R Basics

Haewon Lee

Table of contents

1 Basics

1.1 R data types

1.1.1 variable type 종류

- numeric, complex

```
(numeric_value <- pi)
```

```
[1] 3.141593
```

```
(1+sqrt(2)*1i)*(1-sqrt(2)*1i)  # complex 연산
```

```
[1] 3+0i
```

```
options(digits=20) # 20자리 표현 (defualt=7)
pi
```

```
[1] 3.141592653589793116
```

- integer

```
options(digits=10)
(integer_value <- 42L) #반드시 정수라야 하는 경우에는 뒤에 L을 붙인다.
```

```
[1] 42
```

```
typeof(1+1); typeof(1L+1L); typeof(1:3)
```

```
[1] "double"
```

```
[1] "integer"
```

```
[1] "integer"
```

- logical 논리 값 TRUE | FALSE 두가지 값중 하나를 갖는다. 비교를 할 때에는 == 를 사용 (=는 input 명령이 됨)

```
1<0 ; 1>0 ; 1<"a"; "a">"A"; 3==6 #한줄에 여러 명령을 쓸 때에는 semicolon ; 로 구분
```

```
[1] FALSE
```

```
[1] TRUE
```

```
[1] TRUE
```

```
[1] FALSE
```

```
[1] FALSE
```

- charactor : “abc”, “a”, “a123xz” 등 quotation mark로 된 문자열 ” = \” 로 표기 줄바꿈 = \n

```r
letters[5:10]; paste("ab","cde", sep = "")
```

```
[1] "e" "f" "g" "h" "i" "j"
```

```
[1] "abcde"
```

```r
as.character(345); as.numeric("23.5")
```

```
[1] "345"
```

```
[1] 23.5
```

```r
sub("a","x", "father and grandpa"); gsub("a","x", "father and grandpa")
```

```
[1] "fxther and grandpa"
```

```
[1] "fxther xnd grxndpx"
```

```r
(ex2 <- 'The "R" project for statistical computing')
```

```
[1] "The \"R\" project for statistical computing"
```

- Escape characters in R :
  - `\t` Insert a tab in the text at this point.
  - `\b` Insert a backspace in the text at this point.
  - `\n` Insert a newline in the text at this point.
  - `\r` Insert a carriage return in the text at this point.
  - `\f` Insert a formfeed in the text at this point.
  - `\s` Insert a space in the text at this point.
  - `\'` Insert a single quote character in the text at this point.
  - `\"` Insert a double quote character in the text at this point.
  - `\\` Insert a backslash character in the text at this point.
- raw : used for binary data
- time : r 에서는 시간을 다루는 방법이 매우 다양함 POSIX 시간변수는 복잡한 list 형태로 되어 있음

```
(time1 <- as.POSIXlt("1960-01-01")); class(time1); typeof(time1)
```

```
[1] "1960-01-01 KST"
```

```
[1] "POSIXlt" "POSIXt"
```

```
[1] "list"
```

```
first <- "2022-08-20 08:15:22" ; second <- "2022-01-01 20:04:48"
difftime(first, second); difftime(first, second, units = "hours")
```

```
Time difference of 230.507338 days
```

```
Time difference of 5532.176111 hours
```

```
first2 <- as.POSIXlt(first); second2 <- as.POSIXlt(second)
second2 - first2
```

```
Time difference of -230.507338 days
```

```
## difftime(first, second, units = "months")
## match.arg(units)에서 다음과 같은 에러가 발생했습니다:
##  'arg' should be one of "auto", "secs", "mins", "hours", "days", "weeks"
```

1.1.2 data types

- vector : R에서는 모든 변수가 벡터 (열) 로 되어 있다. 다음 연산결과를 예상해 보시오

```
1:3 + 2:4 ; 1:10 + 1:2
```

```
[1] 3 5 7
```

```
 [1]  2  4  4  6  6  8  8 10 10 12
```

```
paste(LETTERS[1:10],1:3,sep = "-"); paste(LETTERS[1:3],1:10)
```

```
 [1] "A-1" "B-2" "C-3" "D-1" "E-2" "F-3" "G-1" "H-2" "I-3" "J-1"
```

```
 [1] "A 1"  "B 2"  "C 3"  "A 4"  "B 5"  "C 6"  "A 7"  "B 8"  "C 9"  "A 10"
```

vector의 특징은 모든 요소가 단일한 것이라는 점이다. NA 값을 제외하고는 모든 요소가 같아야 하기 때문에 서로 다른 성질의 것을 입력하게 되면 에러가 생기거나 변형된다.

```r
c(1,2,3); c(1,2,3,"a")
```

```
[1] 1 2 3
```

```
[1] "1" "2" "3" "a"
```

- array : multidimensional vector

```r
(arr1 <- array(data=1:90, dim = c(6,5,3))) # 3Dimensional array
```

```
, , 1

     [,1] [,2] [,3] [,4] [,5]
[1,]    1    7   13   19   25
[2,]    2    8   14   20   26
[3,]    3    9   15   21   27
[4,]    4   10   16   22   28
[5,]    5   11   17   23   29
[6,]    6   12   18   24   30

, , 2

     [,1] [,2] [,3] [,4] [,5]
[1,]   31   37   43   49   55
[2,]   32   38   44   50   56
[3,]   33   39   45   51   57
[4,]   34   40   46   52   58
[5,]   35   41   47   53   59
[6,]   36   42   48   54   60

, , 3

     [,1] [,2] [,3] [,4] [,5]
[1,]   61   67   73   79   85
[2,]   62   68   74   80   86
[3,]   63   69   75   81   87
[4,]   64   70   76   82   88
[5,]   65   71   77   83   89
[6,]   66   72   78   84   90
```

```r
arr1[6,4,2]  # 3Dimensional indexing
```

```
[1] 54
```

```r
which(arr1==54, arr.ind = TRUE )
```

```
     dim1 dim2 dim3
[1,]    6    4    2
```

- matrix : 2dimensional vector

```
matrix(data = c(3,4,5,6,7,8),
       nrow=2,
       ncol=3,  # nrow=2 하나만 지정해도 ncol=3은 내부적으로 결정됨
       byrow = TRUE,  # data assign 하는 방향
       dimnames = list(c("pt_1", "pt_2"),        # row names
                       c("var1","var2","var3"))  # col names
              )
```

```
     var1 var2 var3
pt_1    3    4    5
pt_2    6    7    8
```

```
x <- 2:9 ; names(x) <- x  # x의 이름을 부여
x %o% x  # = outer function : outer(x,x, FUN="*")
```

```
   2  3  4  5  6  7  8  9
2  4  6  8 10 12 14 16 18
3  6  9 12 15 18 21 24 27
4  8 12 16 20 24 28 32 36
5 10 15 20 25 30 35 40 45
6 12 18 24 30 36 42 48 54
7 14 21 28 35 42 49 56 63
8 16 24 32 40 48 56 64 72
9 18 27 36 45 54 63 72 81
```

- data frame : vector를 구성요소로 한 list의 형태 (외형적으로 보면 2dimension으로 보인다)

  dataframe의 구성요소는 vector들 (각각의 vector는 동일한 데이터 타입이라야 함)

```
## dataframe 만들기
df1 <- data.frame( col1 = c("A", "B", "Anyone", "None"),
                   col2 = c(160, 170, 180, 200),
                   col3 = c(TRUE, FALSE, FALSE, TRUE)
                   )
df1
```

```
    col1 col2  col3
1      A  160  TRUE
2      B  170 FALSE
3 Anyone  180 FALSE
4   None  200  TRUE
```

데이터프레임 이름 <- data.frame(컬럼이름= c(data_1, …. , data_n), …..) 이런 형식으로 데이터 프레임을 만들 수 있다. 데이터프레임이 R의 기본적인 데이터 양식이기 때문에 이를 다루는 방법이 다양하게 존재함

```
## dataframe cell 찾기
df1[3,2] #3행 2열의 데이터
```

```
[1] 180
```

```
## column 이름으로 찾기
df1$col1 ; df1[, "col1"]; df1["col1"] ### df1의 col1 열을 찾는 방법들
```

```
[1] "A"      "B"      "Anyone" "None"

[1] "A"      "B"      "Anyone" "None"

    col1
1      A
2      B
3 Anyone
4   None
```

```
df1[,1]
```

```
[1] "A"      "B"      "Anyone" "None"
```

- list : R에만 있는 독특한 데이터타입이다. 이것은 모든 데이터 타입을 담을 수 있는 형태이고 자료의 길이가 달라도 같이 담을 수가 있게 되어 있다. 또한 리스트 속에 리스트를 넣을 수 있기에 다단계로 nesting 되는 구조로 만들 수 있다.

```
sample_list <- list(data1=df1, data2 = arr1, data3 = x%o%x)
str(sample_list)
```

```
List of 3
 $ data1:'data.frame':	4 obs. of  3 variables:
  ..$ col1: chr [1:4] "A" "B" "Anyone" "None"
  ..$ col2: num [1:4] 160 170 180 200
  ..$ col3: logi [1:4] TRUE FALSE FALSE TRUE
 $ data2: int [1:6, 1:5, 1:3] 1 2 3 4 5 6 7 8 9 10 ...
 $ data3: num [1:8, 1:8] 4 6 8 10 12 14 16 18 6 9 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:8] "2" "3" "4" "5" ...
  .. ..$ : chr [1:8] "2" "3" "4" "5" ...
```

```
sample_list$data1
```

```
    col1 col2  col3
1      A  160  TRUE
2      B  170 FALSE
3 Anyone  180 FALSE
4   None  200  TRUE
```

```r
sample_list$data1[,3]
```

```
[1]  TRUE FALSE FALSE  TRUE
```

1.2 Built-in Data sets in R

1.2.1 소개 및 개요 : R에는 내장된 데이터세트가 있다. 테스트용, 교육용 및 연습용으로 이러한 데이터세트를 사용하면 좋다.

- 사용법

```
data("volcano") ## built-in dataset 중에서 volcano 사용
library(survival)
data(package="survival") ## survival package에 어떤 데이터 세트들이 있는지 확인
data(cancer)  ## data(cancer, package="survival") 와 같이 사용해도 된다.
str(lung) ## cancer dataset에는 다양한 암종류의 생존분석용 데이터가 들어가 있다.
```

```
'data.frame':	228 obs. of  10 variables:
 $ inst     : num  3 3 3 5 1 12 7 11 1 7 ...
 $ time     : num  306 455 1010 210 883 ...
 $ status   : num  2 2 1 2 2 1 2 2 2 2 ...
 $ age      : num  74 68 56 57 60 74 68 71 53 61 ...
 $ sex      : num  1 1 1 1 1 1 2 2 1 1 ...
 $ ph.ecog  : num  1 0 0 1 0 1 2 2 1 2 ...
 $ ph.karno : num  90 90 90 90 100 50 70 60 70 70 ...
 $ pat.karno: num  100 90 90 60 90 80 60 80 80 70 ...
 $ meal.cal : num  1175 1225 NA 1150 NA ...
 $ wt.loss  : num  NA 15 15 11 0 0 10 1 16 34 ...
```

- rotterdam : breast cancer dataset in survival package

```
library(moonBook)
mytable(grade~. , data=rotterdam)
```

```
       Descriptive Statistics by 'grade'
________________________________________________
                   2              3           p
                (N=794)        (N=2188)
------------------------------------------------
 pid        1328.4 ± 865.1  1569.0 ± 860.9  0.000
 year        1987.9 ±   3.1   1988.3 ±   3.0  0.004
 age          54.4 ± 12.7     55.3 ± 13.1   0.086
 meno                                       0.000
   - 0        392 (49.4%)     920 (42.0%)
   - 1        402 (50.6%)    1268 (58.0%)
 size                                       0.000
   - <=20     462 (58.2%)     925 (42.3%)
   - 20-50    290 (36.5%)    1001 (45.7%)
   - >50      42 ( 5.3%)      262 (12.0%)
 nodes         2.0 ±   3.7      3.0 ±   4.6   0.000
 pgr         236.2 ± 385.8   134.9 ± 242.8  0.000
 er          179.8 ± 291.9   161.8 ± 265.0  0.127
```

```
hormon                                         0.000
  - 0        735 (92.6%)    1908 (87.2%)
  - 1        59 ( 7.4%)      280 (12.8%)
chemo                                          1.000
  - 0        640 (80.6%)    1762 (80.5%)
  - 1        154 (19.4%)     426 (19.5%)
rtime     2458.3 ± 1408.6 1967.1 ± 1370.8 0.000
recur                                          0.000
  - 0        480 (60.5%)     984 (45.0%)
  - 1        314 (39.5%)    1204 (55.0%)
dtime     2908.9 ± 1278.6 2495.2 ± 1287.8 0.000
death                                          0.000
  - 0        532 (67.0%)    1178 (53.8%)
  - 1        262 (33.0%)    1010 (46.2%)
-------------------------------------------------
```

```r
mytable(grade~. , data=rotterdam) %>% mylatex() %>% cat
```

Descriptive Statistics by grade

	2 (N=794)	3 (N=2188)	p
pid	1328.4 ± 865.1	1569.0 ± 860.9	0.000
year	1987.9 ± 3.1	1988.3 ± 3.0	0.004
age	54.4 ± 12.7	55.3 ± 13.1	0.086
meno			0.000
- 0	392 (49.4%)	920 (42.0%)	
- 1	402 (50.6%)	1268 (58.0%)	
size			0.000
- <=20	462 (58.2%)	925 (42.3%)	
- 20-50	290 (36.5%)	1001 (45.7%)	
- >50	42 (5.3%)	262 (12.0%)	
nodes	2.0 ± 3.7	3.0 ± 4.6	0.000
pgr	236.2 ± 385.8	134.9 ± 242.8	0.000
er	179.8 ± 291.9	161.8 ± 265.0	0.127
hormon			0.000
- 0	735 (92.6%)	1908 (87.2%)	
- 1	59 (7.4%)	280 (12.8%)	
chemo			1.000
- 0	640 (80.6%)	1762 (80.5%)	
- 1	154 (19.4%)	426 (19.5%)	
rtime	2458.3 ± 1408.6	1967.1 ± 1370.8	0.000
recur			0.000
- 0	480 (60.5%)	984 (45.0%)	
- 1	314 (39.5%)	1204 (55.0%)	
dtime	2908.9 ± 1278.6	2495.2 ± 1287.8	0.000
death			0.000
- 0	532 (67.0%)	1178 (53.8%)	
- 1	262 (33.0%)	1010 (46.2%)	

```r
## LaTeX을 이용하여 깔끔한 논문형식의 테이블을 만들 수 있다.
```

1.3 Basic statistics functions

1.3.1 t-test

R function : t.test -

option arguments : alternative = c("two.sided", "less", "greater"), formula (종속변수~ 독립변수)

help files : ?t.test 를 치면 함수의 argument, values(results), detail에 대해서 설명이 나옴

```
group1 <- rotterdam[ rotterdam$grade == 2, "age"]
group2 <- rotterdam[ rotterdam$grade != 2, "age"]
t.test(group1, group2)    ## unmatched 임의의 두개의 vector로 비교
```

```
    Welch Two Sample t-test

data:  group1 and group2
t = -1.7436947, df = 1444.4213, p-value = 0.08142509
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.9599008823  0.1152640033
sample estimates:
  mean of x   mean of y
54.38161209 55.30393053
```

```
t.test(age~meno,data=rotterdam)   ## matched 한개의 데이터프레임에서 paired t-test
```

```
    Welch Two Sample t-test

data:  age by meno
t = -76.545414, df = 2972.8397, p-value < 2.2204e-16
alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0
95 percent confidence interval:
 -21.53192159 -20.45636342
sample estimates:
mean in group 0 mean in group 1
    43.30106707     64.29520958
```

1.3.2 χ^2 (chi-square) test

R function : chisq.test

```
table(rotterdam[,c("hormon","size")])
```

```
      size
hormon <=20 20-50  >50
     0 1283  1119  241
     1  104   172   63
```

```
chisq.test(table(rotterdam[,c("hormon","size")]), correct = TRUE)
```

```
	Pearson's Chi-squared test

data:  table(rotterdam[, c("hormon", "size")])
X-squared = 51.920064, df = 2, p-value = 5.317424e-12
```

```
chisq.test(rotterdam$hormon, rotterdam$chemo)
```

```
	Pearson's Chi-squared test with Yates' continuity correction

data:  rotterdam$hormon and rotterdam$chemo
X-squared = 29.771106, df = 1, p-value = 4.861842e-08
```

```
x <- matrix(c(12, 5, 7, 7), ncol = 2)  ## matrix를 만들어서 검정하는 방법
x
```

```
     [,1] [,2]
[1,]   12    7
[2,]    5    7
```

```
chisq.test(x)$p.value   ## chisq test의 결과물은 list이다 여기서 p.value 부분만 출력
```

```
[1] 0.4233054243
```

```
chisq.test(x, simulate.p.value = TRUE, B = 10000)$p.value
```

```
[1] 0.2860713929
```

1.3.3 generalized linear regression and Loess smoothing (LOcal regrESSion)

R function : glm (generalized linear models) 다중 선형회귀

```
data(economics, package="ggplot2")
economics$index <- 1:nrow(economics) # create index variable
glm_model1 <- glm(psavert~pop, data = economics)
summary(glm_model1)
```

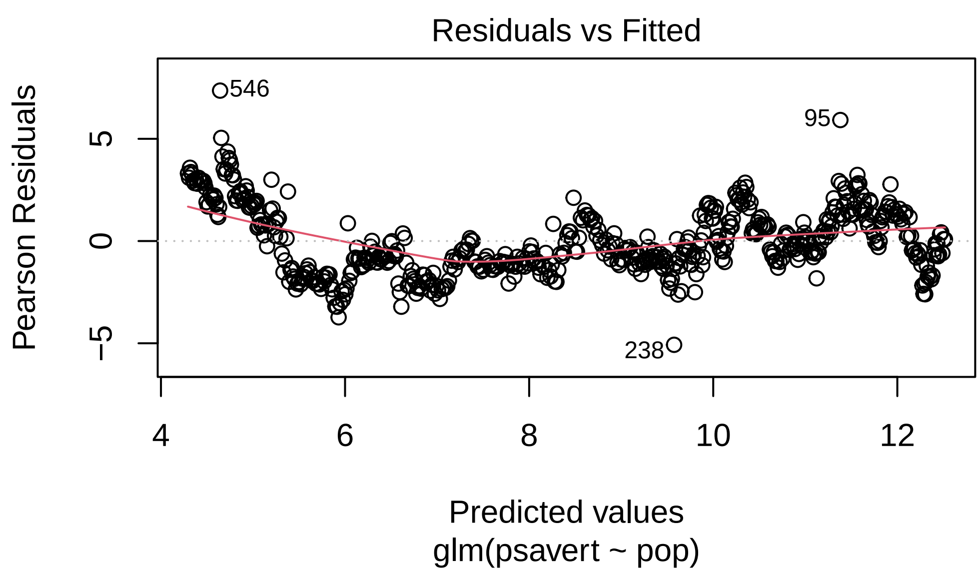
```
Call:
glm(formula = psavert ~ pop, data = economics)

Coefficients:
                 Estimate    Std. Error   t value   Pr(>|t|)
(Intercept)  2.594603e+01  4.811677e-01  53.92305 < 2.22e-16 ***
pop         -6.757974e-05  1.852367e-06 -36.48290 < 2.22e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 2.645600898)
```

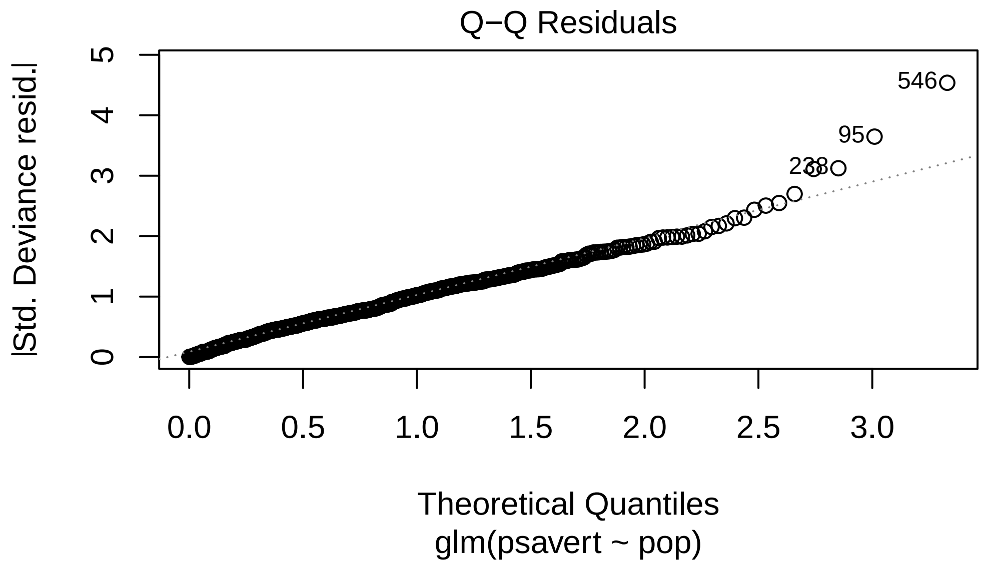
```
    Null deviance: 5034.5843  on 573  degrees of freedom
Residual deviance: 1513.2837  on 572  degrees of freedom
AIC: 2191.3815

Number of Fisher Scoring iterations: 2
```

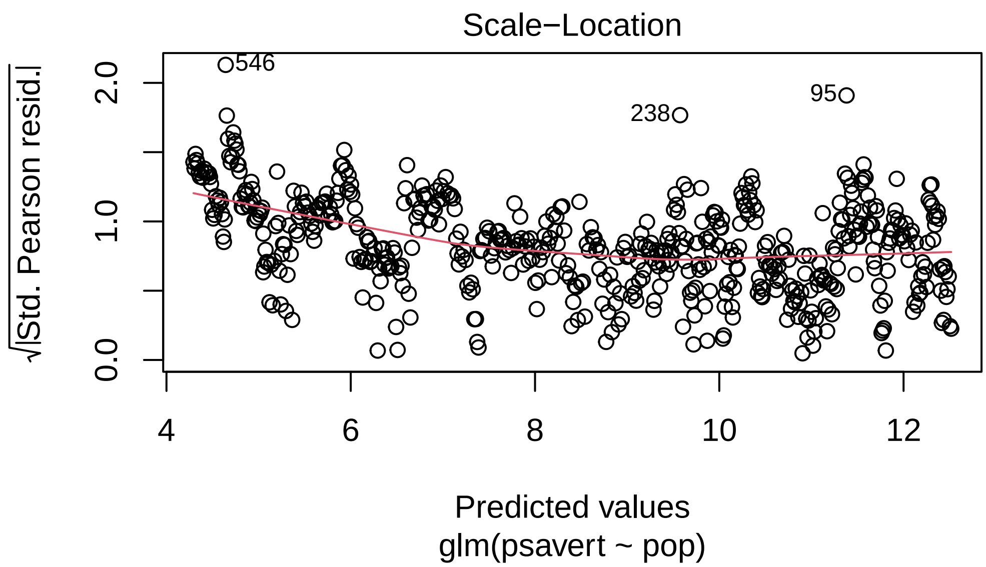
```
anova(glm_model1)
```

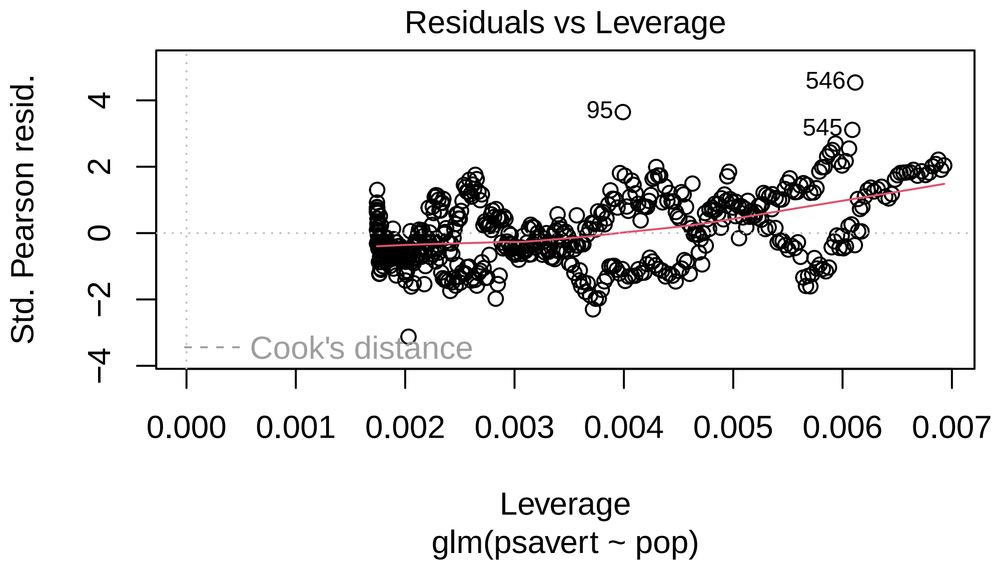
```
Analysis of Deviance Table

Model: gaussian, link: identity

Response: psavert

Terms added sequentially (first to last)


     Df  Deviance Resid. Df Resid. Dev
NULL                    573  5034.5843
pop   1 3521.3005       572  1513.2837
```

```
plot(glm_model1)
```

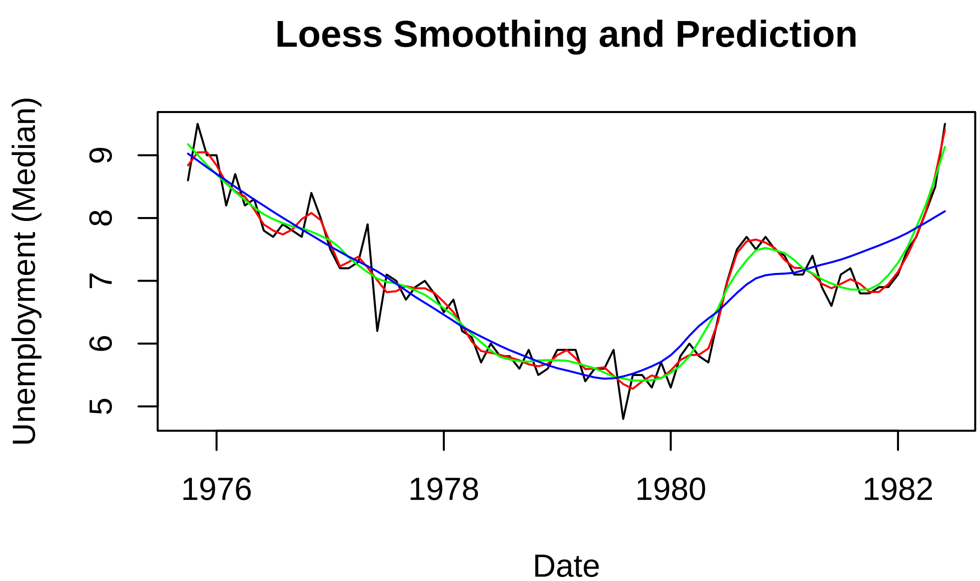



Q−Q Residuals
|Std. Deviance resid.|
0
1
2
3
4
5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
546
95
238
Theoretical Quantiles
glm(psavert ~ pop)

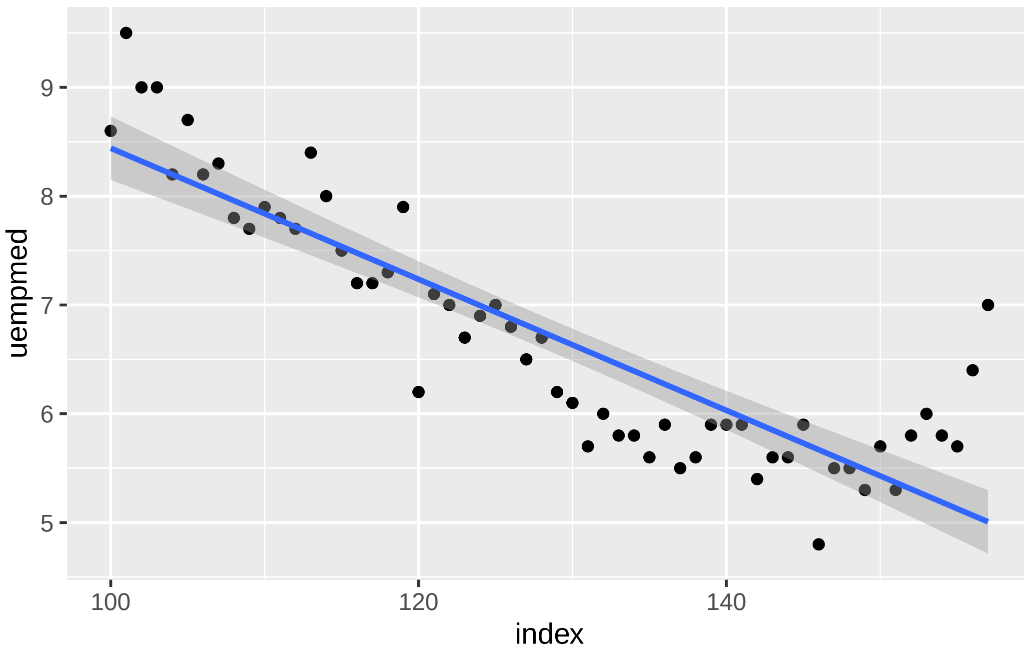

Scale−Location
√|Std. Pearson resid.|
0.0
1.0
2.0
4
6
8
10
12
546
238
95
Predicted values
glm(psavert ~ pop)




```
economics <- economics[100:180, ] ### narrow span
loessMod10 <- loess(uempmed ~ index, data=economics, span=0.10) # 10% smoothing span
loessMod25 <- loess(uempmed ~ index, data=economics, span=0.25)
loessMod50 <- loess(uempmed ~ index, data=economics, span=0.50)
smoothed10 <- predict(loessMod10)
smoothed25 <- predict(loessMod25)
smoothed50 <- predict(loessMod50)
plot(economics$uempmed, x=economics$date, type="l", main="Loess Smoothing and Prediction", xlab="Date",
lines(smoothed10, x=economics$date, col="red")
lines(smoothed25, x=economics$date, col="green")
lines(smoothed50, x=economics$date, col="blue")
```



```r
economics <- economics[1:58,]
library(ggplot2)
ggplot(data=economics, aes(x=index, y=uempmed))+
  geom_point()+
  geom_smooth(method = "lm")
```



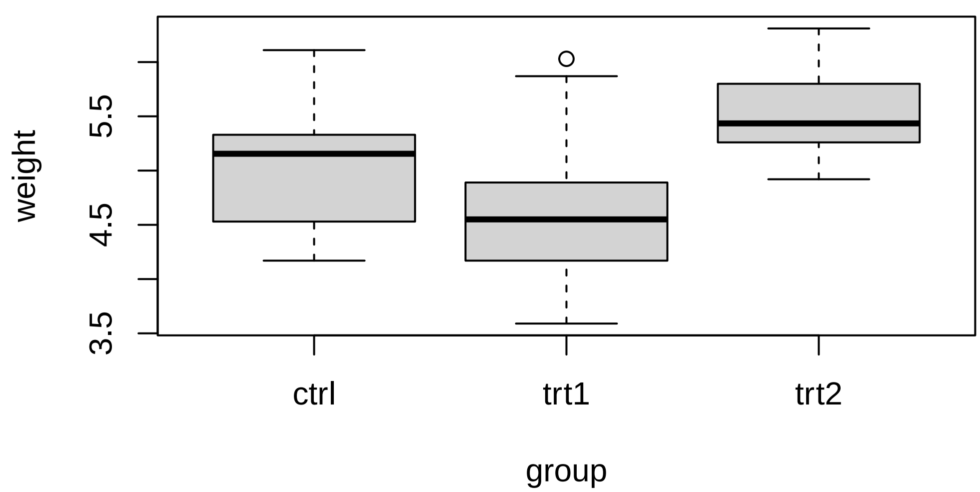
1.3.4 One-way ANOVA

```r
library(psych)
PlantGrowth  ## 내장 dataset
```

```
   weight group
1    4.17  ctrl
2    5.58  ctrl
3    5.18  ctrl
4    6.11  ctrl
5    4.50  ctrl
6    4.61  ctrl
7    5.17  ctrl
8    4.53  ctrl
9    5.33  ctrl
10   5.14  ctrl
11   4.81  trt1
12   4.17  trt1
13   4.41  trt1
14   3.59  trt1
15   5.87  trt1
```

```
16   3.83  trt1
17   6.03  trt1
18   4.89  trt1
19   4.32  trt1
20   4.69  trt1
21   6.31  trt2
22   5.12  trt2
23   5.54  trt2
24   5.50  trt2
25   5.37  trt2
26   5.29  trt2
27   4.92  trt2
28   6.15  trt2
29   5.80  trt2
30   5.26  trt2
```

```r
plot(weight~group, data = PlantGrowth)  ## Boxplot으로 자동으로 그려준다.
```




```r
with(PlantGrowth, describeBy(weight,group))
```

```
 Descriptive statistics by group
group: ctrl
   vars  n mean   sd median trimmed  mad  min  max range skew kurtosis   se
X1    1 10 5.03 0.58   5.15       5 0.72 4.17 6.11  1.94 0.23    -1.12 0.18
------------------------------------------------------------
group: trt1
   vars  n mean   sd median trimmed  mad  min  max range skew kurtosis   se
X1    1 10 4.66 0.79   4.55    4.62 0.53 3.59 6.03  2.44 0.47     -1.1 0.25
------------------------------------------------------------
group: trt2
   vars  n mean   sd median trimmed  mad  min  max range skew kurtosis   se
X1    1 10 5.53 0.44   5.44     5.5 0.36 4.92 6.31  1.39 0.48    -1.16 0.14
```

```r
bartlett.test(PlantGrowth$weight ~ PlantGrowth$group)  ## 등분산 가정을 체크함
```

```
	Bartlett test of homogeneity of variances

data:  PlantGrowth$weight by PlantGrowth$group
Bartlett's K-squared = 2.8785738, df = 2, p-value = 0.2370968
```

```r
aov_model <- aov(weight~group, data = PlantGrowth)
summary(aov_model)
```

```
            Df   Sum Sq   Mean Sq F value  Pr(>F)  
group        2  3.76634 1.8831700 4.84609 0.01591 *
Residuals   27 10.49209 0.3885959                  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

1.3.5 Correlation tests

Pearson correlation formula

$$r = \frac{\sum(x - m_x)(y - m_y)}{\sqrt{\sum(x - m_x)^2 \sum(y - m_y)^2}}$$

Spearman correlation formula : non-parametric

$$\rho = \frac{\sum(x' - m_{x'})(y' - m_{y'})}{\sqrt{\sum(x' - m_{x'})^2 \sum(y' - m_{y'})^2}}$$

$$\text{where } x' = rank(x) \quad \text{and} \quad y' = rank(y)$$

Kendall correlation formula : non-parametric

$$\tau = \frac{n_c - n_d}{\frac{1}{2}n(n-1)}$$

$$\text{where} \quad n_c : \text{number of concordant pairs,} \quad n_d : \text{number of concordant pairs,} \quad n : \text{size of } x + y$$

```r
res <- cor.test(economics$index, economics$uempmed, method = "pearson")
res
```

```
	Pearson's product-moment correlation

data:  economics$index and economics$uempmed
t = -13.654567, df = 56, p-value < 2.2204e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
```

```
 -0.9255815657 -0.7998071541
sample estimates:
         cor
-0.8769389168
```

```r
res$p.value  ## res는 리스트형태로 나오는 결과물이다. 여기에서 필요한 값만 골라냄
```

```
[1] 1.822639345e-19
```

```r
res$estimate
```

```
         cor
-0.8769389168
```

```r
res2 <- cor.test(economics$index, economics$uempmed, method = "spearman")
res2
```

```
    Spearman's rank correlation rho

data:  economics$index and economics$uempmed
S = 60517.721, p-value < 2.2204e-16
alternative hypothesis: true rho is not equal to 0
sample estimates:
        rho
-0.861568223
```

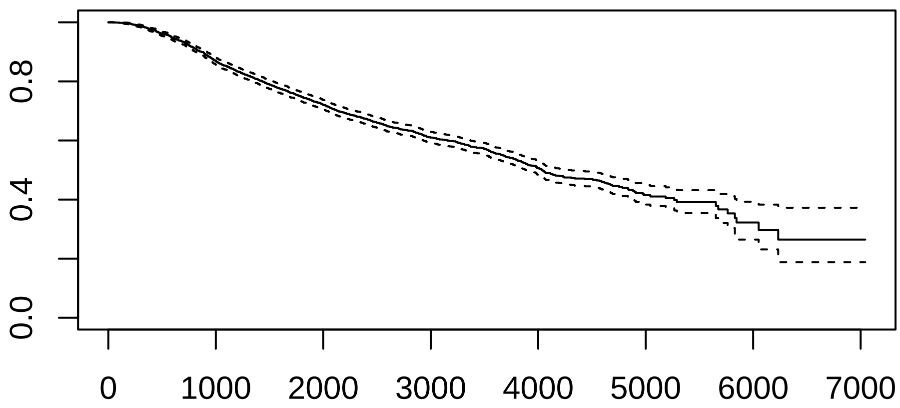
```r
res3 <- cor.test(economics$index, economics$uempmed, method = "kendall")
res3
```

```
    Kendall's rank correlation tau

data:  economics$index and economics$uempmed
z = -7.8580733, p-value = 3.90087e-15
alternative hypothesis: true tau is not equal to 0
sample estimates:
         tau
-0.7167487038
```

1.3.6 Survival analysis

- Kaplan Meier Analysis - Basic survival model survival::Surv

```r
km <- Surv(rotterdam$dtime, event = rotterdam$death)  ## default type : "right"
plot(km)  ## km - Surv class (time, status) 가지고 있는 리스트
```




```r
median(km); mean(km)  ## Surv 객체에 대한 method 함수들이 있다. plot.Surv포함
```

```
$quantile
  50 
4033 

$lower
  50 
3888 

$upper
  50 
4309 

[1] 1302.8833
```

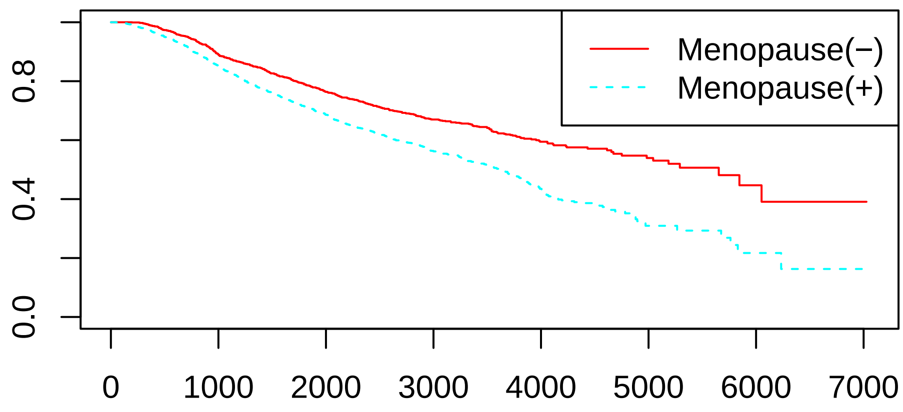
- Kaplan Meier Analysis - survfit model

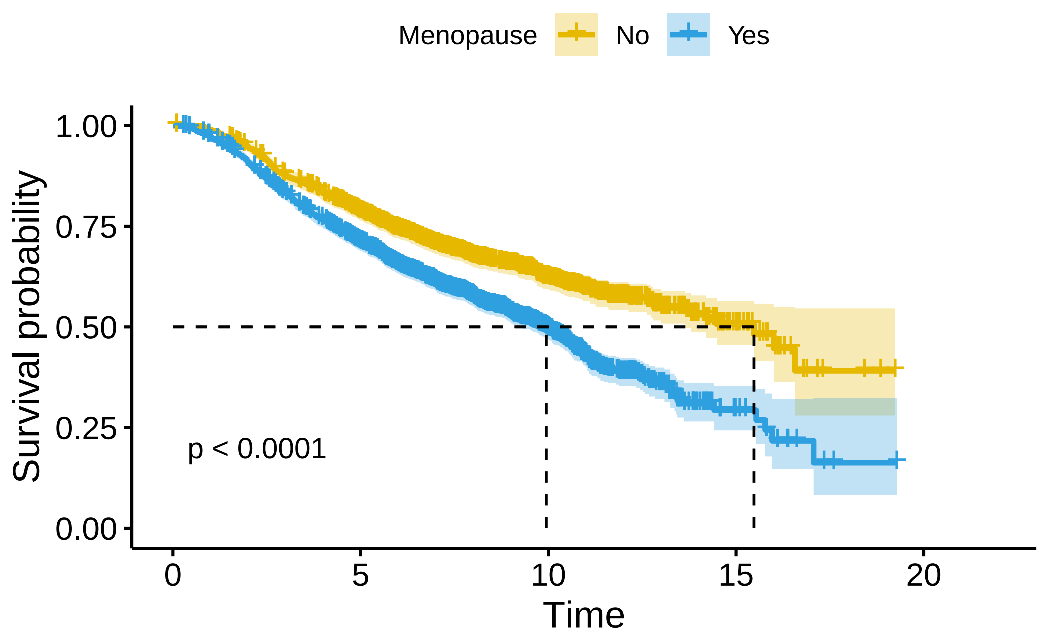
```r
km_fit <- survfit(km~rotterdam$meno)
summary(km_fit, c(365*1:19))   ### 정해진 time에 맞는 생존테이블표를 만든다.
```

```
Call: survfit(formula = km ~ rotterdam$meno)

                rotterdam$meno=0 
 time n.risk n.event survival    std.err lower 95% CI upper 95% CI
  365   1298      13 0.990084 0.00273656     0.984735     0.995462
  730   1236      56 0.947311 0.00617381     0.935287     0.959489
 1095   1140      90 0.878196 0.00905336     0.860630     0.896121
 1460   1071      59 0.832587 0.01034896     0.812549     0.853120
 1825    973      59 0.786049 0.01140771     0.764005     0.808729
 2190    865      50 0.744541 0.01222354     0.720964     0.768888
 2555    754      43 0.706203 0.01291841     0.681332     0.731982
 2920    611      31 0.674528 0.01353781     0.648509     0.701590
 3285    480      15 0.656234 0.01397343     0.629410     0.684201
 3650    345      21 0.622823 0.01505424     0.594005     0.653039
 4015    217      13 0.594613 0.01631412     0.563482     0.627463
```

```
4380    138       6 0.575323 0.01759653     0.541848     0.610866
4745     88       4 0.553799 0.01999709     0.515960     0.594412
5110     54       3 0.530422 0.02334386     0.486587     0.578207
5475     29       2 0.506485 0.02783275     0.454769     0.564082
5840     14       1 0.481161 0.03617160     0.415241     0.557545
6205      5       2 0.390943 0.06657871     0.279996     0.545853
6570      3       0 0.390943 0.06657871     0.279996     0.545853
6935      1       0 0.390943 0.06657871     0.279996     0.545853

                rotterdam$meno=1
time n.risk n.event survival    std.err lower 95% CI upper 95% CI
 365   1616      46 0.972378 0.00401599    0.9645389     0.980281
 730   1508     103 0.910256 0.00701496    0.8966099     0.924109
1095   1366     129 0.832077 0.00918891    0.8142608     0.850283
1460   1245     111 0.764188 0.01045754    0.7439639     0.784962
1825   1111      87 0.709950 0.01121688    0.6883018     0.732278
2190    944      82 0.655456 0.01186297    0.6326122     0.679124
2555    819      58 0.613767 0.01230810    0.5901113     0.638371
2920    642      45 0.577382 0.01272281    0.5529763     0.602864
3285    474      42 0.535877 0.01333692    0.5103646     0.562665
3650    342      31 0.495578 0.01418038    0.4685496     0.524165
4015    188      35 0.430288 0.01613537    0.3997973     0.463104
4380    113      17 0.386353 0.01771621    0.3531444     0.422684
4745     62       6 0.357899 0.01988720    0.3209686     0.399079
5110     28       7 0.309356 0.02431136    0.2651946     0.360871
5475     14       1 0.293074 0.02795732    0.2430961     0.353326
5840      8       3 0.217092 0.04323095    0.1469392     0.320737
6205      4       0 0.217092 0.04323095    0.1469392     0.320737
6570      1       1 0.162819 0.05710016    0.0818823     0.323757
6935      1       0 0.162819 0.05710016    0.0818823     0.323757
```

```r
plot(km_fit, col = rainbow(2), lty=1:2)
legend("topright", legend = c("Menopause(-)","Menopause(+)"),
       col= rainbow(2), lty=1:2)
library(survminer)
ggsurvplot(km_fit, data = rotterdam,
           conf.int = T, xscale = 365.2425,  ## xscale can be "d_y"
           break.x.by = 5*365.2425,
           pval = T, pval.size =4, surv.median.line = "hv",
           risk.table = FALSE, ## if TRUE, risk table is displayed under graph
           legend.title="Menopause", legend.labs=c("No","Yes"),
           palette = c("#E7B800", "#2E9FDF"),)
```






```
## ggplot + surviminer package
```

- Cox Proportional model

$$hazard\ function\ h(t) = \lim_{\Delta t \to 0} \frac{\Pr[(t \leq T < t + \Delta t | T \geq t)]}{\Delta t} \quad = \quad \frac{p(t)}{S(t)}$$

$$\log h_i(t) = \alpha + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_k x_{ik}$$

```
args(coxph)
```

```
function (formula, data, weights, subset, na.action, init, control,
    ties = c("efron", "breslow", "exact"), singular.ok = TRUE,
    robust, model = FALSE, x = FALSE, y = TRUE, tt, method = ties,
    id, cluster, istate, statedata, nocenter = c(-1, 0, 1), ...)
NULL
```

```r
library(carData)   ## Rossi data set 이용하기 위해서 사용
library(car)       ## Anova function
colnames(Rossi)  # emp1-52 : factor (yes or no)
```

```
 [1] "week"   "arrest" "fin"    "age"    "race"   "wexp"   "mar"    "paro"  
 [9] "prio"   "educ"   "emp1"   "emp2"   "emp3"   "emp4"   "emp5"   "emp6"  
[17] "emp7"   "emp8"   "emp9"   "emp10"  "emp11"  "emp12"  "emp13"  "emp14" 
[25] "emp15"  "emp16"  "emp17"  "emp18"  "emp19"  "emp20"  "emp21"  "emp22" 
[33] "emp23"  "emp24"  "emp25"  "emp26"  "emp27"  "emp28"  "emp29"  "emp30" 
[41] "emp31"  "emp32"  "emp33"  "emp34"  "emp35"  "emp36"  "emp37"  "emp38" 
[49] "emp39"  "emp40"  "emp41"  "emp42"  "emp43"  "emp44"  "emp45"  "emp46" 
[57] "emp47"  "emp48"  "emp49"  "emp50"  "emp51"  "emp52" 
```

```r
cox_model1 <- coxph(Surv(week, arrest) ~
                     fin + age +  race + wexp + mar + paro + prio,
                     data = Rossi)
summary(cox_model1)
```

```
Call:
coxph(formula = Surv(week, arrest) ~ fin + age + race + wexp + 
    mar + paro + prio, data = Rossi)

  n= 432, number of events= 114 

                      coef   exp(coef)    se(coef)        z  Pr(>|z|)   
finyes         -0.37942217  0.68425668  0.19137948 -1.98256 0.0474161 * 
age            -0.05743774  0.94418067  0.02199947 -2.61087 0.0090312 **
raceother      -0.31389979  0.73059224  0.30799278 -1.01918 0.3081180   
wexpyes        -0.14979570  0.86088384  0.21222430 -0.70584 0.4802897   
marnot married  0.43370388  1.54296190  0.38186806  1.13574 0.2560642   
paroyes        -0.08487108  0.91863070  0.19575667 -0.43355 0.6646124   
prio            0.09149708  1.09581358  0.02864855  3.19378 0.0014042 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

               exp(coef) exp(-coef) lower .95 upper .95
finyes         0.6842567  1.4614399 0.4702367 0.9956841
age            0.9441807  1.0591193 0.9043345 0.9857825
raceother      0.7305922  1.3687526 0.3994948 1.3361001
wexpyes        0.8608838  1.1615969 0.5679354 1.3049390
marnot married 1.5429619  0.6481041 0.7299759 3.2613836
paroyes        0.9186307  1.0885767 0.6259110 1.3482466
prio           1.0958136  0.9125640 1.0359791 1.1591039

Concordance= 0.64  (se = 0.027 )
Likelihood ratio test= 33.27  on 7 df,   p=2.36e-05
Wald test            = 32.11  on 7 df,   p=3.87e-05
```

```
Score (logrank) test = 33.53  on 7 df,   p=2.11e-05
```

```r
Anova(cox_model1)
```

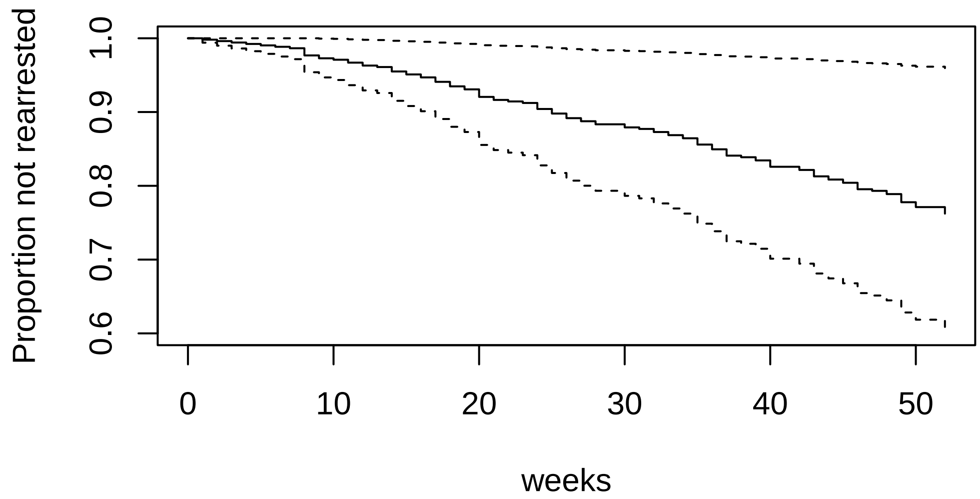
```
Analysis of Deviance Table (Type II tests)
      LR Chisq Df Pr(>Chisq)
fin  3.9862101  1  0.0458741 *
age  7.9880173  1  0.0047088 **
race 1.1251518  1  0.2888118
wexp 0.5003372  1  0.4793520
mar  1.4311793  1  0.2315721
paro 0.1869702  1  0.6654503
prio 8.9765972  1  0.0027346 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
anova(cox_model1)
```

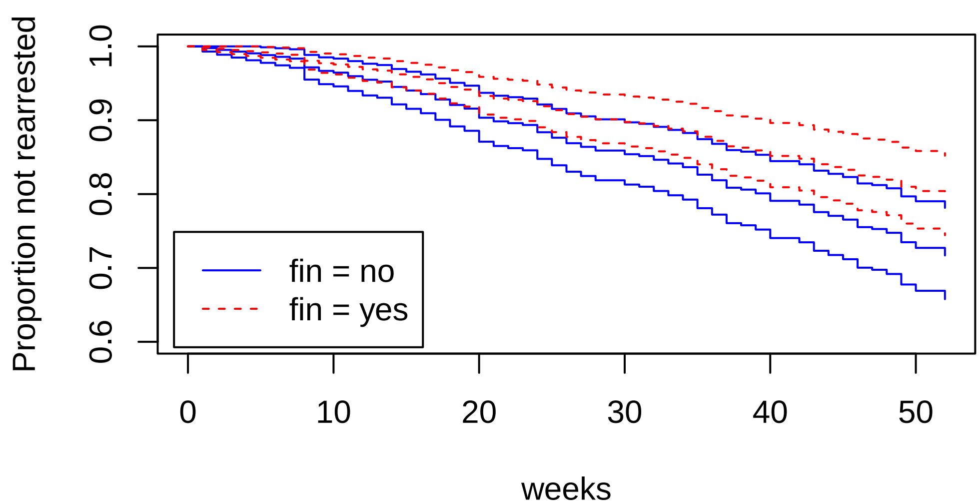
```
Analysis of Deviance Table
 Cox model: response is Surv(week, arrest)
Terms added sequentially (first to last)

         loglik    Chisq Df Pr(>|Chi|)
NULL -675.38063
fin  -673.46210  3.83706  1 0.05013146 .
age  -666.23582 14.45257  1 0.00014373 ***
race -665.84148  0.78867  1 0.37450208
wexp -664.21789  3.24717  1 0.07154674 .
mar  -663.57584  1.28411  1 0.25713587
paro -663.23596  0.67976  1 0.40966904
prio -658.74766  8.97660  1 0.00273459 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

모델의 전체적인 생존곡선을 알고 싶으면 survfit 함수를 이용해서 생존곡선을 그릴 수 있다.

```r
plot(survfit(cox_model1), ylim = c(0.6,1),xlab = "weeks",
     ylab = "Proportion not rearrested")
```




```r
Rossi.fin <- with(Rossi, data.frame(fin=c(0, 1),
        age=rep(mean(age), 2), race=rep(mean(race == "other"), 2),
        wexp=rep(mean(wexp == "yes"), 2), mar=rep(mean(mar == "not married"), 2),
        paro=rep(mean(paro == "yes"), 2), prio=rep(mean(prio), 2)))
## fin = 0,1 이것을 두그룹으로 나누고 나머지 변수들은 평균적인 값으로 고정해 버림
plot(survfit(cox_model1, newdata = Rossi.fin), conf.int = T,
     lty = c(1,2), ylim = c(0.6,1),xlab = "weeks",
     ylab = "Proportion not rearrested", col = c("blue","red")
     )
legend("bottomleft", legend=c("fin = no", "fin = yes"), lty=c(1 ,2),
       col=c("blue","red") , inset=0.02)
```



2 Advanced Techniques

2.1 Data Manipulation

2.1.1 Data reading

data file이 존재하는 디렉토리를 먼저 설정해주어야 한다. 이를 위한 명령어는 setwd() = set working directory 라는 의미 setwd("G:/R project/nomogram") 와 같이 디렉토리를 설정해줄 수도 있지만,

만약 디렉토리를 찾기 어렵다면 setwd(choose.dir()) 와 같은 명령으로 파일탐색기를 열어서 디렉토리를 선택할 수 있다. 현재 사용할 xlsx 파일들이 다음 디렉토리에 있다고 가정하자

```
setwd("G:/R project/nomogram")
library(readxl)
dir(pattern = "*.xls")
```

```
 [1] "datasummary.xlsx"    "Patient_info.xls"    "Survdata.xls"
 [4] "폐암-항암.xlsx"      "폐암 -환자정보.xlsx" "폐암 op.xlsx"
 [7] "폐암 RT.xlsx"        "폐암_OP.xls"         "폐암_RT.xls"
[10] "폐암_추적조사.xls"   "폐암_항암치료.xls"   "폐암환자 통계.xlsx"
```

```
xlsxfiles <- dir(pattern = "*.xls")
ptinfo <- read_xlsx(xlsxfiles[5])
```

2.1.2 Binding tables

데이터프레임 결합 방법들

rbind(), cbind(), merge()

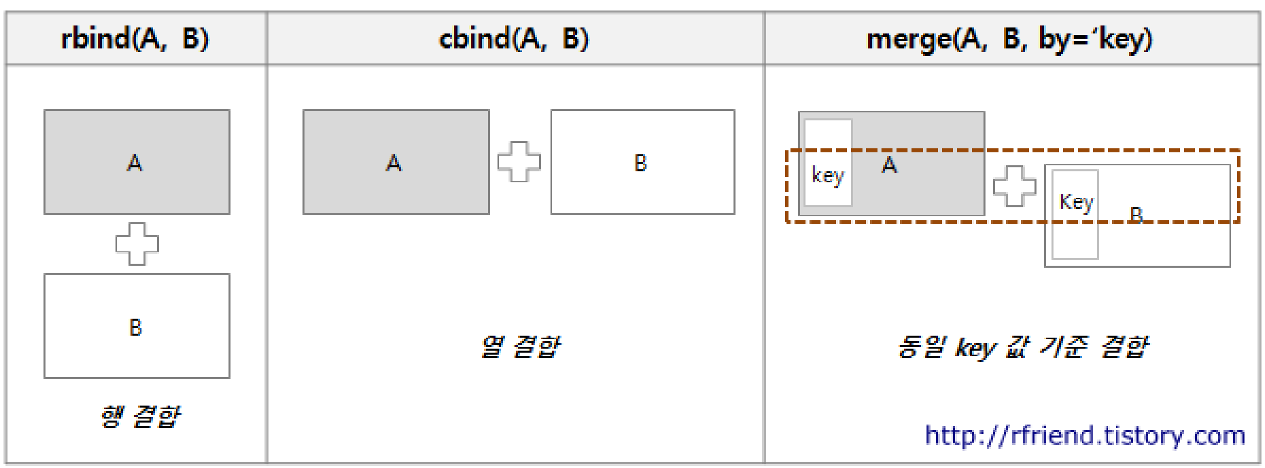


Figure 1: 데이터프레임 결합방법

** 당연한 이야기지만 rbind는 컬럼의 갯수가 같아야 하고, cbind는 행의 갯수가 같아야 함

2.1.3 Join (Merge) tables

merge function

merge(x, y, by = intersect(names(x), names(y)), ## 공통된 컬럼하나를 결합용 키로 선택

by.x = by, by.y = by, all = FALSE, all.x = all, all.y = all, ## x와 y의 결합용 키의 이름이 서로 다를 경우에는 독립적으로 지정

sort = TRUE, suffixes = c(".x",".y"), no.dups = TRUE,

incomparables = NULL, ...)

```r
df1 <- data.frame( ID = 1:10, Name = c("Lee","Kim","Park", "Kang",
                   "Shin", "Lim", "Kwon", "Choi", "Nam", "Baek" ),
                   Score = as.integer(rnorm(10, 80,6 ))
                   )
df2 <- data.frame( ID = sample(1:10, 9, replace = F),
                   Department = sample( c("IM","GS","GY","PD" ),9, replace = T),
                   Age = as.integer(rnorm(9, 40,6 )) )
df1
```

```
   ID Name Score
1   1  Lee    80
2   2  Kim    81
3   3 Park    82
4   4 Kang    80
5   5 Shin    79
6   6  Lim    92
7   7 Kwon    82
8   8 Choi    81
9   9  Nam    76
10 10 Baek    79
```

```r
df2
```

```
  ID Department Age
1  3         IM  39
2  6         GY  44
3  8         IM  37
4  5         PD  47
5  2         IM  36
6  4         GY  40
7  7         PD  40
8 10         PD  38
9  9         IM  37
```

```r
merged_df <- merge(df1,df2, by="ID", all = TRUE)  # full join
merged_df
```

```
   ID Name Score Department Age
1   1  Lee    80       <NA>  NA
2   2  Kim    81         IM  36
3   3 Park    82         IM  39
4   4 Kang    80         GY  40
5   5 Shin    79         PD  47
6   6  Lim    92         GY  44
7   7 Kwon    82         PD  40
8   8 Choi    81         IM  37
9   9  Nam    76         IM  37
10 10 Baek    79         PD  38
```

2.1.4 Types of Join

merge 함수를 실행하여 데이터를 결합할 때에는 데이터 join 방법이 다음과 같이 4가지가 있다.

두개의 df에서 모든 데이터가 완전하게 존재하지 않기 때문에 일치하지 않는 부분에 대한 처리규칙이 중요하다.

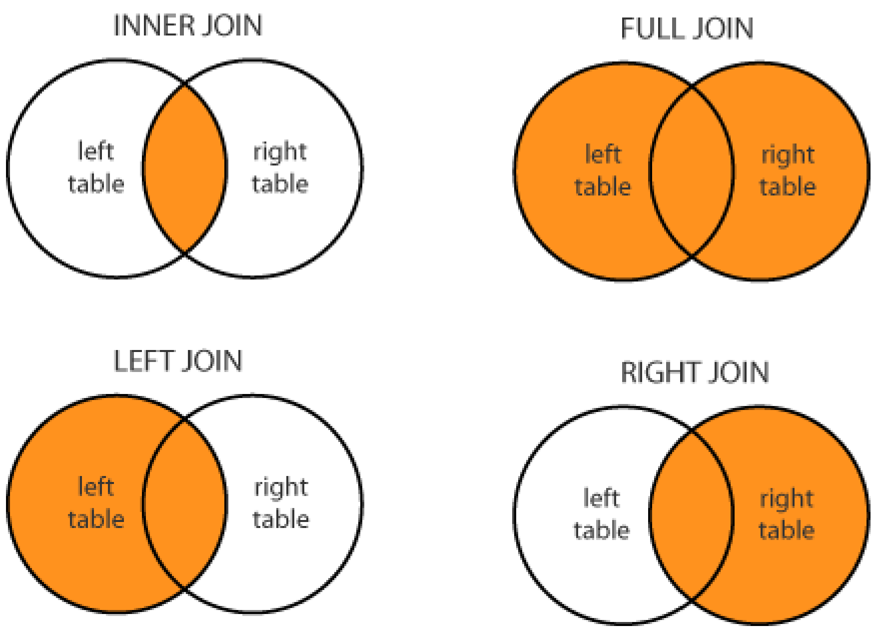


Figure 2: Types of Join

merge 함수의 옵션에서 all = TRUE 를 선택하면 full join, all.x 는 left join, all.y는 right join이 된다.

all= FALSE 인 경우에는 당연히 inner join

dplyr package에는 개별적인 join 함수가 있는데 그것을 사용해도 됨

inner_join(df1, df2), left_join(df1, df2), right_join(df1, df2), full_join(df1, df2)

left_join(df2, df1) : alternative right join

2.1.5 Reshape data

2.2 Pipeline operator

library magrittr 를 사용하면 pipeline 연산자를 쓸 수 있게 된다. %>% 형식이다.

만약 c("A","B","C") 라는 데이터를 "ABC" 로 paste 한 다음에 다시 tolower 함수를 적용하여 "abc"로 변환하는 작업을 한다고 하자. 그런 경우에는 다음과 같이 코딩을 해야 한다. 하지만 pipeline operator를 사용하면 함수 중첩을 줄이고 코드를 이해하기 쉽게 사용할 수 있다.

```
library(magrittr)
tolower(paste(c("A","B","C"), collapse = ""))
```

```
[1] "abc"
```

```
c("A","B","C") %>% paste(., collapse = "") %>% tolower  ## paste 함수에는 여러 인자가 들어가는데
```

```
[1] "abc"
```

```
## 첫번째 인자로 들어가기 위해서 . 을 사용함
```

3 LaTeX codes in quarto

3.1 Basic LaTeX code

$$f(r) = \int_0^\infty e^{-\frac{x^2+y^2}{2}} dx$$

$$f(x) = e^{\pi i}$$

3.2 Tikz pictures