Packages in R

Panchatcharam M

Associate Professor

Department of Mathematics and Statistics,
IIT Tirupati

PACKAGES

- ✓ A set of R functions
- ✓ A set of compiled code
- ✓ A Set of sample data

- ✓ Library directory in R environment
- ✓ A few packages are installed by default and can be called directly
- ✓ A few packages are installed but should be loaded explicitly

- ✓ **CRAN: Official Repos**
- ✓ **Biconductor: Topic-specific Rep**
- ✓ **GitHub: Open source project repo**

library()

✓ **Lists all the R packages installed**

```
Packages in library 'C:/Users/mpanc/AppData/Local/R/win-library/4.3':

cli                             Helpers for Developing Command Line Interfaces
colorspace                      A Toolbox for Manipulating and Assessing Colors and Palettes
distributions3                  Probability Distributions as S3 Objects
ellipsis                        Tools for Working with ...
fansi                           ANSI Control Sequence Aware String Functions
farver                          High Performance Colour Space Manipulation
ggplot2                         Create Elegant Data Visualisations Using the Grammar of Graphics
glue                            Interpreted String Literals
gtable                          Arrange 'Grobs' in Tables
isoband                         Generate Isolines and Isobands from Regularly Spaced Elevation Grids
labeling                        Axis Labeling
lestat                          A Package for Learning Statistics
lifecycle                       Manage the Life Cycle of your Package Functions
magrittr                        A Forward-Pipe Operator for R
munsell                         Utilities for Using Munsell Colours
pillar                          Coloured Formatting for Columns
pkgconfig                       Private Configuration for 'R' Packages
R6                              Encapsulated Classes with Reference Semantics
RColorBrewer                    ColorBrewer Palettes
rlang                           Functions for Base Types and Core R and 'Tidyverse' Features
scales                          Scale Functions for Visualization
tibble                          Simple Data Frames
utf8                            Unicode Text Processing
vctrs                           Vector Helpers
VGAM                            Vector Generalized Linear and Additive Models
viridisLite                     Colorblind-Friendly Color Maps (Lite Version)
withr                           Run Code 'With' Temporarily Modified Global State

Packages in library 'C:/Program Files/R/R-4.3.3/library':

base                            The R Base Package
boot                            Bootstrap Functions (Originally by Angelo Canty for S)
class                           Functions for Classification
cluster                         "Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al.
codetools                       Code Analysis Tools for R
compiler                        The R Compiler Package
datasets                        The R Datasets Package
foreign                         Read Data Stored by 'Minitab', 'S', 'SAS', 'SPSS', 'Stata', 'Systat',
                                'Weka', 'dBase', ...
graphics                        The R Graphics Package
grDevices                       The R Graphics Devices and Support for Colours and Fonts
grid                            The Grid Graphics Package
KernSmooth                      Functions for Kernel Smoothing Supporting Wand & Jones (1995)
lattice                         Trellis Graphics for R
MASS                            Support Functions and Datasets for Venables and Ripley's MASS
Matrix                          Sparse and Dense Matrix Classes and Methods
methods                         Formal Methods and Classes
mgcv                            Mixed GAM Computation Vehicle with Automatic Smoothness Estimation
nlme                            Linear and Nonlinear Mixed Effects Models
nnet                            Feed-Forward Neural Networks and Multinomial Log-Linear Models
parallel                        Support for Parallel Computation in R
rpart                           Recursive Partitioning and Regression Trees
spatial                         Functions for Kriging and Point Pattern Analysis
splines                         Regression Spline Functions and Classes
stats                           The R Stats Package
stats4                          Statistical Functions using S4 Classes
survival                        Survival Analysis
tcltk                           Tcl/Tk Interface
tools                           Tools for Package Development
translations                    The R Translations Package
utils                           The R Utils Package
```

Install R Packages

```
install.packages("Package Name")
install.packages("VGAM")
install.packages(c("VGAM","distributions3"))
installed.packages()
update.packages()
```

Load R Packages

```
#you can load using library function
library(graphics)
Library(graphics,ggplot2,caret)
#you can load using require function
require(graphics)
require(ggplot2)
require(caret)
```

library() : Load a package and refers to a place where package is available

packages(): Collection functions bundled. You can share your functions with others using package

Built-in functions
Statistics and probability

Built-in Functions-Normal Distribution

- ✓ **R has various statistical probability functions to perform the statistical task**
- ✓ **You can find normal density, normal quartile, etc**
- ✓ **Density, distribution function, quantile function and random generation for the normal distribution with mean equal to mean and standard deviation equal to sd.**

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Built-in Functions-Normal Distribution

```
normal(expectation = 0, lambda, P = 1)
```

Function	Description
Expectation	The expectation of the distribution
lambda	The natural logarithm of the standard deviation of the distribution.
P	Precision of the distribution, inverse of the variance.

```
require(lestat)
dist <- normal(3, log(0.7))
variance(dist)
dist <- normal(5, log(0.49)/2)
variance(dist)
dist <- normal(7, P = 2)
variance(dist)
```

Built-in Functions-Normal Distribution

```
Normal(mu = 0, sigma = 1)
```

Function	Description
Normal	It creates a normal distribution
Mu	Mean
Sigma	Standard deviation

Built-in Functions-Normal Distribution

```
dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
rnorm(n, mean = 0, sd = 1)
```

Function	Description
dnorm	Density
pnorm	Distribution Function
qnorm	Quantile Function
rnorm	Random deviates

Built-in Functions-Normal Distribution

```
dnorm(x, mean = 0, sd = 1, log = FALSE)
pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
rnorm(n, mean = 0, sd = 1)
```

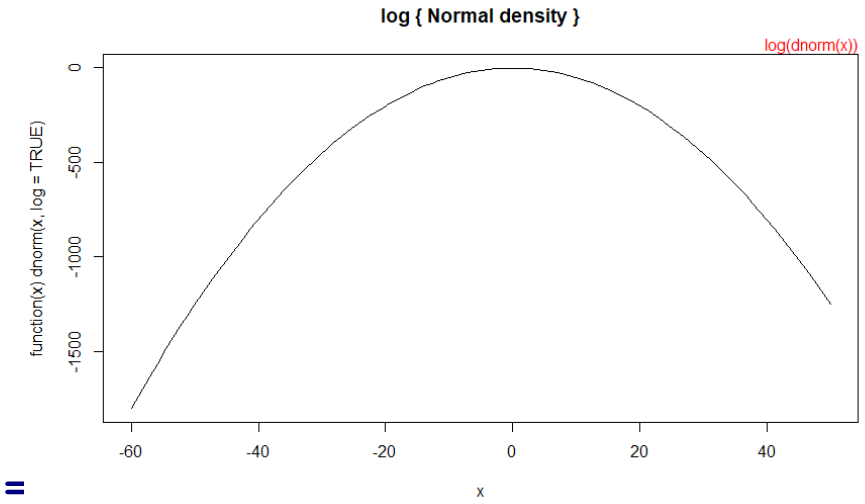
Argument	Description
x, q	vector of quantiles.
p	vector of probabilities.
n	number of observations. If length(n) > 1, the length is taken to be the number required.
mean	vector of means.
sd	vector of standard deviations.
log, log.p	logical; if TRUE, probabilities p are given as log(p).
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$ Otherwise $P[X > x]$

Built-in Functions-Normal Distribution

```
require(graphics)

dnorm(0) == 1/sqrt(2*pi)
dnorm(1) == exp(-1/2)/sqrt(2*pi)
dnorm(1) == 1/sqrt(2*pi*exp(1))

par(mfrow = c(2,1))
plot(function(x) dnorm(x, log = TRUE), -60, 50,
     main = "log { Normal density }")
curve(log(dnorm(x)), add = TRUE, col = "red", lwd =
mtext("dnorm(x, log=TRUE)", adj = 0)
mtext("log(dnorm(x))", col = "red", adj = 1)
```



Built-in Functions-Bivariate Normal Distribution

```
dbinorm(x1, x2, mean1 = 0, mean2 = 0, var1 = 1, var2 = 1, cov12 = 0,log = FALSE)
pbinorm(q1, q2, mean1 = 0, mean2 = 0, var1 = 1, var2 = 1, cov12 = 0)
rbinorm(n,      mean1 = 0, mean2 = 0, var1 = 1, var2 = 1, cov12 = 0)
pnorm2(x1, x2, mean1 = 0, mean2 = 0, var1 = 1, var2 = 1, cov12 = 0)
```

Function	Description
dbinorm	Density
pbinorm	Distribution Function
rbinorm	Random Deviates (n by 2 matrix)

Built-in Functions-Bivariate Normal Distribution

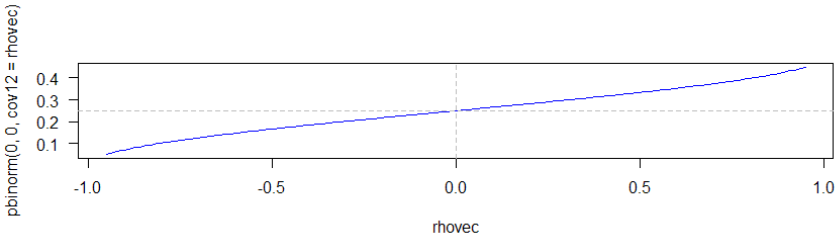
```
dbinorm(x1, x2, mean1 = 0, mean2 = 0, var1 = 1, var2 = 1, cov12 = 0,log = FALSE)
pbinorm(q1, q2, mean1 = 0, mean2 = 0, var1 = 1, var2 = 1, cov12 = 0)
rbinorm(n,      mean1 = 0, mean2 = 0, var1 = 1, var2 = 1, cov12 = 0)
pnorm2(x1, x2, mean1 = 0, mean2 = 0, var1 = 1, var2 = 1, cov12 = 0)
```

Argument	Description
x1,x2, q1,q2	vector of quantiles.
n	number of observations. If length(n) > 1, the length is taken to be the number required.
mean1,mean2,var1,var2	vector of means, variances
Cov12	Covariance
Log	Logical. If log = TRUE then the logarithm of the density is returned.

Built-in Functions-Bivariate Normal Distribution

```
install.packages('VGAM')
require(VGAM)
require(graphics)
yvec <- c(-5, -1.96, 0, 1.96, 5)
ymat <- expand.grid(yvec, yvec)
cbind(ymat, pbinorm(ymat[, 1], ymat[, 2]))

rhovec <- seq(-0.95, 0.95, by = 0.01)
plot(rhovec, pbinorm(0, 0, cov12 = rhovec),
     type = "l", col = "blue", las = 1)
abline(v = 0, h = 0.25, col = "gray", lty = "dashed")
```



Built-in Functions-Binormal Distribution

Maximum likelihood estimation of the five parameters of a bivariate normal distribution.

```
binormal(lmean1 = "identitylink", lmean2 = "identitylink",
         lsd1   = "loglink",      lsd2   = "loglink",
         lrho   = "rhobitlink",
         imean1 = NULL,           imean2 = NULL,
         isd1   = NULL,           isd2   = NULL,
         irho   = NULL,           imethod = 1,
         eq.mean = FALSE,         eq.sd = FALSE,
         zero = c("sd", "rho"), rho.arg = NA)
```

Function	Description
lmean1,lmean2	Link functions applied to the means,
lsd1,lsd2	Link functions applied to the standard deviations,
Lrho	Link functions applied to the standard rho,

Built-in Functions-Bivariate Poisson Distribution

```
dpois(x, lambda, log = FALSE)
ppois(q, lambda, lower.tail = TRUE, log.p = FALSE)
qpois(p, lambda, lower.tail = TRUE, log.p = FALSE)
rpois(n, lambda)
```

Function	Description
dpois	Density
ppois	Distribution Function
rpois	Random Deviates
qpois	Quantile Function

$$p(x) = \frac{\lambda^x e^{-\lambda}}{x!}$$

Built-in Functions - Poisson Distribution

```
dpois(x, lambda, log = FALSE)
ppois(q, lambda, lower.tail = TRUE, log.p = FALSE)
qpois(p, lambda, lower.tail = TRUE, log.p = FALSE)
rpois(n, lambda)
```

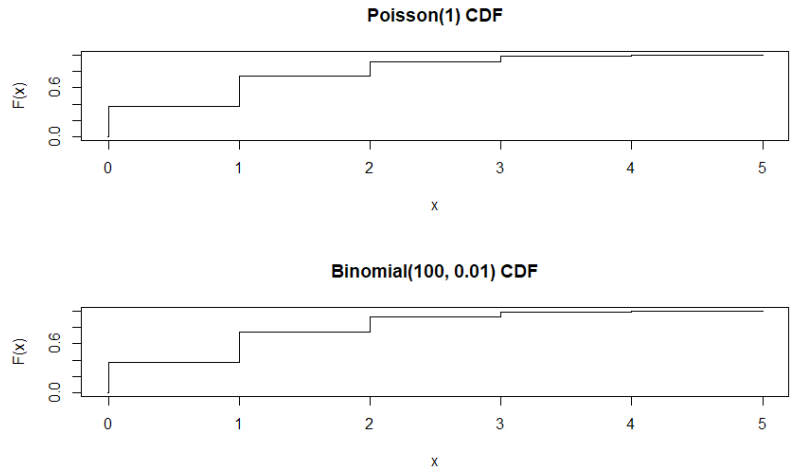
Argument	Description
x, q	vector of quantiles, where x is non-negative integer
p	vector of probabilities.
n	Number of random values to return
Lambda	vector of means (non-negative).
sd	vector of standard deviations.
log, log.p	logical; if TRUE, probabilities p are given as log(p).
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$ Otherwise $P[X > x]$

Built-in Functions - Poisson Distribution

```
require(graphics)

-log(dpois(0:7, lambda = 1) * gamma(1+ 0:7)) # == 1
Ni <- rpois(50, lambda = 4); table(factor(Ni, 0:max(Ni)))
1 - ppois(10*(15:25), lambda = 100)  # becomes 0 (cancellation)
ppois(10*(15:25), lambda = 100, lower.tail = FALSE)  # no cancellation

par(mfrow = c(2, 1))
x <- seq(-0.01, 5, 0.01)
plot(x, ppois(x, 1), type = "s", ylab = "F(x)", main = "Poisson(1) CDF")
plot(x, pbinom(x, 100, 0.01), type = "s", ylab = "F(x)",
     main = "Binomial(100, 0.01) CDF")
```



Built-in Functions-Uniform Distribution

```
dunif(x, min = 0, max = 1, log = FALSE)
punif(q, min = 0, max = 1, lower.tail = TRUE, log.p = FALSE)
qunif(p, min = 0, max = 1, lower.tail = TRUE, log.p = FALSE)
runif(n, min = 0, max = 1)
```

Function	Description
dunif	Density
punif	Distribution Function
runif	Random Deviates
qunif	Quantile Function

$$f(x) = \frac{1}{max - min}$$

Built-in Functions - Poisson Distribution

```
dunif(x, min = 0, max = 1, log = FALSE)
punif(q, min = 0, max = 1, lower.tail = TRUE, log.p = FALSE)
qunif(p, min = 0, max = 1, lower.tail = TRUE, log.p = FALSE)
runif(n, min = 0, max = 1)
```

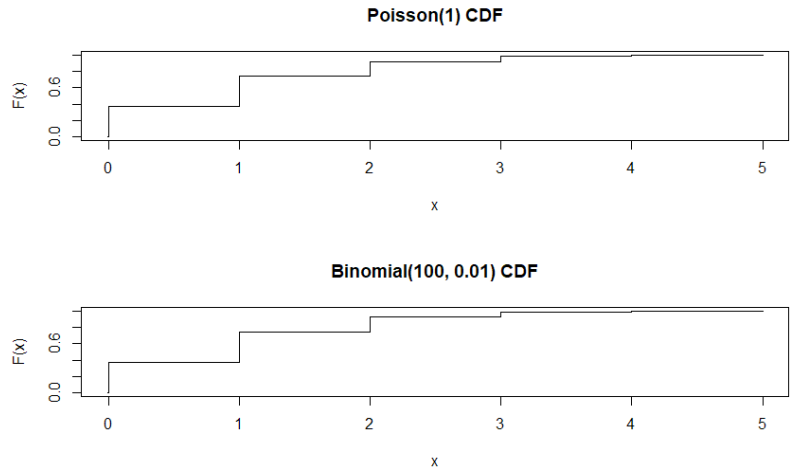
Argument	Description
x, q	vector of quantiles
p	vector of probabilities.
n	number of observations. If length(n) > 1, the length is taken to be the number required.
min,max	lower and upper limits of the distribution. Must be finite.
log, log.p	logical; if TRUE, probabilities p are given as log(p).
lower.tail	logical; if TRUE (default), probabilities are $P[X \leq x]$ Otherwise $P[X > x]$

Built-in Functions - Poisson Distribution

```
require(graphics)

-log(dpois(0:7, lambda = 1) * gamma(1+ 0:7)) # == 1
Ni <- rpois(50, lambda = 4); table(factor(Ni, 0:max(Ni)))
1 - ppois(10*(15:25), lambda = 100)  # becomes 0 (cancellation)
ppois(10*(15:25), lambda = 100, lower.tail = FALSE)  # no cancellation

par(mfrow = c(2, 1))
x <- seq(-0.01, 5, 0.01)
plot(x, ppois(x, 1), type = "s", ylab = "F(x)", main = "Poisson(1) CDF")
plot(x, pbinom(x, 100, 0.01), type = "s", ylab = "F(x)",
     main = "Binomial(100, 0.01) CDF")
```



CHARTS

➢ ***Values in data vector as height of the bars***

✓ Default Syntax

```
barplot(height, width = 1, space = NULL,
        names.arg = NULL, legend.text = NULL, beside = FALSE,
        horiz = FALSE, density = NULL, angle = 45,
        col = NULL, border = par("fg"),
        main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
        xlim = NULL, ylim = NULL, xpd = TRUE, log = "",
        axes = TRUE, axisnames = TRUE,
        cex.axis = par("cex.axis"), cex.names = par("cex.axis"),
        inside = TRUE, plot = TRUE, axis.lty = 0, offset = 0,
        add = FALSE, ann = !add && par("ann"), args.legend = NULL,
…)
```

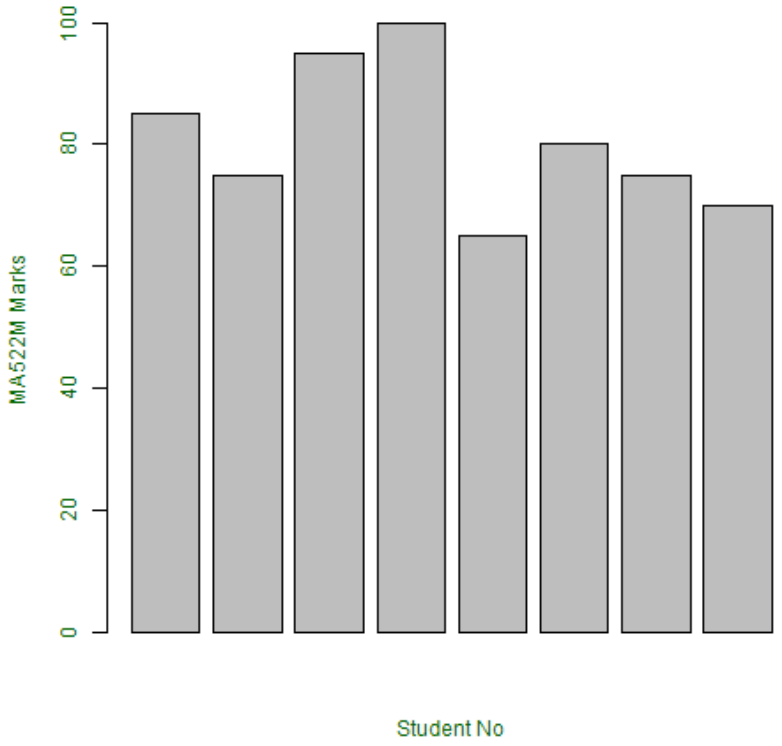
- ***Values in data vector as height of the bars***

✓ For Formula

```
barplot(formula, data, subset, na.action,
        horiz = FALSE, xlab = NULL, ylab = NULL, …)
```

Bar Plot/Chart-Example

```
marks=c(85,75,95,100,65,80,75,70)
png(file="barplot.png")
barplot(marks,xlab="Student No",ylab="MA522M
Marks",col.axis="darkgreen",col.lab="darkgreen")
dev.off()
```



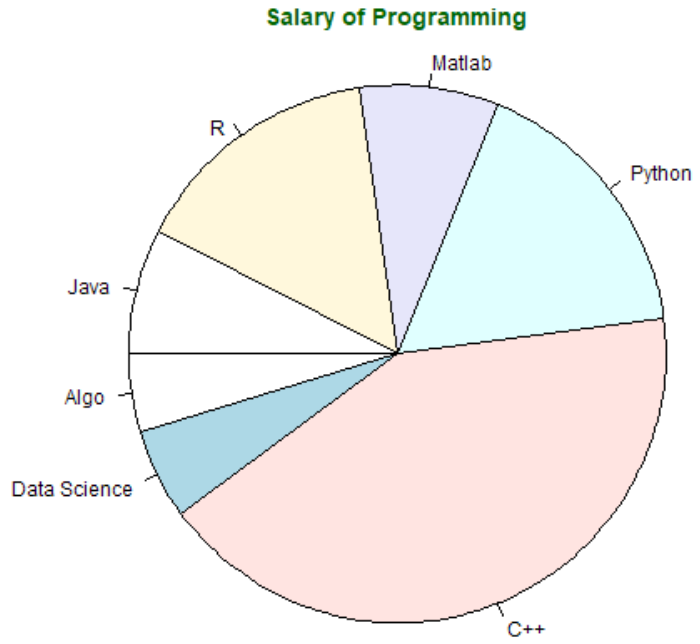
➢ ***Circular chart divided into different segments according to the ratio***

✓ Syntax

```
pie(x, labels = names(x), edges = 200, radius = 0.8,
    clockwise = FALSE, init.angle = if(clockwise) 90 else 0,
    density = NULL, angle = 45, col = NULL, border = NULL,
    lty = NULL, main = NULL, …)
```

Pie Plot/Chart-Example

```
salary=c(28,32,250,100,50,90,45)
names(salary)=c("Algo","Data Science","C++","Python","Matlab","R","Java")
png(file="pieplot.png")
pie(salary,labels = names(salary),main="Salary of Programming",radius = -
1,col.main="darkgreen")
dev.off()
```



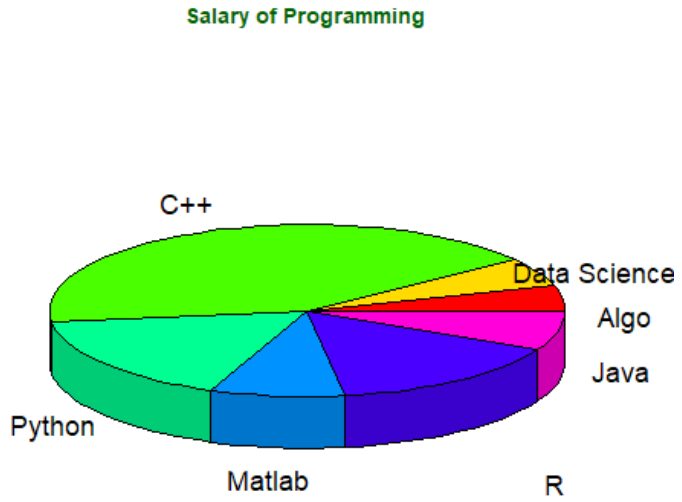
➢ ***Circular chart divided into different segments according to the ratio***

✓ Syntax

```
pie3D(x,edges=NA,radius=1,height=0.1,theta=pi/6,start=0,border=par("fg"),
  col=NULL,labels=NULL,labelpos=NULL,labelcol=par("fg"),labelcex=1.5,
  sector.order=NULL,explode=0,shade=0.8,mar=c(4,4,4,4),pty="s",...)
```

Pie3D Plot/Chart-Example

```
install.packages("plotrix")
require(plotrix)
salary=c(28,32,250,100,50,90,45)
names(salary)=c("Algo","Data Science","C++","Python","Matlab","R","Java")
png(file="pieplot3D.png")
pie3D(salary,labels = names(salary),main="Salary of
Programming",col.main="darkgreen")
dev.off()
```

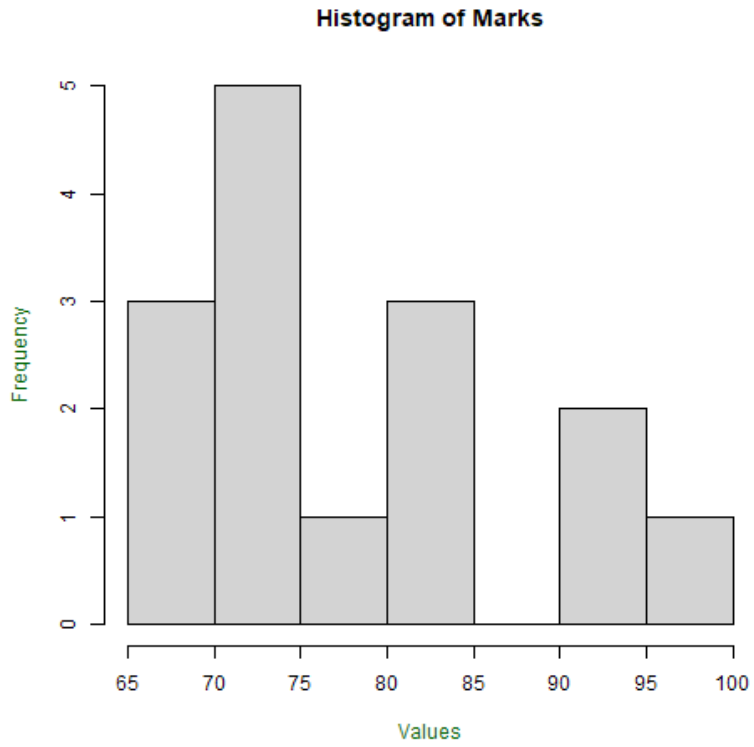


✓ Syntax

```
hist(x, breaks = "Sturges",
     freq = NULL, probability = !freq,
     include.lowest = TRUE, right = TRUE,
     density = NULL, angle = 45, col = NULL, border = NULL,
     main = paste("Histogram of" , xname),
     xlim = range(breaks), ylim = NULL,
     xlab = xname, ylab,
     axes = TRUE, plot = TRUE, labels = FALSE,
     nclass = NULL, warn.unused = TRUE, …)
```

Histogram-Example

```
marks=c(85,75,95,100,65,80,75,70,65,85,75,95,75,85,75)
png(file="histo.png")
hist(marks,main="Histogram of Marks",xlab="Values",col.lab="darkgreen")
dev.off()
```



➢ ***A matrix of scatterplots***

✓ Default Syntax

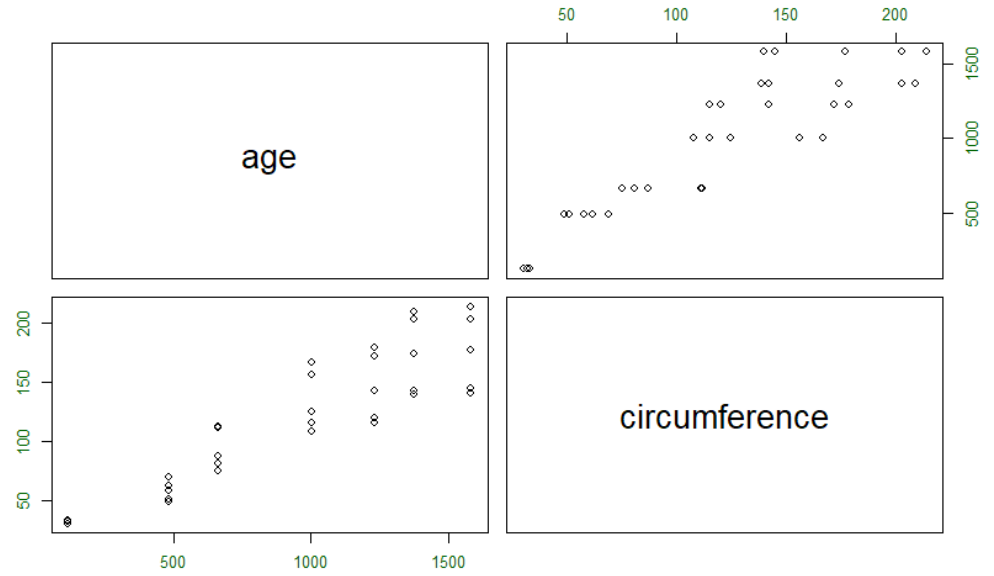
```
pairs(x, labels, panel = points, …,
      horInd = 1:nc, verInd = 1:nc,
      lower.panel = panel, upper.panel = panel,
      diag.panel = NULL, text.panel = textPanel,
      label.pos = 0.5 + has.diag/3, line.main = 3,
      cex.labels = NULL, font.labels = 1,
      row1attop = TRUE, gap = 1, log = "",
      horOdd = !row1attop, verOdd = !row1attop)
```

- ***Values in data vector as height of the bars***

✓ For Formula

```
pairs(formula, data = NULL, …, subset,
      na.action = stats::na.pass)

pairs(~age+circumference,data=Orange,col.axis="darkgreen")
```



➢ *Explore Other plots*

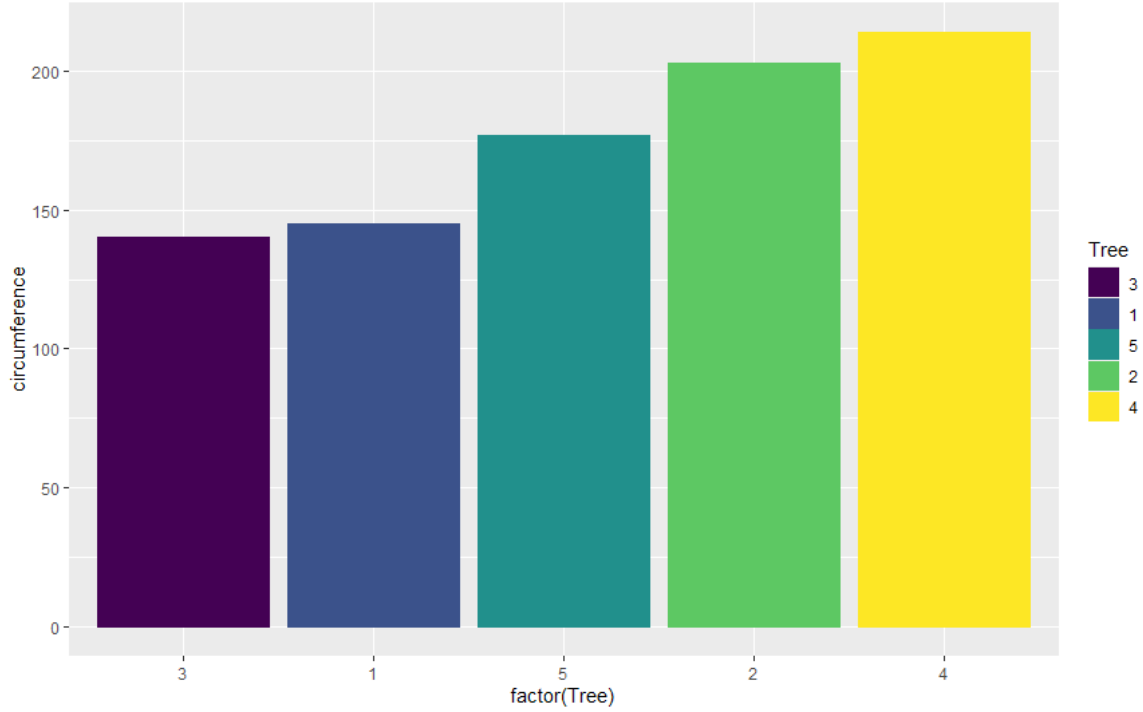
```
assocplot
abline
boxplot
curve
cdplot
coplot
contour
matplot
polygon
spineplot
stars
stem
```

```
sunflowerplot
```

https://www.rdocumentation.org/packages/graphics/versions/3.6.2

```
library(ggplot2)
ggplot(Orange,aes(x=factor(Tree),y=circumference,fill=Tree,color=Tree))
+geom_bar(stat='identity',position='dodge')
```

- ***Ggplot2 is a system for declaratively creating graphics***
- ***Map variables to aesthetics***
- https://ggplot2.tidyverse.org/



Built-in R data

- ***R has a collection of dataset***
- ***In fact, it has tons of built-in datasets that can be used for demo or benchmark purposes***
- ***There are few widely used datasets***
- ***For more details, please visit: https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/00Index.html***

Mostly used datasets

Dataset	Description
datasets	Base R datasets
airquality	New York Air Quality Measurements
ToothGrowth	Experiment studying the effect of Vitamin C
PlantGrowth	Yields obtained under a control of two different treatment conditions
USArrests	Statistics about violent crime rates by the US state
AirPassengers	Monthly Airline Passenger Numbers 1949-1960
mtcars	Motor Trend Car Road Tests
Iris	Edgara Anderson's Iris Data
Orange	Growth of Orange Trees
Quakes	Locations of Earthquakes in Fuji
JohnsonJohnson	Quarterly Earning of Johnson & Johnson Share
women	Average Heights and Weights for American Women
uspop	Populations recorded by the US Census

- ***Data extracted from 1974 Motor Trend US Magazine***
- ***32 observations, 11 columns***

mpg	Miles/(US) gallon
cyl	Number of cylinders
disp	Displacement (cu.in.)
hp	Gross horsepower
drat	Rear axle ratio
wt	Weight (1000 lbs)
qsec	1/4 mile time
vs	Engine (0 = V-shaped, 1 = straight)
am	Transmission (0 = automatic, 1 = manual)
gear	Number of forward gears
carb	Number of carburetors

- ***Classic Box & Jenkins airline data***
- ***Monthly totals of international airline passengers***
- ***1949-1960***

- ***Daily airquality measurements of New York, May – Sep 1973***

Ozone	Ozone (ppb)
Solar.R	Solar R (lang)
Wind	Wind (mph)
Temp	Temperature (degrees F)
Month	Month (1--12)
Day	Day of month (1--31)

- Famous (Fisher's or Anderson's) iris data
- Measurements in centimeters of the variables sepal length and width
- Petal length and width
- 50 flowers from each of 3 species of iris

- Growth of orange trees
- 35 rows and 3 columns
- Tree, age, circumference

```
summary(mtcars[1])
      mpg
 Min.   :10.40
 1st Qu.:15.43
 Median :19.20
 Mean   :20.09
 3rd Qu.:22.80
 Max.   :33.90
```

```
summary(airquality)
     Ozone           Solar.R          Wind             Temp           Month            Day
 Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00   Min.   :5.000   Min.   : 1.0
 1st Qu.: 18.00   1st Qu.:115.8   1st Qu.: 7.400   1st Qu.:72.00   1st Qu.:6.000   1st Qu.: 8.0
 Median : 31.50   Median :205.0   Median : 9.700   Median :79.00   Median :7.000   Median :16.0
 Mean   : 42.13   Mean   :185.9   Mean   : 9.958   Mean   :77.88   Mean   :6.993   Mean   :15.8
 3rd Qu.: 63.25   3rd Qu.:258.8   3rd Qu.:11.500   3rd Qu.:85.00   3rd Qu.:8.000   3rd Qu.:23.0
 Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00   Max.   :9.000   Max.   :31.0
 NA's   :37       NA's   :7
```

```
> print(AirPassengers)
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949 112 118 132 129 121 135 148 148 136 119 104 118
1950 115 126 141 135 125 149 170 170 158 133 114 140
1951 145 150 178 163 172 178 199 199 184 162 146 166
1952 171 180 193 181 183 218 230 242 209 191 172 194
1953 196 196 236 235 229 243 264 272 237 211 180 201
1954 204 188 235 227 234 264 302 293 259 229 203 229
1955 242 233 267 269 270 315 364 347 312 274 237 278
1956 284 277 317 313 318 374 413 405 355 306 271 306
1957 315 301 356 348 355 422 465 467 404 347 305 336
1958 340 318 362 348 363 435 491 505 404 359 310 337
1959 360 342 406 396 420 472 548 559 463 407 362 405
1960 417 391 419 461 472 535 622 606 508 461 390 432
```

READ CSV

- ***It is part of utils packages***

```
read.csv(file, header = TRUE, sep = ",", quote = "\"",
         dec = ".", fill = TRUE, comment.char = "", …)
```

```
read.csv(tf, fill = TRUE) # 1 column
ncol <- max(count.fields(tf, sep = ","))
read.csv(tf, fill = TRUE, header = FALSE,
         col.names = paste0("V", seq_len(ncol)))
```

Read csv

```
data=read.csv("input.csv")
print(data)


print(is.data.frame(data))
print(ncol(data))
print(nrow(data))
```

Read Table

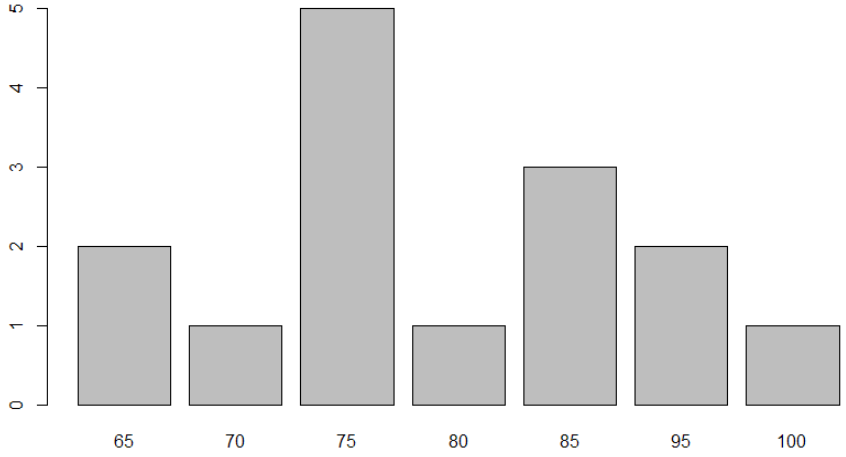
```
read.table(file, header = FALSE, sep = "", quote = "\"'",
           dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),
           row.names, col.names, as.is = !stringsAsFactors,
           na.strings = "NA", colClasses = NA, nrows = -1,
           skip = 0, check.names = TRUE, fill = !blank.lines.skip,
           strip.white = FALSE, blank.lines.skip = TRUE,
           comment.char = "#",
           allowEscapes = FALSE, flush = FALSE,
           stringsAsFactors = default.stringsAsFactors(),
           fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)
```

FREQUENCY TABLE

- ***A list of objects with the frequency of each***
- ***Useful for categorical data***

✓ **Default Syntax**

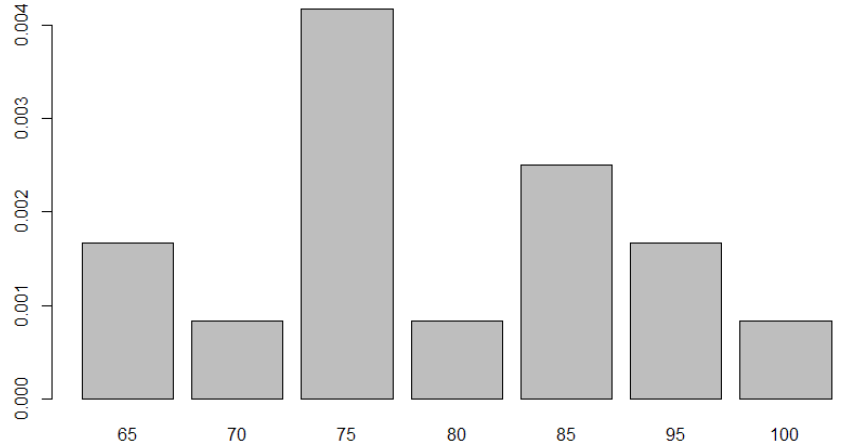
```
table(data)
```




```
marks=c(85,75,95,100,65,80,75,70,65,85,75,95,75,85,75
freq=table(marks)
print(freq)
marks
 65  70  75  80  85  95 100
  2   1   5   1   3   2   1
barplot(freq)
```

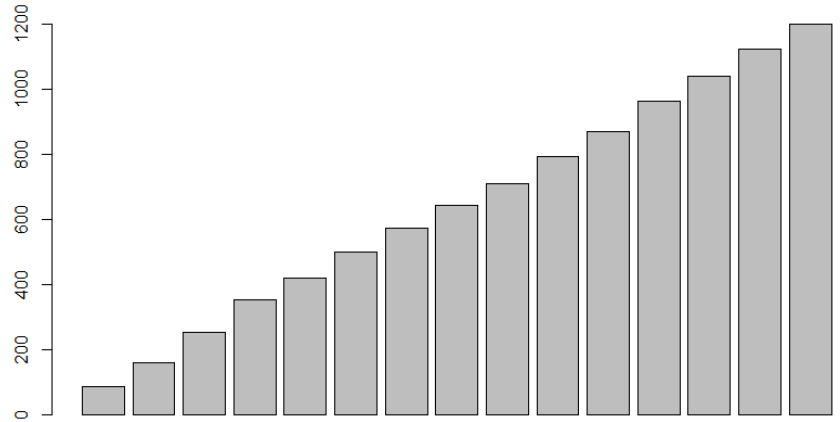
Frequency table with proportions

```
marks=c(85,75,95,100,65,80,75,70,65,85,75,95,75,85,75)
freq=table(marks)/sum(marks)
print(freq)
marks
          65           70           75           80           85
95          100
0.0016666667 0.0008333333 0.0041666667 0.0008333333 0.0025000000
0.0016666667 0.0008333333
barplot(freq)
```



Cumulative Frequency table

```
marks=c(85,75,95,100,65,80,75,70,65,85,75,95,75,85,75)
freq=cumsum(marks)
print(freq)
 [1]   85  160  255  355  420  500  575  645  710  795  870  965
1040 1125 1200
barplot(freq)
```



Two-way Frequency Table

```
marks=c(85,75,95,100,65,80,75,70,65,85,75,95,75,85,75)
passfaile=c(1,1,1,1,0,1,1,1,0,1,1,1,1,1,1)
table(marks,passfaile)
     passfaile
marks 0 1
  65  2 0
  70  0 1
  75  0 5
  80  0 1
  85  0 3
  95  0 2
  100 0 1
```

```
library(dplyr)
marks=c(85,75,95,100,65,80,75,70,65,85,75,95,75,85,75)
pf=c(1,1,1,1,0,1,1,1,0,1,1,1,1,1,1)
data=data.frame(Marks=marks,pf=pf)
tot=data %>% group_by(Marks) %>% summarize(tot=sum(Marks))
print(tot)
# A tibble: 7 × 2
  Marks   tot
  <dbl> <dbl>
1    65   130
2    70    70
3    75   375
4    80    80
5    85   255
6    95   190
7   100   100
```

```
read.table(header = TRUE, text = "
a b
1 2
3 4
")
# }
```

Write CSV

- ***It is part of utils packages***

```
write.csv(data,path)
x=1:5
name=c("MA23M001","MA23M002","MA23M003","MA23M004","MA23M005")
marks=c(100,98,77,89,67)
DF=data.frame(x,name,marks)
write.csv(DF,"students.csv")
```

READ/WRITE XLSX

- ***It is part of xlsx packages***

```
install.packages("xlsx")

require(xlsx)

readxlsx=read.xlsx(path,sheeIndex=1)

writexlsx=write.xlsx(data,path,col.names=TRUE,row.name=TRUE,s
heetName="Sheet2",append=TRUE)
```

- ***It is part of utils packages***

```
x=1:5
name=c("MA23M001","MA23M002","MA23M003","MA23M004","MA23M005")
marks=c(100,98,77,89,67)
DF=data.frame(x,name,marks)
write.xlsx(DF, file = "employee.xlsx", col.names=TRUE,
row.names=TRUE,sheetName="Sheet2",append = TRUE)
read.xlsx("employee.xlsx",sheetIndex = 1)
```

READ/WRITE JSON

- ***It is part of rjson packages***

```
install.packages("rjson")
require(rjson)
readjson=fromJSON(path,sheeIndex=1)
jsondata=toJSON(data)
write(jsondata,path)
```

- ***It is part of rjson packages***

```
list1 <- vector(mode="list", length=2)
list1[[1]] <- c("sunflower", "guava", "hibiscus")
list1[[2]] <- c("flower", "fruit", "flower")

# creating the data for JSON file
jsonData <- toJSON(list1)

# writing into JSON file
write(jsonData, "result.json")

# Give the created file name to the function
result <- fromJSON(file = "result.json")

# Print the result
print(result)
```