R Tutorial for SAMSI Undergraduate Modoeling Workshop

Xinyi Li and Wenjia Wang, SAMSI

05/30/2019

About R

R is a free software environment for statistical computing and graphics:

- a different implementation of S developed at Bell Lab;
- provides a wide variety of statistical and graphical techniques, and is highly extensible;
- open source;
- powerful IDE (integrated development environment), such as Rstudio.

Install R

1. Download the most recent version of R. The R FAQs and the R Installation and Adminstration Manual contain detailed instructions for installing R on various platforms (Linux, OS X, and Windows being the main ones).
2. Start the R program; on Windows and OS X, this will usually mean double-clicking on the R application, on UNIX-like systems, type "R" at a shell prompt.
3. As a first step with R, start the R help browser by tying `help.start()` in the R command window. For help on any function, e.g. the "mean" function, type `?mean`.

Install RStudio

1. Go to RStudio and click on the "Download RStudio" button.
2. Click on "Download RStudio Desktop."
3. Click on the version recommended for your system, or the latest Windows version, and save the executable file. Run the .exe file and follow the installation instructions.

Data types

R has a wide variety of data types including

- scalars,
- vectors (numerical, character, logical),
- matrices,
- data frames,
- and lists.

We can use variables without definition in advance.

Use the assignment operator <- or = to create new variables.

```
x <- 1
print(x)
```

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

```
x = 2
print(x)
```

```
## [1] 2
```

Scalar

```
num = 3
print(num)
```

```
## [1] 3
```

```
print(typeof(num))
```

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

```
num = 3.14
num.int = as.integer(num)
print(num.int)
```

```
## [1] 3
```

```
print(typeof(num.int))
```

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

Vector

```
x = 1:3
print(x)
```

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

```
y = c(4, 5, 6, 7)
y[1]  # subsetting
```

```
## [1] 4
```

```
y[-1] # subsetting
```

```
## [1] 5 6 7
```

```
y[c(1,4)]  # subsetting
```

```
## [1] 4 7
```

```
y[-c(1,4)] # subsetting
```

```
## [1] 5 6
```

```
z = c(y[c(1,4)], y[-c(1,4)])
print(z)
```

```
## [1] 4 7 5 6
```

Matrix

```r
a = seq(1, 9, length.out=9)
A = matrix(a, nrow=3, ncol=3)
print(A)
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```r
print(typeof(A))
```

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

```r
print(class(A))
```

```
## [1] "matrix"
```

```r
A[1:2, 1]  # subsetting the first two elements in the first column
```

```
## [1] 1 2
```

```r
A[1:2, c(1,2)]
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
```

```r
A[1:2, ]  # subsetting the first two rows
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
```

Array

```r
b = seq(1, 8, by=1)
B = array(data=b, dim=c(2,2,2))
print(B)
```

```
## , , 1
##
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
##
## , , 2
##
##      [,1] [,2]
## [1,]    5    7
## [2,]    6    8
```

```r
class(B)
```

```
## [1] "array"
```

```
B1 = B[ , , 1]

B2 = B[ , , 2]

C = cbind(B1, B2)
print(C)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    3    5    7
## [2,]    2    4    6    8
```

List

```
l1 = list(1, 2, 3)
print(l1)
```

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

```
class(l1)
```

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

```
names(l1) <- c("a", "b", "c")
print(l1)
```

```
## $a
## [1] 1
##
## $b
## [1] 2
##
## $c
## [1] 3
```

```
l1[[1]]  ## subsetting the first element
```

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

```
l1$a  ## subsetting the element named a
```

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

```
l1$a = 4
print(l1$a)
```

```
## [1] 4
```

We can use `R` as a calculater, e.g. $2 * 2$, $\log(2)$, $\sqrt{2}$, 2^3.

```
x = 2
print(x * 2)
```

```
## [1] 4
```

```
print(log(x))
```

```
## [1] 0.6931472
```

```
print(sqrt(x))
```

```
## [1] 1.414214
```

```
print(x ^ 3)
```

```
## [1] 8
```

```
print(x ** 3)
```

```
## [1] 8
```

Linear Algebra

```
A = matrix(c(2, 3, 1, 5), nrow=2, ncol=2)

## transpose
t(A)
```

```
##      [,1] [,2]
## [1,]    2    3
## [2,]    1    5
```

```
## matrix addition
B = matrix(c(2, 2, 3, 5), nrow=2, ncol=2)

A+B
```

```
##      [,1] [,2]
## [1,]    4    4
## [2,]    5   10
```

```
## matrix multiplication
A%*%B
```

```
##      [,1] [,2]
## [1,]    6   11
## [2,]   16   34
```

```
### elementwise multiplication
A * B
```

```
##      [,1] [,2]
## [1,]    4    3
## [2,]    6   25
```

```
A = matrix(c(2, 1, 1, 5), nrow=2, ncol=2)
b = c(1,2)
## solve the system Ax = b
solve(A, b)
```

```
## [1] 0.3333333 0.3333333
```

```
## compute cholesky decomposition
R = chol(A)
## use triagular solvers
backsolve(R, backsolve(R, b,transpose=TRUE))
```

```
## [1] 0.3333333 0.3333333
```

Data frame

```
y = 10:12
print(y)
```

```
## [1] 10 11 12
```

```
z = c(1, 3, 5)
print(z)
```

```
## [1] 1 3 5
```

```
print(z[1])
```

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

```
df = data.frame(y = y, z = z)
print(df)
```

```
##    y z
## 1 10 1
## 2 11 3
## 3 12 5
```

```
print(class(df))
```

```
## [1] "data.frame"
```

```
print(df$y)
```

```
## [1] 10 11 12
```

```
print(df$z)
```

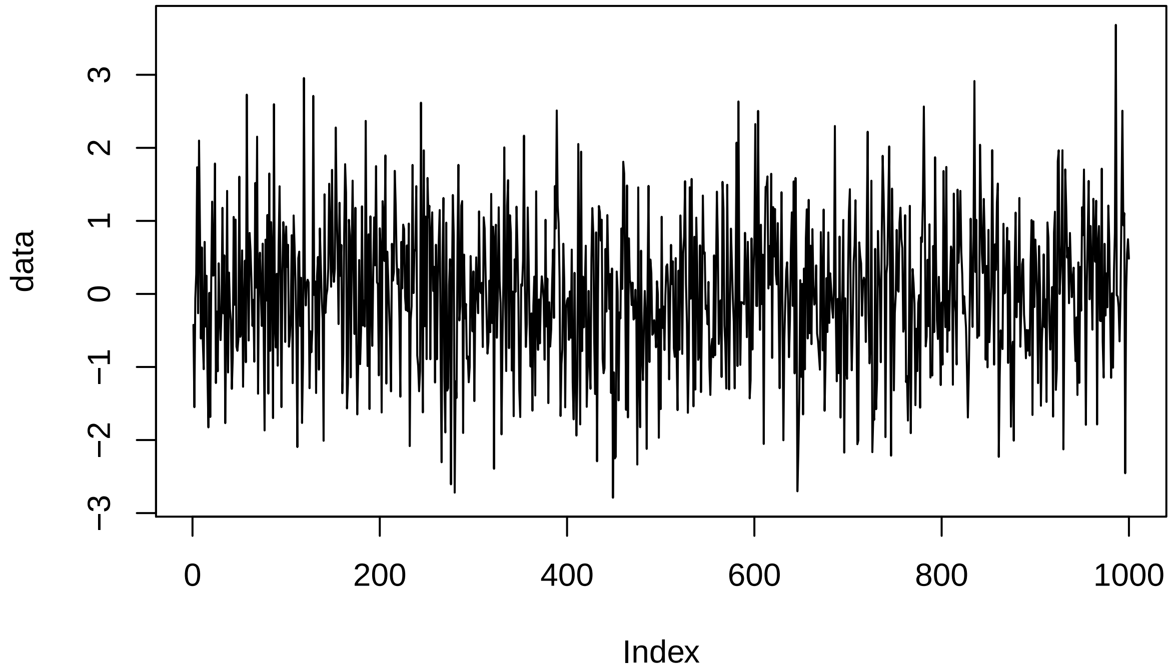
```
## [1] 1 3 5
```

Exercise: Create a data frame containing name, gender, grades, etc.

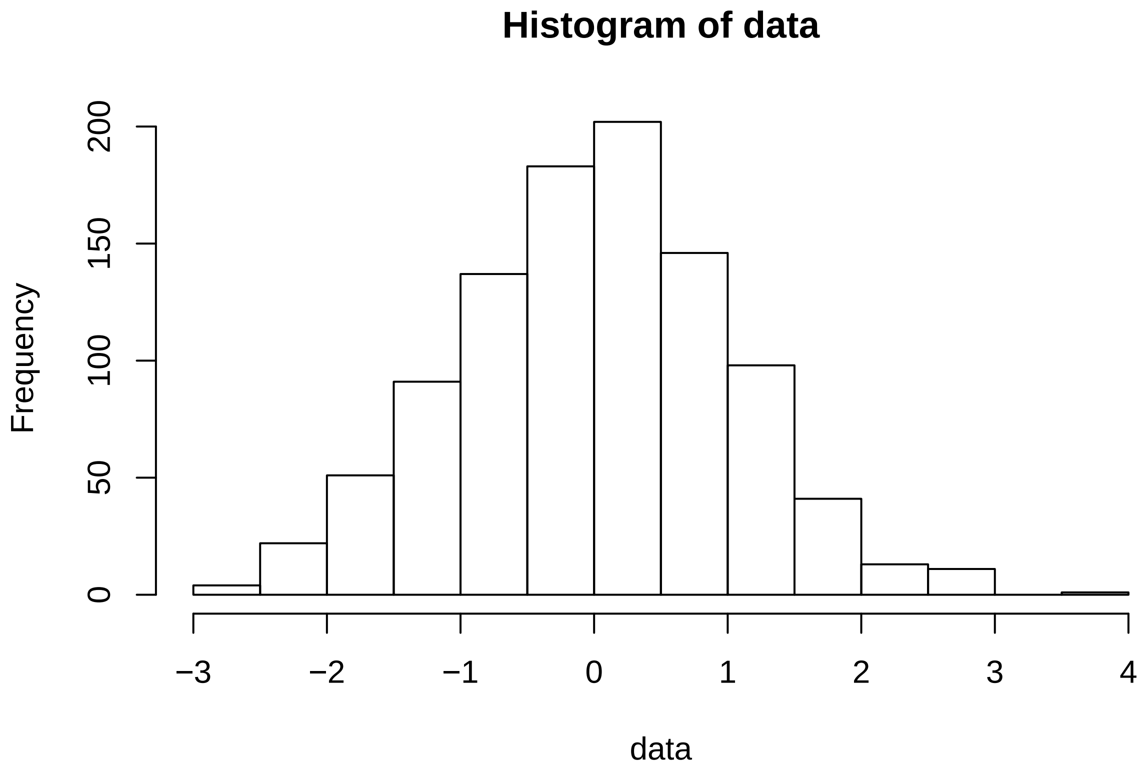
Basic plots

Use of "hist" function

```
set.seed(2018)
data = rnorm(1000)
plot(data, type = 'l')
```



```
hist(data)
```



Use of "plot" function

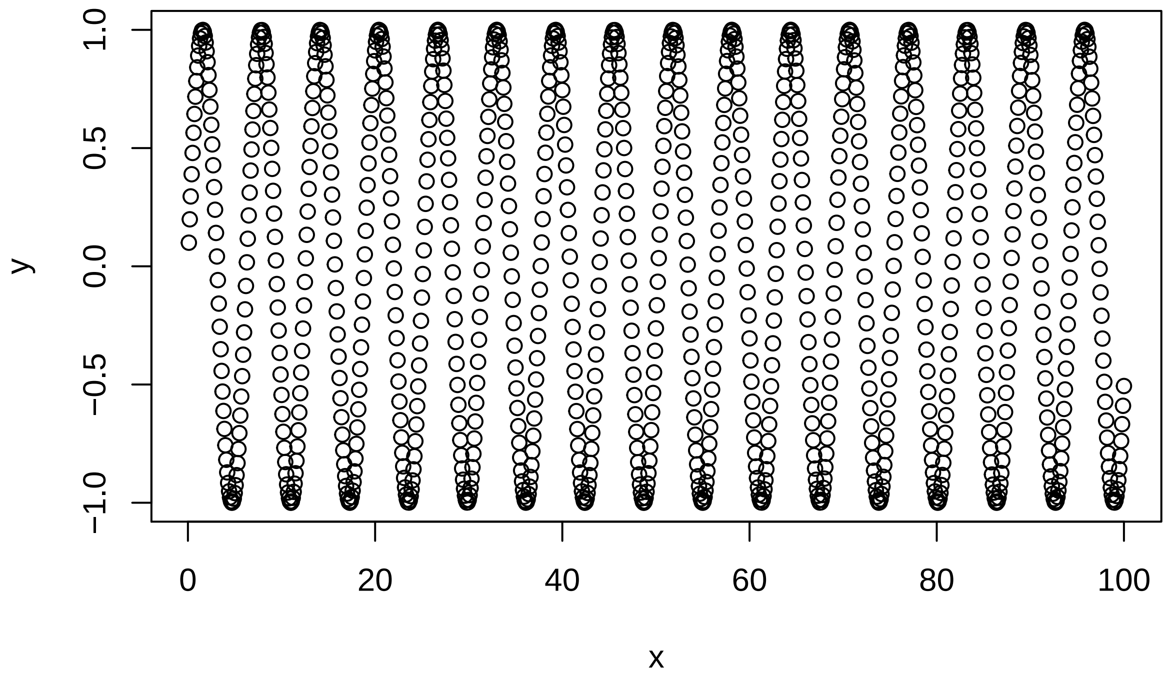
```r
x = 1:1000/10
y = sin(x)
print(head(x))
```

```
## [1] 0.1 0.2 0.3 0.4 0.5 0.6
```

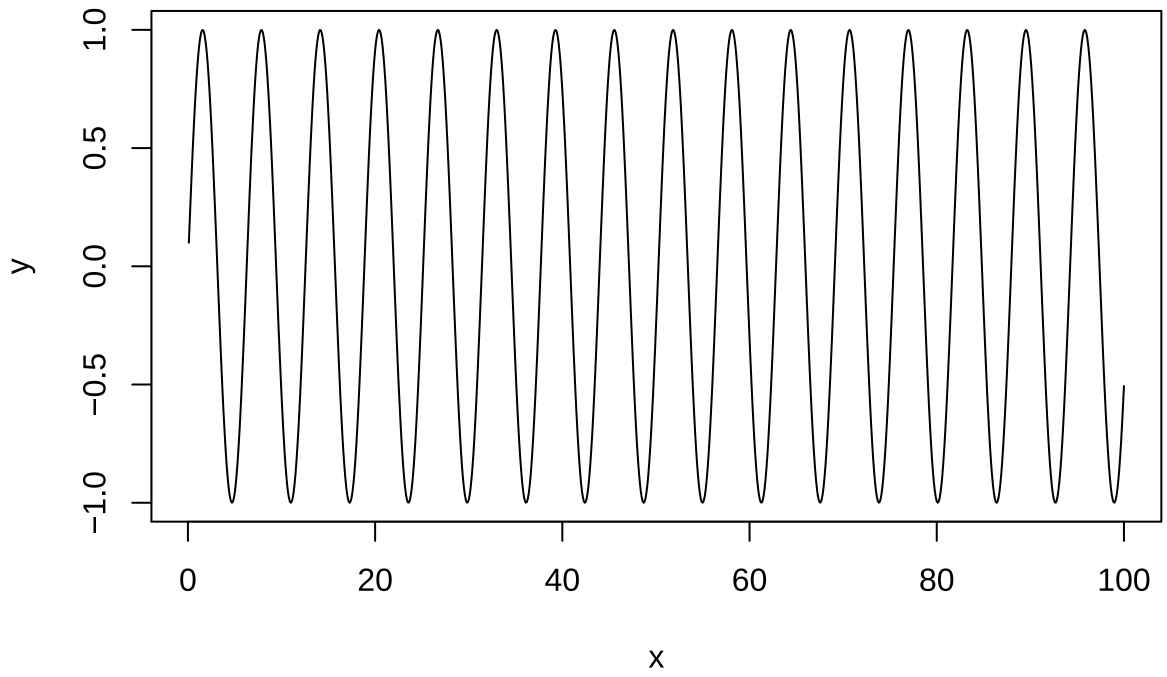
```r
print(tail(y))
```

```
## [1] -0.8577953 -0.8021964 -0.7385822 -0.6675884 -0.5899242 -0.5063656
```

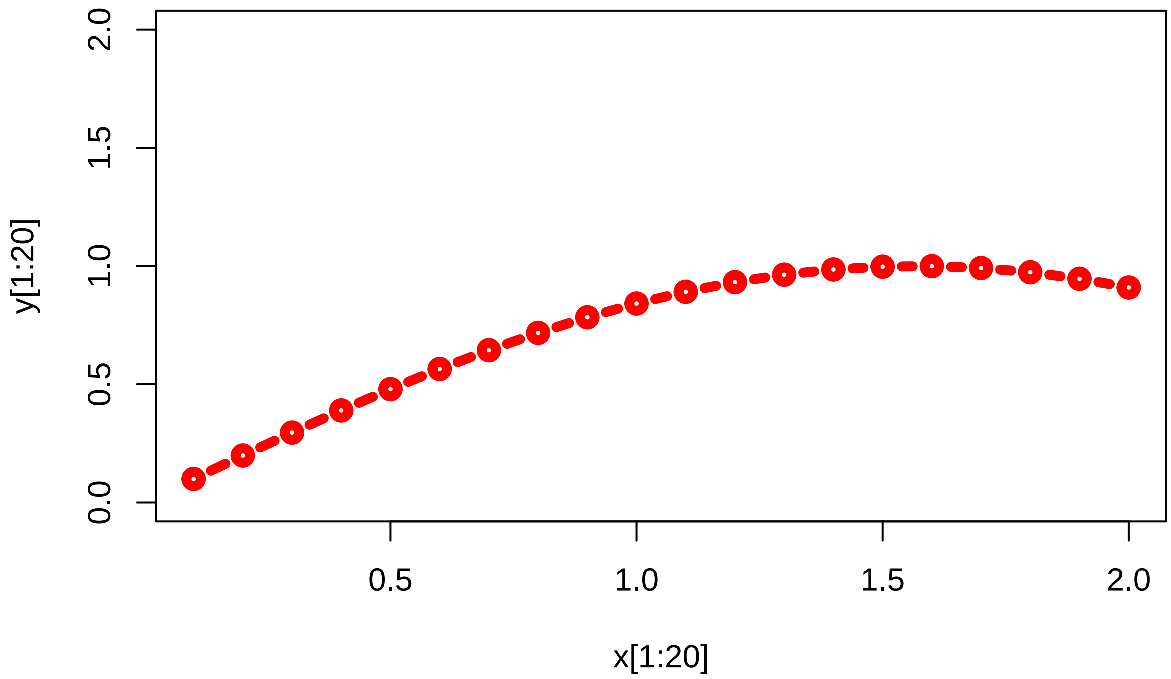
```r
plot(x, y)
```



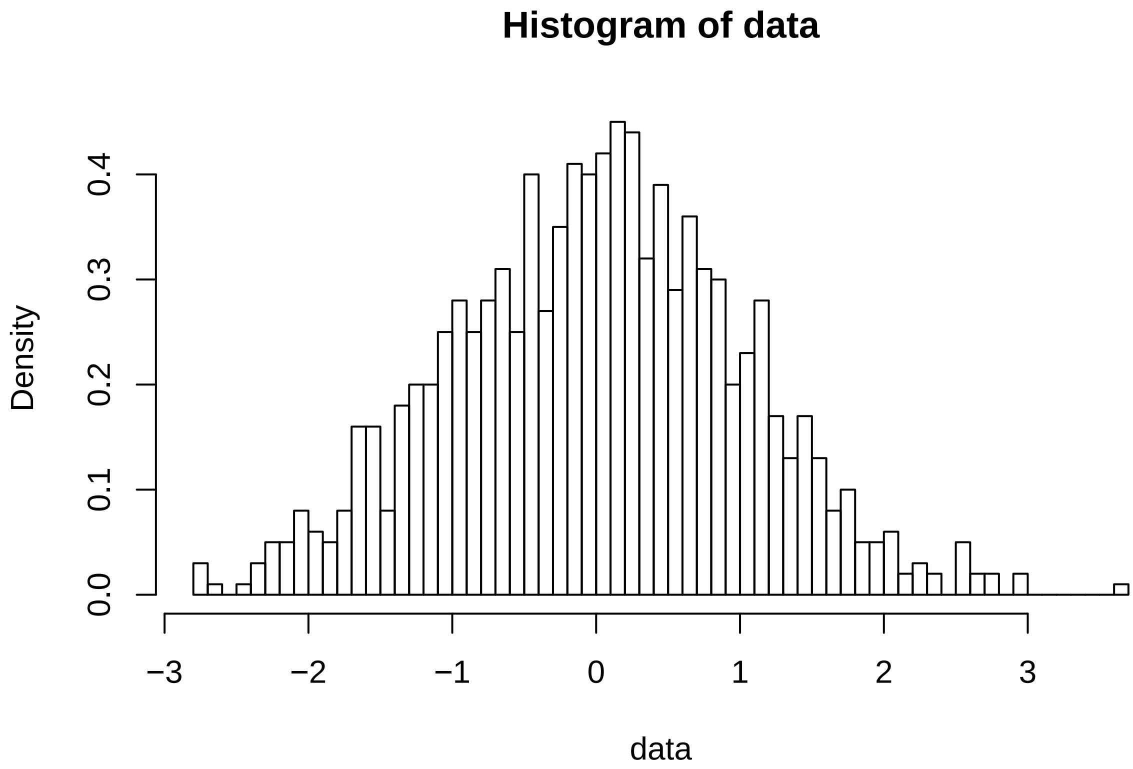
```r
plot(x, y, type = "l")
```



```r
plot(x[1:20], y[1:20], type = 'b', col = 'red', lwd = 5, ylim = c(0, 2))
```



Exercise: Use `hist()` and change parameters to generate the figure as follow.



Importing Data

Download data from https://github.com/LiXinyi/SAMSI_Diversity_Workshop/blob/master/CanadianWeather_month.csv. Original data are available at R package `fda`.

- Importing a single file

```
dat = read.csv("CanadianWeather_month.csv", header = TRUE)
print(class(dat))
```

```
## [1] "data.frame"
```

```
print(dim(dat))
```

```
## [1] 420   4
```

```
print(head(dat))
```

```
##        Temp   Precip Month    Region
## 1 -4.654839 4.651613   Jan St. Johns
## 2 -5.325000 4.735714   Feb St. Johns
## 3 -2.532258 4.235484   Mar St. Johns
## 4  1.256667 3.616667   Apr St. Johns
## 5  5.793548 3.251613   May St. Johns
## 6 10.786667 3.270000   Jun St. Johns
```

```
print(names(dat))
```

```
## [1] "Temp"   "Precip" "Month"  "Region"
```

```
print(table(dat$Month))
```

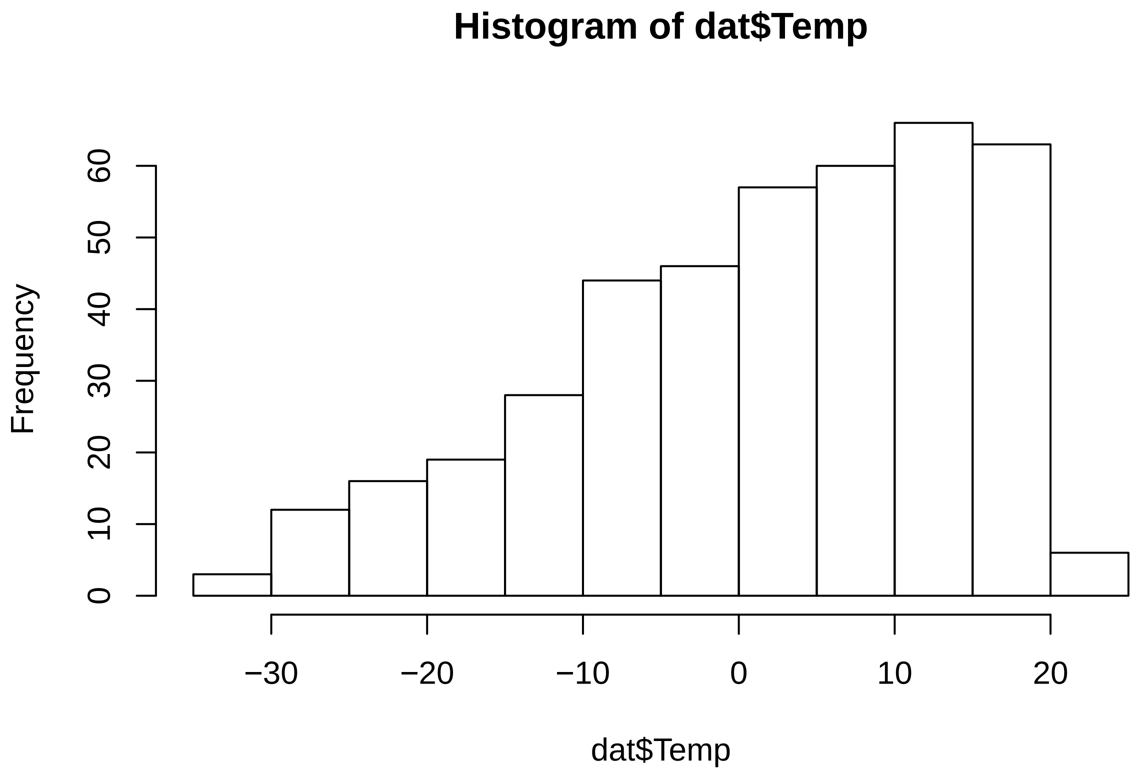
```
##
## Apr Aug Dec Feb Jan Jul Jun Mar May Nov Oct Sep
##  35  35  35  35  35  35  35  35  35  35  35  35
```

```
print(table(dat$Region))
```

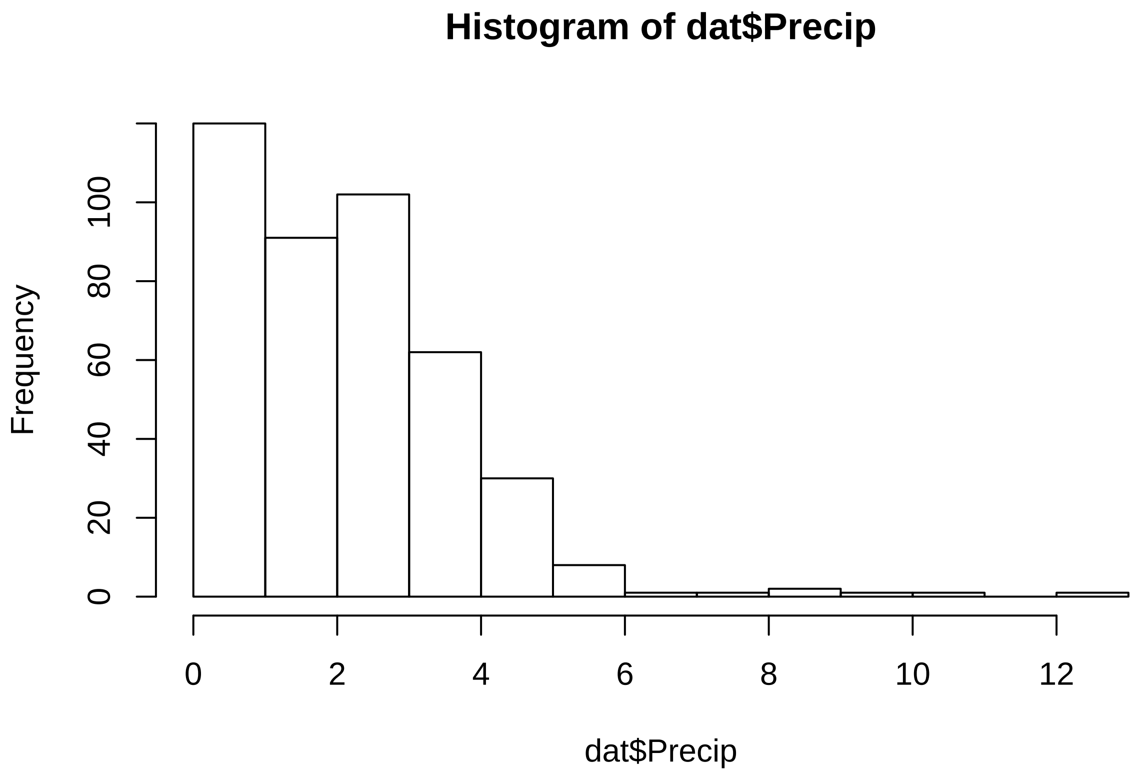
```
##
##      Arvida Bagottville     Calgary  Charlottvl   Churchill      Dawson
##          12          12          12          12          12          12
##    Edmonton Fredericton     Halifax      Inuvik     Iqaluit    Kamloops
##          12          12          12          12          12          12
##      London    Montreal      Ottawa  Pr. Albert  Pr. George  Pr. Rupert
##          12          12          12          12          12          12
##      Quebec      Regina    Resolute Scheffervll  Sherbrooke   St. Johns
##          12          12          12          12          12          12
##      Sydney     The Pas  Thunderbay     Toronto Uranium Cty   Vancouver
##          12          12          12          12          12          12
##    Victoria  Whitehorse    Winnipeg    Yarmouth Yellowknife
##          12          12          12          12          12
```

Exercise: Use `hist()` and `plot()` to get basic idea of the data.

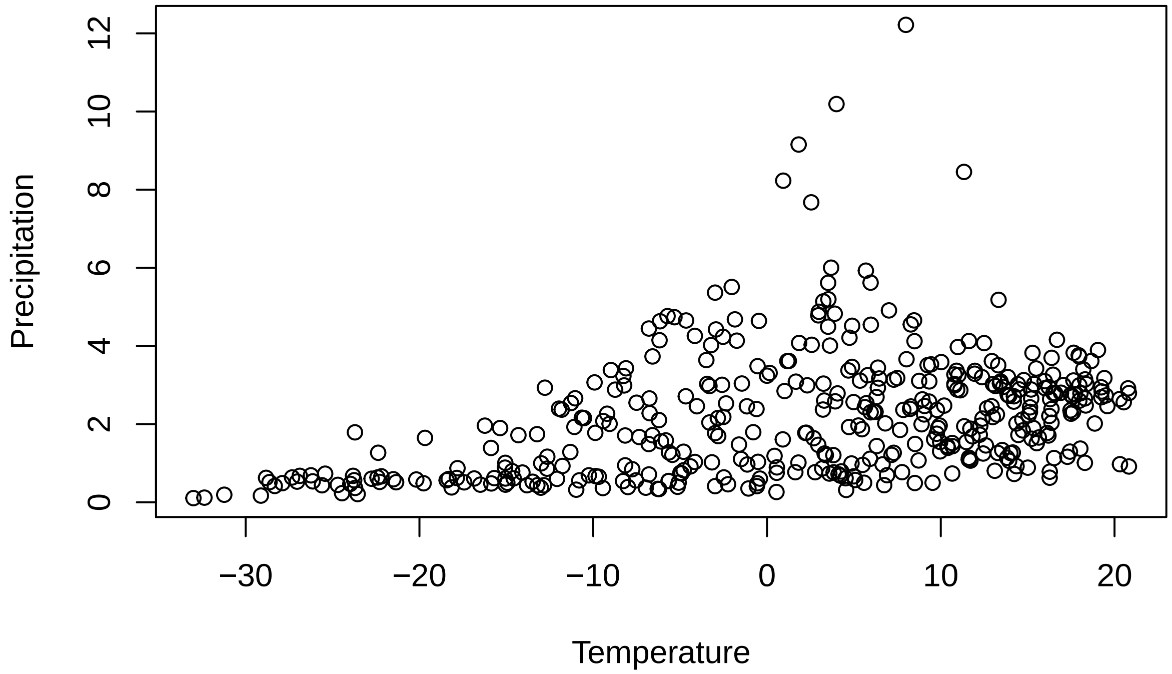
```
hist(dat$Temp)
```



```r
hist(dat$Precip)
```



```r
plot(dat$Temp, dat$Precip, xlab = "Temperature", ylab = "Precipitation")
```



Packages

How to install a package

Install from source

Download the add-on R package, for example, "fda", put it in the directory "/data/Rpackages", and install the package using the command:

```
install.packages("fda", lib = "/data/Rpackages")
```

Install from repository

Vast array of packages are available at the Comprehensive R Archive Network (CRAN) and BioConductor repositories. Both CRAN and BioConductor are open source, well structured, tested and operating. While both repositories provide abundant packages covering various data analysis tasks, BioConductor is more focused on providing tools for the analysis of high-throughtput genomic data. In addition, there are slight differences in the command for package installation.

- Install from CRAN (e.g. R package "fda"):

```
install.packages("fda", repos = "http://cran.us.r-project.org")
```

- Install from BioConductor (e.g. R package "dada2"):

```
## try http:// if https:// URLs are not supported
source("https://bioconductor.org/biocLite.R")
```

```
## Bioconductor version 3.6 (BiocInstaller 1.28.0), ?biocLite for help

## A new version of Bioconductor is available after installing the most
##   recent version of R; see http://bioconductor.org/install
```

```
biocLite()
```

```
## BioC_mirror: https://bioconductor.org

## Using Bioconductor 3.6 (BiocInstaller 1.28.0), R 3.4.0 (2017-04-21).

## installation path not writeable, unable to update packages: boot, class,
##   cluster, codetools, foreign, lattice, Matrix, mgcv, nlme, rpart,
##   survival

## Old packages: 'assertthat', 'backports', 'BH', 'callr', 'caret',
##   'caTools', 'cli', 'colorspace', 'commonmark', 'curl', 'data.table',
##   'deldir', 'desc', 'devtools', 'digest', 'dplyr', 'e1071', 'earth',
##   'evaluate', 'flexclust', 'forcats', 'foreach', 'formatR', 'GenSA',
##   'geoR', 'ggplot2', 'git2r', 'glmnet', 'glue', 'gower', 'GPfit',
##   'gtable', 'haven', 'highr', 'httpuv', 'httr', 'installr', 'interp',
##   'ipred', 'iterators', 'jsonlite', 'knitr', 'KRLS', 'later', 'lazyeval',
##   'lhs', 'lme4', 'maptools', 'markdown', 'MASS', 'matlib', 'MaxPro',
##   'mime', 'mlegp', 'modeltools', 'nloptr', 'openssl', 'openxlsx',
##   'pillar', 'plotmo', 'plotrix', 'polynom', 'pROC', 'processx', 'ps',
##   'purrr', 'quantreg', 'R.matlab', 'R.oo', 'R.utils', 'RandomFields',
##   'RandomFieldsUtils', 'randtoolbox', 'Rcpp', 'RcppEigen', 'readr',
##   'readxl', 'recipes', 'rgl', 'rio', 'rJava', 'rlang', 'rmarkdown',
##   'roxygen2', 'rprojroot', 'rstudioapi', 'shiny', 'sp', 'stringi',
##   'stringr', 'testthat', 'tibble', 'tidyr', 'tinytex', 'xfun', 'xgboost',
##   'xml2', 'xtable', 'zip'
```

```
biocLite("dada2")
```

Sampling from Basic Distributions

```
## Uniform distribution
runif(n = 2, min = 0, max = 1)
```

```
## [1] 0.6007556 0.8535367
```

```
## Normal distribution
rnorm(n = 2, mean = 0, sd = 1)
```

```
## [1] -1.925972 -2.606258
```

- Multivariate Normal Distribution

```
options(warn=-1)
library("mvtnorm")
library("plotrix")
## Need package mvtnorm
Sigma <- matrix(c(10,3,3,2),2,2)
rmvnorm(n = 1, mean = rep(0, nrow(Sigma)), sigma = Sigma)
```

```
##           [,1]       [,2]
## [1,] 0.1945828 -0.2268338
```

Calculating π

```
## Need package plotrix
plot(0.5,0.5,type ="p",asp = 1, xlim=c(0,1)
     ,ylim=c(0,1),color = "black")
draw.circle(0.5,0.5,0.5,nv=1000
            ,border=NULL,col=NA,lty=1,lwd=1)
rect( 0, 0, 1, 1)

## Need package plotrix
plot(0.5,0.5,type ="p",asp = 1, xlim=c(0,1)
     ,ylim=c(0,1),color = "black")
draw.circle(0.5,0.5,0.5,nv=1000
            ,border=NULL,col=NA,lty=1,lwd=1)
rect( 0, 0, 1, 1)

plot.new()
plot(0.5,0.5,type ="p",asp = 1, xlim=c(0,1),ylim=c(0,1),color = "black")
draw.circle(0.5,0.5,0.5,nv=1000,border=NULL,col=NA,lty=1,lwd=1)
rect( 0, 0, 1, 1)
mcpiwplot <- function(n){
  m = 0
  x <- runif(n)
  y <- runif(n)
  x1 <- x - 0.5
  y1 <- y - 0.5 #circle has center at (0.5,0.5)
  r2 <- x1^2 + y1^2
  for(i in 1:n){
    if(r2[i] <= 0.25){
      m <- m+1
      points(x[i],y[i] , col = "red")
    }
    else{
      points(x[i],y[i], col = "blue")
    }
  }
  piapprox <- m/(0.25*n)
  return(((piapprox-pi)/pi))
}
mcpi <- function(n){
  m = 0
  x <- runif(n)
  y <- runif(n)
  x1 <- x - 0.5
  y1 <- y - 0.5 #circle has center at (0.5,0.5)
  r2 <- x1^2 + y1^2
  for(i in 1:n){
    if(r2[i] <= 0.25){
      m <- m+1
    }
  }
  piapprox <- m/(0.25*n)
  return(((piapprox-pi)/pi))
}
```

```r
mcpiwplot(50)

m = 20
error <- 1:m
for(i in 1:m){
  errore <- 1:100
  for(j in 1:100){
    errore[j] <- mcpi(i*15)
  }
  error[i] <- sum(abs(errore))/100
}
plot.new()
plot(1:m*20,error,type = "l",xlab = "Sample size", ylab ="Error" )
```