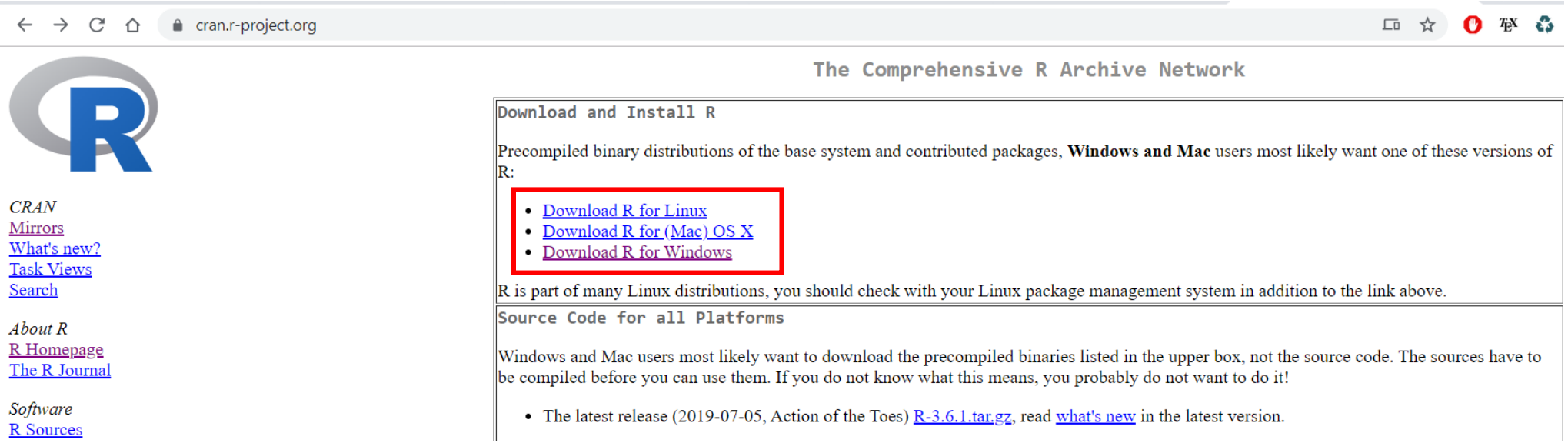
Introduction to R

Download and Installation

- https://cran.r-project.org/



IDE

- Use the free IDE – **Rstudio** for homeworks.

https://rstudio.com/products/rstudio/download/#download

[Create, Edit, knit, .. R markdown (.rmd) files using **Rstudio**]

Basic Data Types

- Numeric (Float/Double): 10.5, 5e-06, ..
- Integer: 3, -3, 7L , …
- Complex: 2+3i, complex(real=2,imaginary=3), …
- Logical: TRUE, FALSE, T, F
- Character: “a”, “abc”, “2.718”, …

Operators

- Assignment: = , <-
  Example: x = 5, x <- 5, z <- c(1,2,3)
- Boolean: &, |, !
- Relation: <, >, <=, >=, ==, !=
- Arithmetic: +, - , *, /, ^, ** (power), %% (modulo), %*% (matrix multiplication)

Vectors

- A **vector** is a sequence of elements of the same basic data type

- Construction: using c(), :, rep, seq(), output of a function, ….

1. x <- c(1,2,3,5.5), x <- c(“a”, “b”, “abc”), x <- c(T, F, F, T)
2. y <- 10:15 # unit spacing
3. y <- rep(2,6); y <- rep(“abc”,10)
4. z <- seq(0,1,by=0.2) # spacing by 0.2
5. v <- sample(1:5, size=10, replace=TRUE)

Vectors

- Constructing named vector:
  1. v <- c(x=2, y=3, z=10.2, …)

- Assign names to vector v
  1. names(v) <- c(‘x’, ‘y’, ‘z’, ‘a’, ‘b’, …)
  2. names(v)[4] <- “tom”

- Indexing of vector v <u>(index starts with 1 not 0)</u>
  1. v[1] , v[2], … # gives the corresponding element for the vector v
  2. v[“x”], v[“tom”] , … # for named vectors

Matrix

- **Matrix** of same basic data type

- Construction using matrix(), rbind(), cbind(), …

1. X <- matrix(c(1, 2, 3, 4, 5, 7.7), nrow=2, ncol=3) # by default, fill matrix by column
2. X <- matrix(c(1, 2, 3, 4, 5, 7.7), nrow=2, ncol=3, byrow=TRUE) # fill matrix by row
3. Y <- rbind(c(1,2,3), c(3,4,5), c(3,4,5)), Y <- rbind(X, c(2.3, 4, 5), X)
4. Z <- cbind(c(1,2,3), c(4,3,4), c(3,4,5))), Z <- cbind(c(0.1,0.2),X, c(8, 5)),

   Z <- cbind(X,X,X,X)

rbind() & cbind() – very useful

- X <- rbind(R1, R2, R3, …) X = $\begin{pmatrix} R1 \\ R2 \\ R3 \\ \vdots \end{pmatrix}$, given dimensions match

- X <- cbind(C1,C2,C3, …) X = $\begin{pmatrix} C1 & C2 & C3 & \dots \end{pmatrix}$, given dimensions match

Matrix

- Assign names for matrix M = $\begin{pmatrix} 1 & 2 & 3 \\ 5 & 3 & 6 \end{pmatrix}$

  1. dimnames(M) <- list(c(“x”, “y”), c(“tom”, “dick”, “harry”))
  2. rownames(M) <- c(“x”, “y”),
  3. colnames(M) <- c(“tom”, “dick”, “harry”)

- Indexing of matrix M

  1. M[2,3] # first index is row number, second index is column number
  2. M[“y”, “harry”]

Some useful Matrix operations

1. M1 + M2 # addition
2. M1 * M2 # element-wise multiplication
3. M1 %*% M2 # matrix multiplication
4. nrow(M), ncol(M) # number of rows and columns respectively
5. t(M) # transpose
6. solve(M) # inverse
7. diag(c(1, 2, 4)) # Construct diagonal matrix
8. eigen(M) # Eigen decomposition
9. det(M) # Computes determinant

List

- A **list** is a sequence of objects of different data types and lengths.
- Construction: using list()
  1. L <- list(5, c(1,2,3), matrix(rep(2,6),2,3), list(5, “abc”))

     # L becomes a sequence of a number, vector, matrix and list
- Naming (similar to vectors)
  1. L <- list(num=5, vec=c(1,2,3), mat=matrix(rep(2,6),2,3), list=list(e1=5, e2=“abc”))
  2. names(L) <- c(“num”, “vec”, “mat”, “list’); names(L$list) <- c(“e1”, “e2”)
- Indexing:
  1. L[1] # returns the a list containing first element
  2. L[[1]] # returns the first element
  3. L[[“num”]] or L$num # returns the named element with name “num”

Data frame

- A **data frame** is a list of vectors (of different data types) of equal length.
- It is used for storing data table. The vectors are the columns of the table

- Construction: using data.frame(), read.table(), read.xls(), read.csv(), …
  1. df <- data.frame(c(“Bob”, “Alice”, “John”), c(45,30,55),c(“Toyota”, “Nissan”, “Tesla”))
  2. df <- data.frame(name = c(“Bob”, “Alice”, “John”), age = c(45,30,55), car = c(“Toyota”, “Nissan”, “Tesla”))

Data frame

- Naming (same as Matrix)

  rownames(df) <- c(“person1”, “person2”, “person3”)

  dimnames(df) <- list(.. , ..)

- Indexing (similar to Matrix and list)

  1. df[2,3]
  2. df[“person1”, “age”]
  3. df$age

Selective Indexing

- For matrix or data frame D:
  1. D[,2] # gives the second column
  2. D[3,] # gives the third row
  3. D[c(r1, r2, ..), c(c1, c2, ..)] # gives the sub-matrix/sub-dataframe of selected rows and columns
  4. D[c(T, F, T, ..), c(F, F, T, ..)] # gives the sub-matrix /sub-dataframe for selected rows and columns corresponding to the “T” values
  5. D[D[,2]<40 ,] # gives the sub-matrix /sub-dataframe with those rows whose 1st column is less than 40
  6. D$age[D$age<40] <- 30 # setting with selective indexing

if, else, ifelse & for

- if (condition) {
  statement
  } else if (condition2) {
  statement2
  } else {
  statement3
  }

- v is a vector
- for (i in v) {
  statement _i
  }

- ifelse (c(T,F,T,T), c(5,5,5,5), c(2,2,2,2)) # result: 5 2 5 5, kind of selective indexing
- ifelse (vec < 4, 0 , vec) # replace the elements less than 4 with 0
- ifelse (M == 0, NA, M) # returns a copy of matrix M with the 0's replaced with NA

Functions (user-defined)

- function_name <- function(arguments){

```
        function_definition
        return (output)          # or just write: output, last computed variable is returned
}
```

- Example: “oddcount” takes a vector and counts the number of odd elements

```
oddcount <- function (x) {
        k <- 0                          # assign 0 to k
        for (n in x) {
            if (n %% 2 == 1) k <- k+1   # %% is the modulo operator
        }
        return (k)
}
```

apply

- Apply a given function to elements/vectors of a matrix
  - apply(M, 1, sum) # apply sum on rows
  - apply(M, 2, sum) # apply sum on columns
  - apply(M, c(1,2), sum) # apply sum on each element, no effect of sum

lapply, sapply

- Applies function on each element of the list. lapply returns result in list format, sapply returns in vector format.
  - lapply(df, mean) # apply “mean” on each column of data frame “df”
  - sapply(df, sum) # apply “sum” on each column of data frame “df”

Factors, split & tapply

- Factors are like categories: To know the distinct factors use “levels”
  1. levels(df$name) # give the distinct names in the data frame “df”

- split() is used to divide a vector of data into categories given by a vector of factors.
  1. split(dfage, dfgender) # split the age data of “df" into groups of gender

- tapply() is used to combine split() and lapply(). It splits the data into categories and apply a function on the data for each category
  1. tapply(dfage, dfgender, mean) # gives the mean age for each gender

Random sampling

- sample(c(1,2,4,7) , 10, replace=TRUE)

  # draw 10 random samples from set {1,2,4,7} with replacement

- sample(c(1,2,4,7) , 3, replace=FALSE)

  # draw 3 random samples from set {1,2,4,7} without replacement

Probability density, distribution and quantiles

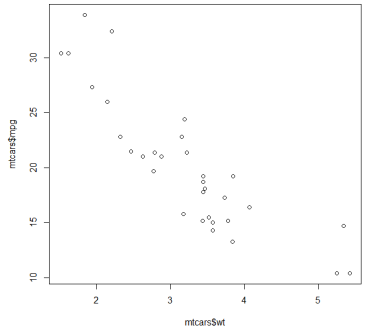
Density	CDF	Quantile (CDF^{-1})
dunif(x, …)	punif(x, …)	qunif(p, …)
dnorm(x , …)	pnorm(x , …)	qnorm(p , …)
dbinom(x , …)	pbinom(x , …)	qbinom(p , …)
dpois(x , …)	ppois(x , …)	qpois(p , …)
dexp(x, …)	pexp(x, …)	qexp(p, …)
dchisq(x, …)	pchisq(x, …)	qchisq(p, …)

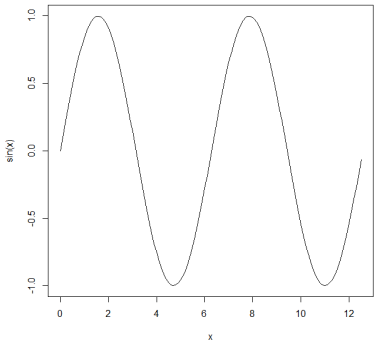
Some useful functions

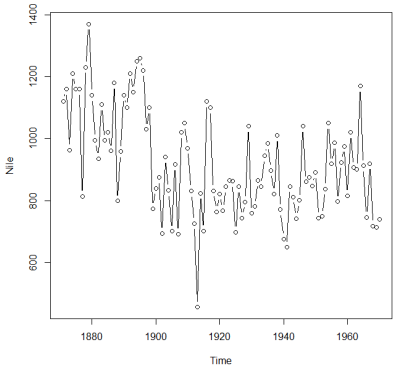
- help(“rbind”, help_type=“text”) # opens docs for “rbind” in R-console
- example(“rbind”) # opens examples of “rbind” in console
- head(mtcars)

- sqrt(), exp(), log(),
- length(), sum(), min(), max(), mean()
- combn(v,m) # gives all combinations of m elements from vector v
- choose(n,k)
- integrate(function(t) t^2+t, 0, 1)$value

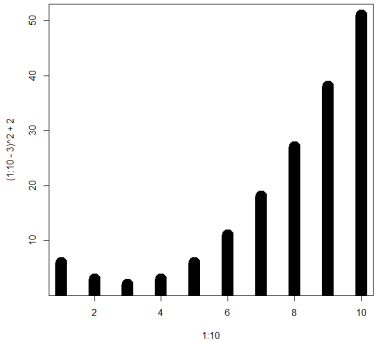
plot





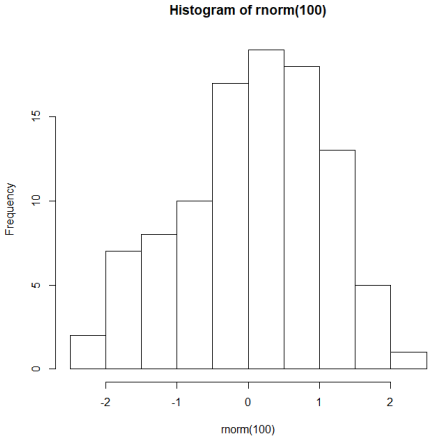


- plot(x, y, “p”) # plot with points
- plot(x, y, “l”) # plot with lines
- plot(x, y, “b”) # plot with both points and lines
- plot(x, y, “h”, lwd=10) # bar plot



hist

- hist(x)



Reference:

1. https://github.com/matloff/fasteR
2. http://www.r-tutor.com/r-introduction
3. Appendix of textbook:

   http://heather.cs.ucdavis.edu/~matloff/132/PLN/probstatbook/ProbStatBookF19.pdf

Learn & practice:

1. help(), example(), Youtube videos and Google search
2. Learn R in R console with swirl(): https://swirlstats.com/students.html
3. https://www.datacamp.com/courses/free-introduction-to-r