




[Pull requests](#) [Issues](#) [Gist](#)


  

 [kwlee58](#) / [BMI](#)


[Unwatch](#) 1 [Star](#) 0 [Fork](#) 0

[Code](#) [Issues 0](#) [Pull requests 0](#) [Wiki](#) [Pulse](#) [Graphs](#) [Settings](#)

Branch: [master](#) [BMI / R / BMI_160304.Rmd](#) [Find file](#) [Copy path](#)

 [kwlee58](#) minor correction a804cc6 21 minutes ago

1 contributor

186 lines (135 sloc) 6.89 KB [Raw](#) [Blame](#) [History](#)   

title	author	date	output
Quetelet's Body Mass Index	coop711	`r Sys.Date()`	html_document

Data

rn96.txt 자료 읽어들이기

- "../data/rn96.txt" 는 파일 경로이므로 rn96.txt 를 다운로드받은 폴더로 지정하면 되나 가급적 R source 를 저장하는 폴더(예를 들어서, 폴더 R)와 원시 자료를 저장하는 폴더(예를 들어서, 폴더 data)를 구분해 두는 것이 효과적이다. 원 자료를 보고 변수명에 해당하는 header 매개변수를 TRUE 로 설정한다. <- 는 assignment. 아래 주어진 세 가지 표현은 모두 동일한 결과를 가져온다.

```
rn96 <- read.table("../data/rn96.txt", header = TRUE)
rn96
(rn96.2 <- read.table("../data/rn96.txt", header = TRUE))
assign("rn96.3", read.table("../data/rn96.txt", header = TRUE))
rn96.3
```

- str() 은 자료의 구조를 살펴는 함수이며, value는 없다. rn96 의 자료구조를 살펴보면, int (integer) class가 나온다. 정수이지만 사칙 연산에는 아무런 영향이 없다.
- sapply() 는 rn96 의 각 변수에 mode() 또는 class() 를 적용한다.

```
str(rn96)
sapply(rn96, mode)
sapply(rn96, class)
```

Summary Statistics

- height 와 weight 의 기초통계 살펴기

```
summary(rn96)
```

- 평균과 표준편차를 각 변수별로 살펴보려면 apply() 를 이용한다. 이 때 options(digits = 2) 가 없으면 출력결과가 어떻게 달라지는지 살펴보자. 출력 결과에 이름을 붙이는 방법을 보여주고 있다.

```
options("digits")
options(digits = 2)
apply(rn96, 2, mean)
apply(rn96, 2, sd)
c(mean(rn96$height), sd(rn96$height))
c(mean(rn96$weight), sd(rn96$weight))
c(Mean = mean(rn96$height), SD = sd(rn96$height))
c(Mean = mean(rn96$weight), SD = sd(rn96$weight))
options(digits = 7)
```

Base Graphics

height 와 weight 의 산점도 그리기.

- x 축에 독립변수로 들어가는 변수와 y 축에 들어가는 종속변수를 설정하는 여러가지 방법이 있음을 알 수 있다. 동일한 `plot()` 이지만 출력 결과에서 default로 나오는 각 축의 label 값이 서로 다를 수 있다.
- 출력결과에서 도표의 크기는 `par()` 에서 조정할 수도 있고, R Studio의 R markdown 에서 조정할 수도 있다. Graphic 수업에서 보다 자세히 다를 예정이다.

```
plot(weight ~ height, data = rn96)
plot(rn96$height, rn96$weight)
plot(rn96[, 1], rn96[, 2])
```

선형회귀선 추가하기.

- 선형모형으로 적합하는 `lm()` 의 결과물을 활용하고 있다.

```
plot(weight~height, data = rn96)
abline(lm(weight ~ height, data = rn96)$coefficient)
```

- 선형모형으로 분석하기 위하여 별도의 R 오브젝트로 저장한다.

```
rn96.lm <- lm(weight ~ height, data = rn96)
```

회귀계수와 관련 통계량 살피기.

- `summary()` 를 이용하여 선형모형 분석에 등장하는 각종 통계를 살펴볼 수 있다.

```
summary(rn96.lm)
```

- 1차 회귀식으로는 살피기 힘든 국소적인 변화를 살피기 위하여 `lowess()` 를 이용한 local smoother를 추가한다.

```
plot(weight ~ height, data = rn96)
abline(lm(weight ~ height, data = rn96)$coefficient)
lines(lowess(rn96$height, rn96$weight), col = "red")
```

BMI 계산하고 줄기-잎 그리기

- 조금 편하게 작업하기 위해서 `attach()` 를 사용해 보자. `attach()` 는 `rn96` 이라는 데이터프레임을 계속해서 불러서 `rn96$height` 와 같은 형식으로 사용하는 대신에 `height` 라고 줄여서 쓸 수 있는 편리함은 있지만 사용 후 `detach()` 로 떼어내는 것을 잊지 말아야 한다. `search()` 로부터 `rn96` 이 검색 목록에 올라가 있는 것을 알 수 있다.

```
attach(rn96)
search()
```

BMI 계산

- 체질량지수라고 알려져 있는 BMI(Body Mass Index) 공식은 $\frac{\text{몸무게(kg)}}{\text{키}^2(\text{m})}$ 로 주어진다. 이는 벨기에의 수학자, 천문학 자이자 사회통계학자로 알려져 있는 아돌프 케틀레의 업적 중 하나이다. 아래 계산에서 `round()` 를 씌우지 않으면 어떤 출력이 나오는지 살펴 보고, `digits =` 를 바꿔 가며 결과를 비교해 보자. 위에서 `par(digits = 2)` 라고 설정했을 때 하교의 차이를 생각해 보자.
- `rn96` 에 BMI 계산 결과를 합쳐 보기 위해서 `cbind()` (column끼리 묶는다)를 사용하였다. 키, 몸무게, BMI가 모두 숫자변수이기 때문이다.
- `head()` , `tail()` 은 괄호 안에 들어가는 자료의 첫 6개와 끝 6개를 보여준다. 갯수를 조정하려면 `n =` 매개변수를 사용한다.

```
(BMI <- weight/(height / 100)^2)
(BMI <- round(weight/(height/100)^2, digits=1))
```

```
head(cbind(rn96, BMI))
tail(cbind(rn96, BMI), n = 10)
```

BMI 값들의 줄기-잎 그림 그리기

- John W. Tukey의 수많은 업적 중의 하나인 줄기-잎 그림은 자료의 윤곽 뿐 아니라 개별 값도 함께 파악할 수 있는 유용한 도구이다. R에서는 `stem()` 이라는 함수로 계산한다. 많이 쓰이는 매개변수로는 `scale = 2` 이 있다.

```
stem(BMI)
stem(BMI, scale=2)
```

- `weight` 와 `height` 의 줄기-잎 그림

```
stem(height)
stem(weight)
```

정규성(normality) 살펴보기

- `qqnorm()` 을 이용하여 각 변수가 정규분포에 가까운지 시각적으로 살펴보자.

```
qqnorm(weight)
qqnorm(height)
qqnorm(BMI)
```

- 검색 목록에서 `rn96` 을 떼어낼 시간이다.

```
detach()
search()
```

작업 폴더 정리하기

- `save()` 를 이용하면 작업 디렉토리에서 꼭 필요한 객체들만 모아서 저장해 놓을 수 있고, `save.image()` 를 이용하면 현재 작업 디렉토리에 있는 모든 객체를 저장하게 된다. 불러들일 때는 `load()` 를 이용한다. `rm()` 은 현재 디렉토리에 있는 객체 중에 삭제하고 싶은 것을 골라서 삭제하는 기능을 갖는다. 당연히 사용할 때 주의하여야 한다. 저장하는 다양한 방법을 살펴보자.
- 작업 history를 저장하고 나중에 편집해서 다시 활용하려면 `savehistory()` 를 이용한다.

```
ls()
save("rn96", "BMI", file = "./rn96_1.rda")
save(list = c("rn96", "BMI"), file = "./rn96_2.rda")
save(list = ls(), file = "./rn96_3.rda")
save.image(file = "./rn96_4.rda")
rm(list = ls())
ls()
load("./rn96_1.rda")
ls()
rm(list = ls())
ls()
load("./rn96_2.rda")
ls()
rm(list = ls())
ls()
load("./rn96_3.rda")
ls()
rm(list = ls())
load("./rn96_4.rda")
ls()
# savehistory(file = "./rn96.Rhistory")
```

