

## 국어성적



## (참고) 버블 차트

```
# 버블 차트  
library(googlevis)  
Fruits
```

googleVis 패키지의 내장 데이터 Fruits 사용

```
bubble <- gvisBubbleChart(Fruits, idvar="Fruit", xvar="Sales", yvar="Expenses", colorvar="Year",  
                           sizevar="Profit", options=list(height=600, weight=200))  
plot(bubble)
```

```
> Fruits
```

	Fruit	Year	Location	Sales	Expenses	Profit	Date
1	Apples	2008	West	98	78	20	2008-12-31
2	Apples	2009	West	111	79	32	2009-12-31
3	Apples	2010	West	89	76	13	2010-12-31
4	Oranges	2008	East	96	81	15	2008-12-31
5	Bananas	2008	East	85	76	9	2008-12-31
6	Oranges	2009	East	93	80	13	2009-12-31
7	Bananas	2009	East	94	78	16	2009-12-31
8	Oranges	2010	East	98	91	7	2010-12-31
9	Bananas	2010	East	81	71	10	2010-12-31

## (참고) 버블 차트

# 버블 차트

```
library(googlevis)
```

```
Fruits
```

```
bubble <- gvisBubbleChart(Fruits, idvar="Fruit", xvar="Sales", yvar="Expenses", colorvar="Year",  
                           sizevar="Profit", options=list(height=600, weight=200))
```

```
plot(bubble)
```

idvar : 버블에 표시할 이름

xvar : x축에 plotting될 수치형 벡터

yvar : y축에 plotting될 수치형 벡터

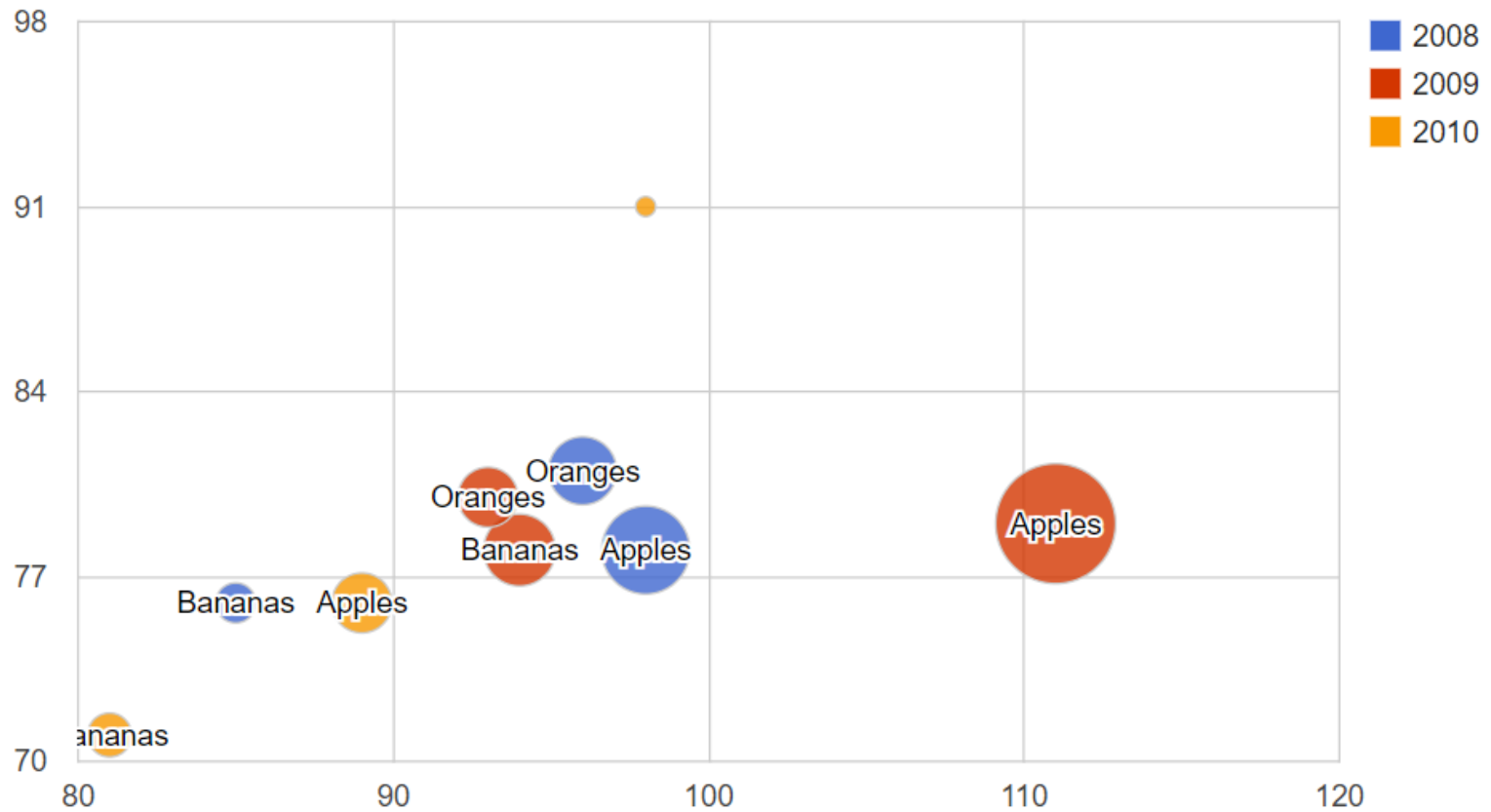
Colorvar : 컬러 변수로 년도(Year)설정

sizevar : 크기 변수를 이익(Profit)으로 설정

## (참고) 버블 차트

```
# 버블 차트
library(googlevis)
Fruits
bubble <- gvisBubbleChart(Fruits, idvar="Fruit", xvar="Sales", yvar="Expenses", colorvar="Year",
                           sizevar="Profit", options=list(height=600, weight=200))
plot(bubble)
```

<http://127.0.0.1:29348/custom/googleVis/BubbleChartID3d701a31286c.html> 그래프 구현





## matplot() 함수

하나의 x 벡터 값에 대하여 여러 개의 y 벡터 값을 가지는 산점도를 그리는 함수

Ex) `matplot(x, y, type, lty, lwd, pch, col, cex, ...)`

# x, y : 산점도를 그리기 위한 벡터 또는 행렬의 값으로 행의 개수는 모두 같아야 한다. 만일 둘 중에 하나만 주어지는 경우 주어진 값은 y로 해석하며 x는 1:n으로 설정한다.

# type : 산점도를 나타내는 것으로 점의 경우는 'p', 선의 경우는 'l' 등

# lty, lwd : 선의 종류 및 굵기

# pch : 점을 찍을 경우 점의 종류

# col : 점 또는 선에 사용할 색상 설정

# cex : 점의 크기

```
# x 변수 벡터 만들기
Year <- 2013:2018

# SK 시즌별 승률
SK <- c(0.496, 0.484, 0.486, 0.479, 0.524, 0.650)

# 한화 시즌별 승률
HanHwa <- c(0.331, 0.389, 0.472, 0.468, 0.430, 0.564)

# 넥센 시즌별 승률
Nexen<- c(0.571, 0.619, 0.545, 0.538, 0.486, 0.476)

# y 변수 데이터 프레임 만들기
Winning_Rate <- data.frame(SK, HanHwa, Nexen)
x11()

# 그래프로 나타내기
matplot(Year, Winning_Rate, type='o',
        pch=c(1,2,3), col=c("red","green", "blue"))

# legend 함수로 범례 추가하기
Team=c("SK", "HanHwa", "Nexen")
legend("top", Team,col=c("red","green","blue"),
      pch=c(1,2,3), title="시즌별 승률")
```

하나의 x 벡터 값으로 2013년부터  
2018년까지 숫자형 벡터를 만듦

```
# x 변수 벡터 만들기
Year <- 2013:2018

# SK 시즌별 승률
SK <- c(0.496, 0.484, 0.486, 0.479, 0.524, 0.650)

# 한화 시즌별 승률
HanHwa <- c(0.331, 0.389, 0.472, 0.468, 0.430, 0.564)

# 넥센 시즌별 승률
Nexen <- c(0.571, 0.619, 0.545, 0.538, 0.486, 0.476)

# y 변수 데이터 프레임 만들기
Winning_Rate <- data.frame(SK, HanHwa, Nexen)
x11()

# 그래프로 나타내기
matplot(Year, Winning_Rate, type='o',
        pch=c(1,2,3), col=c("red", "green", "blue"))

# legend 함수로 범례 추가하기
Team=c("SK", "HanHwa", "Nexen")
legend("top", Team, col=c("red", "green", "blue"),
      pch=c(1,2,3), title="시즌별 승률")
```

2013시즌부터 현재 진행되고 있는  
2018시즌까지 각 팀의 승률을 벡터  
로 만들었다.

출처 : 네이버 스포츠

```
# x 변수 벡터 만들기
Year <- 2013:2018

# SK 시즌별 승률
SK <- c(0.496, 0.484, 0.486, 0.479, 0.524, 0.650)

# 한화 시즌별 승률
HanHwa <- c(0.331, 0.389, 0.472, 0.468, 0.430, 0.564)

# 넥센 시즌별 승률
Nexen<- c(0.571, 0.619, 0.545, 0.538, 0.486, 0.476)

# y 변수 데이터 프레임 만들기
Winning_Rate <- data.frame(SK, HanHwa, Nexen)
x11()

# 그래프로 나타내기
matplot(Year, Winning_Rate, type='o',
        pch=c(1,2,3), col=c("red","green", "blue"))

# legend 함수로 범례 추가하기
Team=c("SK", "HanHwa", "Nexen")
legend("top", Team,col=c("red","green","blue"),
      pch=c(1,2,3), title="시즌별 승률")
```

여러 개의 y 벡터 값을 한번에 표현하기 위해 데이터 프레임을 만들었다.

```
# x 변수 벡터 만들기
Year <- 2013:2018

# SK 시즌별 승률
SK <- c(0.496, 0.484, 0.486, 0.479, 0.524, 0.650)

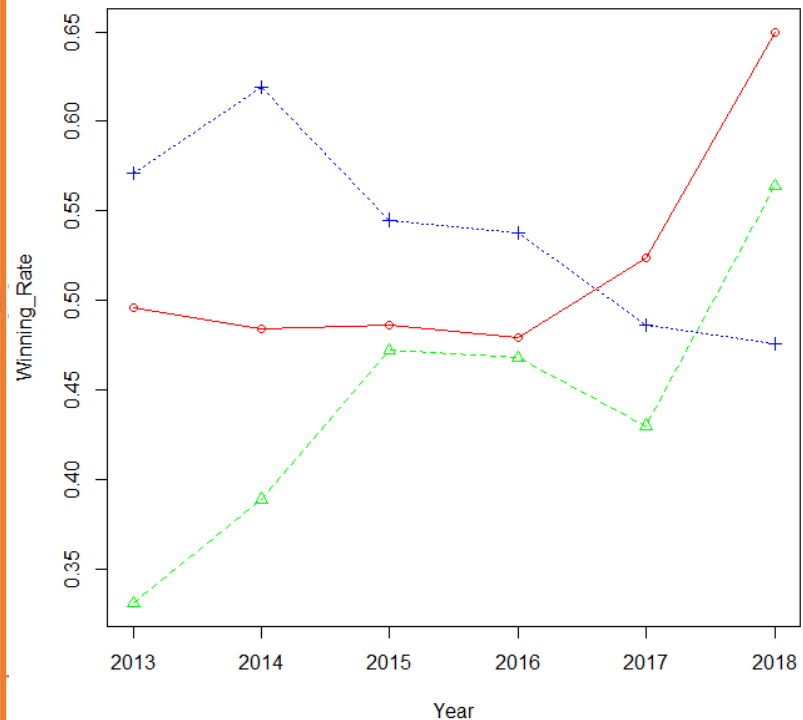
# 한화 시즌별 승률
HanHwa <- c(0.331, 0.389, 0.472, 0.468, 0.430, 0.564)

# 넥센 시즌별 승률
Nexen <- c(0.571, 0.619, 0.545, 0.538, 0.486, 0.476)

# y 변수 데이터 프레임 만들기
Winning_Rate <- data.frame(SK, HanHwa, Nexen)
x11()

# 그래프로 나타내기
matplot(Year, Winning_Rate, type='o',
        pch=c(1,2,3), col=c("red","green", "blue"))

# legend 함수로 범례 추가하기
Team=c("SK", "HanHwa", "Nexen")
legend("top", Team,col=c("red","green","blue"),
      pch=c(1,2,3), title="시즌별 승률")
```



```
# x 변수 벡터 만들기
Year <- 2013:2018

# SK 시즌별 승률
SK <- c(0.496, 0.484, 0.486, 0.479, 0.524, 0.650)

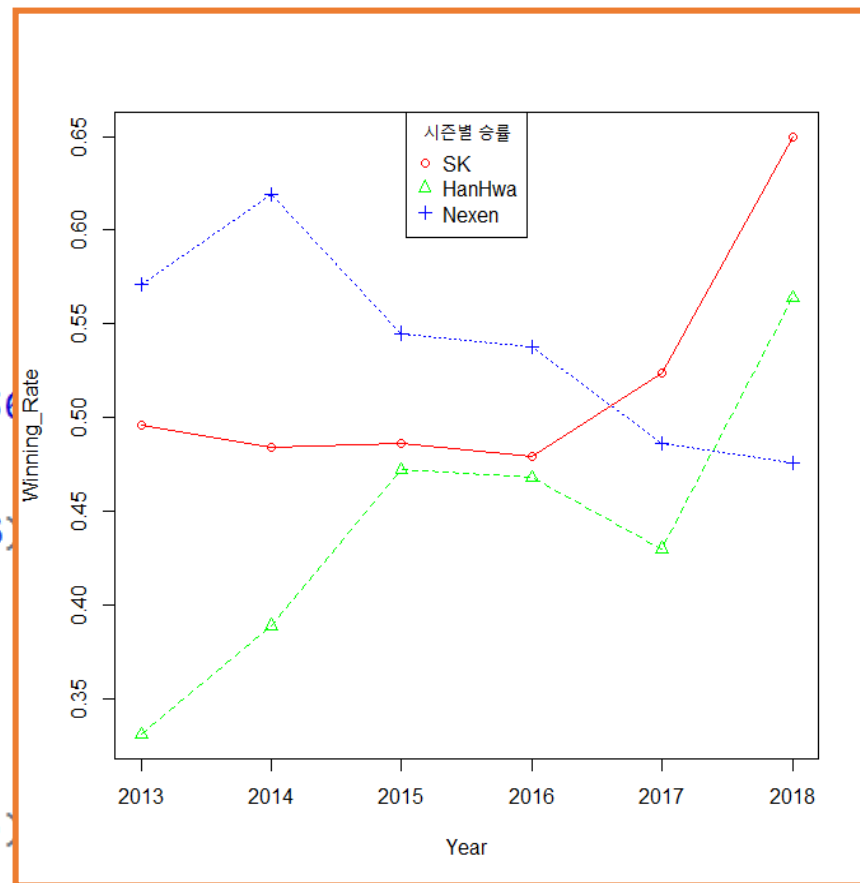
# 한화 시즌별 승률
HanHwa <- c(0.331, 0.389, 0.472, 0.468, 0.430, 0.564)

# 넥센 시즌별 승률
Nexen <- c(0.571, 0.619, 0.545, 0.538, 0.486, 0.476)

# y 변수 데이터 프레임 만들기
Winning_Rate <- data.frame(SK, HanHwa, Nexen)
x11()

# 그래프로 나타내기
matplot(Year, Winning_Rate, type='o',
        pch=c(1,2,3), col=c("red","green", "blue"))
```

```
# legend 함수로 범례 추가하기
Team=c("SK", "HanHwa", "Nexen")
legend("top", Team,col=c("red","green","blue"),
      pch=c(1,2,3), title="시즌별 승률")
```



# 저수준 그래픽 함수(low level graphic functions)

함수	기능
points()	지정한 좌표에 점을 찍는 함수
abline()	$y=a+bx$ 의 직선을 그리는 함수
legend()	범례를 출력하는 함수
text()	Plot 영역의 (x,y) 좌표에 문자를 출력하는 함수



## abline() 함수

x-y 평면에  $y=a+bx$ 의 직선,  $y=h$ ,  $x=v$ 를 그리는 함수

Ex) `abline(a, b, lty, col, other options)` #  $y = a + bx$

`abline(h = a, lty, col, other options)` #  $y = a$

`abline(v = b, lty, col, other options)` #  $x = b$

`abline(lm object)` # 회귀 직선

# `lty` : line type으로 1-solid line, 2-dashed line 등

# `col` : 직선의 색상

```
# R 내장 cars 데이터
```

```
cars
```

```
cars[1:4,]
```

```
#회귀 모형 적합
```

```
z <- lm(dist ~ speed, data = cars)
```

```
summary(z)
```

```
# cars 데이터를 산점도로 나타내기
```

```
x11()
```

```
par(mfrow=c(1,1))
```

```
plot(cars, main = "abline")
```

```
# horizontal
```

```
abline(h = 20)
```

```
abline(h = 30)
```

```
# vertical
```

```
abline(v = 20, col = 'blue')
```

```
# y = a + bx
```

```
abline(a = 40, b = 4, col = 'red')
```

```
# reg 인수
```

```
abline(z, lty = 2, lwd = 2, col = 'green')
```

```
# coef 인수
```

```
abline(z$coef, lty = 3, lwd = 2, col = 'red')
```

```
> cars[1:4,]  
  speed dist  
1      4    2  
2      4   10  
3      7    4  
4      7   22
```

```
# R 내장 cars 데이터
```

```
cars
```

```
cars[1:4,]
```

```
#회귀 모형 적합
```

```
z <- lm(dist ~ speed, data = cars)
summary(z)
```

```
# cars 데이터를 산점도로 나타내기
```

```
x11()
```

```
par(mfrow=c(1,1))
```

```
plot(cars, main = "abline")
```

```
# horizontal
```

```
abline(h = 20)
```

```
abline(h = 30)
```

```
# vertical
```

```
abline(v = 20, col = 'blue')
```

```
# y = a + bx
```

```
abline(a = 40, b = 4, col = 'red')
```

```
# reg 인수
```

```
abline(z, lty = 2, lwd = 2, col = 'green')
```

```
# coef 인수
```

```
abline(z$coef, lty = 3, lwd = 2, col = 'red')
```

```
> summary(z)
```

```
Call:
```

```
lm(formula = dist ~ speed, data = cars)
```

```
Residuals:
```

	Min	1Q	Median	3Q	Max
	-29.069	-9.525	-2.272	9.215	43.201

```
Coefficients:
```

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	-17.5791	6.7584	-2.601	0.0123 *
speed	3.9324	0.4155	9.464	1.49e-12 ***

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 15.38 on 48 degrees of freedom
```

```
Multiple R-squared:  0.6511,    Adjusted R-squared:  0.6438
```

```
F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```

```
# R 내장 cars 데이터
```

```
cars
```

```
cars[1:4,]
```

```
#회귀 모형 적합
```

```
z <- lm(dist ~ speed, data = cars)
```

```
summary(z)
```

```
# cars 데이터를 산점도로 나타내기
```

```
x11()
```

```
par(mfrow=c(1,1))
```

```
plot(cars, main = "abline")
```

```
# horizontal
```

```
abline(h = 20)
```

```
abline(h = 30)
```

```
# vertical
```

```
abline(v = 20, col = 'blue')
```

```
#  $y = a + bx$ 
```

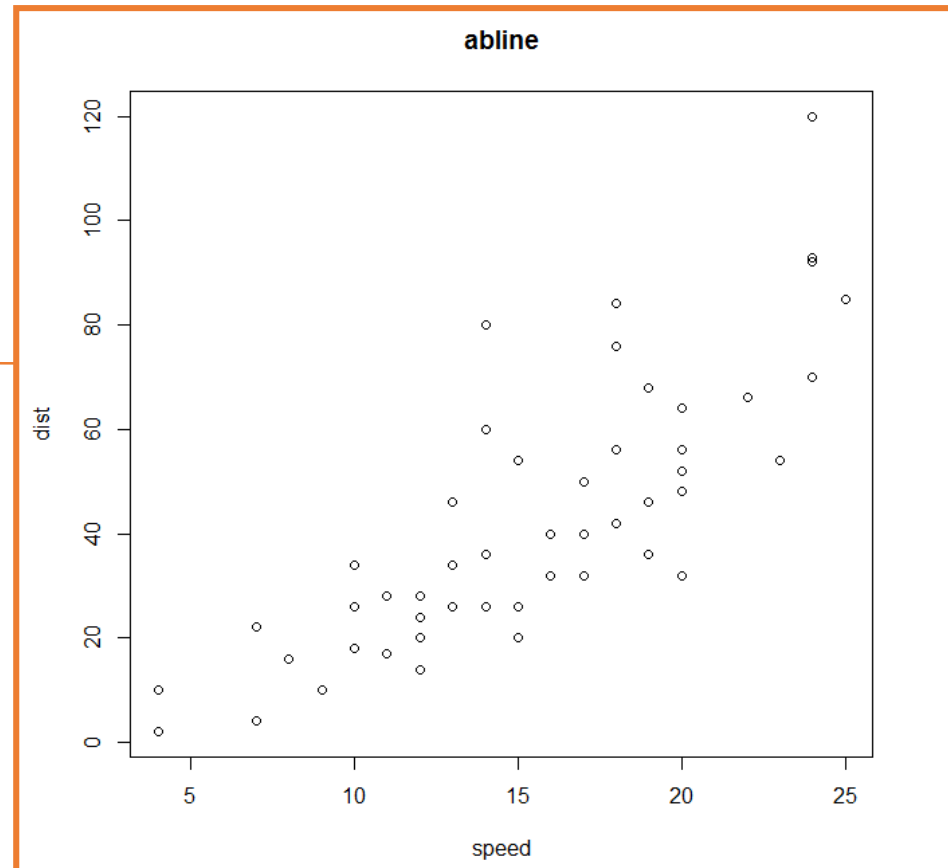
```
abline(a = 40, b = 4, col = 'red')
```

```
# reg 인수
```

```
abline(z, lty = 2, lwd = 2, col = 'green')
```

```
# coef 인수
```

```
abline(z$coef, lty = 3, lwd = 2, col = 'red')
```



```
# R 내장 cars 데이터
```

```
cars
cars[1:4,]
```

```
#회귀 모형 적합
```

```
z <- lm(dist ~ speed, data = cars)
summary(z)
```

```
# cars 데이터를 산점도로 나타내기
```

```
x11()
par(mfrow=c(1,1))
plot(cars, main = "abline")
```

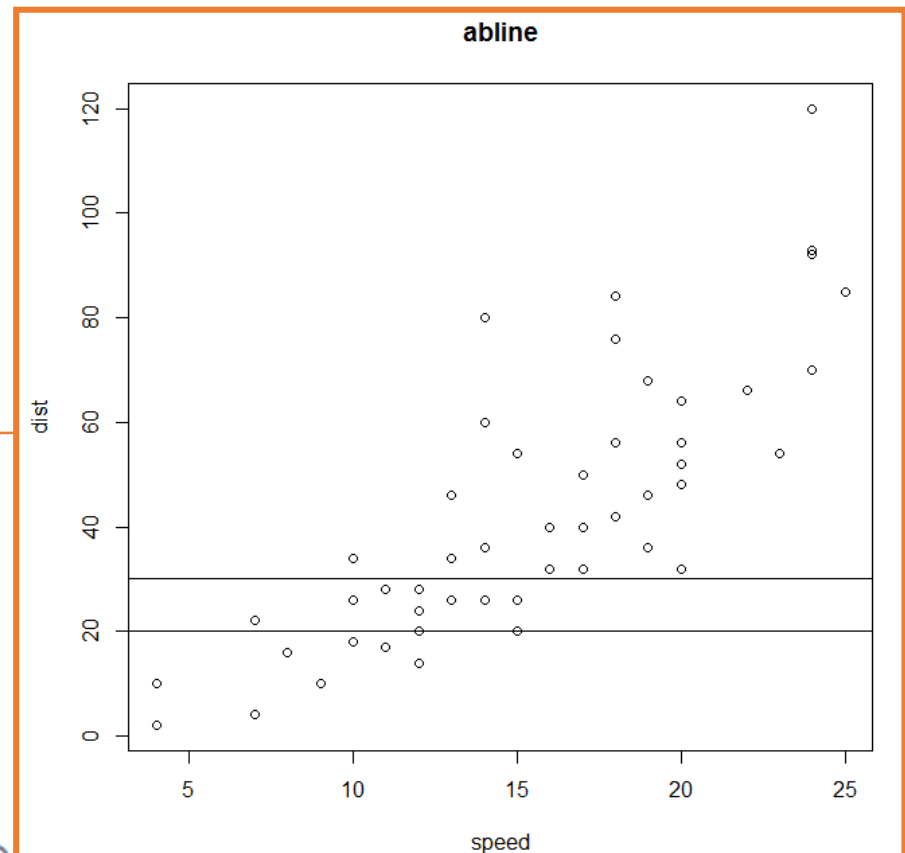
```
# horizontal
abline(h = 20)
abline(h = 30)
```

```
# vertical
abline(v = 20, col = 'blue')
```

```
# y = a + bx
abline(a = 40, b = 4, col = 'red')
```

```
# reg 인수
abline(z, lty = 2, lwd = 2, col = 'green')
```

```
# coef 인수
abline(z$coef, lty = 3, lwd = 2, col = 'red')
```



```
# R 내장 cars 데이터
```

```
cars
```

```
cars[1:4,]
```

```
#회귀 모형 적합
```

```
z <- lm(dist ~ speed, data = cars)
```

```
summary(z)
```

```
# cars 데이터를 산점도로 나타내기
```

```
x11()
```

```
par(mfrow=c(1,1))
```

```
plot(cars, main = "abline")
```

```
# horizontal
```

```
abline(h = 20)
```

```
abline(h = 30)
```

```
# vertical
```

```
abline(v = 20, col = 'blue')
```

```
# y = a + bx
```

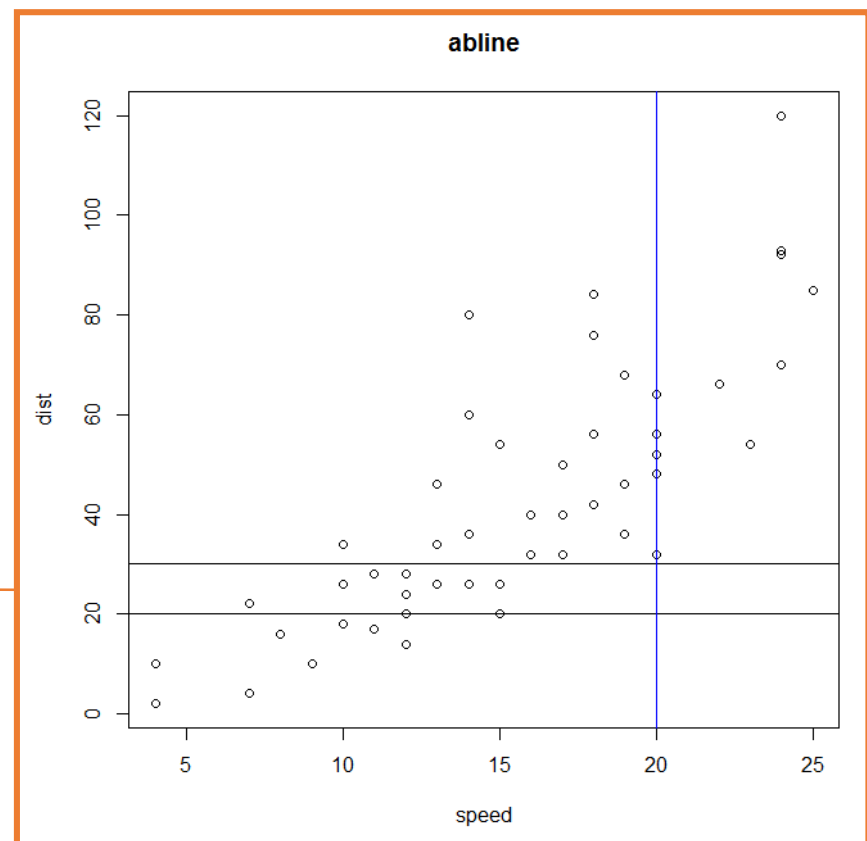
```
abline(a = 40, b = 4, col = 'red')
```

```
# reg 인수
```

```
abline(z, lty = 2, lwd = 2, col = 'green')
```

```
# coef 인수
```

```
abline(z$coef, lty = 3, lwd = 2, col = 'red')
```



```
# R 내장 cars 데이터
```

```
cars
```

```
cars[1:4,]
```

```
#회귀 모형 적합
```

```
z <- lm(dist ~ speed, data = cars)
```

```
summary(z)
```

```
# cars 데이터를 산점도로 나타내기
```

```
x11()
```

```
par(mfrow=c(1,1))
```

```
plot(cars, main = "abline")
```

```
# horizontal
```

```
abline(h = 20)
```

```
abline(h = 30)
```

```
# vertical
```

```
abline(v = 20, col = 'blue')
```

```
#  $y = a + bx$ 
```

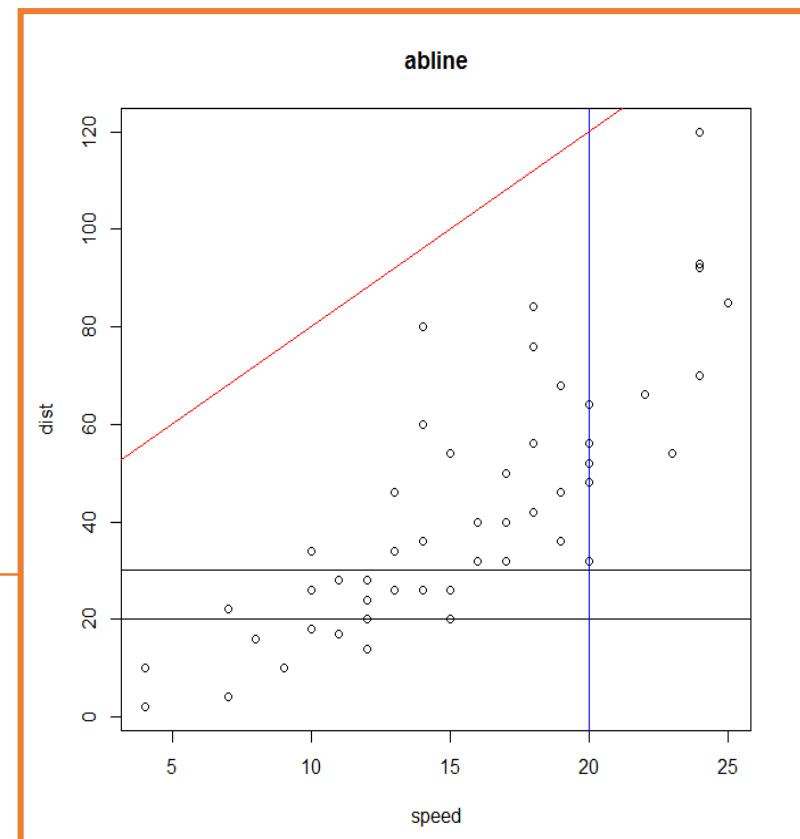
```
abline(a = 40, b = 4, col = 'red')
```

```
# reg 인수
```

```
abline(z, lty = 2, lwd = 2, col = 'green')
```

```
# coef 인수
```

```
abline(z$coef, lty = 3, lwd = 2, col = 'red')
```



```
# R 내장 cars 데이터
```

```
cars
```

```
cars[1:4,]
```

```
#회귀 모형 적합
```

```
z <- lm(dist ~ speed, data = cars)
```

```
summary(z)
```

```
# cars 데이터를 산점도로 나타내기
```

```
x11()
```

```
par(mfrow=c(1,1))
```

```
plot(cars, main = "abline")
```

```
# horizontal
```

```
abline(h = 20)
```

```
abline(h = 30)
```

```
# vertical
```

```
abline(v = 20, col = 'blue')
```

```
#  $y = a + bx$ 
```

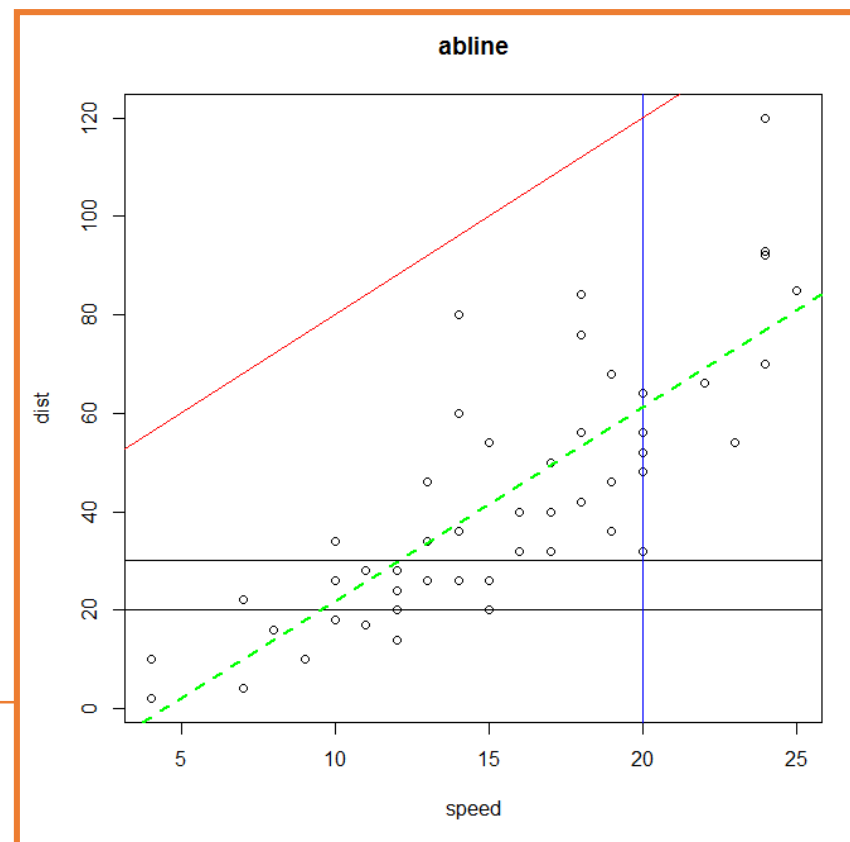
```
abline(a = 40, b = 4, col = 'red')
```

```
# reg 인수
```

```
abline(z, lty = 2, lwd = 2, col = 'green')
```

```
# coef 인수
```

```
abline(z$coef, lty = 3, lwd = 2, col = 'red')
```



```
# R 내장 cars 데이터
```

```
cars
```

```
cars[1:4,]
```

```
#회귀 모형 적합
```

```
z <- lm(dist ~ speed, data = cars)
```

```
summary(z)
```

```
# cars 데이터를 산점도로 나타내기
```

```
x11()
```

```
par(mfrow=c(1,1))
```

```
plot(cars, main = "abline")
```

```
# horizontal
```

```
abline(h = 20)
```

```
abline(h = 30)
```

```
# vertical
```

```
abline(v = 20, col = 'blue')
```

```
# y = a + bx
```

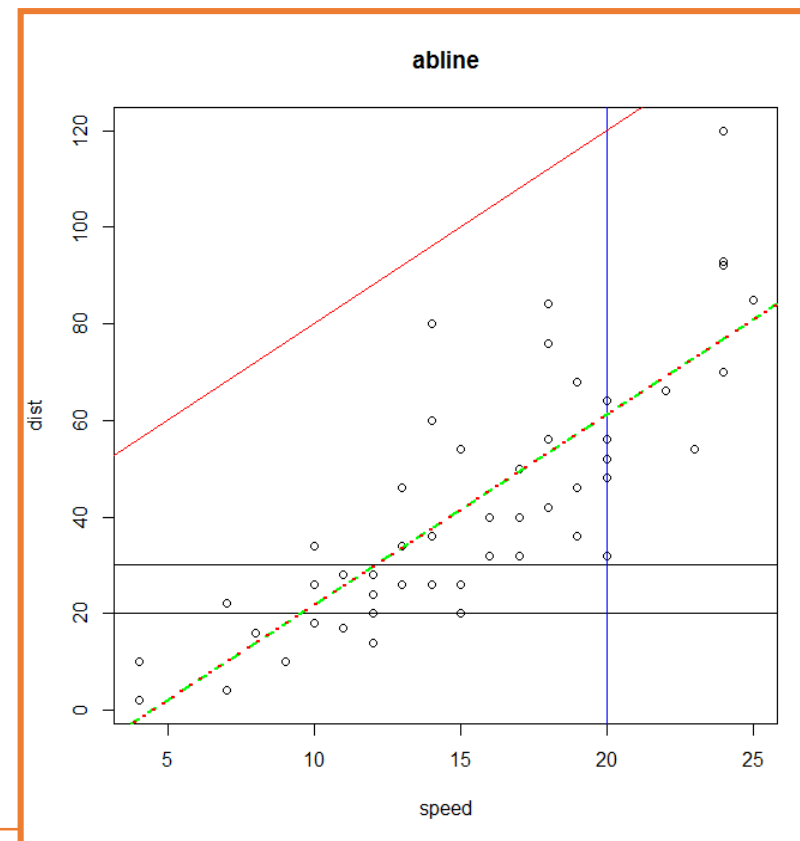
```
abline(a = 40, b = 4, col = 'red')
```

```
# reg 인수
```

```
abline(z, lty = 2, lwd = 2, col = 'green')
```

```
# coef 인수
```

```
abline(z$coef, lty = 3, lwd = 2, col = 'red')
```





## legend()함수

범례를 출력하는 함수로 두 개 이상의 그래프를 출력한 plot에서 각 그래프의 구분을 나타내기 위한 정보를 나타내기 위해 사용

`legend(x, y, legend, pch, lty, fill, col, ...)`

`x, y` : legend를 출력할 위치 지정

ex) `x=a, y=b` : 좌표 (a,b) 에 범례를 출력

위치를 나타내는 문자사용

ex) "topright", "bottomleft", "center" 등

`pch` : 점에 대한 범례일 경우, 점을 구분하기 위해 사용

`lty` : 선에 대한 범례일 경우, 선의 type을 구분하기 위해 사용

`fill` : 면에 대한 범례일 경우, 면의 색상을 구분하기 위해 사용

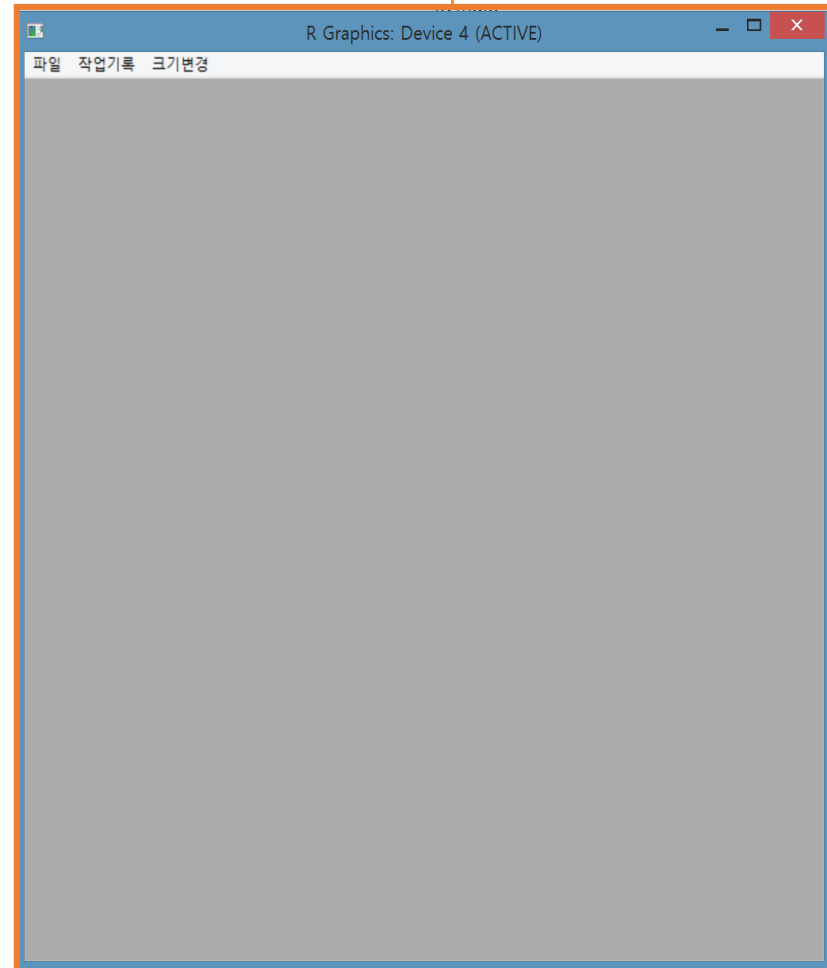
`pch`와 `lty` 동시 사용 : 점과 선을 동시에 사용한 그래프의 범례

```
# R graphic Device 실행하기  
x11()
```

```
# type 을 no plotting으로 설정하여 plot 표현하기  
plot(1:10, type= "n", xlab="", ylab="", main="legend")
```

```
# 위치를 나타내는 용어로 범례 나타내기  
legend("bottomright", "(x,y)", pch=1, title="bottomright")  
legend("bottom", "(x,y)", pch=1, title="bottom")  
legend("bottomleft", "(x,y)", pch=1, title="bottomleft")  
legend("left", "(x,y)", pch=1, title="left")  
legend("topleft", "(x,y)", pch=1, title="topleft")  
legend("top", "(x,y)", pch=1, title="top")  
legend("topright", "(x,y)", pch=1, title="topright")  
legend("right", "(x,y)", pch=1, title="right")  
legend("center", "(x,y)", pch=1, title="center")  
legends <- c("Legend1", "Legend2")
```

```
# x,y 좌표를 통해서 범례 나타내기  
legend(3,8, legend = legends, pch=1:2, col=1:2)  
legend(7,8, legend = legends, pch=1:2, col=1:2, lty=1:2)  
legend(3,4, legend = legends, fill=1:2)  
legend(7,4, legend = legends, fill=1:2, density=30)
```

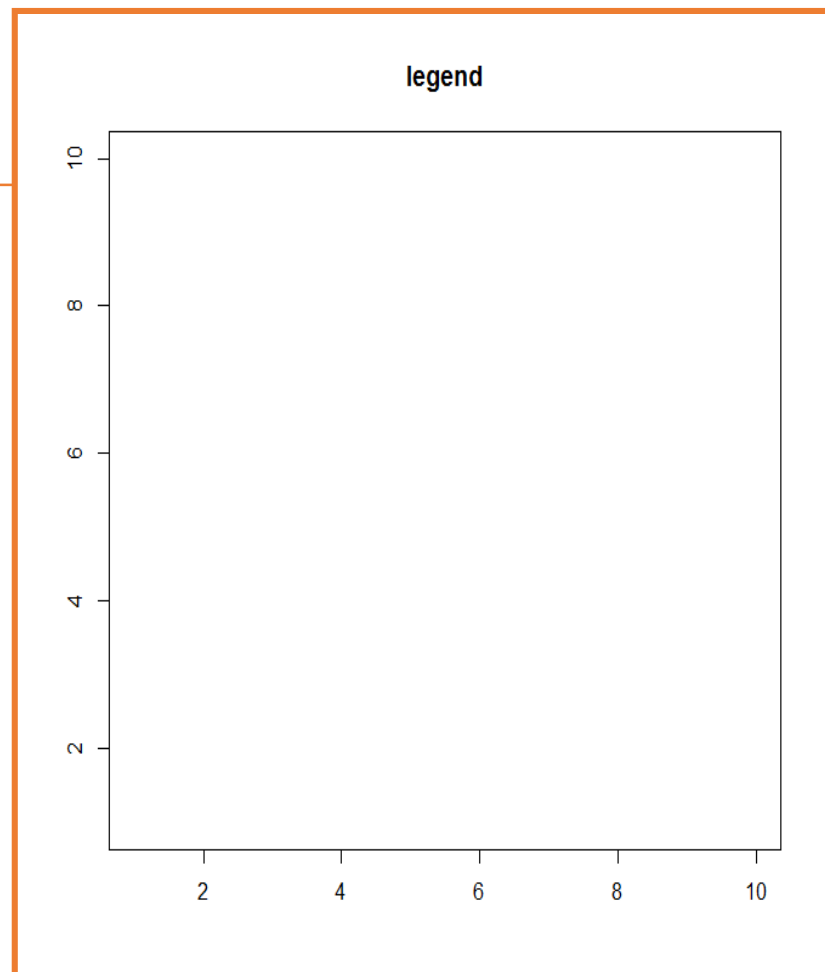


```
# R graphic Device 실행하기  
x11()
```

```
# type 을 no plotting으로 설정하여 plot 표현하기  
plot(1:10, type= "n", xlab="", ylab="", main="legend")
```

```
# 위치를 나타내는 용어로 범례 나타내기  
legend("bottomright", "(x,y)", pch=1, title="bottomright")  
legend("bottom", "(x,y)", pch=1, title="bottom")  
legend("bottomleft", "(x,y)", pch=1, title="bottomleft")  
legend("left", "(x,y)", pch=1, title="left")  
legend("topleft", "(x,y)", pch=1, title="topleft")  
legend("top", "(x,y)", pch=1, title="top")  
legend("topright", "(x,y)", pch=1, title="topright")  
legend("right", "(x,y)", pch=1, title="right")  
legend("center", "(x,y)", pch=1, title="center")  
legends <- c("Legend1", "Legend2")
```

```
# x,y 좌표를 통해서 범례 나타내기  
legend(3,8, legend = legends, pch=1:2, col=1:2)  
legend(7,8, legend = legends, pch=1:2, col=1:2, lty=1:2)  
legend(3,4, legend = legends, fill=1:2)  
legend(7,4, legend = legends, fill=1:2, density=30)
```



```
# R graphic Device 실행하기
```

```
x11()
```

```
# type 을 no plotting으로 설정하여 plot 표현하기
```

```
plot(1:10, type= "n", xlab="", ylab="", main="legend")
```

```
# 위치를 나타내는 용어로 범례 나타내기
```

```
legend("bottomright", "(x,y)", pch=1, title="bottomright")
```

```
legend("bottom", "(x,y)", pch=1, title="bottom")
```

```
legend("bottomleft", "(x,y)", pch=1, title="bottomleft")
```

```
legend("left", "(x,y)", pch=1, title="left")
```

```
legend("topleft", "(x,y)", pch=1, title="topleft")
```

```
legend("top", "(x,y)", pch=1, title="top")
```

```
legend("topright", "(x,y)", pch=1, title="topright")
```

```
legend("right", "(x,y)", pch=1, title="right")
```

```
legend("center", "(x,y)", pch=1, title="center")
```

```
legends <- c("Legend1", "Legend2")
```

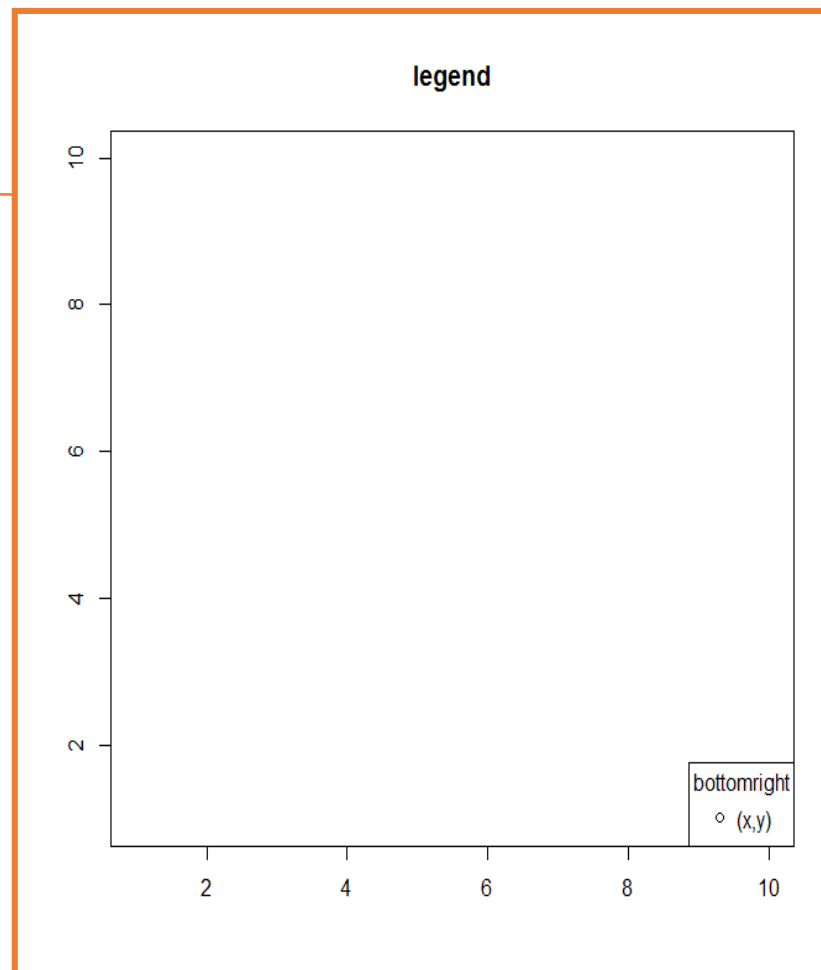
```
# x,y 좌표를 통해서 범례 나타내기
```

```
legend(3,8, legend = legends, pch=1:2, col=1:2)
```

```
legend(7,8, legend = legends, pch=1:2, col=1:2, lty=1:2)
```

```
legend(3,4, legend = legends, fill=1:2)
```

```
legend(7,4, legend = legends, fill=1:2, density=30)
```

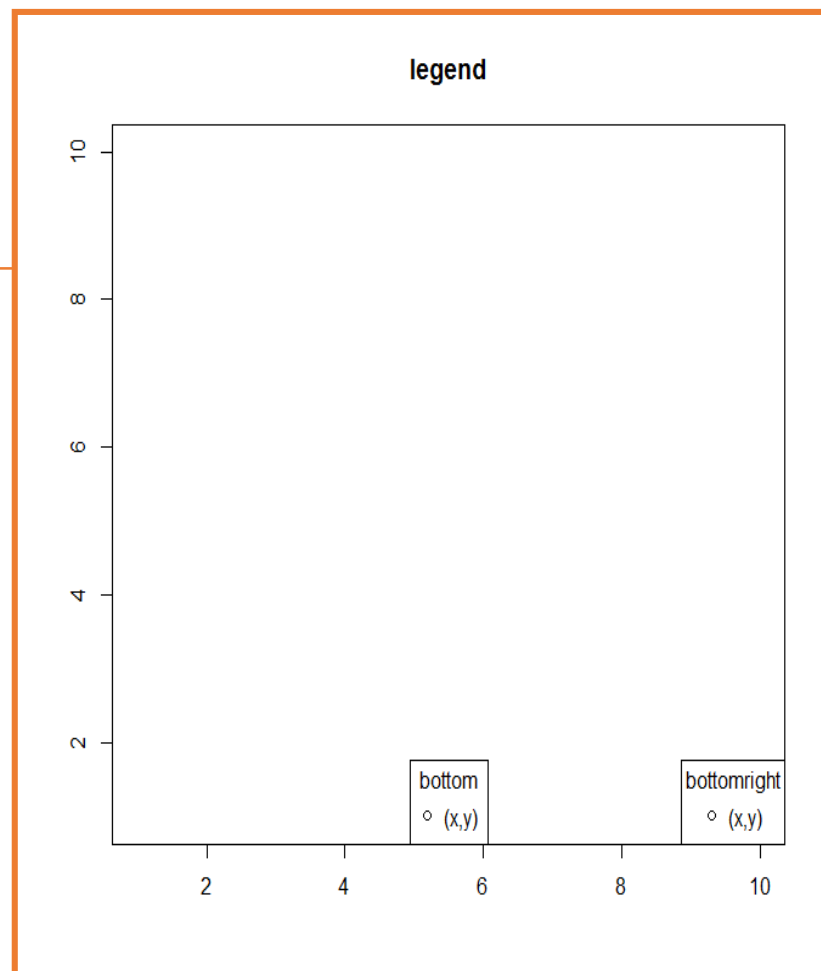


```
# R graphic Device 실행하기
x11()

# type 을 no plotting으로 설정하여 plot 표현하기
plot(1:10, type= "n", xlab="", ylab="", main="legend")

# 위치를 나타내는 용어로 범례 나타내기
legend("bottomright", "(x,y)", pch=1, title="bottomright")
legend("bottom", "(x,y)", pch=1, title="bottom")
legend("bottomleft", "(x,y)", pch=1, title="bottomleft")
legend("left", "(x,y)", pch=1, title="left")
legend("topleft", "(x,y)", pch=1, title="topleft")
legend("top", "(x,y)", pch=1, title="top")
legend("topright", "(x,y)", pch=1, title="topright")
legend("right", "(x,y)", pch=1, title="right")
legend("center", "(x,y)", pch=1, title="center")
legends <- c("Legend1", "Legend2")

# x,y 좌표를 통해서 범례 나타내기
legend(3,8, legend = legends, pch=1:2, col=1:2)
legend(7,8, legend = legends, pch=1:2, col=1:2, lty=1:2)
legend(3,4, legend = legends, fill=1:2)
legend(7,4, legend = legends, fill=1:2, density=30)
```

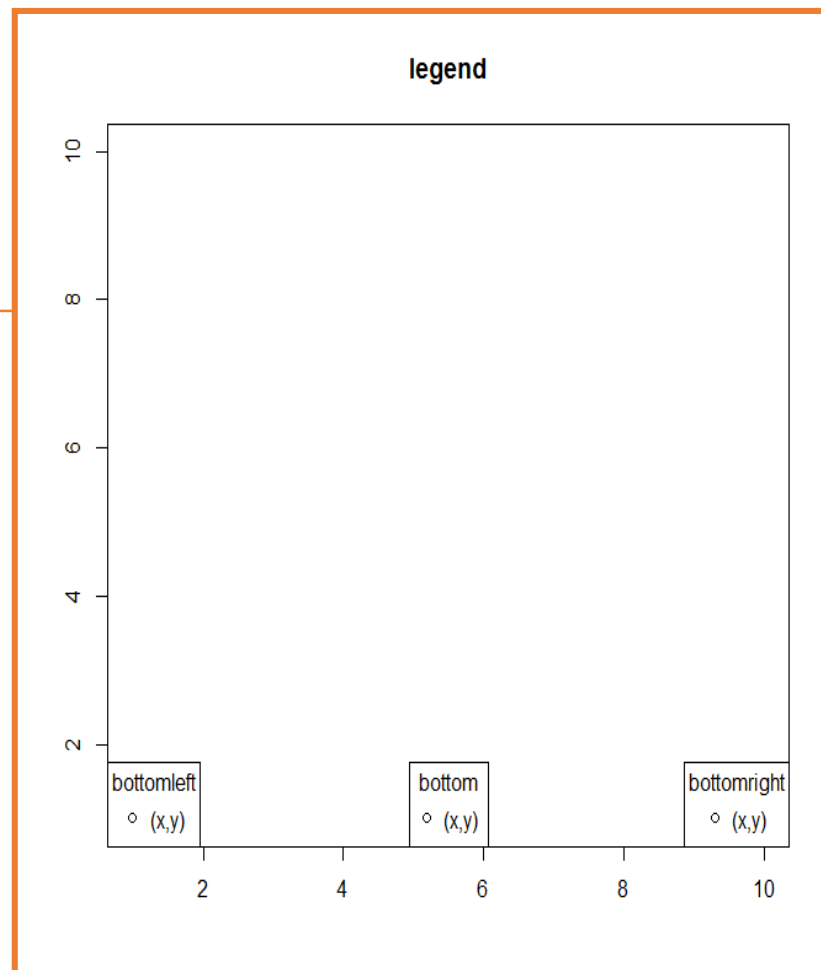


```
# R graphic Device 실행하기
x11()

# type 을 no plotting으로 설정하여 plot 표현하기
plot(1:10, type= "n", xlab="", ylab="", main="legend")

# 위치를 나타내는 용어로 범례 나타내기
legend("bottomright", "(x,y)", pch=1, title="bottomright")
legend("bottom", "(x,y)", pch=1, title="bottom")
legend("bottomleft", "(x,y)", pch=1, title="bottomleft")
legend("left", "(x,y)", pch=1, title="left")
legend("topleft", "(x,y)", pch=1, title="topleft")
legend("top", "(x,y)", pch=1, title="top")
legend("topright", "(x,y)", pch=1, title="topright")
legend("right", "(x,y)", pch=1, title="right")
legend("center", "(x,y)", pch=1, title="center")
legends <- c("Legend1", "Legend2")

# x,y 좌표를 통해서 범례 나타내기
legend(3,8, legend = legends, pch=1:2, col=1:2)
legend(7,8, legend = legends, pch=1:2, col=1:2, lty=1:2)
legend(3,4, legend = legends, fill=1:2)
legend(7,4, legend = legends, fill=1:2, density=30)
```

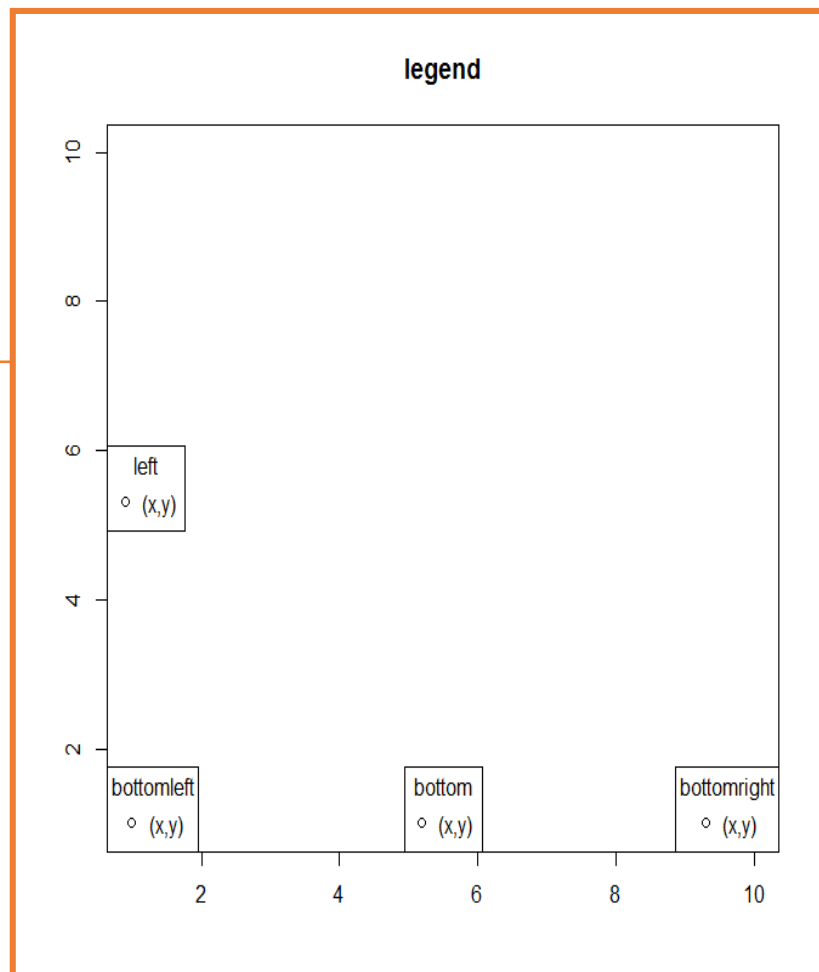


```
# R graphic Device 실행하기
x11()

# type 을 no plotting으로 설정하여 plot 표현하기
plot(1:10, type= "n", xlab="", ylab="", main="legend")

# 위치를 나타내는 용어로 범례 나타내기
legend("bottomright", "(x,y)", pch=1, title="bottomright")
legend("bottom", "(x,y)", pch=1, title="bottom")
legend("bottomleft", "(x,y)", pch=1, title="bottomleft")
legend("left", "(x,y)", pch=1, title="left")
legend("topleft", "(x,y)", pch=1, title="topleft")
legend("top", "(x,y)", pch=1, title="top")
legend("topright", "(x,y)", pch=1, title="topright")
legend("right", "(x,y)", pch=1, title="right")
legend("center", "(x,y)", pch=1, title="center")
legends <- c("Legend1", "Legend2")

# x,y 좌표를 통해서 범례 나타내기
legend(3,8, legend = legends, pch=1:2, col=1:2)
legend(7,8, legend = legends, pch=1:2, col=1:2, lty=1:2)
legend(3,4, legend = legends, fill=1:2)
legend(7,4, legend = legends, fill=1:2, density=30)
```



```
# R graphic Device 실행하기
x11()
```

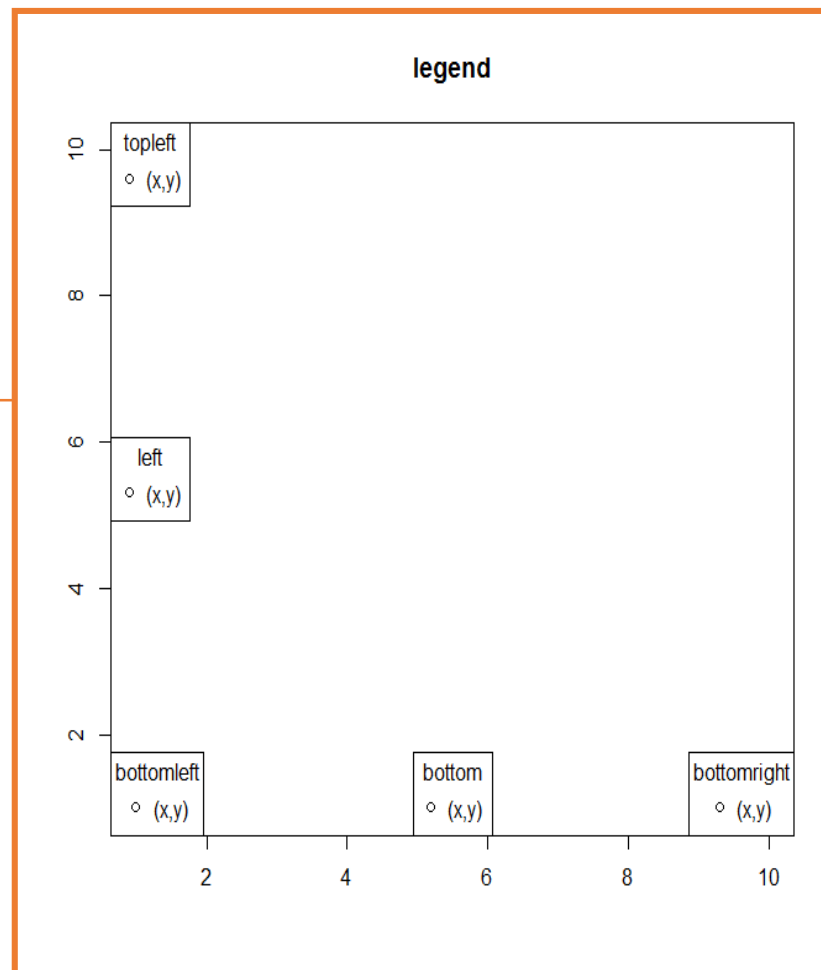
```
# type 을 no plotting으로 설정하여 plot 표현하기
plot(1:10, type= "n", xlab="", ylab="", main="legend")
```

```
# 위치를 나타내는 용어로 범례 나타내기
```

```
legend("bottomright", "(x,y)", pch=1, title="bottomright")
legend("bottom", "(x,y)", pch=1, title="bottom")
legend("bottomleft", "(x,y)", pch=1, title="bottomleft")
legend("left", "(x,y)", pch=1, title="left")
legend("topleft", "(x,y)", pch=1, title="topleft")
legend("top", "(x,y)", pch=1, title="top")
legend("topright", "(x,y)", pch=1, title="topright")
legend("right", "(x,y)", pch=1, title="right")
legend("center", "(x,y)", pch=1, title="center")
legends <- c("Legend1", "Legend2")
```

```
# x,y 좌표를 통해서 범례 나타내기
```

```
legend(3,8, legend = legends, pch=1:2, col=1:2)
legend(7,8, legend = legends, pch=1:2, col=1:2, lty=1:2)
legend(3,4, legend = legends, fill=1:2)
legend(7,4, legend = legends, fill=1:2, density=30)
```



```
# R graphic Device 실행하기
```

```
x11()
```

```
# type 을 no plotting으로 설정하여 plot 표현하기
```

```
plot(1:10, type="n", xlab="", ylab="", main="legend")
```

```
# 위치를 나타내는 용어로 범례 나타내기
```

```
legend("bottomright", "(x,y)", pch=1, title="bottomright")
```

```
legend("bottom", "(x,y)", pch=1, title="bottom")
```

```
legend("bottomleft", "(x,y)", pch=1, title="bottomleft")
```

```
legend("left", "(x,y)", pch=1, title="left")
```

```
legend("topleft", "(x,y)", pch=1, title="topleft")
```

```
legend("top", "(x,y)", pch=1, title="top")
```

```
legend("topright", "(x,y)", pch=1, title="topright")
```

```
legend("right", "(x,y)", pch=1, title="right")
```

```
legend("center", "(x,y)", pch=1, title="center")
```

```
legends <- c("Legend1", "Legend2")
```

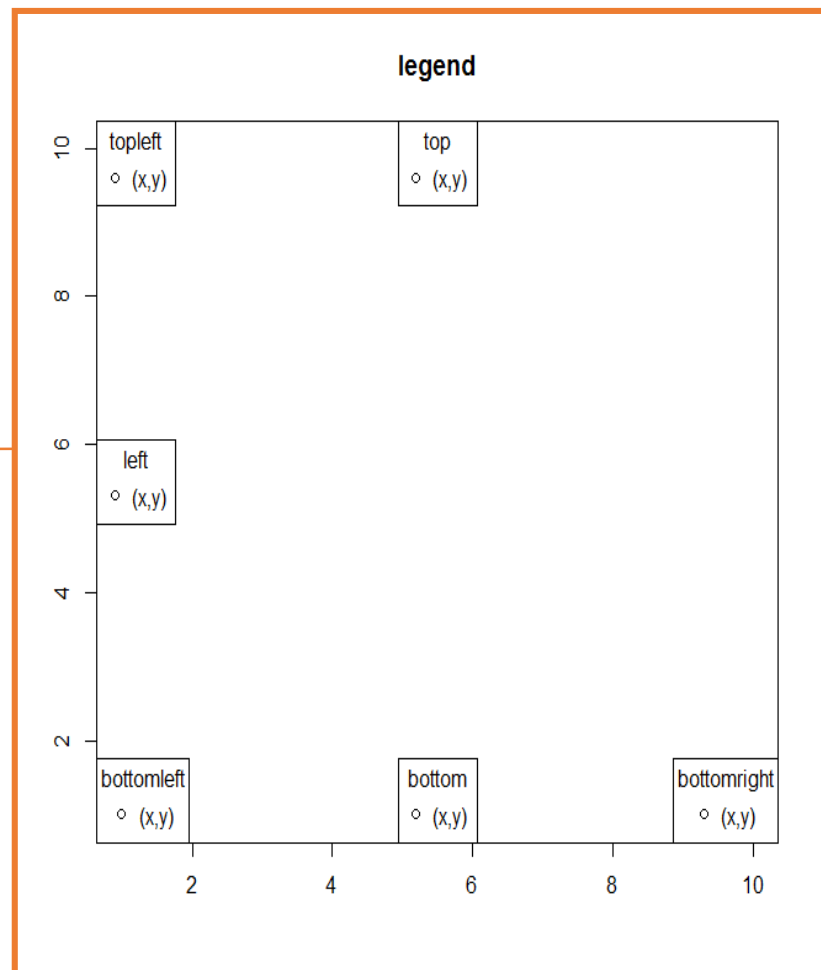
```
# x,y 좌표를 통해서 범례 나타내기
```

```
legend(3,8, legend = legends, pch=1:2, col=1:2)
```

```
legend(7,8, legend = legends, pch=1:2, col=1:2, lty=1:2)
```

```
legend(3,4, legend = legends, fill=1:2)
```

```
legend(7,4, legend = legends, fill=1:2, density=30)
```

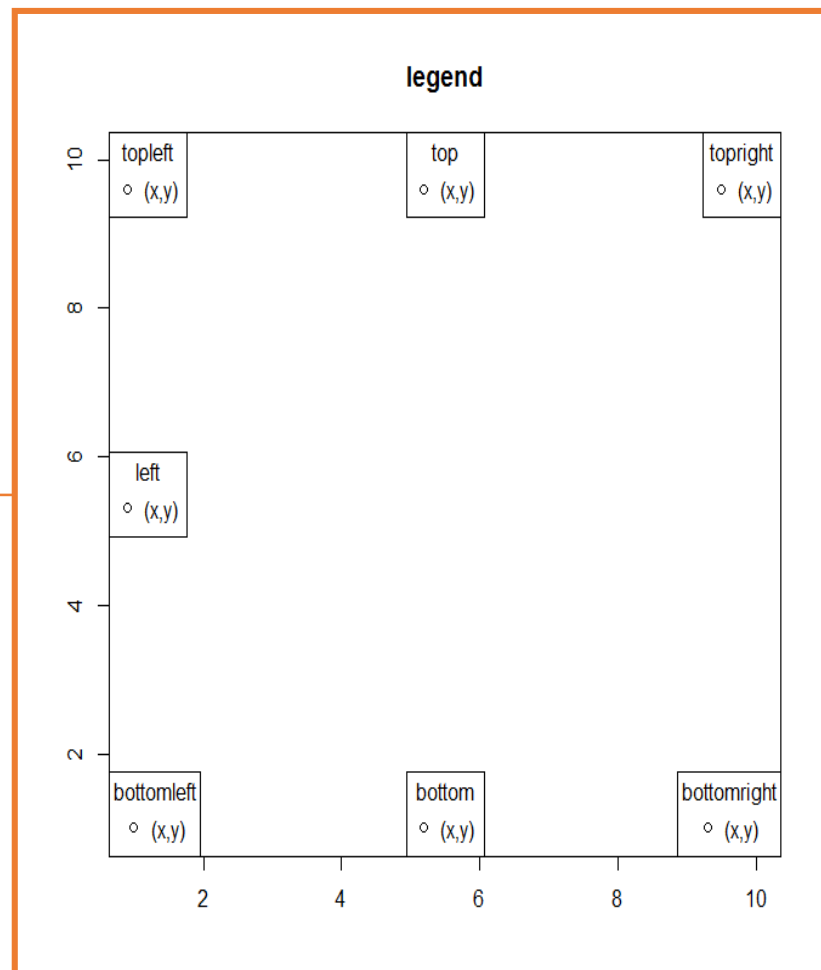


```
# R graphic Device 실행하기
x11()

# type 을 no plotting으로 설정하여 plot 표현하기
plot(1:10, type= "n", xlab="", ylab="", main="legend")

# 위치를 나타내는 용어로 범례 나타내기
legend("bottomright", "(x,y)", pch=1, title="bottomright")
legend("bottom", "(x,y)", pch=1, title="bottom")
legend("bottomleft", "(x,y)", pch=1, title="bottomleft")
legend("left", "(x,y)", pch=1, title="left")
legend("topleft", "(x,y)", pch=1, title="topleft")
legend("top", "(x,y)", pch=1, title="top")
legend("topright", "(x,y)", pch=1, title="topright")
legend("right", "(x,y)", pch=1, title="right")
legend("center", "(x,y)", pch=1, title="center")
legends <- c("Legend1", "Legend2")

# x,y 좌표를 통해서 범례 나타내기
legend(3,8, legend = legends, pch=1:2, col=1:2)
legend(7,8, legend = legends, pch=1:2, col=1:2, lty=1:2)
legend(3,4, legend = legends, fill=1:2)
legend(7,4, legend = legends, fill=1:2, density=30)
```



```
# R graphic Device 실행하기
```

```
x11()
```

```
# type 을 no plotting으로 설정하여 plot 표현하기
```

```
plot(1:10, type="n", xlab="", ylab="", main="legend")
```

```
# 위치를 나타내는 용어로 범례 나타내기
```

```
legend("bottomright", "(x,y)", pch=1, title="bottomright")
```

```
legend("bottom", "(x,y)", pch=1, title="bottom")
```

```
legend("bottomleft", "(x,y)", pch=1, title="bottomleft")
```

```
legend("left", "(x,y)", pch=1, title="left")
```

```
legend("topleft", "(x,y)", pch=1, title="topleft")
```

```
legend("top", "(x,y)", pch=1, title="top")
```

```
legend("topright", "(x,y)", pch=1, title="topright")
```

```
legend("right", "(x,y)", pch=1, title="right")
```

```
legend("center", "(x,y)", pch=1, title="center")
```

```
legends <- c("Legend1", "Legend2")
```

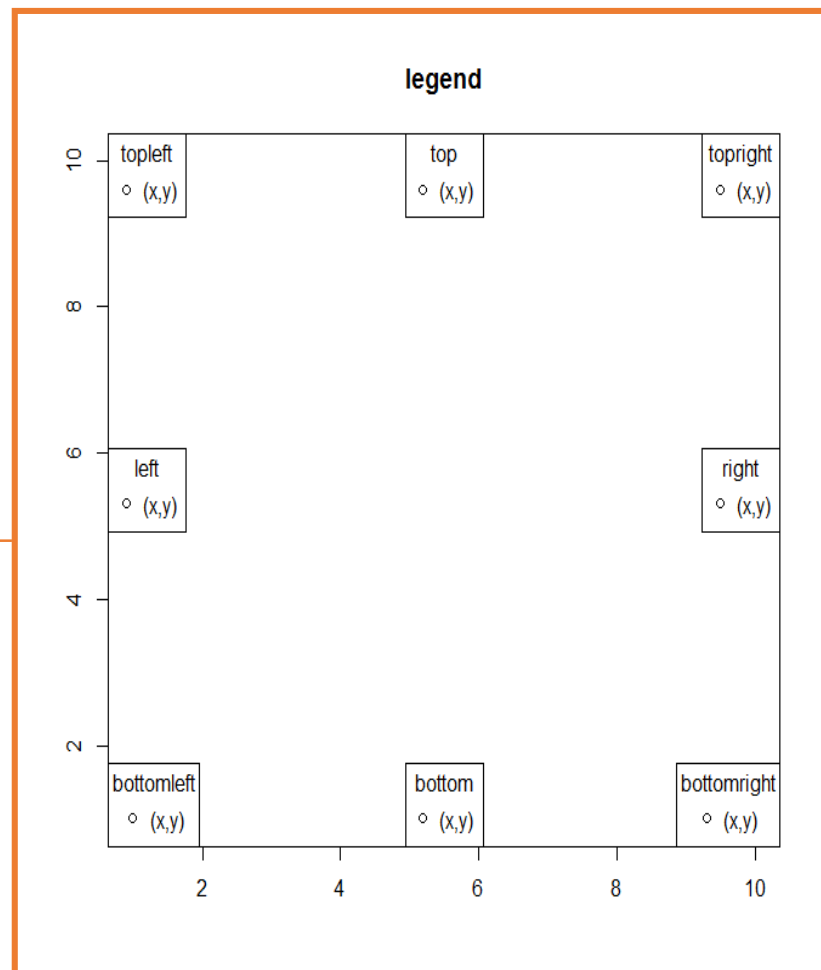
```
# x,y 좌표를 통해서 범례 나타내기
```

```
legend(3,8, legend = legends, pch=1:2, col=1:2)
```

```
legend(7,8, legend = legends, pch=1:2, col=1:2, lty=1:2)
```

```
legend(3,4, legend = legends, fill=1:2)
```

```
legend(7,4, legend = legends, fill=1:2, density=30)
```

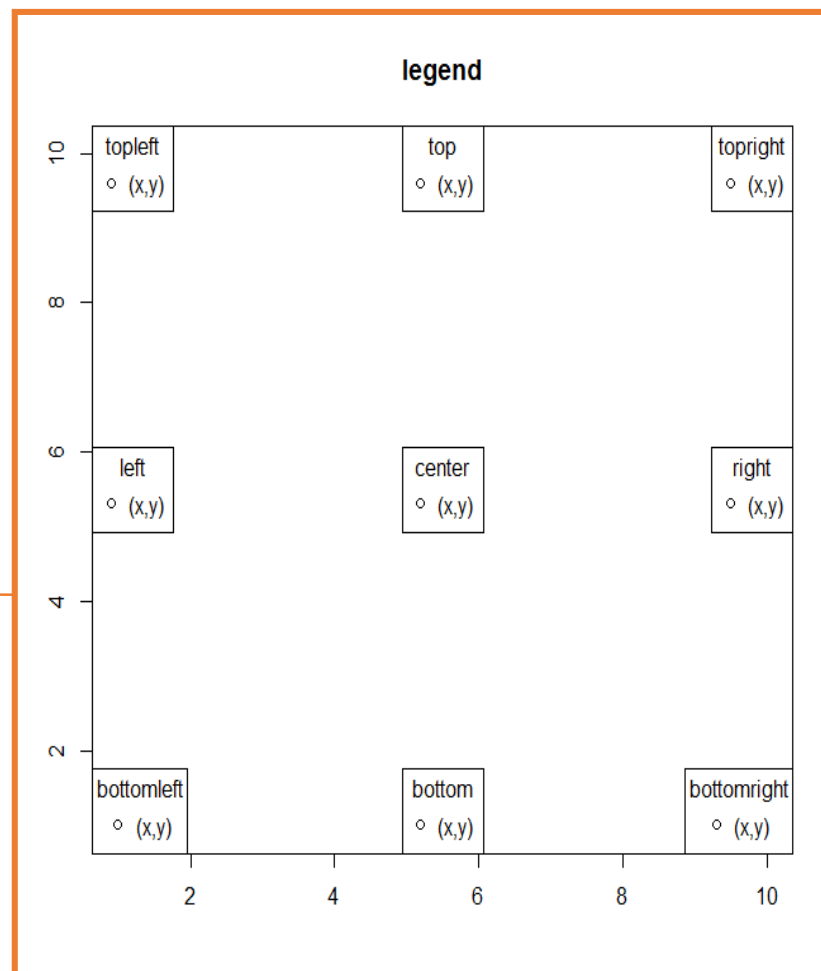


```
# R graphic Device 실행하기
x11()

# type 을 no plotting으로 설정하여 plot 표현하기
plot(1:10, type="n", xlab="", ylab="", main="legend")

# 위치를 나타내는 용어로 범례 나타내기
legend("bottomright", "(x,y)", pch=1, title="bottomright")
legend("bottom", "(x,y)", pch=1, title="bottom")
legend("bottomleft", "(x,y)", pch=1, title="bottomleft")
legend("left", "(x,y)", pch=1, title="left")
legend("topleft", "(x,y)", pch=1, title="topleft")
legend("top", "(x,y)", pch=1, title="top")
legend("topright", "(x,y)", pch=1, title="topright")
legend("right", "(x,y)", pch=1, title="right")
legend("center", "(x,y)", pch=1, title="center")
legends <- c("Legend1", "Legend2")

# x,y 좌표를 통해서 범례 나타내기
legend(3,8, legend = legends, pch=1:2, col=1:2)
legend(7,8, legend = legends, pch=1:2, col=1:2, lty=1:2)
legend(3,4, legend = legends, fill=1:2)
legend(7,4, legend = legends, fill=1:2, density=30)
```

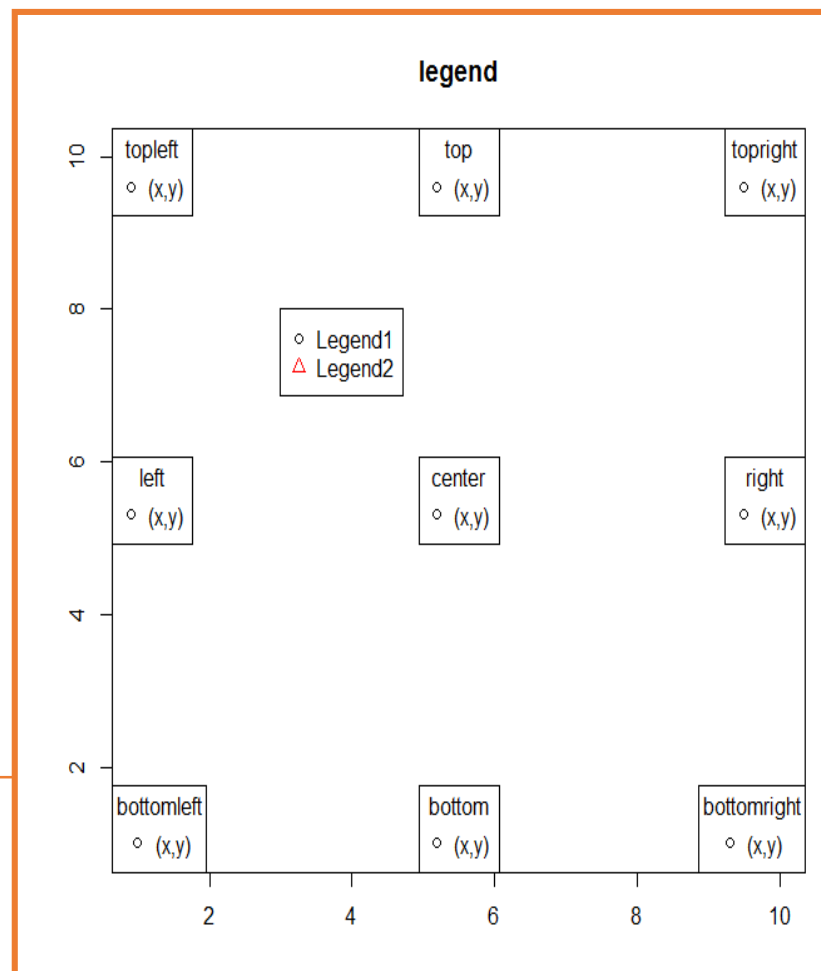


```
# R graphic Device 실행하기
x11()

# type 을 no plotting으로 설정하여 plot 표현하기
plot(1:10, type= "n", xlab="", ylab="", main="legend")

# 위치를 나타내는 용어로 범례 나타내기
legend("bottomright", "(x,y)", pch=1, title="bottomright")
legend("bottom", "(x,y)", pch=1, title="bottom")
legend("bottomleft", "(x,y)", pch=1, title="bottomleft")
legend("left", "(x,y)", pch=1, title="left")
legend("topleft", "(x,y)", pch=1, title="topleft")
legend("top", "(x,y)", pch=1, title="top")
legend("topright", "(x,y)", pch=1, title="topright")
legend("right", "(x,y)", pch=1, title="right")
legend("center", "(x,y)", pch=1, title="center")
legends <- c("Legend1", "Legend2")

# x,y 좌표를 통해서 범례 나타내기
legend(3,8, legend = legends, pch=1:2, col=1:2)
legend(7,8, legend = legends, pch=1:2, col=1:2, lty=1:2)
legend(3,4, legend = legends, fill=1:2)
legend(7,4, legend = legends, fill=1:2, density=30)
```

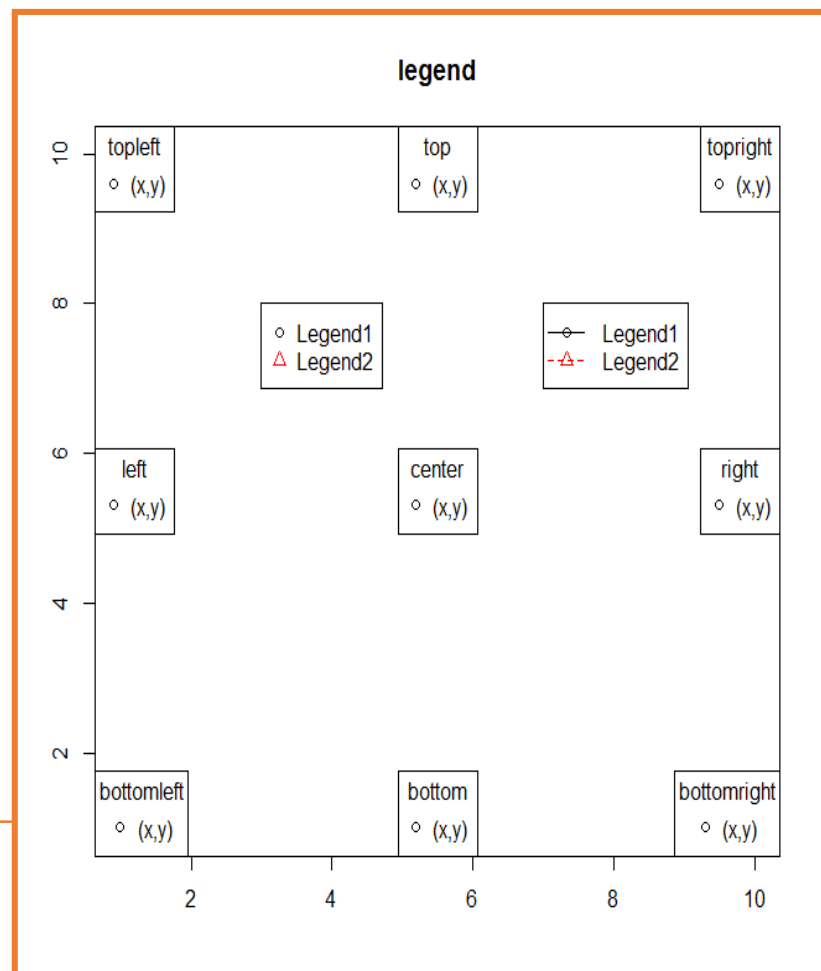


```
# R graphic Device 실행하기
x11()

# type 을 no plotting으로 설정하여 plot 표현하기
plot(1:10, type= "n", xlab="", ylab="", main="legend")

# 위치를 나타내는 용어로 범례 나타내기
legend("bottomright", "(x,y)", pch=1, title="bottomright")
legend("bottom", "(x,y)", pch=1, title="bottom")
legend("bottomleft", "(x,y)", pch=1, title="bottomleft")
legend("left", "(x,y)", pch=1, title="left")
legend("topleft", "(x,y)", pch=1, title="topleft")
legend("top", "(x,y)", pch=1, title="top")
legend("topright", "(x,y)", pch=1, title="topright")
legend("right", "(x,y)", pch=1, title="right")
legend("center", "(x,y)", pch=1, title="center")
legends <- c("Legend1", "Legend2")

# x,y 좌표를 통해서 범례 나타내기
legend(3,8, legend = legends, pch=1:2, col=1:2)
legend(7,8, legend = legends, pch=1:2, col=1:2, lty=1:2)
legend(3,4, legend = legends, fill=1:2)
legend(7,4, legend = legends, fill=1:2, density=30)
```

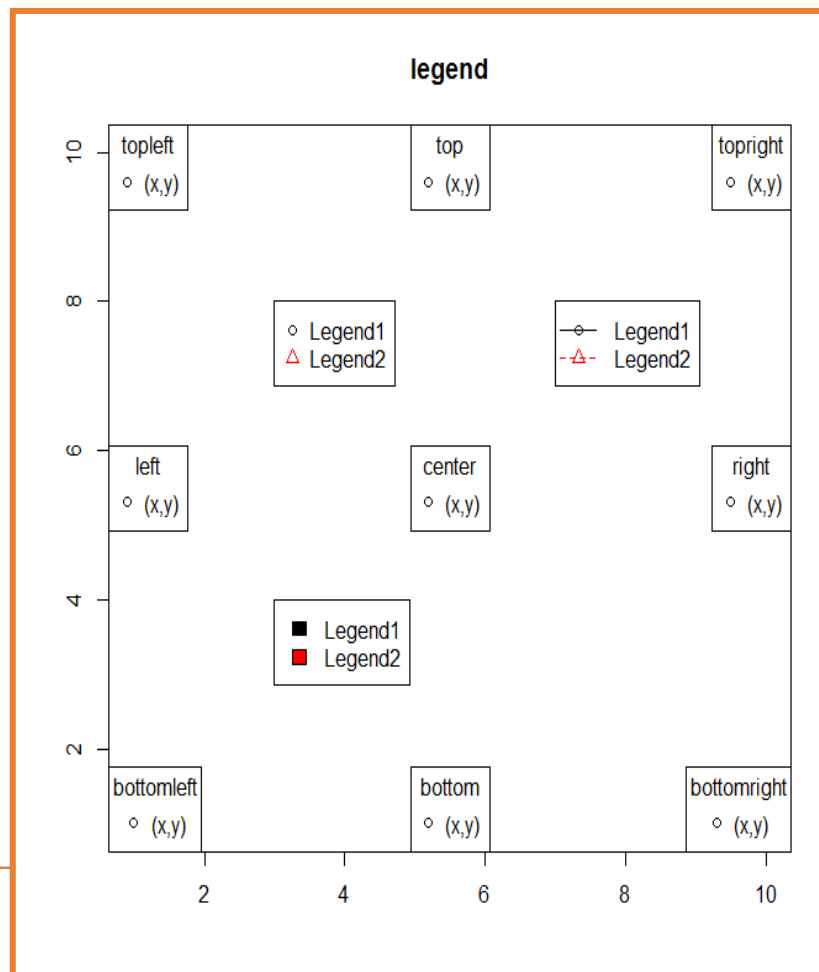


```
# R graphic Device 실행하기
x11()

# type 을 no plotting으로 설정하여 plot 표현하기
plot(1:10, type="n", xlab="", ylab="", main="legend")

# 위치를 나타내는 용어로 범례 나타내기
legend("bottomright", "(x,y)", pch=1, title="bottomright")
legend("bottom", "(x,y)", pch=1, title="bottom")
legend("bottomleft", "(x,y)", pch=1, title="bottomleft")
legend("left", "(x,y)", pch=1, title="left")
legend("topleft", "(x,y)", pch=1, title="topleft")
legend("top", "(x,y)", pch=1, title="top")
legend("topright", "(x,y)", pch=1, title="topright")
legend("right", "(x,y)", pch=1, title="right")
legend("center", "(x,y)", pch=1, title="center")
legends <- c("Legend1", "Legend2")

# x,y 좌표를 통해서 범례 나타내기
legend(3,8, legend = legends, pch=1:2, col=1:2)
legend(7,8, legend = legends, pch=1:2, col=1:2, lty=1:2)
legend(3,4, legend = legends, fill=1:2)
legend(7,4, legend = legends, fill=1:2, density=30)
```

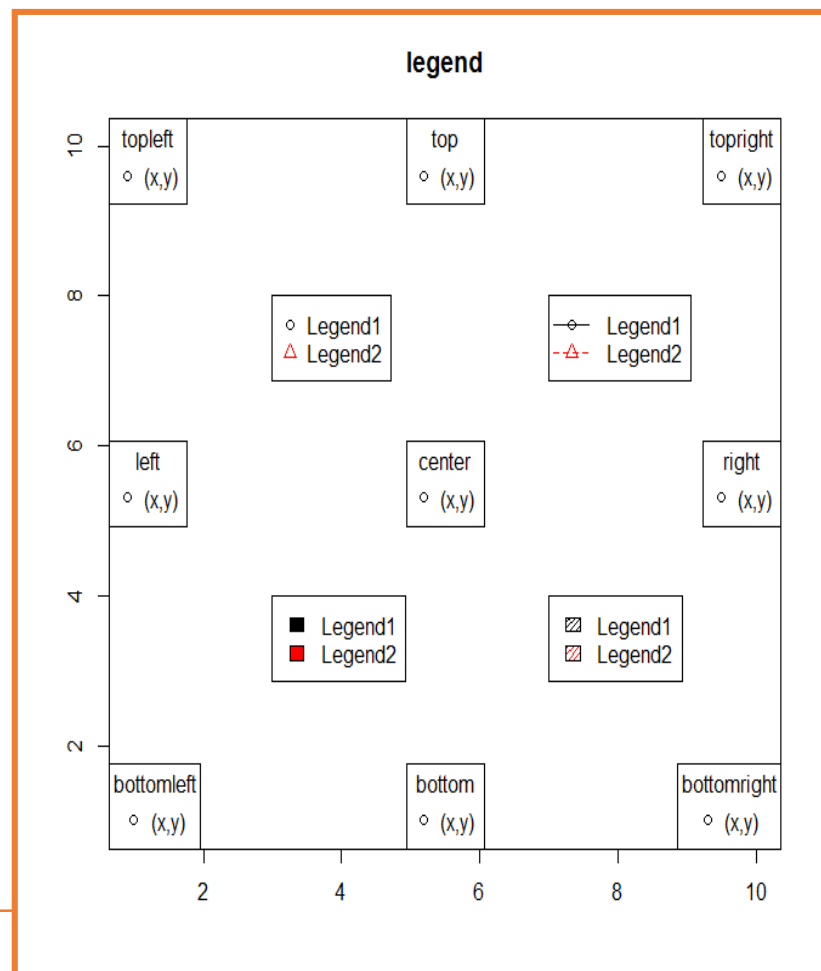


```
# R graphic Device 실행하기
x11()

# type 을 no plotting으로 설정하여 plot 표현하기
plot(1:10, type= "n", xlab="", ylab="", main="legend")

# 위치를 나타내는 용어로 범례 나타내기
legend("bottomright", "(x,y)", pch=1, title="bottomright")
legend("bottom", "(x,y)", pch=1, title="bottom")
legend("bottomleft", "(x,y)", pch=1, title="bottomleft")
legend("left", "(x,y)", pch=1, title="left")
legend("topleft", "(x,y)", pch=1, title="topleft")
legend("top", "(x,y)", pch=1, title="top")
legend("topright", "(x,y)", pch=1, title="topright")
legend("right", "(x,y)", pch=1, title="right")
legend("center", "(x,y)", pch=1, title="center")
legends <- c("Legend1", "Legend2")

# x,y 좌표를 통해서 범례 나타내기
legend(3,8, legend = legends, pch=1:2, col=1:2)
legend(7,8, legend = legends, pch=1:2, col=1:2, lty=1:2)
legend(3,4, legend = legends, fill=1:2)
legend(7,4, legend = legends, fill=1:2, density=30)
```





# 5. ggplot2 패키지



## ggplot2 패키지

- ✓ 뉴질랜드 Rice 대학교의 Hadley Wickham 교수가 만든 그래픽스 패키지로 Grammar of Graphics 개념의 구현체로 현재 R 그래픽스에서 주로 사용되는 R 패키지
- ✓ ggplot2는 문법처럼 체계를 갖추고 있기 때문에 간단한 코드를 추가하거나 삭제하면서 그래프를 쉽고 편하게 그릴 수 있다.

- ✓ 주요 함수

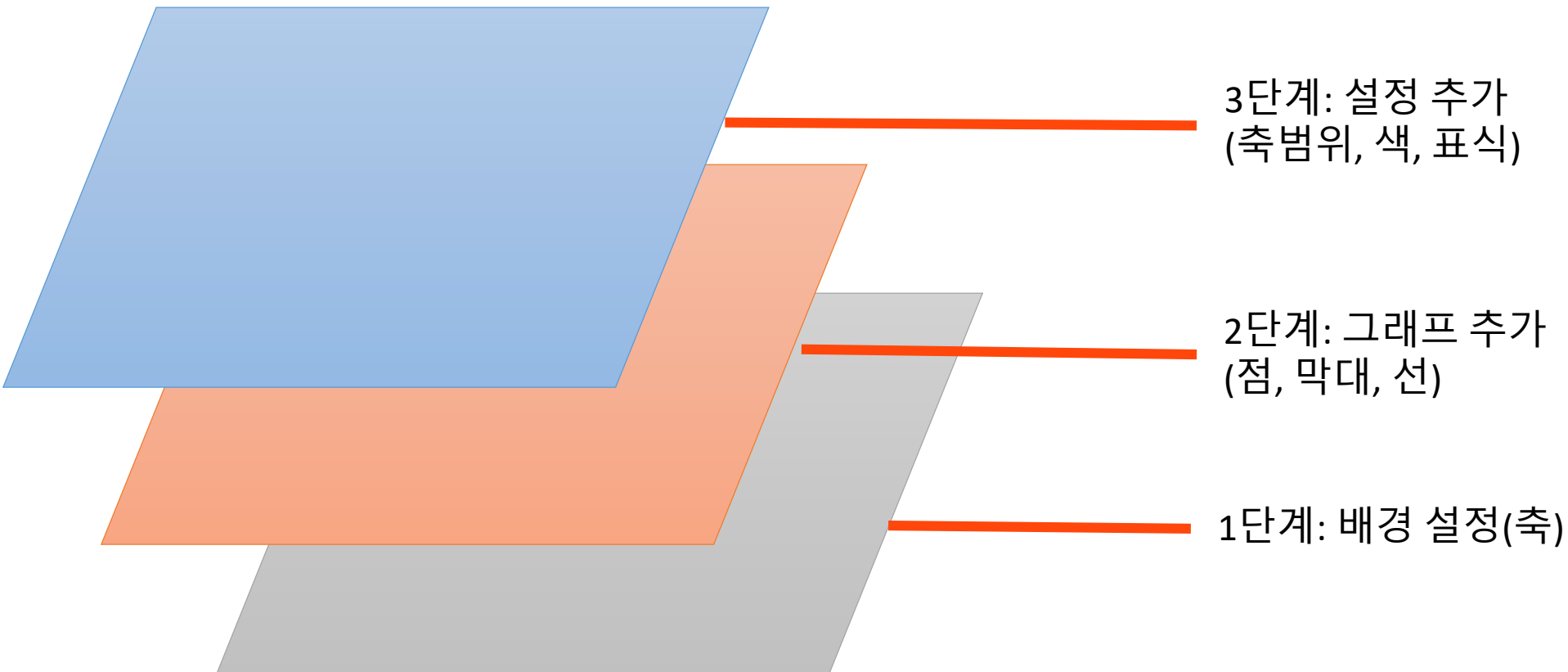
ggplot(data = 데이터 셋명) : 데이터를 불러오는 역할

geom\_function() : 어떤 그래프를 그릴지 정하는 함수

mapping = aes(항목1=값1, 항목2=값2) : geom\_function() 의 옵션으로 사용하여 보여지는 부분을 설정한다.

position(x, y축), color(색상), fill(채우기), shape(모양), linetype(선 형태), size(크기) 등

## ggplot2 레이어 구조 이해하기



## 산점도 - 변수 간 관계 표현하기

```
# ggplot2 패키지 설치하기  
install.packages("ggplot2")  
library(ggplot2)
```

1. R 시각화 패키지인 ggplot2 패키지 설치
2. library(ggplot2)로 패키지 로드하기

```
# 1단계 배경설정(축)  
ggplot(data=mpg, aes(x = displ, y = hwy))
```

```
# 배경에 산점도 추가  
ggplot(data=mpg, aes(x = displ, y = hwy)) + geom_point()
```

```
# x축 범위 3~6으로 지정  
ggplot(data=mpg, aes(x = displ, y = hwy)) + geom_point() + xlim(3,6)
```

```
# x축 범위 3~6, y축 범위 10~30으로 지정  
ggplot(data=mpg, aes(x = displ, y = hwy)) + geom_point() + xlim(3,6) + ylim(10,30)
```

# 산점도 - 변수 간 관계 표현하기

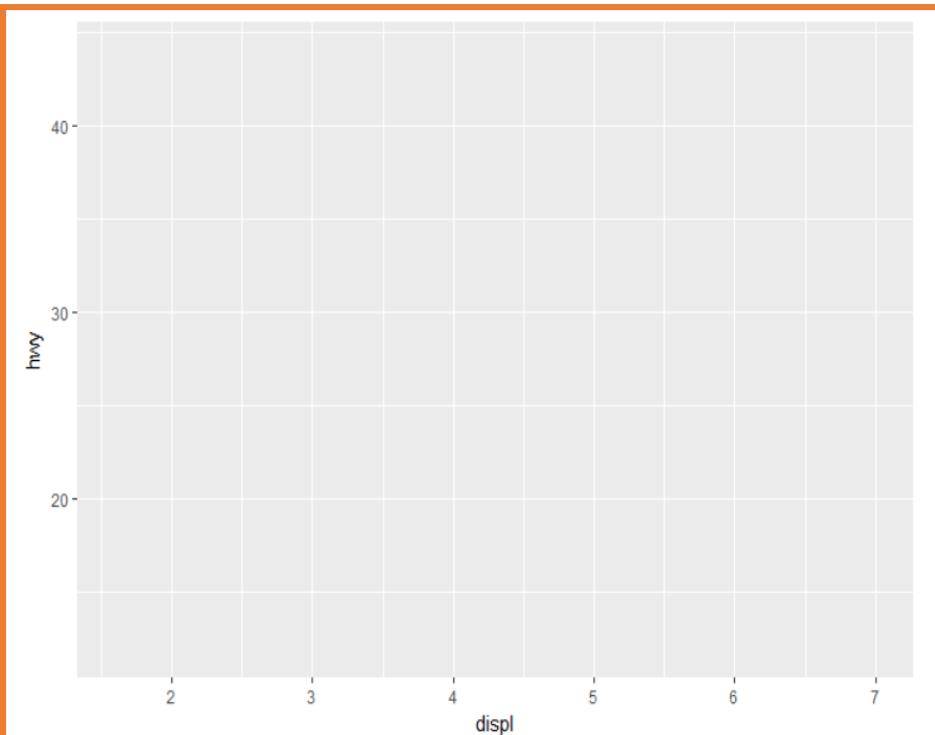
```
# ggplot2 패키지 설치하기  
install.packages("ggplot2")  
library(ggplot2)
```

```
# 1단계 배경설정(축)  
ggplot(data=mpg, aes(x = displ, y = hwy))
```

```
# 배경에 산점도 추가  
ggplot(data=mpg, aes(x = displ, y = hwy)) + geom_point()
```

```
# x축 범위 3~6으로 지정  
ggplot(data=mpg, aes(x = displ, y = hwy)) + geom_point() + xlim(3,6)
```

```
# x축 범위 3~6, y축 범위 10~30으로 지정  
ggplot(data=mpg, aes(x = displ, y = hwy)) + geom_point() + xlim(3,6) + ylim(10,30)
```



# 산점도 - 변수 간 관계 표현하기

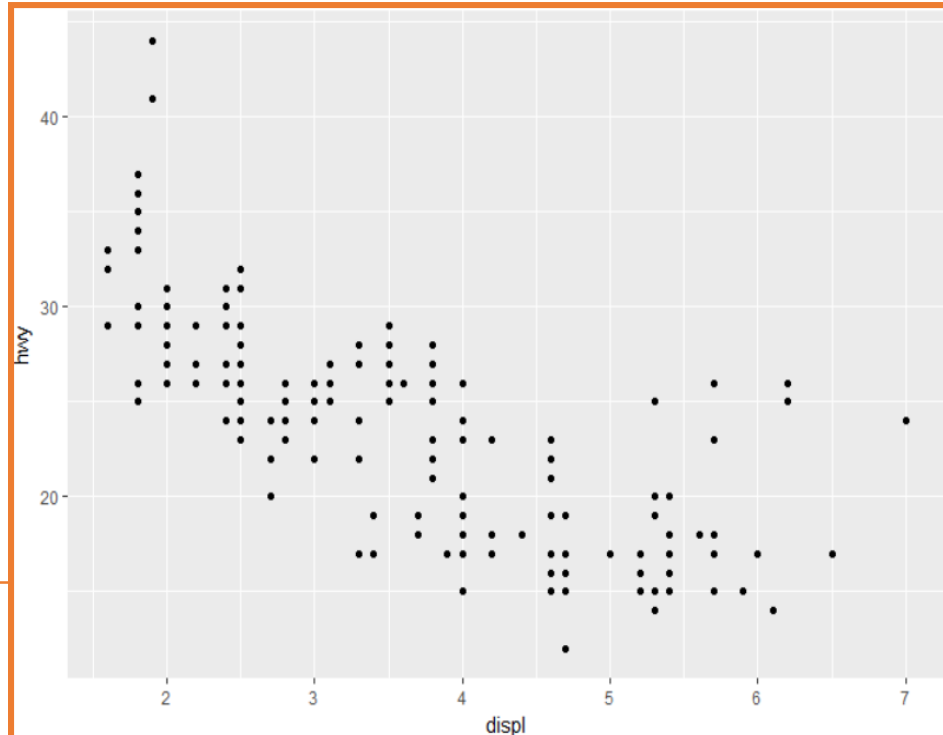
```
# ggplot2 패키지 설치하기  
install.packages("ggplot2")  
library(ggplot2)
```

```
# 1단계 배경설정(축)  
ggplot(data=mpg, aes(x = displ, y = hwy))
```

```
# 배경에 산점도 추가  
ggplot(data=mpg, aes(x = displ, y = hwy)) + geom_point()
```

```
# x축 범위 3~6으로 지정  
ggplot(data=mpg, aes(x = displ, y = hwy)) + geom_point() + xlim(3,6)
```

```
# x축 범위 3~6, y축 범위 10~30으로 지정  
ggplot(data=mpg, aes(x = displ, y = hwy)) + geom_point() + xlim(3,6) + ylim(10,30)
```



# 산점도 - 변수 간 관계 표현하기

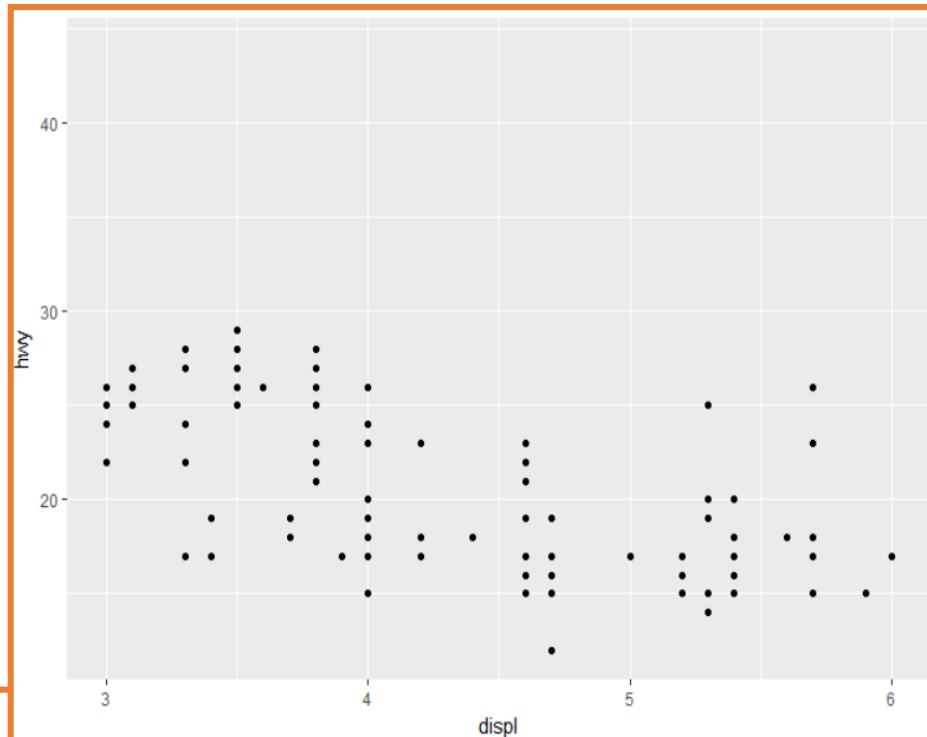
```
# ggplot2 패키지 설치하기  
install.packages("ggplot2")  
library(ggplot2)
```

```
# 1단계 배경설정(축)  
ggplot(data=mpg, aes(x = displ, y = hwy))
```

```
# 배경에 산점도 추가  
ggplot(data=mpg, aes(x = displ, y = hwy)) + geom_point()
```

```
# x축 범위 3~6으로 지정  
ggplot(data=mpg, aes(x = displ, y = hwy)) + geom_point() + xlim(3,6)
```

```
# x축 범위 3~6, y축 범위 10~30으로 지정  
ggplot(data=mpg, aes(x = displ, y = hwy)) + geom_point() + xlim(3,6) + ylim(10,30)
```



# 산점도 - 변수 간 관계 표현하기

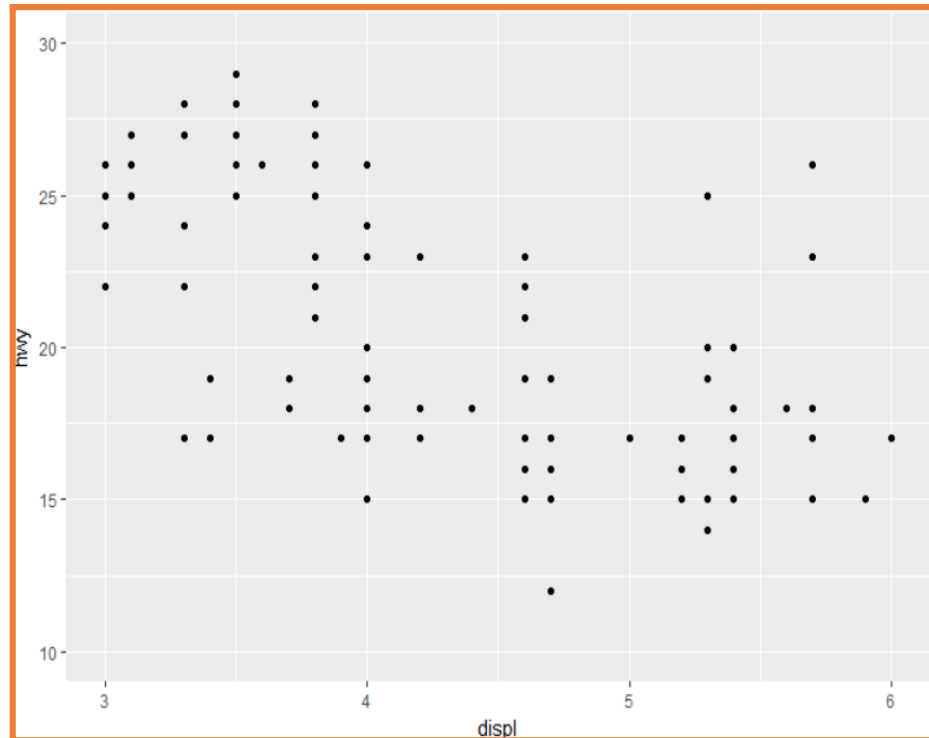
```
# ggplot2 패키지 설치하기  
install.packages("ggplot2")  
library(ggplot2)
```

```
# 1단계 배경설정(축)  
ggplot(data=mpg, aes(x = displ, y = hwy))
```

```
# 배경에 산점도 추가  
ggplot(data=mpg, aes(x = displ, y = hwy)) + geom_point()
```

```
# x축 범위 3~6으로 지정  
ggplot(data=mpg, aes(x = displ, y = hwy)) + geom_point() + xlim(3,6)
```

```
# x축 범위 3~6, y축 범위 10~30으로 지정  
ggplot(data=mpg, aes(x = displ, y = hwy)) + geom_point() + xlim(3,6) + ylim(10,30)
```



# 막대/선 그래프

```
# 데이터 전처리 패키지 설치
install.packages("dplyr")
library(dplyr)
```

```
# mpg 데이터를 drv로 분류하고 hwy평균으로 요약
df_mpg <- mpg %>% group_by(drv) %>%
  summarise(mean_hwy = mean(hwy))
df_mpg
```

```
# geom_col()으로 범주의 크기를 표현
ggplot(data = df_mpg, aes(x = drv, y = mean_hwy)) + geom_col()
```

```
# reorder를 사용하여 범주를 크기순으로 정렬
ggplot(data = df_mpg, aes(x=reorder(drv, -mean_hwy), y = mean_hwy)) + geom_col()
```

```
# 막대 그래프로 표현하기
ggplot(data=mpg, aes(x=drv)) + geom_bar()
ggplot(data=mpg, aes(x=hwy)) + geom_bar()
```

1. 데이터 전처리를 위해 dplyr 패키지를 설치
2. library(dplyr)로 패키지 로드하기

# 막대/선 그래프

```
# 데이터 전처리 패키지 설치
install.packages("dplyr")
library(dplyr)
```

```
# mpg 데이터를 drv로 분류하고 hwy평균으로 요약
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         27.7
3 r         21.0
```

```
# geom_col()으로 범주의 크기를 표현
ggplot(data = df_mpg, aes(x = drv, y = mean_hwy)) + geom_col()
```

```
# reorder를 사용하여 범주를 크기순으로 정렬
ggplot(data = df_mpg, aes(x=reorder(drv, -mean_hwy), y = mean_hwy)) + geom_col()
```

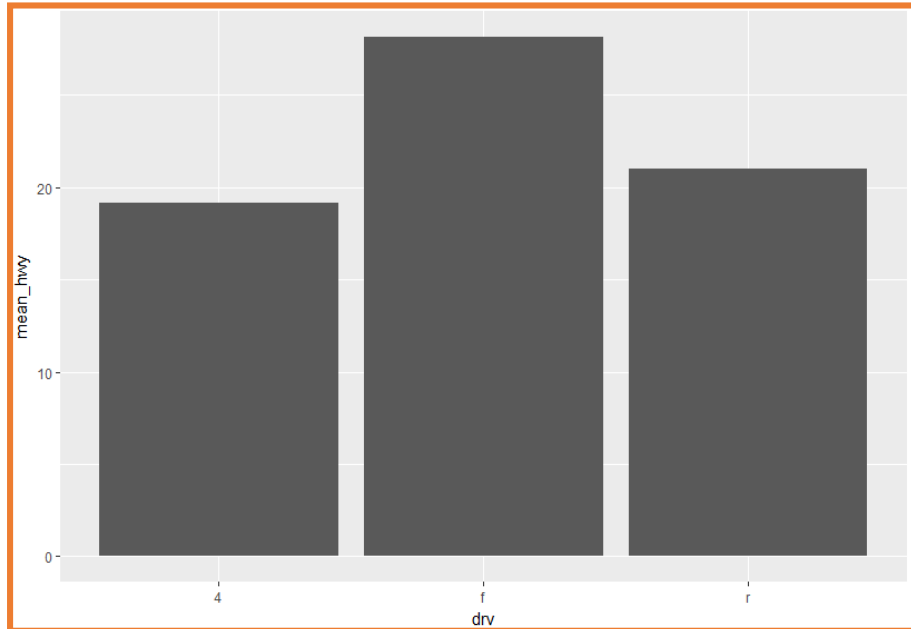
```
# 막대 그래프로 표현하기
ggplot(data=mpg, aes(x=drv)) + geom_bar()
ggplot(data=mpg, aes(x=hwy)) + geom_bar()
```

# 막대/선 그래프

```
# 데이터 전처리 패키지 설치
install.packages("dplyr")
library(dplyr)
```

```
# mpg 데이터를 drv로 분류하고 hwy평균으로 요약
df_mpg <- mpg %>% group_by(drv) %>%
  summarise(mean_hwy = mean(hwy))

df_mpg
```



```
# geom_col()으로 범주의 크기를 표현
ggplot(data = df_mpg, aes(x = drv, y = mean_hwy)) + geom_col()
```

```
# reorder를 사용하여 범주를 크기순으로 정렬
ggplot(data = df_mpg, aes(x=reorder(drv, -mean_hwy), y = mean_hwy)) + geom_col()
```

```
# 막대 그래프로 표현하기
ggplot(data=mpg, aes(x=drv)) + geom_bar()
ggplot(data=mpg, aes(x=hwy)) + geom_bar()
```

# 막대/선 그래프

```
# 데이터 전처리 패키지 설치
install.packages("dplyr")
library(dplyr)
```

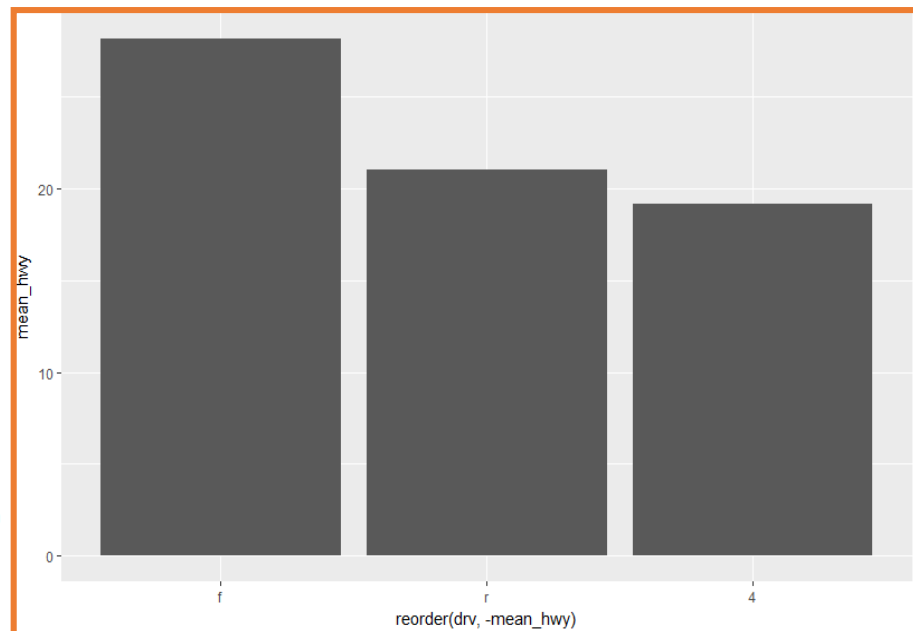
```
# mpg 데이터를 drv로 분류하고 hwy평균으로 요약
df_mpg <- mpg %>% group_by(drv) %>%
  summarise(mean_hwy = mean(hwy))

df_mpg
```

```
# geom_col()으로 범주의 크기를 표현
ggplot(data = df_mpg, aes(x = drv, y = mean_hwy)) + geom_col()
```

```
# reorder를 사용하여 범주를 크기순으로 정렬
ggplot(data = df_mpg, aes(x=reorder(drv, -mean_hwy), y = mean_hwy)) + geom_col()
```

```
# 막대 그래프로 표현하기
ggplot(data=mpg, aes(x=drv)) + geom_bar()
ggplot(data=mpg, aes(x=hwy)) + geom_bar()
```



# 막대/선 그래프

```
# 데이터 전처리 패키지 설치
install.packages("dplyr")
library(dplyr)
```

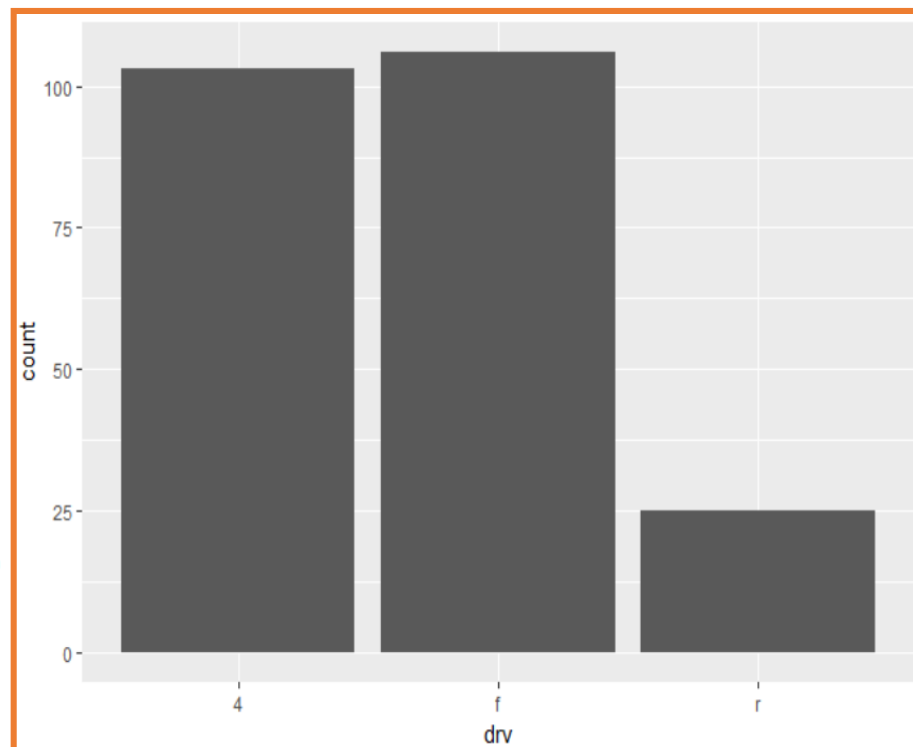
```
# mpg 데이터를 drv로 분류하고 hwy평균으로 요약
df_mpg <- mpg %>% group_by(drv) %>%
  summarise(mean_hwy = mean(hwy))

df_mpg
```

```
# geom_col()으로 범주의 크기를 표현
ggplot(data = df_mpg, aes(x = drv, y = mean_hwy)) + geom_col()
```

```
# reorder를 사용하여 범주를 크기순으로 정렬
ggplot(data = df_mpg, aes(x=reorder(drv, -mean_hwy), y = mean_hwy)) + geom_col()
```

```
# 막대 그래프로 표현하기
ggplot(data=mpg, aes(x=drv)) + geom_bar()
ggplot(data=mpg, aes(x=hwy)) + geom_bar()
```



# 막대/선 그래프

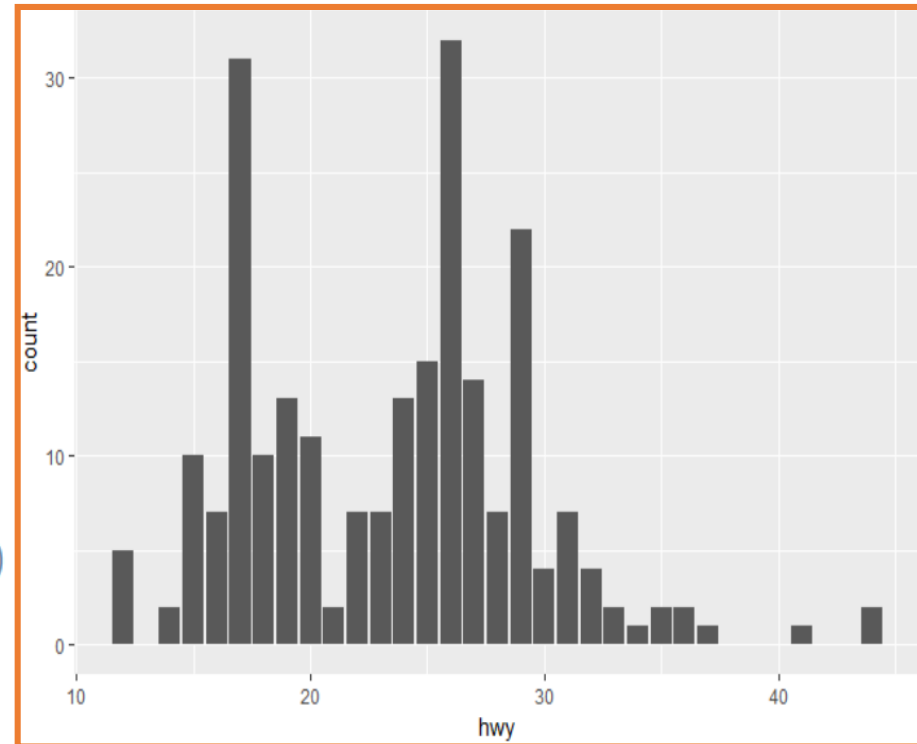
```
# 데이터 전처리 패키지 설치  
install.packages("dplyr")  
library(dplyr)
```

```
# mpg 데이터를 drv로 분류하고 hwy평균으로 요약  
df_mpg <- mpg %>% group_by(drv) %>%  
  summarise(mean_hwy = mean(hwy))  
df_mpg
```

```
# geom_col()으로 범주의 크기를 표현  
ggplot(data = df_mpg, aes(x = drv, y = mean_hwy)) + geom_col()
```

```
# reorder를 사용하여 범주를 크기순으로 정렬  
ggplot(data = df_mpg, aes(x=reorder(drv, -mean_hwy), y = mean_hwy)) + geom_col()
```

```
# 막대 그래프로 표현하기  
ggplot(data=mpg, aes(x=drv)) + geom_bar()  
ggplot(data=mpg, aes(x=hwy)) + geom_bar()
```



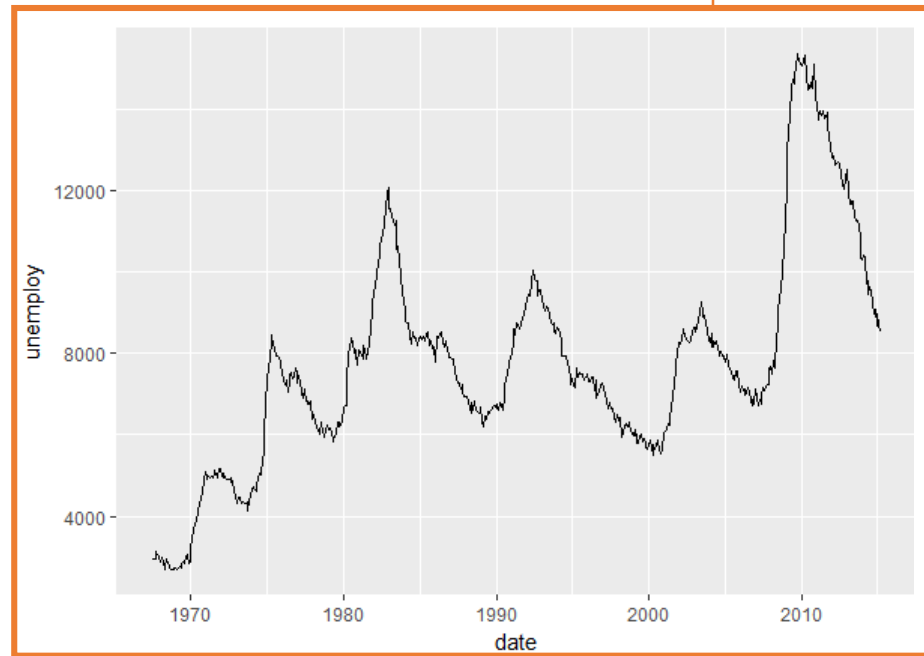
# 막대/선 그래프

# 선 그래프 - 시간에 따라 달라지는 데이터 표현하기

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

# 상자그림 - 집단 간 분포 차이 표현하기

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



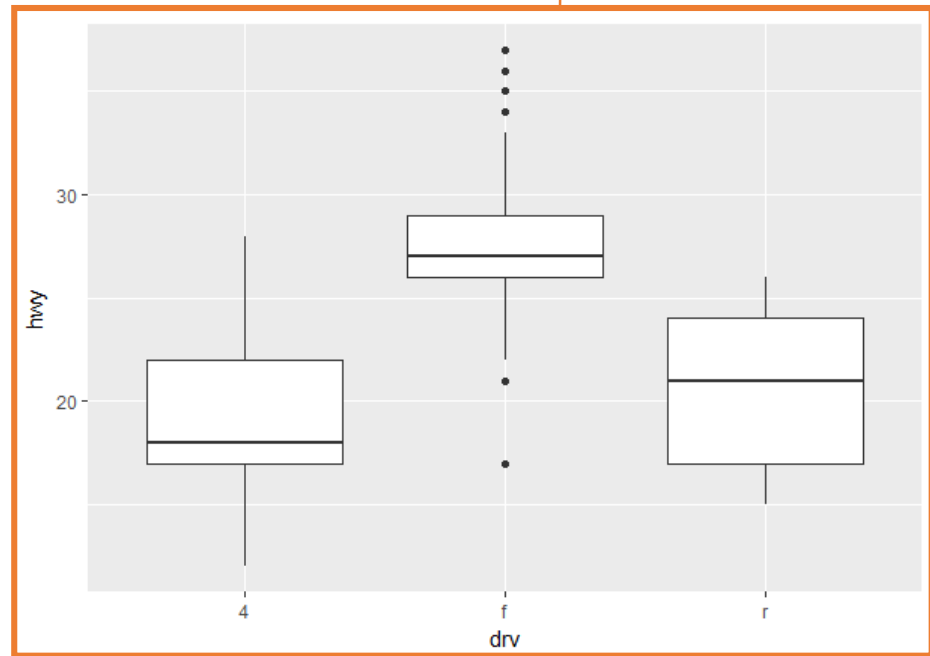
# 막대/선 그래프

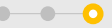
# 선그래프 - 시간에 따라 달라지는 데이터 표현하기

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

# 상자그림 - 집단 간 분포 차이 표현하기

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





## 6. ggmap 패키지



## ggmap 패키지

- ✓ GIS 데이터를 지도위에 표현하는 주제도를 위한 ggplot2기반의 시각화 도구
- ✓ Google Maps 등과 같은 지도 정보를 가져와 표현한다.
- ✓ 원하는 장소를 표현하기 위해서는 경도, 위도 값 혹은 Google Maps에 등록된 지명 이름이 필요하다.
- ✓ 경도, 위도 값은 geocode("찾고싶은지명") 함수를 통해서 얻을 수 있다.

```
> geocode("Incheon")  
Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Incheon&sensor=false  
      lon      lat  
1 126.7052 37.45626
```

```
# 데이터 불러오기
# http://naver.me/5VJtb8wq 데이터 다운받기
data <- read.csv("subway.csv", header=T)
head(data)
```

```
# 필요한 열 뽑아내기 (전처리 작업)
data2 <- data[,c(2,3,8,9)]
colnames(data2) <- c("전철역명", "호선", "x좌표", "y좌표")
head(data2)
```

```
# 2호선만 추출하기
s_2 <- data2 %>% filter(호선=='2')
head(s_2)
str(s_2)
```

```
# 3호선만 추출하기
s_3 <- data2 %>% filter(호선=='3')
head(s_3)
str(s_3)
```

```
> head(data)
  전철역코드 전철역명 호선 외부코드 사이버스테이션 X좌표 Y좌표 X좌표.WGS Y좌표.WGS
1      2729 건대입구   7      727      212 515365 1121815 37.54069 127.0702
2       212 건대입구   2      212      212 515365 1121815 37.54069 127.0702
3       213 구의      2      213      213 518867 1120790 37.53708 127.0859
4       214 강변      2      214      214 520755 1120262 37.53509 127.0947
5       215 잠실나루   2      215      215 522630 1116225 37.52073 127.1038
6      2815 잠실      8      814      216 522390 1114282 37.51395 127.1022
```

```
# 데이터 불러오기
# http://naver.me/5VJtb8wq 데이터 다운받기
data <- read.csv("subway.csv", header=T)
head(data)
```

```
# 필요한 열 뽑아내기 (전처리 작업)
data2 <- data[,c(2,3,8,9)]
colnames(data2) <- c("전철역명", "호선", "x좌표", "y좌표")
head(data2)
```

```
# 2호선만 추출하기
s_2 <- data2 %>% filter(호선=='2')
head(s_2)
str(s_2)
```

```
# 3호선만 추출하기
s_3 <- data2 %>% filter(호선=='3')
head(s_3)
str(s_3)
```

```
> head(data2)
  전철역명 호선   x좌표   y좌표
1 건대입구   7 37.54069 127.0702
2 건대입구   2 37.54069 127.0702
3 구의       2 37.53708 127.0859
4 강변       2 37.53509 127.0947
5 잠실나루   2 37.52073 127.1038
6 잠실       8 37.51395 127.1022
```

```
# 데이터 불러오기
# http://naver.me/5VJtb8wq 데이터 다운받기
data <- read.csv("subway.csv", header=T)
head(data)

# 필요한 열 뽑아내기 (전처리 작업)
data2 <- data[,c(2,3,8,9)]
colnames(data2) <- c("전철역명", "호선", "x좌표", "y좌표")
head(data2)
```

```
# 2호선만 추출하기
s_2 <- data2 %>% filter(호선=='2')
head(s_2)
str(s_2)
```

```
# 3호선만 추출하기
s_3 <- data2 %>% filter(호선=='3')
head(s_3)
str(s_3)
```

```
> head(s_2)
  전철역명 호선   x좌표   y좌표
1 건대입구   2 37.54069 127.0702
2 구의       2 37.53708 127.0859
3 강변       2 37.53509 127.0947
4 잠실나루   2 37.52073 127.1038
5 잠실       2 37.51395 127.1022
6 잠실새내   2 37.51169 127.0862

> str(s_2)
'data.frame': 51 obs. of 4 variables:
 $ 전철역명: Factor w/ 582 levels "가능","가락시장",...: 29 73 21 479 478 480 496 2
60 297 403 ...
 $ 호선 : Factor w/ 20 levels "1","2","3","4",...: 2 2 2 2 2 2 2 2 2 ...
 $ x좌표 : num 37.5 37.5 37.5 37.5 37.5 ...
 $ y좌표 : num 127 127 127 127 127 ...
```

```
# 데이터 불러오기
# http://naver.me/5VJtb8wq 데이터 다운받기
data <- read.csv("subway.csv", header=T)
head(data)
```

```
# 필요한 열 뽑아내기 (전처리 작업)
data2 <- data[,c(2,3,8,9)]
colnames(data2) <- c("전철역명", "호선", "x좌표", "y좌표")
head(data2)
```

```
# 2호선만 추출하기
s_2 <- data2 %>% filter(호선=='2')
head(s_2)
str(s_2)
```

```
# 3호선만 추출하기
s_3 <- data2 %>% filter(호선=='3')
head(s_3)
str(s_3)
```

```
> head(s_3)
  전철역명 호선   x좌표   y좌표
1 종로3가   3 37.57161 126.9918
2 지축     3 37.64805 126.9140
3 구파발   3 37.63676 126.9188
4 연신내   3 37.61900 126.9210
5 불광     3 37.61047 126.9299
6 녹번     3 37.60093 126.9358

> str(s_3)
'data.frame': 44 obs. of 4 variables:
 $ 전철역명: Factor w/ 582 levels "가능","가락시장",..
.: 494 512 75 406 249 113 570 203 147 37 ...
 $ 호선    : Factor w/ 20 levels "1","2","3","4",..
.: 3 3 3 3 3 3 3 3 3 3 3 ...
 $ x좌표   : num  37.6 37.6 37.6 37.6 37.6 ...
 $ y좌표   : num  127 127 127 127 127 ...
```

```
# get_map을 이용하여 지도 받아오기
center <- c(mean(s_2$y좌표), mean(s_2$x좌표))
seoul <- get_map(center, zoom=11, maptype="roadmap")
```

```
# R Graphic Device
x11()
ggmap(seoul)
```

```
# 지도에 2호선 표시하기
ggmap(seoul) + geom_point(data=s_2,
  aes(x=y좌표, y=x좌표),
  size=2.5,
  alpha=0.7, #투명도 조절
  col="green") +
  geom_text(data=s_2, aes(x=y좌표, y=x좌표+0.005,
    label=전철역명),
    size=2.4) #label : 지도에 역명 표시
```

```
# 2, 3호선 같이 표시하기
ggmap(seoul) + geom_point(data=s_2,
  aes(x=y좌표, y=x좌표),
  size=2.5,
  alpha=0.7,
  col="green") +
  geom_point(data=s_3, aes(x=y좌표, y=x좌표),
    size=2.5, alpha=0.7, col='orange') +
  geom_text(data=s_2, aes(x=y좌표, y=x좌표+0.005, label=전철역명),
    size=2.4)+
  geom_text(data=s_3, aes(x=y좌표, y=x좌표+0.005, label=전철역명),
    size=2.8) #label : 지도에 역명 표시
```

```
> center <- c(mean(s_2$y좌표), mean(s_2$x좌표))
> seoul <- get_map(center, zoom=11, maptype="roadmap")
Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=37.52872,126.986613&zoom=11&size=640x640&scale=2&maptype=roadmap&language=en-EN&sensor=false
```

get\_map() : 지도를 얻는 함수  
 center : 좌표값(경도, 위도)  
 zoom : 확대, 축소 옵션  
 maptype : 지도의 유형

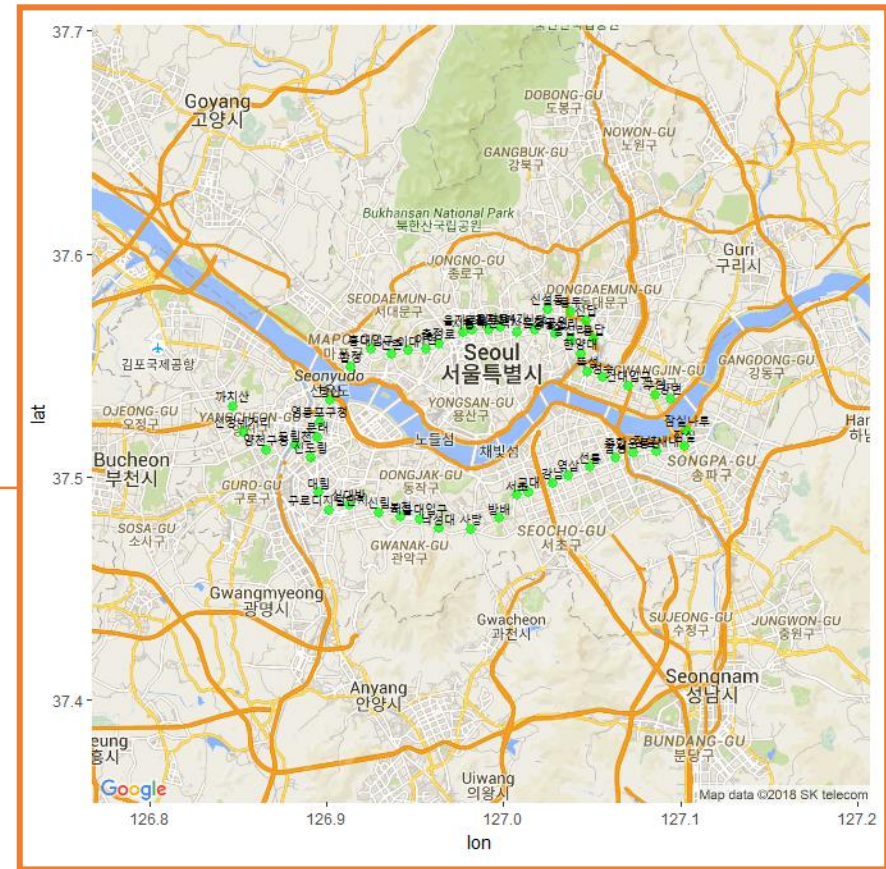


```
# get_map을 이용하여 지도 받아오기
center <- c(mean(s_2$y좌표), mean(s_2$x좌표))
seoul <- get_map(center, zoom=11, maptype="roadmap")
```

```
# R Graphic Device
x11()
ggmap(seoul)
```

```
# 지도에 2호선 표시하기
ggmap(seoul) + geom_point(data=s_2,
                           aes(x=y좌표, y=x좌표),
                           size=2.5,
                           alpha=0.7, #투명도 조절
                           col="green") +
  geom_text(data=s_2, aes(x=y좌표, y=x좌표+0.005,
                           label=전철역명),
            size=2.4) #label : 지도에 역명 표시
```

```
# 2, 3호선 같이 표시하기
ggmap(seoul) + geom_point(data=s_2,
                           aes(x=y좌표, y=x좌표),
                           size=2.5,
                           alpha=0.7,
                           col="green") +
  geom_point(data=s_3, aes(x=y좌표, y=x좌표),
            size=2.5, alpha=0.7, col='orange') +
  geom_text(data=s_2, aes(x=y좌표, y=x좌표+0.005, label=전철역명),
            size=2.4)+
  geom_text(data=s_3, aes(x=y좌표, y=x좌표+0.005, label=전철역명),
            size=2.8) #label : 지도에 역명 표시
```



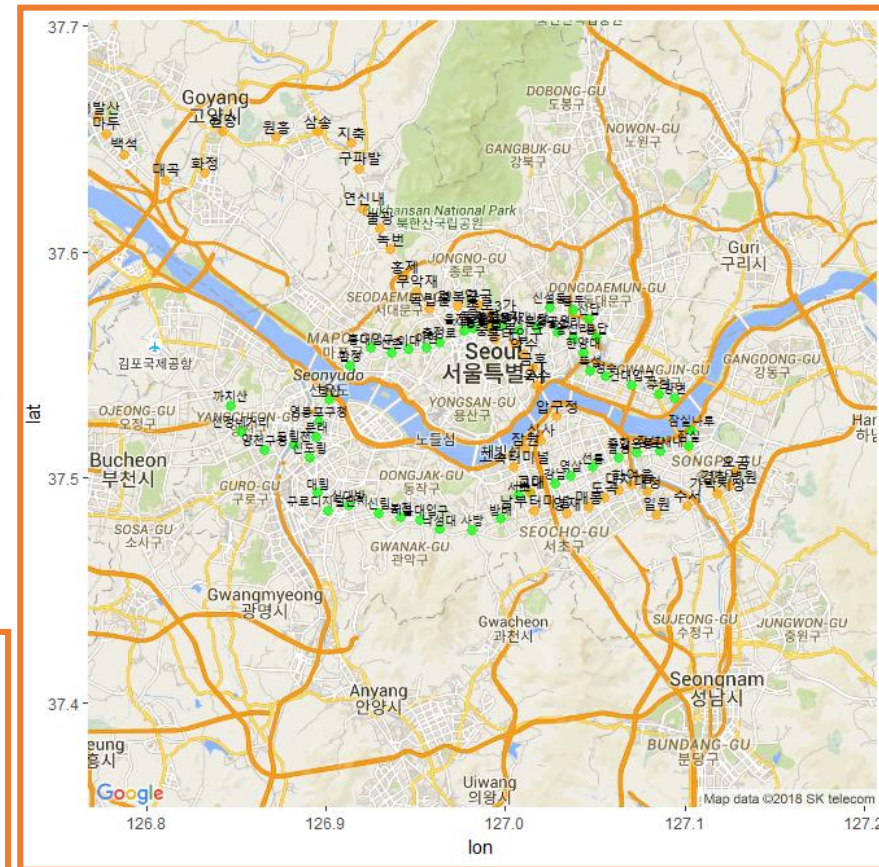
geom\_point() : 산점도를 표현하는 함수  
 aes : 좌표(x,y)에 경도, 위도를 차례대로 입력한다.

```
# get_map을 이용하여 지도 받아오기
center <- c(mean(s_2$y좌표), mean(s_2$x좌표))
seoul <- get_map(center, zoom=11, maptype="roadmap")

# R Graphic Device
x11()
ggmap(seoul)

# 지도에 2호선 표시하기
ggmap(seoul) + geom_point(data=s_2,
                           aes(x=y좌표, y=x좌표),
                           size=2.5,
                           alpha=0.7, #투명도 조절
                           col="green") +
  geom_text(data=s_2, aes(x=y좌표, y=x좌표+0.005,
                          label=전철역명),
            size=2.4) #label : 지도에 역명 표시
```

```
# 2, 3호선 같이 표시하기
ggmap(seoul) + geom_point(data=s_2,
                           aes(x=y좌표, y=x좌표),
                           size=2.5,
                           alpha=0.7,
                           col="green") +
  geom_point(data=s_3, aes(x=y좌표, y=x좌표),
            size=2.5, alpha=0.7, col='orange') +
  geom_text(data=s_2, aes(x=y좌표, y=x좌표+0.005, label=전철역명),
            size=2.4)+
  geom_text(data=s_3, aes(x=y좌표, y=x좌표+0.005, label=전철역명),
            size=2.8) #label : 지도에 역명 표시
```

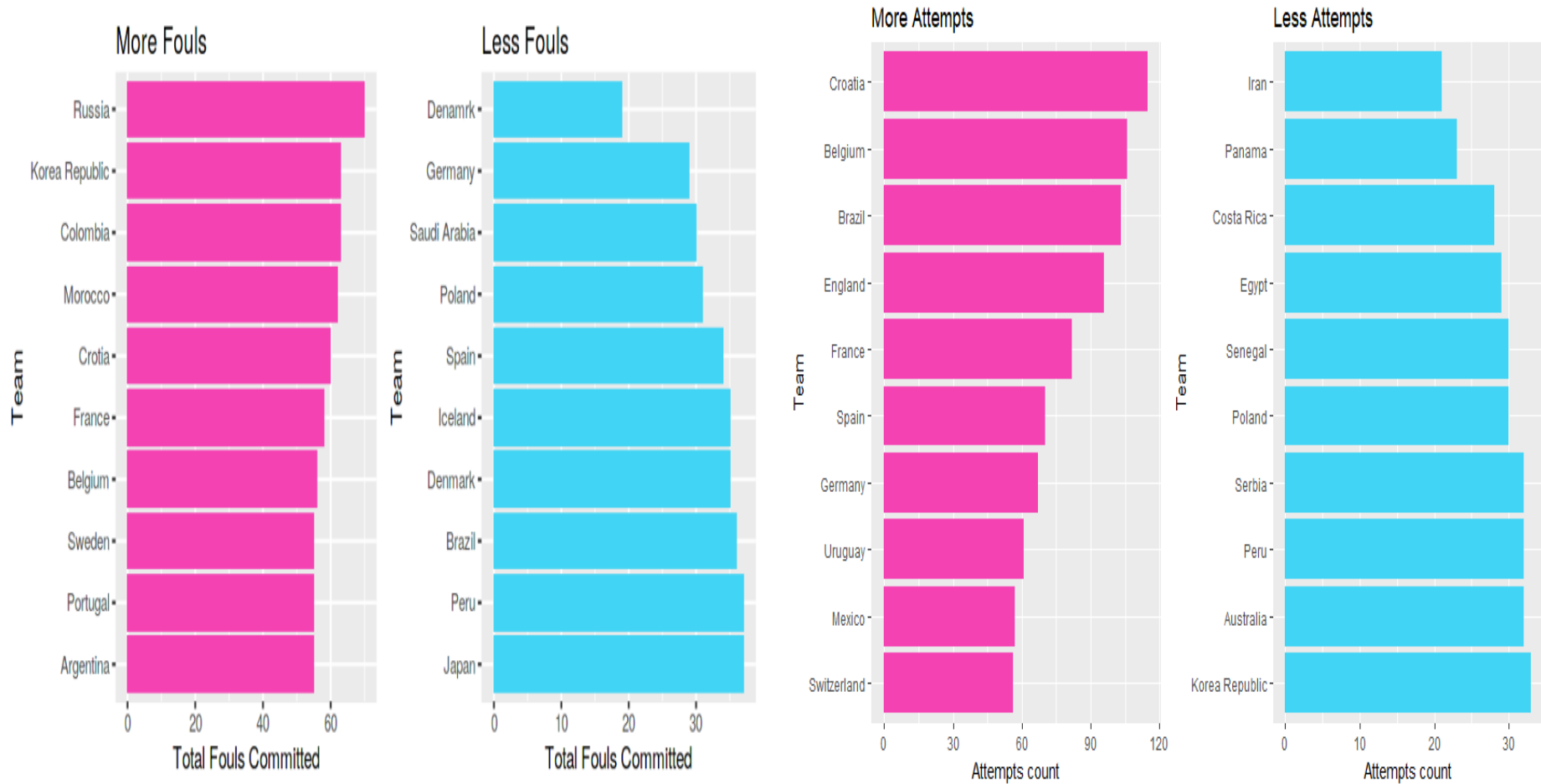




## 7. 시각화 실전예제



## Kaggle, FIFA 월드컵 데이터 분석 예제





# Kaggle이란?





캐글(Kaggle)은 2010년에 설립된 예측모델 및 분석 대회 플랫폼이다.

기업 및 단체에서 데이터와 해결과제를 등록하면, 데이터 과학자들이 이를 해결하는 모델을 개발하고 경쟁한다.



<https://www.kaggle.com/>

Q
Competitions
Datasets
Kernels
Discussion
Learn
...

**Eric Antoine Scuccimarra**

replied to a message  
an hour ago

**Distribution of the dataset**

started by [meryam elmounne](#) 5 days ago

[https://github.com/escuccim/mias-mammography/blob/master/write\\_to\\_tfrecords\\_9.ipynb](https://github.com/escuccim/mias-mammography/blob/master/write_to_tfrecords_9.ipynb)

The last 3 cells at the bottom of the notebook read in the data from the tfrecords files and display selected images. You would need to modify the code to read all of the images from the tfrecords files and save them to disk instead of displaying them.

in [DDSM Mammography](#)
6

**Mark**

commented on a kernel  
2 hours ago

**Summary functions and maps workbook**

created by [Yash Shevde](#) 4 months ago

From the [Pandas documentation](#) it says:

Return the row label of the maximum value. If multiple values equal the maximum, the first row label with that value is returned.

**ChanKyeong Kim**  
 Joined 20 days ago


**Novice**

- ☐ Add your bio
- ☐ Add your location
- ☐ Add your occupation
- ☐ Add your organization
- ☒ SMS verify your account
- ☐ [Run 1 kernel](#)
- ☐ [Make 1 competition submission](#)
- ☒ Make 1 comment
- ☒ Cast 1 upvote

WorldQuant Research (Israel) Ltd. is hiring

**Financial Data Scientist**

📍 Ramat Gan, Israel



Competitions
Datasets
Kernels
Discussion
Learn
...
🔔
👤


## Competitions

Documentation
InClass


General
InClass
Sort by
Grouped


All Categories

### 2 Entered Competitions




**Titanic: Machine Learning from Disaster**  
 Start here! Predict survival on the Titanic and get familiar with ML basics  
Getting Started · Ongoing · 📄 tutorial, tabular data, binary classification



**Knowledge**  
 10,155 teams



**House Prices: Advanced Regression Techniques**  
 Predict sales prices and practice feature engineering, RFs, and gradient boosting  
Getting Started · Ongoing · 📄 tabular data, regression



**Knowledge**  
 4,672 teams

### 17 Active Competitions




**TGS Salt Identification Challenge**  
 Segment salt deposits beneath the Earth's surface  
Featured · 2 months to go · 📄 geology, image data

**\$100,000**  
 982 teams




**Home Credit Default Risk**  
 Can you predict how capable each applicant is of repaying a loan?  
Featured · 22 days to go · 📄 home, banking, tabular data

**\$70,000**  
 6,006 teams




**Airbus Ship Detection Challenge**  
 Find ships on satellite images as quickly as possible  
Featured · 2 months to go · 📄 image data, object detection, object segmentation

**\$60,000**  
 225 teams




**Santander Value Prediction Challenge**  
 Predict the value of transactions for potential customers.  
Featured · 13 days to go · 📄 finance, banking

**\$60,000**  
 4,184 teams



**Google AI Open Images - Object Detection Track**  
 Detect objects in varied and complex images.  
Featured · 23 days to go

**\$30,000**  
 340 teams



**TrackML Particle Tracking Challenge**  
 High Energy Physics particle tracking in CERN detectors

**\$25,000**  
 625 teams

**Kaggle** Search kaggle Competitions **Datasets** Kernels Discussion Learn ...

**Datasets** Documentation New Dataset

Join Kaggle's newest Data Science for Good challenge with PASSNYC. Click to learn more and participate to win from \$15,000 in prizes.

Public Your Datasets Favorites Sort by Hotness

9,298 Datasets Sizes File types Licenses Tags Search datasets

Rank	Dataset Name	Description	Tags	File Type	Size	License	Views	Downloads
56	<b>PASSNYC: Data Science for Good Challenge</b>	Help PASSNYC determine which schools need their services the most PASSNYC updated a month ago	education, demograph..., data visuali..., recommen...	CSV	163.8 KB	CC0	142	22
106	<b>120 years of Olympic history: athletes and results</b>	basic bio data on athletes and medal results from Athens 1896 to Rio 2016 Randi H Griffin updated 2 months ago	olympic ga..., sports, history	CSV	5.4 MB	CC0	26	1
14	<b>Telco Customer Churn</b>	Focused customer retention programs BlastChar updated 6 months ago	telecommu..., churn analy...	CSV	171.6 KB	Other	6	0
41	<b>Boxing bouts</b>	Various information about a lot of boxing matches Alexander Slonsky updated 4 months ago	boxing, sports	CSV	5.3 MB	CC0	8	3
145	<b>Mobile App Store (7200 apps)</b>	Analytics for Mobile Apps Ramanathan updated 2 months ago	business, internet, mobile web	CSV	5.6 MB	GPL	37	7
44	<b>Air Quality in Madrid (2001-2018)</b>	Different pollution levels in Madrid from 2001 to 2018 Decide Soluciones updated 2 months ago	time series, pollution	Other	150.6 MB	Other	10	2
9	<b>Seattle Trade Permits</b>	From City of Seattle Open Data City of Seattle updated 15 days ago	socrata	CSV	13.3 MB	CC0	1	0
25	<b>Genetic Variant Classifications</b>	Predict whether a variant will have conflicting clinical classifications. Kevin Arvai updated 4 months ago	healthcare, human gen..., biology, + 2 more...	CSV	3.4 MB	CC0	3	1

The screenshot shows the Kaggle website interface for the 'PASSNYC: Data Science for Good Challenge'. The top navigation bar includes links for Competitions, Datasets, Kernels, Discussion, and Learn. The main header features the Kaggle logo and a search bar. Below the header, a featured dataset banner for 'PASSNYC: Data Science for Good Challenge' is displayed, with a '562 voters' badge. The dataset is by 'PASSNYC and 6 collaborators' and was last updated a month ago. A navigation bar below the banner has tabs for 'Data', 'Overview', 'Kernels', 'Discussion', and 'Activity'. The 'Data' tab is highlighted with a red box. To the right of the tabs are buttons for 'Download (164 KB)' and 'New Kernel'. The main content area shows the dataset details for 'Data (164 KB)'. It includes an API link, a download command, and a 'Download All' button. The content is divided into three sections: 'Data Sources' (listing '2016 School Explore...' and 'D5 SHSA1 Registrat...'), 'About this file' (describing 'PASSNYC School Explorer'), and 'Columns' (listing various attributes like 'Adjusted Grade', 'New?', 'Other Location Code in LCGMS', 'School Name', 'SED Code', 'Location Code', 'District', and 'Latitude').

Data 탭: Data에 대한 설명 review 및 다운로드가 가능

**Featured Dataset**

## PASSNYC: Data Science for Good Challenge

Help PASSNYC determine which schools need their services the most

PASSNYC and 6 collaborators · last updated a month ago

562 voters

Data Overview **Kernels** Discussion Activity

Download (164 KB) New Kernel

Public Your Work Favorites

Sort by Hotness

Outputs Languages Types Search kernels

Rank	Kernel Title	Author	Time Ago	Tags	Views	Py	R	Rmd	Comments
1	PASSNYC COMPETITION: PROXIMITY BASED ANALYSIS	1	2h ago		0				0
50	Make them SHSAT ready (Model, prblms & Solns)		13h ago	eda, data cleaning, data visualization, binary classification	31				31
7	First PASS(NYC) at Kaggle		3h ago		2				2
18	School's of New York .		16h ago	eda, geospatial analysis, data visualization, clustering	6				6
57	Recommendations to PASSNYC regarding the SHSAT		19h ago	storytelling, geospatial analysis, data visualization, recommendation	16				16
1	NYC School District-level Pattern Recognition!		1h ago	united states, beginner, eda, data visualization, pattern recognition	0				0
0	PASSNYC. Numeric and Categorical Variables R		1h ago	data visualization, preprocessing	0				0
20	Recommendations to PASSNYC based on Data Analysis		17h ago		8				8
9	Driving factors, Model and Proxy measures		12h ago		0				0

## Kernels 탭 : 분석과제에 대한 Solution 수록

Source : <https://www.kaggle.com/passnyc/data-science-for-good>

The screenshot shows the Kaggle website interface for the 'PASSNYC: Data Science for Good Challenge'. The header includes the Kaggle logo, a search bar, and navigation links for Competitions, Datasets, Kernels, Discussion, and Learn. The main banner features the challenge title and a '562 voters' badge. Below the banner, there are tabs for Data, Overview, Kernels (selected), Discussion, and Activity. A 'Download (164 KB)' button and a 'New Kernel' button are also visible. The 'Kernels' section shows a list of public kernels. A red box highlights the 'Languages' dropdown menu, which is open, showing options: R, Python, R, SQLite, and Julia. The list of kernels includes titles like 'First PASS(NYC) at Kaggle', 'Recommendations to PASSNYC regarding...', 'NYC School District-level Pattern Recognition...', 'PASSNYC. Numeric and Categorical Variables R', 'Target Schools & Action Recommended to PASSNYC', 'PASSNYC Modelling Approach Summary', 'Schools In Need | PASSNYC', 'Recommendations to PASSNYC for Maximum Impact', and 'Recommendations for PASSNYC'.

Languages 탭 : 분석과제에 대한 Solution 수록

```
install.packages("tidyverse")
library(tidyverse)
options(repr.plot.width=8, repr.plot.height=4)
mypink = "#f442b3"
myblue = "#42d4f4"

setwd("C:\\Users\\EXEMPC\\Desktop\\EduData\\Dataset")
fifa18 <- read.csv("FIFA.CSV")
```

“tidyverse” : 공통된 원리를 공유하고 함께 원활하게 작동되도록 설계된 R 패키지 모음

```
fifa18$bpp2 <- 100 - fifa18$Ball.Possession..
fifa18$bpp_diff <- fifa18$Ball.Possession.. - fifa18$bpp2

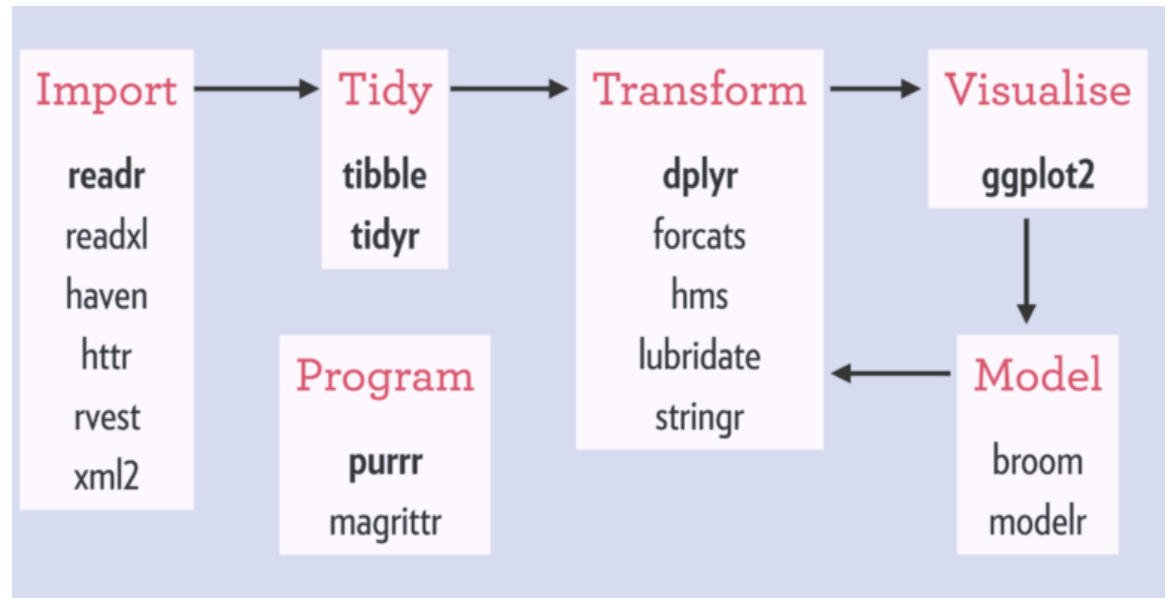
cat("Total Teams: ",length(unique(fifa18$Team)),"\n")
# cat(paste(fifa18$Team,collapse = "\n"))
unique(fifa18$Team)

# which Team in which Match had the maximum ball possession?
fifa18 %>%
  arrange(desc(Ball.Possession..)) %>%
  mutate(Team = reorder(Team,Ball.Possession..)) %>%
  head(1) %>%
  select(Date:Ball.Possession..)

# which match had the closest fight in terms of Ball Possession % ?
fifa18 %>%
  arrange(abs(bpp_diff)) %>%
  head(1) %>%
  select(Team, Ball.Possession..)
```

## (참고) tidyverse 패키지

Components



```
install.packages("tidyverse")
library(tidyverse)
options(repr.plot.width=8, repr.plot.height=4)
mypink = "#f442b3"
myblue = "#42d4f4"

setwd("C:\\\\Users\\\\EXEMPC\\\\Desktop\\\\EduData\\\\Dataset")
fifa18 <- read.csv("FIFA.CSV")

fifa18$bpp2 <- 100 - fifa18$Ball.Possession..
fifa18$bpp_diff <- fifa18$Ball.Possession.. - fifa18$bpp2

cat("Total Teams: ",length(unique(fifa18$Team)),"\n")
# cat(paste(fifa18$Team,collapse = "\n"))
unique(fifa18$Team)

# Which Team in which Match had the maximum ball possession?
fifa18 %>%
  arrange(desc(Ball.Possession..)) %>%
  mutate(Team = reorder(Team,Ball.Possession..)) %>%
  head(1) %>%
  select(Date:Ball.Possession..)

# Which match had the closest fight in terms of Ball Possession % ?
fifa18 %>%
  arrange(abs(bpp_diff)) %>%
  head(1) %>%
  select(Team, Ball.Possession..)
```

options() : global 옵션 설정

“repr.plot.width=8 , repr.plot.height=4”  
: 그래프의 폭을 8, 높이를 4로 설정

```
install.packages("tidyverse")
library(tidyverse)
options(repr.plot.width=8, repr.plot.height=4)
mypink = "#f442b3"
myblue = "#42d4f4"
```

“mypink , myblue” : 그래프에 표현할  
색상을 RGB컬러로 설정

```
setwd("C:\\Users\\EXEMPC\\Desktop\\EduData\\Dataset")
fifa18 <- read.csv("FIFA.CSV")
```

```
fifa18$bpp2 <- 100 - fifa18$Ball.Possession..
fifa18$bpp_diff <- fifa18$Ball.Possession.. - fifa18$bpp2
```

```
cat("Total Teams: ",length(unique(fifa18$Team)),"\n")
# cat(paste(fifa18$Team,collapse = "\n"))
unique(fifa18$Team)
```

```
# which Team in which Match had the maximum ball possession?
```

```
fifa18 %>%
  arrange(desc(Ball.Possession..)) %>%
  mutate(Team = reorder(Team,Ball.Possession..)) %>%
  head(1) %>%
  select(Date:Ball.Possession..)
```

```
# which match had the closest fight in terms of Ball Possession % ?
```

```
fifa18 %>%
  arrange(abs(bpp_diff)) %>%
  head(1) %>%
  select(Team, Ball.Possession..)
```

```
install.packages("tidyverse")
library(tidyverse)
options(repr.plot.width=8, repr.plot.height=4)
mypink = "#f442b3"
myblue = "#42d4f4"
```

```
setwd("C:\\Users\\EXEMPC\\Desktop\\EduData\\Dataset")
fifa18 <- read.csv("FIFA.CSV")
```

```
fifa18$bpp2 <- 100 - fifa18$Ball.Possession..
fifa18$bpp_diff <- fifa18$Ball.Possession.. - fifa18$bpp2
```

```
cat("Total Teams: ", length(unique(fifa18$Team)), "\n")
# cat(paste(fifa18$Team, collapse = "\n"))
unique(fifa18$Team)
```

```
# which Team in which Match had the maximum ball possession?
```

```
fifa18 %>%
  arrange(desc(Ball.Possession..)) %>%
  mutate(Team = reorder(Team, Ball.Possession..)) %>%
  head(1) %>%
  select(Date:Ball.Possession..)
```

```
# which match had the closest fight in terms of Ball Possession % ?
```

```
fifa18 %>%
  arrange(abs(bpp_diff)) %>%
  head(1) %>%
  select(Team, Ball.Possession..)
```

setwd() : 워킹디렉토리 설정  
read.csv(): csv 파일 불러오기

▲	Date	Team	Opponent	Goal.Scored	Ball.Possession..	Attempts	On.Target	Off.Target	Blocked	Corners	Offsides
1	14-06-2018	Russia	Saudi Arabia	5	40	13	7	3	3	6	3
2	14-06-2018	Saudi Arabia	Russia	0	60	6	0	3	3	2	1
3	15-06-2018	Egypt	Uruguay	0	43	8	3	3	2	0	1
4	15-06-2018	Uruguay	Egypt	1	57	14	4	6	4	5	1
5	15-06-2018	Morocco	Iran	0	64	13	3	6	4	5	0
6	15-06-2018	Iran	Morocco	1	36	8	2	5	1	2	0
7	15-06-2018	Portugal	Spain	3	39	8	3	2	3	4	1
8	15-06-2018	Spain	Portugal	3	61	12	5	5	2	5	3
9	16-06-2018	France	Australia	2	51	12	5	4	3	5	0
10	16-06-2018	Australia	France	1	49	4	1	2	1	1	0
11	16-06-2018	Argentina	Iceland	1	72	26	7	9	10	10	0
12	16-06-2018	Iceland	Argentina	1	28	9	3	5	1	2	0
13	16-06-2018	Peru	Denmark	0	52	18	6	7	5	3	5
14	16-06-2018	Denmark	Peru	1	48	10	3	5	2	7	3
15	17-06-2018	Croatia	Nigeria	2	54	11	2	7	2	6	2
16	17-06-2018	Nigeria	Croatia	0	46	14	2	5	7	5	1
17	17-06-2018	Costa Rica	Serbia	0	50	10	3	3	4	5	1
18	17-06-2018	Serbia	Costa Rica	1	50	10	3	5	2	4	3
19	17-06-2018	Germany	Mexico	0	60	25	9	9	7	8	1
20	17-06-2018	Mexico	Germany	1	40	12	4	6	2	1	2
21	17-06-2018	Brazil	Switzerland	1	52	20	4	9	7	7	1
22	17-06-2018	Switzerland	Brazil	1	40	6	2	4	2	2	1

Date : 시합 날짜

Team :시합팀

Opponent : 상대팀

Goal Scored : 득점 수

Ball Possession% : 볼 점유율

Attempts : 득점 시도 횟수

On-Target : 유효슈팅

Off-Target : 비유효슈팅

Corners : 코너킥 수

Offsides : 오프사이드 수

Free Kicks : 프리킥 수

Saves : 골키퍼 선방 수

Passes : 팀 패스 횟수

Distance Covered (Kms) : 팀원들의 총 이동거리

:

```
install.packages("tidyverse")
library(tidyverse)
options(repr.plot.width=8, repr.plot.height=4)
mypink = "#f442b3"
myblue = "#42d4f4"

setwd("C:\\Users\\EXEMPC\\Desktop\\EduData\\Dataset")
fifa18 <- read.csv("FIFA.CSV")
```

```
fifa18$bpp2 <- 100 - fifa18$Ball.Possession..
fifa18$bpp_diff <- fifa18$Ball.Possession.. - fifa18$bpp2
```

```
cat("Total Teams: ", length(unique(fifa18$Team)), "\n")
# cat(paste(fifa18$Team, collapse = "\n"))
unique(fifa18$Team)
```

```
# which Team in which Match had the maximum ball
fifa18 %>%
  arrange(desc(Ball.Possession..)) %>%
  mutate(Team = reorder(Team, Ball.Possession..))
head(1) %>%
  select(Date:Ball.Possession..)
```

```
# which match had the closest fight in terms of Ball Possession % ?
fifa18 %>%
  arrange(abs(bpp_diff)) %>%
  head(1) %>%
  select(Team, Ball.Possession..)
```

“fifa18@bpp2” : 100 – 볼 점유율  
“fifa18@bpp\_diff” : 2 x 볼 점유율 - 100

```
install.packages("tidyverse")
library(tidyverse)
options(repr.plot.width=8, repr.plot.height=4)
mypink = "#f442b3"
myblue = "#42d4f4"

setwd("C:\\Users\\EXEMPC\\Desktop\\EduData\\Dataset")
fifa18 <- read.csv("FIFA.CSV")

fifa18$bpp2 <- 100 - fifa18$Ball.Possession..
fifa18$bpp_diff <- fifa18$Ball.Possession.. - fifa18$bpp2

cat("Total Teams: ", length(unique(fifa18$Team)), "\n")
# cat(paste(fifa18$Team, collapse = "\n"))
unique(fifa18$Team)

# which Team in which Match had the maximum ball possession?
fifa18 %>%
  arrange(desc(Ball.Possession..)) %>%
  mutate(Team = reorder(Team, Ball.Possession..))
  head(1) %>%
  select(Date:Ball.Possession..)

# which match had the closest fight in terms of
fifa18 %>%
  arrange(abs(bpp_diff)) %>%
  head(1) %>%
  select(Team, Ball.Possession..)
```

cat() : 콘솔창에 출력하는 함수  
unique() : 중복을 제외하고 출력

```
install.packages("tidyverse")
library(tidyverse)
options(repr.plot.width=8, repr.plot.height=4)
mypink = "#f442b3"
myblue = "#42d4f4"

setwd("C:\\Users\\EXEMPC\\Desktop\\EduData\\Data")
fifa18 <- read.csv("FIFA.CSV")

fifa18$bpp2 <- 100 - fifa18$Ball.Possession..
fifa18$bpp_diff <- fifa18$Ball.Possession.. - fi

cat("Total Teams: ", length(unique(fifa18$Team)),
# cat(paste(fifa18$Team, collapse = "\n"))
unique(fifa18$Team))
```

```
# which Team in which Match had the maximum ball possession?
fifa18 %>%
  arrange(desc(Ball.Possession..)) %>%
  mutate(Team = reorder(Team, Ball.Possession..)) %>%
  head(1) %>%
  select(Date:Ball.Possession..)
```

```
# which match had the closest fight in terms of Ball Possession % ?
fifa18 %>%
  arrange(abs(bpp_diff)) %>%
  head(1) %>%
  select(Team, Ball.Possession..)
```

%>% : 파이프 연산자

여러 함수를 동시에 사용가능

arrange() : 오름차순 or 내림차순 정렬

mutate() : 파생변수를 추가

head() : 1~6행까지 출력

select() : 특정 열만 출력

```
install.packages("tidyverse")
library(tidyverse)
options(repr.plot.width=8, repr.plot.height=4)
mypink = "#f442b3"
myblue = "#42d4f4"

setwd("C:\\Users\\EXEMPC\\Desktop\\EduData\\Data")
fifa18 <- read.csv("FIFA.CSV")

fifa18$bpp2 <- 100 - fifa18$Ball.Possession..
fifa18$bpp_diff <- fifa18$Ball.Possession.. - fi

cat("Total Teams: ", length(unique(fifa18$Team)),
# cat(paste(fifa18$Team, collapse = "\n"))
unique(fifa18$Team))

# which Team in which Match had the maximum ball
fifa18 %>%
  arrange(desc(Ball.Possession..)) %>%
  mutate(Team = reorder(Team, Ball.Possession..)) %>%
  head(1) %>%
  select(Date:Ball.Possession..)

# which match had the closest fight in terms of Ball Possession % ?
fifa18 %>%
  arrange(abs(bpp_diff)) %>%
  head(1) %>%
  select(Team, Ball.Possession..)
```

%>% : 파이프 연산자

여러 함수를 동시에 사용가능

arrange() : 오름차순 or 내림차순 정렬

select() : 특정 열만 출력

```
# Fouls
fifa18 %>%
  group_by(Team) %>%
  summarise(`Total Fouls Committed` = sum(Fouls.Committed)) %>%
  arrange(desc(`Total Fouls Committed`)) %>%
  mutate(Team = reorder(Team, `Total Fouls Committed`)) %>%
  head(10) %>%
  ggplot() + geom_bar(aes(Team, `Total Fouls Committed`), stat = "identity", fill = mypink) +
  labs(title = "More Fouls") +
  coord_flip() -> p1
```

```
fifa18 %>%
  group_by(Team) %>%
  summarise(`Total Fouls Committed` = sum(Fouls.Committed)) %>%
  arrange(desc(`Total Fouls Committed`)) %>%
  mutate(Team = reorder(Team, `Total Fouls Committed`)) %>%
  tail(10) %>%
  ggplot() + geom_bar(aes(Team, `Total Fouls Committed`), stat = "identity", fill = mypink) +
  labs(title = "Less Fouls") +
  coord_flip() -> p2
```

```
cowplot::plot_grid(p1, p2)
```

group\_by() : 특정 열 별로 데이터 분할

summarise() : 요약 통계량 출력

arrange() : 오름차순 or 내림차순 정렬

mutate() : 파생 변수 출력

geom\_bar() : 막대그래프 출력

stat = "identity" -> 데이터 프레임의 값을 그대로 사용하여  
그래프 출력

fill = 막대 그래프 색상 설정

labs() : 제목 설정 , coord\_flip() : x축, y축의 변경

```
# Fouls
fifa18 %>%
  group_by(Team) %>%
  summarise(`Total Fouls Committed` = sum(Fouls.Committed)) %>%
  arrange(desc(`Total Fouls Committed`)) %>%
  mutate(Team = reorder(Team, `Total Fouls Committed`)) %>%
  head(10) %>%
  ggplot() + geom_bar(aes(Team, `Total Fouls Committed`), stat = "identity", fill = mypink) +
  labs(title = "More Fouls") +
  coord_flip() -> p1
```

```
fifa18 %>%
  group_by(Team) %>%
  summarise(`Total Fouls Committed` = sum(Fouls.Committed)) %>%
  arrange(desc(`Total Fouls Committed`)) %>%
  mutate(Team = reorder(Team, -`Total Fouls Committed`)) %>%
  tail(10) %>%
  ggplot() + geom_bar(aes(Team, `Total Fouls Committed`), stat = "identity", fill = myblue) +
  labs(title = "Less Fouls") +
  coord_flip() -> p2
```

```
cowplot::plot_grid(p1,p2)
```

tail() : 뒤쪽 행 출력

p1 -> More Fouls : 파울을 많이 한 상위 10팀

p2 -> Less Fouls : 파울을 적게 한 하위 10팀

```
# Fouls
fifa18 %>%
  group_by(Team) %>%
  summarise(`Total Fouls Committed` = sum(Fouls.Committed)) %>%
  arrange(desc(`Total Fouls Committed`)) %>%
  mutate(Team = reorder(Team, `Total Fouls Committed`)) %>%
  head(10) %>%
  ggplot() + geom_bar(aes(Team, `Total Fouls Committed`), stat = "identity", fill = mypink) +
  labs(title = "More Fouls") +
  coord_flip() -> p1

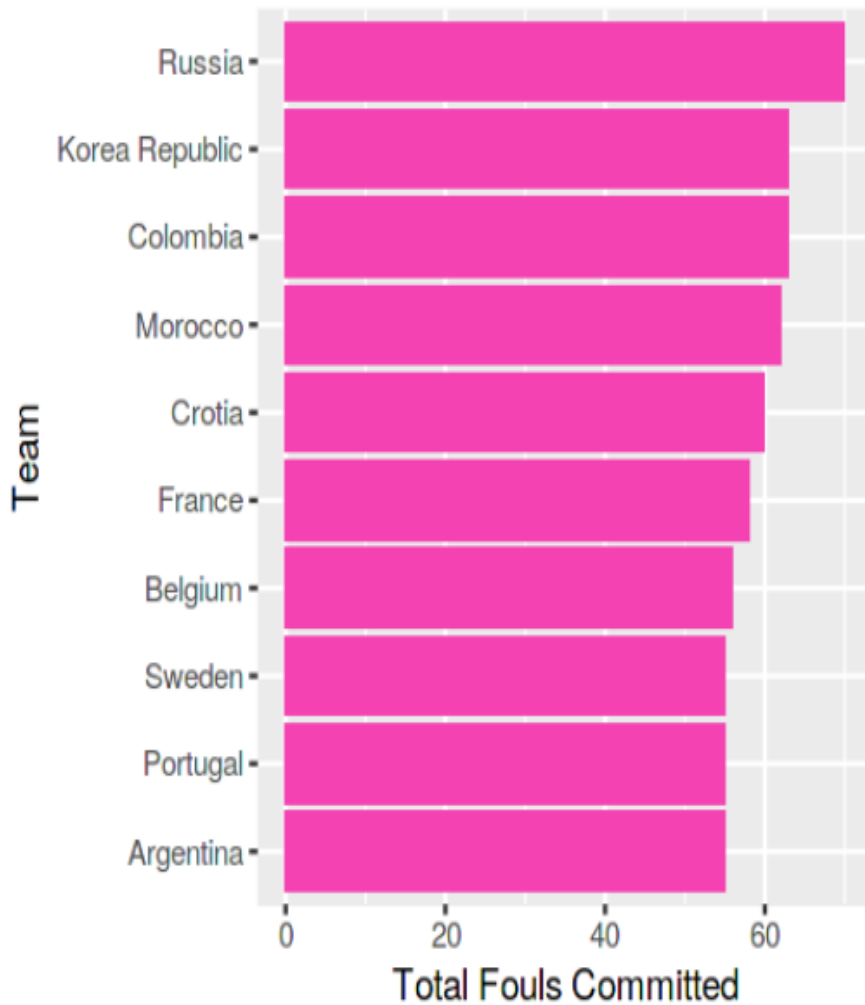
fifa18 %>%
  group_by(Team) %>%
  summarise(`Total Fouls Committed` = sum(Fouls.Committed)) %>%
  arrange(desc(`Total Fouls Committed`)) %>%
  mutate(Team = reorder(Team, -`Total Fouls Committed`)) %>%
  tail(10) %>%
  ggplot() + geom_bar(aes(Team, `Total Fouls Committed`), stat = "identity", fill = myblue) +
  labs(title = "Less Fouls") +
  coord_flip() -> p2
```

```
cowplot::plot_grid(p1,p2)
```

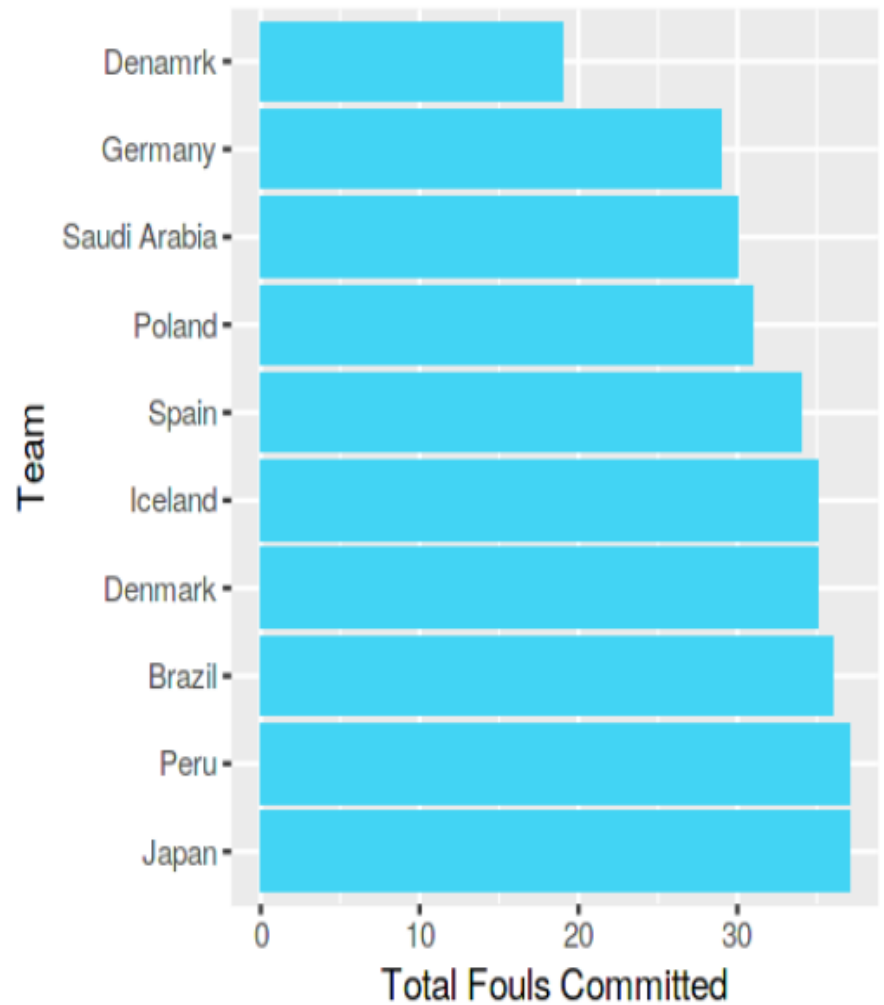
“cowplot”: 'ggplot 2'패키지에 대한 몇가지 유용한  
확장 및 수정. 특히, 여러 개의 플롯을 하나로 쉽게  
결합하는 패키지

plot\_grid() : 여러 개의 플롯을 그리드로 정리한다.

### More Fouls



### Less Fouls



```
# Just looking for Correlation!
options(repr.plot.width=10, repr.plot.height=4)
install.packages("corrplot")
library(corrplot)
```

```
fifa18 %>%
  select_if(is.numeric) %>%
  drop_na() %>%
  cor() %>%
  corrplot(type = "upper", m
```

```
# Relationship between Goals
```

```
fifa18 %>%
  group_by(Team) %>%
  summarise(`Total Passes` =
            `Total Goals` = )
  ggplot(aes(`Total Passes`,
            geom_smooth()
```

```
# Goals ~ Passes Relationship
options(repr.plot.width=10, r
fifa18 %>%
```

```
  ggplot(aes(Goal.Scored, Passes)) + geom_jitter(alpha = 1/4) +
  facet_wrap(~ Team, ncol = 8) +
  geom_smooth(method="lm", se = F)
```

```
# Distance Covered vs Passes
```

```
options(repr.plot.width=10, repr.plot.height=4)
```

```
fifa18 %>%
  ggplot(aes(Distance.Covered..Kms., Passes)) + geom_jitter(alpha = 1/4) +
  stat_smooth()
```

“repr.plot.width=10, repr.plot.height=4”

그래프의 폭을 10, 높이를 4로 설정

“corrplot” : 상관행렬, 신뢰구간의 그래프를 제공하는  
패키지

```
# Just looking for Correlation!
options(repr.plot.width=10, repr.plot.height=4)
install.packages("corrplot")
library(corrplot)
```

```
fifa18 %>%
  select_if(is.numeric) %>%
  drop_na() %>%
  cor() %>%
  corrplot(type = "upper", method = "square", tl.cex = 0.5)
```

```
# Relationship between Goals and Passes
```

```
fifa18 %>%
  group_by(Team) %>%
  summarise(`Total Passes` = sum(Passes),
            `Total Goals` = sum(Goals))
ggplot(aes(`Total Passes`, `Total Goals`)) +
  geom_smooth()
```

```
# Goals ~ Passes Relationship by Team
options(repr.plot.width=10, repr.plot.height=4)
fifa18 %>%
```

```
  ggplot(aes(Goals.Scored, Passes)) +
  facet_wrap(~ Team, ncol = 8) +
  geom_smooth(method="lm", se = F)
```

```
# Distance Covered vs Passes
```

```
options(repr.plot.width=10, repr.plot.height=4)
```

```
fifa18 %>%
  ggplot(aes(Distance.Covered..Kms., Passes)) + geom_jitter(alpha = 1/4) +
  stat_smooth()
```

select\_if(): 괄호의 조건이 TRUE값인 열들만 선택  
drop\_na() : Missing Value가 포함된 행을 제외  
cor() : x와 y의 분산, 공분산 또는 상관관계를 계산  
corrplot() : 상관행렬, 신뢰구간의 그래프 출력

```
# Just looking for correlation!
options(repr.plot.width=10, repr.plot.height=4)
install.packages("corrplot")
library(corrplot)
```

```
fifa18 %>%
  select_if(is.numeric) %>%
  drop_na() %>%
  cor() %>%
  corrplot(type = "upper", method = "square", tl.cex = 0.5)
```

# Relationship between Goals and Passes

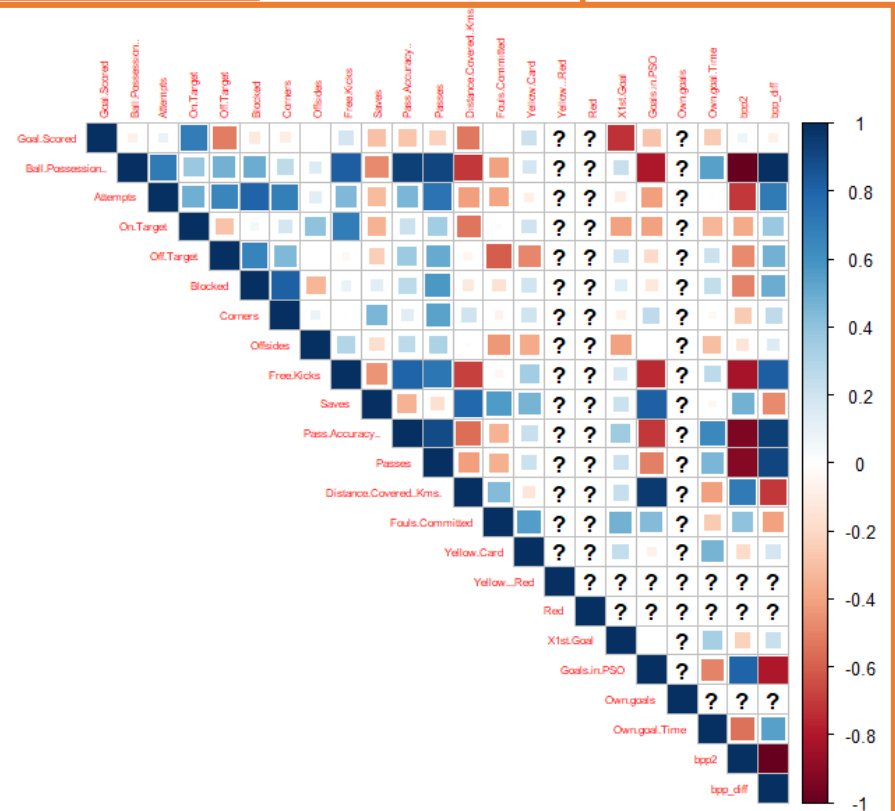
```
fifa18 %>%
  group_by(Team) %>%
  summarise(`Total Passes` = sum(
    `Total Goals` = sum(G
  ggplot(aes(`Total Passes`, `Total Goals`))
  geom_smooth()
```

# Goals ~ Passes Relationship by Team

```
options(repr.plot.width=10, repr.plot.height=4)
fifa18 %>%
  ggplot(aes(Goal.Scored, Passes))
  facet_wrap(~ Team, ncol = 8) +
  geom_smooth(method="lm", se = F)
```

# Distance Covered vs Passes

```
options(repr.plot.width=10, repr.plot.height=4)
fifa18 %>%
  ggplot(aes(Distance.Covered..Kms., Passes))
  stat_smooth()
```



```
# Just looking for Correlation!
options(repr.plot.width=10, repr.plot.height=4)
install.packages("corrplot")
library(corrplot)

fifa18 %>%
  select_if(is.numeric) %>%
  drop_na() %>%
  cor() %>%
  corrplot(type = "upper", method = "square", tl.cex = 0.5)
```

```
# Relationship between Goals and Passes
```

```
fifa18 %>%
  group_by(Team) %>%
  summarise(`Total Passes` = sum(Passes),
            `Total Goals` = sum(Goal.Scored)) %>%
  ggplot(aes(`Total Passes`, `Total Goals`)) + geom_jitter(alpha = 1/4) +
  geom_smooth()
```

group\_by() : 특정 열로 데이터 분할

geom\_jitter() : 각 점의 위치에 랜덤 변동을 추가한 그래프

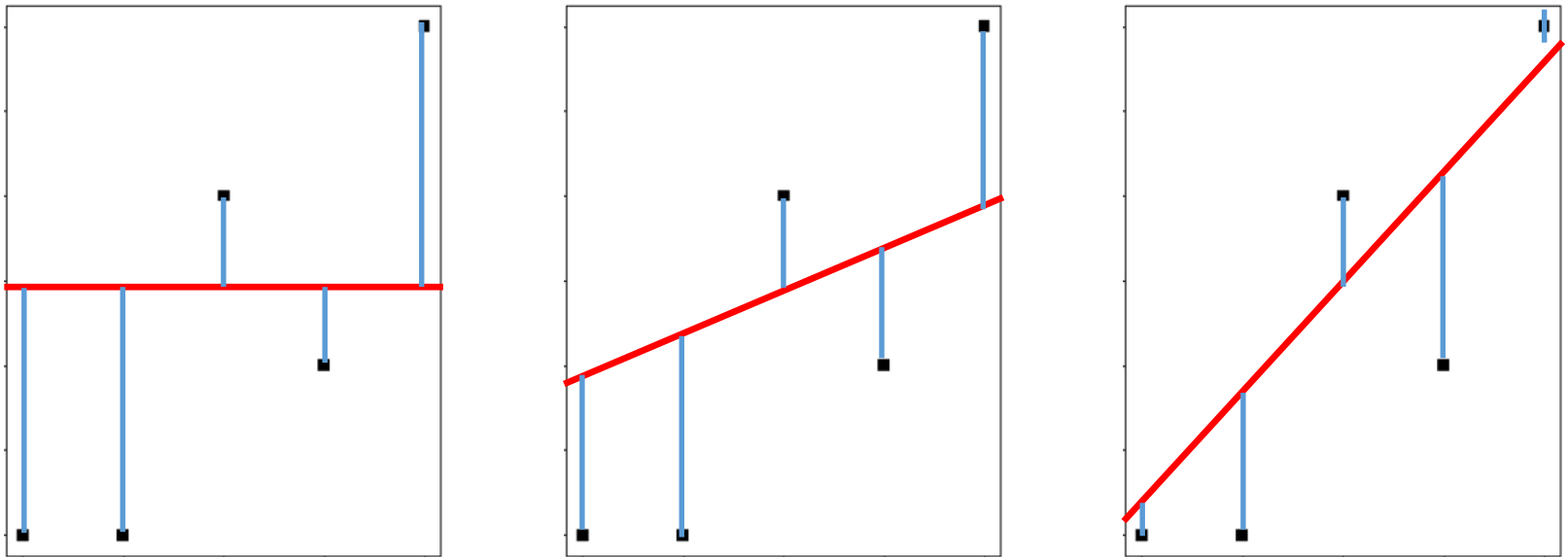
geom\_smooth() : 국소다항회귀 기반 회귀선으로 변화 추이를 도식화

\*회귀선 : 데이터를 가장 잘 표현하는 선

가장 잘 표현하는 ? : 실제값과 예측값 사이의 오차가 최소화되었다.

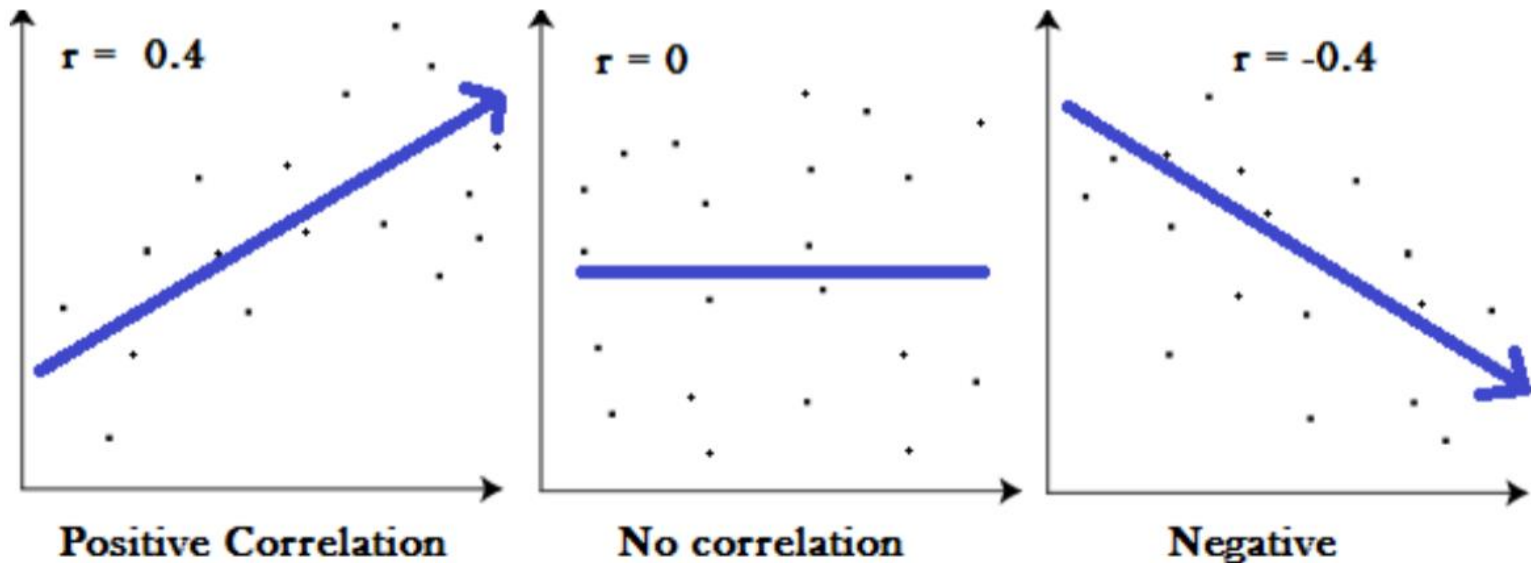
## (참고) 회귀선

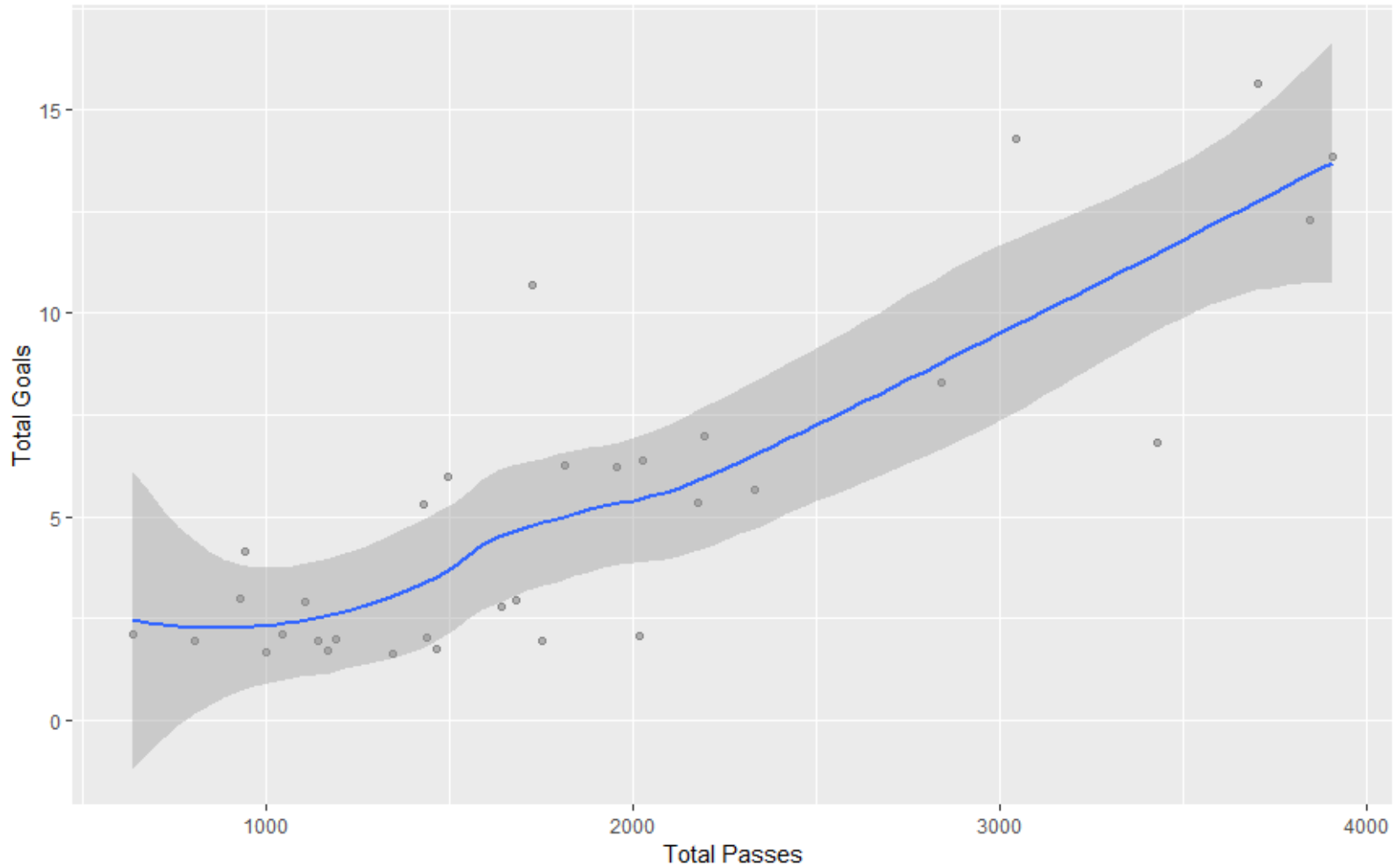
- 실제  $y$  값:  $y = \beta_0 + \beta_1 x + \varepsilon$
- 예측된  $y$  값:  $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$
- 목적 :  $\hat{y}$ 과  $y$ 의 **차이를 최소화**(정확히는  $y$ 축 거리의 제곱의 합)
- 회귀계수  $\beta_1$ 은 설명변수  $x$ 가 한 단위(1) 증가할 때 종속변수가 얼마나 변화하는지 **상관관계(correlation)**를 보여주는 지표



## (참고) 상관계수

- ✓ 두 변수간의 연관성에 대한 측도
- ✓ 표본상관계수( $r$ )
  - $-1 \leq R \leq 1$
  - $|r|$  값이 1에 가까울수록 강한 상관관계
  - $|r|$  값이 0에 가까울수록 약한 상관관계



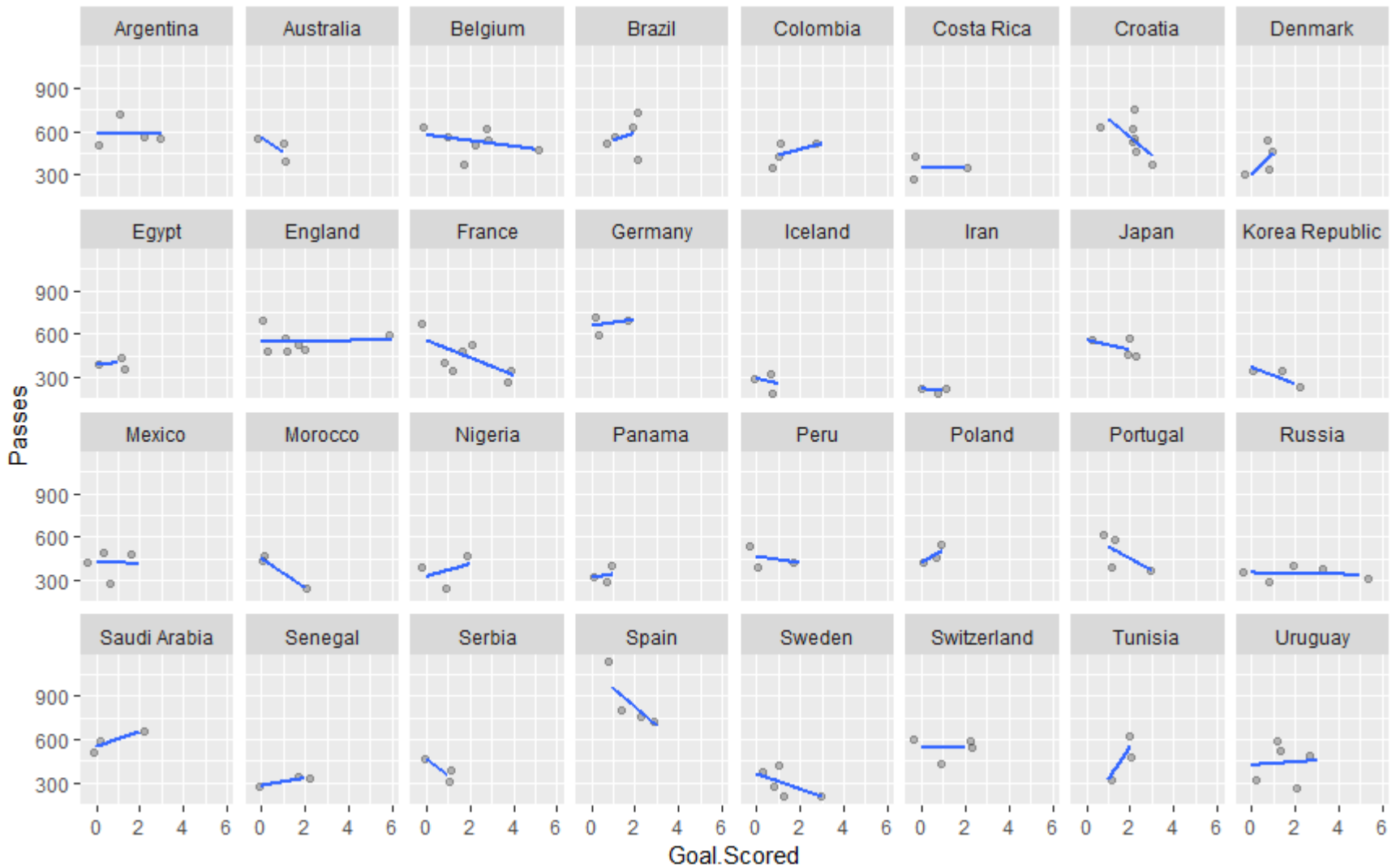


```
# Just looking for Correlation!
options(repr.plot.width=10, repr.plot.height=4)
install.packages("corrplot")
library(corrplot)
```

facet\_wrap(~ Team, ncol = 8) : 가로 또는 세로 방향의 변수를 사용  
하는 대신 면을 8개의 행으로 감싸서 배치  
stat\_smooth : geom\_smooth와 같은 인수를 사용, 비표준 geom으로  
결과를 표시하지 않으려면 geom\_smooth 사용

```
# Goals ~ Passes Relationship by Team
options(repr.plot.width=10, repr.plot.height=8)
fifa18 %>%
  ggplot(aes(Goal.Scored, Passes)) + geom_jitter(alpha = 1/4) +
  facet_wrap(~ Team, ncol = 8) +
  geom_smooth(method="lm", se = F)
```

```
# Distance Covered vs Passes
options(repr.plot.width=10, repr.plot.height=4)
fifa18 %>%
  ggplot(aes(Distance.Covered..Kms., Passes)) + geom_jitter(alpha = 1/4) +
  stat_smooth()
```



```
# Just looking for correlation!
options(repr.plot.width=10, repr.plot.height=4)
install.packages("corrplot")
library(corrplot)

fifa18 %>%
  select_if(is.numeric) %>%
  drop_na() %>%
  cor() %>%
  corrplot(type = "upper", method = "square", tl.cex = 0.5)
```

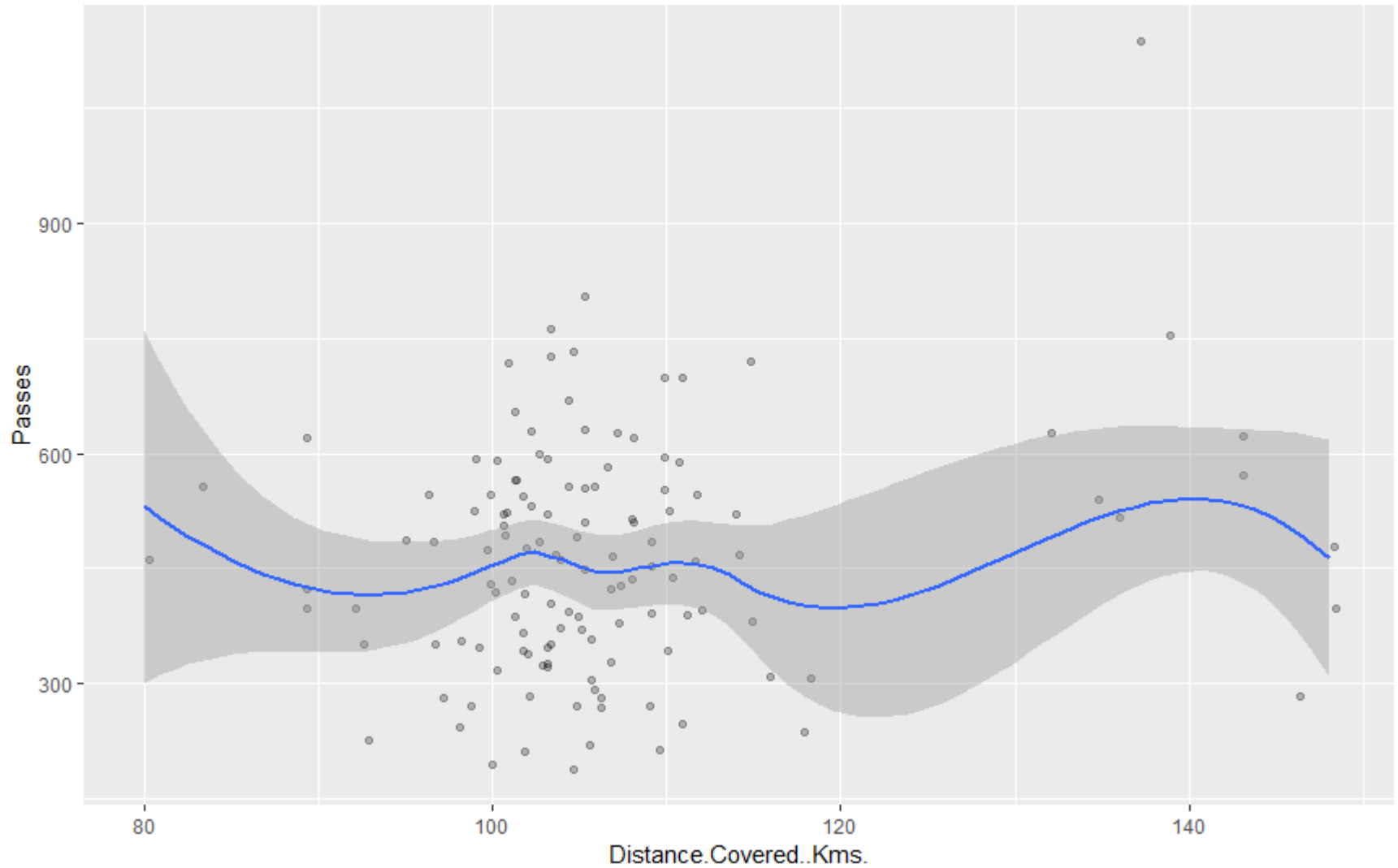
```
# Relationship
```

```
fifa18 %>%
  group_by(Team)
  summarise(`1` = sum(Passes), `2` = sum(Distance.Covered..Kms.))
  ggplot(aes(`1`, `2`))
  geom_smooth()
```

```
# Goals ~ Passes
options(repr.plot.width=10, repr.plot.height=4)
fifa18 %>%
  ggplot(aes(Passes, Goals))
  facet_wrap(~League)
  geom_smooth()
```

x축을 팀원들의 총 이동거리, y축을 팀 패스 횟수로 맵핑  
geom\_jitter()로 랜덤 변동을 추가

```
# Distance Covered vs Passes
options(repr.plot.width=10, repr.plot.height=4)
fifa18 %>%
  ggplot(aes(Distance.Covered..Kms., Passes)) + geom_jitter(alpha = 1/4) +
  stat_smooth()
```



```
# Ball Possesstion % vs Goals scp
options(repr.plot.width=10, repr.plot.height=4)
fifa18 %>%
  ggplot(aes(Ball.Possession..,Goal.Scored)) + geom_jitter(alpha = 1/4) +
  stat_smooth()
```

```
# More Attempts vs Less Attempts
fifa18 %>%
  group_by(Team) %>%
  summarise(`Total Fouls Committed` =
  arrange(desc(`Total Fouls Committed`
  mutate(Team = reorder(Team, `Total
  head(10) %>%
  ggplot() + geom_bar(aes(Team, `Total
  labs(title = "More Attempts",
        y = "Attempts count") +
  coord_flip() -> p1
```

x축을 볼점유율, y축을 팀 득점수로 맵핑  
geom\_jitter()로 랜덤 변동을 추가

```
fifa18 %>%
  group_by(Team) %>%
  summarise(`Total Fouls Committed` = sum(Attempts)) %>%
  arrange(desc(`Total Fouls Committed`)) %>%
  mutate(Team = reorder(Team, -`Total Fouls Committed`)) %>%
  tail(10) %>%
  ggplot() + geom_bar(aes(Team, `Total Fouls Committed`), stat = "identity", fill = myblue) +
  labs(title = "Less Attempts",
        y = "Attempts count") +
  coord_flip() -> p2

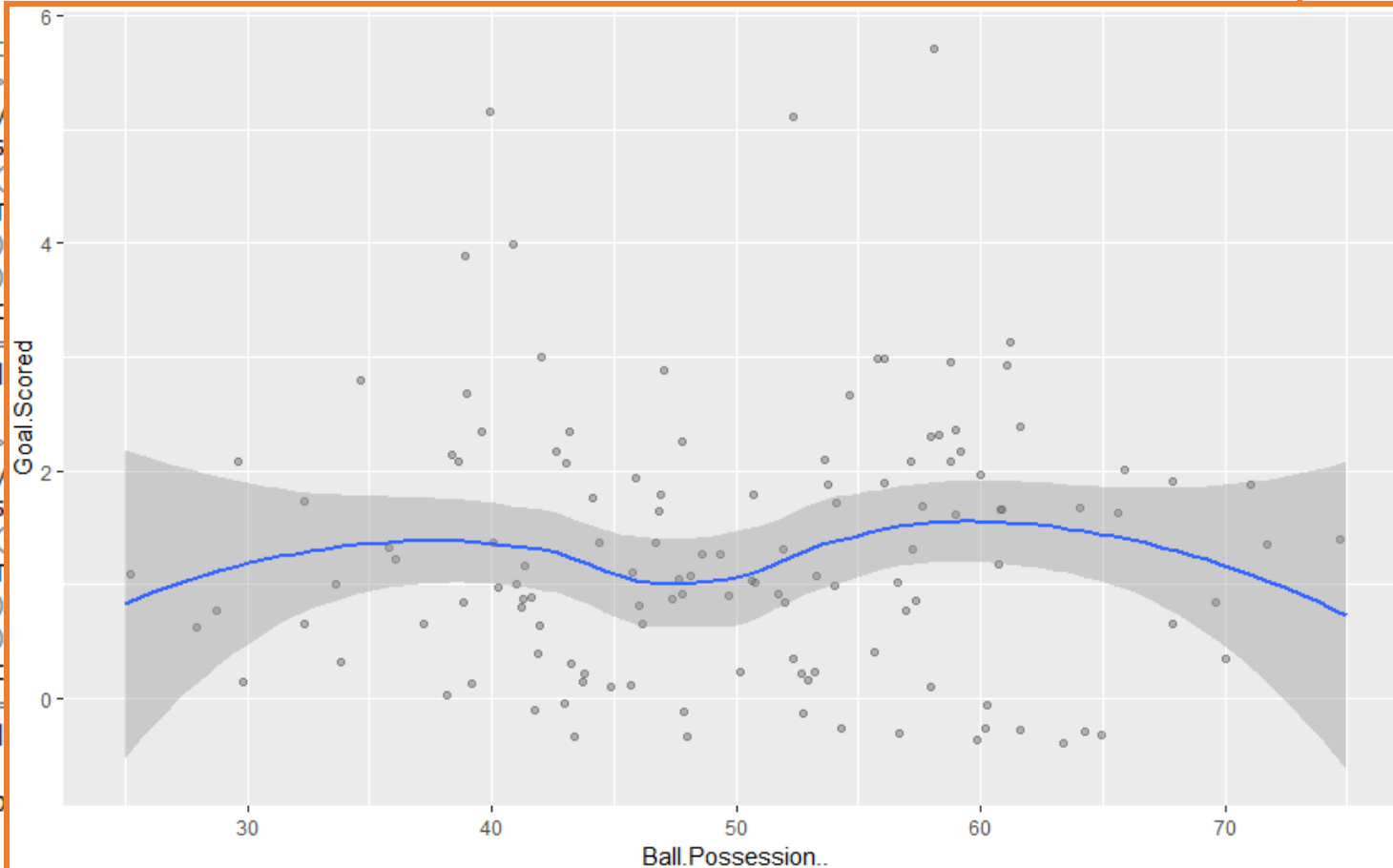
cowplot::plot_grid(p1,p2)
```

```
# Ball Possession % vs Goals scp
options(repr.plot.width=10, repr.plot.height=4)
fifa18 %>%
  ggplot(aes(Ball.Possession.., Goal.Scored)) + geom_jitter(alpha = 1/4) +
  stat_smooth()
```

```
# More Att
fifa18 %>%
  group_by
  summaris
  arrange(
  mutate(T
  head(10)
  ggplot()
  labs(tit
  y =
  coord_fl

fifa18 %>%
  group_by
  summaris
  arrange(
  mutate(T
  tail(10)
  ggplot()
  labs(tit
  y =
  coord_fl

cowplot::p
```



```
# Ball Possession % vs Goals Scp
options(repr.plot.width=10, repr.plot.height=4)
fifa18 %>%
  ggplot(aes(Ball.Possession..,Goal.Scored)) + geom_jitter(alpha = 1/4) +
  stat_smooth()
```

```
# More Attempts vs Less Attempts
fifa18 %>%
  group_by(Team) %>%
  summarise(`Total Fouls Committed` = sum(Attempts)) %>%
  arrange(desc(`Total Fouls Committed`)) %>%
  mutate(Team = reorder(Team, `Total Fouls Committed`)) %>%
  head(10) %>%
  ggplot() + geom_bar(aes(Team, `Total Fouls Committed`), stat = "identity", fill = mypink) +
  labs(title = "More Attempts",
       y = "Attempts count") +
  coord_flip() -> p1
```

group\_by() : Team별로 데이터 분할

summarise() : 득점시도횟수의 합계로 데이터 요약

arrange() : 내림차순으로 정렬

mutate() : Team이라는 파생 변수 생성

coord\_flip() : x축과 y축 변경

```
# Ball Possession % vs Goals Scp
options(repr.plot.width=10, repr.plot.height=4)
fifa18 %>%
  ggplot(aes(Ball.Possession.., Goal.Scored)) + geom_jitter(alpha = 1/4) +
```

group\_by() : Team별로 데이터 분할

summarise() : 득점시도횟수의 합계로 데이터 요약

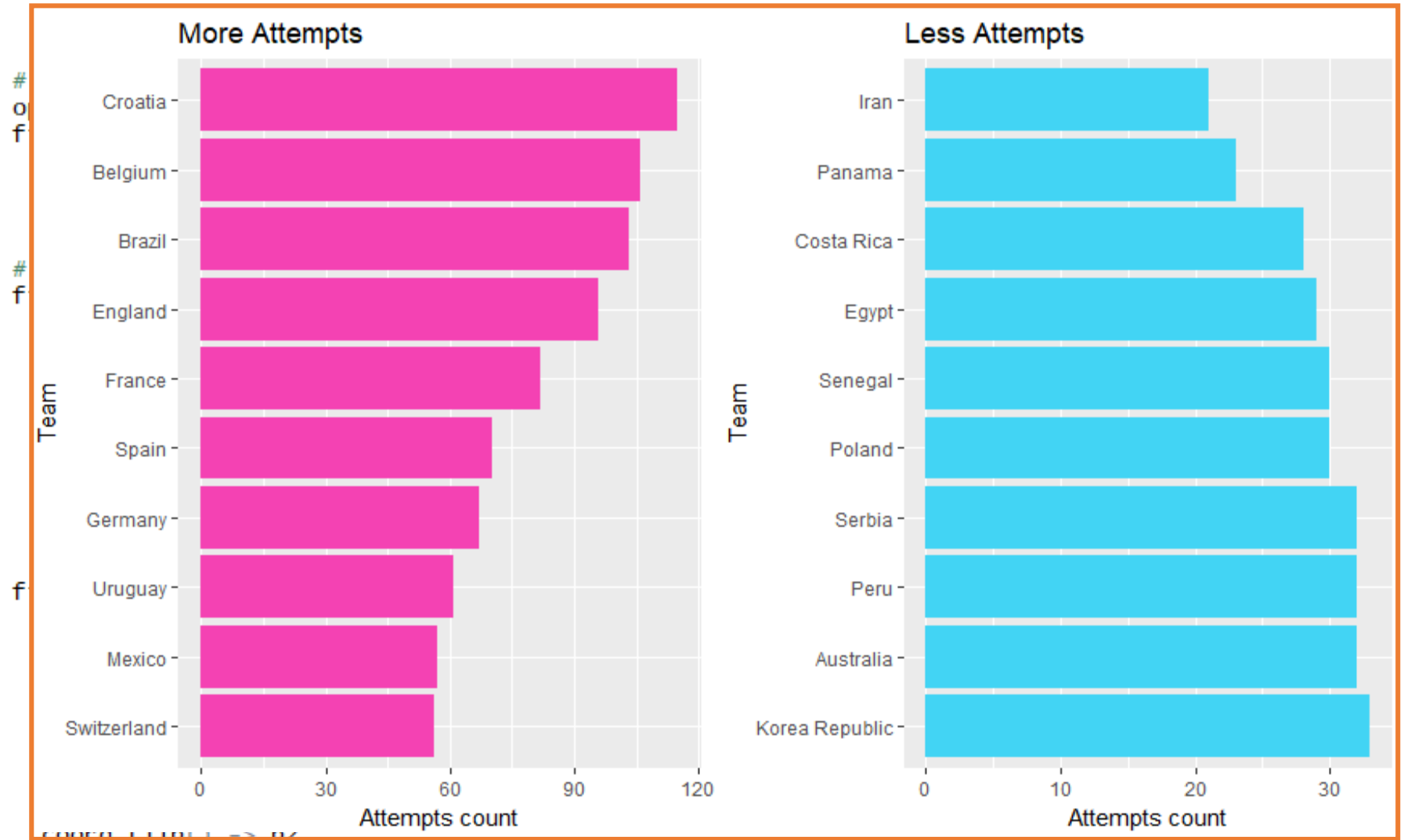
arrange() : 내림차순으로 정렬

mutate() : Team이라는 파생 변수 생성

coord\_flip() : X축과 Y축 변경

```
fifa18 %>%
  group_by(Team) %>%
  summarise(`Total Fouls Committed` = sum(Attempts)) %>%
  arrange(desc(`Total Fouls Committed`)) %>%
  mutate(Team = reorder(Team, -`Total Fouls Committed`)) %>%
  tail(10) %>%
  ggplot() + geom_bar(aes(Team, `Total Fouls Committed`), stat = "identity", fill = myblue) +
  labs(title = "Less Attempts",
        y = "Attempts count") +
  coord_flip() -> p2
```

```
cowplot::plot_grid(p1,p2)
```



`cowplot::plot_grid(p1,p2)`

## Kaggle, Titanic Data 머신 러닝 분석 예제



```
## Kaggle Randomforest Analysis
```

```
# install Packages for Analysis
install.packages("ggplot2")
install.packages("randomForest")
library(ggplot2)
library(randomForest)
```

```
# train / test dataset load from Titanic Data
set.seed(1)
train <- read.csv("train.csv")
test <- read.csv("test.csv")
```

```
# Data Preprocessing
extractFeatures
```

```
  features <- c("Pclass",
               "Age",
               "Sex",
               "Parch",
               "Sibsp",
               "Fare",
               "Embarked")
```

```
  fea <- data[,features]
  fea$Age[is.na(fea$Age)] <- -1
  fea$Fare[is.na(fea$Fare)] <- median(fea$Fare, na.rm=TRUE)
  fea$Embarked[fea$Embarked==""] = "S"
  fea$Sex <- as.factor(fea$Sex)
  fea$Embarked <- as.factor(fea$Embarked)
  return(fea)
```

```
}
```

ggplot2 : 시각화 패키지  
randomForest : 분류, 회귀분석을 위한 Breiman의 알고리즘

```
## Kaggle Randomforest Analysis
```

```
# install Packages for Analysis
```

```
install.packages("ggplot2")
```

```
install.packages("randomForest")
```

```
library(ggplot2)
```

```
library(randomForest)
```

```
# train / test dataset load from Titanic Data
```

```
set.seed(1)
```

```
train <- read.csv("train.csv", stringsAsFactors=FALSE)
```

```
test  <- read.csv("test.csv", stringsAsFactors=FALSE)
```

```
# Data Preprocessing
```

```
extractFeatures <- function(data) {
```

```
  features <- c("Pclass",  
               "Age",  
               "Sex",  
               "Parch",  
               "Sibsp",  
               "Fare",  
               "Embarked")
```

```
  fea <- data[,features]
```

```
  fea$Age[is.na(fea$Age)] <- -1
```

```
  fea$Fare[is.na(fea$Fare)] <- median(fea$Fare, na.rm=TRUE)
```

```
  fea$Embarked[fea$Embarked==""] = "S"
```

```
  fea$Sex      <- as.factor(fea$Sex)
```

```
  fea$Embarked <- as.factor(fea$Embarked)
```

```
  return(fea)
```

```
}
```

set.seed(1) : 난수의 초기값 설정  
train : train dataset 할당  
test : test dataset 할당

```
## Kaggle Randomforest Analysis

# install Packages for Analysis
install.packages("ggplot2")
install.packages("randomForest")
library(ggplot2)
library(randomForest)

# train / test dataset load from Titanic Data
set.seed(1)
train <- read.csv("train.csv", stringsAsFactors=FALSE)
test  <- read.csv("test.csv", stringsAsFactors=FALSE)
```

```
# Data Preprocessing
extractFeatures <- function(data) {
  features <- c("Pclass",
               "Age",
               "Sex",
               "Parch",
               "Sibsp",
               "Fare",
               "Embarked")

  fea <- data[,features]
  fea$Age[is.na(fea$Age)] <- -1
  fea$Fare[is.na(fea$Fare)] <- median(fea$Fare, na.rm=TRUE)
  fea$Embarked[fea$Embarked==""] = "S"
  fea$Sex      <- as.factor(fea$Sex)
  fea$Embarked <- as.factor(fea$Embarked)
  return(fea)
}
```

function() : 사용자정의함수  
변수 설명

Pclass : 승객클래스-티켓클래스  
1st=Upper, 2nd=Middle,  
3rd=Lower

Age : 나이, 1보다 작으면 분수

Sex : 성별

Parch : 가족관계 정의

Parent = 어머니, 아버지

Child = 딸, 아들, 의붓딸, 의붓아들

Sibsp: 가족관계 정의

Sibling = 형제자매

Spouse = 배우자

Fare : 요금

Embarked : 승선항

C=체르부르크, Q=퀸즈 타운,

S=사우샘프턴

```
# Perform Randomforest  
rf <- randomForest(extractFeatures(train), as.factor(train$Survived), ntree=100, importance=TRUE)
```

randomForest() : 분류와 회귀 알고리즘을 수행하는 함수

extractFeatures(train) : 예측 변수 행렬 (x)

as.factor(train\$Survived) : 응답 벡터 (y)

ntree = 100 : 나무의 수(모든 입력 행이 최소 100번씩 예측됨)

importance : 예측 변수의 중요도 평가 여부(logical)

```
p <- ggplot(featureImportance, aes(x=reorder(Feature, Importance), y=Importance)) +  
  geom_bar(stat="identity", fill="#53cfff") +  
  coord_flip() +  
  theme_light(base_size=20) +  
  xlab("") +  
  ylab("Importance") +  
  ggtitle("Random Forest Feature Importance\n") +  
  theme(plot.title=element_text(size=18))
```

```
# save the png file  
ggsave("2_feature_importance.png", p)
```

(참고) 랜덤포레스트(RandomForest)란?

- ✓ 다수의 의사결정나무를 사용하여 예측하는 알고리즘, 예를 들어 의사결정나무 10개 중 6개가 Yes, 4개가 No라고 예측하면 Yes라고 예측한다.



랜덤포레스트의 장점

- ✓ 기존 하나의 의사결정나무를 사용할 때보다 overfitting(과적합)문제를 피할 수 있다.
- ✓ 변수 중요도를 평가하고 이것을 변수 선택에 사용할 수 있다, 일반적인 선형모델보다 좋은 성능을 보여준다.

```
# Perform Randomforest
rf <- randomForest(extractFeatures(train), as.factor(train$Survived), ntree=100, importance=TRUE)

# Prediction for testset
submission <- data.frame(PassengerId = test$PassengerId)
submission$Survived <- predict(rf, extractFeatures(test))

# Save csv file
write.csv(submission, file = "1_random_forest_r_submission.csv", row.names=FALSE)

# Import test data
imp <- import(test)
fea <- featureNames(imp)

# Visualize feature importance
p <- ggplot(submission, aes(x=PassengerId, y=Survived)) +
  geom_bar(stat="identity", fill="#53cfff") +
  coord_flip() +
  theme_light(base_size=20) +
  xlab("") +
  ylab("Importance") +
  ggtitle("Random Forest Feature Importance\n") +
  theme(plot.title=element_text(size=18))

# save the png file
ggsave("2_feature_importance.png", p)
```

PassengerId(승객 번호) : test데이터의 컬럼  
Survived(생존 여부): rf(랜덤포레스트 수행객체)를 통해 test데이터로 예측수행

```
# Perform Randomforest
rf <- randomForest(extractFeatures(train), as.factor(train$Survived), ntree=100, importance=TRUE)

# Prediction for testset
submission <- data.frame(PassengerId = test$PassengerId)
submission$Survived <- predict(rf, extractFeatures(test))

# Save csv file
write.csv(submission, file = "1_random_forest_r_submission.csv", row.names=FALSE)
```

예측 수행결과를 csv파일로 저장

```
geom_bar(stat = identity, fill = "#3399ff") +
coord_flip() +
theme_light(base_size=20) +
xlab("") +
ylab("Importance") +
ggtitle("Random Forest Feature Importance\n") +
theme(plot.title=element_text(size=18))

# save the png file
ggsave("2_feature_importance.png", p)
```

```
# Perform Randomforest
rf <- randomForest(extractFeatures(train), as.factor(train$Survived), ntree=100, importance=TRUE)

# Prediction for testset
submission <- data.frame(PassengerId = test$PassengerId)
submission$Survived <- predict(rf, extractFeatures(test))

# Save csv file
write.csv(submission, file = "1_random_forest_r_submission.csv", row.names=FALSE)

# Importance measurement
imp <- importance(rf, type=1)
featureImportance <- data.frame(Feature=row.names(imp), Importance=imp[,1])

# visualization
p <- ggplot(featureImportance, aes(x=reorder(Feature, Importance), y=Importance)) +
  geom_bar(stat="identity", fill="#52cfff")

# save the png file
ggsave("2_feature_importance.png", p)
```

importance() : 변수(variables) 중요도 측정 함수  
featureImportance : 중요도 측정 결과를 데이터 프레임을 생성

# Random Forest Feature Importance

imp[,1])

ggplot(): X축: Feature, Y축: Importance로 맵핑  
geom\_bar(): 막대그래프로 표현  
stat="identity" -> y변수가 데이터 셋에 포함될 경우 사용  
coord\_flip(): X축과 Y축 변경

```
featureImportance <- data.frame(Feature=row.names(imp), Importance=imp[,1])
```

```
# visualization
p <- ggplot(featureImportance, aes(x=reorder(Feature, Importance), y=Importance)) +
  geom_bar(stat="identity", fill="#53cfff") +
  coord_flip() +
  theme_light(base_size=20) +
  xlab("") +
  ylab("Importance") +
  ggtitle("Random Forest Feature Importance\n") +
  theme(plot.title=element_text(size=18))
```

```
# save the png file
ggsave("2_feature_importance.png", p)
```

```
# Perform Randomforest
rf <- randomForest(extractFeatures(train), as.factor(train$Survived), ntree=100, importance=TRUE)

# Prediction for testset
submission <- data.frame(PassengerId = test$PassengerId)
submission$Survived <- predict(rf, extractFeatures(test))

# Save csv file
write.csv(submission, file = "1_random_forest_r_submission.csv", row.names=FALSE)

# Importance measurement
imp <- importance(rf, type=1)
featureImportance <- data.frame(Feature=row.names(imp), Importance=imp[,1])

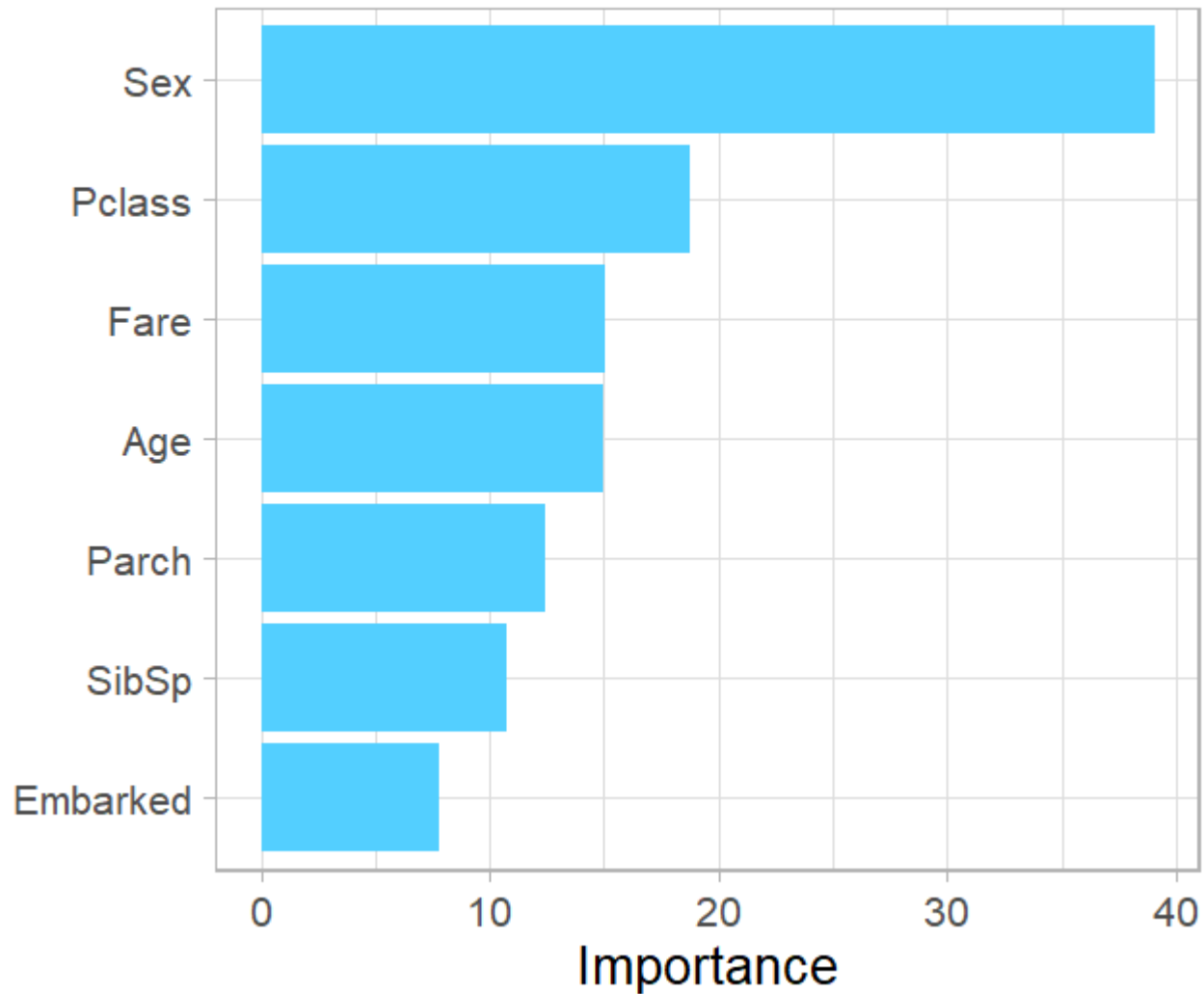
# visualization
```

ggsave() : ggplot 결과를 png 파일로 저장

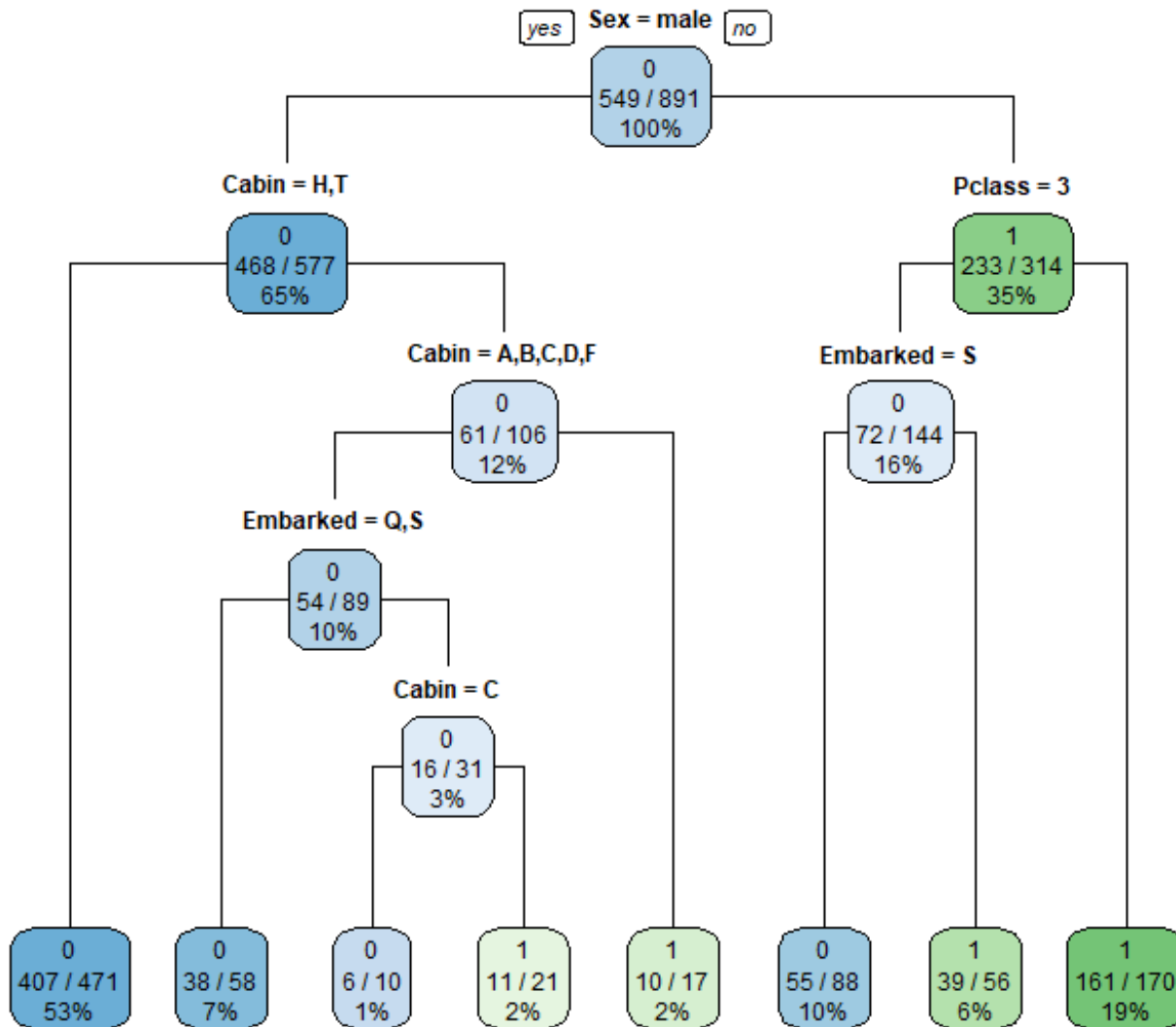
```
xlab("") +
ylab("Importance") +
ggtitle("Random Forest Feature Importance\n") +
theme(plot.title=element_text(size=18))

# save the png file
ggsave("2_feature_importance.png", p)
```

## Random Forest Feature Importance



## (참고) Kaggle 의사결정나무 분석

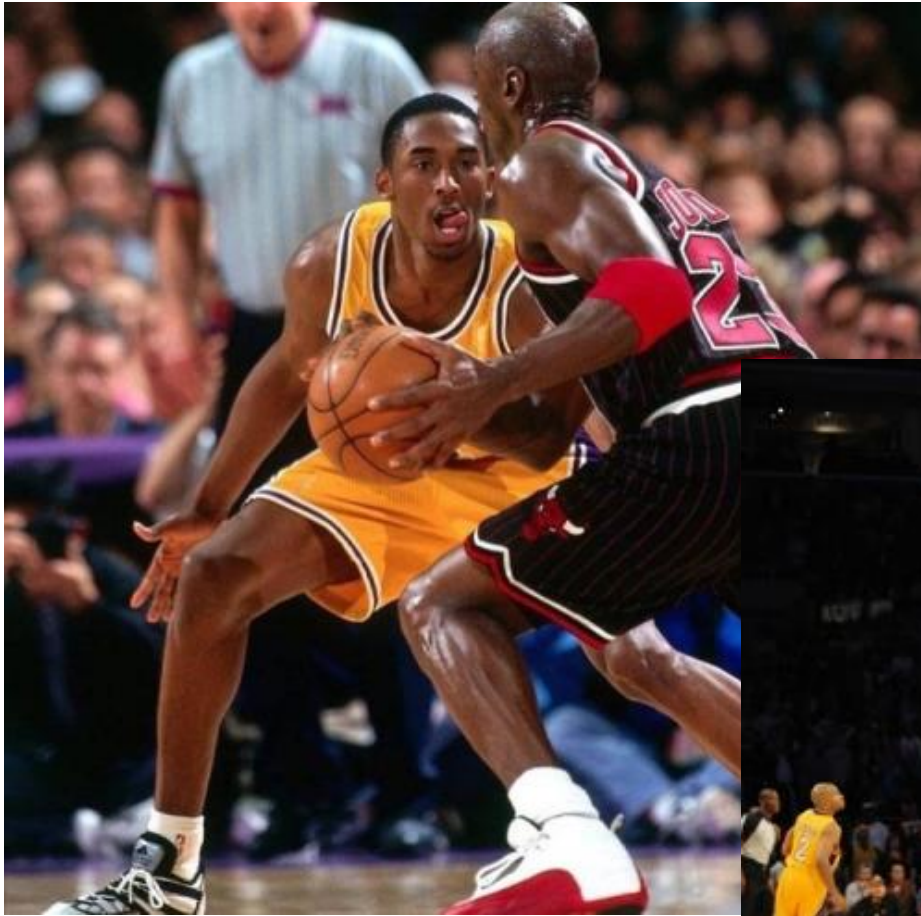


1. 성별이 남자(male) 이면서 객실 번호 (Cabin)가 H,T로 시작 하는 승객의 생존율이 가장 낮았다.

2. 성별이 여자 (female)이면서 Pclass(티켓 등급)가 3 이 아닌 승객은 생존율이 높다.

3. 성별이 여자 (female)이면서 Pclass(티켓 등급)가 3 이고 승선항이 사우샘프턴이면 생존율이 낮다.

## Kaggle, Kobe Bryant Shot Selection 분석 예제



```
## Kaggle Exploring Kobe's Shot  
library(dplyr)  
library(ggplot2)
```

“dplyr” : 데이터 전처리 패키지  
“ggplot2” : 데이터 시각화 패키지

```
data <- read.csv("C:/Users/EXEMPC/Desktop/EduData/Dataset/data.csv/data.csv", stringsAsFactors = FALSE)  
  
train <- data[!is.na(data$shot_made_flag),]  
test <- data[is.na(data$shot_made_flag),]  
  
train$shot_made_flag <- as.factor(train$shot_made_flag)  
  
names(train)  
  
# a plot to see accuracy by feature  
pplot <- function(feats) {  
  feat <- substitute(feats)  
  ggplot(data = train, aes_q(x = feat)) +  
    geom_bar(aes(fill = shot_made_flag), stat = "count", position = "fill") +  
    scale_fill_brewer(palette = "Set1", direction = +1) +  
    ggtitle(paste("accuracy by", feat))  
}  
  
# a plot to see position by feature  
courtplot <- function(feats) {  
  feat <- substitute(feats)  
  train %>%  
    ggplot(aes(x = lon, y = lat)) +  
    geom_point(aes_q(color = feat), alpha = 0.7, size = 3) +  
    ylim(c(33.7, 34.0883)) +  
    scale_color_brewer(palette = "Set1") +  
    theme_void() +  
    ggtitle(paste(feats))  
}
```

```
## Kaggle Exploring Kobe's Shot
library(dplyr)
library(ggplot2)
```

```
data <- read.csv("C:/Users/EXEMPC/Desktop/EduData/Dataset/data.csv/data.csv", stringsAsFactors = FALSE)
```

```
train <- data[!is.na(data$shot_made_flag),]
test <- data[is.na(data$shot_made_flag),]
```

```
train$shot_made_flag
names(train)
```

```
# a plot to see accur
pplot <- function(fe
  feat <- substitute
  ggplot(data = trai
    geom_bar(aes(fi
    scale_fill_brew
    ggtitle(paste("%
```

```
}
```

```
# a plot to see posi
courtplot <- functio
  feat <- substitute
  train %>%
    ggplot(aes(x = 1
    geom_point(aes_c
    ylim(c(33.7, 34.
    scale_color_brew
    theme_void() +
    ggtitle(paste(feat))
}
```

data : 코비 브라이언트가 20년 경력 동안 시도한 모든 필드 골의 위치와 상황을 담고 있다.

분석 목적 : 골이 들어갔는지 예측하는 것

Shot\_made\_flag (슛 성공 여부)

특징 : shot\_made\_flag 변수에 5000개의 결측값(missing value)이 존재

```
## Kaggle Exploring Kobe's Shot
library(dplyr)
library(ggplot2)

data <- read.csv("C:/Users/EXEMPC/Desktop/EduData/Dataset/data.csv/data.csv", stringsAsFactors = FALSE)

train <- data[!is.na(data$shot_made_flag),]
test <- data[is.na(data$shot_made_flag),]

train$shot_made_flag <- as.factor(train$shot_made_flag)

names(train)

# a plot to see accuracy by feature
pplot <- function(feats) {
  feat <- substitute(feats)
  ggplot(data = train, aes_q(x = feat)) +
    geom_bar(aes(fill = shot_made_flag)) +
    scale_fill_brewer(palette = "Set1") +
    ggtitle(paste("accuracy by", feat))
}

# a plot to see position by feature
courtplot <- function(feats) {
  feat <- substitute(feats)
  train %>%
    ggplot(aes(x = lon, y = lat)) +
    geom_point(aes_q(color = feat), alpha = 0.7, size = 3) +
    ylim(c(33.7, 34.0883)) +
    scale_color_brewer(palette = "Set1") +
    theme_void() +
    ggtitle(paste(feats))
}
```

유효한 예측을 위해 shot\_made\_flag(슛 성공 여부) 변수에 결측치가 없는 행을 train set으로 할당  
shot\_made\_flag(슛 성공 여부) 변수에 결측치가 있는 행을 test set으로 할당하여 예측 수행

```
## Kaggle Exploring Kobe's Shot
library(dplyr)
library(ggplot2)

data <- read.csv("C:/Users/EXEMPC/Desktop/EduData/Dataset/data.csv/data.csv", stringsAsFactors = FALSE)

train <- data[!is.na(data$shot_made_flag),]
test <- data[is.na(data$shot_made_flag),]

train$shot_made_flag <- as.factor(train$shot_made_flag)

names(train)

#a plot to see accuracy by feature
pplot <- function(feats) {
  feat <- substitute(feats)
  ggplot(data = train, aes_q(x = feat, y = accuracy)) +
    geom_bar(aes(fill = shot_made_flag)) +
    scale_fill_brewer(palette = "Set1") +
    ggtitle(paste("accuracy by", feat))
}

# a plot to see position by feature
courtplot <- function(feats) {
  feat <- substitute(feats)
  train %>%
    ggplot(aes(x = lon, y = lat)) +
    geom_point(aes_q(color = feat), alpha = 0.7, size = 3) +
    ylim(c(33.7, 34.0883)) +
    scale_color_brewer(palette = "Set1") +
    theme_void() +
    ggtitle(paste(feats))
}
```

shot\_made\_flag 변수를 factor로 변환하여 할당

```
## Kaggle Exploring Kobe's Shot
```

```
library(dplyr)
```

```
library(ggplot2)
```

```
> names(train)
```

```
[1] "action_type"      "combined_shot_type" "game_event_id"      "game_id"            "lat"
[6] "loc_x"            "loc_y"              "lon"                "minutes_remaining"  "period"
[11] "playoffs"         "season"             "seconds_remaining"  "shot_distance"      "shot_made_flag"
[16] "shot_type"        "shot_zone_area"     "shot_zone_basic"    "shot_zone_range"    "team_id"
[21] "team_name"        "game_date"          "matchup"            "opponent"           "shot_id"
```

```
names(train)
```

action\_type : 시도한 슛의 종류, combined\_shot\_type : 슛의 종류를 6가지로 통합,  
 game\_event\_id : 슛 id, game\_id : 경기 id, lat : 위도, loc\_x : 위치 x 좌표,  
 loc\_y : 위치 y 좌표, lon : 경도, minutes\_remaining : 남은 쿼터 시간(분), period : 쿼터,  
 playoffs: 플레이오프 경기 구분(1 or 0), season : 시즌 구분, seconds\_remaining : 남은  
 쿼터 시간(초), shot\_distance : 슛 거리(ft), shot\_made\_flag : 슛 성공 여부, shot\_type :  
 2점슛 or 3점슛, shot\_zone\_area : 슛을 한 코트 위치, shot\_zone\_basic : 코트 공간  
 (Mid-Range, Restricted Area, In the Paint), shot\_zone\_range: ft범위, team\_id: 팀 식별  
 번호, team\_name: 팀이름, game\_date: 시합 날짜, matchup : 홈팀 vs 어웨이팀,  
 opponent : 상대팀, shot\_id: 슛 식별 번호

```
## Kaggle Exploring Kobe's Shot
library(dplyr)
library(ggplot2)

data <- read.csv("C:/Users/EXEMPC/Desktop/EduData/Dataset/data.csv/data.csv", stringsAsFactors = FALSE)

train <- data[!is.na(data$shot_made_flag),]
test <- data[is.na(data$shot_made_flag),]

train$shot_made_flag <- as.factor(train$shot_made_flag)

names(train)
```

```
#a plot to see accuracy by feature
pplot <- function(feats) {
  feat <- substitute(feats)
  ggplot(data = train, aes_q(x = feat)) +
    geom_bar(aes(fill = shot_made_flag), stat = "count", position = "fill") +
    scale_fill_brewer(palette = "Set1", direction = +1) +
    ggtitle(paste("accuracy by", feat))
}
```

```
# a plot to see position by feature
```

```
courtplot <- function(feats) {
  feat <- substitute(feats)
  train %>%
    ggplot(aes(x = feat, y = y)) +
    geom_point(aes(size = size)) +
    ylim(c(33.7, 35.5)) +
    scale_color_brewer(palette = "Set1", direction = +1) +
    theme_void() +
    ggtitle(paste("position by", feat))
}
```

pplot() : 요인 수준별 정확도 표시하는 사용자 정의 함수  
substitute() : 대체 및 인용 표현  
geom\_bar() : 막대 그래프 표현  
scale\_fill\_brewer: 색상 표현  
ggtitle() : 제목 추가

```
## Kaggle Exploring Kobe's Shot
library(dplyr)
library(ggplot2)

data <- read.csv("C:/Users/EXEMPC/Desktop/EduData/Dataset/data.csv/data.csv", stringsAsFactors = FALSE)

train <- data[!is.na(data$shot_made_flag),]
test <- data[is.na(data$shot_made_flag),]

train$shot_made_flag <- as.factor(train$shot_made_flag)
```

courtplot() : 위치 별로 형상을 매핑  
Substitute() : 대체 및 인용 표현  
geom\_point() : 산점도 표현  
scale\_color\_brewer: 색상 표현  
ggtitle() : 제목 추가

```
}

# a plot to see position by feature
courtplot <- function(feats) {
  feat <- substitute(feats)
  train %>%
    ggplot(aes(x = lon, y = lat)) +
    geom_point(aes_q(color = feat), alpha = 0.7, size = 3) +
    ylim(c(33.7, 34.0883)) +
    scale_color_brewer(palette = "Set1") +
    theme_void() +
    ggtitle(paste(feats))
}
```

```
# Let's take a look at the locations for the various shot_types
courtplot(combined_shot_type) #hard to see here.
```

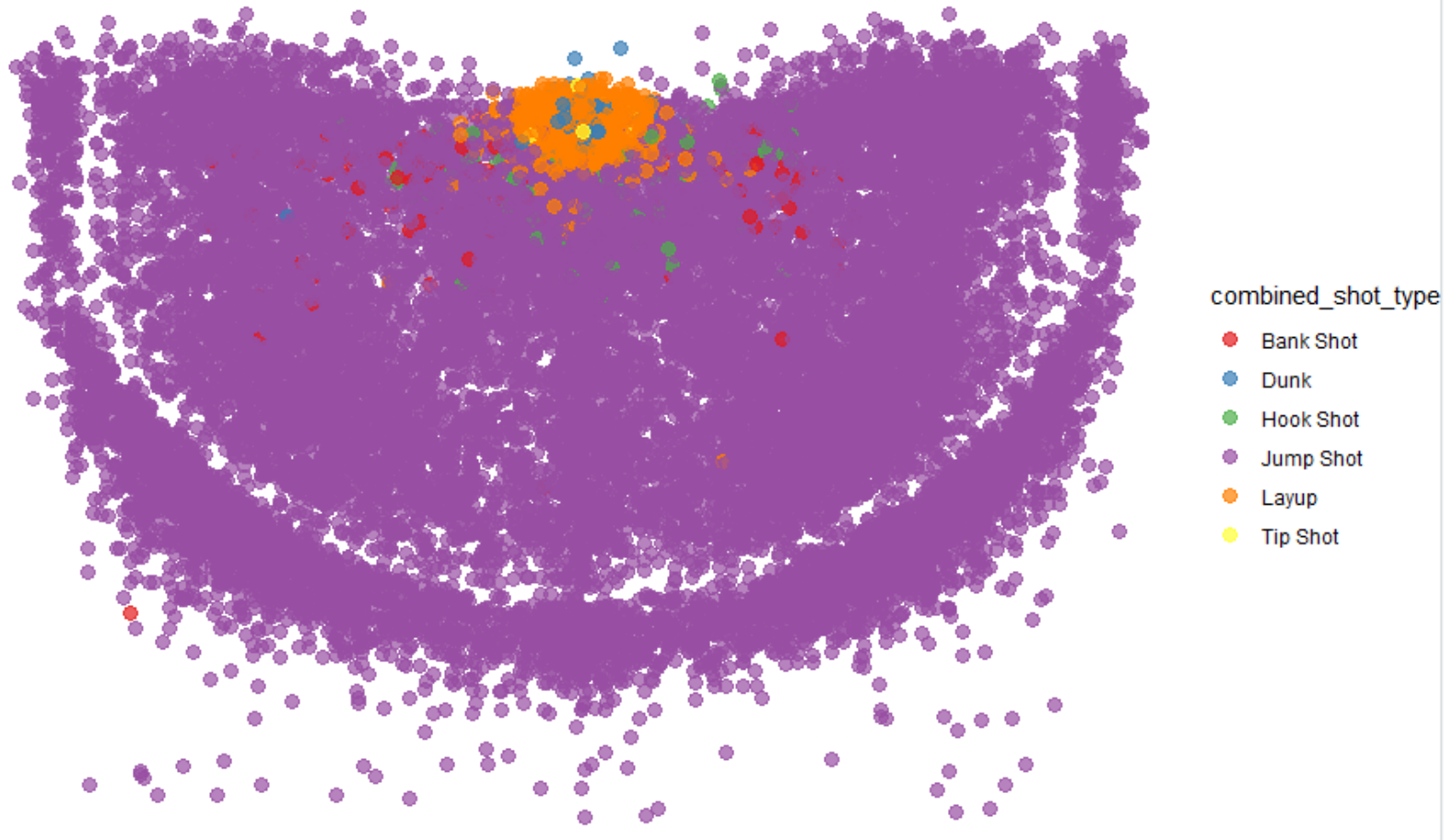
```
# using the ggplot
ggplot() +
  geom_point(data = filter(train, combined_shot_type == "Jump Shot"),
             aes(x = loc_x, y = loc_y, color = shot_made_flag)) +
  geom_point(data = filter(train, combined_shot_type == "Jump Shot"),
             aes(x = loc_x, y = loc_y, color = shot_made_flag)) +
  ylim(c(33.7, 34.4)) +
  scale_color_brewer(palette = "Set1") +
  theme_void() +
  ggtitle("Shot Types")
```

courtplot() : 위치 별로 형상을 맵핑

```
ggplot() +
  geom_point(data = filter(train, shot_distance < 5),
             aes(x = loc_x, y = loc_y,
                  color = shot_made_flag),
             alpha = 0.7, size = 3) +
  scale_color_brewer(palette = "Set1") +
  geom_point(aes(x = 0, y = 0), size = 5, shape = 4) +
  theme_void() +
  ggtitle("Shots from up close")
```

```
# Let's also take a look at all the shots plotted on the court and color them based on whether Kobe made them or not:
ggplot(train, aes(x = loc_x, y = loc_y)) +
  geom_point(aes(color = shot_made_flag), alpha = 0.5, size = 0.5) +
  ylim(c(-50, 400)) +
  theme_void() +
  scale_color_brewer(palette = "Set1") +
  facet_grid(~ shot_made_flag) +
  labs(title = "Shots Made(Blue) vs. Shots Missed(Red)")
```

combined\_shot\_type



Jump Shot의 빈도가 대부분, 슛 타입을 파악하기 힘들다.

```
# Let's take a look at the locations for the various shot_types
courtplot(combined_shot_type) #hard to see here.
```

```
# using the ggplot
ggplot() +
  geom_point(data = filter(train, combined_shot_type == "Jump Shot"),
             aes(x = lon, y = lat), color = "grey", alpha = 0.3, size = 2) +
  geom_point(data = filter(train, combined_shot_type != "Jump Shot"),
             aes(x = lon, y = lat,
                 color = combined_shot_type), alpha = 0.7, size = 3) +
  ylim(c(33.7, 34.0883)) +
  scale_color_brewer(palette = "Set1") +
  theme_void() +
  ggtitle("Shot Types")
```

```
ggplot() +
  geom_point(data = filter(train, combined_shot_type != "Jump Shot").
```

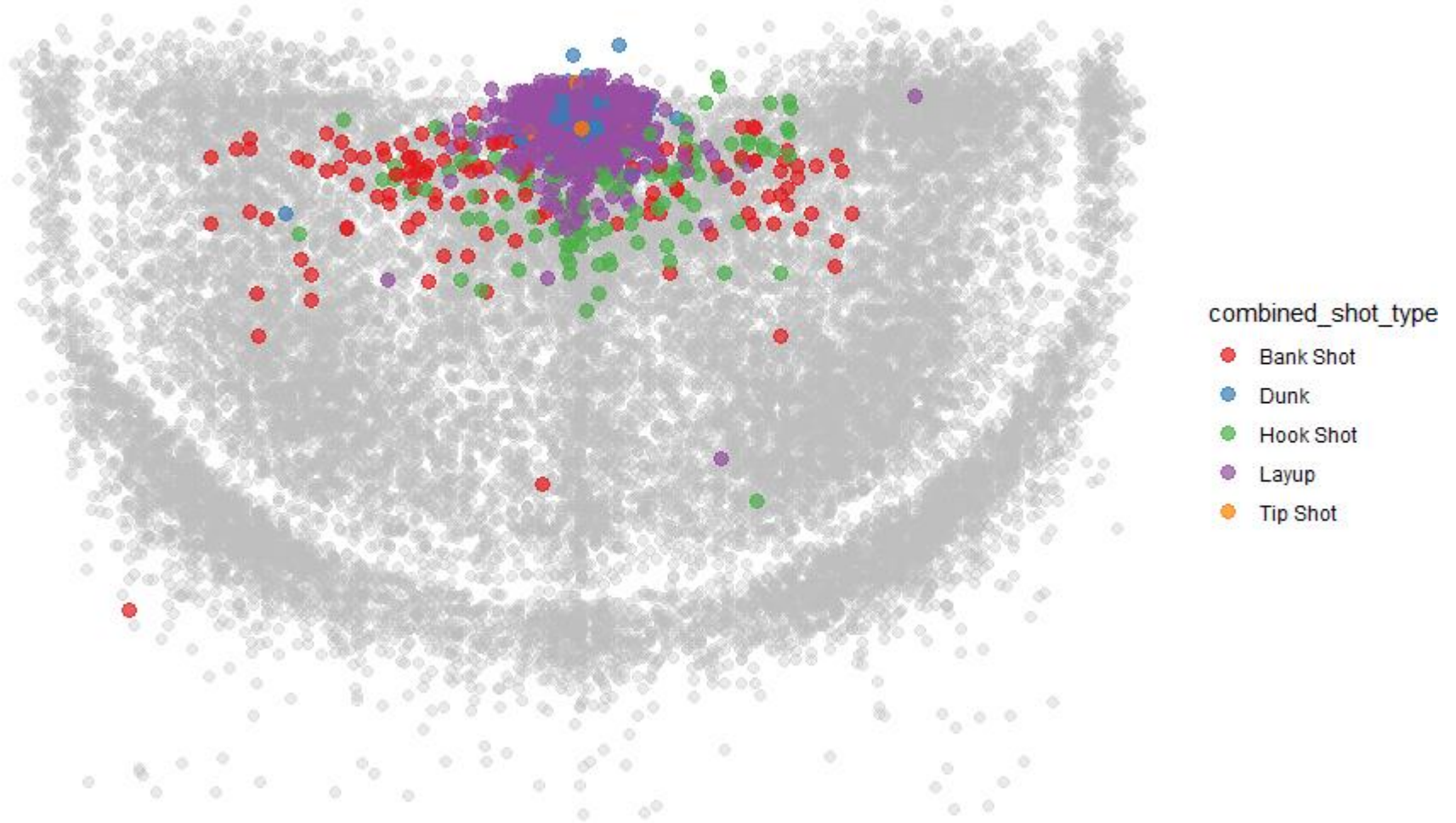
```
scale_color_brewer(palette = "Set1") +
  geom_point(aes(x = lon, y = lat, color = combined_shot_type), alpha = 0.7, size = 3) +
  theme_void() +
  ggtitle("Shot Types")
```

Jump Shot을 grey 색상과 투명도 0.3으로 표현  
Jump shot이 아닌 것들은 combined\_shot\_type 별로 색상을 다르게 표현

```
# Let's also look at the locations of shots made vs. missed
ggplot(train, aes(lon, lat, color = shot_made_flag)) +
  geom_point(alpha = 0.3, size = 0.3) +
  ylim(c(-50, 400)) +
  theme_void() +
  scale_color_brewer(palette = "Set1") +
  facet_grid(~ shot_made_flag) +
  labs(title = "Shots Made(Blue) vs. Shots Missed(Red)")
```

t:

## Shot Types



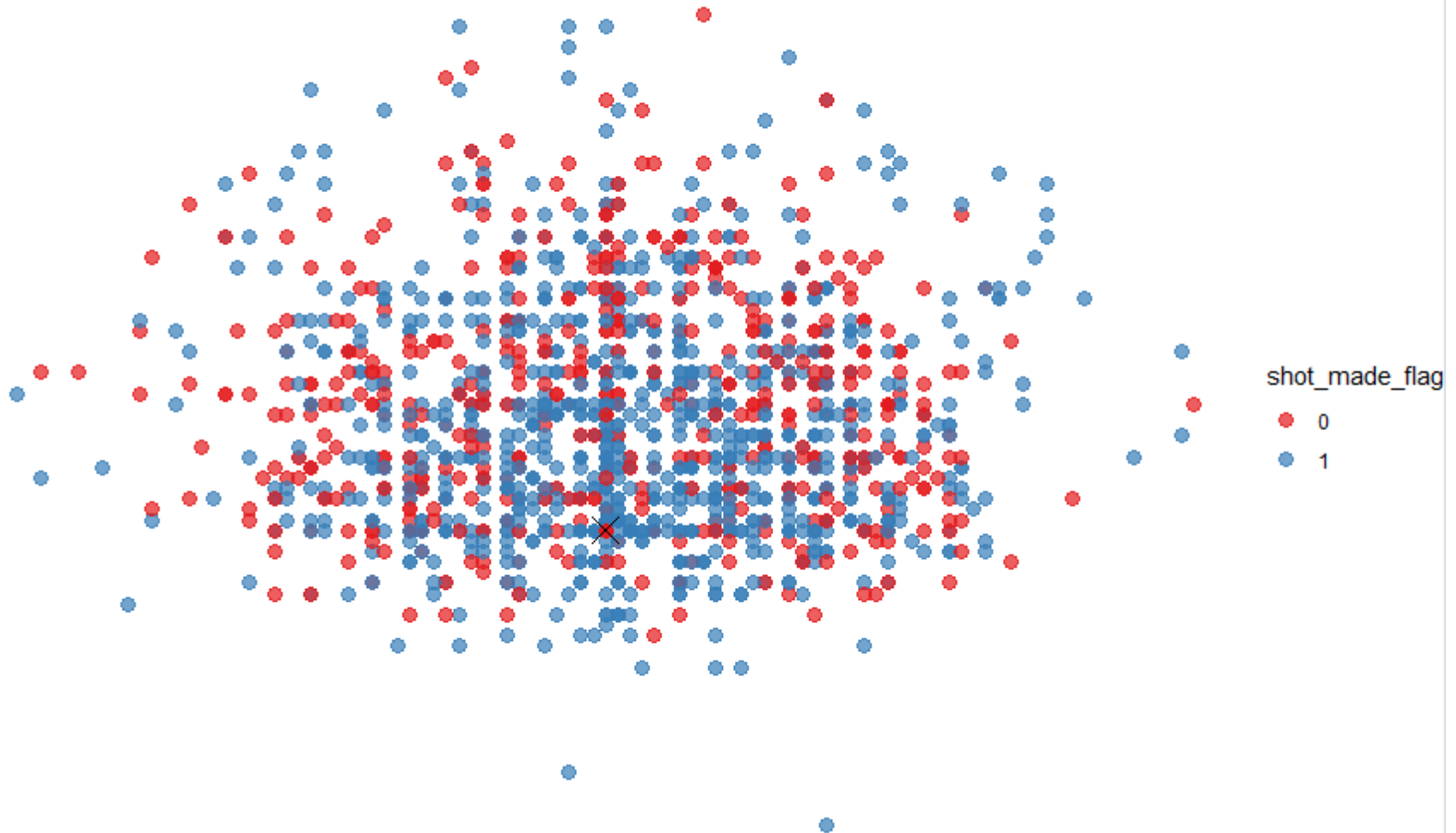
Jump Shot이 아니며 shot\_distance가 5미만인 shot type을 산점도로 표시  
shot\_made\_flag(슛 성공여부)를 기준으로 색상 구분

```
ggplot() +  
  geom_point(data = filter(train, combined_shot_type != "Jump Shot",  
                           shot_distance < 5),  
            aes(x = loc_x, y = loc_y,  
                color = shot_made_flag),  
            alpha = 0.7, size = 3) +  
  scale_color_brewer(palette = "Set1") +  
  geom_point(aes(x = 0, y = 0), size = 5, shape = 4) +  
  theme_void() +  
  ggtitle("Shots from up close")
```

```
# Let's also take a look at all the shots plotted on the court and color them based on whether Kobe made them or not:  
ggplot(train, aes(x = loc_x, y = loc_y)) +  
  geom_point(aes(color = shot_made_flag), alpha = 0.5, size = 0.5) +  
  ylim(c(-50, 400)) +  
  theme_void() +  
  scale_color_brewer(palette = "Set1") +  
  facet_grid(~ shot_made_flag) +  
  labs(title = "Shots Made(Blue) vs. Shots Missed(Red)")
```

## 5피트(약 1.5미터) 이내의 슛 정확도

Shots from up close



```
# Let's take a look at the locations for the various shot_types
courtplot(combined_shot_type) #hard to see here.
```

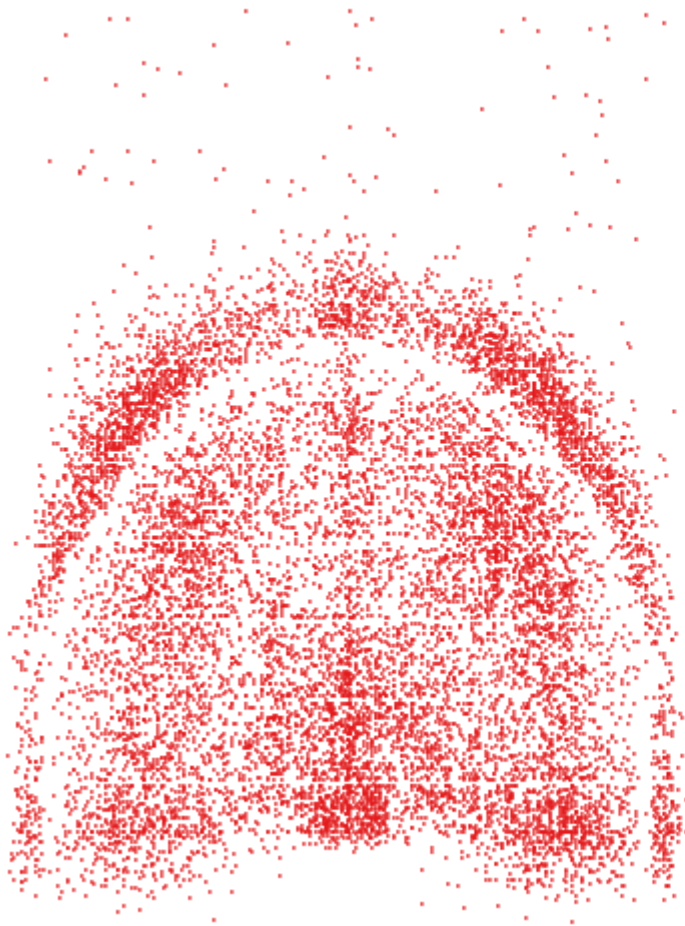
shot\_made\_flag(슛 성공 여부)를 기준으로 색상 표시  
 theme\_void() : 빈 테마 표시  
 scale\_color\_brewer(): 색상 표현  
 facet\_grid() : 집단간 비교를 위한 면 분할  
 labs() : 축, 범례 및 플롯 레이블 수정

```
# Let's also take a look at all the shots plotted on the court and color them based on whether Kobe made them or not:
ggplot(train, aes(x = loc_x, y = loc_y)) +
  geom_point(aes(color = shot_made_flag), alpha = 0.5, size = 0.5) +
  ylim(c(-50, 400)) +
  theme_void() +
  scale_color_brewer(palette = "Set1") +
  facet_grid(~ shot_made_flag) +
  labs(title = "Shots Made(Blue) vs. Shots Missed(Red)")
```

Shots Made(Blue) vs. Shots Missed(Red)

0

1



shot\_made\_flag  
 • 0  
 • 1

거리가 정확도에 큰 영향을 미치고 있다. 빨강(슛 실패), 파랑(슛 성공)

```
# Let's plot shot Distribution by x_bins
train$x_bins <- cut(train$loc_x, breaks = 25)
pplot(x_bins) + geom_bar() + ggtitle("Shot Distribution by x_bins") +
  theme(axis.text.x = element_blank())
```

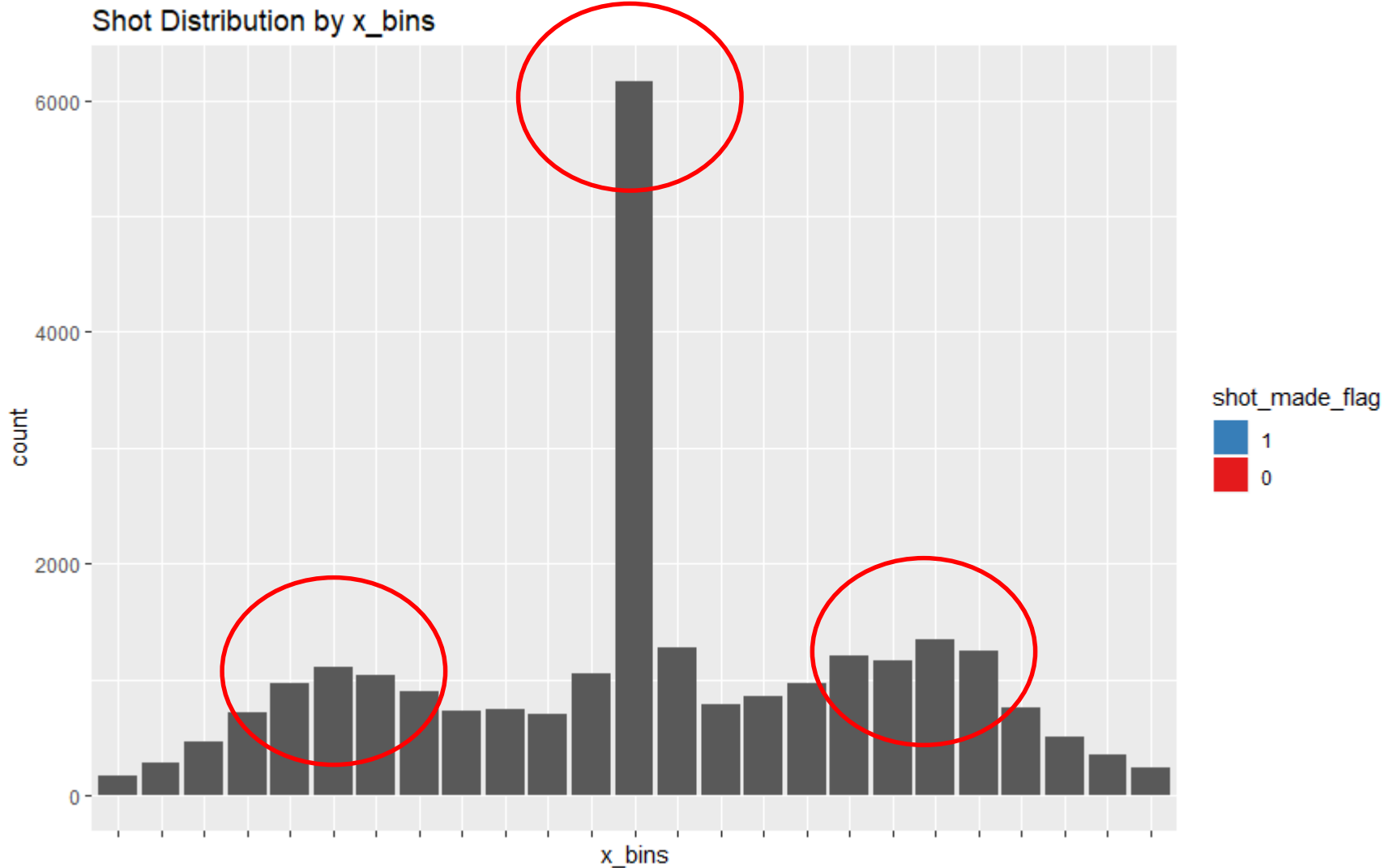
cut(train\$loc\_x, breaks = 25) : loc\_x 변수를 factor 타입으로 변환  
pplot() : 요인 수준별 정확도를 표시  
geom\_bar() : 막대 그래프 맵핑  
ggtitle() : 제목 설정  
theme() : 테마설정

#And let's look at different factors plotted by accuracy:

```
pplot(minutes_remaining)
pplot(period)
pplot(seconds_remaining)
```

#Let's also take a look at the histogram by seconds\_remaining

```
pplot(seconds_remaining) + geom_bar() + ggtitle("Histogram of Shots by second_remaining")
pplot(season) + coord_flip()
pplot(shot_distance) + xlim(0, 60)
pplot(combined_shot_type)
pplot(shot_type)
pplot(shot_zone_area) + coord_flip()
pplot(shot_zone_basic) + coord_flip()
pplot(opponent) + coord_flip()
```



대부분의 슛 시도는 골밑 부분, 골대에 충분히 가까워지면 점프슛 보다는 Restricted Area에서의 슛을 시도하고 있음

```
# Let's plot shot Distribution by x_bins
train$x_bins <- cut(train$loc_x, breaks = 25)
pplot(x_bins) + geom_bar() + ggtitle("Shot Distribution by x_bins") +
  theme(axis.text.x = element_blank())
```

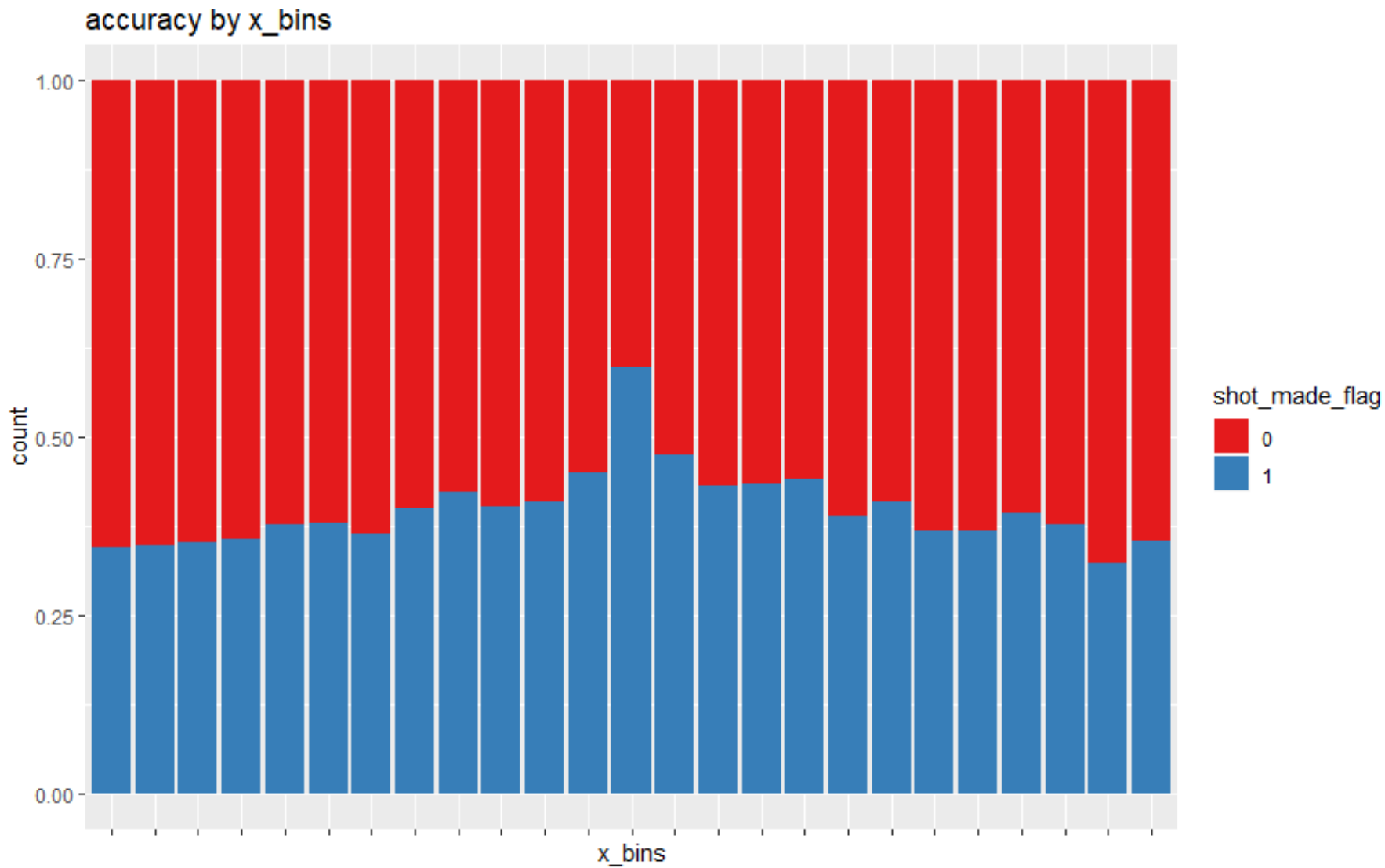
```
pplot(x_bins) + theme(axis.text.x = element_blank())
```

x\_bins의 정확도를 나타냄

```
#Let's also plot the different spatial features:
courtplot(shot_zone_area)
courtplot(shot_zone_basic)
courtplot(shot_zone_range)
```

```
#And let's look at different factors plotted by accuracy:
pplot(minutes_remaining)
pplot(period)
pplot(seconds_remaining)
```

```
#Let's also take a look at the histogram by seconds_remaining
pplot(seconds_remaining) + geom_bar() + ggtitle("Histogram of Shots by second_remaining")
pplot(season) + coord_flip()
pplot(shot_distance) + xlim(0, 60)
pplot(combined_shot_type)
pplot(shot_type)
pplot(shot_zone_area) + coord_flip()
pplot(shot_zone_basic) + coord_flip()
pplot(opponent) + coord_flip()
```



```
# Let's plot shot Distribution by x_bins
train$x_bins <- cut(train$loc_x, breaks = 25)
pplot(x_bins) + geom_bar() + ggtitle("Shot Distribution by x_bins") +
  theme(axis.text.x = element_blank())
```

```
pplot(x_bins) + theme(axis.text.x = element_blank())
```

```
train %>% count(action_type) %>%
  arrange(desc(n)) %>% filter(n < 20) -> actions
```

train 데이터의 action\_type 변수를 카운트 후  
내림차순으로 정렬, 카운트가 20미만인 데이터를  
actions에 할당

```
courtplot(shot_zone_area)
courtplot(shot_zone_basic)
courtplot(shot_zone_range)
```

```
#And let's look at different factors plotted by accuracy:
```

```
pplot(minutes_remaining)
pplot(period)
pplot(seconds_remaining)
```

```
#Let's also take a look at the histogram by seconds_remaining
```

```
pplot(seconds_remaining) + geom_bar() + ggtitle("Histogram of Shots by second_remaining")
pplot(season) + coord_flip()
pplot(shot_distance) + xlim(0, 60)
pplot(combined_shot_type)
pplot(shot_type)
pplot(shot_zone_area) + coord_flip()
pplot(shot_zone_basic) + coord_flip()
pplot(opponent) + coord_flip()
```

```
# Let's plot shot Distribution by x_bins
train$x_bins <- cut(train$loc_x, breaks = 25)
pplot(x_bins) + geom_bar() + ggtitle("Shot Distribution by x_bins") +
  theme(axis.text.x = element_blank())

pplot(x_bins) + theme(axis.text.x = element_blank())

train %>% count(action_type) %>%
  arrange(desc(n)) %>% filter(n < 20) -> actions
train$action_type[train$action_type %in% actions$action_type] <- "other"

prop.table(table(train$action_type, train$shot_made_flag),1) -> temp
```

x 를 action\_type(시도한 슛의 종류), y를 shot\_made\_flag(슛 성공 여부) 로 하는  
비율표를 생성하여 temp에 할당

```
#And let's look at different factors plotted by accuracy:
pplot(minutes_remaining)
pplot(period)
pplot(seconds_remaining)

#Let's also take a look at the histogram by seconds_remaining
pplot(seconds_remaining) + geom_bar() + ggtitle("Histogram of Shots by second_remaining")
pplot(season) + coord_flip()
pplot(shot_distance) + xlim(0, 60)
pplot(combined_shot_type)
pplot(shot_type)
pplot(shot_zone_area) + coord_flip()
pplot(shot_zone_basic) + coord_flip()
pplot(opponent) + coord_flip()
```

```
# Let's plot shot Distribution by x_bins
train$x_bins <- cut(train$loc_x, breaks = 25)
pplot(x_bins) + geom_bar() + ggtitle("Shot Distribution by x_bins") +
  theme(axis.text.x = element_blank())

pplot(x_bins) + theme(axis.text.x = element_blank())

train %>% count(action_type) %>%
  arrange(desc(n)) %>% filter(n < 20) -> actions
train$action_type[train$action_type %in% actions$action_type] <- "other"

prop.table(table(train$action_type, train$shot_made_flag), 1) -> temp
as.data.frame.matrix(temp) -> temp
```

table 형태의 temp를 데이터 프레임으로 변환

```
#And let's look at different factors plotted by accuracy:
pplot(minutes_remaining)
pplot(period)
pplot(seconds_remaining)
```

```
#Let's also take a look at the histogram by seconds_remaining
pplot(seconds_remaining) + geom_bar() + ggtitle("Histogram of Shots by second_remaining")
pplot(season) + coord_flip()
pplot(shot_distance) + xlim(0, 60)
pplot(combined_shot_type)
pplot(shot_type)
pplot(shot_zone_area) + coord_flip()
pplot(shot_zone_basic) + coord_flip()
pplot(opponent) + coord_flip()
```

```
# Let's plot shot Distribution by x_bins
train$x_bins <- cut(train$loc_x, breaks = 25)
pplot(x_bins) + geom_bar() + ggtitle("Shot Distribution by x_bins") +
  theme(axis.text.x = element_blank())

pplot(x_bins) + theme(axis.text.x = element_blank())

train %>% count(action_type) %>%
  arrange(desc(n)) %>% filter(n < 20) -> actions
train$action_type[train$action_type %in% actions$action_type] <- "other"

prop.table(table(train$action_type, train$shot_made_flag),1) -> temp
as.data.frame.matrix(temp) -> temp
temp$shot <- rownames(temp)
```

Temp의 행 이름을 shot이라는 새로운 변수에 할당

```
#And let's look at different factors plotted by accuracy.
pplot(minutes_remaining)
pplot(period)
pplot(seconds_remaining)

#Let's also take a look at the histogram by seconds_remaining
pplot(seconds_remaining) + geom_bar() + ggtitle("Histogram of Shots by second_remaining")
pplot(season) + coord_flip()
pplot(shot_distance) + xlim(0, 60)
pplot(combined_shot_type)
pplot(shot_type)
pplot(shot_zone_area) + coord_flip()
pplot(shot_zone_basic) + coord_flip()
pplot(opponent) + coord_flip()
```

```
# Let's plot shot Distribution by x_bins
train$x_bins <- cut(train$loc_x, breaks = 25)
pplot(x_bins) + geom_bar() + ggtitle("Shot Distribution by x_bins") +
  theme(axis.text.x = element_blank())

pplot(x_bins) + theme(axis.text.x = element_blank())

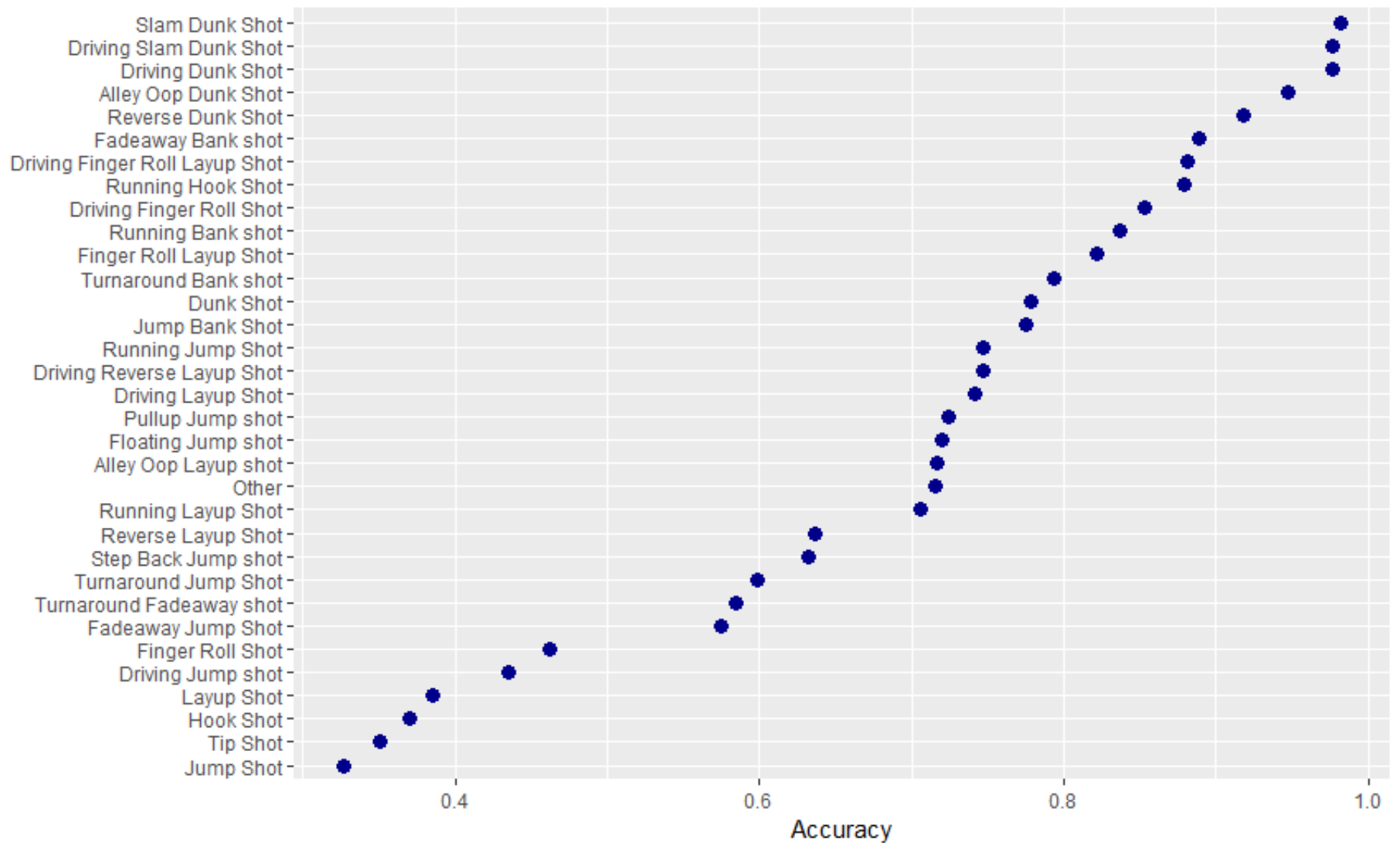
train %>% count(action_type) %>%
  arrange(desc(n)) %>% filter(n < 20) -> actions
train$action_type[train$action_type %in% actions$action_type] <- "other"

prop.table(table(train$action_type, train$shot_made_flag),1) -> temp
as.data.frame.matrix(temp) -> temp
temp$shot <- rownames(temp)
ggplot(temp, aes(x = reorder(shot, `1`), y = 1)) +
  geom_point(aes(y = `1`), size = 3, color = "dark blue", stat = "identity") +
  coord_flip() +
  labs(y = "Accuracy", x = "", title = "Accuracy by shot_type")
```

ggplot(temp, aes(x = reorder(shot, '1'), y=1)) : temp 데이터를 객체로 하고  
 x축: shot(시도한 슛 종류)를 크기 순서대로 정렬, y축 : 슛 성공(1)  
 geom\_point() : 산 점도 표현, coord\_flip() : x축과 y축 변경

```
pplot(seconds_remaining) + geom_bar() + ggtitle("Histogram of Shots by second_remaining")
pplot(season) + coord_flip()
pplot(shot_distance) + xlim(0, 60)
pplot(combined_shot_type)
pplot(shot_type)
pplot(shot_zone_area) + coord_flip()
pplot(shot_zone_basic) + coord_flip()
pplot(opponent) + coord_flip()
```

Accuracy by Shot\_type



```
# Let's plot shot Distribution by x_bins
train$x_bins <- cut(train$loc_x, breaks = 25)
pplot(x_bins) + geom_bar() + ggtitle("Shot Distribution by x_bins") +
  theme(axis.text.x = element_blank())

pplot(x_bins) + theme(axis.text.x = element_blank())
```

courtplot() : 위치 별로 형상을 맵핑  
shot\_zone\_area 변수 : 슛을 한 코트위치

```
#Let's also plot the different spatial features:
courtplot(shot_zone_area)
courtplot(shot_zone_basic)
courtplot(shot_zone_range)
```

```
#And let's look at different factors plotted by accuracy:
pplot(minutes_remaining)
pplot(period)
pplot(seconds_remaining)
```

```
#Let's also take a look at the histogram by seconds_remaining
pplot(seconds_remaining) + geom_bar() + ggtitle("Histogram of Shots by second_remaining")
pplot(season) + coord_flip()
pplot(shot_distance) + xlim(0, 60)
pplot(combined_shot_type)
pplot(shot_type)
pplot(shot_zone_area) + coord_flip()
pplot(shot_zone_basic) + coord_flip()
pplot(opponent) + coord_flip()
```

shot\_zone\_area



```
# Let's plot shot Distribution by x_bins
train$x_bins <- cut(train$loc_x, breaks = 25)
pplot(x_bins) + geom_bar() + ggtitle("Shot Distribution by x_bins") +
  theme(axis.text.x = element_blank())

pplot(x_bins) + theme(axis.text.x = element_blank())

train %>% count(action_type) %>%
  arrange(desc(n)) %>% filter(n < 20) -> actions
```

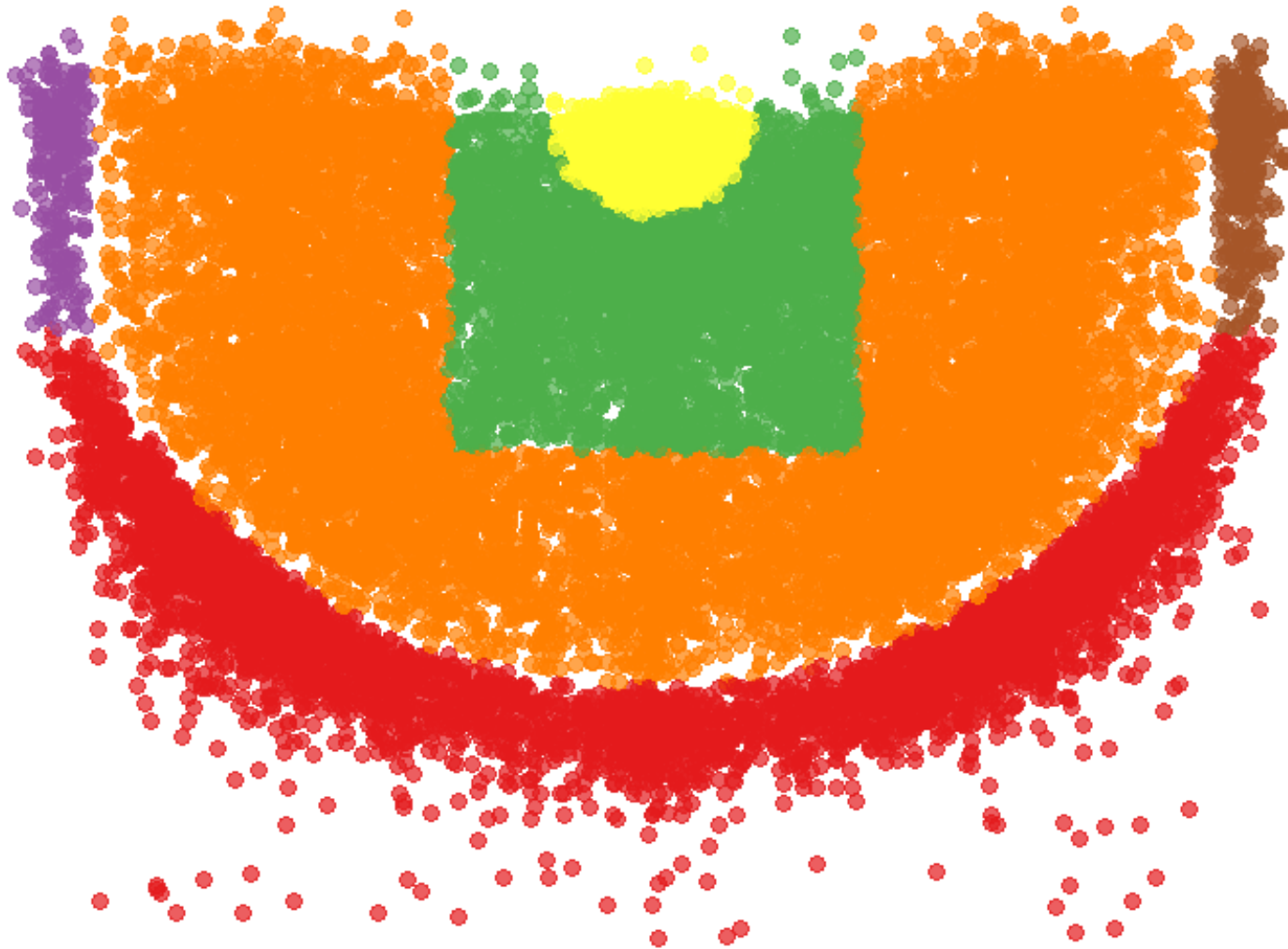
courtplot() : 위치 별로 형상을 맵핑  
shot\_zone\_basic 변수 : 코트 공간

```
courtplot(shot_zone_basic)
courtplot(shot_zone_range)
```

```
#And let's look at different factors plotted by accuracy:
pplot(minutes_remaining)
pplot(period)
pplot(seconds_remaining)
```

```
#Let's also take a look at the histogram by seconds_remaining
pplot(seconds_remaining) + geom_bar() + ggtitle("Histogram of Shots by second_remaining")
pplot(season) + coord_flip()
pplot(shot_distance) + xlim(0, 60)
pplot(combined_shot_type)
pplot(shot_type)
pplot(shot_zone_area) + coord_flip()
pplot(shot_zone_basic) + coord_flip()
pplot(opponent) + coord_flip()
```

shot\_zone\_basic



shot\_zone\_basic

- Above the Break 3
- Backcourt
- In The Paint (Non-RA)
- Left Corner 3
- Mid-Range
- Restricted Area
- Right Corner 3

```
# Let's plot shot Distribution by x_bins
train$x_bins <- cut(train$loc_x, breaks = 25)
pplot(x_bins) + geom_bar() + ggtitle("Shot Distribution by x_bins") +
  theme(axis.text.x = element_blank())

pplot(x_bins) + theme(axis.text.x = element_blank())

train %>% count(action_type) %>%
  arrange(desc(n)) %>% filter(n < 20) -> actions
train$action_type[train$action_type %in% actions$action_type] <- "other"
```

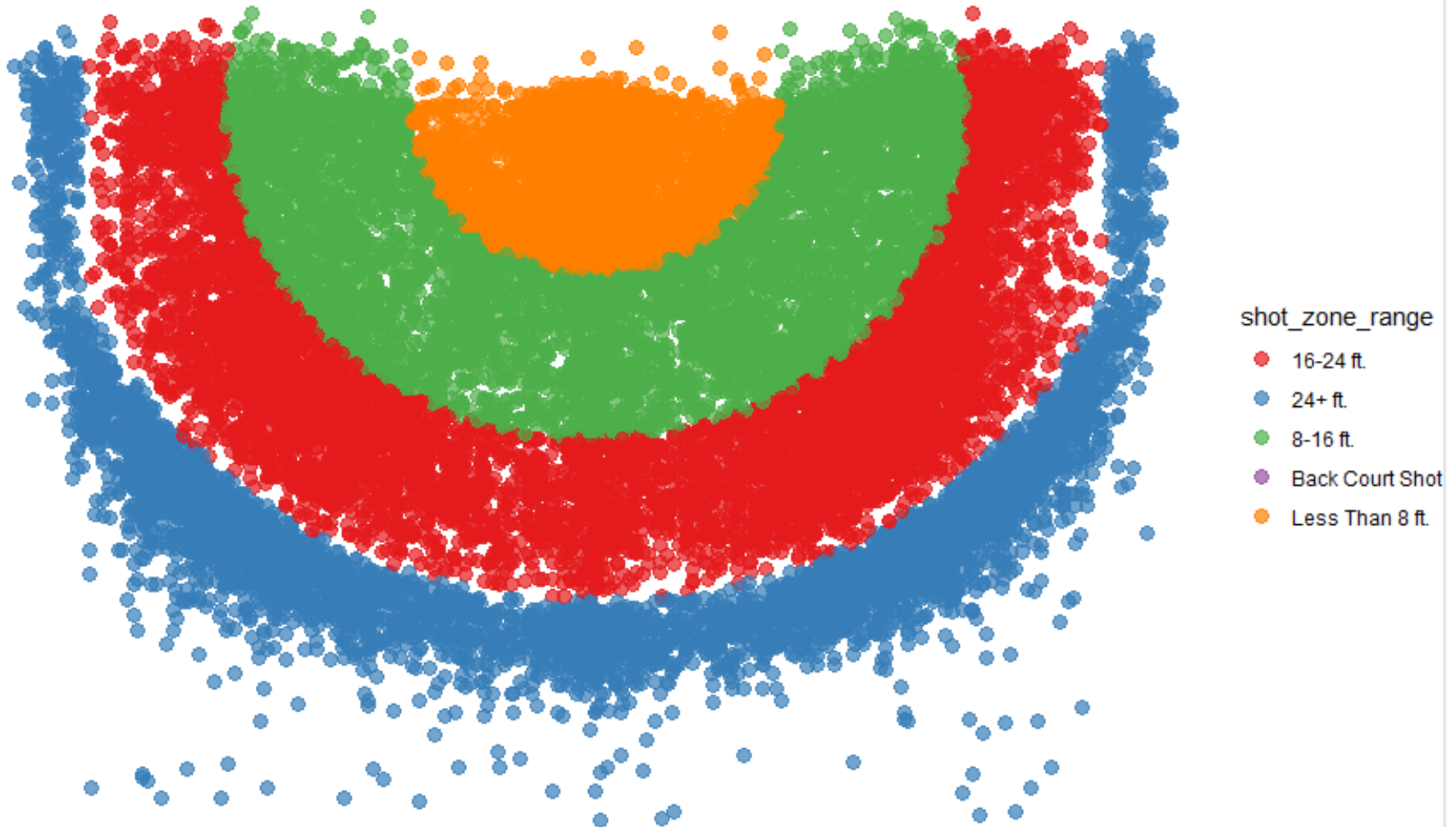
courtplot() : 위치 별로 형상을 맵핑  
shot\_zone\_range 변수 : 거리(ft) 범위

```
courtplot(shot_zone_range)
```

```
#And let's look at different factors plotted by accuracy:
pplot(minutes_remaining)
pplot(period)
pplot(seconds_remaining)
```

```
#Let's also take a look at the histogram by seconds_remaining
pplot(seconds_remaining) + geom_bar() + ggtitle("Histogram of Shots by second_remaining")
pplot(season) + coord_flip()
pplot(shot_distance) + xlim(0, 60)
pplot(combined_shot_type)
pplot(shot_type)
pplot(shot_zone_area) + coord_flip()
pplot(shot_zone_basic) + coord_flip()
pplot(opponent) + coord_flip()
```

shot\_zone\_range



```
# Let's plot shot Distribution by x_bins
train$x_bins <- cut(train$loc_x, breaks = 25)
pplot(x_bins) + geom_bar() + ggtitle("Shot Distribution by x_bins") +
  theme(axis.text.x = element_blank())

pplot(x_bins) + theme(axis.text.x = element_blank())

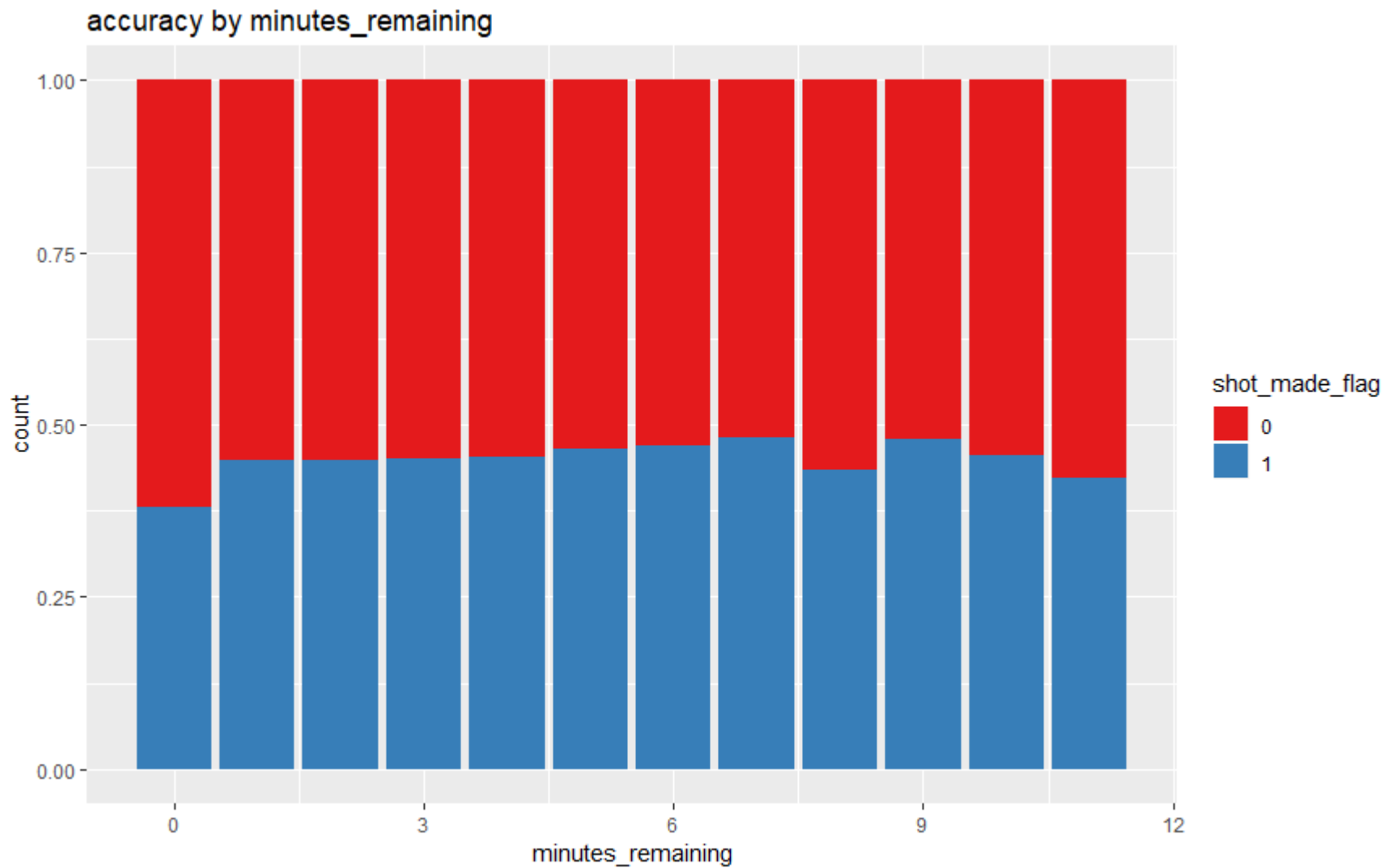
train %>% count(action_type) %>%
  arrange(desc(n)) %>% filter(n < 20) -> actions
train$action_type[train$action_type %in% actions$action_type] <- "other"

prop.table(table(train$action_type, train$shot_made_flag),1) -> temp
as.data.frame.matrix(temp) -> temp
```

pplot() : 요인 수준 별 정확도를 표현  
minutes\_remaining 변수 : 남은 쿼터 시간(한 쿼터는 12분)

```
pplot(minutes_remaining)
pplot(period)
pplot(seconds_remaining)
```

```
#Let's also take a look at the histogram by seconds_remaining
pplot(seconds_remaining) + geom_bar() + ggtitle("Histogram of Shots by second_remaining")
pplot(season) + coord_flip()
pplot(shot_distance) + xlim(0, 60)
pplot(combined_shot_type)
pplot(shot_type)
pplot(shot_zone_area) + coord_flip()
pplot(shot_zone_basic) + coord_flip()
pplot(opponent) + coord_flip()
```



```
# Let's plot shot Distribution by x_bins
train$x_bins <- cut(train$loc_x, breaks = 25)
pplot(x_bins) + geom_bar() + ggtitle("Shot Distribution by x_bins") +
  theme(axis.text.x = element_blank())

pplot(x_bins) + theme(axis.text.x = element_blank())

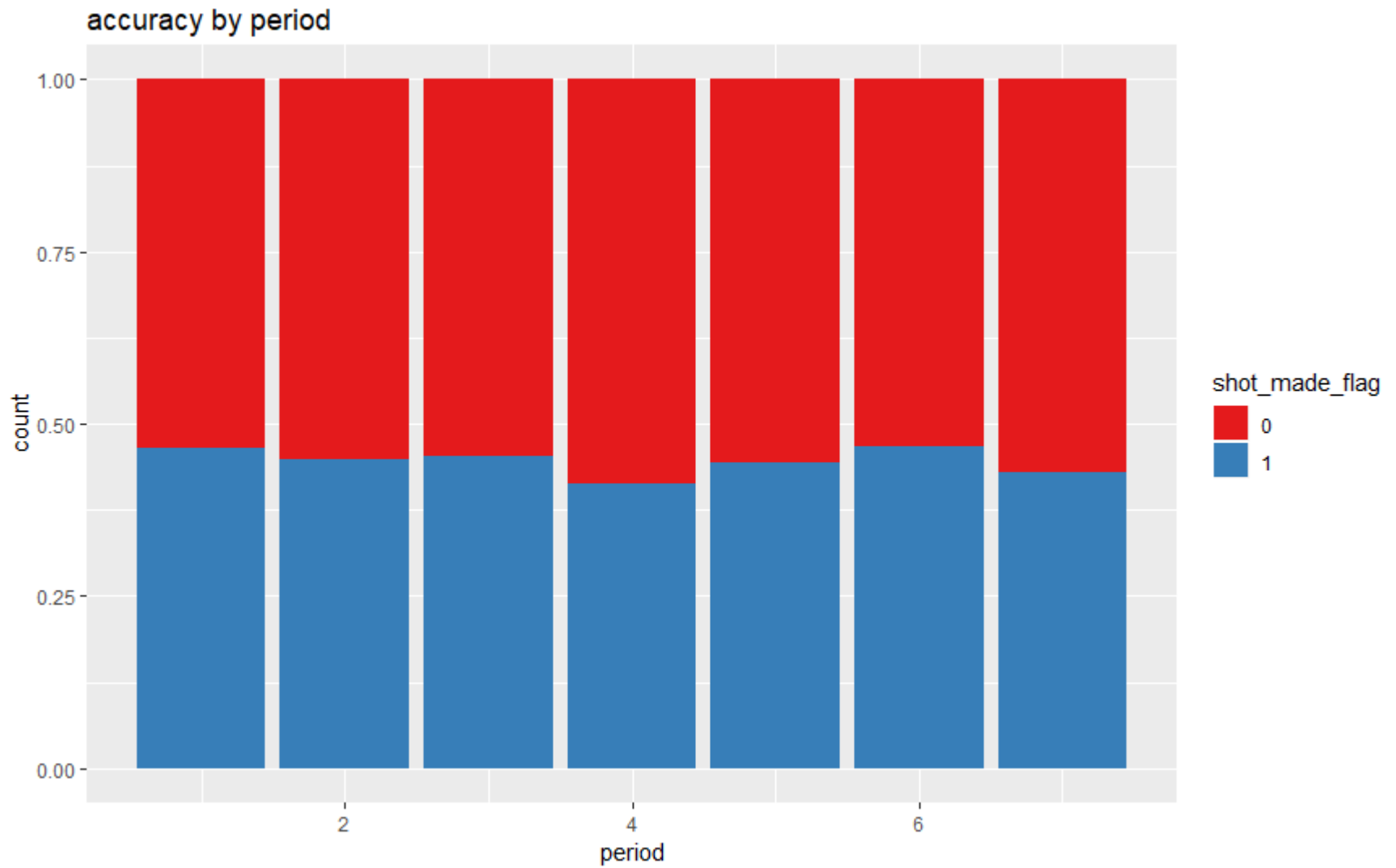
train %>% count(action_type) %>%
  arrange(desc(n)) %>% filter(n < 20) -> actions
train$action_type[train$action_type %in% actions$action_type] <- "other"

prop.table(table(train$action_type, train$shot_made_flag),1) -> temp
as.data.frame.matrix(temp) -> temp
temp$shot <- rownames(temp)
```

pplot() : 요인 수준 별 정확도를 표현  
period 변수 : 쿼터(1~4, 5~7: 연장 쿼터)

```
pplot(period)
pplot(seconds_remaining)
```

```
#Let's also take a look at the histogram by seconds_remaining
pplot(seconds_remaining) + geom_bar() + ggtitle("Histogram of Shots by second_remaining")
pplot(season) + coord_flip()
pplot(shot_distance) + xlim(0, 60)
pplot(combined_shot_type)
pplot(shot_type)
pplot(shot_zone_area) + coord_flip()
pplot(shot_zone_basic) + coord_flip()
pplot(opponent) + coord_flip()
```



```
# Let's plot shot Distribution by x_bins
train$x_bins <- cut(train$loc_x, breaks = 25)
pplot(x_bins) + geom_bar() + ggtitle("Shot Distribution by x_bins") +
  theme(axis.text.x = element_blank())

pplot(x_bins) + theme(axis.text.x = element_blank())

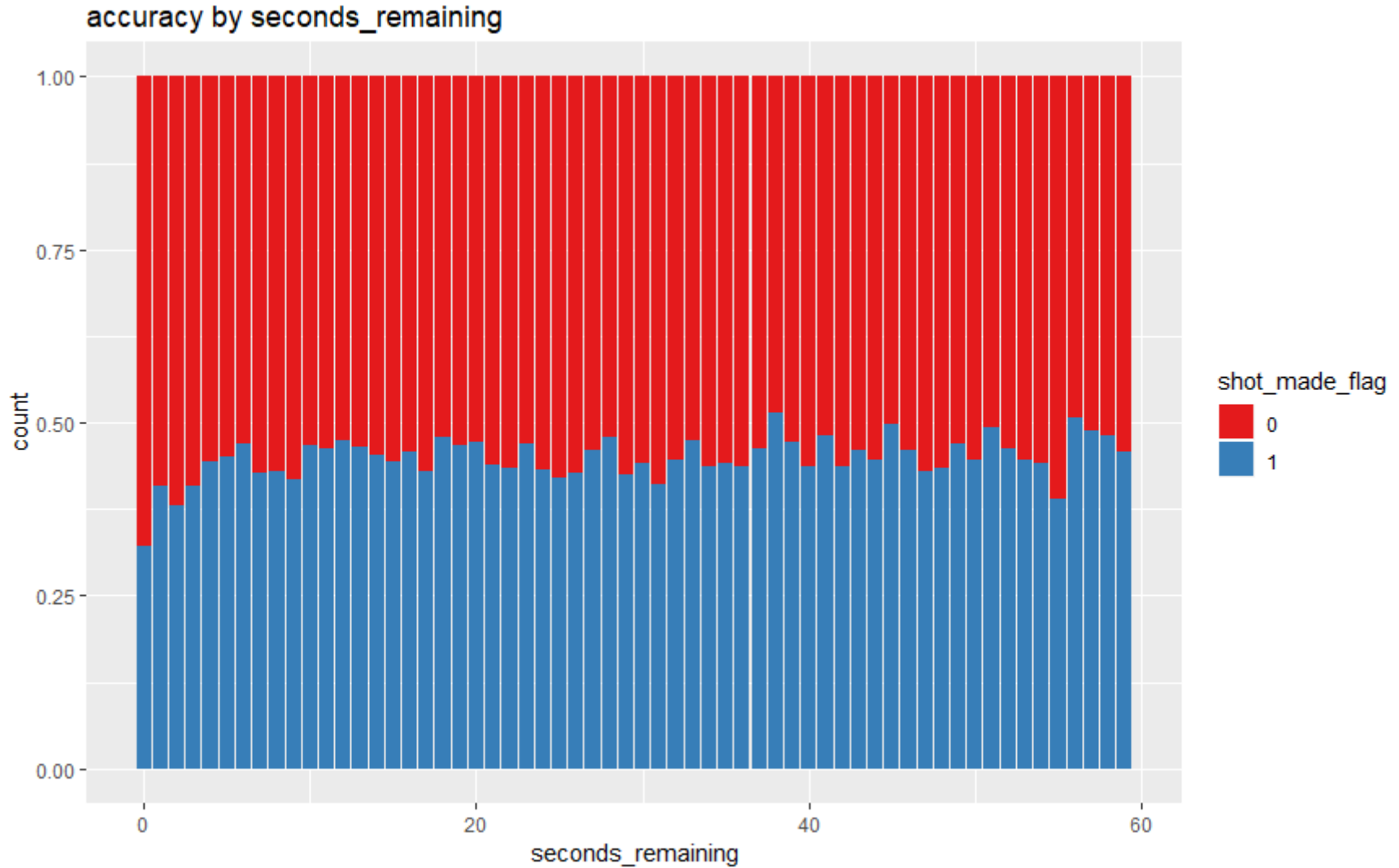
train %>% count(action_type) %>%
  arrange(desc(n)) %>% filter(n < 20) -> actions
train$action_type[train$action_type %in% actions$action_type] <- "other"

prop.table(table(train$action_type, train$shot_made_flag),1) -> temp
as.data.frame.matrix(temp) -> temp
temp$shot <- rownames(temp)
ggplot(temp, aes(x = reorder(shot, `1`), y = 1)) +
```

pplot() : 요인 수준 별 정확도를 표현  
seconds\_remaining 변수 : 남은 쿼터 시간(초)

```
pplot(seconds_remaining)
```

```
#Let's also take a look at the histogram by seconds_remaining
pplot(seconds_remaining) + geom_bar() + ggtitle("Histogram of Shots by second_remaining")
pplot(season) + coord_flip()
pplot(shot_distance) + xlim(0, 60)
pplot(combined_shot_type)
pplot(shot_type)
pplot(shot_zone_area) + coord_flip()
pplot(shot_zone_basic) + coord_flip()
pplot(opponent) + coord_flip()
```



마지막 순간에 슛 성공률이 감소하나 1분(60초)내에서 일관된 슛 성공률을 보여줌

```
# Let's plot shot Distribution by x_bins
train$x_bins <- cut(train$loc_x, breaks = 25)
pplot(x_bins) + geom_bar() + ggtitle("Shot Distribution by x_bins") +
  theme(axis.text.x = element_blank())

pplot(x_bins) + theme(axis.text.x = element_blank())

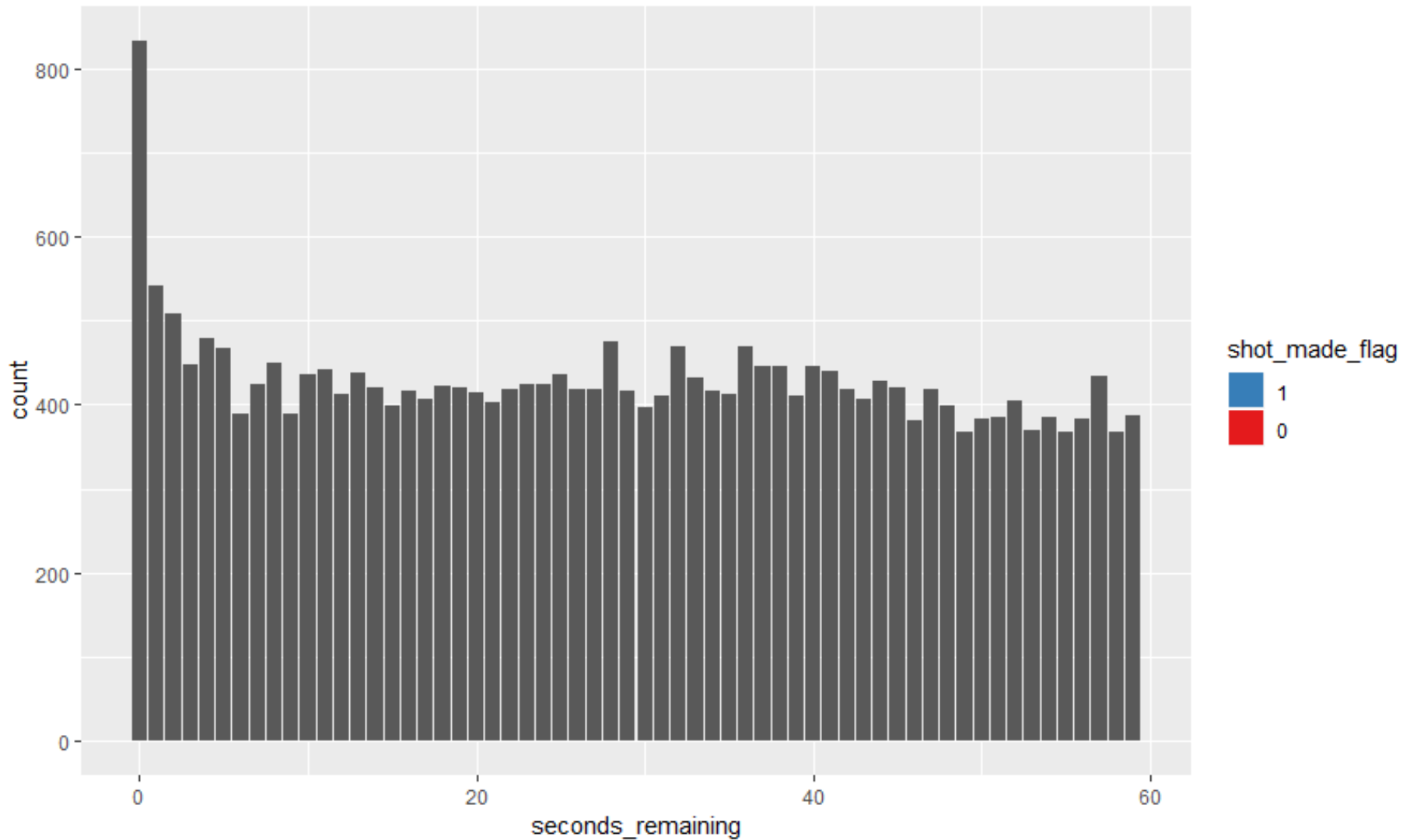
train %>% count(action_type) %>%
  arrange(desc(n)) %>% filter(n < 20) -> actions
train$action_type[train$action_type %in% actions$action_type] <- "other"

prop.table(table(train$action_type, train$shot_made_flag),1) -> temp
as.data.frame.matrix(temp) -> temp
temp$shot <- rownames(temp)
ggplot(temp, aes(x = reorder(shot, `1`), y = 1)) +
  geom_point(aes(y = `1`), size = 3, color = "dark blue", stat = "identity") +
```

pplot() : 요인 수준 별 정확도를 표현  
seconds\_remaining 변수 : 남은 쿼터 시간(초)  
geom\_bar() : 막대 그래프 표현  
ggtitle() : 제목 설정

```
#Let's also take a look at the histogram by seconds_remaining
pplot(seconds_remaining) + geom_bar() + ggtitle("Histogram of Shots by second_remaining")
pplot(season) + coord_flip()
pplot(shot_distance) + xlim(0, 60)
pplot(combined_shot_type)
pplot(shot_type)
pplot(shot_zone_area) + coord_flip()
pplot(shot_zone_basic) + coord_flip()
pplot(opponent) + coord_flip()
```

Histogram of Shots by second\_remaining



```
# Let's plot shot Distribution by x_bins
train$x_bins <- cut(train$loc_x, breaks = 25)
pplot(x_bins) + geom_bar() + ggtitle("Shot Distribution by x_bins") +
  theme(axis.text.x = element_blank())

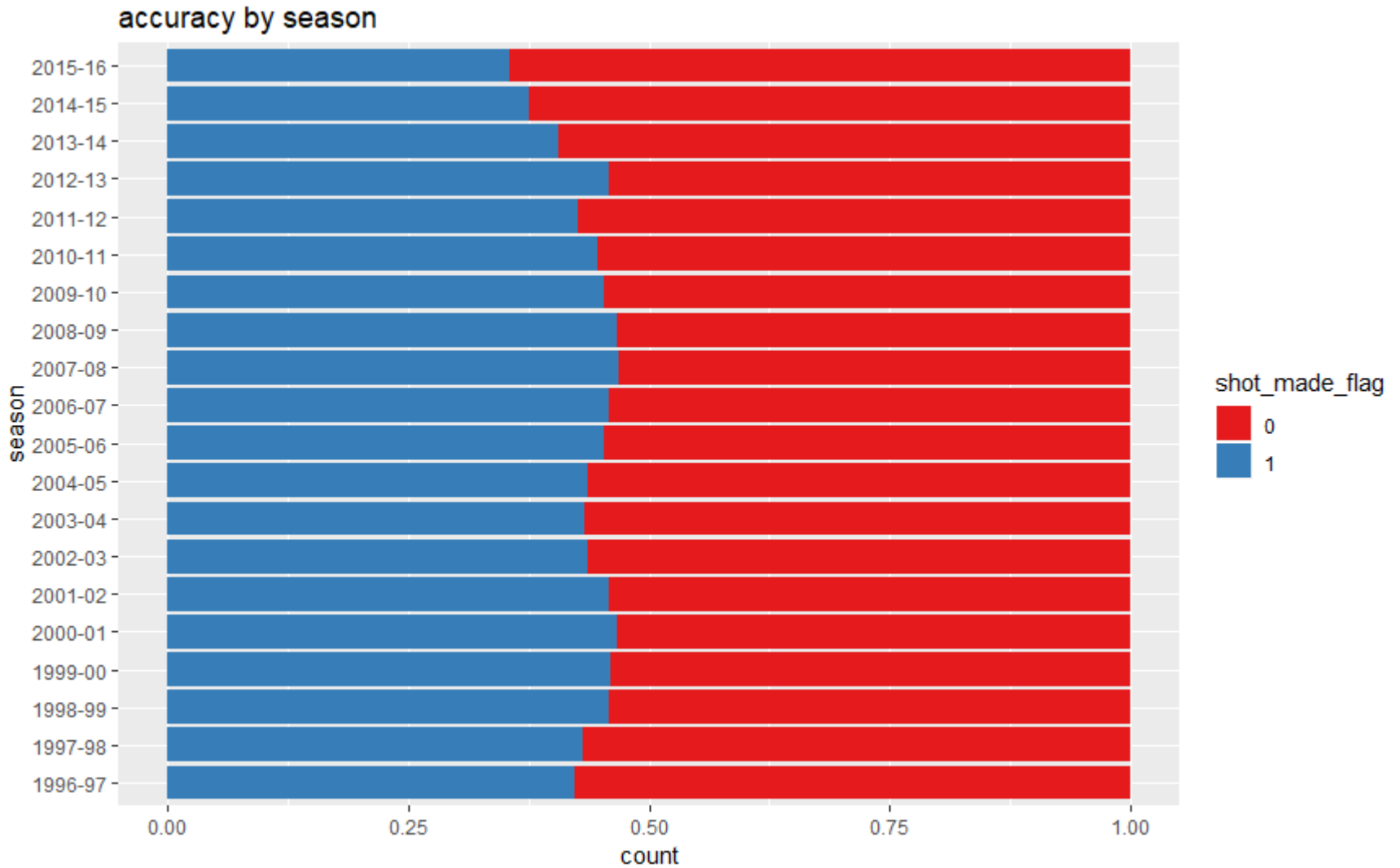
pplot(x_bins) + theme(axis.text.x = element_blank())

train %>% count(action_type) %>%
  arrange(desc(n)) %>% filter(n < 20) -> actions
train$action_type[train$action_type %in% actions$action_type] <- "other"

prop.table(table(train$action_type, train$shot_made_flag),1) -> temp
as.data.frame.matrix(temp) -> temp
temp$shot <- rownames(temp)
ggplot(temp, aes(x = reorder(shot, `1`), y = 1)) +
  geom_point(aes(y = `1`), size = 3, color = "dark blue", stat = "identity") +
  coord_flip() +
  labs(y = "Accuracy", x = "", title = "Accuracy by shot_type")
```

pplot() : 요인 수준 별 정확도를 표현  
 season 변수: 시즌 구분  
 coord\_flip() : x축과 y축 변경

```
pplot(season) + coord_flip()
pplot(shot_distance) + xlim(0, 60)
pplot(combined_shot_type)
pplot(shot_type)
pplot(shot_zone_area) + coord_flip()
pplot(shot_zone_basic) + coord_flip()
pplot(opponent) + coord_flip()
```



05-06~10-11 전성기 시즌에 높은 슛 성공률, 15-16 은퇴시즌에는 가장 낮은 슛 성공률을 기록함

```
# Let's plot shot Distribution by x_bins
train$x_bins <- cut(train$loc_x, breaks = 25)
pplot(x_bins) + geom_bar() + ggtitle("Shot Distribution by x_bins") +
  theme(axis.text.x = element_blank())

pplot(x_bins) + theme(axis.text.x = element_blank())

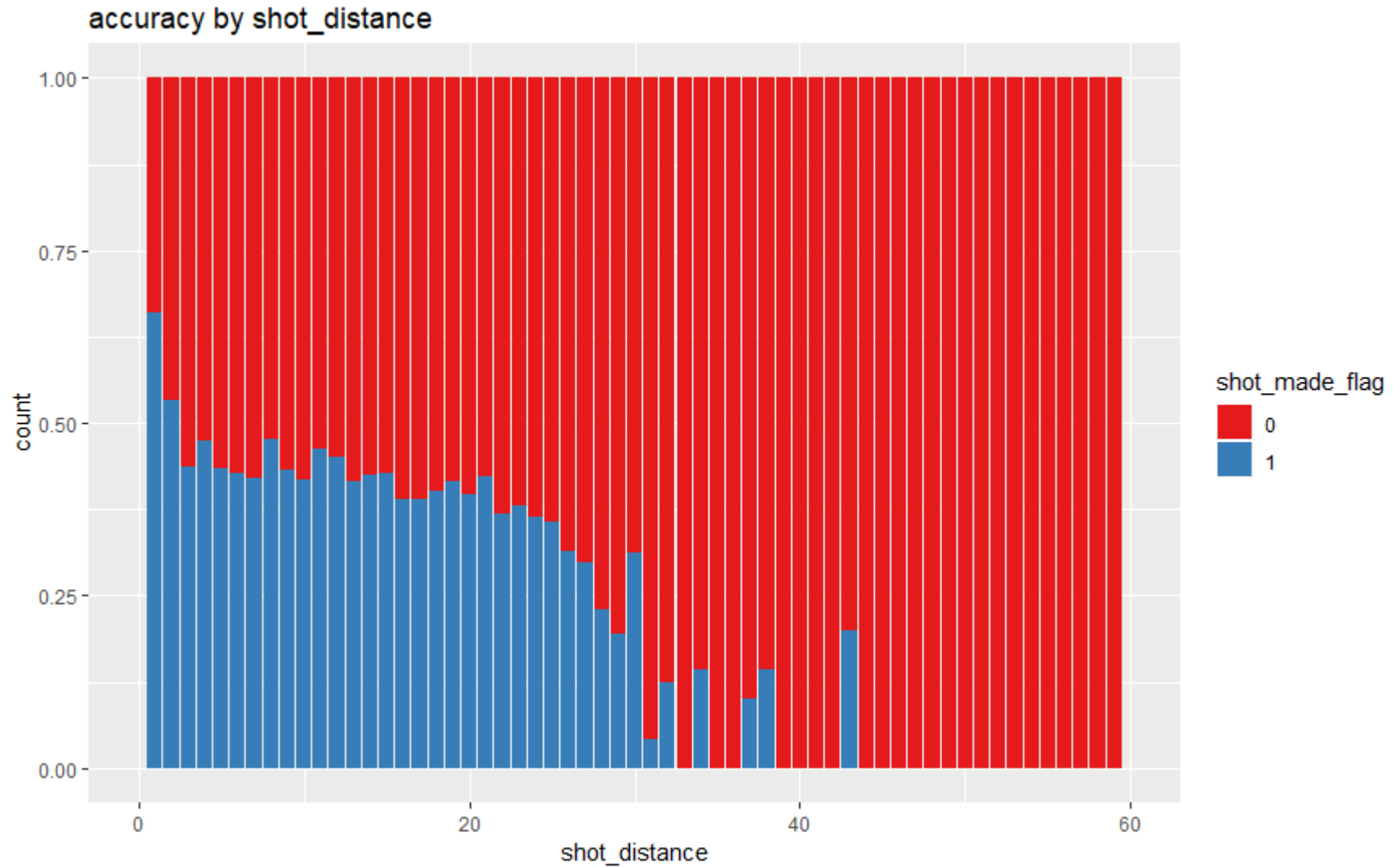
train %>% count(action_type) %>%
  arrange(desc(n)) %>% filter(n < 20) -> actions
train$action_type[train$action_type %in% actions$action_type] <- "other"

prop.table(table(train$action_type, train$shot_made_flag),1) -> temp
as.data.frame.matrix(temp) -> temp
temp$shot <- rownames(temp)
ggplot(temp, aes(x = reorder(shot, `1`), y = 1)) +
  geom_point(aes(y = `1`), size = 3, color = "dark blue", stat = "identity") +
  coord_flip() +
  labs(y = "Accuracy", x = "", title = "Accuracy by shot_type")

#Let's also plot the different spatial features:
```

pplot() : 요인 수준 별 정확도를 표현  
shot\_distance 변수: 쏠 거리(ft)  
xlim(0, 60) : x축 범위 설정

```
pplot(shot_distance) + xlim(0, 60)
pplot(combined_shot_type)
pplot(shot_type)
pplot(shot_zone_area) + coord_flip()
pplot(shot_zone_basic) + coord_flip()
pplot(opponent) + coord_flip()
```



```
# Let's plot shot Distribution by x_bins
train$x_bins <- cut(train$loc_x, breaks = 25)
pplot(x_bins) + geom_bar() + ggtitle("Shot Distribution by x_bins") +
  theme(axis.text.x = element_blank())

pplot(x_bins) + theme(axis.text.x = element_blank())

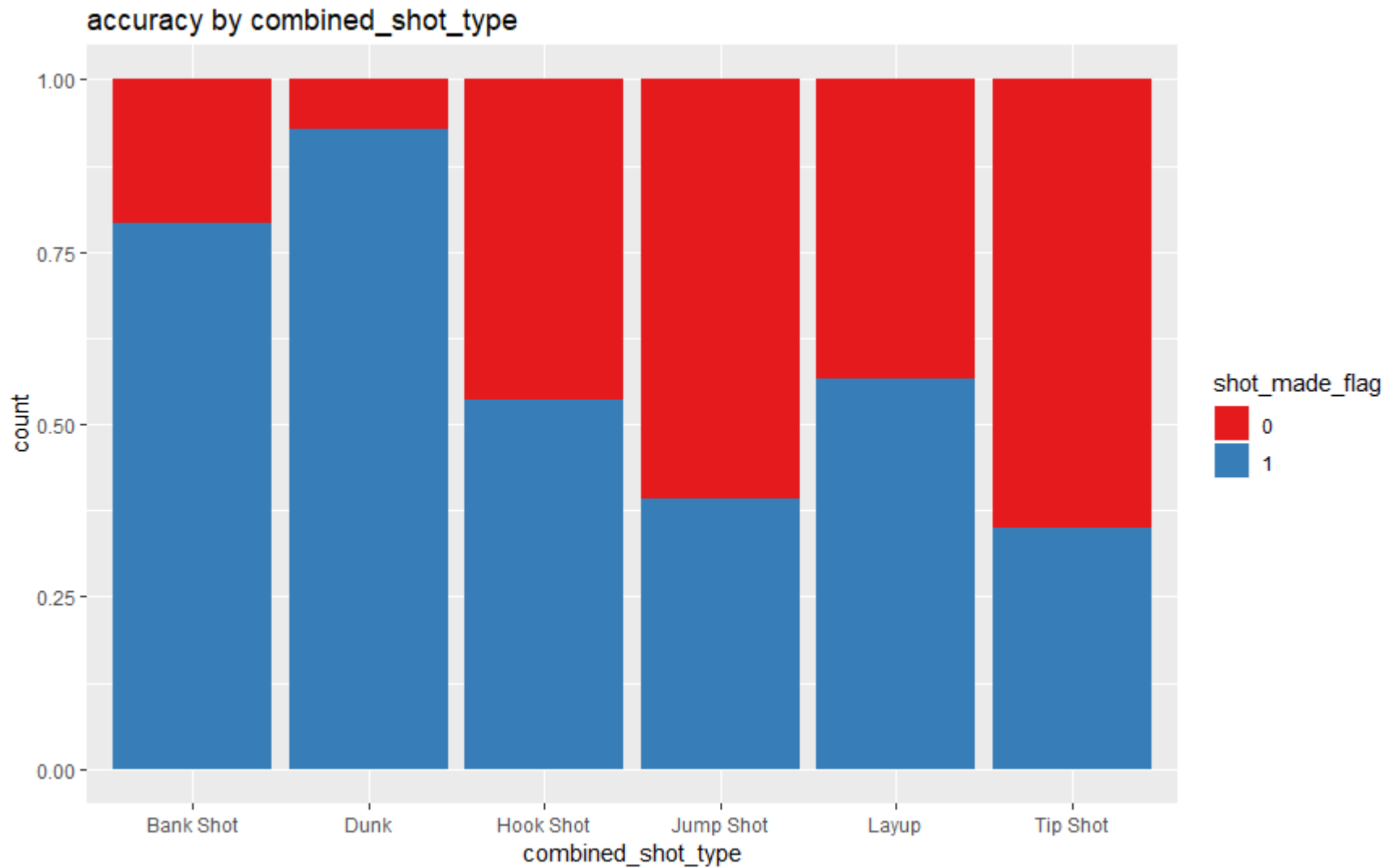
train %>% count(action_type) %>%
  arrange(desc(n)) %>% filter(n < 20) -> actions
train$action_type[train$action_type %in% actions$action_type] <- "other"

prop.table(table(train$action_type, train$shot_made_flag),1) -> temp
as.data.frame.matrix(temp) -> temp
temp$shot <- rownames(temp)
ggplot(temp, aes(x = reorder(shot, `1`), y = 1)) +
  geom_point(aes(y = `1`), size = 3, color = "dark blue", stat = "identity") +
  coord_flip() +
  labs(y = "Accuracy", x = "", title = "Accuracy by shot_type")

#Let's also plot the different spatial features:
courtpplot(shot_zone_area)
```

pplot() : 요인 수준 별 정확도를 표현  
combined\_shot\_type 변수: 슛의 종류를 6가지로 통합

```
pplot(combined_shot_type)
pplot(shot_type)
pplot(shot_zone_area) + coord_flip()
pplot(shot_zone_basic) + coord_flip()
pplot(opponent) + coord_flip()
```



```
# Let's plot shot Distribution by x_bins
train$x_bins <- cut(train$loc_x, breaks = 25)
pplot(x_bins) + geom_bar() + ggtitle("Shot Distribution by x_bins") +
  theme(axis.text.x = element_blank())

pplot(x_bins) + theme(axis.text.x = element_blank())

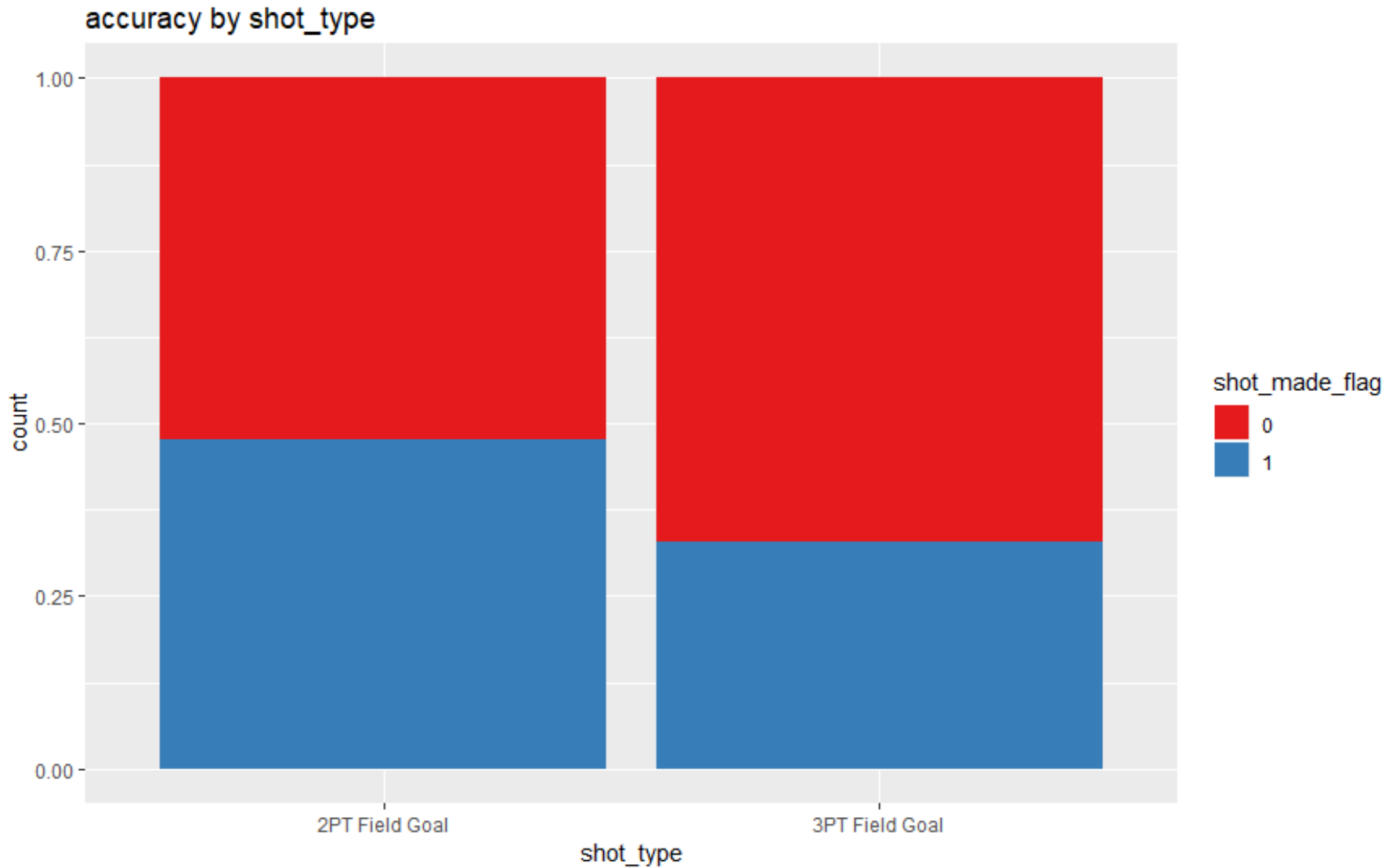
train %>% count(action_type) %>%
  arrange(desc(n)) %>% filter(n < 20) -> actions
train$action_type[train$action_type %in% actions$action_type] <- "other"

prop.table(table(train$action_type, train$shot_made_flag),1) -> temp
as.data.frame.matrix(temp) -> temp
temp$shot <- rownames(temp)
ggplot(temp, aes(x = reorder(shot, `1`), y = 1)) +
  geom_point(aes(y = `1`), size = 3, color = "dark blue", stat = "identity") +
  coord_flip() +
  labs(y = "Accuracy", x = "", title = "Accuracy by shot_type")

#Let's also plot the different spatial features:
courtplot(shot_zone_area)
courtplot(shot_zone_basic)
```

pplot() : 요인 수준 별 정확도를 표현  
shot\_type 변수: 2점슛 or 3점슛

```
pplot(shot_type)
pplot(shot_zone_area) + coord_flip()
pplot(shot_zone_basic) + coord_flip()
pplot(opponent) + coord_flip()
```



```
# Let's plot shot Distribution by x_bins
train$x_bins <- cut(train$loc_x, breaks = 25)
pplot(x_bins) + geom_bar() + ggtitle("Shot Distribution by x_bins") +
  theme(axis.text.x = element_blank())

pplot(x_bins) + theme(axis.text.x = element_blank())

train %>% count(action_type) %>%
  arrange(desc(n)) %>% filter(n < 20) -> actions
train$action_type[train$action_type %in% actions$action_type] <- "other"

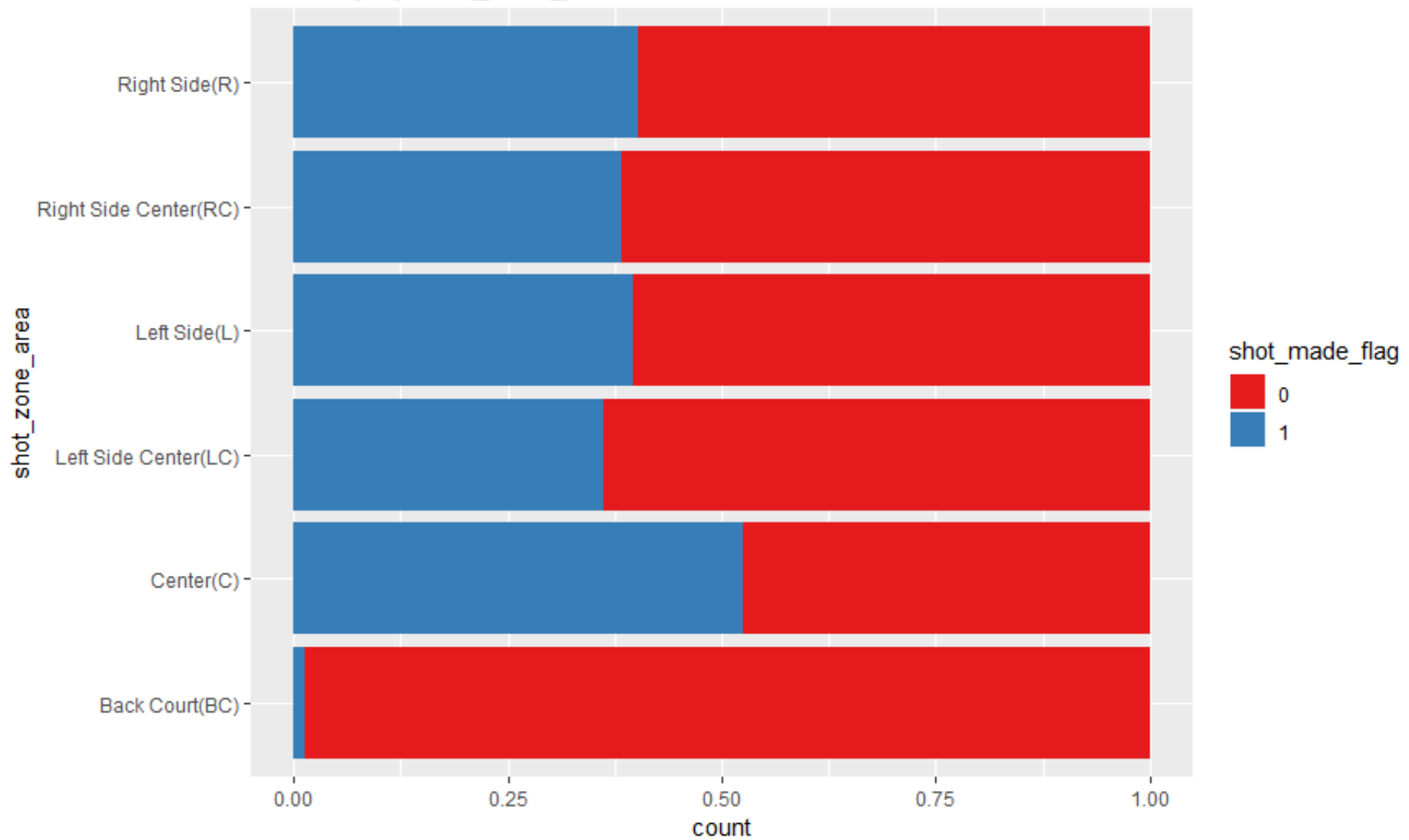
prop.table(table(train$action_type, train$shot_made_flag),1) -> temp
as.data.frame.matrix(temp) -> temp
temp$shot <- rownames(temp)
ggplot(temp, aes(x = reorder(shot, `1`), y = 1)) +
  geom_point(aes(y = `1`), size = 3, color = "dark blue", stat = "identity") +
  coord_flip() +
  labs(y = "Accuracy", x = "", title = "Accuracy by shot_type")

#Let's also plot the different spatial features:
courtplot(shot_zone_area)
courtplot(shot_zone_basic)
courtplot(shot_zone_range)
```

pplot() : 요인 수준 별 정확도를 표현  
shot\_zone\_area 변수: 슛을 한 코트 위치  
coord\_flip() : x축과 y축 변경

```
pplot(shot_zone_area) + coord_flip()
pplot(shot_zone_basic) + coord_flip()
pplot(opponent) + coord_flip()
```

accuracy by shot\_zone\_area



```
# Let's plot shot Distribution by x_bins
train$x_bins <- cut(train$loc_x, breaks = 25)
pplot(x_bins) + geom_bar() + ggtitle("Shot Distribution by x_bins") +
  theme(axis.text.x = element_blank())

pplot(x_bins) + theme(axis.text.x = element_blank())

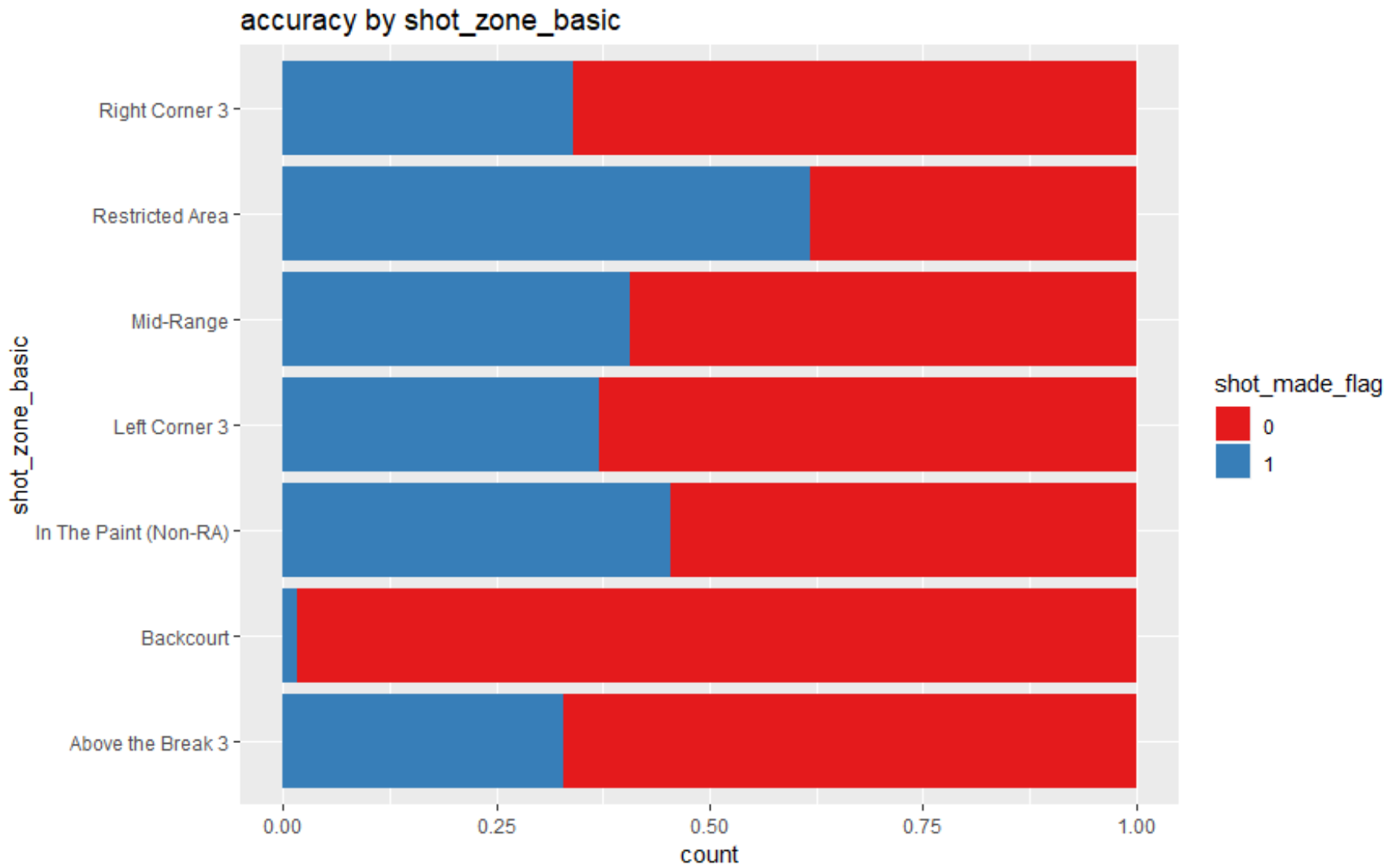
train %>% count(action_type) %>%
  arrange(desc(n)) %>% filter(n < 20) -> actions
train$action_type[train$action_type %in% actions$action_type] <- "other"

prop.table(table(train$action_type, train$shot_made_flag),1) -> temp
as.data.frame.matrix(temp) -> temp
temp$shot <- rownames(temp)
ggplot(temp, aes(x = reorder(shot, `1`), y = 1)) +
  geom_point(aes(y = `1`), size = 3, color = "dark blue", stat = "identity") +
  coord_flip() +
  labs(y = "Accuracy", x = "", title = "Accuracy by shot_type")

#Let's also plot the different spatial features:
courtplot(shot_zone_area)
courtplot(shot_zone_basic)
courtplot(shot_zone_range)
```

pplot() : 요인 수준 별 정확도를 표현  
shot\_zone\_basic 변수: 코트 공간  
coord\_flip() : x축과 y축 변경

```
pplot(shot_zone_basic) + coord_flip()
pplot(opponent) + coord_flip()
```



```
# Let's plot shot Distribution by x_bins
train$x_bins <- cut(train$loc_x, breaks = 25)
pplot(x_bins) + geom_bar() + ggtitle("Shot Distribution by x_bins") +
  theme(axis.text.x = element_blank())

pplot(x_bins) + theme(axis.text.x = element_blank())

train %>% count(action_type) %>%
  arrange(desc(n)) %>% filter(n < 20) -> actions
train$action_type[train$action_type %in% actions$action_type] <- "other"

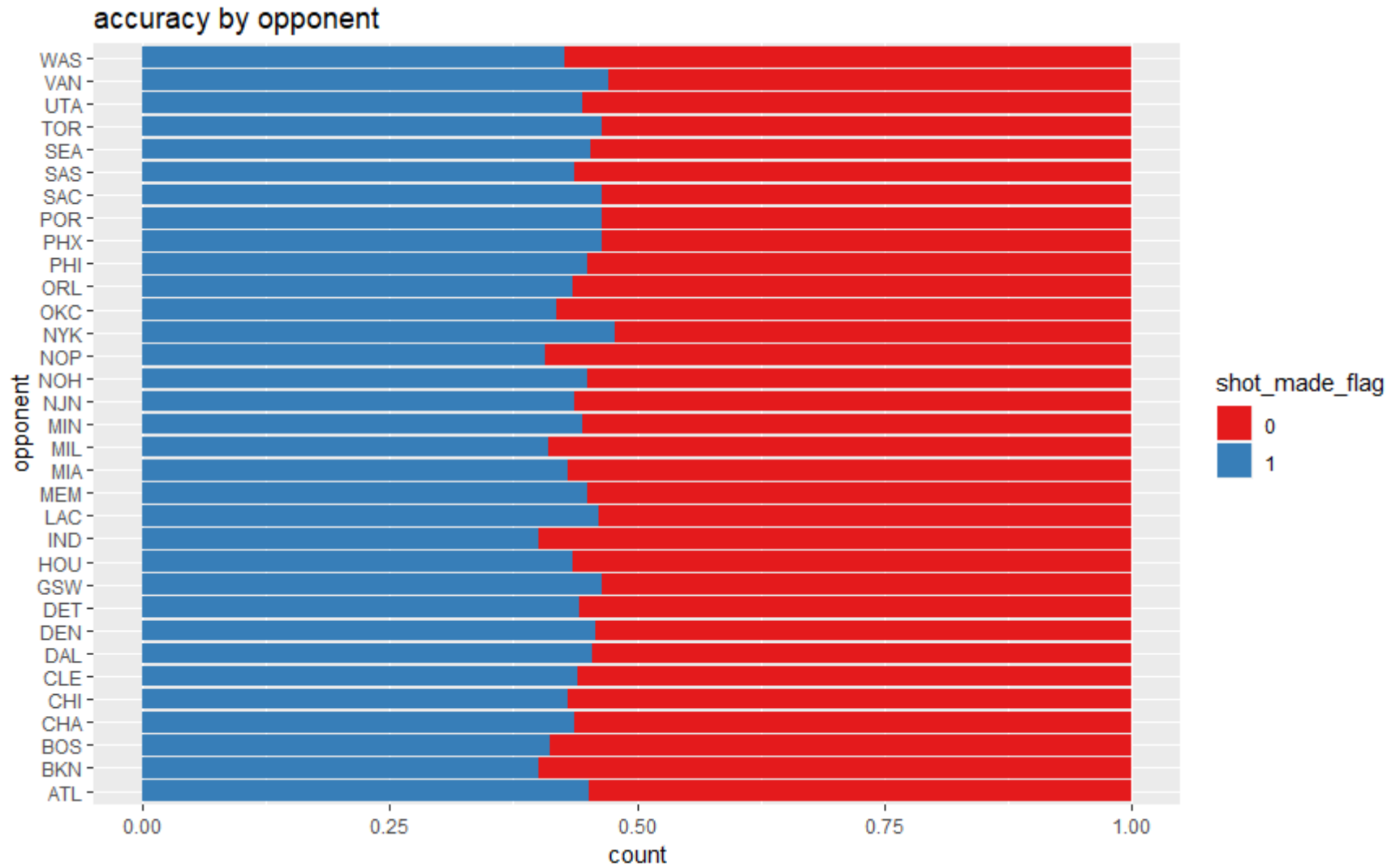
prop.table(table(train$action_type, train$shot_made_flag),1) -> temp
as.data.frame.matrix(temp) -> temp
temp$shot <- rownames(temp)
ggplot(temp, aes(x = reorder(shot, `1`), y = 1)) +
  geom_point(aes(y = `1`), size = 3, color = "dark blue", stat = "identity") +
  coord_flip() +
  labs(y = "Accuracy", x = "", title = "Accuracy by Shot_type")

#Let's also plot the different spatial features:
courtplot(shot_zone_area)
courtplot(shot_zone_basic)
courtplot(shot_zone_range)

#And let's look at different factors plotted by accuracy:
```

pplot() : 요인 수준 별 정확도를 표현  
opponent 변수: 상대팀  
coord\_flip() : x축과 y축 변경

```
pplot(opponent) + coord_flip()
```



# Reference

김영우, “Do it! 쉽게 배우는 R 데이터 분석”, 이지스퍼블리싱(2017)

박찬성, 우현중, 김희석, “통계와 R을 함께 배우는 R까지 2”, 느린생각(2016)

<https://shiny.rstudio.com/gallery/google-charts.html>

<https://blog.naver.com/liberty264/221080779529>

[https://ko.wikipedia.org/wiki/%ED%94%8C%EB%A1%9C%EB%A0%8C%EC%8A%A4\\_%EB%82%98%EC%9D%B4%ED%8C%85%EA%B2%8C%EC%9D%BC](https://ko.wikipedia.org/wiki/%ED%94%8C%EB%A1%9C%EB%A0%8C%EC%8A%A4_%EB%82%98%EC%9D%B4%ED%8C%85%EA%B2%8C%EC%9D%BC)

<http://sei80.com/?p=30031>

<https://peltiertech.com/bullet-charts-in-excel/>

<http://www.hippochart.com/gallery/galcATEGORY.aspx?cate=ChartType&type=Line>

<https://www.tableau.com/ko-kr/learn/whitepapers/which-chart-or-graph-is-right-for-you>

[http://www.numerousmoney.com/bbs/zboard.php?id=MoneyStory&page=30&sn1=&divpage=1&sn=off&ss=on&sc=on&select\\_arrange=headnum&desc=asc&no=478](http://www.numerousmoney.com/bbs/zboard.php?id=MoneyStory&page=30&sn1=&divpage=1&sn=off&ss=on&sc=on&select_arrange=headnum&desc=asc&no=478)

<http://pathway.kr/194>

<https://m.blog.naver.com/PostView.nhn?blogId=kna2511&logNo=209851052&proxyReferer=https%3A%2F%2Fwww.google.co.kr%2F>

<http://info-graphics.kr/?p=855>

<http://blog.naver.com/PostView.nhn?blogId=samsjang&logNo=220802839948&categoryNo=0&parentCategoryNo=10&viewDate=&currentPage=1&postListTopCurrentPage=1&from=search>

<http://igija.tistory.com/category/%ED%86%B5%EA%B3%84+%EC%A7%80%EB%8F%84>

<http://m.dbguide.net/about.db?cmd=view&boardUid=166641&boardConfigUid=19&boardStep=0&categoryUid=1296&movePage=1>

<http://help.bestez.com/qwayneo/help/2000.htm>

<https://www.statcrunch.com/5.0/viewresult.php?resid=2054219>

<https://m.blog.naver.com/PostView.nhn?blogId=2030busan&logNo=220895293245&proxyReferer=https%3A%2F%2Fwww.google.co.kr%2F>

<http://v.auto.daum.net/v/opSExedAAI>

<http://www.fnnews.com/news/201702221711338369>

<http://www.visualdive.com>

<http://infographicslab203.com/portfolio/poster-%EC%9D%B8%EC%8A%A4%ED%84%B4%ED%8A%B8-%EC%8B%9D%ED%92%88%EC%9D%98-%EB%8C%80%EB%AA%85%EC%82%AC-%EB%9D%BC%EB%A9%B4/infographic-korean-ramen/>

<http://mlbpark.donga.com/mlbpark/b.php?&b=bullpen2&id=3311048>

<https://help.plot.ly/excel/box-plot/>

[http://data.si.re.kr/sites/default/files/2015-BR-10\\_103\\_%EA%B7%B8%EB%9E%98%ED%94%84\\_6-86\\_The%20Number%20of%20Cars%20per%20Each%20Household.jpg](http://data.si.re.kr/sites/default/files/2015-BR-10_103_%EA%B7%B8%EB%9E%98%ED%94%84_6-86_The%20Number%20of%20Cars%20per%20Each%20Household.jpg)

[http://support.sas.com/documentation/cdl\\_alternate/ko/vaug/68027/HTML/default/p1f3lpz99yf7o4n1injxy3a1vn3.htm](http://support.sas.com/documentation/cdl_alternate/ko/vaug/68027/HTML/default/p1f3lpz99yf7o4n1injxy3a1vn3.htm)

# Reference

타이타닉, <https://www.kaggle.com/philippsp/interactive-dashboards-in-r>

FIFA 월드컵 데이터 분석, <https://www.kaggle.com/nulldata/fifa-18-what-happened-in-the-league>

Randomforest 분석, <https://www.kaggle.com/benhamner/random-forest-benchmark-r>

<https://m.blog.naver.com/PostView.nhn?blogId=hsganbatte&logNo=220573189325&proxyReferer=http%3A%2F%2Fwww.google.com%2Furl%3Fsa%3Di%26rct%3Dj%26q%3D%26esrc%3Ds%26source%3Dimages%26cd%3D%26cad%3Drja%26uact%3D8%26ved%3D2ahUKEwicgPWLsLfcAhXFi7wKHQumDYgQjRx6BAgBEAU%26url%3Dhttp%253A%252F%252Fm.blog.naver.com%252Fhsganbatte%252F220573189325%26psig%3DAOvVaw19evRafYPkFKX0sDQjUwnl%26ust%3D1532508709211566>

<http://whatisthenext.tistory.com/108>

<https://m.post.naver.com/viewer/postView.nhn?volumeNo=15318712&memberNo=9332686&vType=VERTICAL>

<https://post.naver.com/viewer/postView.nhn?volumeNo=4570264&memberNo=2433587>

<https://news.naver.com/main/read.nhn?oid=030&aid=0002423191>

Kobe shot selection 분석, <https://www.kaggle.com/apapiu/exploring-kobe-s-shots>

# Q&A



# 감사합니다

엑셈 | 김찬경 연구원