# Finance Analytics: R 프로그래밍 기초

권태연

한국외대 국제금융학과

# R 시작하기

# 통계프로그램 R 이란?

R is a language and environment for statistical computing and graphics.

- 데이터분석을 위한 소프트웨어
- 통계학자들이 디자인하고 통계학자들을 위한 개발 플랫폼
- 최신의 알고리즘 및 라이브러리 제공
- 데이터와 관련된 입출력, 핸들링, 관리, 분석, 그래픽 등
- 오픈소스, 무료
- 수 천명의 contributors, 2백만이 넘는 사용자
- 빅데이터분석 텍스트마이닝 등에 대한 수요 확산으로 각광

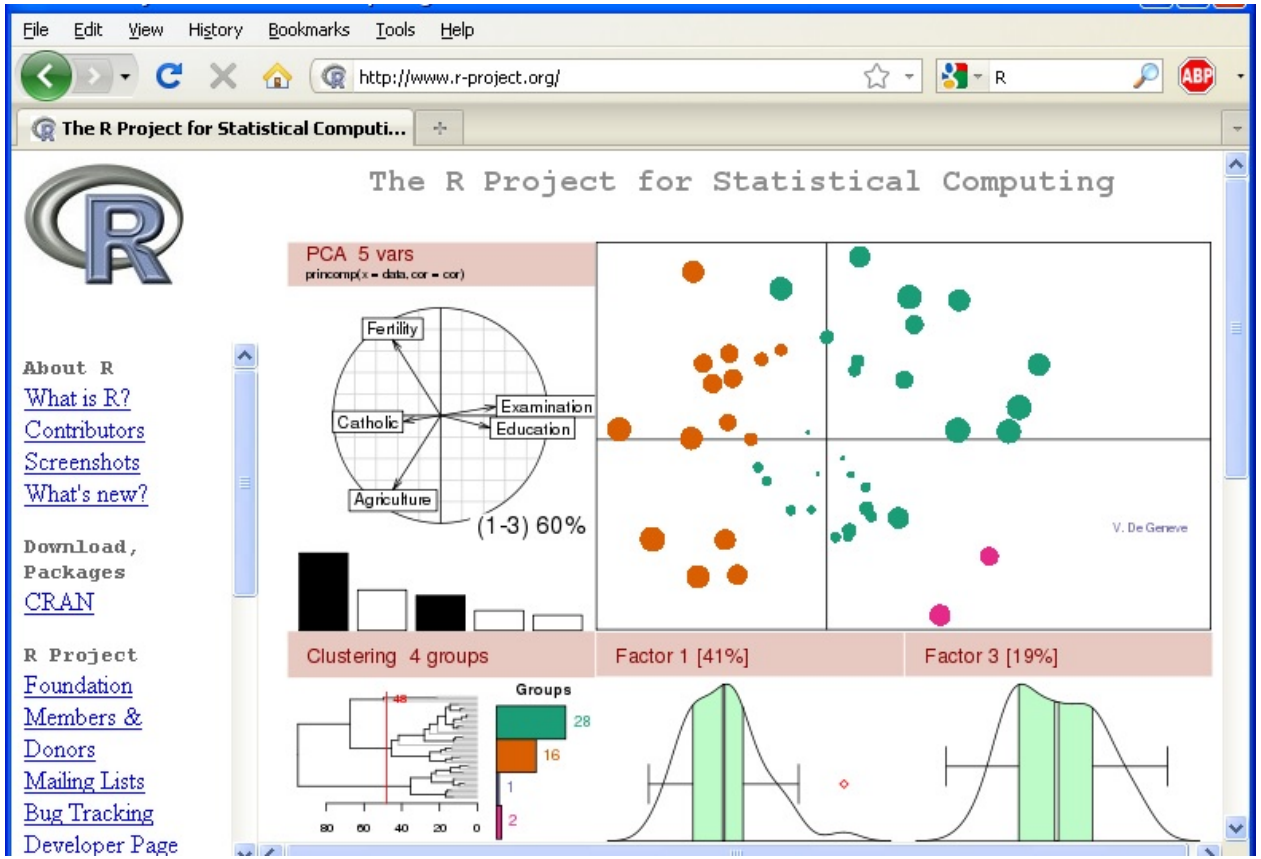
# Big Data and R?

# R 프로그래밍

- 다운로드 및 FAQ: r-project.org
- library 설치, 지정
  package $->$ install package OR >install.packages("NAME of Package")
- R 과 다른 프로그램언어(SPSS, SAS, C, S등)과의 비교
  - R의 장점: 무료, 사용자중심의 데이터 분석 툴 개발, 다른 프로그램(C,C+,S,S-plus)언어와 호환성 높음
  - R의 단점: 프로그램밍 언어의 습득필요, 텍스트 편집기사용해야함, 대용량데이터의 처리의 용이성 떨어짐

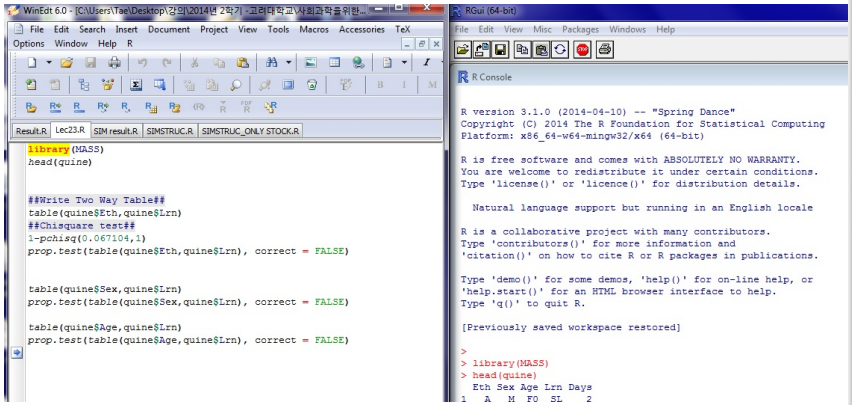
# R설치하기

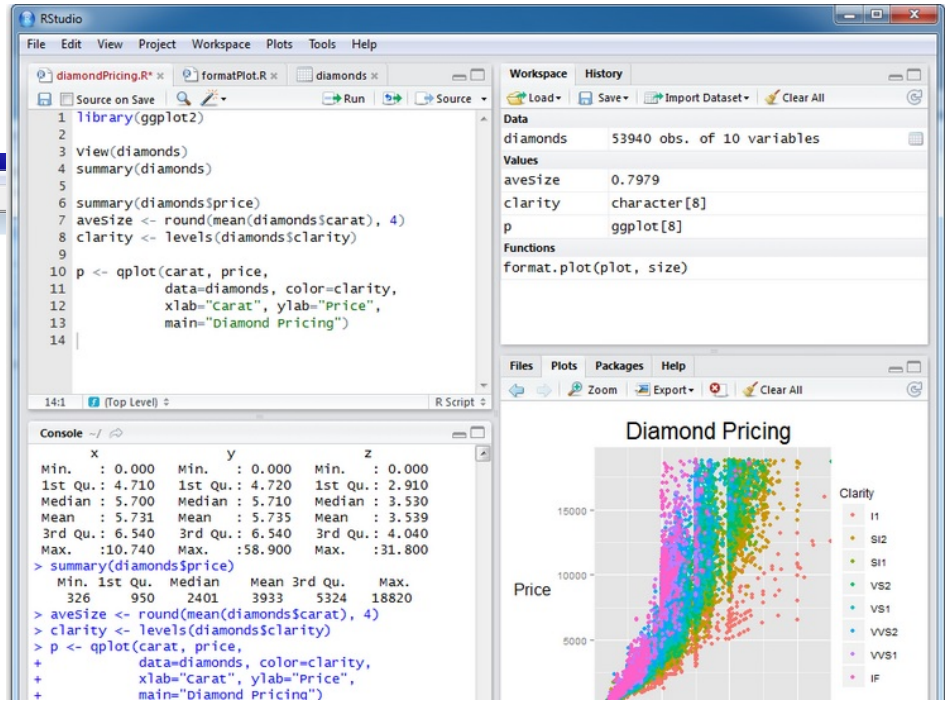
- *http://cran.r-project.org*에서 다운로드
- Windows, Linux, Solaris, Mac등의 다양한 운영체계에서 실행가능



# R설치하기

- 다양한 프로그램 편집기와 함께 사용할 것을 권장함 (X-emacs, R-WinEdt등(메모장도 사용가능))
- 혹은 R-studio설치 및 사용
- 혹은 Anaconda
- 자세한 설치가이드는 별도 자료 참조



# SAS? vs R? vs Python? (주관적인 관점에서)

- SAS : PROC 문을 이용하여 데이터 분석을 위함,

  자료의 처리 (cleaning), 다양한 통계분석 방법에 의한 자료 분석, 결과 요약에 강점
- R : 통계분석 방법에 의한 자료분석과 결과요약 측면에서는 SAS 보다 불편, 예쁘지 않음
  - 모의실험하기 좋음. MC simulation
  - SAS PROC IML 에서 가능
  - 무료...
- python : 프로그래밍 언어
  - 무료
  - 사용자 많음, 통계분석에 국한되지 않음.
  - ML, DL..

https://spectrum.ieee.org/at-work/tech-careers/top-programming-language-2020

# R 시작

- 콘솔 (console) : 사용자의 명령을 입력 받을수 있는 환경
- 프롬프트(prompt) : >
- R Console의 프롬프트(>) 옆에 한 줄씩 명령어(command) 입력한 후 [Enter]키
- 윗 화살표 키의 사용 : 이전입력 명령어

1. 계산기
   > 5+49

   [1] 54 : [1] - 결과의 첫번째 요소

2. 프로그램 주석은 # 이용 : # 뒤의 명령어는 시행하지 않음
   > #Sequence from 1 to 5

   > 1:5

   [1] 1 2 3 4 5

# R 종료

- > q()
- 현재 작업공간 (workspce) 이미지 저장 여부
- *.RData* 확장자로 저장하면, 현재 작업 중인 세션 내에서 수행한 명령어 계산결과 저장
- 프로그램은 따로 저장해야 함

# R을 이용한 계산기능

- 사칙연산 : $+, -, *, /$
- 거듭제곱(power)계산 : ^

  ex: $2^5$ 의 계산

  $>$ 2^5
- 나눗셈의 나머지: %%
- 나눗셈 결과중 정수부분만: %/%

  ex: $31 \div 7$ 의 계산

  $>$ 31/7

  [1] 4.428571

  $>$ 31%%7

  [1] 3

  $>$ 31%/%7

  [1] 4

  $>$ 7*4+3

  [1] 31

# R 객체 (벡터, 행렬, 데이터셋) 설정

# 이름을 이용한 저장공간

- 작업공간에 계산된 결과 및 다양한 객체 (데이터 셋, 행렬)들을 저장한다.
- 할당기호 : $<-$ or $=$
- 계산된 결과를 반복적으로 사용하고자 할 때,

  ex: 복리이자의 계산 : 현재 잔고 $3000, 연(복리)이율이 0.25% 일때 30년뒤의 잔액은?

```
> #복리이지자의 계산
> interest.30<-1.0025^30
> initial.balance<-3000
> final.balance <- initial.balance*interest.30
> final.balance
[1] 3233.35
```

# 함수의 이용

- R에서의 대부분 작업은 함수(function)으로 이루어짐
- q() : 종료함수
- objects() : R 작업공간 내에 저장되어 있는 개체를 화면에 출력함
- 산술평균 : mean() , 합계 : sum(), 분산 : var(), 표준편차 : sd(), 요약 통계량 : summary()
- R은 대/소문자 구분함 : 함수 이름 및 객체 이름의 대소문자 구분 필요
- 연습문제 : 다음의 객체 x,
  $x <- 1:11$
  의 표준편차를 sum() 혹은 mean()함수를 이용하여 구해보고 sd()함수를 이용하여 구하여 비교해 보자.

# 벡터의 생성

- 벡터의 생성 : c( ) 이용

  $> c(0, 7, 8)$

- 벡터의 연결 : c( ) 이용

  $>$ x $<-$ c(0,7,8)

  $>$ y $<-$ 5:10

  $>$ z $<-$ c(x,y)

  $>$ x

  [1] 0 7 8

  $>$ y

  [1] 5 6 7 8 9 10

  $>$ z

  [1] 0 7 8 5 6 7 8 9 10

# 벡터의 구성 요소 부분 선택하기

- length() : 벡터의 길이 출력
- 대괄호 [ ] 를 이용 : 벡터이름[ ]

```
> z
[1] 0 7 8 5 6 7 8 9 10
> length(z)
[1] 9
> z[5]
[1] 6
> z[1:5]
[1] 0 7 8 5 6
> z[c(1,3,5,7,9)]
[1] 0 8 6 8 10
> z[-c(1,3,5,7,9)]
[1] 7 5 7 9
> z[c(2,4,6,8)]
[1] 7 5 7 9
```

# 벡터 연산

- 앞의 연산자 모두 사용가능
- 벡터와 상수간 연산 : 벡터의 각 구성요소(element)와 상수간 연산

```
> x< −c(0,7,8)
> x*3
[1] 0 21 24
> z< − x^2
> z
[1] 0 49 64
```

- 벡터와 벡터간 연산 : 같은 위치에 놓인 구성요소(element)끼리 연산
  두 벡터의 크기가 같아야 함, 같지 않다면 크기가 작은 벡터를 반복 사용하여 큰 벡터와 맞추어 연산한다.

```
> x< −c(0,7,8)
> y< −c(1,2,3)
> x+y
[1] 1 9 11
```

# 행렬

- 벡터를 행렬의 형식으로 재배열 : matrix()
- 행렬 구성요소
- sub-matrix

```
> m< −matrix(1:6, nrow=2,ncol=3)
> m
   [,1][,2][,3]
[1,] 1   3   5
[2,] 2   4   6
> m[1,2]
[1] 3
> m[1,]
[1] 1 3 5
> m[,1]
[1] 1 2
> m[1 : 2,1 : 2]
   [,1][,2]
[1,] 1   3
[2,] 2   4
```

# 데이터 프레임 (data frame)

- 행렬의 각 열이 변수로 각각의 이름을 갖는다.
- dataset명$변수명 으로 하나의 변수 선택 할 수 있다. 이때, 이는 벡터로 인식된다.
- 행렬을 데이터 프레임 형태로 바꾸어 저장할 수 있다 : data.frame

```
> color <- c("red","yellow","blue")
> numbers <- c(1,2,3)
> dat<-data.frame(colors, numbers, more=c(4,5,6))
> dat
  colors numbers more
1    red       1    4
2 yellow       2    5
3   blue       3    6
> dat$more
[1] 4 5 6
```

# R 내장 함수의 이용

# 내장함수 및 도움말

- 내장함수의 온라인 도움말 *?*함수명 혹은 *help(*함수명*)*
- *example(*함수명*)*
- https://www.r-project.org/search.html

# 기본내장함수

- mean() : 평균
- median() : 중위수
- var() : 분산
- summary() : 요약통계량
- sd() : 표준편차

```
x <- c(0:10, 50)
xm <- mean(x)
c(xm, mean(x, trim = 0.10))
[1] 8.75 5.50
```

# 연습문제

다음의 자료는 온실로 들어오는 태양 복사열에 대한 관측표본이다.
11.1 10.6 6.3 8.8 10.7 11.2 8.9 12.2

- 이 자료를 solar.radiation이라는 이름을 가진 객체에 할당하자
- 평균 중앙값 분산을 구해보자
- solar.radiation이 가지는 각 자료의 값에 10을 더한 결과를 sr10에 저장하고 sr10의 평균 중앙값 분산을 구해본다. 어떤 통계량이 변화했으며, 얼마나 많이 변화했는지 살펴보자.
- 분산을 구하기 위해 사용되는 var() 함수는 다음의 두가지 중 어떠한 공식을 사용하는지 알아보아라 $(1/n)\sum_{i=1}^{n}(x_i - \bar{x})^2$, $(1/(n-1))\sum_{i=1}^{n}(x_i - \bar{x})^2$

# 관계연산자

- <, >, == (equal), >=, <=, ! = (not equal) , ! (not)
- 결과가 TRUE 혹은 FALSE로 나타남

```
> a< −c(3,6,9)
> a>4
[1] FALSE TRUE TRUE
> !a > 4
[1] TRUE FALSE FALSE
```

- 관계연산자 결과가 TRUE인 element만 추출하여 새로운 벡터생성

```
> a[a > 4]
[1] 6 9
> a[a!=3]
[1] 6 9
```

- 두 개 이상의 관계연산자 이용: & (and), | (or)

```
> a > 4 & a < 9
[1] FALSE TRUE FALSE
> a[a>4 | a==3]
[1] 3 6 9
```

# 객체 입출력

# 데이터 입출력 - 1. 디렉토리 지정

- 데이터 혹은 행렬 저장 시, 디렉토리를 사전에 지정
- setwd("c:/mydata2015")
- setwd("c:\\mydata2015")

# 데이터 입출력 - 2. 외부 데이터 불러오기

- 텍스트문서 Data.R $<-$ read.table(file=" 경로 ")
  *Data.R*$<-$ *read.table(file="C: \\mydata2015\\new.txt")*
  *Data.R*$<-$ *read.table(file="new.txt")* 사전에 디렉토리가 지정되어 있는 경우
- 그외 엑셀, csv, spss파일 사용가능 read.csv(), read.xls(), read.spss()
- 경로, 데이터명 및 변수명은 영문숫자혼합, 띄어쓰기 불가, 대소문자구문
- Dataset확인 : names(데이터명) head(데이터명) dim(데이터명)

# 데이터 입출력 - 3. 데이터 (행렬) 내보내기

- 텍스트문서 write.table(Data.R, file=" 경로 ")
  *write.table(Data.R, file="C: \\mydata2015\\new.txt")*
  *write.table(Data.R, file="new.txt")* 사전에 디렉토리가 지정되어 있는 경우
- 그외 엑셀, csv, spss파일 사용가능 write.csv(), write.xls(), write.spss()

# 데이터 입출력 - 4. 작업내용 모두 저장

- 현재까지의 모든 작업내용(함수, 객체등) 을 저장할 수 있다.
- .RData 확장자를 사용하여 저장한다.

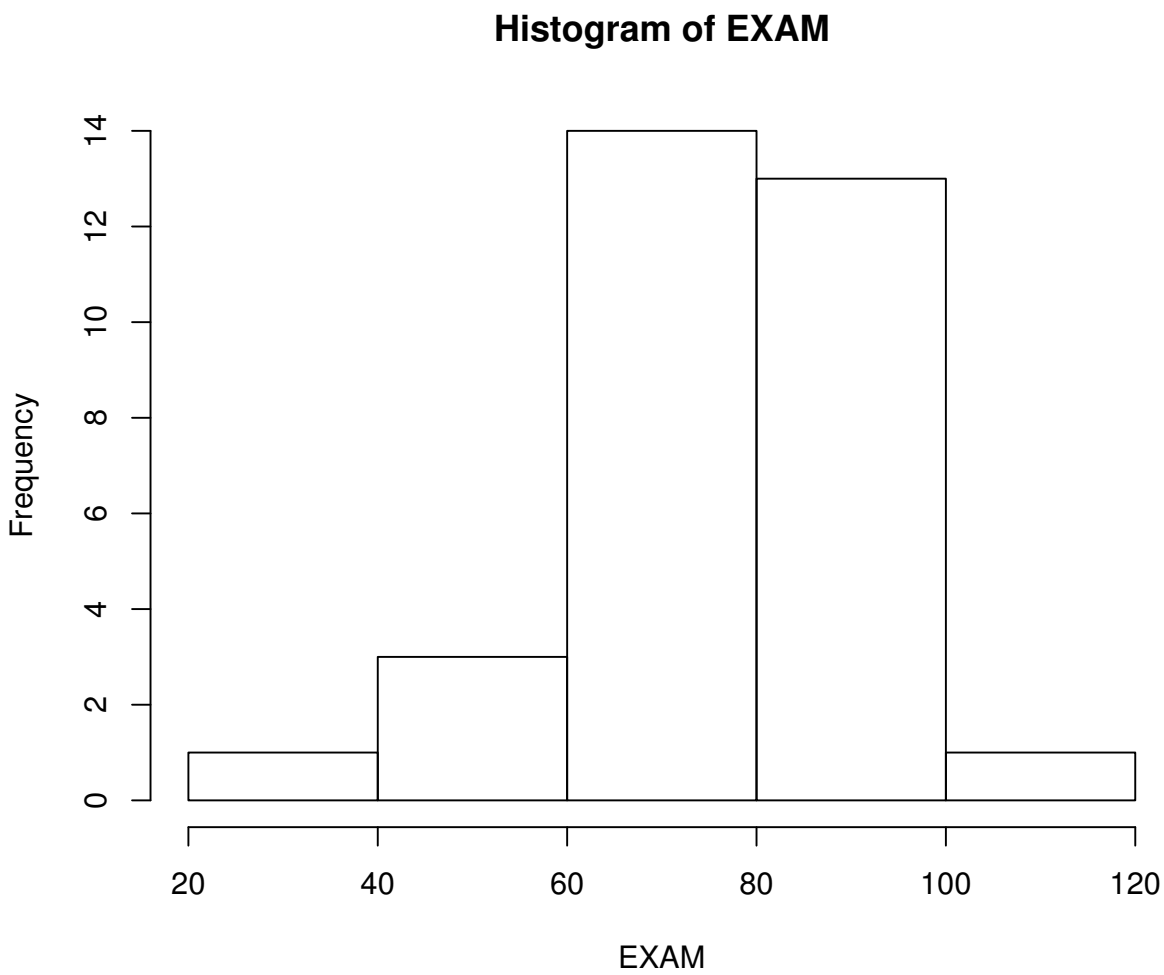
# 연습문제

test2.csv파일을 R데이터셋으로 만들어보자
exam과 exam2 점수의 평균인 exam3변수를 만들어보자

# 그래프 & 확률분포함수

# 1. 히스토그램

>hist(EXAM)



# Density, distribution function, quantile function and random generation

- 알려진 특별한 분포들에 대한 p.d.f., c.d.f., quantile function 및 그들로 부터의 random sampling을 하는 함수들
- For Normal distribution: $X \sim Normal(\mu, \sigma)$ 일때,
  - dnorm(x, mean=$\mu$, sd=$\sigma$) : $X = x$에서의 normal distribution p.d.f. 값 산출
  - pnorm(q, mean=$\mu$, sd=$\sigma$) : $X = q$에서의 c.d.f. 값 산출, $Pr(X \leq q)$ 값 산출
  - qnorm(p, mean=$\mu$, sd=$\sigma$) : $Pr(X \leq x) = p$를 만족하는 $x$값 산출, quantile function
  - rnorm(n, mean=$\mu$, sd=$\sigma$) : $Normal(\mu, \sigma)$ 에서 size=n의 random sample 생성
- Poisson distribution : dpois(), ppois(), qpois(), rpois()
- Binomial distribution : dbinom(), pbinom(), qbionm(), rbinom()

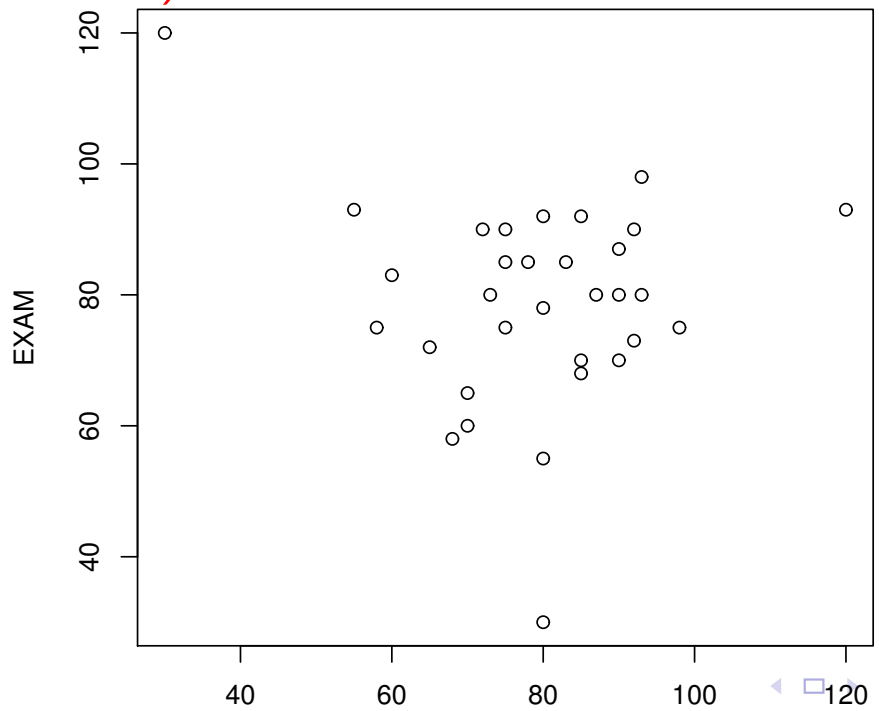
# 연습

- 표준정규분포, t분포(자유도=2,3,4), Gamma distribution (shape=2,scale=2)에서 각각 size=1000인 표본을 무작위 추출하여 생성하고, 각각의 표본의 분포를 histogram을 그려 요약하시오.
- 앞선 Lecture 노트의 Normal distribution관련문제 해결

# 2. 산점도

- 산점도 (Scatter plot) : 서로 다른 두개의 변수(두개 모두 연속형 변수)간의 관계를 살펴보기 위한 그래프
- 두개의 변수를 하나는 가로축 (x축), 다른 하나는 세로축 (y축)에 mapping
- *plot()* 함수를 사용
  1. plot(x축 변수 벡터, y축 변수 벡터)
  2. plot(y축 변수 벡터 ~ x축 변수 벡터)

```
>plot(EXAM~EXAM2)
```



# 2. 산점도

산점도 option

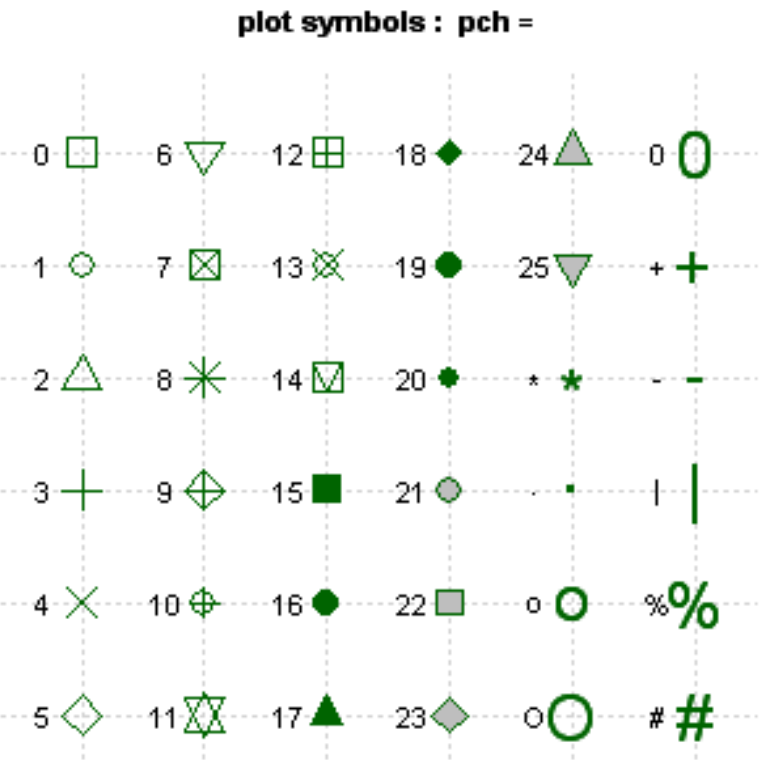
- type=" " : 산점도의 타입 결정, 선으로 연결할 수 있다.
- col= " " : 산점도의 색 결정
- pch= : 산점도를 point로 그렸을 때, point의 모양
- lty= : 산점도를 선으로 연결하였을 때, 선의 모양
- ylim=c(y0,y1), xlim=c(x0,x1) : x축, y축 조정

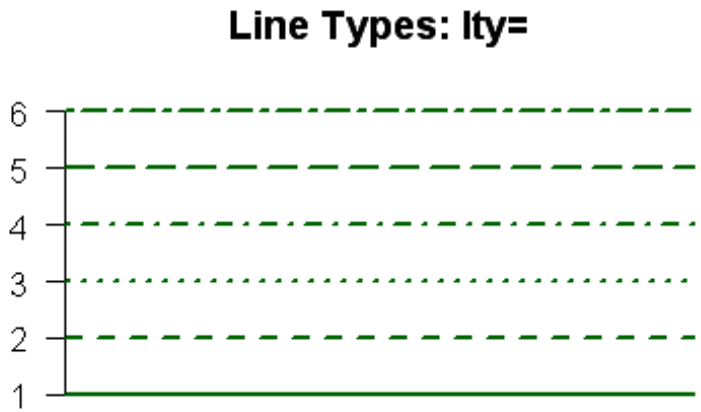
```
> plot(EXAM,EXAM2,col="red", pch=3)
> plot(EXAM,EXAM2,col="red", type="n")
>plot(sort(EXAM),sort(EXAM2),type="l",lty=2)
```

# 2. 산점도

| type | description |
|---|---|
| p | points |
| l | lines |
| o | overplotted points and lines |
| b, c | points (empty if "c") joined by lines |
| s, S | stair steps |
| h | histogram-like vertical lines |
| n | does not produce any points or lines |

# 2. 산점도


plot symbols : pch =
0
6
12
18
24
0
1
7
13
19
25
+
2
8
14
20
*
-
3
9
15
21
|
4
10
16
22
o
%
5
11
17
23
#


Line Types: lty=
6
5
4
3
2
1

# 연습문제

1. 다음의 데이터 셋을 만들고

```
> indexfinger
  sex length width
1   M    7.9   2.3
2   F    6.5   1.7
3   M    8.4   2.6
4   F    5.5   1.7
5   F    6.5   1.9
6   M    8.0   2.1
7   F    7.0   1.8
8   M    7.5   1.9
>
```

2. width =y, length=x로 한 산점도를 그려보자
3. 성별로 따로따로 그려보자. (하나의 플롯안에 겹쳐서)
4. 성별로 따로따로 그려보자. (두개의 플롯으로 나누어서)

# 선형회귀모형 적합

# 선형회귀모형 적합

- $lm(y \sim x) : y = a + bx + e$의 단순 선형 회귀모형 적합
- $lm(y \sim x1+x2) : y = a + bx_1 + cx_2 + e$의 다중 선형 회귀모형적합
- summary()를 이용하여 모형추정치 요약

```
> model <- lm(EXAM ~ EXAM2)
> model
> summary(model)
```

# 플롯에 추가하기

이미 생성된 그래프의 플롯 영영에 다양한 요소들을 추가할 수 있다.

- points(x,y) : 점추가
- lines(x,y) : 선추가
- text(x,y,labels) : 텍스트추가
- abline(a,b) 혹은 abline(lm(y ~ x)) : y=a+bx추가
- abline(h=y) : 수평선 추가
- abline(v=x) : 수직선 추가
- polygon(x,y) : 다각형 추가
- segments(x0,y0,x1,y1) : 선분 (line segment) 추가
- arrows(x0,y0,x1,y1) : 화살표 추가
- symbols(x,y) : 원(circle), 사각형(square) 추가
- legend(x,y,legend) : 범례(legend) 추가

** par(mfrow=c(m,n)) :여러개의 그래프 그래프행렬로 나타낼 수 있다.

# 연습문제

1. 다음의 데이터 셋을 만들고

```
> indexfinger
  sex length width
1   M    7.9   2.3
2   F    6.5   1.7
3   M    8.4   2.6
4   F    5.5   1.7
5   F    6.5   1.9
6   M    8.0   2.1
7   F    7.0   1.8
8   M    7.5   1.9
>
```

2. width =y, length=x로 한 단순선형회귀모형을 피팅하고
3. 산점도와 산점도 위에 regression line을 그려보자.

# FINANCE_DATA.csv

- 출처: Yahoo Fiance, Kosis(Kospi200)
- Period: 2009.04  2014.04 (5년), 월별자료
- 변수 : Kospi200지수, S&P500지수, Nikkei225지수, 삼성전자, 신한금융그룹의 월말가격(Closing Price)