11. 일변량 자료 탐색

수많은 변수들로 이루어져있는 통계 자료가 주어지면 각 개별 변수들이 갖고 있는 정보를 탐색하는 것이 통계분석의 첫 단계가 된다. 각 개별 변수들의 정보를 따로따로 탐색하는 것은 결국 하나의 변수로 이루어진 일변량(univariate) 자료를 탐색하는 것이라 할 수 있겠다. 일변량 자료가 갖고 있는 정보 중 우리가 가장 관심을 두는 사항은 어떤 분포 형태를 갖고 있는가 하는 점이 될 것이다. 주어진 자료의 분포 형태를 그래프로 잘 표현하려면, 자료의 속성에 가장 적합한 그래프를 선택해야 할 것이다. 일변량 자료의 분포는 대부분 그래프를 통해서 나타내게 되지만, 요약 통계량으로 표현하는 것이 더 편리한 경우도 있다.

본 장에서는 범주형 데이터와 연속형 데이터로 구분하여 각각의 경우에 사용할 수 있는 그래프의 작성방법을 살펴볼 것이며, 연속형 데이터의 요약통계량을 산출하는 방법에 대해서도 살펴볼 것이다. 그래프는 패키지 graphics와 ggplot2를 대상으로 살펴볼 것이며, 따라서 4장과 8장에서 소개된 내용이 자세한 추가 설명 없이 사용될 것이다. 패키지 ggplot2나 dplyr의 함수를 사용하기 위해서는 library(tidyverse)를 먼저 실행해야 한다는 것을 잊지 말도록 하자.

11.1 일변량 범주형 자료를 위한 그래프

범주형 자료란 데이터가 여러 범주들로만 이루어진 경우를 의미하는 것으로 명목형 자료와 순서형 자료로 구분된다. 명목형 자료란 범주들 간에 순서척도가 없는 경우로, 거주지, 성별, 종교 등이 여기에 해당되고, 순서형 자료란 강의 만족도와 같이 범주들을 순서에 따라 나열할 수 있는 경우를 지칭한다. 본 절에서는 막대 그래프(bar chart), 파이 그래프(pie chart)와 Cleveland의 점 그래프(dot chart)를 살펴볼 것인데, 이 그래프들은 명목형과 순서형 구분 없이 모두에게 잘 적용되는 그래프들이다.

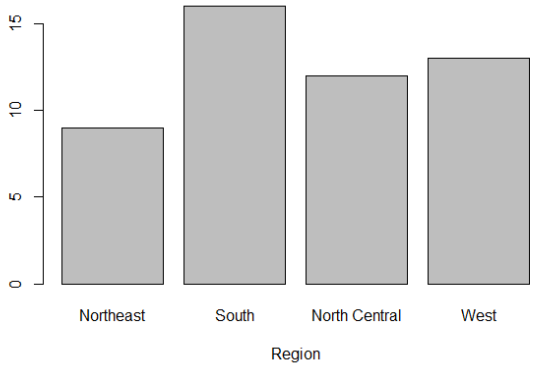
11.1.1 막대 그래프

범주형 자료에 대한 그래프로 가장 흔하게 볼 수 있는 그래프로서, 각 범주의 도수를 막대의 높이로 나타내는 그래프이다. 패키지 graphics에서 막대 그래프의 작성은 함수 barplot()과 plot()으로 할 수 있는데, 함수 barplot()에는 각 범주의 도수를 나타내는 도수분포표를 벡터로 입력해야 하며, 함수 plot()에는 요인을 입력해야 한다.

패키지 ggplot2에서 막대 그래프의 작성은 함수 geom_bar()를 사용하면 되며, 도수분포표를 자료로 사용하는 경우에는 geom_bar(stat="identity") 또는 함수 geom_col()을 사용하면 된다.

예제로 요인 state.region의 막대 그래프를 작성해 보자. 요인 state.region는 미국 50개 주를 4개의 지역 범주로 구분한 명목형 자료이다. 요인이므로 함수 plot()을 사용하면 간편하게 막대 그래프를 작성할 수 있다.

```
> # 그림 11.1
> plot(state.region, xlab="Region")
```



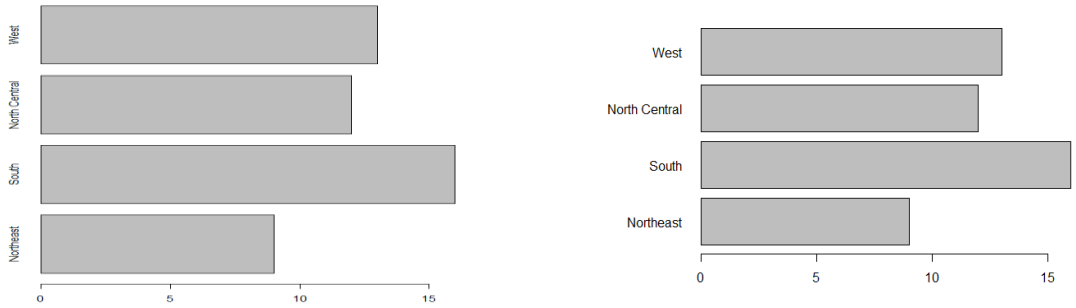
<그림 11.1> 요인 state.region의 막대 그래프

옆으로 누운 형태의 평행한 막대 그래프를 작성하고자 한다면 옵션 horiz=TRUE를 추가로 입력해야 한다. 이 경우 각 막대의 라벨 방향은 축의 방향과 평행하게 나타나는 것이 디폴트이므로 옆으로 누운 막대 그래프는 라벨 방향이 Y축과 평행하게 되어 밑에서 위 방향으로 작성된다. 만일 라벨 방향을 평행 즉, 왼쪽에서 오른쪽 방향으로 작성하고자 한다면 XY축의 라벨 방향을 결정하는 그래픽 모수인 las의 값을 조정해야 한다. 그래픽 모수 las는 디폴트인 0이면 축의 방향과 평행, 1이면 평행, 2이면 축의 방향과 수직, 3이면 수직으로 XY축의 라벨을 작성되도록 한다.

옆으로 누운 막대 그래프의 경우에 각 막대의 라벨을 평행하게 작성하려면 라벨이 작성되는 내부 마진 영역 중 왼쪽 내부 마진 영역의 크기를 충분히 확보해야 한다. 내부 마진을 조절하는 그래픽 모수인 mar의 디폴트 값은 c(5.1,4.1,4.1,2.1)인데, 왼쪽 영역의 크기인 4.1을 7.1로 확장시키고 막대 그래프를 작성해 보자. 그래픽 모수 mar의 변경은 함수 par()로 할 수 있는데, 일단 변경이 되면 변경된 값이 계속 유지되어 이 후 작업에도 적용된다. 따라서 변경 전 그래픽 모수의 디폴트 값을 저장하고, 해당 작업이 종료되면 원래 값으로 복위시키는 것이 필요하다.

```
> # 그림 11.2 (a)
> plot(state.region, horiz=TRUE)

> # 그림 11.2 (b)
> opar <- par(no.readonly=TRUE)
> par(mar=c(5.1,7.1,4.1,2.1))
> plot(state.region, horiz=TRUE, las=1)
```

```
> par(opar)
```

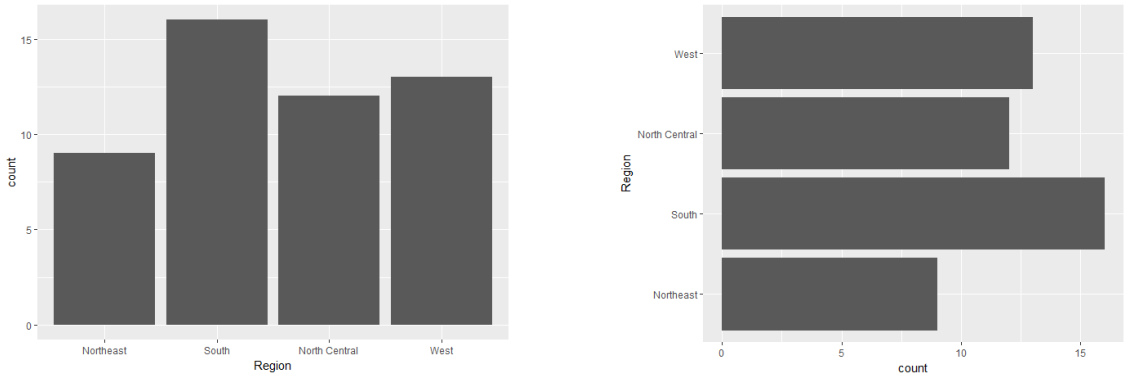


(a) (b)

<그림 11.2> 요인 state.region의 옆으로 누운 형태의 막대 그래프

패키지 ggplot2에서 막대 그래프를 작성하기 위해서는 함수 geom_bar()를 사용해야 한다. 또한 옆으로 누운 형태의 막대 그래프 작성은 함수 coord_flip()을 추가하면 된다.

```
> # 그림 11.3 (a)
> ggplot(data.frame(state.region)) +
    geom_bar(aes(x=state.region)) +
    labs(x="Region")

> # 그림 11.3 (b)
> ggplot(data.frame(state.region)) +
    geom_bar(aes(x=state.region)) +
    labs(x="Region") +
    coord_flip()
```



(a) (b)

<그림 11.3> 패키지 ggplot2에서 작성된 막대 그래프

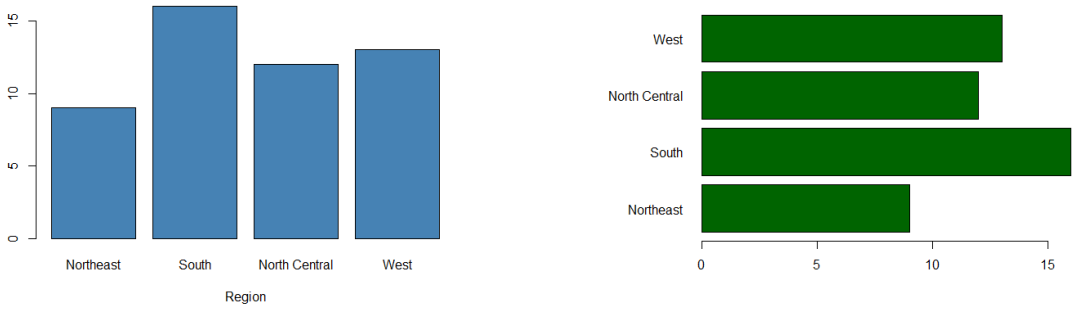
함수 barplot()으로 막대 그래프를 작성하기 위해서는 각 범주에 대한 도수를 먼저 구해야 하는데, 이 작업은 함수 table()로 다음과 같이 할 수 있다.

```
> counts <- table(state.region)
> counts
state.region
```

```
     Northeast          South  North Central           West
             9             16             12             13
```

범주형 자료 state.region은 Northeast, South, North Central, West의 4개 범주로 이루어졌고, 각 범주에 대한 도수가 9, 16, 12, 13임을 알 수 있다. 이제 객체 counts로 막대 그래프를 작성해보자.

```
> # 그림 11.4 (a)
> barplot(counts, xlab="Region", col="steelblue")

> # 그림 11.4 (b)
> opar <- par(no.readonly=TRUE)
> par(mar=c(5.1,7.1,4.1,2.1))
> barplot(counts, horiz=TRUE, las=1, col="dark green")
> par(opar)
```



(a) (b)

<그림 11.4> 함수 barplot()에 의한 막대 그래프

패키지 ggplot2에서는 자료가 반드시 데이터 프레임의 형태로 입력되어야 한다. 함수 table()로 생성된 객체 counts를 데이터 프레임으로 변경시켜 보자.

```
> df_1 <- as.data.frame(counts)
> df_1
   state.region Freq
1     Northeast    9
2         South   16
3 North Central   12
4          West   13
```

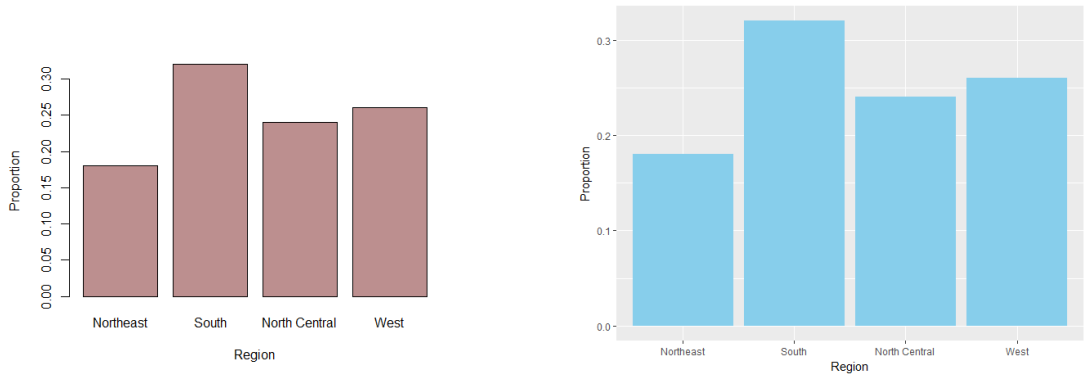
도수분포표를 자료로 입력한 경우에는 함수 geom_bar()에서 사용될 stat을 "identity"로 변경하거나 혹은 함수 geom_col()을 대신 사용해야 한다. 매핑은 데이터 프레임 df_1의 변수 state.region을 시각적 요소 x에 연결하고 변수 Freq를 시각적 요소 y에 연결해야 한다.

```
> # 그림 11.3 (a)
> ggplot(df_1, aes(x=state.region, y=Freq)) +
    geom_col() +
```

```
    labs(x="Region", y="")

> # 그림 11.3 (b)
> ggplot(df_1, aes(x=state.region, y=Freq)) +
    geom_col() +
    labs(x="Region", y="") +
    coord_flip()
```

많은 경우에 막대 그래프는 각 범주의 빈도수를 이용하여 작성되지만, 경우에 따라서는 각 범주의 상대 도수를 이용하여 작성해야 할 때도 있다. 함수 barplot()의 경우에는 상대 도수분포표를 자료로 입력하면 되는데, 상대 도수분포표의 작성은 함수 table()로 생성된 객체를 함수 prop.table()에 입력하면 된다. 함수 ggplot()의 경우에는 geom_bar()가 생성한 변수 ..prop..를 시각적 요소 y에 매핑하고, 시각적 요소 group에 임의의 값을 지정해 주면 된다.

```
> props <- prop.table(counts)
> props
state.region
    Northeast         South  North Central          West
         0.18          0.32           0.24          0.26
> # 그림 11.5 (a)
> barplot(props, xlab="Region", ylab="Proportion", col="rosybrown")

> # 그림 11.5 (b)
> ggplot(data.frame(state.region)) +
    geom_bar(aes(x=state.region, y=..prop.., group=1), fill="skyblue") +
    labs(x="Region", y="Proportion")
```



(a) (b)

<그림 11.5> 상대도수에 의한 막대 그래프 작성

11.1.2 파이 그래프

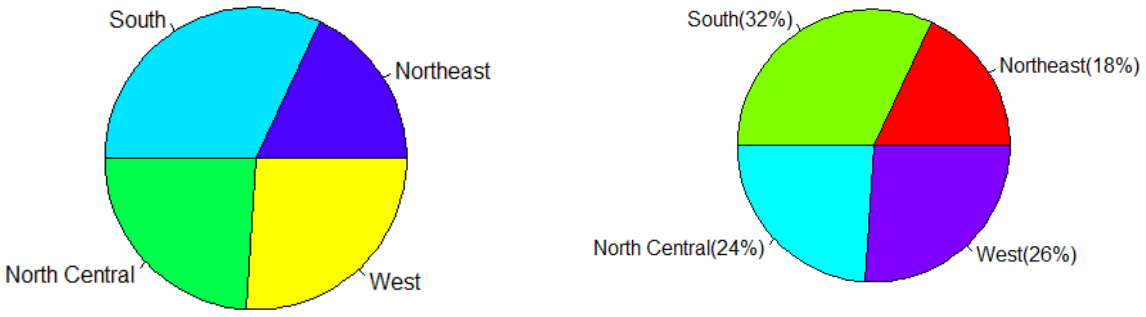
범주형 자료에 대한 그래프로 실생활에서 많이 사용되는 그래프 중 하나가 바로 파이 그래프일 것이다. 파이 그래프는 각 범주의 상대도수에 비례한 면적으로 원을 나누어 나타내는 그래프이다.

그러나 면적의 차이를 시각적으로 구분하는 것은 길이의 차이를 구분하는 것보다 훨씬 어렵기 때문에 대부분의 통계학자들은 파이 그래프의 가치를 막대 그래프 보다 낮게 평가하고 있다. 파이조각의 면적을 비교하는데 도움을 줄 수 있는 방안으로 각 파이조각에 면적의 백분율을 라벨로 추가해주는 것을 생각해 볼 수 있다.

패키지 graphics에서 파이 그래프를 작성하는 함수는 pie()이며 일반적인 사용법은 pie(x, labels=names(x), ...)와 같다. 변수 x는 파이 각 조각의 면적, 즉 도수를 나타내는 숫자형 벡터이며, 각 조각의 라벨은 x의 이름이 되는 것이 디폴트이며, labels에 문자형 벡터로 따로 지정할 수 있다.

예제 데이터로는 막대 그래프 작성에서 살펴본 state.region을 다시 사용해보자. 기본적인 형태의 파이 그래프는 다음과 같이 작성된다.

```
> # 그림 11.6 (a)
> counts <- table(state.region)
> pie(counts, col=topo.colors(length(counts)))
```



(a) (b)

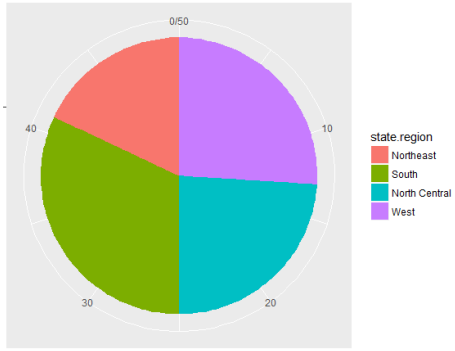
<그림 11.6> 요인 state.region의 파이 그래프

객체 counts의 각 숫자에는 요인 state.region의 해당되는 level이 이름으로 지정되어 있어서 따로 labels를 지정할 필요가 없다. 그러나 각 파이 조각에 면적의 백분율을 라벨로 추가하려면 적절한 문자형 벡터를 미리 생성하여 labels에 지정해야 한다. 각 조각의 백분율은 함수 prop.table()로 얻어진 상대도수에 100을 곱하여 얻을 수 있다.

```
> pct <- prop.table(counts)*100
> region <- paste0(names(pct), "(", pct, "%)")
> region
[1] "Northeast(18%)"     "South(32%)"         "North Central(24%)"
[4] "West(26%)"

> # 그림 11.6 (b)
> pie(counts, labels=region, col=rainbow(length(pct)))
```

패키지 ggplot2에서 파이 그래프를 작성하기 위해서는 함수 geom_bar()에 시각적 요소 fill을 사용하여 쌓아 올린 막대 그래프를 작성하고, 함수 coord_polar()에서 theta에 "y"를 연결하여 상대도수에 비례한 각도를 계산해야 한다.

```
> # 그림 11.7
> ggplot(data.frame(state.region)) +
    geom_bar(aes(x="", fill=state.region), width=1) +
    labs(x="", y="") +
    coord_polar(theta="y")
```



<그림 11.7> 패키지 ggplot2에서 작성된 파이 그래프

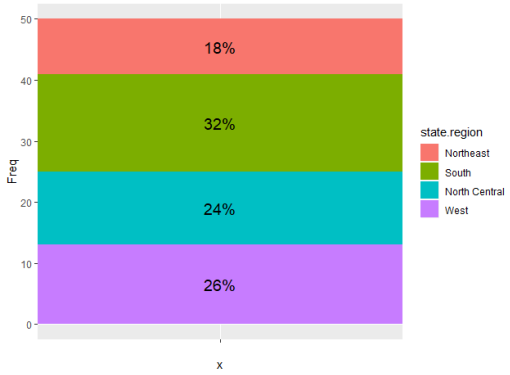
패키지 ggplot2에서 각 파이 조각의 백분율을 라벨로 추가해 보자. 이 경우에 라벨로 사용할 각 조각의 백분율을 데이터 프레임에 변수로 포함시키는 것이 필요하다. 숫자를 '%' 기호가 포함된 백분율로 변환시킬 때 편하게 사용할 수 있는 함수가 패키지 scales에 있는 percent()이다. 함수 table()로 생성된 객체를 데이터 프레임으로 변환시키면서, 각 조각의 백분율을 문자형 변수로 추가해서 df_2를 만들어 보자.

```
> library(scales)

> counts <- table(state.region)
> df_2 <- as.data.frame(counts) %>%
    mutate(pct=percent(Freq/sum(Freq)))

> df_2
   state.region Freq pct
1     Northeast    9 18%
2         South   16 32%
3 North Central   12 24%
4          West   13 26%
```

이제 파이 그래프를 작성하기 바로 전 단계인 쌓아 올린 막대 그래프를 작성해 보자. 그림 11.8에서 볼 수 있듯이 쌓아 올려지는 조각의 순서는 데이터 프레임 df_2의 첫 번째 행이 제일 위에 위치하고 두 번째 행이 그 밑으로 배치되는 위에서 아래 방향이 된다. 그래프에 라벨을 추가할 때 사용되는 함수는 geom_text()이다. 데이터 프레임 df_2의 변수 pct를 시각적 요소

label에 매핑하고, 라벨의 위치를 position에 지정하면 된다. 라벨의 위치는 쌓아 올려진 각 조각의 중간 부분이 적절할 것으로 보이는데, 함수 position_stack()에서 변수 vjust에 0.5를 입력하면 된다.

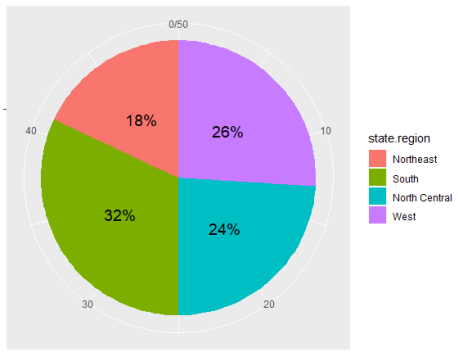
```
> # 그림 11.8
> bar <- df_2 %>%
    ggplot(aes(x="", y=Freq, fill=state.region)) +
    geom_col(width=1) +
    geom_text(aes(label=pct), size=5,
              position=position_stack(vjust=0.5))
> bar
```



<그림 11.8> 쌓아 올린 막대 그래프에 백분율 라벨 추가하기

각 파이 조각에 백분율을 라벨로 추가한 파이 그래프 작성은 객체 bar를 이용하여 다음과 같이 작성할 수 있다.

```
> # 그림 11.9
> bar + coord_polar("y") +  labs(x="", y="")
```

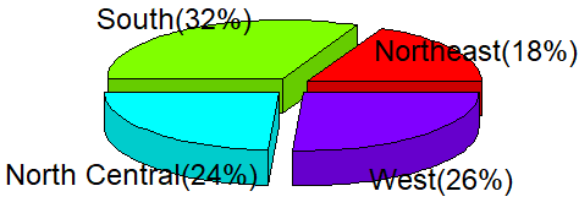


<그림 11.9> 패키지 ggplot2에서 작성된 파이 그래프

신문이나 잡지 등에 작성된 파이 그래프 중에는 3차원 효과를 준 파이 그래프도 종종 발견된다. 3D 파이 그래프는 패키지 plotrix의 함수 pie3D()로 작성할 수 있다. 3차원 파이 그래프도

파이 그래프가 갖고 있는 근본적인 문제에서 자유로울 수는 없으며, 따라서 비록 시각적으로는 화려한 그래프이지만 정보의 효과적인 전달이라는 측면에서는 문제가 많은 그래프라고 할 수 있다.

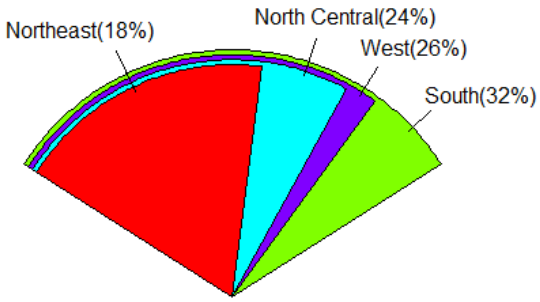
```
> counts <- table(state.region)
> pct <- prop.table(counts)*100
> region <- paste0(names(pct), "(", pct, "%)")

> # 그림 11.10
> library(plotrix)
> pie3D(counts, labels=region, explode=0.1)
```



<그림 11.10> 패키지 plotrix의 3차원 파이 그래프

파이 그래프의 가장 큰 문제는 파이조각들의 면적을 서로 비교하는 것이 매우 힘들다는 점이다. 이러한 문제를 해결하는 방안으로 제시된 그래프가 Lemon and Tyagi (2009)가 제안한 fan plot이다. R에서는 패키지 plotrix에 함수 fan.plot()으로 작성할 수 있다.

```
> # 그림 11.11
> library(plotrix)
> fan.plot(counts, labels=region)
```



<그림 11.11> 패키지 plotrix의 Fan plot

Fan plot에서는 조각들이 크기 순으로 서로 겹치도록 재배치되는데, 조각의 반지름을 조절하여 모든 조각을 볼 수 있도록 하였다. 그림 11.11에서 우리는 조각의 크기가 South > West > North Central > Northeast의 순서로 되어 있음을 쉽게 알 수 있다. 많이 사용되고 있지는 않지만 파이 그래프보다는 훨씬 효과적인 그래프라고 할 수 있다.

11.1.3 Cleveland의 점 그래프

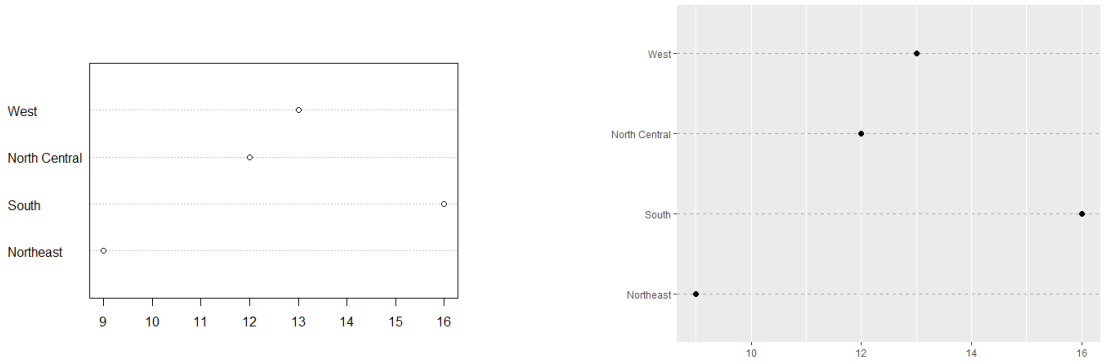
점 그래프는 통계 그래픽스 분야에 큰 기여를 한 Cleveland가 개발한 그래프이다. 비록 막대 그래프나 파이 그래프처럼 화려하게 치장할 수는 없으나 범주형 데이터의 속성을 정확하게 표시할 수 있는 이상적인 그래프로 인정받고 있다. 패키지 graphics에서 점 그래프는 함수 dotchart()로 작성되며, 일반적인 사용법은 dotchart(x, labels, ...)가 된다. 변수 x는 각 범주의 도수를 나타내는 숫자형 벡터이고 labels는 x의 각 점에 대한 라벨을 제공하는 문자형 벡터이다. 패키지 ggplot2에서는 함수 geom_point()로 점 그래프를 작성할 수 있다.

예제 데이터로는 앞 절에서 살펴본 state.region을 다시 사용해보자. 함수 table()로 작성된 객체는 숫자형 벡터가 아니다. 이것을 함수 dotchart()에 바로 입력하게 되면 그래프는 작성이 되지만, 입력된 자료가 숫자형 벡터가 아니어서 함수 as.numeric()을 사용했다는 경고 문구가 뜬다. 따라서 함수 as.numeric()으로 유형을 변환시키고 입력하는 것이 경고 없이 그래프를 작성하는 방법이 된다.

```
> # 그림 11.12(a)
> counts <- table(state.region)
> dotchart(as.numeric(counts), labels=names(counts))
```

패키지 ggplot2에서 점 그래프 작성은 도수분포표 자료를 함수 geom_point()에 적용시키면 된다. 함수 theme()은 그래프의 디폴트 형태를 변경하고자 할 때 사용되는 것으로, 이 경우에는 그래프에 수평선을 추가하기 위해 사용되었다. 자세한 내용은 Wickham(2015)를 참고하기 바란다.

```
> # 그림 11.12 (b)
> as.data.frame(counts) %>%
    ggplot(aes(x=Freq,y=state.region)) +
    geom_point(size=2) +
    theme(panel.grid.major.y=element_line(linetype=2,
          color="darkgray")) +
    labs(x="",y="")
```



(a) (b)

<그림 11.12> 요인 state.region의 점 그래프

11.2 일변량 연속형 자료를 위한 그래프

일변량 연속형 자료에 대해서 우리가 가장 관심을 갖고 있는 정보는 자료의 분포 형태일 것이다. 자료의 중심은 어디인가? 자료의 퍼짐 정도는 어떠한가? 이러한 질문에 효과적인 답을 얻기 위해서는 적절한 그래프를 잘 선택해야 할 것이다.

11.2.1 줄기-잎 그림

줄기-잎 그림은 비교적 소규모 자료의 분포를 나타내는데 적합한 그래프라고 할 수 있다. 자료를 '줄기'와 '잎'으로 구분하고, '줄기'를 수직으로 세운 후에 해당되는 '줄기'에 '잎'들을 크기 순서로 붙여 나타내는 그림이다. 줄기-잎 그림은 함수 stem()으로 작성할 수 있으며, 일반적인 사용법은 stem(x, scale=1)이다. 변수 x는 숫자형 벡터이며, scale은 줄기-잎 그림의 길이를 조절하는 것으로 scale=2가 되면 대략 2배 길이의 그래프가 그려진다. 그래프의 길이를 늘이는 방법은 대부분 '줄기'를 더 세분화시키는 방식으로 이루어진다.

예제 데이터로는 데이터 프레임 women을 사용해보자. 여기에는 30세에서 39세 사이의 미국 여성들의 키와 몸무게 데이터가 height와 weight로 입력되어 있다. 두 변수의 줄기-잎 그림을 그려보자. 줄기-잎 그림은 Console 창에 결과가 나타난다.

```
> with(women, stem(height))

  The decimal point is 1 digit(s) to the right of the |

  5 | 89
  6 | 01234
  6 | 56789
  7 | 012

> with(women, stem(weight))

  The decimal point is 1 digit(s) to the right of the |

  11 | 57
  12 | 0369
  13 | 259
  14 | 26
  15 | 049
  16 | 4
```

옵션 scale을 이용하여 그래프의 길이를 늘여야 하는 경우에 대한 예제를 살펴보자.

```
> x <- c(98,102,114,122,132,144,106,117,151,118,124,115)
> stem(x)

  The decimal point is 1 digit(s) to the right of the |
```

```
 8 | 8
10 | 264578
12 | 242
14 | 41
```

자료의 개수에 비해 '줄기'가 지나치게 많은 경우로, 디폴트 scale 값에서는 '줄기'가 통합되어 실제로는 없는 줄기가 나타났다. 이것을 scale=2로 조정하여 작성해 보자.

```
> stem(x, scale=2)

  The decimal point is 1 digit(s) to the right of the |

 9 | 8
10 | 26
11 | 4578
12 | 24
13 | 2
14 | 4
15 | 1
```

11.2.2 상자그림

상자그림은 box plot 혹은 box-and-whiskers plot이라고 불리는 그래프로 John Tukey에 의하여 개발되었다. 5개의 요약 통계량인 최소값, 25% 백분위수(Q_1), 중앙값, 75% 백분위수(Q_3), 최대값을 이용하여 작성하는 그래프이다. 단순한 형태의 그래프이나, 분포의 중심, 퍼짐 정도, 치우침 정도(skewness), 꼬리의 길이 등이 상당히 명확하게 나타난다.

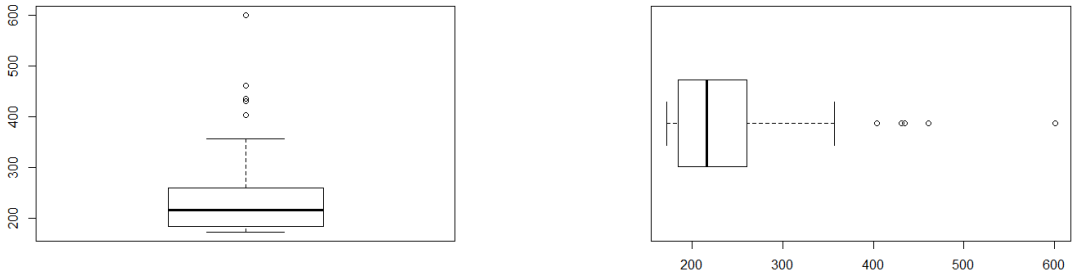
상자그림의 작성에서 25% 백분위수, 중앙값, 그리고 75% 백분위수로 표현되는 상자로부터 양쪽 꼬리 방향으로 그려지는 whisker의 길이에 대한 두 가지 대안이 있는데, 첫 번째 방법은 최소값과 최대값까지 단순하게 연결하는 것이다. 두 번째 방법은 상자의 길이(IQR)의 1.5배가 넘지 않는 관찰값까지 연결하고, 그 범위를 초과하는 관찰값들은 따로 점으로 표시하는 방법이다. 첫 번째 방법은 꼬리 부분에 있을 수 있는 이상값에 대한 정보를 전혀 표현할 수 없기 때문에 최근에는 거의 사용되지 않는 방법이라 하겠다. 따라서 두 번째 방법으로만 상자그림을 작성하겠다.

패키지 graphics에서 상자그림의 작성은 함수 boxplot()으로 작성할 수 있으며, 기본적인 사용법은 boxplot(x, horizontal=FALSE, ...)가 된다. 변수 x는 숫자형 벡터이고 옵션 horizontal은 상자그림의 방향을 정하는 것으로 디폴트는 수직방향이 된다.

예제 데이터로는 패키지 UsingR에 있는 데이터 프레임 alltime.movies를 사용해보자. 데이터 프레임 alltime.movies에는 2003년까지 미국에서 상영된 영화 중 총수입이 가장 많았던 79개 영화의 총수입(Gross)과 처음 상영된 년도(Release.Year)가 변수로 있고, 영화제목은 행 이름으로 입력되어 있다. 총수입의 상자그림을 작성해보자.

```
> library(UsingR)
```

```
> attach(alltime.movies)
> boxplot(Gross)                          # 그림 11.13 (a)
> boxplot(Gross, horizontal=TRUE)         # 그림 11.13 (b)
> detach(alltime.movies)
```



(a) (b)

<그림 11.13> 데이터 프레임 alltime.movies의 변수 Gross의 상자그림

그림 11.13에 있는 상자그림과 같이 점으로 표시된 이상값이 나타나게 되면, 어떤 케이스에 해당되는 것인지 확인하는 것은 매우 중요한 작업이 될 것이다. 함수 boxplot()과 같은 그래프 함수를 실행시키면 해당되는 그래프를 작성하는 것 외에도 몇 가지 통계량을 계산하여 출력한다. 출력되는 결과가 자동으로 Console 창에 나타나지 않기 때문에 자연스럽게 인식하지는 못하지만 다음과 같이 객체에 할당을 하면 찾아볼 수 있게 된다. 옵션 plot=FALSE를 추가해서 그래프는 작성되지 않는다.

```
> my_box <- boxplot(alltime.movies$Gross, plot=FALSE)
> names(my_box)
[1] "stats" "n"     "conf"  "out"   "group" "names"
```

객체 my_box는 리스트이다. 여섯 개의 구성요소 중 my_box$stats와 my_box$out의 내용을 살펴보자. 객체 my_box$stats에 저장되어 있는 내용은 다음과 같다.

```
> my_box$stats
     [,1]
[1,]  172
[2,]  184
[3,]  216
[4,]  260
[5,]  357
```

이 숫자들은 그림 11.13의 상자그림에서 아래쪽 whisker의 끝점, 상자의 하단(25% 백분위수), 중앙값, 상자의 상단(75% 백분위수), 위쪽 whisker의 끝점을 각각 나타내고 있다. 또한 my_box$out에는 whisker를 벗어나 점으로 표시된 변수 Gross 값이 저장되어 있다.

```
> my_box$out
[1] 601 461 435 431 404
```

이 관찰값들이 어떤 케이스, 즉 어떤 영화의 총수입에 해당되는지를 알아보자. 우선 영화제목이 데이터 프레임 alltime.movies의 rownames로 입력되어 있는데, 이것을 데이터 프레임에 변수로 포함시켜 보자.

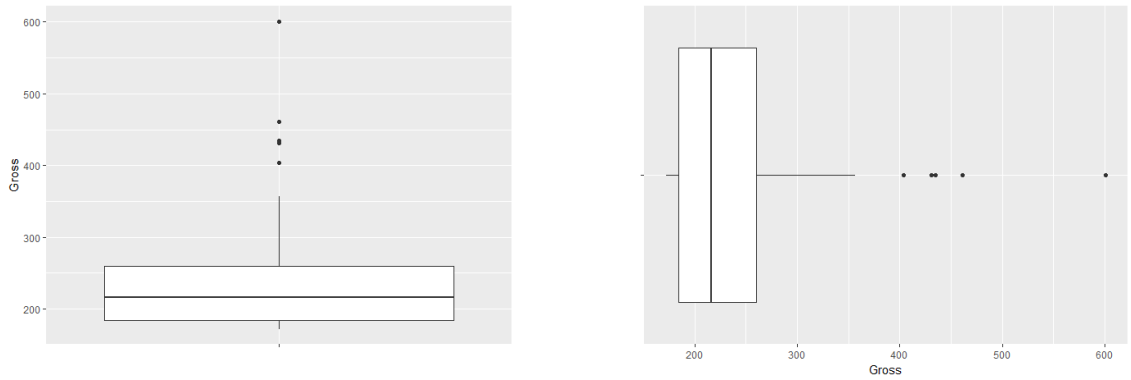
```
> alltime <- as_tibble(alltime.movies) %>%
    rownames_to_column(var="Movie.Title")
> print(alltime, n=3)
# A tibble: 79 x 3
  Movie.Title                          Gross Release.Year
  <chr>                                <dbl>        <dbl>
1 "Titanic                            "  601         1997
2 "Star Wars                          "  461         1977
3 "E.T.                               "  435         1982
# ... with 76 more rows
```

함수 as_tibble()과 rownames_to_column()은 패키지 tibble의 함수이며, 그 의미는 6장에서 살펴보았다. 이제 데이터 프레임 alltime에서 변수 Gross의 값이 my_box$out의 값과 같은 케이스를 선택하여 출력해 보자. 작업에 사용된 함수 filter()는 패키지 dplyr의 함수로써, 사용법은 7장에서 살펴보았다.

```
> top_movies <- alltime %>%
    filter(Gross %in% my_box$out)
> top_movies
# A tibble: 5 x 3
  Movie.Title                          Gross Release.Year
  <chr>                                <dbl>        <dbl>
1 "Titanic                            "  601         1997
2 "Star Wars                          "  461         1977
3 "E.T.                               "  435         1982
4 "Star Wars: The Phantom Menace      "  431         1999
5 "Spider-Man                         "  404         2002
```

패키지 ggplot2에서 상자그림을 작성하기 위해서는 함수 geom_boxplot()을 사용해야 한다. 한 변수의 상자그림을 작성하는 경우에도 시각적 요소 x와 y가 모두 필요한데, x에는 하나의 값만을 갖는 문자나 요인을 연결하고, y에 연속형 변수를 매핑하면 된다. 상자그림을 옆으로 누운 방향으로 작성하기 위해서는 함수 coord_flip()을 추가하면 된다.

```
> bp <- ggplot(alltime.movies, aes(x="", y=Gross)) +
    geom_boxplot() +
    labs(x="")
> bp                    # 그림 11.14 (a)
> bp + coord_flip()     # 그림 11.14 (b)
```

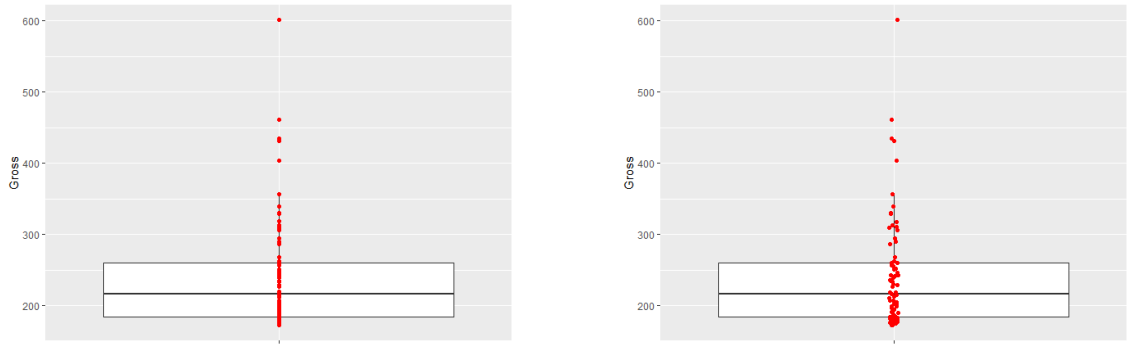


(a) (b)

<그림 11.14> 패키지 ggplot2에서 작성된 상자그림

상자그림에 자료의 위치를 점으로 함께 나타내 보자. 이 작업은 함수 geom_point()를 추가하면 되는데, 한 가지 주의할 점은 함수 geom_boxplot()에서 이상값을 점으로 표시하지 않도록 하는 것이 좋다는 것이다. 이상값이 나타나지 않도록 하기 위해서는 옵션 outlier.shape=NA 또는 ourlier.color=NA 또는 outlier.size=-1을 함수 geom_boxplot()에 포함시키면 된다.

```
> # 그림 11.15 (a)
> ggplot(alltime.movies, aes(x="", y=Gross)) +
    geom_boxplot(outlier.shape=NA) +
    geom_point(color="red") +
    labs(x="")
```



(a) (b)

<그림 11.15> 상자그림에 자료의 위치 추가

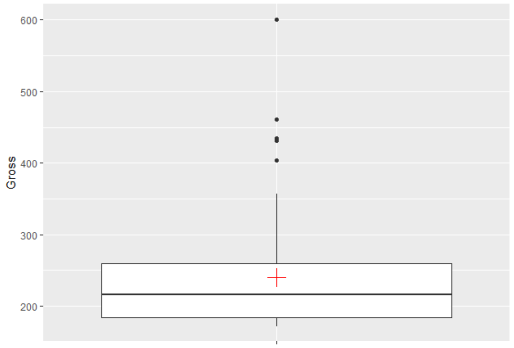
상자그림의 상자 안에 많은 점들이 서로 겹쳐져 있는 것을 볼 수 있다. 이런 경우에 점들의 위치에 약간 난수를 더해 조정해 주면 점들이 서로 겹쳐지는 문제를 어느 정도 해결할 수 있는데, 함수 geom_jitter()로 할 수 있는 작업이다.

```
> # 그림 11.15 (b)
> ggplot(alltime.movies, aes(x="", y=Gross)) +
    geom_boxplot(outlier.shape=NA) +
    geom_jitter(color="red", width=0.01) +
```

```
    labs(x="")
```

상자그림은 자료의 분위수를 기반으로 작성된 그래프이다. 따라서 중앙값의 위치는 표시가 되지만 평균값의 위치는 나타나지 않는 것이 일반적인 모습이다. 하지만 평균값의 위치를 추가적으로 표시하는 것이 자료의 분포를 이해하는데 도움이 될 수도 있을 것이다. 자료의 요약 통계량을 계산해서 그래프에 표시하는 작업은 함수 stat_summary()를 사용하면 할 수 있다. 이 함수는 하나의 x값에 대하여 주어진 y값의 요약 통계량을 계산하는 것으로 상자그림의 경우에 유용하게 사용할 수 있다. 원하는 요약 통계량은 변수 fun.y에 지정하고, 원하는 그래프 형태를 변수 geom에 지정하면 된다. 평균값이 빨간 십자 형태로 상자그림에 표시되었음을 알 수 있다.

```
> # 그림 11.16
> ggplot(alltime.movies,aes(x="", y=Gross)) +
    geom_boxplot() +
    stat_summary(fun.y="mean", geom="point",
                 color="red", pch=3, size=5) +
    labs(x="")
```

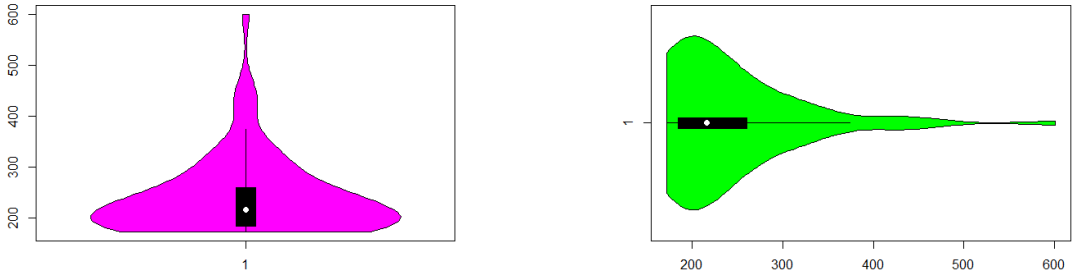


<그림 11.16> 상자그림에 평균값 표시 추가

- Violin plot

상자그림에 부가적인 정보를 추가하여 일종의 변종 상자그림이라고 할 수 있는 그래프로 violin plot이 있다. 이 그래프는 상자그림과 확률밀도함수 그래프의 조합이라고 할 수 있는데, 확률밀도함수 그래프는 11.2.4절에서 찾아볼 수 있다. Violin plot의 작성은 패키지 vioplot의 함수 vioplot()으로 할 수 있으며, 기본적인 사용법은 함수 boxplot()과 동일하다.

```
> library(vioplot)
> attach(alltime.movies)
> vioplot(Gross)                                  # 그림 11.17 (a)
> vioplot(Gross, horizontal=TRUE, col="green")    # 그림 11.17 (b)
> detach(alltime.movies)
```



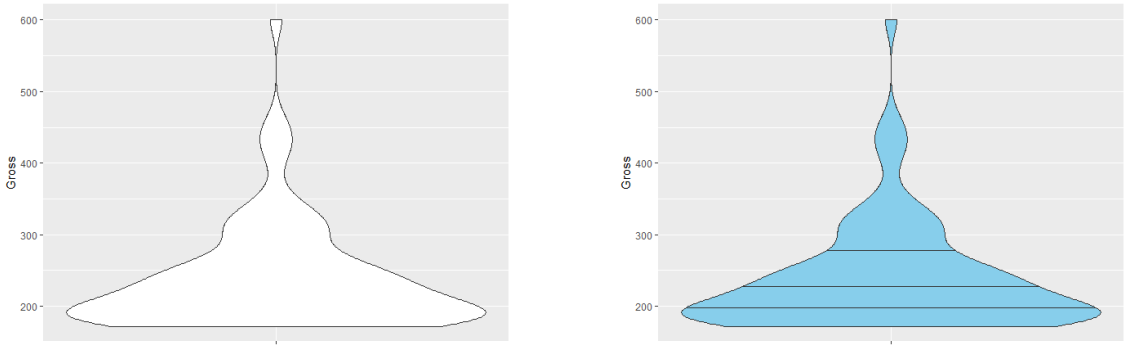
(a) (b)

<그림 11.17> 패키지 `vioplot`의 Violin plot

그림 11.17에서 볼 수 있듯이 violin plot은 상자그림을 중심으로 확률밀도함수 그래프를 대칭으로 그려놓은 그래프이다. 하얀 점은 중앙값을 나타내고 있으며, 검은 상자는 25% 백분위수에서 75% 백분위수까지 그려져 있고, whisker는 검은 색 실선으로 그려져 있다. 상자그림 외부의 그림은 확률밀도함수 그래프가 그려져 있어서 시각적으로 조금 더 분명하게 분포 형태에 대한 정보를 제공해주고 있다. 많이 사용되고 있지는 않지만 상당히 효과적으로 사용할 수 있는 그래프라고 할 것이다.

패키지 `ggplot2`에서 violin plot의 작성은 함수 `geom_violin()`으로 할 수 있다. 이 함수가 작성하는 기본적인 그래프에는 상자그림 없이 확률밀도함수 그래프만이 대칭으로 작성되어 있는 형태이다.

```
> # 그림 11.18 (a)
> vio <- ggplot(data=alltime.movies, aes(x="", y=Gross)) +
    labs(x="")
> vio + geom_violin()
```



(a) (b)

<그림 11.18> 패키지 `ggplot2`에서 작성된 violin plot 1

만일 분위수를 함께 표시하고자 한다면 변수 `draw_quantiles`에 표시를 원하는 분위수를 지정해야 한다.

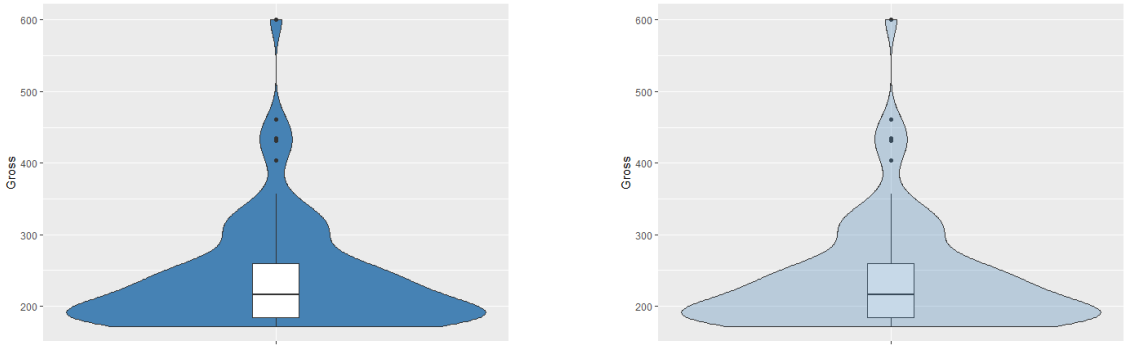
```
> # 그림 11.18 (b)
> vio + geom_violin(draw_quantiles=c(0.25,0.5,0.75),
                    fill="skyblue")
```

상자그림과 함께 violin plot을 작성하고자 한다면 함수 geom_boxplot()을 함께 사용하면 된다.

```
> # 그림 11.19 (a)
> vio + geom_violin(fill="steelblue") +
    geom_boxplot(width=0.1)
```

함수 geom_violin()을 함수 geom_boxplot()의 뒤에 추가한다면 투명도를 조절하는 시각적 요소인 alpha의 값을 낮게 지정하여 투명도를 높여야 상자그림이 보이게 된다.

```
> # 그림 11.19 (b)
> vio + geom_boxplot(width=0.1) +
    geom_violin(fill="steelblue", alpha=0.3)
```



(a) (b)

<그림 11.19> 패키지 ggplot2에서 작성된 violin plot 2

11.2.3 히스토그램

히스토그램은 연속형 자료의 분포를 시각화하는데 가장 많이 사용되는 그래프다. 작성방법은 자료의 전 범위를 서로 겹치지 않는 구간(bin)으로 구분하고, 각 구간에 속한 자료의 빈도를 구한다. 이어서 각각의 구간을 X축으로, 그 구간에 대한 빈도 수를 나타내는 값을 Y축으로 하여 막대를 그리면 된다. 패키지 graphics에서 히스토그램의 작성은 함수 hist()로 할 수 있으며, 기본적인 사용법은 hist(x, breaks="Sturges", freq, probability=!freq, ...)과 같다. x는 숫자형 벡터이고, breaks는 구간 설정 방법에 관련된 것으로 다음 중 하나의 형태를 선택해야 한다.

1) 각 구간을 나누는 점들로 이루어진 벡터. 예: breaks=seq(35,105,length=10)

2) 구간의 개수를 표시하는 하나의 숫자. 예: breaks=10

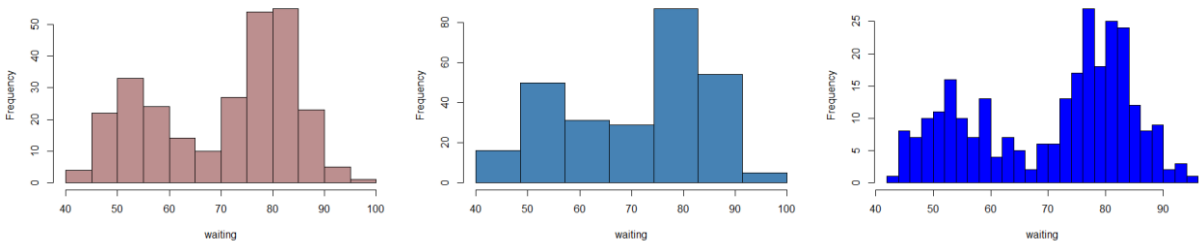
3) 구간의 개수를 계산하는 알고리즘을 칭하는 문자열. 디폴트는 "Sturges"이고 다른 선택 가능한 문자열은 "Scott"과 "FD"가 있다.

옵션 freq에 TRUE를 지정하면 히스토그램의 높이가 빈도로 표현되고, FALSE가 되면 확률밀도로 표현되어 히스토그램의 전체 면적이 1로 조정된다. 디폴트 값은 모든 구간의 폭이 동일하고 옵션 probability가 지정되어 있지 않으면 TRUE가 된다. 옵션 probability는 freq의 반대 값이 지정되는 것으로 두 옵션 중 하나만 사용하면 된다. 즉, 확률밀도로 히스토그램을 작성하고자 한다면 freq = FALSE 혹은 probability = TRUE 중 하나만 지정한다.

패키지 ggplot2에서는 함수 geom_histogram()으로 히스토그램을 작성할 수 있다. 구간 설정은 구간의 개수를 bins에 지정하거나, 구간의 폭을 binwidth에 지정하면 된다. 디폴트는 bins=30으로 히스토그램을 작성한다.

예제 데이터로 데이터 프레임 faithful을 사용해보자. 이 데이터 프레임에는 미국 Yellowstone 국립공원에 있는 Old Faithful이라는 간헐천의 분출지속시간(eruptions)과 분출간격(waiting)이 변수로 있다. 이 중 분출간격에 대한 분포를 히스토그램으로 나타내보자. 히스토그램의 형태는 구간을 어떻게 설정하는가에 따라 크게 변하게 된다.

```
> attach(faithful)
> # 그림 11.20 (a)
> hist(waiting, col="rosybrown", main="")

> # 그림 11.20 (b)
> hist(waiting, breaks=seq(40,100,length=8),
       col="steelblue", main="")

> # 그림 11.20 (c)
> hist(waiting, breaks=20, col="blue", main="")
> detach(faithful)
```



(a) (b) (c)

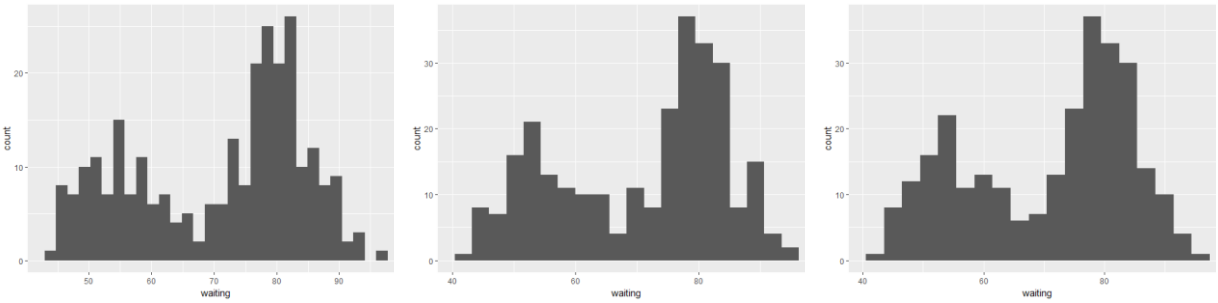
<그림 11.20> 구간 설정에 따른 히스토그램 형태의 변화

그림 11.20에서 볼 수 있듯이 구간을 어떻게 설정하는가에 따라 히스토그램의 형태는 크게 변하게 된다. 디폴트 구간으로 작성된 히스토그램이 만족스러운 결과를 보여주는 경우도 많지만

전적으로 의존하기에는 약간 아쉬운 점이 있다. 이렇듯 구간설정 문제는 히스토그램의 중요한 문제라 할 수 있으며, 적절한 구간설정에 관한 이론적 접근에 대해서는 Scott(1992)을 참조하기 바란다.

패키지 ggplot2에서 히스토그램의 작성은 다음과 같이 할 수 있다.

```
> h <- ggplot(faithful, aes(x=waiting))

> # 그림 11.21 (a)
> h + geom_histogram()
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

> # 그림 11.21 (b)
> h + geom_histogram(bins=20)

> # 그림 11.21 (c)
> h + geom_histogram(binwidth=3)
```



(a) (b) (c)

<그림 11.21> 패키지 ggplot2에서 작성한 히스토그램

히스토그램의 또 한가지 문제는 연속형 데이터의 분포를 계단 형태로 표현함으로 해서 생기는 어색함이라고 할 수 있다. 이것은 히스토그램의 태생적인 한계라고 할 수 있는데, 이 문제는 다음 절에서 살펴볼 확률밀도함수 그래프를 히스토그램에 겹쳐서 나타내거나, 도수분포다각형을 작성하는 것으로 어느 정도 완화할 수 있다.

11.2.4 확률밀도함수 그래프

연속형 자료의 분포형태를 알아보기 위한 그래프로서 지금까지 줄기-잎 그림, 상자그림, 히스토그램을 살펴보았다. 이 그래프들은 모두 나름의 장점이 있는 좋은 그래프들이다. 그러나 줄기-잎 그림은 대규모의 데이터에는 적합하지 않다는 단점이 있고, 상자그림은 사분위수를 기본으로 하여 나타낸 그래프이기 때문에 분포의 세밀한 특징이 잘 나타나지 않을 수 있다. 또한 연속형 자료의 분포는 매끄러운 곡선의 형태를 지니고 있으나 히스토그램은 항상 계단함수의 형태를 보일 수 밖에 없다는 한계가 있다.

대규모의 데이터에도 쉽게 적용되며 분포의 세밀한 특징도 잘 나타내고, 또한 매끄러운 곡선으로 분포의 형태를 나타낼 수 있는 방법이 바로 확률밀도함수를 추정하여 그래프로 나타내는 것이다. 분포 형태를 매우 효과적으로 나타낼 수 있기 때문에 최근 많이 사용되고 있는 방법으로서 이론적인 내용에 대해서는 Simonoff(1996)나 Wand and Jones (1995) 등을 참조하기 바란다.

패키지 graphics에서 확률밀도함수 그래프를 작성하는 방법은 먼저 함수 density()로 확률밀도함수를 추정하고, 그 추정 결과를 함수 plot()을 사용하여 그래프로 나타내는 것이다. 확률밀도함수 추정에 사용되는 함수 density()의 기본적인 사용법은 density(x, ...)가 되며, 데이터 x 외에도 추정 결과에 큰 영향을 미치는 몇몇 옵션이 있으나, 이 옵션들에 대한 설명은 모두 생략하겠다. 데이터 프레임 faithful의 변수 waiting의 밀도함수를 추정해보자.

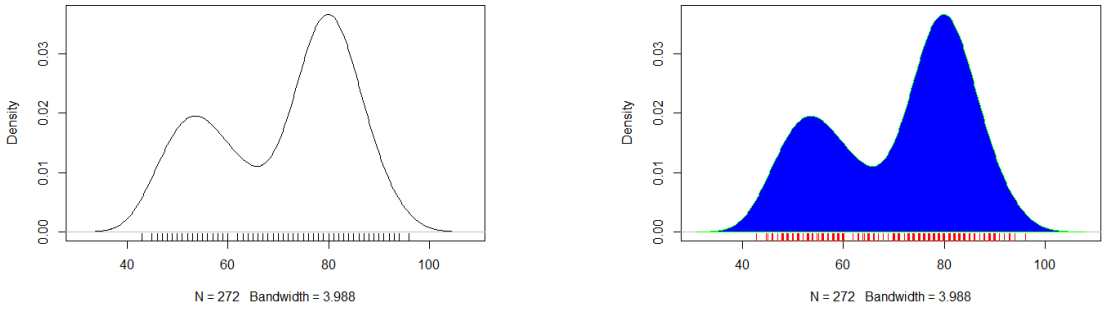
```
> fit <- density(faithful$waiting)
> names(fit)
[1] "x"         "y"         "bw"        "n"         "call"
[6] "data.name" "has.na"
```

객체 fit은 추정 결과가 할당된 리스트이다. 이 중 fit$x는 확률밀도함수가 추정된 X축 좌표이고, fit$y는 확률밀도함수의 추정 결과가 된다. 이들 두 요소는 확률밀도함수 그래프의 주요 요소가 된다. 함수 plot()으로 확률밀도함수 그래프를 작성해보자.

```
> # 그림 11.22 (a)
> plot(fit, main="")
> rug(faithful$waiting)

> # 그림 11.22 (b)
> plot(fit, main="")
> polygon(fit, col="blue", border="green")
> rug(jitter(faithful$waiting), col="red")
```

포괄적 그래프 함수인 plot()에 density()에 의해 생성된 객체가 입력되면 확률밀도함수 결과에 적합한 그래프가 작성된다. 따라서 명령문 plot(fit)을 실행시키면 함수 plot()은 객체 fit에 있는 요소 중 fit$x와 fit$y를 사용하여 그림 11.22 (a)를 작성한다. 그래프의 X축과 Y축에는 디폴트 라벨이 추가되는데, 특히 X축 라벨은 데이터의 크기와 확률밀도함수의 중요한 모수인 bandwidth의 값이 출력된다.

확률밀도함수 그래프를 색으로 치장하고자 한다면, 낮은-수준의 그래프 함수 polygon()을 사용하면 된다. 기본적인 사용법은 polygon(x, y, ...)으로 벡터 x와 y가 제공하는 꼭지점 좌표를 차례로 연결하여 기존의 그래프에 다각형을 추가하는 기능을 갖고 있다. 또한 옵션 col을 추가하면 다각형 내부를 색으로 채울 수 있고, border는 외곽선의 색을 지정한다.



(a) (b)

<그림 11.22> 확률밀도함수 그래프

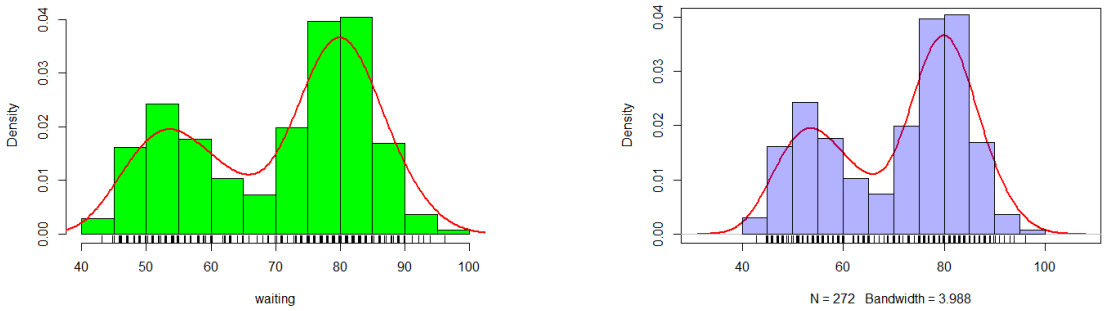
그림 11.22에 있는 두 그래프의 하단에 각각 추가된 rug plot의 결과가 다르게 나타남을 알 수 있는데, 이것은 rug()과 rug(jitter())의 차이로 인한 것이다. 함수 rug()는 X축에 눈금으로 자료의 위치를 표시하는 기능을 갖고 있는데, 자료의 값이 같으면 모두 한 개의 눈금으로 표시가 된다. 따라서 동일한 값이 많은 자료의 경우에는 실제 자료 크기보다 훨씬 적은 수의 눈금이 표시가 된다. 이러한 문제는 함수 jitter()로 어느 정도 해결이 되는데, jitter(x)는 x의 각 요소에 약간의 오차를 가감하는 기능을 갖고 있다. 비록 waiting에는 값이 같은 자료가 많더라도 jitter(waiting)에는 같은 값의 자료가 없게 된다. 가감되는 오차는 매우 적은 값이어서 원래의 값과의 차이는 무시할 수 있는 수준이 된다.

확률밀도함수 그래프는 단독으로 작성되는 경우도 많이 있지만, 히스토그램과 겹쳐서 작성하는 것도 의미 있는 작업이 된다. 히스토그램과 확률밀도함수 그래프를 겹쳐서 그리기 위해서는 우선 히스토그램을 확률밀도로 표현해야 되므로 함수 hist()에 옵션 freq=FALSE를 입력해야 한다. 두 그래프를 겹치게 작성하는 첫 번째 방법은 우선 함수 hist()로 히스토그램을 작성하고 함수 density()로 추정될 결과는 함수 lines()를 이용하여 히스토그램에 추가하는 방법이다.

```
> # 그림 11.23 (a)
> attach(faithful)
> hist(waiting, freq=FALSE, col="green", main="")
> rug(jitter(waiting))
> lines(fit, col="red", lwd=2)
```

두 번째 방법은 확률밀도함수 추정 결과를 함수 plot()으로 작성하고, 이어서 함수 hist()에 옵션 add=TRUE를 입력하여, 마치 낮은 수준의 그래프 함수처럼 히스토그램을 추가하는 방법이다. 이 경우 추가되는 히스토그램에 색을 넣으려면 투명도를 높여야 확률밀도함수 그래프가 보이게 된다. 투명한 색을 사용하려면 함수 rgb()를 사용해야 한다. 기본적인 사용법은 rgb(red, green, blue, alpha)로써, 각 요소들은 [0,1] 사이의 값을 갖게 되는데 색은 red, green, blue의 조합으로 결정되고, 투명도는 alpha로 결정된다. alpha의 값이 0에 가까울수록 더 투명한 색을 얻을 수 있다.

```
> # 그림 11.23 (b)
> plot(fit, col="red", lwd=2, main="", ylim=c(0,0.04))
> hist(waiting, freq=FALSE, col=rgb(0,0,1,alpha=0.3), add=TRUE)
> rug(jitter(waiting))
> detach(faithful)
```

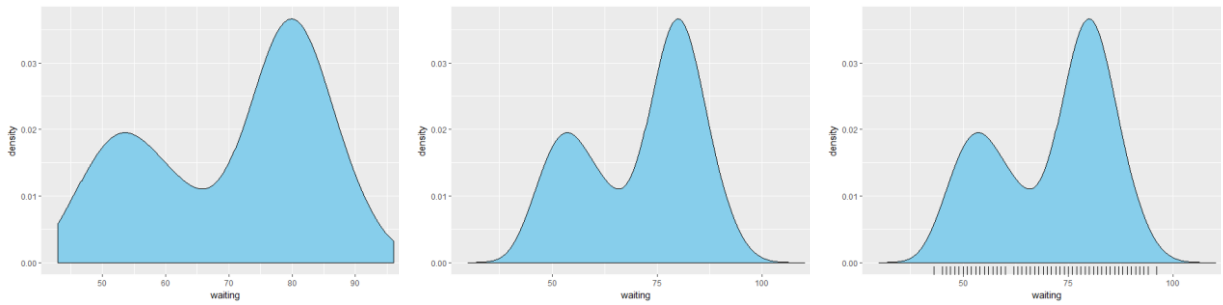


(a) (b)

<그림 11.23> 히스토그램과 확률밀도함수 그래프를 겹쳐지게 작성

패키지 ggplot2에서는 함수 geom_density()로 확률밀도함수 그래프를 바로 작성할 수 있다. 함수 geom_density()의 디폴트 stat인 함수 stat_density()에서 확률밀도함수를 추정하고, 그 결과를 geom 함수에서 그래프로 작성을 한다. 다만 이 때 주의할 점은 그래프를 표시할 X축의 구간을 자료의 실제 구간보다 더 넓게 잡아주어야 한다는 것인데, 그렇지 않았을 경우에는 꼬리가 잘린 형태의 확률밀도함수가 추정되어 그래프로 작성된다. 또한 Rug plot은 함수 geom_rug()를 추가하면 작성된다.

```
> p <- ggplot(faithful, aes(x=waiting)) +
    geom_density(fill="skyblue")

> # 그림 11.24 (a)
> p

> # 그림 11.24 (b)
> p + xlim(30,110)

> # 그림 11.14 (c)
> p + geom_rug() + xlim(30,110)
```

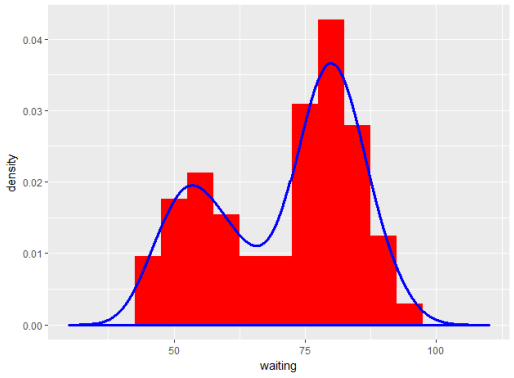


(a) (b) (c)

<그림 11.24> 패키지 ggplot2에서 작성된 확률밀도함수 그래프

히스토그램과 확률밀도함수 그래프를 겹치게 작성하기 위해서는 함수 geom_histogram()에서 시각적 요소 y에 변수 ..density..를 매핑해야 히스토그램이 확률밀도로 표현된다. 이어서 함수 geom_density()로 확률밀도함수 그래프를 추가하면 된다.

```
> # 그림 11.25
> ggplot(faithful, aes(x=waiting, y=..density..)) +
    geom_histogram(fill="red", binwidth=5) +
    geom_density(color="blue", size=1.1) +
    xlim(30,110)
```



<그림 11.25> 히스토그램과 확률밀도함수 그래프를 겹쳐지게 작성

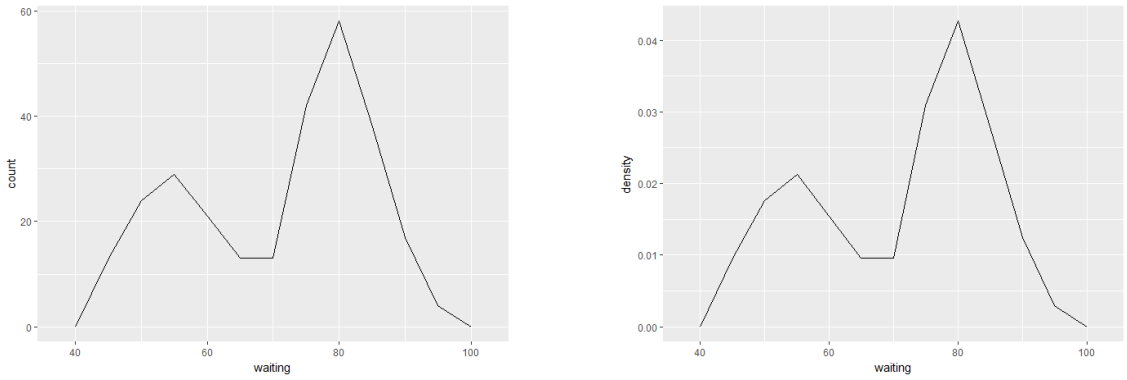
11.2.5 기타 유용한 그래프

지금까지 살펴본 그래프 외에 연속형 변수의 분포를 나타내는 유용한 그래프를 더 살펴보자.

- 도수분포다각형(Frequency polygon)

히스토그램이 각 구간에 속한 자료의 도수를 높이로 하는 막대로 분포를 나타내는 데 반하여 도수분포다각형은 각 구간의 도수를 선으로 연결한 다각형으로 분포를 나타내는 그래프이다. 그래프의 작성은 패키지 ggplot2의 함수 geom_freqpoly()로 할 수 있는데, 사용법은 함수 geom_histogram()과 동일하다. 예제로 데이터 프레임 faithful의 변수 waiting에 대한 도수분포다각형을 작성해 보자.

```
> pp <- ggplot(faithful, aes(x=waiting))

> # 그림 11.26 (a)
> pp + geom_freqpoly(binwidth=5)

> # 그림 11.26 (b)
> pp + geom_freqpoly(aes(y=..density..), binwidth=5)
```



(a) (b)

<그림 11.26> 패키지 ggplot2에서 작성된 도수분포다각형

도수분포다각형은 한 범주형 변수로 구분되는 그룹별로 연속형 변수의 분포를 비교하는 경우에 히스토그램보다 더 유용하게 사용될 수 있는 그래프이다.

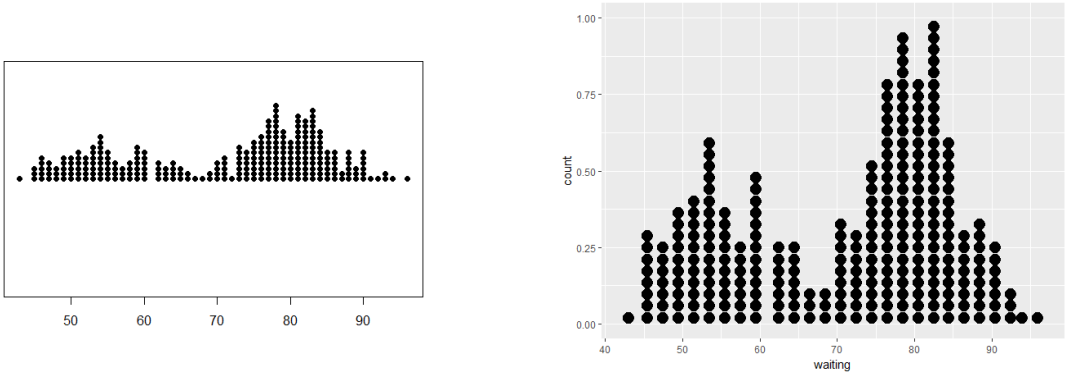
- 점 그래프(dot plot)

소규모 데이터의 분포를 표현할 때 유용하게 사용할 수 있는 그래프이다. 자료의 전 범위를 구간(bin)으로 구분하여 각 구간에 속한 자료 한 개당 하나의 점을 위로 쌓아 올리는 방식으로 작성되는 그래프이다.

패키지 graphics에서는 함수 stripchart()로 작성할 수 있고, 패키지 ggplot2에서는 함수 geom_dotplot()으로 할 수 있다. 다만 함수 stripchart()는 구간을 설정하지 않고 같은 값을 갖는 자료들을 대상으로 겹쳐 놓거나("overplot"), 약간 흩뜨려 놓거나("jitter") 또는 위로 쌓아 놓는("stack") 방식으로 그래프를 작성한다. 어떤 방식을 선택할 것인지는 method에 해당되는 문자를 지정하면 된다. 구간 설정을 할 수 없다는 점에서 함수 stripchart()로 작성한 점 그래프는 이산형 자료에 적합한 그래프라고 할 수 있다.

반면에 함수 geom_dotplot()에서는 구간의 간격을 변수 binwidth에 지정할 수 있기 때문에 연속형 자료의 분포를 잘 표현할 수 있는 그래프를 작성할 수 있다. 다만 점들은 위로 쌓아 올리는 방식으로만 그래프가 작성된다. 이제 데이터 프레임 faithful의 변수 waiting에 대한 점 그래프를 작성해 보자.

```
> # 그림 11.27 (a)
> stripchart(faithful$waiting, method="stack", pch=19)

> # 그림 11.27 (b)
> ggplot(faithful, aes(x=waiting)) +
    geom_dotplot(binwidth=1.5)
```



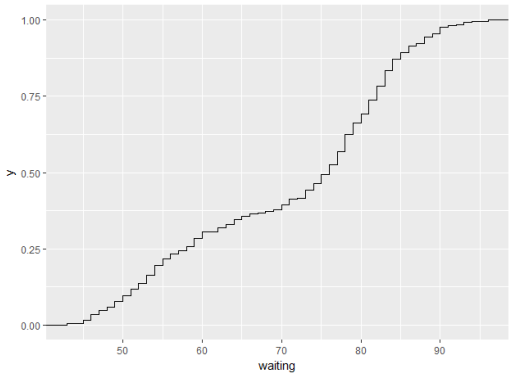
(a) (b)

<그림 11.27> 패키지 graphics와 ggplot2에서 각각 작성된 점 그래프

- 경험적 누적분포함수 그래프

경험적 누적분포함수(empirical cumulative distribution function)는 각 자료 값에서 $1/n$의 높이를 갖고 증가하는 계단함수이다. 한 변수의 분포형태를 매우 정확하게 표현할 수 있는 그래프라고 할 수 있다. 다만 확률밀도함수가 아닌 분포함수를 나타내는 그래프이기 때문에 익숙하지 않은 사용자도 있을 것이다. 그래프의 작성은 패키지 ggplot2에서 함수 stat_ecdf()로 할 수 있다. 데이터 프레임 faithful의 변수 waiting의 경험적 누적분포함수 그래프를 작성해 보자.

```
> # 그림 11.28
> ggplot(faithful, aes(x=waiting)) +
    stat_ecdf()
```



<그림 11.28> 패키지 ggplot2에서 작성된 경험적 누적분포함수 그래프

11.3 일변량 자료의 요약통계

일변량 자료 분석의 첫 단계는 그래프를 이용하여 자료의 분포에 어떤 특징이 있는지를 살펴보는 것이다. 분포의 특징을 시각적으로 나타내는 것에 더하여 자료의 정보를 몇 개의 숫자로 요약해 주는 요약통계가 추가되면 다음 단계의 분석, 즉 다른 변수와의 비교 및 관계 탐구 등으로 더

나아갈 수 있게 된다. 본 절에서는 연속형 자료를 대상으로 자료의 중심을 나타내는 요약통계와 자료의 퍼짐을 나타내는 요약통계에 관련된 R 함수를 살펴보고자 한다.

11.3.1 자료 중심에 대한 요약통계

자료의 중심을 나타내는 요약통계량으로 많이 사용되는 것이 평균과 중앙값이며, 함수 mean()과 median()으로 각각 계산할 수 있다. 통계학개론에서 소개되는 최빈값은 연속형 데이터의 경우에 큰 의미가 없는 통계량이며, 바로 계산해 주는 함수도 없으나 어렵지 않게 최빈값을 계산할 수 있다. 데이터 프레임 faithful의 변수 waiting에 대한 세 가지 요약통계량의 값을 계산해보자.

```
> attach(faithful)

> mean(waiting)
[1] 70.89706

> median(waiting)
[1] 76

> my_id <- which.max(table(waiting))
> table(waiting)[my_id]
78
15
> detach(faithful)
```

평균값과 중앙값은 각각 70.897과 76으로 계산되었다. 최빈값의 계산과정은 변수 waiting의 도수분포표를 함수 table()로 작성하고, 도수분포표에서 도수가 가장 큰 값을 보이는 자료의 위치를 함수 which.max()로 확인한 후 그 값을 대괄호에 적용시켜 해당하는 값과 빈도를 출력한 것이다. 최대빈도를 보이는 숫자가 78이라는 것을 알 수 있다.

자료에 결측값이 있는 경우, 함수 mean()과 median()의 계산 결과는 NA가 된다. 이런 경우에는 옵션 na.rm=TRUE를 포함시키면 자료에서 NA를 제외하고 나머지 자료만을 대상으로 계산을 하게 된다. 데이터 프레임 airquality의 변수 Ozone에는 다수의 결측값이 포함되어 있다. 변수 Ozone의 평균값과 중앙값을 각각 계산해보자.

```
> attach(airquality)
> mean(Ozone); median(Ozone)
[1] NA
[1] NA
> mean(Ozone, na.rm=TRUE); median(Ozone, na.rm=TRUE)
[1] 42.12931
[1] 31.5
> detach(airquality)
```

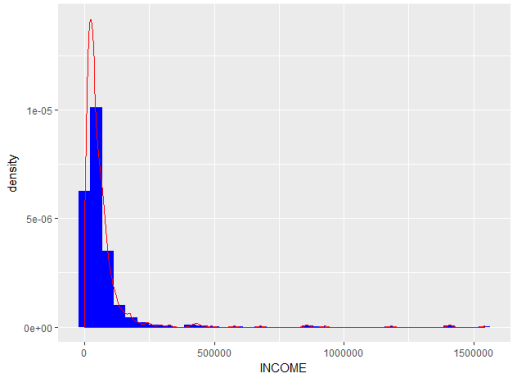
분포의 형태가 좌우대칭인 경우에는 평균값이 가장 좋은 요약통계량이 되지만 좌측 또는

우측으로 심하게 치우친 분포의 경우에는 극단값에 큰 영향을 받지 않는 중앙값이 분포의 중심을 더 정확하게 나타내는 요약통계량이 될 것이다. 따라서 만일 평균값과 중앙값이 크게 차이가 난다면 치우친 형태의 분포가 될 가능성이 높을 것이다. 패키지 UsingR에 있는 데이터 프레임 cfb는 2001년 미국 소비자 재정상태에 관한 조사 데이터인데, 이 중 변수 INCOME에는 가구당 소득이 들어있다. 변수 INCOME의 분포형태를 살펴보자.

```
> library(UsingR)
> attach(cfb)
> mean(INCOME); median(INCOME)
[1] 63402.66
[1] 38032.7
> detach(cfb)
```

평균값이 중앙값의 약 1.67배가 될 정도로 크게 차이가 나는 것을 볼 때 우측으로 심하게 치우친 분포임을 알 수 있다. 변수 INCOME의 정확한 분포의 형태를 알아보기 위하여 히스토그램을 작성해 보자.

```
> # 그림 11.29
> ggplot(cfb, aes(x=INCOME, y=..density..)) +
    geom_histogram(bins=35, fill="blue") +
    geom_density(color="red")
```



<그림 11.29> 변수 INCOME의 히스토그램

심하게 치우친 분포에 대해서는 대부분 변수변환을 통해 좌우대칭에 가까운 분포로 형태를 바꿔주게 된다. 변수 INCOME의 분포는 우측으로 심하게 치우친 분포이므로 로그 혹은 제곱근 변환을 통해 좌우대칭 분포로 변환시킬 수 있는데, 여기에서는 로그 변환을 실시해보자.

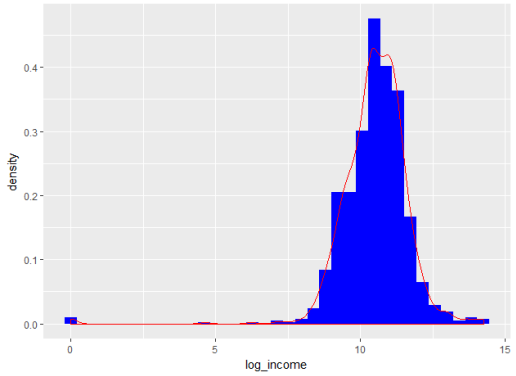
```
> log_income <- log(cfb$INCOME)

> range(log_income)
[1]    -Inf 14.2485
> range(cfb$INCOME)
[1]       0 1541866
```

위 실행결과는 로그 변환에서 주의할 점을 알려주고 있는데, 로그 변환된 변수 log_income에 -Inf가 있다는 점이다. 이것은 변수 INCOME에 0이 있기 때문인데, 해결방안으로는 모든 데이터를 우측으로 1만큼 이동 시키는 것이다.

```
> log_income=log(cfb$INCOME+1)
> range(log_income)
[1]  0.0000 14.2485
```

로그 변환된 변수 INCOME의 분포를 히스토그램으로 나타내보자.

```
> # 그림 11.30
> ggplot(data.frame(log_income),
         aes(x=log_income, y=..density..)) +
    geom_histogram(fill="blue", bins=35) +
    geom_density(color="red")
```



<그림 11.30> 로그 변환된 변수 INCOME의 히스토그램

우측으로 심하게 치우친 모습을 보인 그림 11.29의 분포가 로그 변환으로 좌우대칭의 모습을 보이는 그림 11.30의 분포로 바뀐 것을 알 수 있다. 그림 11.30의 좌측 끝부분에 있는 극단값은 변수 INCOME의 값이 0인 4개 케이스에 해당되는 것이다.

11.3.2 자료 퍼짐에 대한 요약통계

자료의 중심에 대한 요약통계만으로는 자료의 분포에 대한 정확한 묘사가 불가능하다고 할 수 있는데, 다음에 주어진 시험점수 모의 자료를 살펴보자.

	1차 시험(test1)	2차 시험(test2)
점수	75, 77, 80, 82, 85, 87, 88	50, 57, 80, 82, 86, 100, 100
평균	82	79.3

두 시험의 평균은 비슷하지만 자료의 퍼짐 정도에는 큰 차이가 있음을 알 수 있다. 자료의 퍼짐 정도를 측정하는 통계량에는 범위(range), 사분위범위(Interquartile range), 표준편차 및 분산 등이 있다.

우선 1차 시험과 2차 시험 데이터의 범위를 구해보자. 범위는 가장 단순한 자료 퍼짐 정도의 측정도구이다. 많은 수의 자료 중 오직 두 값(최소값, 최대값)만으로 계산하는 것이므로 전체 자료의 성격을 올바로 담아낼 수는 없다. 자료의 범위는 최소값과 최대값을 계산해주는 함수 range()의 결과를 함수 diff()에 입력하여 두 값의 차이를 계산하면 된다.

```
> test1 <- c(75,77,80,82,85,87,88)
> test2 <- c(50,57,80,82,86,100,100)
> diff(range(test1))
[1] 13
> diff(range(test2))
[1] 50
```

다음으로 사분위범위와 관련된 함수에 대해서 살펴보자. 사분위범위는 분위수 혹은 백분위수를 기반으로 정의되는 것이므로 분위수에 대해서 먼저 살펴보자. p 분위수란 전체 자료 중 $100\times p$ %의 자료보다는 크고 나머지 $100\times(1-p)$ %의 자료보다는 작은 수를 의미한다. 단, 여기서 p 는 $0\le p\le 1$ 을 만족시켜야 한다. 백분위수는 분위수를 백분율로 표시한 것이다. 따라서 중앙값은 0.5분위수 및 50백분위수가 된다. 이어서 4분위수란 전체 데이터를 크기 순으로 나열했을 때 4등분하는 3개의 수를 의미하는 것으로 0.25분위수(Q_1), 중앙값, 0.75분위수(Q_3)를 의미하는 것이다. 이제 사분위범위를 정의할 수 있는데, 사분위범위란 0.75분위수와 0.25분위수의 차이를 의미한다.

분위수의 계산은 함수 quantile()로 하며, 기본적인 사용법은 quantile(x, probs=seq(0, 1,by=0.25), ...)가 된다. 함수 quantile()의 옵션 probs에는 원하는 확률값을 지정할 수 있으며, 생략되면 최소값(0%), 최대값(100%)를 포함한 4분위수가 출력된다. 이제 시험점수 모의 데이터에 함수 quantile()을 적용시켜보자.

```
> quantile(test1)
   0%  25%  50%  75% 100%
75.0 78.5 82.0 86.0 88.0
> quantile(test2)
   0%   25%   50%   75%   100%
 50.0  68.5  82.0  93.0 100.0
> quantile(test1, probs=.25)
 25%
78.5
> quantile(test2, probs=.25)
 25%
68.5
```

사분위범위는 함수 IQR()로 계산할 수 있다.

```
> IQR(test1)
[1] 7.5
> quantile(test1, probs=0.75)-quantile(test1, probs=0.25)
75%
7.5
> IQR(test2)
[1] 24.5
> quantile(test2, probs=0.75)-quantile(test2, probs=0.25)
 75%
24.5
```

자료의 퍼짐 정도를 측정하는 도구로 가장 많이 사용되는 통계량이 표준편차와 분산이다. 두 통계량은 각 자료들이 자료의 평균에서부터 떨어진 거리를 이용하여 퍼짐 정도를 측정하고 있다. 대략적으로 표준편차는 각 관찰값들이 평균에서부터 떨어져있는 거리의 평균값이라고 해석할 수 있다. 표준편차와 분산은 함수 sd()와 var()로 계산할 수 있다.

```
> var(test1); var(test2)
[1] 24.66667
[1] 377.5714
> sd(test1); sd(test2)
[1] 4.966555
[1] 19.4312
```

지금까지 살펴본 함수들과 함께 어울리며 일변량 자료의 요약통계를 계산하는데 효과적으로 사용되는 함수가 summary()이다. 이 함수도 generic 함수로서 입력되는 객체의 class에 따라 다른 분석 결과가 출력되는데, 숫자형 자료가 입력되면 4분위수와 평균값, 최소값과 최대값이 계산된다.

```
> library(UsingR)
> summary(cfb$INCOME)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      0   20558   38033   63403   69898 1541866
```

변수 INCOME의 평균값이 0.75분위수와 무척 가까운 값임을 알 수 있는데, 실제 평균값이 몇 분위수에 해당되는지 어떻게 계산할 수 있겠는가? 이것은 변수 INCOME의 각 관찰값들 중 평균값보다 작은 값들의 비율을 계산하면 되는데, 다음과 같이 간단하게 계산할 수 있다.

```
> with(cfb, mean(INCOME<=mean(INCOME)))
[1] 0.705
```

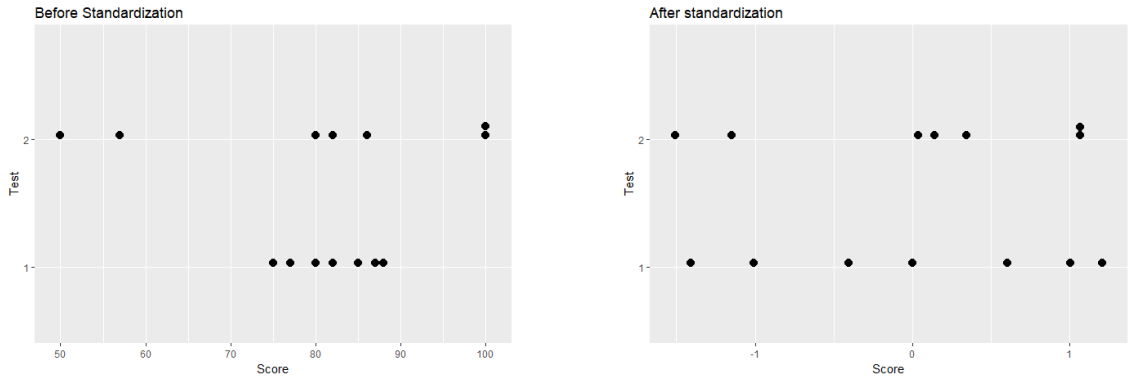
따라서 변수 INCOME의 평균값은 0.705분위수에 해당한다는 것을 알 수 있다.

중심과 퍼짐 정도가 다른 자료들은 표준화 과정을 통하여 동일한 중심과 퍼짐 정도를 갖게 된다. 자료의 표준화는 함수 scale()로 할 수 있으며, 일반적인 사용법은 scale(x, center = TRUE, scale = TRUE)가 된다. x는 숫자형 행렬 또는 벡터가 되는데, 행렬인 경우에는 각 열

단위로 표준화가 진행된다. 시험점수 자료를 행렬로 묶어서 표준화를 실시해보자.

```
> scale(cbind(test1,test2))
         test1      test2
[1,] -1.4094277 -1.50714924
[2,] -1.0067341 -1.14690381
[3,] -0.4026936  0.03675974
[4,]  0.0000000  0.13968700
[5,]  0.6040404  0.34554153
[6,]  1.0067341  1.06603239
[7,]  1.2080809  1.06603239
attr(,"scaled:center")
    test1    test2
82.00000 79.28571
attr(,"scaled:scale")
    test1     test2
 4.966555 19.431197
```

시험점수 자료의 변환 전 분포와 변환 후 분포를 나타낸 그래프가 그림 11.31에 있다.



(a) (b)

<그림 11.31> 표준화 전후의 자료 분포

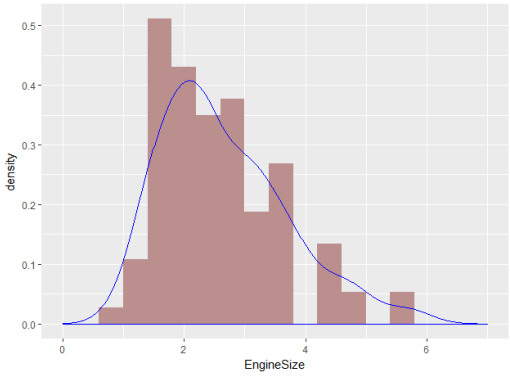
11.4 11장을 마치며

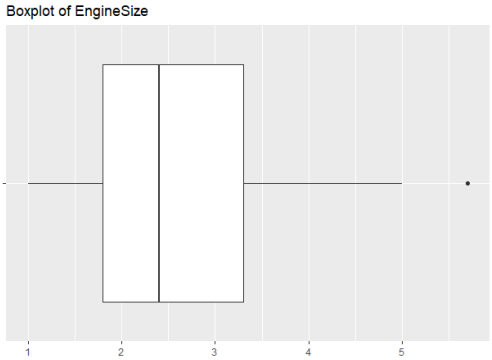
11장에서 우리는 일변량 범주형 자료와 연속형 자료가 갖고 있는 정보를 어떻게 그래프로 묘사할 것인지를 살펴보았다. 자료의 유형별로 많은 그래프가 소개되었다. 또한 자료의 정보를 몇몇 숫자로 어떻게 요약할 것이지도 살펴보았다. 요약된 정보를 바탕으로 다음 장에서는 일변량 자료에 대한 통계적 추론을 실시해보자.

11.5 연습문제

1. 패키지 MASS에 있는 데이터 프레임 Cars93는 1993년 미국에서 판매된 93 종류의 자동차에 대한 데이터로 27개 변수로 구성되어 있다.

   1) 변수 EngineSize에 대한 히스토그램과 상자그림을 다음과 같이 작성하라.



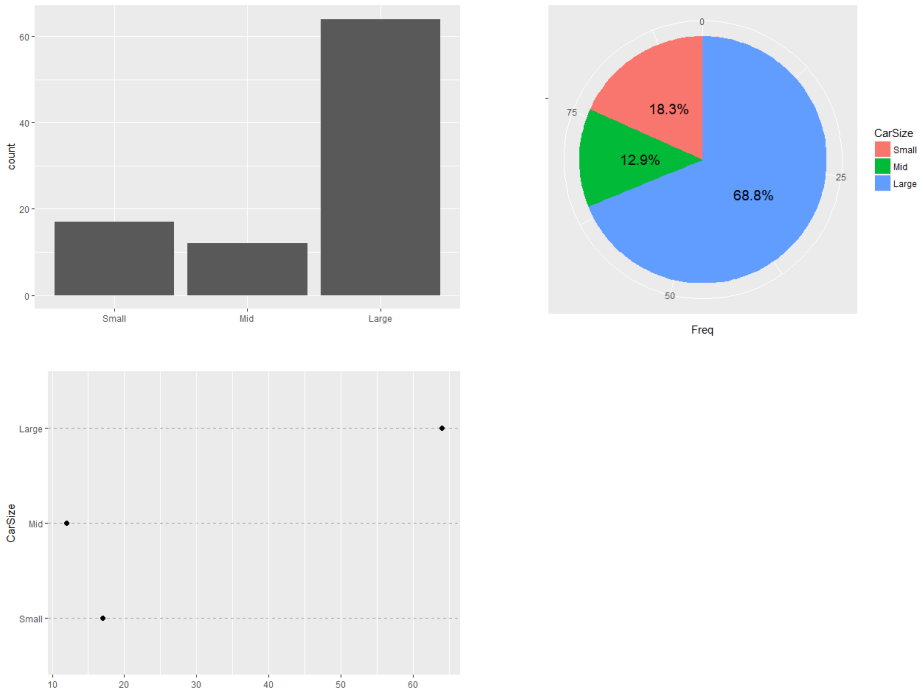


   2) 상자그림에 점으로 표시된 관찰값에 대한 변수 Manufacturer와 Model의 값을 다음과 같이 출력하라.

```
  Manufacturer       Model
1        Buick Roadmaster
2    Chevrolet    Corvette
```

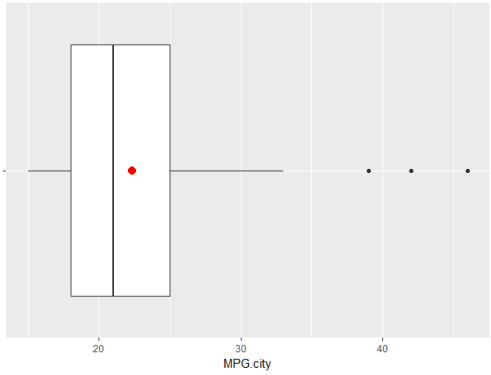
   3) 변수 EngineSize에 대한 평균과 표준편차를 계산하되 상자그림에서 점으로 표시된 두 관찰값을 제외하고 계산하라. 참고로 평균은 2.60, 그리고 표준편차는 0.944로 계산되었다.

   4) 변수 EngineSize를 1.6 이하(Small), 1.6에서 2.0 이하(Mid), 2.0 초과(Large)의 세 범주로 구분하여 각 범주에 대한 도수를 구하고, 이것에 대한 막대 그래프, 파이 그래프, Cleveland의 점 그래프를 다음과 같이 작성하라.



2. 패키지 MASS에 있는 데이터 프레임 Cars93에 대한 문제이다.

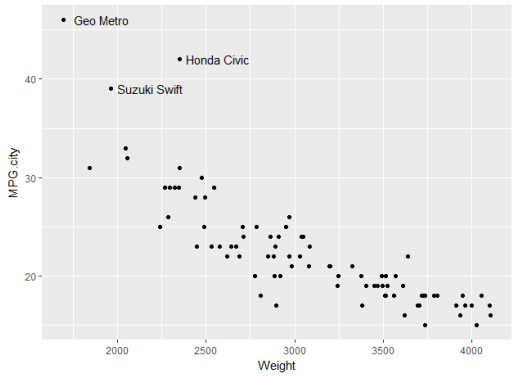
1) 변수 MPG.city의 상자그림을 다음과 같이 작성하라. 상자 안의 빨간 점은 변수 MPG.city의 평균값 위치를 표시한 것이다.



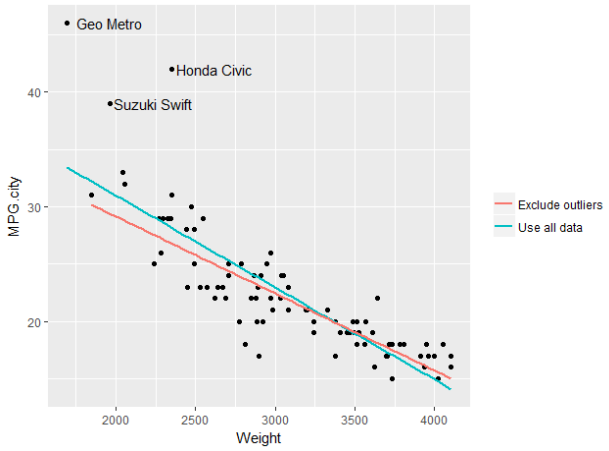
2) 변수 MPG.city의 상자그림에 점으로 표시된 세 관찰값의 변수 Manufacturer, Model, MPG.city, Weight만으로 이루어진 데이터 프레임을 구성하되, 다음과 같이 변수 Weight의 크기 순으로 관찰값을 재배열하라.

```
  Manufacturer Model MPG.city Weight
1        Honda Civic        42   2350
2       Suzuki Swift        39   1965
3          Geo Metro        46   1695
```

3) 변수 MPG.city와 Weight의 산점도를 작성하되, 변수 MPG.city의 상자그림에서 점으로 표시된 세 자동차의 Manufacturer와 Model 이름을 다음과 같이 해당하는 점 옆에 표시하라.



4) 변수 MPG.city와 Weight의 산점도에 변수 MPG.city를 반응변수, Weight를 설명변수로 하는 회귀직선을 추가하되, 모든 자료를 대상으로 하는 모형과 변수 MPG.city의 상자그림에서 점으로 표시된 세 관찰값을 제외한 자료를 대상으로 하는 모형의 회귀직선을 다음과 같이 추가하라.



Hint: 회귀직선 추가는 geom_smooth(method="lm", se=FALSE)로 할 수 있다.

3. 125명의 학생이 수강하고 있는 통계학 개론의 시험성적이 평균이 75점, 표준편차가 10점인 정규분포를 한다고 가정하자.

1) 위 가정사항에 따른 125개 데이터를 생성하라. 단, 100점이 넘는 데이터는 100점으로 수정하고, 반올림으로 데이터의 소수점을 제거하라.

2) 생성된 시험점수의 히스토그램을 작성하되 확률밀도함수 그림을 겹쳐서 표시하라.

3) 학점은 다음의 기준에 의하여 상대평가로 부여된다.

A	B	C	D	F
20%	30%	30%	15%	5%

위 기준에 의하여 125명에게 학점을 부여하라. 부여된 학점의 상대도수분포표를 작성하여 부여된 학점이 기준에 부합되었는지 여부도 확인하라.