EDA 3장 과제

주의사항을 숙지하였고 모든 책임을 지겠습니다.

2019122041 송유진

---

1. EDA의 네 가지 주제에 대하여 설명하라. 한 주제당 3줄 이하.

1. Revelation(그래프를 통한 현시성)

: 현시성이란 데이터를 그래프로 시각화함으로써 구조를 효율적으로 파악하는 것. EDA에서는 다양한 그래프 작성 기법들(히스토그램, Box Plot, 산점도 등)이 사용되는데 해당 시각화를 통해 훨씬 직관적으로 인사이트를 얻을 수 있음.

2. Residual(잔차 계산)

: 잔차는 각 개별 관측값이 자료의 주 경향으로부터 얼마나 벗어났는지를 뜻함. 잔차를 구하면 어떠한 특정 데이터가 보통의 경우와 다른 경향을 가지고 있는지를 알 수 있음.

3. Re-expression(재표현)

: 재표현이란 단순한 분석과 해석을 위해 원래의 변수를 적당한 척도로 바꾸는 것을 의미. 보통 로그 변환이나 제곱근 변환 등을 통해 원래의 변수를 바꾸며 이를 통해 분포의 대칭성, 선형성, 분산 안정성 등 데이터의 구조를 파악하는데 유용함.

4. Resistance(저항성의 강조)

: 저항성이란 자료의 일부가 파손되었을 때, 영향을 적게 받는 성질. 자료가 파손된다는 것은 자료 일부가 뜬금없는 값으로 대체되는 경우를 의미하며 EDA의 관점에서 평균보다는 일부 자료의 파손에 저항적인 중위수가 바람직한 대푯값 측도로서 선호됨.

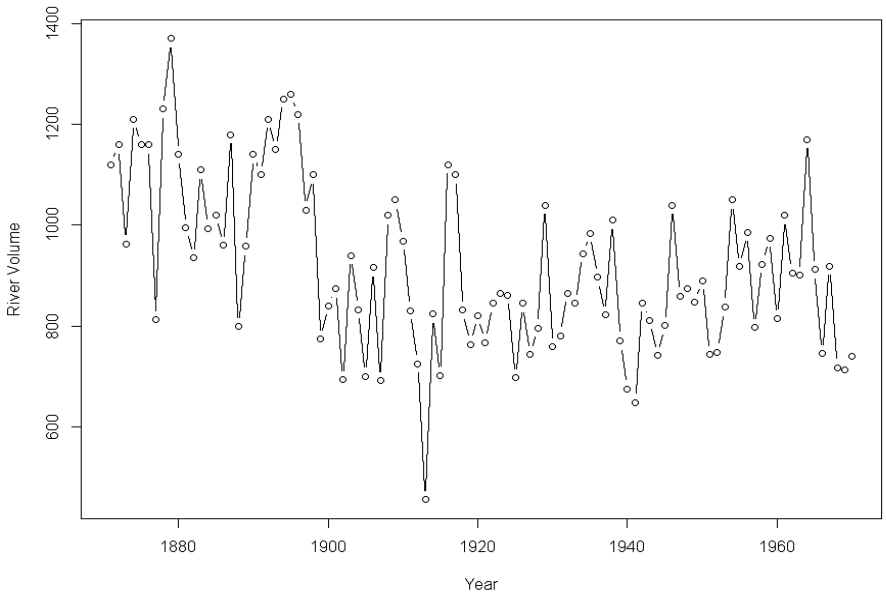
2. Ashwan에서 측정한 Nile 강의 유량 자료 (R datasets:: Nile)
가. 줄기그림을 그리고 자료에 대하여 설명하여라. depth 등이 들어간 '완벽한' 줄기그림을 손으로 작성하여라. R이 만들어 준 부분을 오려 붙이고 나머지 연필로 완성하여라. HWP, WORD 등으로 편집하여도 된다.

줄기그림을 그리기 전, R datasets Nile 데이터의 특징을 살펴보기 위해 summary(), length() 함수를 활용했다.

```
> Nile
Time Series:
Start = 1871
End = 1970
Frequency = 1
  [1] 1120 1160  963 1210 1160
  [6] 1160  813 1230 1370 1140
 [11]  995  935 1110  994 1020
 [16]  960 1180  799  958 1140
 [21] 1100 1210 1150 1250 1260
 [26] 1220 1030 1100  774  840
 [31]  874  694  940  833  701
 [36]  916  692 1020 1050  969
 [41]  831  726  456  824  702
 [46] 1120 1100  832  764  821
 [51]  768  845  864  862  698
 [56]  845  744  796 1040  759
 [61]  781  865  845  944  984
 [66]  897  822 1010  771  676
 [71]  649  846  812  742  801
 [76] 1040  860  874  848  890
 [81]  744  749  838 1050  918
 [86]  986  797  923  975  815
 [91] 1020  906  901 1170  912
 [96]  746  919  718  714  740
> summary(Nile)
   Min. 1st Qu.  Median    Mean
  456.0   798.5   893.5   919.4
3rd Qu.    Max.
 1032.5  1370.0
> length(Nile)
[1] 100
```

summary()를 통해 해당 데이터셋의 최소값은 456, 최대값은 1370이며 중앙값은 893.5임을 알 수 있었다. 그리고 데이터셋은 총 100개 연도(1871~1970)의 나일강 유량을 기록한 시계열 자료임을 알 수 있었다.

plot(Nile, xlab = "Year", ylab = "River Volume", type ="b")
해당 코드를 활용해 time series plot를 그려본 결과는 다음과 같다.



상단의 time series plot을 보며 발견한 흥미로운 부분은 1900년 이후 유량의 분포 범위대가 달라졌다는 점이었다. 1900년 이전에는 1000-1400 사이에서 유량이 변화하고 있지만 그 이후에는 주로 800-1000 사이에서 유량이 변화하고 있었다.

1)이는 Ashwan의 '1902년 아스완댐 준공'으로 인한 결과임을 알 수 있었다. 1902년을 기준으로 데이터를 2개로 나누고 stem and leaf를 통해 비교해보는 작업을 수행해보았다.

```
> Nile
Time Series:
Start = 1871
End = 1970
Frequency = 1
  [1] 1120 1160  963 1210 1160 1160  813 1230 1370 1140  995  935 1110  994 1020  960 1180  799  958 1140
 [21] 1100 1210 1150 1250 1260 1220 1030 1100  774  840  874  694  940  833  701  916  692 1020 1050  969
 [41]  831  726  456  824  702 1120 1100  832  764  821  768  845  864  862  698  845  744  796 1040  759
 [61]  781  865  845  944  984  897  822 1010  771  676  649  846  812  742  801 1040  860  874  848  890
 [81]  744  749  838 1050  918  986  797  923  975  815 1020  906  901 1170  912  746  919  718  714  740
> NileBefore <- Nile[1:31]
> NileAfter <- Nile[32:100]
> library(aplpack)
> stem.leaf.backback(NileBefore, NileAfter)
1 | 2: represents 120, leaf unit: 10
          NileBefore       NileAfter
                     |  4* |
                     |  4. |5                  1
                     |  5* |
                     |  5. |
                     |  6* |4                  2
                     |  6. |7999               6
                     |  7* |00112444444       17
  2                 97|  7. |5667899            24
  4                 41|  8* |011222333344444  (15)
  5                  7|  8. |6666799            30
  6                  3|  9* |001111244          23
 11              99665|  9. |6788               14
 13                 32| 10* |12244              10
                      | 10. |55                  5
 (6)            442100| 11* |02                  3
 12              86665| 11. |7                   1
  7               3211| 12* |
  3                 65| 12. |
                      | 13* |
  1                  7| 13. |
                      | 14* |
n:                  31       69
```

31개의 연도는 NileBefore로 나머지는 NileAfter로 생성해주고 aplpack 패키지를 활용해서 두 데이터의 stem and leaf를 동시에 비교해보았다.

()안의 숫자는 median 값이 해당 stem에 위치하고 있다는 의미인데 왼쪽(댐 건설 전)에는

1) https://artsandculture.google.com/entity/%EC%95%84%EC%8A%A4%EC%99%84/m0_38?hl=ko

median이 11 stem에 위치하고 있고 오른쪽(댐 건설 후)에는 median이 8에 위치하고 있다.
즉, 우리는 나일강에 준공된 아스완댐이 유량을 감소시키는 원인이 되었음을 추론할 수 있다.

다시 Nile 데이터로 돌아와 Nile의 stem and leaf 그림을 그려본 결과는 아래와 같다.

```
> stem(Nile)

  The decimal point is 2 digit(s) to the right of the |

   4 | 6
   5 |
   6 | 5899
   7 | 000123444455667778
   8 | 00001122223334455555667779
   9 | 00112222444666788899
  10 | 0122234455
  11 | 00012244566678
  12 | 112356
  13 | 7
```

decimal point가 오른쪽으로 2이기 때문에 자료는 100~1000단위임을 알 수 있다.
Nile 데이터는 800 그리고 1100대에서 봉우리가 생기는 형상을 보이고 있지만 stem의 수가 적어 단봉분포인지, 이봉분포인지 확인하기 어렵다는 한계가 있었다.
800 그리고 1100에서 봉우리가 생기는 것은 앞선 아스완댐 기준으로 stem.leaf.backback에서도 살펴보았던 결과이다.

다음은 Nile 자료에 log를 취한 후 stem and leaf 그림을 그려보았다.

```
> stem(log(Nile))

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

  60 | 2
  62 |
  64 | 8244555789
  66 | 11111234455688890001112233344444666779
  68 | 00122223455677889900233345566
  70 | 00012244566667001134
  72 | 2
```

일반적으로 로그 변환을 취하는 이유는 비대칭적 분포가 심하게 나타나는 데이터의 경우 자료를 더 대칭적으로 변하게끔 만들기 위해서이다. Nile 데이터에 로그를 취한 후 그린 줄기그림이 원래의 줄기그림보다 더 대칭적이게 transform 했음을 알 수 있었다. 그러나 Nile의 경우 비대칭의 정도가 심한 편은 아니기 때문에 로그 변환의 장점이 뚜렷하게 드러나지는 않았다.

해당 결과를 바탕으로 (나)로 이동하여 scale=2 옵션을 통해 stem을 늘려보도록 하겠다.

나. scale=2 옵션을 사용하여 (가)에서의 결과와 비교하여라. 어떤 것이 분포의 특성을 잘 나타내는가?

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

  The decimal point is 2 digit(s) to the right of the |

   4 | 6
   5 |
   5 |
   6 |
   6 | 5899
   7 | 0001234444
   7 | 55667778
   8 | 000011222233344
   8 | 555556667779
   9 | 00112222444
   9 | 66678899
  10 | 01222344
  10 | 55
  11 | 00012244
  11 | 566678
  12 | 1123
  12 | 56
  13 |
  13 | 7
```

scale=2 옵션을 적용해보니 stem의 수가 늘어나 분포의 특성을 파악하기 더 수월해졌음을 알 수 있다. Nile 데이터셋은 800, 1100에서 봉우리를 이루는 이봉분포의 성질을 가지고 있음을 scale 옵션을 주기 이전보다 명확하게 확인할 수 있었다.
즉, (가)에 비해 scale=2 옵션을 주어 stem을 늘린 (나)의 경우가 분포적 특성을 더 잘 나타내주고 있음을 알 수 있다.

로그 변환을 한 줄기그림도 아래와 같이 scale=2의 옵션을 주어 다시 그려보았다.

```
> stem(log(Nile), scale=2)

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

  61 | 2
  62 |
  63 |
  64 | 8
  65 | 244555789
  66 | 1111123445568889
  67 | 00011122333444446666779
  68 | 001222234556778899
  69 | 00233345566
  70 | 00012244566667
  71 | 001134
  72 | 2
```

로그 변환을 한 경우도 원래 Nile 데이터와 같이 scale=2의 옵션을 통해 stem의 수를 더 늘리면 이봉분포의 특성을 더 쉽게 알 수 있었다.

~~다. 문자전서를 만들고 결과를 설명하라. R에 함수가 없으니 원자료와 upward, downward rank를 구하여 세 개 열로 출력하여 depth에 맞는 값을 찾아 표를 수작업 또는 편집하여 완성하여라. (스크립트 짜면 시간이 많이 걸릴 수 있으니 의욕이 넘치는 학생은 일단 손으로 하고 '시간이 남으면' 도전하라. 0.5점 추가 점수.)~~

3. R에서 lattice 패키지를 설치하여 singer 자료를 이용하여라. Alto 1, 2를 Alto 그룹으로, Soprano 1, 2를 Soprano 그룹하여 줄기그림을 그려 두 그룹의 키를 비교하여라. 음역대와 키가 상관이 있는가?

```
> data('singer', package='lattice')
> summary(singer)
     height
 Min.   :60.0
 1st Qu.:65.0
 Median :67.0
 Mean   :67.3
 3rd Qu.:70.0
 Max.   :76.0

     voice.part
 Bass 1   :39
 Soprano 1:36
 Alto 1   :35
 Soprano 2:30
 Alto 2   :27
 Bass 2   :26
 (Other)  :42
```

lattice 패키지의 singer 자료를 summary() 함수를 통해 살펴본 결과 height와 voice.part column으로 이뤄진 데이터임을 확인했다.

Alto =
singer.frame[singer.frame$voice.part=='Alto 1'| singer.frame$voice.part=='Alto 2',]

Soprano =
singer.frame[singer.frame$voice.part=='Soprano 1'| singer.frame$voice.part=='Soprano 2',]

문제에서 지시한 방향대로 상단의 코드를 통해 Alto 1, 2는 Alto로, Soprano 1, 2는 Soprano로 그룹을 할당해주었다. 그리고 stem() 함수를 통해 두 그룹의 키를 비교하기 위한 stem and leaf를 그려보았다.

```
> stem(Alto$height)

  The decimal point is at the |

  60 | 00000
  62 | 0000000000
  64 | 00000000000000000
  66 | 00000000000000000000
  68 | 0000
  70 | 000000
  72 | 0

> stem(Soprano$height)

  The decimal point is at the |

  60 | 00000000
  62 | 0000000000000000
  64 | 00000000000000000000000000
  66 | 000000000000
  68 | 00
  70 | 0
```

각 그룹 별 stem을 비교해보면 Alto는 60부터 72까지, Soprano는 60부터 70으로 Alto가 stem의 개수가 더 많았다.

한 눈에 두 그룹을 비교하기 위해 aplpack 패키지를 활용하여 stem.leaf.backback() 함수를 사용해서 stem and leaf을 그려보았다.

```
> library(aplpack)
경고메시지(들):
패키지 'aplpack'는 R 버전 4.0.5에서 작성되었습니다
> stem.leaf.backback(Alto$height, Soprano$height)
```

```
1 | 2: represents 1.2, leaf unit: 0.1
              Alto$height       Soprano$height
 1                     0| 60* |0000                     4
                        | 60. |
 5                  0000| 61* |0000                     8
                        | 61. |
 8                   000| 62* |00000000000             19
                        | 62. |
15                7000000| 63* |000000                  25
                        | 63. |
23               00000000| 64* |00000                   30
                        | 64. |
(9)              000000000| 65* |00000000000000000000  (20)
                        | 65. |
30          0000000000000| 66* |000000000               16
                        | 66. |
17                 000000| 67* |0000                     7
                        | 67. |
11                   00| 68* |00                       3
                        | 68. |
 9                   00| 69* |
                        | 69. |
 7                 000000| 70* |0                        1
HI: 72
n:                      62       66
```

Alto 그룹의 height의 경우 66에서, Soprano의 경우 62, 65에서 봉우리를 확인할 수 있었다.

더 정확한 기술통계량을 살펴보기 위해 각 그룹 별 summary() 함수를 적용해보았다.

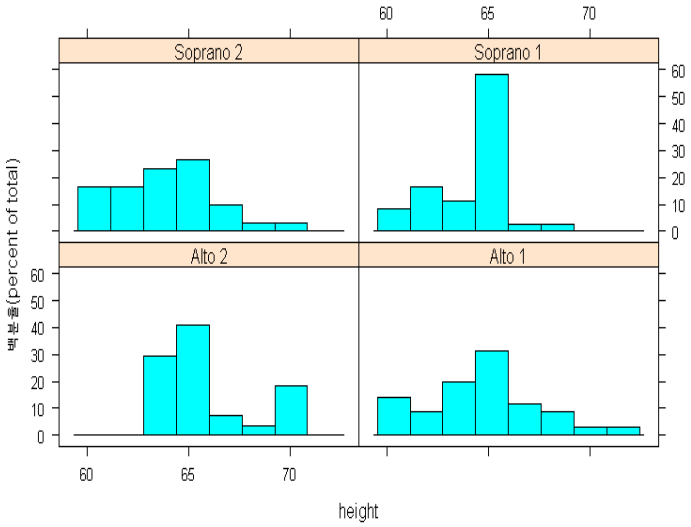
```
> summary(Alto)
     height         voice.part
 Min.   :60.00   Alto 1 :35
 1st Qu.:64.00   Alto 2 :27
 Median :65.00   Bass 2 : 0
 Mean   :65.39   Bass 1 : 0
 3rd Qu.:67.00   Tenor 2: 0
 Max.   :72.00   Tenor 1: 0
                 (Other): 0
> summary(Soprano)
     height          voice.part
 Min.   :60.00   Soprano 1:36
 1st Qu.:62.00   Soprano 2:30
 Median :65.00   Bass 2   : 0
 Mean   :64.12   Bass 1   : 0
 3rd Qu.:65.00   Tenor 2  : 0
 Max.   :70.00   Tenor 1  : 0
                 (Other)  : 0
```

두 그룹의 최소값과 중앙값은 각각 60, 65로 동일했지만 1분위수의 경우 Alto 그룹은 64, Soprano 그룹은 62 그리고 3분위수의 경우 Alto 그룹은 67, Soprano 그룹은 65 그리고 최대값은 Alto 그룹 72, Soprano 그룹은 70으로 확인했다.

즉, Alto 그룹이 Soprano 그룹와 비교했을 때 키가 더 큰 경향이 있음을 확인했다. 더 자세히 살펴보기 위해 아래와 같이 다양한 그래프를 통한 현시성을 확인해보았다.

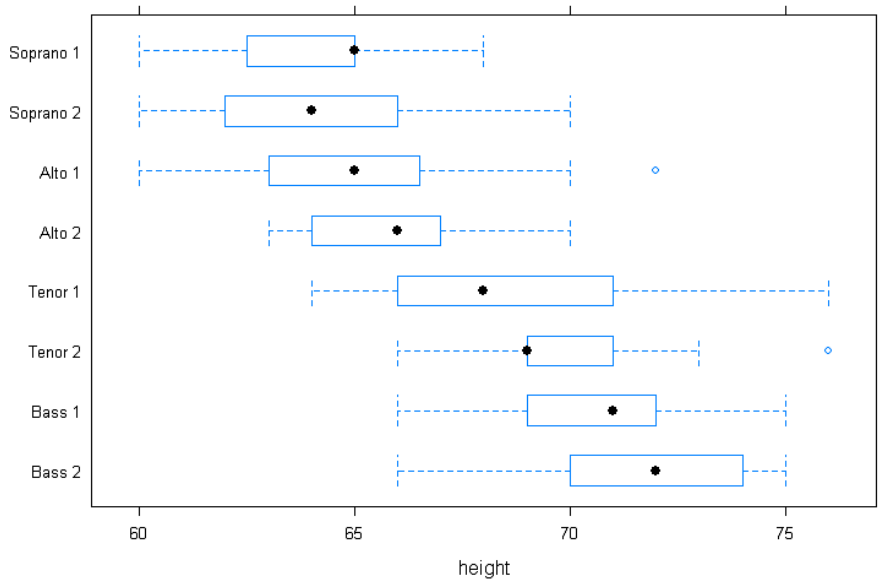
```
Alto_Sop = rbind(Alto, Soprano)
histogram(~height | voice.part, data = Alto_Sop)
```

Alto 데이터프레임과 Soprano 데이터프레임을 rbind로 합친 뒤 Alto_Sop 데이터프레임을 생성하고 다음과 같이 히스토그램을 그려보았다.

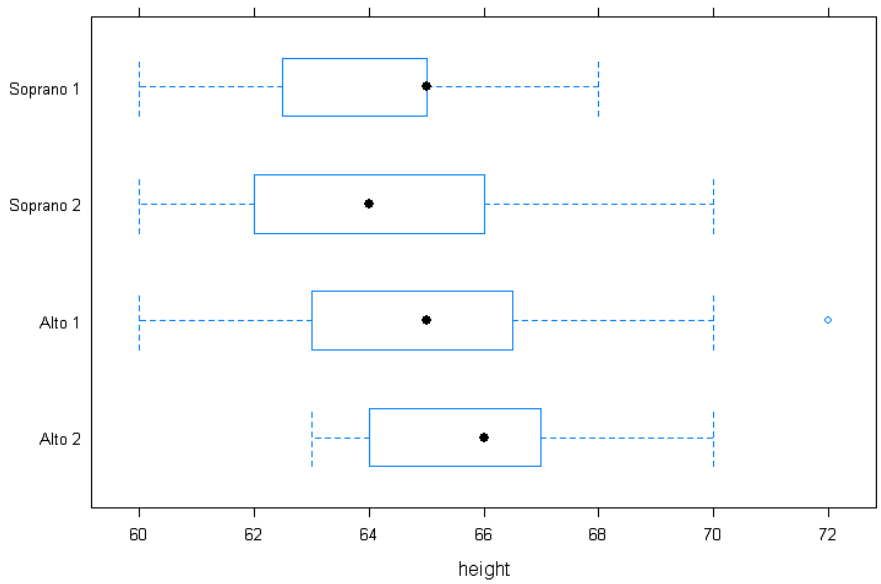


또한, box-and-whisker plot을 그려 각 집단 별 height를 다시 한 번 살펴보았다.

bwplot(voice.part ~ height, data=singer)



bwplot(voice.part ~ height, data=Alto_Sop)



검정색 점은 각 집단 별 평균값을 뜻하고 해당 코드를 통해 집단의 분포를 한눈에 볼 수 있었다. Tensor와 Bass가 포함된 경우에는 Soprano와 Alto의 집단 비교가 어렵기 때문에 상단과 같이 Soprano와 Alto만 그려보았다.
결과적으로, stem and leaf 결과를 다양한 그래프를 통해 다시 한 번 검증해보았는데 Alto 그룹이 Soprano 그룹에 비해 height(키)가 크다고 할 수 있음을 알 수 있다.
bwplot(voice.part ~ height, data=singer)의 결과(Tensor, Bass 포함)에서는 키와 음역대의 상관관계를 더 확실하게 볼 수 있었다.

4. R에서 Seatbelt 파일에 대한 설명은 '?Seatbelts' 하면 알 수 있다.
가. 운전자 1000명당, 운행거리 10000km 당 사망운전자수(killed)를 계산하여라.
killed <- DriversKilled * (drivers / 1000) * (kms / 10000)
**** attach() 함수를 쓸 수 없으면 Seatbelts[,1], Seatbelts[,2], Seatbelts[,5]를 사용하여라.**

```
> seatbelt.frame = data.frame(Seatbelts)
> attach(seatbelt.frame)
> killed = DriversKilled*(drivers/1000)*(kms/10000)
> summary(killed)
   Min. 1st Qu.  Median    Mean 3rd Qu.
  112.4   228.3   290.6   308.4   377.1
   Max.
  631.9
```

attach() 함수를 적용한 이후 운전자 1000명당, 운행거리 10000km 당 사망 운전자 수를 killed로 생성했다. summary() 함수를 통해 killed의 최소값은 112.4, 최대값은 631.9 그리고 중앙값은 290.6임을 알 수 있었다.

나. 사망운전사수의 줄기 그림을 그리고 간단히 서술하여라.

```
> stem(killed)

  The decimal point is 2 digit(s) to the right of the |

  1 | 1334
  1 | 55666667778888899999
  2 | 00000000011111122222223333333333344444444
  2 | 55555555566667777777777888888999
  3 | 00000001111222333333334444
  3 | 55556666667777778888999 9
  4 | 000011111112223444444
  4 | 55566666778999
  5 | 12233
  5 | 567
  6 | 3
```

decimal point가 오른쪽으로 2이기 때문에 자료는 100~1000단위임을 알 수 있다.
사망 운전자 수의 stem and leaf 을 통해서 분포에 왜도가 존재함을 확인했다. killed는 200명대인 stem에서 가장 많은 leaf를 확인했다. 또한, 왼쪽으로 치우친 즉, 오른쪽으로 꼬리가 길게 늘어진 모양임을 알 수 있다.

다. 안전띠 법이 시행되기 전과 후의 사망운전자수의 줄기 그림을 각각 그리고 비교하여라.

```
> # 안전띠 법 시행 이전 사망 운전자 수
> seatbelt0= seatbelt.frame[seatbelt.frame$law==0,]
> killed0 = seatbelt0$DriversKilled*(seatbelt0$drivers/1000)*(seatbelt0$kms/10000)
> stem(killed0)

  The decimal point is 2 digit(s) to the right of the |

  1 | 1334
  1 | 566667788999
  2 | 00000001111222223333333333444444444
  2 | 55555555566667777777777888888999
  3 | 000001111223333333334444
  3 | 555566667777778889999
  4 | 0000111111222344444
  4 | 5556666677899
  5 | 12233
  5 | 567
  6 | 3
```

안전띠 법 시행 이전 사망 운전자 수를 보기 위해 law==0으로 지정한 후 seatbelt0을 생성해주고 제시된 식과 같은 식을 적용해 killed0을 생성해주었다.
그리고 stem() 함수를 활용해 줄기 잎 그림을 그려 보았다.
decimal point가 오른쪽으로 2이기 때문에 자료는 100~1000단위임을 알 수 있고 stem이 2에서 가장 많은 도수가 분포하고 있음을 확인할 수 있었다.

같은 방식으로 안전띠 법 시행 이후 사망 운전자 수를 살펴보았다.

```
> # 안전띠 법 시행 이후 사망 운전자 수
> seatbelt1= seatbelt.frame[seatbelt.frame$law==1,]
> killed1 = seatbelt1$DriversKilled*(seatbelt1$drivers/1000)*(seatbelt1$kms/10000)
> stem(killed1)

  The decimal point is 2 digit(s) to the right of the |

  1 | 56788899
  2 | 0011227
  3 | 002668
  4 | 49
```

법 시행 이전과 동일한 방법으로 seatbelt1과 killed1을 생성하고 stem and leaf를 그려보았다. 법 시행 이전 사망 운전자 수 stem and leaf와 다르게 줄기의 개수가 4라는 뚜렷한 차이점을 확인할 수 있었다.

```
> stem(killed0, scale=0.5)

  The decimal point is 2 digit(s) to the right of the |

  1 | 1334566667788999
  2 | 000000011112222233333333333444444445555555556666777777777888888999
  3 | 00000111122333333333444455556666777777788899999
  4 | 000011111122234444455566667789 9
  5 | 12233567
  6 | 3
```

더 효율적인 분석을 위해 killed0로 그렸던 stem and leaf 그림의 scale 옵션을 scale=0.5로 조정하여 살펴보았다. stem 2에서 많은 도수들이 분포하고 있었으며 killed 데이터와 같이 왼쪽으로 치우친 즉, 오른쪽으로 꼬리가 길게 늘어진 모양임을 알 수 있었다.

해당 분석을 통해 killed0과 killed1의 stem and leaf를 비교해본 결과, 안전띠 법 시행 이전 줄기의 길이는 6이고 시행 이후 줄기의 길이는 4라는 것을 알 수 있었다. 결론적으로 안전띠 법 시행이 사망자 수를 감소시키는 데 영향을 주었음을 알 수 있다.

또한, 안전띠 법 시행 이전에는 stem 2 즉, 200명 대의 사망자 수에서 높은 도수를 보였고 사망자 수의 분산은 오른쪽 꼬리가 긴 형태임을 알 수 있었다.
그러나 안전띠 법 시행 이후에는 stem 1~4 즉, 100~400명 대의 사망자 수 간의 분포가 안전띠 법 시행 이전에 비해 비교적 고르게 나타남을 확인할 수 있었다.

그러나 '안전띠 법 시행 이후 사망자 수 감소'라는 성급한 결론으로 귀결시켜서는 안되는 한계점 또한 존재한다. law column에서 0 레이블과 1 레이블 사이 데이터 불균형이 있기 때문이다. 따라서 해당 결론을 내리기 위해서는 지속적으로 사망자 수가 감소하고 있는지를 주의 깊게 살펴봐야만 할 것이다.

5. 줄기그림과 히스토그램의 차이, 장단점 등을 간단히 표로 작성하여라.
강의노트, 책, 허명회 교수의 R 프로젝트 파일, 인터넷 검색 등 참조.

	정의	공통점	장점	단점
줄기그림	소규모 자료의 분포적 특성을 살펴보기 위하여 작성되는 그래프	1. 외양적인 테두리가 동일함 2. 자료의 분포 개형을 파악하는데 중요한 정보를 제공함 3. 각 구간에 속하는 자료 점의 도수에 비례하는 막대기둥	1. 자료가 크기 순서로 나열되어, 사분위수나 중간값을 쉽게 알 수 있음 2. 이상치의 여부를 쉽게 알 수 있음	1. 줄기그림은 구간폭이 정수이어야 하기 때문에 히스토그램처럼 구간폭을 임의로 정할 수 없음 2. 자료가 많은 경우에는 부적합함
히스토그램	일변량 데이터를 계급으로 나누어 각 도수를 구한 후 x축은 계급, y축은 도수를 막대의 형태로 나타낸 그림		1. 각 계급에 속하는 자료의 수를 한눈에 알아보기 쉬움 2. 자료가 많은 경우 적합함	정보손실 (information loss)이 일어날 수 있음