Lecture 1: Basic R

Zhentao Shi

Feb 11, 2017

Basic

- help system: `?`, '??'. eg. `?seq`, `??sequence`
- `c()`
- arithmetic
- logical vector `& | !=`, `any`, `all`
- factor
- character
- missing values `NA`, `NaN`
- index vector for selection `a[ ]`. Either positive integer or logical vector.

example

```
# logical vectors
logi_1 = c(T,T,F)
logi_2 = c(F,T,T)

logi_12 = logi_1 & logi_2
print(logi_12)
```

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

Arrays

An array is a table of numbers. A matrix is a 2-dimensional array.

- array arithmetic: element-by-element.
- `%*%`, `solve`, `eigen`

example: simple regression—OLS estimation with one x regressor and a constant.

- literally translate $\hat{\beta} = (X'X)^{-1}X'y$ into code.
- t-statistic: $\left(\hat{\beta}_2 - \beta_{02}\right) / \hat{\sigma}_{\hat{\beta}_2}$.

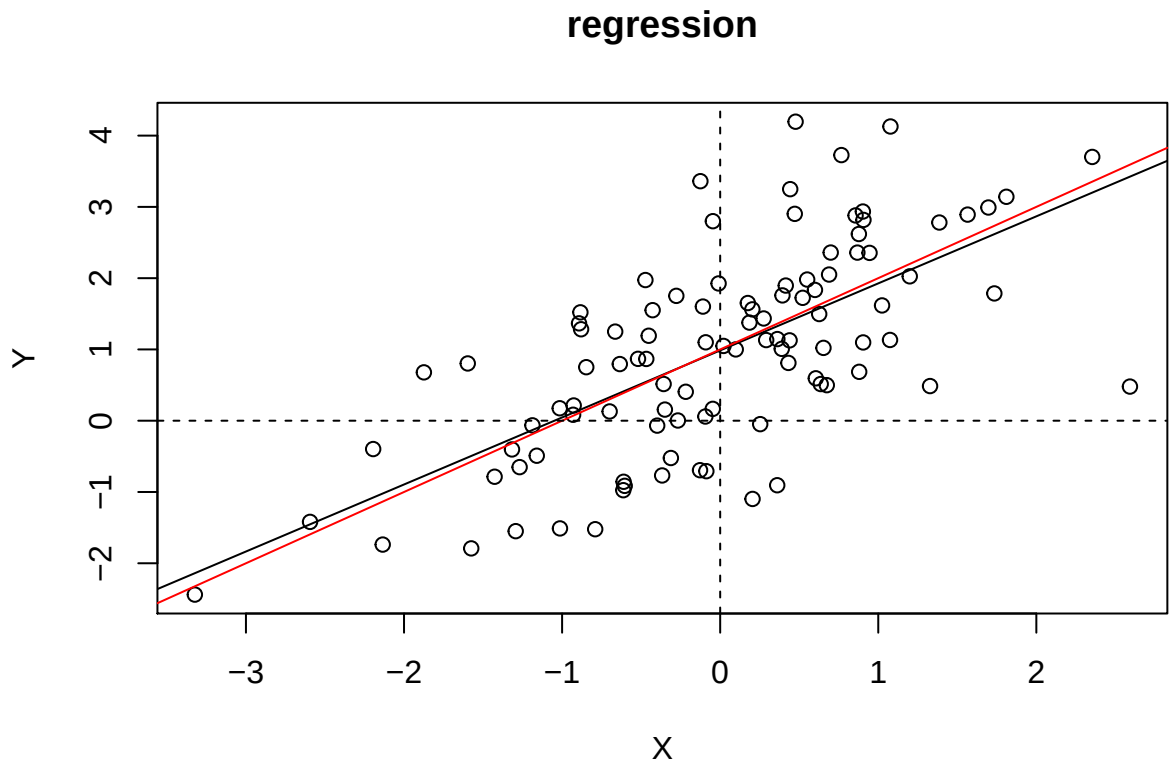
```
# check the OLS fitting
rm(list = ls( ) )
set.seed(111) # can be removed to allow the result to change

# set the parameters
n = 100
b0 = matrix(1, nrow = 2 )

# generate the data
e = rnorm(n)
X = cbind( 1, rnorm(n) )
Y = X %*% b0 + e

# OLS estimation
bhat = solve( t(X) %*% X, t(X)%*% Y )
```

```r
# plot
plot( y = Y, x = X[,2], xlab = "X", ylab = "Y", main = "regression")
abline( a= bhat[1], b = bhat[2])
abline( a = b0[1], b = b0[2], col = "red")
abline( h = 0, lty = 2)
abline( v = 0, lty = 2)
```




```r
# calculate the t-value
bhat2 = bhat[2] # parameter we want to test
e_hat = Y - X %*% bhat
sigma_hat_square = sum(e_hat^2)/ (n-2)
sig_B = solve( t(X) %*% X  ) * sigma_hat_square
t_value_2 = ( bhat2 - b0[2]) / sqrt( sig_B[2,2] )
print(t_value_2)
```

```
## [1] -0.5615293
```

Packages

An initial installation of R is small, but R has an extensive system of add-on packages. A package can be installed and invoked by

```r
install.packages("package_name")
library(package_name)
```

eg: The following packages are useful for parallel computing.

```
library(AER)
```

```
## Loading required package: car
```

```
## Loading required package: lmtest
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: sandwich
```

```
## Loading required package: survival
```

Mixed data types

- list
- data.frame

```
data("CreditCard")
head(CreditCard)
```

```
##   card reports      age income       share expenditure owner selfemp
## 1  yes       0 37.66667 4.5200 0.033269910  124.983300   yes      no
## 2  yes       0 33.25000 2.4200 0.005216942    9.854167    no      no
## 3  yes       0 33.66667 4.5000 0.004155556   15.000000   yes      no
## 4  yes       0 30.50000 2.5400 0.065213780  137.869200    no      no
## 5  yes       0 32.16667 9.7867 0.067050590  546.503300   yes      no
## 6  yes       0 23.25000 2.5000 0.044438400   91.996670    no      no
##   dependents months majorcards active
## 1          3     54          1     12
## 2          3     34          1     13
## 3          4     58          1      5
## 4          0     25          1      7
## 5          2     64          1      5
## 6          0     54          1      1
```

```
help(CreditCard)
```

```
## starting httpd help server ...
```

```
##  done
```

Input and output

- read.table()
- write.table()

example

```
HEX = read.csv("http://ichart.finance.yahoo.com/table.csv?s=0388.HK")
print(head(HEX))
write.csv(HEX, file = "HEX.csv")
```

Statitics

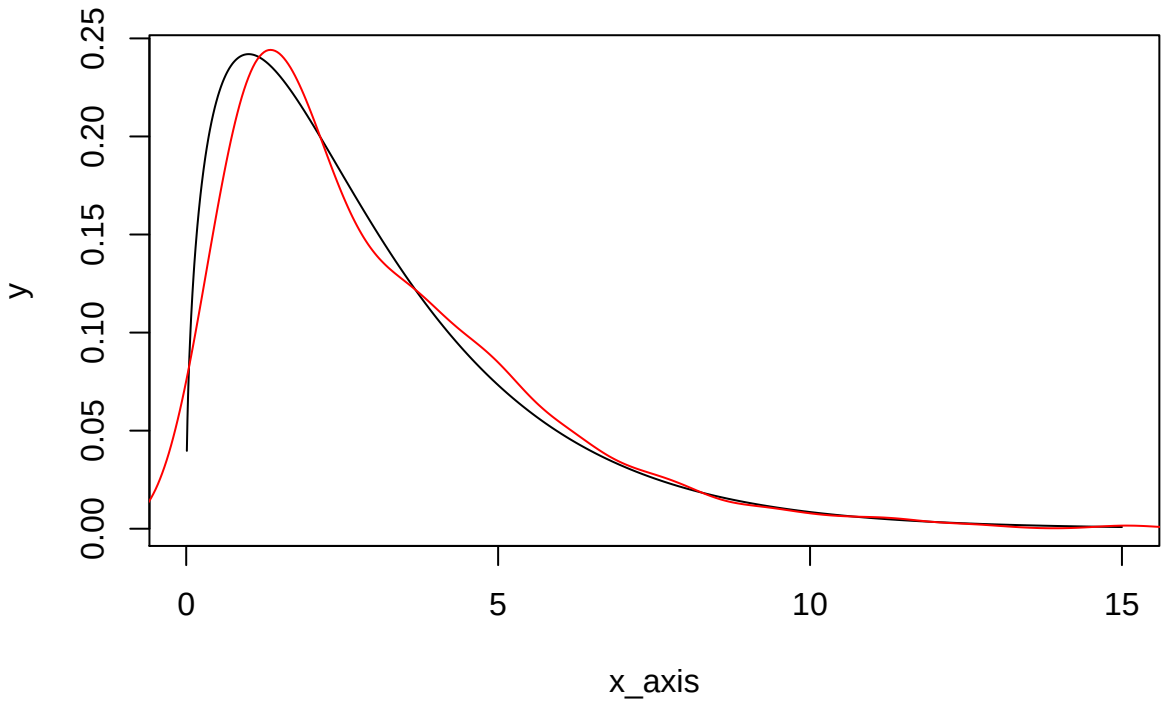
- `p`, `d`, `q`, `r`

example:

1. plot the density of $\chi^2(3)$ over `x_axis = seq(0.01, 15, by = 0.01)`
2. generate 1000 observations for the above distribution. plot the kernel density.
3. calculate the 95-th quantile and the empirical probability of observing a value greater than the 95-th quantile.

```
set.seed(888)
x_axis = seq(0.01, 15, by = 0.01)

y = dchisq(x_axis, df = 3)
plot(y = y, x=x_axis, type = "l")
z = rchisq(1000, df = 3)
lines( density(z), col = "red")
```




```
crit = qchisq(.95, df = 3)

mean( z > crit )
```

```
## [1] 0.047
```

User-defined function

- `fun_name = function(args) {expressions}`

- variables defined in a functions are local.

example: given a sample, calculate the 95% two-sided asymptotic confidence interval according to a central limit theorem.

```
# construct confidence interval

CI = function(x){
  # x is a vector of random variables

  n = length(x)
  mu = mean(x)
  sig = sd(x)
  upper = mu + 1.96/sqrt(n) * sig
  lower = mu - 1.96/sqrt(n) * sig
  return( list( lower = lower, upper = upper) )
}
```

Flow control

- if
- for, while

example: calculate the empirical coverage probability of a poisson distribution of degree of freedom 2.

```
Rep = 1000
sample_size = 100
capture = rep(0, Rep)


pts0 = Sys.time() # check time
for (i in 1:Rep){
  mu = 2
  x = rpois(sample_size, mu)
  bounds = CI(x)
  capture[i] = ( ( bounds$lower <= mu  ) & (mu <= bounds$upper) )
}
mean(capture) # empirical size
```

```
## [1] 0.938
```

```
pts1 = Sys.time() - pts0 # check time elapse
print(pts1)
```

```
## Time difference of 0.06104398 secs
```

Statistical models

- lm(y~x, data =)