راهنمای استفاده از نرم افزار R

گرد آوری : مصطفی کریمی

استاد راهنما : جناب آقای دکتر داوود قزوینی نژاد

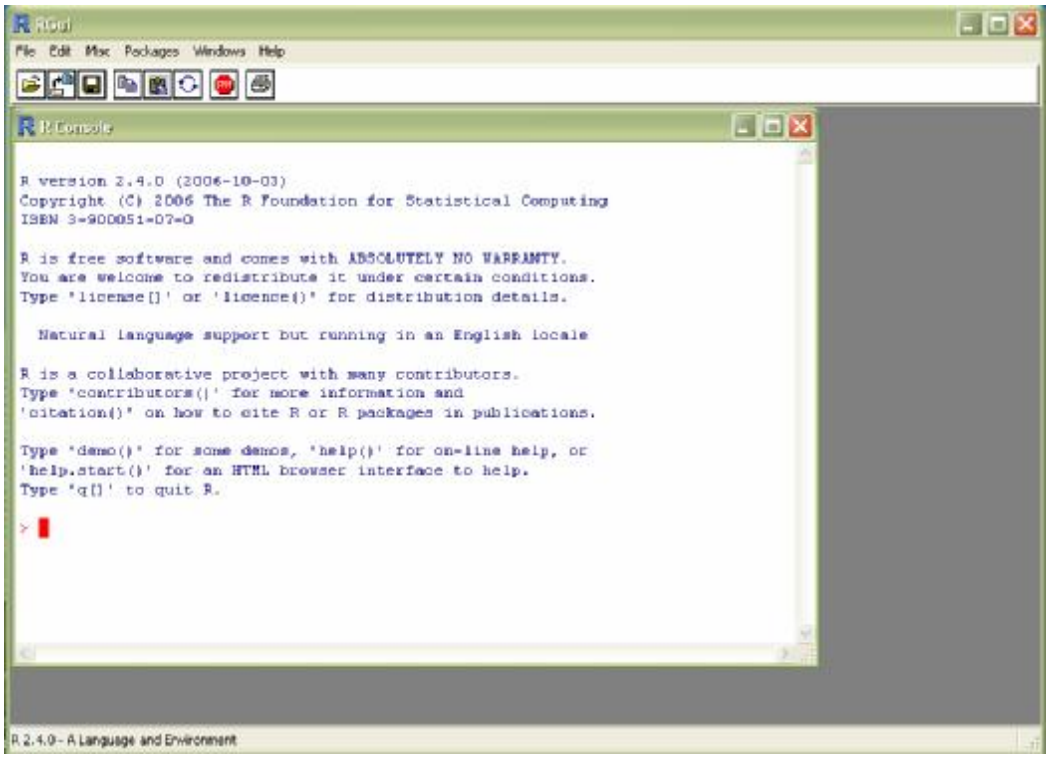
راهنمای استفاده از نرم افزار R

فهرست

کار با نرم افزار R

1. آشنایی و شروع کار

نرم افزار R دارای محیطی شبیه نرم افزار S-Plus بوده و برای انجام هر فعالیت در این نرم افزار باید از دستورها و کدهای مناسب، که اغلب شبیه دستورهای مورد استفاده در نرم افزار S-Plus هستند استفاده کرد. با اجرای نرم افزار R و باز شدن محیط کاری، ابتدا توضیحاتی در مورد نرم افزار ظاهر شده و پس از آن علامت " > " به نمایش در خواهد آمد که نشان دهنده نقطه شروع نوشتن دستورات است. در نرم افزار R همواره دستورات بعد از علامت " > " نوشته می شوند. با فشردن کلید Enter روی صفحه کلید در صورتی که دستور یا کد نوشته شده از نوع دستورات اجرایی باشد اجرا خواهد شد، و در غیر این صورت دستورات به خط بعدی منتقل خواهند شد. برای خروج از نرم افزار از دستور `q()` استفاده می کنیم. در شکل زیر محیط نرم افزار R نشان داده شده است.



برای نوشتن کدها و دستورات در R توجه به چند نکته ضروری به نظر می رسد. نرم افزار R نسبت بـه بزرگـی و کوچکی حروف حساس بوده و حروف بزرگ و کوچک را به صورت متمایز در نظر می گیرد. مثلا A و a در ایـن نرم افزار به دو شئ مختلف اشاره می کنند. برای جدا کردن دستورات می توان از علامت " ; " استفاده کرده و یـا دستور را به خط بعد منتقل کرد. همچنین برای قراردادن گروهی از دستورها در کنار هم از علامـت " { " و " } " استفاده می شود. برای وارد کردن توضیحات در میان دستورها نیز می توان از علامت # استفاده کرد. در صورتی که بخواهیم کدهای قبل را بازخوانی یا ویرایش کنیم می توانیم از کلیدهای مکان نمای بالا و پایین صفحه کلید کمک بگیریم. همچنین در میان خطوط برنامه نیز می توان با استفاده از کلیدهای مکان نمای چپ و راست جابجا شـده و به ویرایش کدها پرداخت. برای پاک کردن صفحه می توان از کلیدهای ترکیبی Ctrl+L استفاده کرد. برای کـسب اطلاعات در مورد توابع و استفاده از فایل help می توان نام تابع را بعد از یک علامت سوال نوشته و به عنوان یک دسـتور مـورد اسـتفاده قـرار داد. همچنـین بـرای جـستجوی یـک کلمـه در فایـل help مـی تـوان از تـابع help.search() استفاده کرد.

2. کار با اشیاء

انواع مختلف اشیاء در R عبارتند از بردارها، آرایه ها و ماتریس ها، فاکتورها، لیست ها، دیتا فریم ها، و توابع. در R یک متغیر تک عضوی به عنوان یک بردار تک عضوی در نظر گرفته می شود. برای مشاهده سـاختار یـک شـئ از تابع str() به همراه نام آن شئ استفاده می کنیم. برای اختصاص یک مقدار به یک شئ از عملگر تخـصیص بـه شکل " <- " یا " -> " و یا علامت " = " استفاده می کنیم. برای نمایش محتویات یک شئ نیز کافیـست نـام آن شئ را به عنوان یک دستور وارد کنیم.

مثال :

```
> Value <- 4
> Value
[1] 4
```

برای مشاهده تمام اشیاء موجود در محیط از تابع objects() یا معادل آن ls() استفاده می کنیم.

مثال :

```
> objects()
[1] "a"     "ab"    "abc"   "Value"
```

برای حذف یک شئ از تابع remove() یا معادل آن rm() با ذکر نام اشـیائی کـه بایـد حـذف شـوند اسـتفاده می کنیم.

مثال :

```
> remove(a,ab,Value)
```

نرم افزار R فاصله های اضافی بین کدها را نادیده می گیرد، بنابراین قرار دادن فاصله زیاد بین کدها مشکلی در اجرای آنها ایجاد نمی کند.

به اشیاء می توان داده هایی را از نوع عددی، کاراکتری، منطقی، یا مختلط تخصیص داد.

مثال :

```
> value.numeric <- 56
> value.character <- " Hello Dear "
> value.logical <- 2<4
> value.comlpex <- 2+3i
```

برای مشاهده نوع یک شئ از تابع `mode()` و برای مشاهده طول یک شئ از تابع `length()` استفاده می کنیم.

مثال :

```
> mode(value.logical)
[1] "logical"
> length(value.logical)
[1] 1

> mode(sin)
[1] "function"
> length(sin)
[1] 1
```

2.1. بردارها

برای ساخت یک بردار در R از تابع `c()` استفاده می کنیم. همچنین برای تخصیص یک بردار به یک شئ می توان از عملگر تخصیص یا تابع `assign()` استفاده کرد.

مثال : هر سه عبارت زیر معادل یکدیگرند.

```
> x <- c(10,9,12,15)
> c(10,9,12,15) -> x
> assign("x",c(10,9,12,15))
```

برای ساخت یک بردار می توان از بردارهای دیگر نیز استفاده کرد.

مثال :

```
> y <- c(x,14,x,8)
```

2.1.1. تولید بردار اعداد متوالی

برای تولید یک بردار از اعداد متوالی بین دوعدد می توان از عملگر " : " استفاده کرد. برای تولید اعداد به این شیوه می توان از عملگرهای ریاضی نیز استفاده کرد.

مثال :

```
> n <- 1:10
> n
 [1]  1  2  3  4  5  6  7  8  9 10
> m <- c(1:10)
> m
 [1]  1  2  3  4  5  6  7  8  9 10
> k <- 10:1
> k
 [1] 10  9  8  7  6  5  4  3  2  1
> l <- 2*(1:5)
> l
[1]  2  4  6  8 10
> p <- 3+5:10
> p
[1]  8  9 10 11 12 13
```

رایج ترین تابعی که برای تولید اعداد متوالی مورد استفاده قرار می گیرد تابع `seq()` است. در این تابع می تـوان با مشخص کردن نقاط شروع و پایان، مشخص کردن گام افزایشی یا کاهشی، و یا مشخص کردن طول بردار مـورد نظر برداری از اعداد متوالی تولید کرد.

مثال :

```
> seq(1,10)
 [1]  1  2  3  4  5  6  7  8  9 10
> seq(1,10,by=2)
[1] 1 3 5 7 9
> seq(1,by=2,length=10)
 [1]  1  3  5  7  9 11 13 15 17 19
```

2.1.2. تولید بردار اعداد تکراری

برای تکرار اعضای یک بردار از تابع `rep()` استفاده می شود. در این تابع ابتدا باید اعدادی را کـه مـی خـواهیم تکرار شوند وارد کنیم، سپس بوسیله گزینه های `each=` و `times=` به ترتیب تعداد دفعاتی را که می خواهیم هر عدد تکرار شود یا کل اعداد تکرار شوند را مشخص می کنیم.

مثال :

```
> x <- 1:5
> rep(x,times=5)
 [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
> rep(x, each=5)
 [1] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5
> rep(1:4,each=2,times=2)
 [1] 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4
```

2.1.3. تولید بردار اعداد تصادفی

برای تولید یک نمونه تصادفی از اعداد دلخواه از تابع `sample()` استفاده می کنیم. ساختار کلی این تابع به شکل زیر است.

```
sample(x,size,replace,prob)
```

در این تابع `x` برداری است که می خواهیم نمونه مورد نظر از آن گرفته شود. در قسمت `size` باید حجم نمونه را مشخص کنیم. در قسمت `replace` مشخص می کنیم که آیا نمونه گیری با جایگذاری باشد یا بدون جایگذاری. در صورتی که `replace` را مساوی `TRUE` قرار دهیم نمونه گیری با جایگذاری انجام می شود و در صورتی که آنرا مساوی `FALSE` قرار دهیم نمونه گیری بدون جایگذاری انجام خواهد شد. در قسمت `prob` نیز می توان برداری شامل وزن ها وارد کرد تا نمونه گیری به صورت وزنی انجام شود.

مثال :

```
> sample(1:20, 15, rep=T)
 [1]  7 16  6 20  9 14  9  1  3  3  5  2 16  7 15
```

2.1.4. ایندکس کردن یک بردار

ایندکس کردن یک شئ عبارت است از فراخوانی اعضای آن شئ. برای ایندکس کردن یک بردار از علامت براکت استفاده می کنیم.

مثال :

```
> x <- c(5,10,8,12,6,14)
> x[3]
[1] 8
> x[2:6]
[1] 10  8 12  6 14
```

2.1.5. استفاده از عملگرها و توابع

برای انجام عملیات ریاضی روی بردارها می توان از عملگرهای ریاضی `+` ، `-` ، `*` ، `/` ، و `^` (توان)، `%%` (باقیمانده)، و `%/%` (خارج قسمت)، و توابع ریاضی مانند `log()`، `exp()`، `sin()`، `cos()`، `tan()`، `sqrt()` (ریشه دوم)، و غیره استفاده کرد. هریک از عملگرها و توابع فوق به صورت عضو به عضو روی عناصر بردار عمل می کنند. مثلا در دستور `y+1` هر عضو بردار `y` با عدد یک جمع می شود. با توجه به این خاصیت برای اعمال عملگرها روی بردارها نیازی نیست که بردارها دارای طول یکسان باشند. همواره طول همه بردارهای موجود در یک عبارت ریاضی به اندازه طول بزرگترین بردار موجود در عبارت در خواهند آمد. در مورد بردارها می توان از برخی توابع آماری نیز استفاده کرد. توابع `min()` و `max()` به ترتیب کوچکترین عضو و

بزرگترین عضو یک بردار را ارائه می کنند. تابع range() برداری با دو عضو ارائه می کند که آن دو عضو به ترتیب کوچکترین و بزرگترین عضو بردارمورد نظر هستند. تابع length() طول یک بردار، تابع sum() حاصل جمع اعضای یک بردار، و تابع prod() حاصل ضرب اعضای یک بردار را ارائه می کند. همچنین توابع mean() و var() و sd() به ترتیب میانگین و واریانس و انحراف استاندارد اعضای یک بردار را ارائه می کنند. توابع cor() و cov() به ترتیب برای محاسبه همبستگی و کواریانس میان اعضای یک بردار مورد استفاده قرار می گیرند. از توابع summary() و quantile() نیز برای محاسبه چندک ها استفاده می شود. تابع sort() اعضای یک بردار را به صورت صعودی مرتب می کند. از تابع round() نیز برای گرد کردن اعداد تا رقم اعشار دلخواه استفاده می شود.

مثال :

```
> x <- c(15,14,19,18,20,15,16,17,18,14)
> sum(x);prod(x)
[1] 166
[1] 1.476849e+12
> mean(x);var(x);sd(x)
[1] 16.6
[1] 4.488889
[1] 2.1187
> sort(x)
 [1] 14 14 15 15 16 17 18 18 19 20
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   14.0    15.0    16.5    16.6    18.0    20.0
> quantile(x)
  0%  25%  50%  75% 100%
14.0 15.0 16.5 18.0 20.0
> x <- x+0.375*x
> x
 [1] 20.625 19.250 26.125 24.750 27.500 20.625 22.000 23.375 24.750
19.250
> round(x,2)
 [1] 20.62 19.25 26.12 24.75 27.50 20.62 22.00 23.38 24.75 19.25
```

2.1.6. داده های گمشده

برای معرفی داده های گمشده در یک بردار از عبارت NA استفاده می کنیم. برای بررسی داده های گمشده در یک بردار از تابع is.na() استفاده می شود. این تابع برداری منطقی به طول بردار مورد نظر ارائه می دهد که مقادیر گمشده متناظر با اعضای بردار در آن با TRUE نشان داده شده اند.

مثال :

```
> x <- c(2,5,NA,3,NA)
> is.na(x)
[1] FALSE FALSE  TRUE FALSE  TRUE
```

2.1.7. بردار کاراکترها

بردار کاراکترها نیز همانند بردار اعداد با استفاده از تابع c() ساخته می شود. برای ایجاد یک بـردار از کاراکترهـا باید آنها را درون علامت های " یا ' قرار داد. در یک بردار از کاراکترها می توان از \n برای رفتن به خط جدید، از \t برا ی جهش، و از \b برای بازگشت استفاده کرد. همچنین با استفاده از تابع paste() می توان دنباله ای از اعداد را به کاراکترها تخصیص داد. در تابع paste() با استفاده از گزینه sep= مشخص می کنیم که کاراکترها و اعداد به وسیله چه کاراکتری از هم جدا شوند.

مثال:

```
> paste(c("X"),1:10,sep="")
 [1] "X1"  "X2"  "X3"  "X4"  "X5"  "X6"  "X7"  "X8"  "X9"  "X10"
> paste(c("X","Y"),rep(1:5,each=2),sep="")
 [1] "X1" "Y1" "X2" "Y2" "X3" "Y3" "X4" "Y4" "X5" "Y5"
```

برای تولید بردارهایی از حروف می توان از بردارهای letters و LETTERS استفاده کرد. دو بردار letters و LETTERS به ترتیب شامل 26 حرف کوچک و بزرگ هستند که با ایندکس کردن این بردارها می توان به ایـن حروف دست یافت.

مثال:

```
> letters[1:10]
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
> LETTERS[10:20]
 [1] "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S" "T"
```

2.2. فاکتورها

فاکتورها از لحاظ ساختاری بسیار شبیه بردارها هستند. از فاکتورها برای گروه بندی و ایجاد تمـایز بـین داده هـای بردارها استفاده می شود. فاکتورها را می توان به طور مستقیم تولید کرده و یا یک بردار را به یک فاکتور تبدیل کرد. برای تبدیل یک بردار به یک فاکتور از تابع factor() استفاده می شود.

مثال : در مورد گروهی از افراد جنسیت و سن را در دو بردار sex و age ثبت کرده ایم. می خواهیم از بردار sex به عنوان فاکتوری برای دسته بندی داده های بردار age استفاده کنیم.

```
> sex <- c( "f","f","f","m","f","m","f","m","f","m")
> age <- c(25,30,23,25,28,26,29,27,24,28)
> sex.f <- factor(sex)
> sex.f
 [1] f f f m f m f m f m
Levels: f m
```

2.2.1. استفاده از یک تابع بر حسب یک فاکتور

با استفاده ازتابع tapply() می توان یک تابع را بر حسب یک فاکتور روی یک بردار اعمال کرد. در تابع tapply() باید بردار داده ها، فاکتور، و تابع مورد نظر را وارد کنیم.

مثال : در مورد داده های سن در مثال قبل می خواهیم میانگین سن را برای هر جنس محاسبه کنیم.

```
> tapply(age,sex.f,mean)
   f    m
26.5 26.5
```

2.2.2.تولید فاکتورهای دارای الگو

برای تولید یک فاکتور با الگوی مشخص از تابع gl() استفاده می شود. ساختار کلی این تابع به شکل زیر است.

```
gl(n,k,length,labels,ordered )
```

در این تابع n مشخص کننده تعداد سطوح، k مشخص کننده تعداد تکرارها، length مشخص کننده طول فاکتور، و labels مشخص کننده برچسب سطوح می باشد. هچنین ordered مشخص می کند که سطوح مرتب شده باشند یا نه.

مثال :

```
> gl(5,2)
 [1] 1 1 2 2 3 3 4 4 5 5
Levels: 1 2 3 4 5
> gl(5,1,10)
 [1] 1 2 3 4 5 1 2 3 4 5
Levels: 1 2 3 4 5
> gl(5,1,10,labels=LETTERS[1:5])
 [1] A B C D E A B C D E
Levels: A B C D E
```

2.3. آرایه ها و ماتریس ها

آرایه ها یکی از اشیاء موجود در R برای کار با داده ها هستند که ما را قادر می سازند تا داده ها را به صورت چند بعدی نگه داری کرده و با آنها کار کنیم. در خالت خاص بردارها آرایه های یک بعدی و ماتریس ها آرایه های دو بعدی هستند. ساده ترین راه ساختن یک آرایه استفاده از تابع dim() و تبدیل یک بردار به یک آرایه است. برای تبدیل یک بردار از داده ها به یک آرایه کافیست یک بردار بعد را شامل اندازه طول های مورد نظر برای هر بعد آرایه به تابع dim() تخصیص دهیم.

مثال :

```
> x <- 1:18
> dim(x) <- c(2,3,3)
> x
, , 1

     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

, , 2

     [,1] [,2] [,3]
[1,]    7    9   11
[2,]    8   10   12

, , 3

     [,1] [,2] [,3]
[1,]   13   15   17
[2,]   14   16   18
> dim(x) <- c(3,6)
> x
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    4    7   10   13   16
[2,]    2    5    8   11   14   17
[3,]    3    6    9   12   15   18
```

یک راه دیگر ساختن آرایه ها و ماتریس ها استفاده از تابع `array()` است. برای ساختن یک آرایه با این تابع باید بردار داده ها و بردار بعد را در تابع مشخص کنیم.

مثال :

```
> x <- 1:20
> y <- array(x,c(5,4))
> y
     [,1] [,2] [,3] [,4]
[1,]    1    6   11   16
[2,]    2    7   12   17
[3,]    3    8   13   18
[4,]    4    9   14   19
[5,]    5   10   15   20
> z <- array(0,c(2,3))
> z
     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0
```

2.3.1. ایندکس کردن یک آرایه

برای ایندکس کردن یک آرایه از علامت براکت استفاده می کنیم.

مثال :

```
> x <- 1:20
> x[15]
[1] 15
> dim(x) <- c(4,5)
> x
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
> x[3,4]
[1] 15
```

2.3.2. ساختن ماتریس

برای ساخت یک ماتریس علاوه بر تابع `dim()` می توان از تابع `matrix()` نیز استفاده کرد. در تابع `matrix()` باید بردار داده ها، تعداد سطر، و تعداد ستون را وارد کنیم. در صورتی که بخواهیم ماتریس داده های بردار را به صورت سطری مرتب کند قسمت `byrow=` را مساوی `TRUE` قرار می دهیم.

مثال :

```
> x <- 1:20
> matrix(x,4,5)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
> matrix(x,4,5,byrow=T)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
[3,]   11   12   13   14   15
[4,]   16   17   18   19   20
```

2.3.3. کار با ماتریس ها

برای کار با ماتریس ها در R توابع و قابلیت هایی وجود دارد که به برخی از آنها اشاره می کنیم.

برای ترانهاده کردن یک ماتریس از تابع `t()` ، برای نمایش تعداد سطرهای یک ماتریس از تابع `nrow()` ، و برای نمایش تعداد ستون های یک ماتریس از تابع `ncol()` استفاده می شود.

مثال :

```
> y
     [,1] [,2] [,3] [,4]
[1,]    1    6   11   16
[2,]    2    7   12   17
[3,]    3    8   13   18
[4,]    4    9   14   19
[5,]    5   10   15   20
> nrow(y)
[1] 5
> ncol(y)
[1] 4
> t(y)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    6    7    8    9   10
[3,]   11   12   13   14   15
[4,]   16   17   18   19   20
```

برای ضرب اعضای متناظر دو ماتریس در هم در ماتریس های هم مرتبه از " * "، و برای ضرب ماتریسی دو ماتریس در هم از " %*% " استفاده می شود.

مثال :

```
> m
     [,1] [,2] [,3] [,4] [,5]
[1,]    2   12   22   32   42
[2,]    4   14   24   34   44
[3,]    6   16   26   36   46
[4,]    8   18   28   38   48
[5,]   10   20   30   40   50
> n
     [,1] [,2] [,3] [,4] [,5]
[1,]    3   18   33   48   63
[2,]    6   21   36   51   66
[3,]    9   24   39   54   69
[4,]   12   27   42   57   72
[5,]   15   30   45   60   75
> m*n
     [,1] [,2] [,3] [,4] [,5]
[1,]    6  216  726 1536 2646
[2,]   24  294  864 1734 2904
[3,]   54  384 1014 1944 3174
[4,]   96  486 1176 2166 3456
[5,]  150  600 1350 2400 3750
> m%*%n
     [,1] [,2] [,3] [,4]  [,5]
[1,] 1290 2940 4590 6240  7890
[2,] 1380 3180 4980 6780  8580
[3,] 1470 3420 5370 7320  9270
[4,] 1560 3660 5760 7860  9960
[5,] 1650 3900 6150 8400 10650
```

برای محاسبه معکوس یک ماتریس و یا حل یک معادله ماتریسی از تابع `solve()` استفاده می شود. اگر در این تابع فقط یک ماتریس را وارد کنیم معکوس آن ماتریس محاسبه خواهد شد، و اگر علاوه بر یک ماتریس ضرایب ماتریس پاسخ را نیز وارد کنیم جواب معادله ماتریسی به شکل b=AX محاسبه خواهد شد.

مثال :

```
> b
     [,1]
[1,]  4.5
[2,]  6.0
[3,]  7.5
[4,]  9.0
> A
     [,1] [,2] [,3] [,4]
[1,]    0    3    2    1
[2,]    1    0    3    2
[3,]    2    1    0    3
[4,]    3    2    1    0
> solve(A)
            [,1]        [,2]        [,3]        [,4]
[1,] -0.20833333  0.04166667  0.04166667  0.29166667
[2,]  0.29166667 -0.20833333  0.04166667  0.04166667
[3,]  0.04166667  0.29166667 -0.20833333  0.04166667
[4,]  0.04166667  0.04166667  0.29166667 -0.20833333
> solve(A,b)
     [,1]
[1,] 2.25
[2,] 0.75
[3,] 0.75
[4,] 0.75
```

برای محاسبه مقادیر ویژه و بردارهای ویژه یک ماتریس از تابع `eigen()` استفاده می کنیم.

مثال :

```
> z
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    3    1    2
[3,]    5    2    1
> eigen(z)
$values
[1]  7.8390863 -0.7106916 -4.1283947

$vectors
           [,1]       [,2]       [,3]
[1,] -0.6465566  0.1916741  0.7383939
[2,] -0.4612978 -0.8691432 -0.1783099
[3,] -0.6075927  0.4559069 -0.6503692
```

در صورتی که بخواهیم با وصل کردن چند بردار یا چند ماتریس به هم یک ماتریس یا بردار جدید بسازیم از توابع `rbind()` و `cbind()` استفاده می کنیم. تابع `rbind()` بردارها یا ماتریس ها را به صورت سطری به هم وصل می کند و تابع `cbind()` این کار را به صورت ستونی انجام می دهد.

مثال :

```
> x
[1]  6  9 12 15
> y
[1]  6.6  9.9 13.2 16.5
> rbind(x,y)
  [,1] [,2] [,3] [,4]
x  6.0  9.0 12.0 15.0
y  6.6  9.9 13.2 16.5
> cbind(x,y)
      x    y
[1,]  6  6.6
[2,]  9  9.9
[3,] 12 13.2
[4,] 15 16.5
```

2.4. لیست ها

یک لیست در R مجموعه ای منظم از اشیاء است که این اشیاء را مولفه های آن لیست می نامیم. ضرورتی ندارد که مولفه های یک لیست همه از یک نوع باشند. بـه عنـوان مثـال یـک لیـست مـی توانـد شـامل بردارهـای عـددی، ماتریس ها، بردارهای منطقی، آرایه ها، بردارهای کاراکتری و غیره باشد. برای ساختن یک لیست از تابع `list()` استفاده می کنیم. در تابع `list()` نام هر مولفه را وارد کرده و بعد از علامـت مـساوی شـئ مـورد نظـر را وارد می کنیم.

مثال :

```
> lst <- list(name="Ferd", wife="Mary", no.childern=3,
child.ages=c(4,7,9))
> lst
$name
[1] "Ferd"

$wife
[1] "Mary"

$no.childern
[1] 3

$child.ages
[1] 4 7 9
```

2.4.1. ایندکس کردن یک لیست

برای ایندکس کردن یک لیست از دو علامت براکت متوالی و برای ایـندکس کـردن یـک مولفـه از لیـست از یـک علامت براکت استفاده می کنیم.

مثال :

```
> lst[[2]]
[1] "Mary"
> lst[["name"]]
[1] "Ferd"
> lst[["child.ages"]][c(1,2)]
[1] 4 7
```

2.5. دیتا فریم ها

دیتا فریم ها از لحاظ ساختار شباهت بسیار زیادی به ماتریس ها دارند. یک دیتا فریم را می توان ماتریسی در نظر گرفت که ستون های آن بردارهایی با ساختارهای مختلف است. یک دیتا فریم در اصل لیستی از بردارهای هم اندازه و با ساختارهای مختلف است. برای ساختن یک دیتا فریم از تابع `data.frame()` استفاده می شود. در این تابع باید نام مورد نظر برای هر ستون را به همراه بردار شامل داده های آن ستون وارد کنیم.

مثال :

```
> name <- c("A","B","C","D")
> age <- c(19,22,18,20)
> sex <- c("M","M","F","M")
> score <- c(18,19,17,20)
> data.frame(Name=name,Age=age,Sex=sex,Score=score)
  Name Age Sex Score
1    A  19   M    18
2    B  22   M    19
3    C  18   F    17
4    D  20   M    20
```

2.5.1. ایندکس کردن یک دیتا فریم

یک دیتا فریم را می توان همانند یک ماتریس یا یک لیست ایندکس کرد. همچنین می توان از علامت "$" برای ایندکس کردن یک دیتا فریم استفاده کرد.

مثال :

```
> df <- data.frame(Name=name,Age=age,Sex=sex,Score=score)
> df[,2]
[1] 19 22 18 20
> df[["Score"]]
[1] 18 19 17 20
> df$Age
[1] 19 22 18 20
```

3. دستکاری داده ها

3.1. نام گذاری داده ها

برای نام گذاری یا مشاهده نام داده های یک شئ از تابع names() استفاده می کنیم. برای نام گـذاری داده هـای یک شئ بردار شامل کراکترها را به تابع names() تخصیص می دهیم و برای مشاهده نام داده های یک شـئ آن شئ را در تابع names() وارد می کنیم.

مثال :

```
> x <- 1:10
> y <- LETTERS[11:20]
> names(x) <- y
> x
 K  L  M  N  O  P  Q  R  S  T
 1  2  3  4  5  6  7  8  9 10
> names(x)
 [1] "K" "L" "M" "N" "O" "P" "Q" "R" "S" "T"
```

3.2. مرتب کردن داده ها

برای مرتب کردن داده ها از تابع sort() استفاده می کنیم. با استفاده از تابع sort() می تـوان داده هـا را بـه صورت افزایشی، کاهشی، و یا جزئی مرتب کرد.

مثال :

```
> x
 [1] 4 5 3 4 2 4 1 1 2 5 3 4 5 2 3
> sort(x)
 [1] 1 1 2 2 2 3 3 3 4 4 4 4 5 5 5
> sort(x, decreasing=T)
 [1] 5 5 5 4 4 4 4 3 3 3 2 2 2 1 1
> sort(x, partial=c(3,4))
 [1] 1 1 2 2 3 3 2 3 4 5 4 4 5 5 4
> x
 [1] 4 5 3 4 2 4 1 1 2 5 3 4 5 2 3
```

3.3. رتبه بندی داده ها

برای رتبه بندی داده ها از تابع rank() استفاده می کنیم.

مثال :

```
> x
 [1] 4 5 3 4 2 4 1 1 2 5 3 4 5 2 3
> rank(x)
 [1] 10.5 14.0  7.0 10.5  4.0 10.5  1.5  1.5  4.0 14.0  7.0 10.5 14.0
4.0  7.0
```

3.4. جدول بندی داده ها

برای جدول بندی داده ها از تابع `table()` استفاده می کنیم.

مثال : بردار sex شامل جنسیت و بردار age شامل سن دانش آموزان دبیرستانی در یک نمونه 50 تایی می باشد. در بردار sex پسر را با 1 و دختر را با 2 نمایش داده ایم.

```
> sex
 [1] 2 2 2 2 1 1 1 2 2 2 1 2 2 1 2 2 1 1 2 2 2 2 1 1 2 2 1 2 2 2 1 1
1 1 2 1 2 1
[39] 1 2 2 2 2 2 1 2 2 1 1 1
> table(sex)
sex
 1  2
21 29
> age
 [1] 17 15 18 15 17 18 17 18 18 16 17 15 16 18 15 16 15 16 15 16 18
18 16 17 16
[26] 18 18 18 18 15 16 18 15 18 18 18 15 15 17 18 18 15 16 17 18 16
17 15 17 17
> table(age)
age
15 16 17 18
12 10 10 18
> table(age,sex)
    sex
age   1  2
  15  4  8
  16  3  7
  17  7  3
  18  7 11
```

3.5. جدا کردن داده ها

برای جدا کردن داده های یک بردار بر حسب یک بردار دیگر از تابع `split()` استفاده می کنیم.

مثال :

```
> split(age,sex)

$`1`
 [1] 17 18 17 17 18 15 16 16 17 18 16 18 15 18 18 15 17 18 15 17 17

$`2`
 [1] 17 15 18 15 18 18 16 15 16 15 16 15 16 18 18 16 18 18 18 15 18
15 18 18 15
[26] 16 17 16 17

> split(sex,age)
$`15`
 [1] 2 2 2 2 1 2 2 1 2 1 2 1

$`16`
 [1] 2 2 2 1 2 1 2 1 2 2

$`17`
 [1] 2 1 1 1 1 1 2 2 1 1

$`18`
 [1] 2 1 2 2 1 2 2 2 1 2 2 1 1 2 1 2 2 1
```

4. توزیع های آماری

برای هر توزیع آماری در R تابعی وجود دارد که می توان با استفاده از آن تابع مقادیر احتمال را برای یک توزیع محاسبه کرده و یا اعداد تصادفی از یک توزیع تولید کرد. در جدول زیر برخی از این توابع معرفی شده اند.

توزیع	تابع	پارامترها
beta	`beta()`	shape1, shape2, ncp
binomial	`binom()`	size, prob
Cauchy	`cauchy()`	location, scale
chi-squared	`chisq()`	df, ncp
exponential	`exp()`	rate
F	`f()`	df1, df2, ncp
gamma	`gamma()`	shape, scale
geometric	`geom()`	prob
hypergeometric	`hyper()`	m, n, k
log-normal	`lnorm()`	meanlog, sdlog
logistic	`logis()`	location, scale
negative binomial	`nbinom()`	size, prob
normal	`norm()`	mean, sd
Poisson	`pois()`	lambda
Student’s t	`t()`	df, ncp
Uniform	`unif()`	min, max
Weibull	`weibull()`	shape, scale
Wilcoxon	`wilcox()`	m, n

برای استفاده از توابع جدول فوق باید بر حسب نوع استفاده از تابع، پیش از نام تابع از حروف p ، q ، d ، یا r استفاده کنیم. از حرف p برای محاسبه احتمال در تابع توزیع، از حرف q برای محاسبه چندک ها، از حرف d برای محاسبه چگالی، و از حرف r برای تولید اعداد تصادفی استفاده می کنیم. برای آشنایی دقیق با نحوه قرار دادن پارامترها و مقادیر در هر تابع می توان نام تابع مورد نظر را بعد از یک علامت سوال به عنوان یک دستور وارد کرد تا صفحه help مربوط به آن تابع باز شود.

مثال :

```
> ?rnorm()
> rnorm(n=10)
 [1] -0.04785585 -0.31453549  1.01482215 -0.33774153 -0.46852454 -
0.42633759
 [7]  0.88107806  0.58547995 -0.51812346  1.05938773
> rnorm(n=10, mean=10, sd=15)
 [1]  12.962637   4.022534  29.767160 -19.041441  42.498506
20.358151
 [7]  13.582569  -2.029278  35.003451  39.089246
> pnorm(q=1.5)
[1] 0.9331928
> qnorm(p=0.05)
[1] -1.644854
> dbinom(x=15, size=20, prob=0.04)
[1] 1.357375e-17
> pbinom(q=5, size=20, prob=0.04)
[1] 0.9999023
```

5. دستورات کنترلی

5.1. دستور شرط

برای اجرای دستورات شرطی در R از دستور `if/else` استفاده می کنیم. ساختار کلی این دستور به شکل زیر است.

```
> if (expr_1) expr_2 else expr_3
```

در این دستور `expr_1` عبارت شرطی است که باید مشخص شود. در صورتی که شرط برقرار باشد دستورات وارد شده در قسمت `expr_2` اجرا خواهند شد و در غیر این صورت دستورات وارد شده پس از `else` در قسمت `expr_3` اجرا می شوند. در قسمت شرط می توان از " && " و " || " به عنوان "و" و "یا" منطقی استفاده کرد. برای گروه بندی دستورات اجرایی از علامت" { "و" } " استفاده می شود. در گروه ها برای جدا کردن دستورات از " ; " استفاده می شود.

مثال :

```
> x <- rnorm(100); m <- mean(x)
> if (m>=0.1||m<=-0.1){y <- "mean is not 0";y} else {y <- "mean is
0";y}
[1] "mean is 0"
```

5.2. دستورات تکرار

در R سه دستور تکرار وجود دارد که عباراتد از for، repeat، و while. ساختار کلی این سه دستور به شکل زیر است.

```
> for (name in expr_1) expr_2

> repeat expr

> while (condition) expr
```

در دستور for در قسمت name نام متغیر شمارنده و در قسمت expr_1 تکرارها را مشخص می کنیم. در قسمت expr_2 نیز دستوراتی را که باید تکرار شوند وارد می کنیم. در دستور repeat در قسمت expr دستوراتی را که باید تکرار شوند وارد می کنیم. برای توقف تکرار دستورات در دستور repeat از کلید Esc استفاده می کنیم. در دستور while در قسمت condition شرط پایان تکرار و در قسمت expr دستوراتی را که باید تکرار شوند وارد می کنیم.

مثال :

```
> for (i in 1:10){dens <- dbinom(i, 10, 0.5); print(dens)}
[1] 0.009765625
[1] 0.04394531
[1] 0.1171875
[1] 0.2050781
[1] 0.2460938
[1] 0.2050781
[1] 0.1171875
[1] 0.04394531
[1] 0.009765625
[1] 0.0009765625
> j <- 0
> while (j<=10){z <- dbinom(j, 10, 0.5);print(z);j=j+1}
[1] 0.0009765625
[1] 0.009765625
[1] 0.04394531
[1] 0.1171875
[1] 0.2050781
[1] 0.2460938
[1] 0.2050781
[1] 0.1171875
[1] 0.04394531
[1] 0.009765625
[1] 0.0009765625
```

از تابع print() برای چاپ خروجی استفاده می شود.

6. نوشتن توابع و عملگرهای شخصی

برای نوشتن توابع شخصی در R از تابع function() استفاده می شود. شکل کلی استفاده از این تابع برای نوشتن توابع شخصی به صورت زیر است.

```
> name <- function(arg_1,arg_2,...) expression
```

در ساختار فوق در قسمت name نام مورد نظر برای تابع را وارد می کنیم. در قسمت arg_1,arg_2,... نیز پارامترهای تابع مورد نظر را وارد می کنیم. در هنگام تعیین پارامترها در صورت تمایل می توان با استفاده از علامت مساوی مقدار پیش فرضی را برای هر پارامتر مشخص کرد. در قسمت expression نیز دستورات بدنه تابع نوشته می شود. در نوشتن دستورات بدنه یک تابع می توان از توابع دیگر نیز استفاده کرده و یا حتی توابع شخصی نوشت. فراخوانی و استفاده از این توابع همانند دیگر توابع R می باشد، البته با این تفاوت که برای تخصیص خروجی یک تابع شخصی به یک شئ از علامت" ->> " به جای " <- " استفاده می شود. با استفاده از شیوه ای مشابه می توان عملگر شخصی نیز نوشت. یک عملگر شخصی نیز با استفاده از تابع function() نوشته می شود. شکل کلی استفاده از تابع function() برای نوشتن عملگر به صورت زیر است.

```
> "%!%" <- function(X,y) { ... }
```

در قسمت " %!% " به جای علامت ! کاراکتر مورد نظر برای نشان دادن عملگر را بین دو علامت % تعیین می کنیم. در قسمت X,y مشخص می کنیم که عملگر بین چه اشیائی عمل کند. در قسمت آخر نیز بدنه عملگر را تعیین می کنیم. عملگر ضرب ماتریس ها، " %*% " ، نمونه ای از عملگرهای شخصی است.

مثال : تابع زیر بردار عددی x را گرفته و برای مقایسه میانه آن با صفر از آزمون علامت استفاده می کند.

```
sign.test <- function (x, mu=0) {
n <- length(x)
y <- sum(x<mu)
p.value <- min(c( pbinom(y,n,.5), pbinom(y,n,.5,lower.tail=F) ))*2
p.value
}
```

گرافیک و رسم نمودار

در R نمودارها و ترسیمات در پنجره ای مجزا به نام پنجره گرافیک نمایش داده می شوند. این پنجره با استفاده از تابع `windows()` باز می شود. البته استفاده از این تابع ضرورتی ندارد زیرا در رسم هر نمودار این پنجره به صورت اتوماتیک باز می شود. برای کسب اطلاعات کامل در مورد تابع `windows()` می توان با استفاده از دستور `?windows()` از فایل help کمک گرفت. در حالت کلی توابع ترسیم نمودار در R را می توان به سه دسته توابع ترسیم سطح بالا، توابع ترسیم سطح پایین، و توابع ترسیم تعاملی تقسیم کرد. با استفاده از توابع ترسیم سطح بالا قادر خواهیم بود نمودارهایی شامل محورها، برچسب ها، عنوان، و مواردی از این قبیل رسم کنیم. توابع ترسیم سطح پایین ما را قادر می سازند که روی نمودارهای موجود تغییراتی اعمال کنیم، و توابع ترسیم تعاملی به ما این امکان را می دهند تا به تبادل اطلاعات با نمودارهای موجود بپردازیم. در ادامه به بررسی انواع توابع و دستورات ترسیم خواهیم پرداخت.

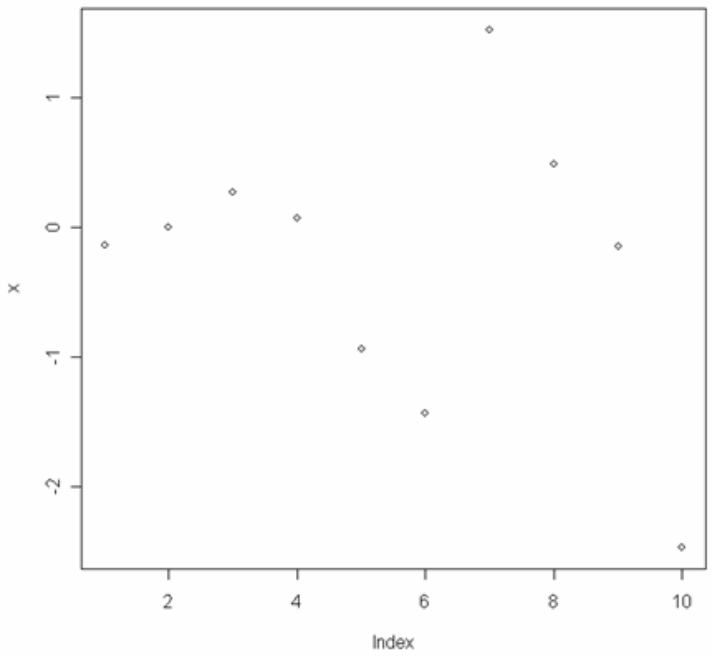
1. دستورهای ترسیم سطح بالا

1.1. تابع `plot()`

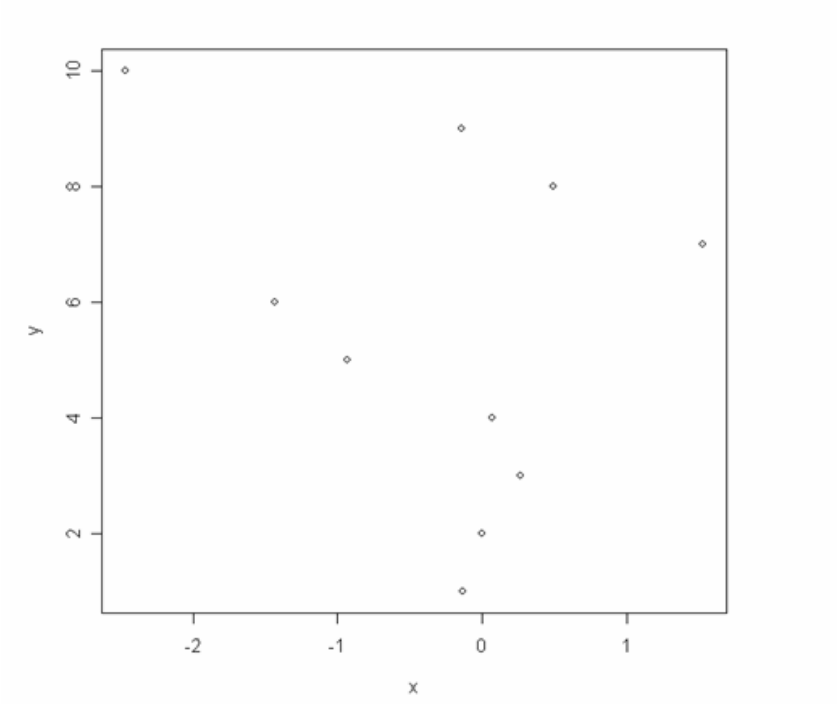
یکی از پرکاربرد ترین توابع رسم نمودار در R تابع `plot()` است. در صورتی که از تابع `plot()` به شکل `plot(x)` یا `plot(x,y)` استفاده کنیم، که `x` و `y` دو بردارند، به ترتیب نمودار بردار `x` و نمودار پراکندگی `x` در مقابل `y` رسم خواهد شد.

مثال :

```
> x <- rnorm(10)
> y <- 1:10
> plot(x)
```



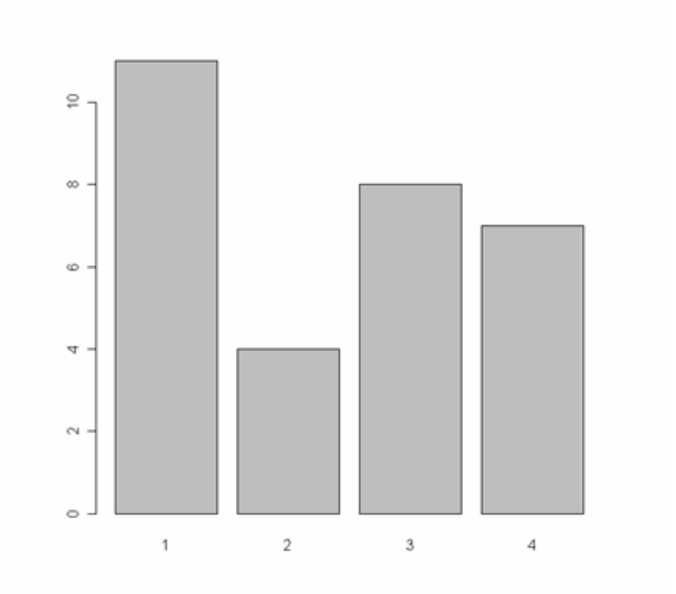

```
> plot(x,y)
```



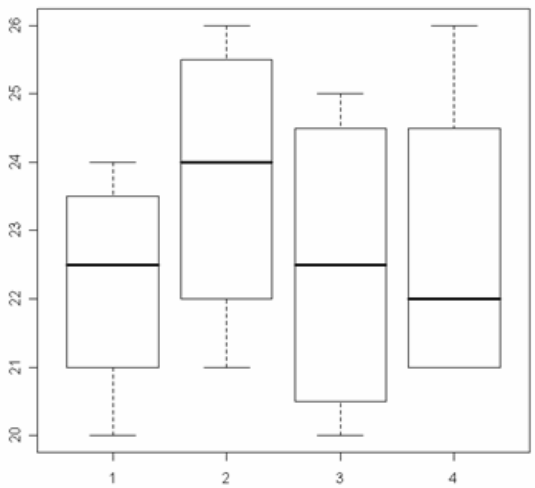
در صورتی که از تابع plot() به شکل plot(f) یا plot(f,y) استفاده کنیم، که f یک فـاکتور و y یـک بردار است، به ترتیب نمودار میله ای f و نمودار جعبه ای y در مقابل f رسم خواهد شد.

مثال :

```
> z <- sample(1:4,30,rep=T)
> z.f <- factor(z)
> plot(z.f)
```




```
> x <- rep(c(1,2,3,4),each=4)
> x.f <- factor(x)
> x.f
 [1] 1 1 1 1 2 2 2 2 3 3 3 3 4 4 4 4
Levels: 1 2 3 4
> y <- c(20,23,22,24,25,26,21,23,20,21,25,24,23,21,21,26)
> plot(x.f,y)
```

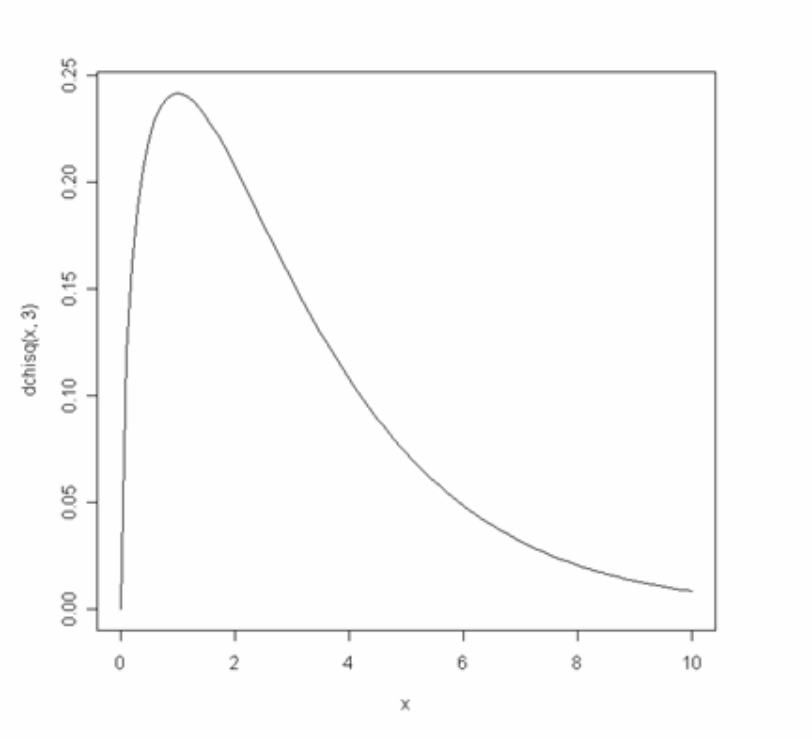


1.2. تابع curve()

از تابع curve() برای رسم منحنی یک تابع در محدوده تعیین شده استفاده می شود. از این تابع می توان برای بررسی منحنی توزیع های آماری استفاده کرد. تابعی که می خواهیم منحنی آن رسم شود یا باید به شکل کلی تابعی از " x " باشد، و یا باید تابعی از یک بردار باشد.

مثال :

```
> curve(dchisq(x,3),0,10)
```

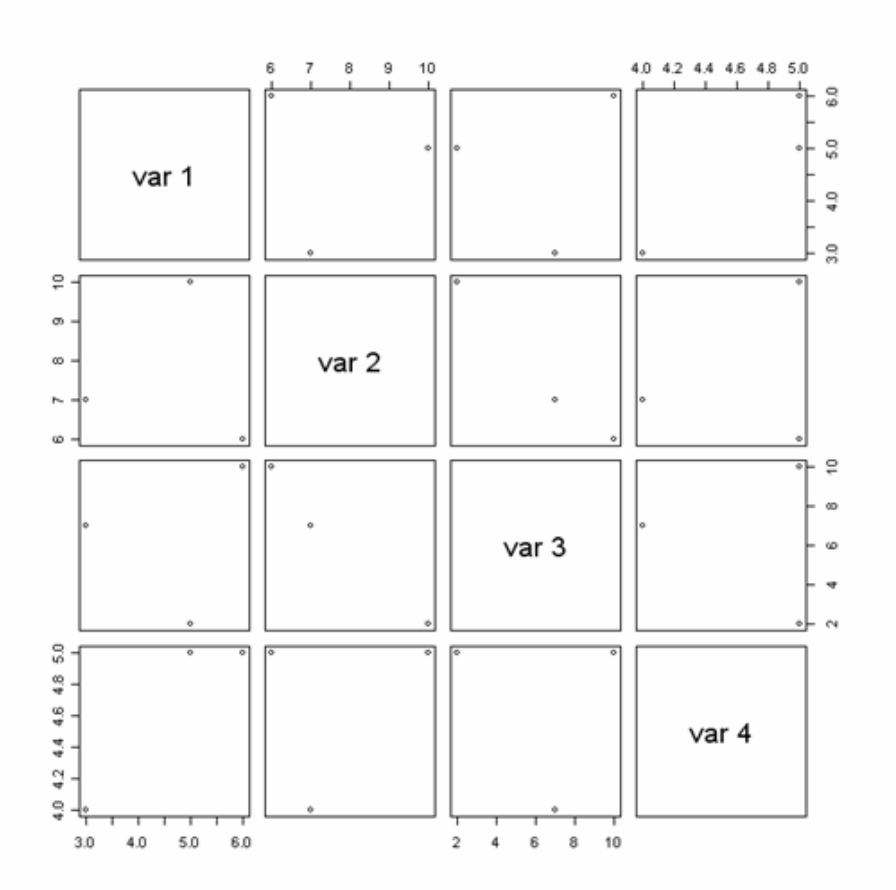


1.3. تابع pairs()

از تابع pairs() برای رسم هر ستون یک ماتریس عددی در مقابل ستون های دیگر استفاده می شود.

مثال :

```
> x <- sample(1:10,12,rep=T)
> dim(x) <- c(3,4)
> x
     [,1] [,2] [,3] [,4]
[1,]    3    7    7    4
[2,]    5   10    2    5
[3,]    6    6   10    5
> pairs(x)
```



1.4. تابع coplot()

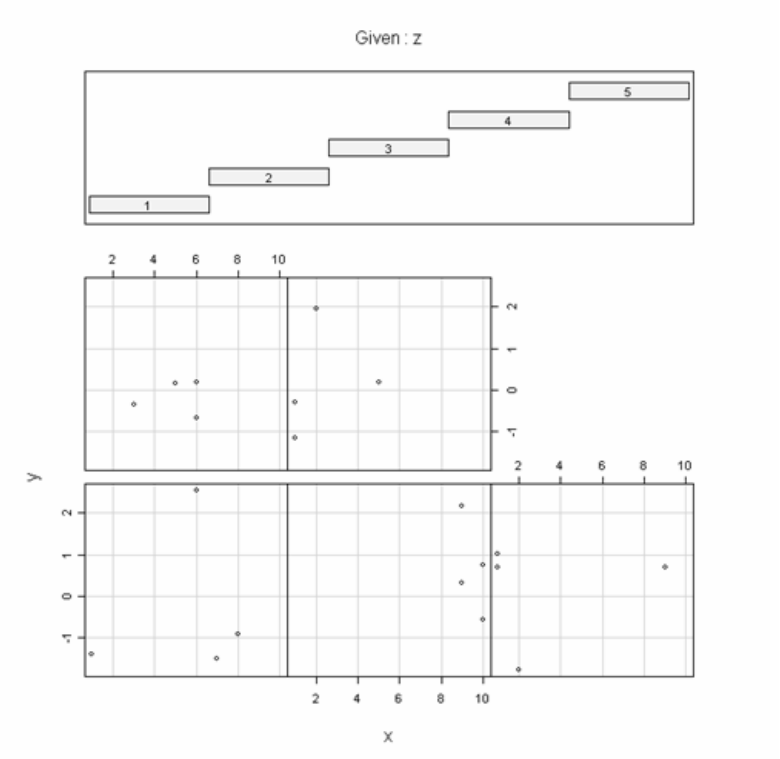
برای رسم یک بردار در مقابل یک بردار دیگر به شرط یک بردار یا یک فاکتور از تابع coplot() با ساختار کلی زیر استفاده می کنیم.

```
> coplot(a~b|c)
```

در صورتی که شرط تابع یک فاکتور باشد دو بردار در سطوح آن فاکتور در مقابل هم رسم خواهند شد.

مثال :

```
> x <- sample(1:10,20,rep=T)
> y <- rnorm(20)
> z <- rep(c(1,2,3,4,5),each=4)
> z <- factor(z)
> coplot(y~x|z)
```

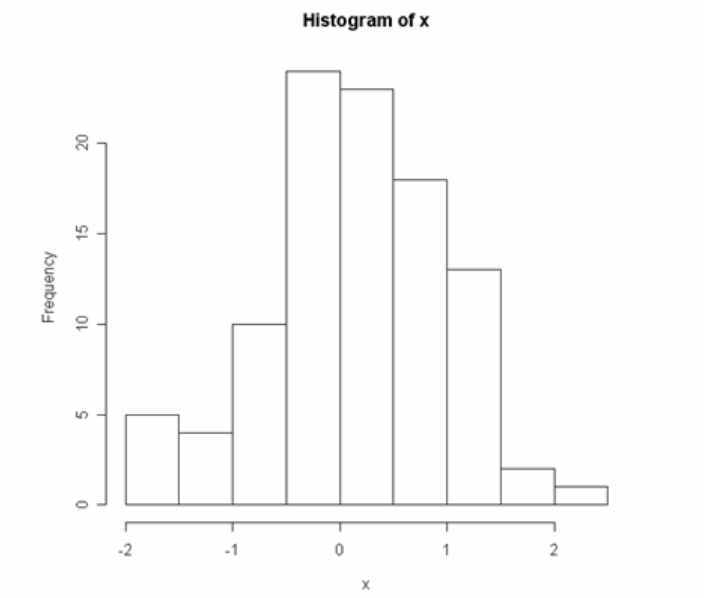


1.5. تابع `hist()`

از تابع `hist()` برای رسم هیستوگرام یک بردار استفاده می شود.

مثال :

```
> x <- rnorm(100)
> hist(x)
```

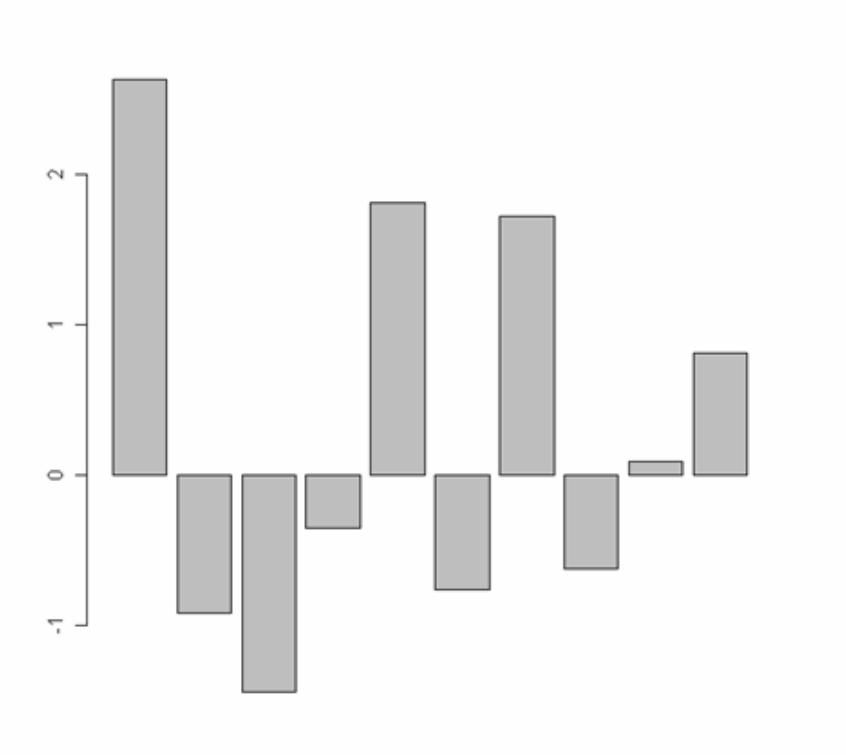


1.6. تابع barplot()

برای رسم نمودار میله ای یک بردار از تابع barplot() استفاده می شود. بردار وارد شده در این تابع بردار فراوانی ها می باشد.

مثال :

```
> x <- rnorm(10)
> barplot(x)
```

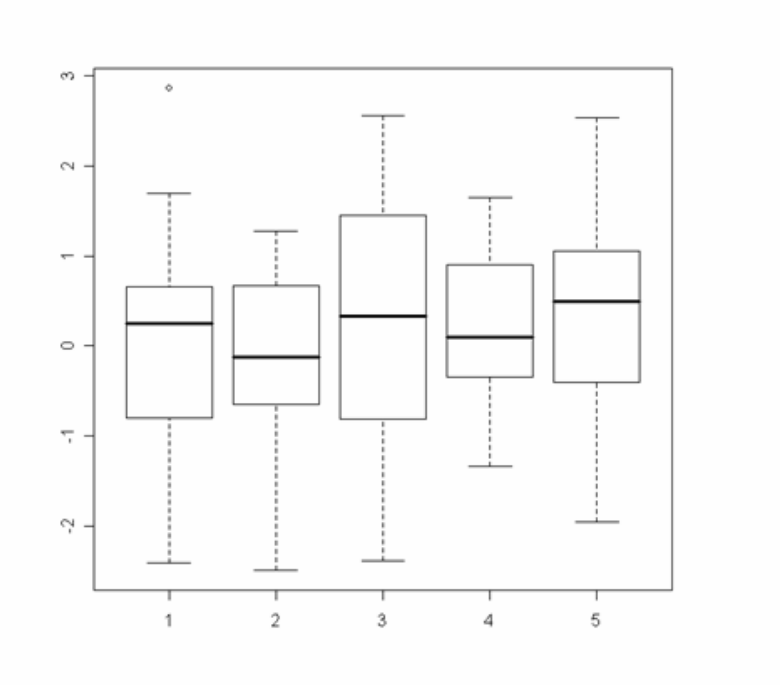


1.7. تابع boxplot()

از تابع boxplot() برای رسم نمودار جعبه ای استفاده می شود. برای رسم نمودار جعبه ای توسط این تابع می توان از یک بردار یا از فرمولی شامل بردار و فاکتور استفاده کرد.

مثال :

```
> x <- rnorm(100)
> y <- factor(rep(1:5,times=20))
> boxplot(x~y)
```

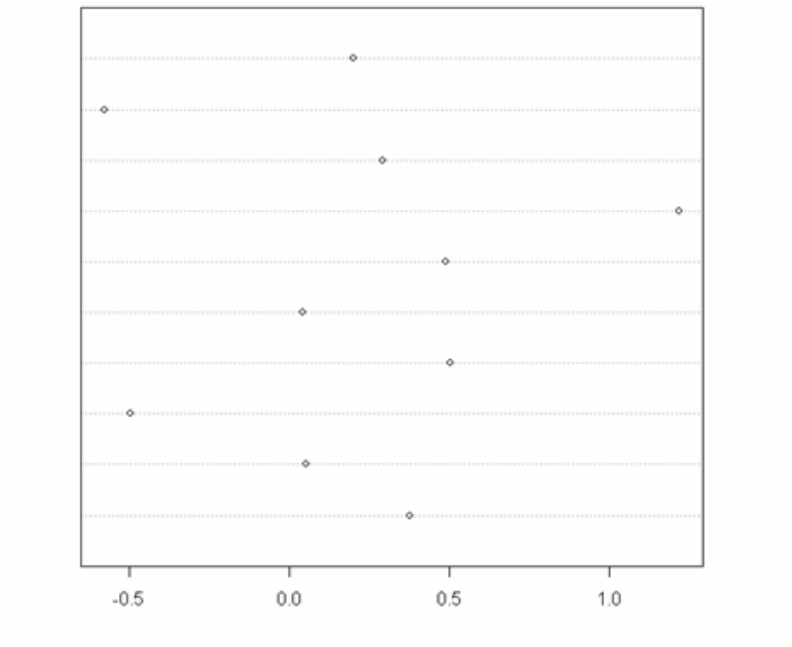


1.8. تابع `dotchart()`

برای رسم نمودار نقطه ای یک بردار از تابع `dotchart()` استفاده می کنیم.

مثال :

```
> x <- rnorm(10)
> dotchart(x)
```

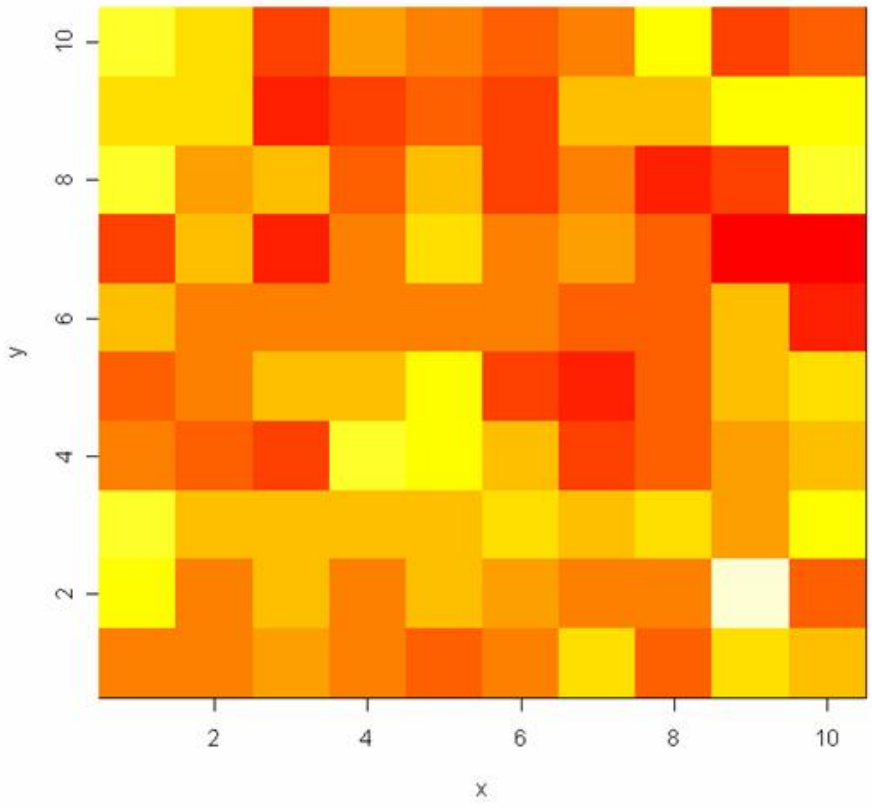


1.9. توابع ترسیم سه متغیری

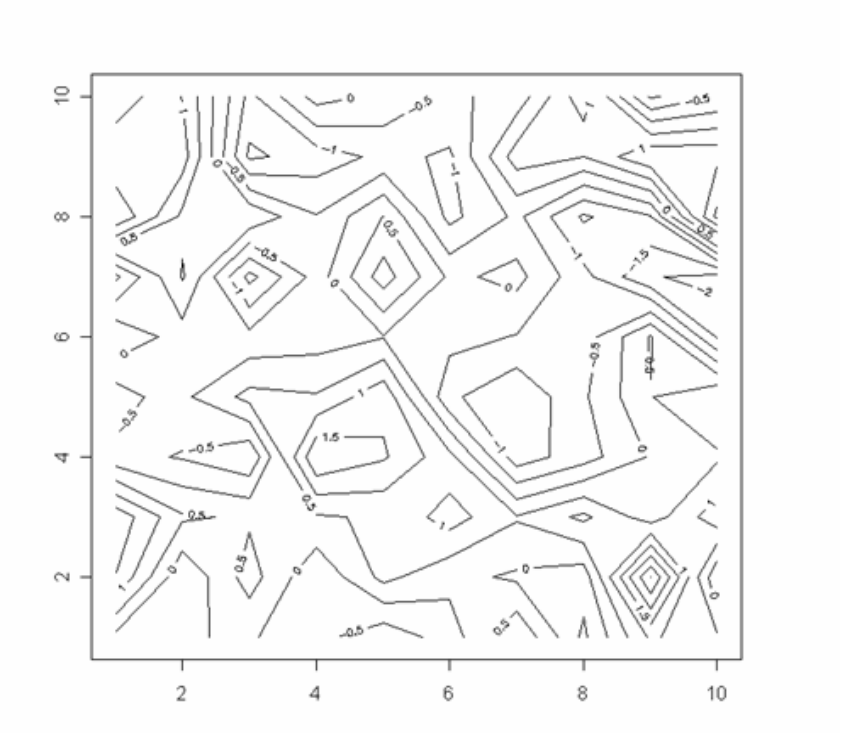
توابع ترسیم سه متغیری عبارتند از `image()` ، `contour()` ، و `persp()`. تفاوت این سه تابع در شیوه نمایش نمودار در آنهاست. تابع `image()` متغیر سوم را با رنگ های متفاوت نشان می دهد. تابع `contour()` متغیر سوم را به صورت نقشه ای نشان می دهد، و تابع `persp()` نمودار را به صورت سه بعدی نمایش می دهد. نکته ای که باید به آن توجه داشت این است که در هر سه تابع فوق باید متغیر سوم را به صورت ماتریس وارد کرد.

مثال :

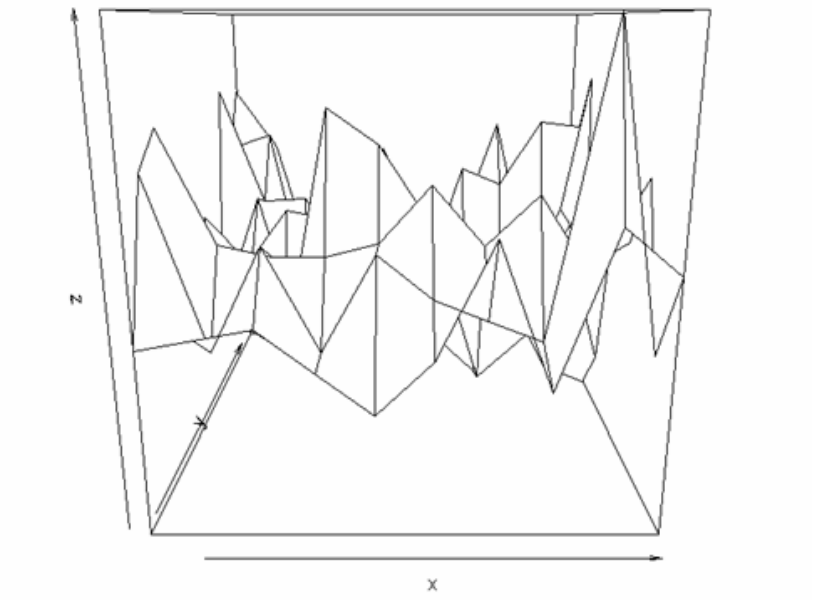
```
> x <- 1:10
> y <- 1:10
> z <- rnorm(100)
> dim(z) <- c(10,10)
> image(x,y,z)
```



```
> contour(x,y,z)
```




```
> persp(x,y,z)
```

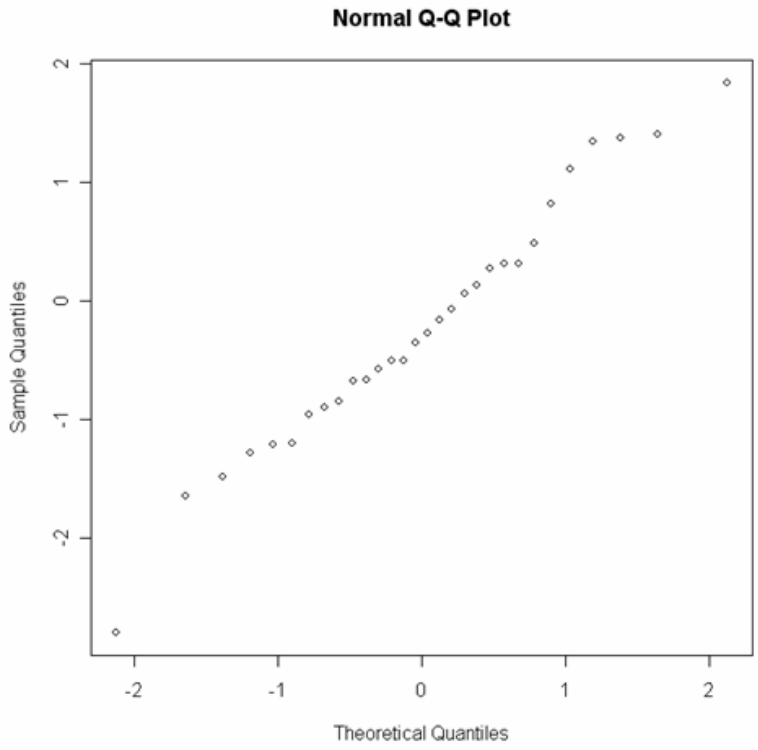


1.10. توابع مقایسه توزیع ها

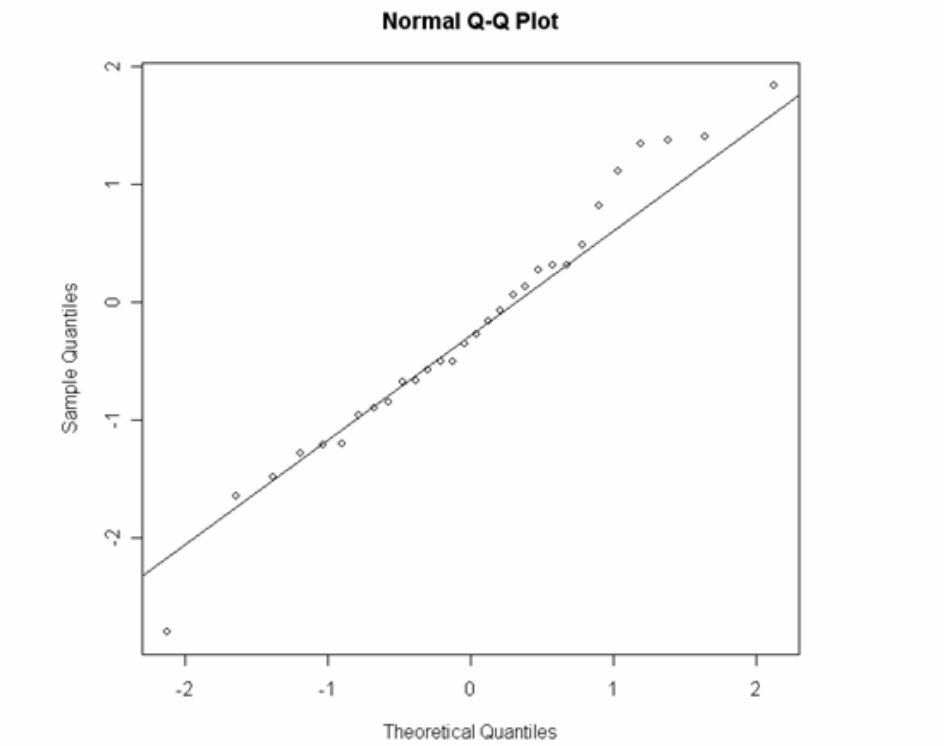
برای بررسی و مقایسه توزیع ها از سه تابع `qqnorm()` ، `qqline()` ، و `qqplot()` استفاده می شود. برای بررسی نرمال بودن توزیع داده های یک بردار از تابع `qqnorm()` استفاده می شود. برای رسم خط نرمال روی نمودار رسم شده از تابع `qqline()` استفاده می شود، و برای مقایسه توزیع دو متغیر و بررسی همتوزیع بودن آنها از تابع `qqplot()` استفاده می شود.

مثال :

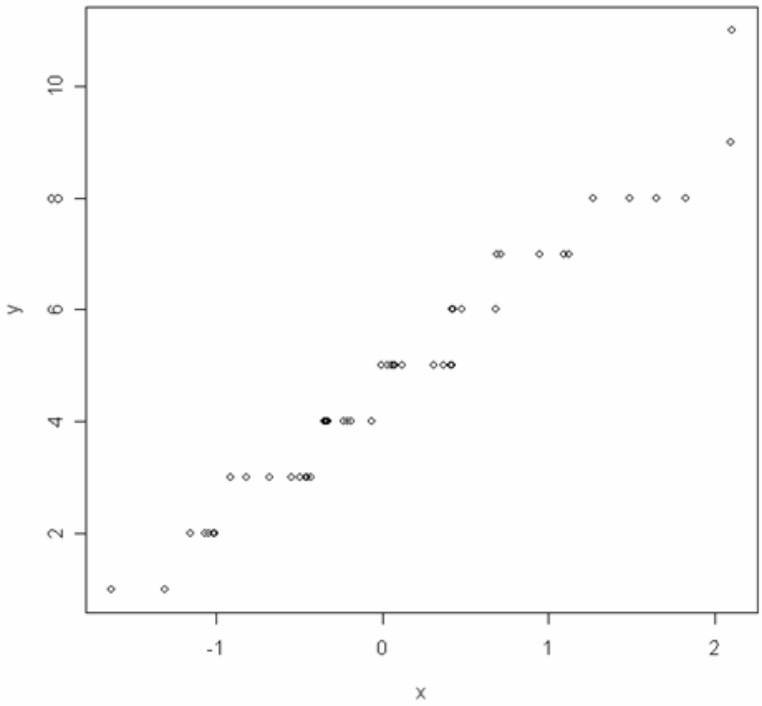
```
> x <- rnorm(30)
> qqnorm(x)
```



```
> qqline(x)
```




```
> x <- rnorm(50)
> y <- rbinom(50,100,0.05)
> qqplot(x,y)
```

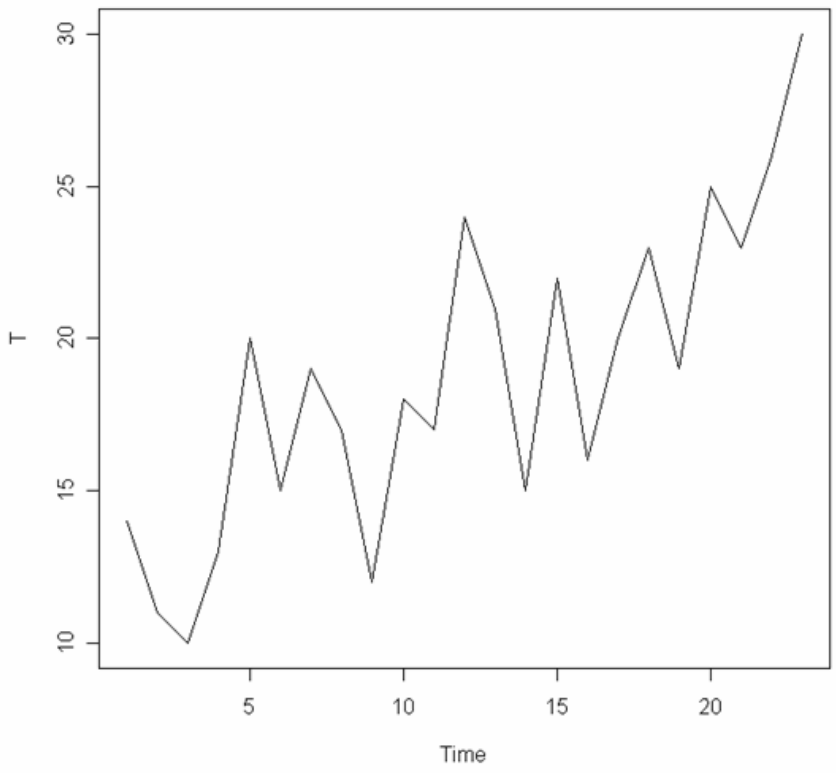


1.11. نمودارهای سری زمانی

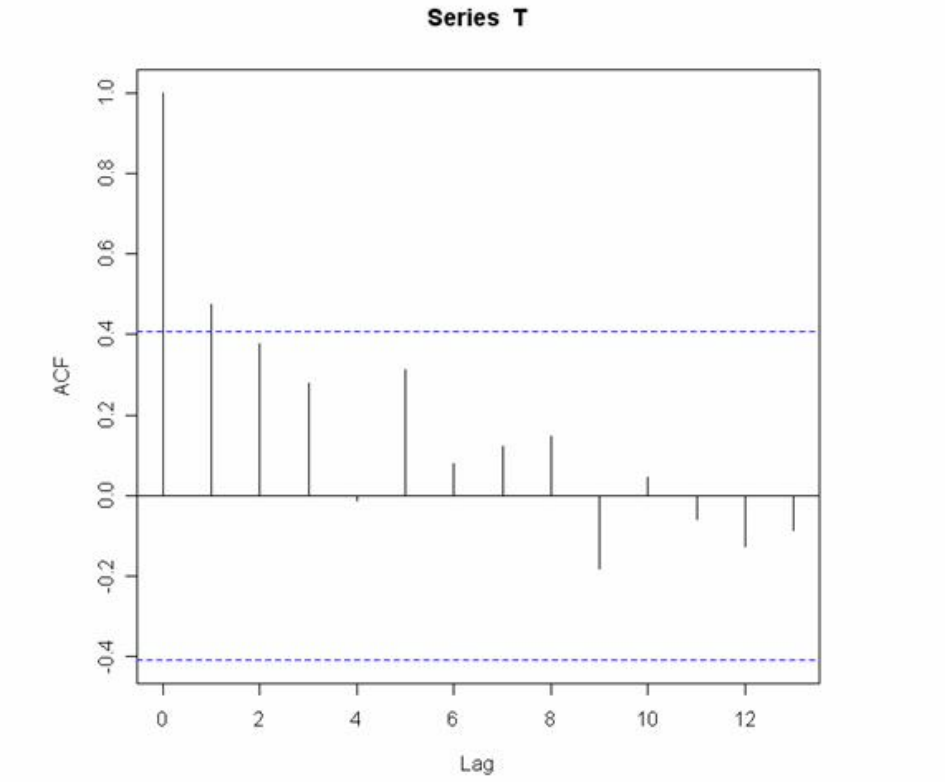
برای رسم نمودارهای سری زمانی ابتدا باید بردار مشاهدات را به سری زمانی تبدیل کنیم. برای تبدیل بردار مشاهدات به سری زمانی از تابع `ts()` استفاده می کنیم. پس از تبدیل یک بردار به یک سری زمانی از توابع `acf()` و `pacf()` برای رسم نمودارهای ACF و PACF سری زمانی مورد نظر استفاده می کنیم.

مثال :

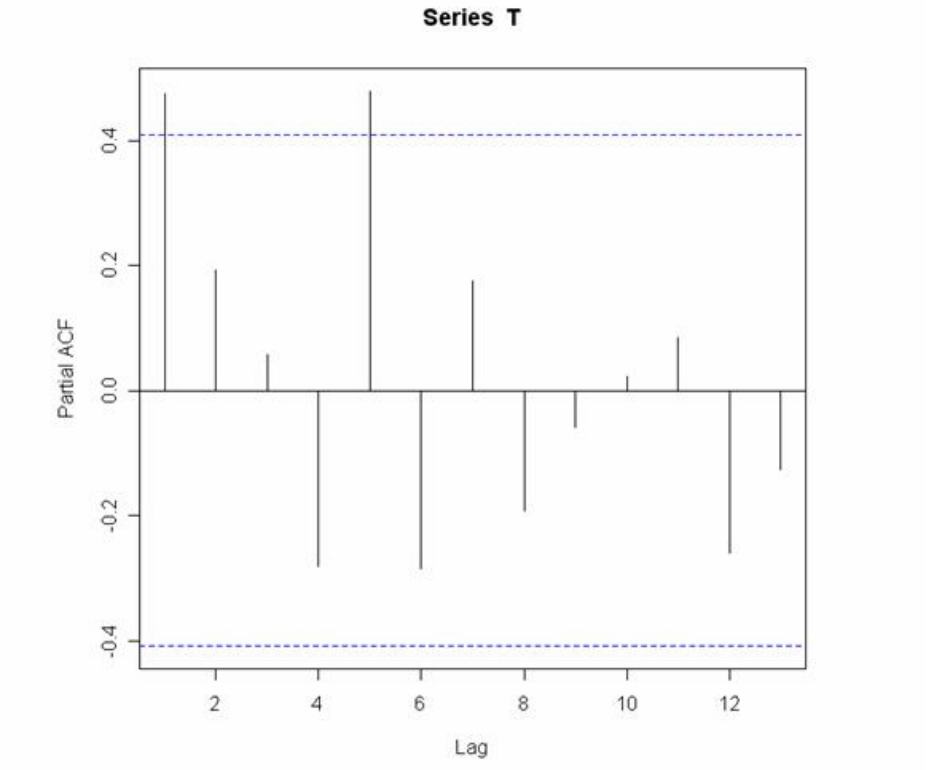
```
> x <-
c(14,11,10,13,20,15,19,17,12,18,17,24,21,15,22,16,20,23,19,25,23,26,
30)
> T <- ts(x)
> plot(T)
```



```
> acf(T)
```




```
> pacf(T)
```



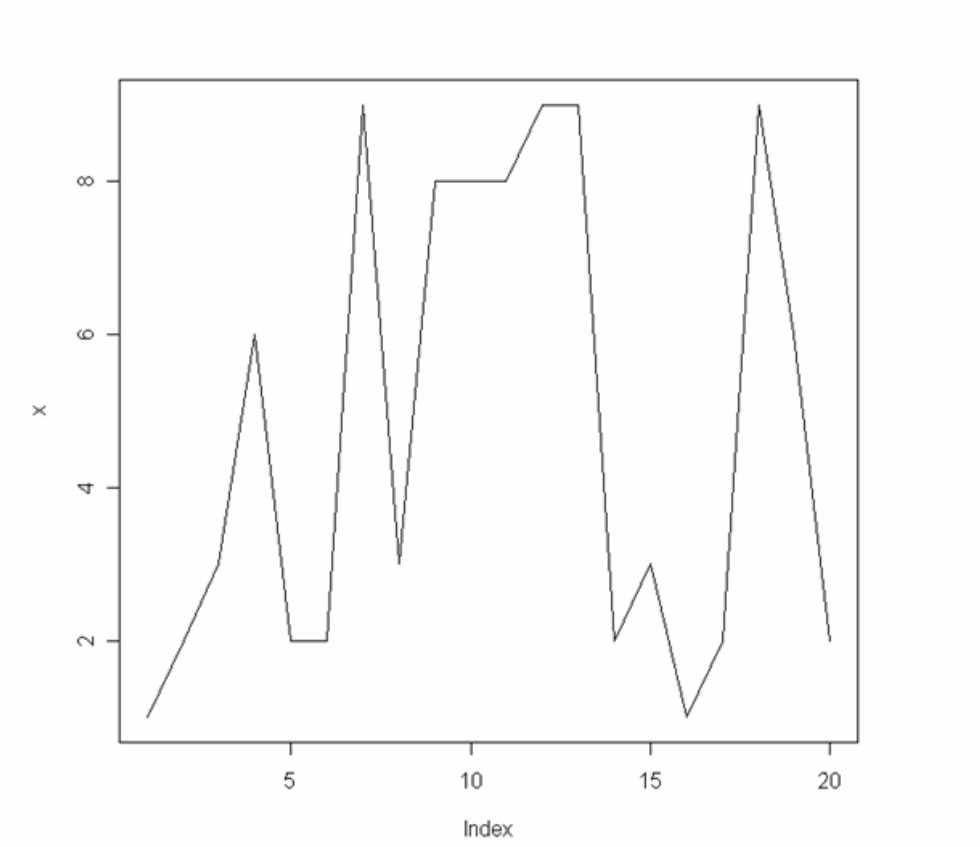
1.12. جزئیات رسم نمودار

در توابع ترسیم سطح بالا آرگومان هایی وجود دارند که با مقدار دهی آنها می توان به صورت دلخواه برخی جزئیات رسم نمودار را تغییر داده و یا جزئیاتی را به نمودار اضافه کرد. برخی از این آرگومان ها به شرح زیرند.

type= : با استفاده از این آرگومان می توان نوع نمایش نمودار را تعیین کرد. مقدار پیش فرض این آرگومان " p " است که مشخص می کند نمودار به صورت نقطه ای نمایش داده شود. می توان این آرگومان را به دلخواه مقدار دهی کرد. از مقدار " l " برای نمایش خطی، از مقدار " b " و " o " برای نمایش خطی و نقطه ای، از مقدار " h " برای نمایش به صورت خط های عمود بر محور افقی، و از مقدار " s " و " S " برای نمایش پله ای استفاده می شود. در صورتی که از مقدار " n " استفاده شود نمودار رسم شده اما چیزی نمایش داده نمی شود.

مثال :

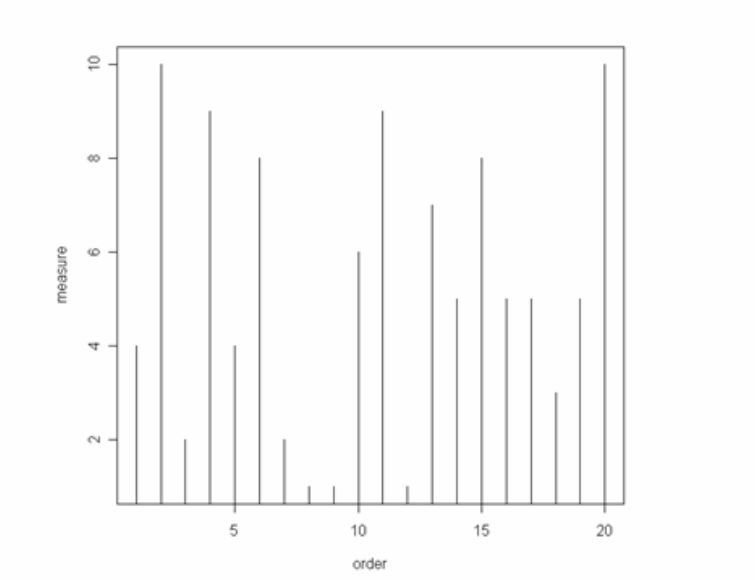
```
> x <- sample(1:10,20,rep=T)
> plot(x)
> plot(x,type="l")
```



xlab= و ylab= : از این دو آرگومان برای برچسب گذاری محور افقی و عمودی استفاده می کنیم.

مثال :

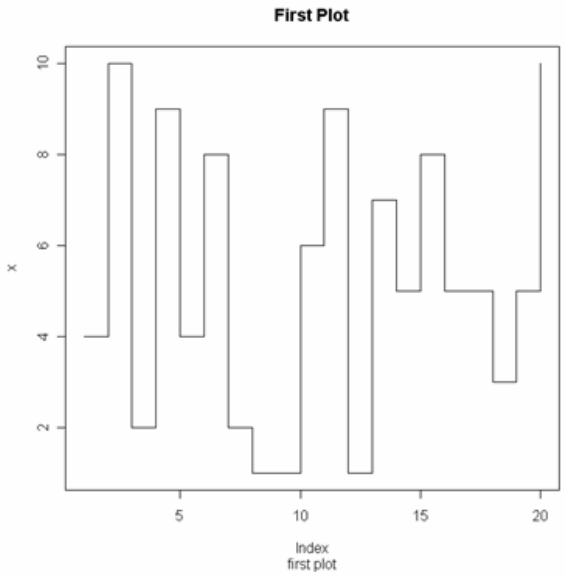
```
> plot(x,type="h",xlab="order",ylab="measure")
```



main= و sub= : از main= برای قرار دادن عنوان نمودار در بالا و از sub= برای قرار دادن زیرعنوان نمودار در پایین استفاده می شود.

مثال :

```
> plot(x,type="s",main="First Plot",sub="first plot")
```



col= : برای انتخاب رنگ نمودار از col= استفاده کرده و در مقابل آن نام یا شماره رنگ مورد نظر را وارد می کنیم.

2. دستورهای ترسیم سطح پایین

دستورهای ترسیم سطح پایین ما را قادر می سازند تا مواردی از قبیل نقطه ها، خطوط، و نوشته ها را به نمودارهای رسم شده توسط دستورهای ترسیم سطح بالا اضافه کنیم. برخی از توابع ترسیم سطح پایین به شرح زیرند.

2.1. تابع `text()`

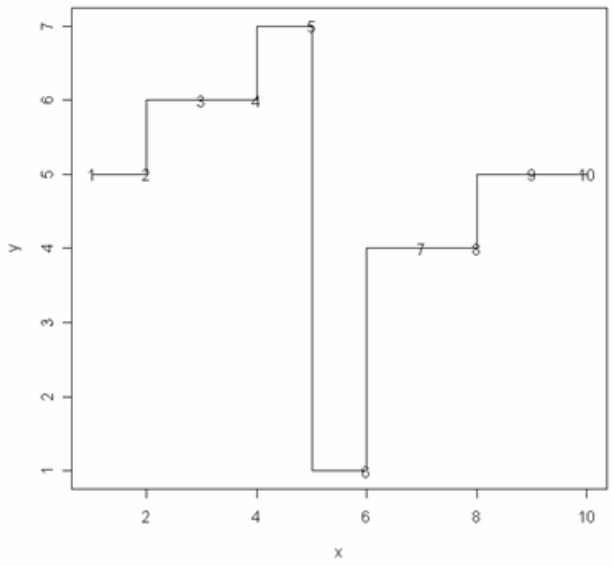
این تابع ما را قادر می سازد تا برداری از اعداد یا کاراکترها را به نقاط یک نمودار اضافه کنیم. ساختار کلی این تابع به شکل زیر است.

```
text(x,y,labels, ...)
```

در این تابع `x` و `y` بردارهای مشخص کننده نقاط مورد نظر، و `lables` بردار شامل اعداد یا کراکترهای مورد نظر است.

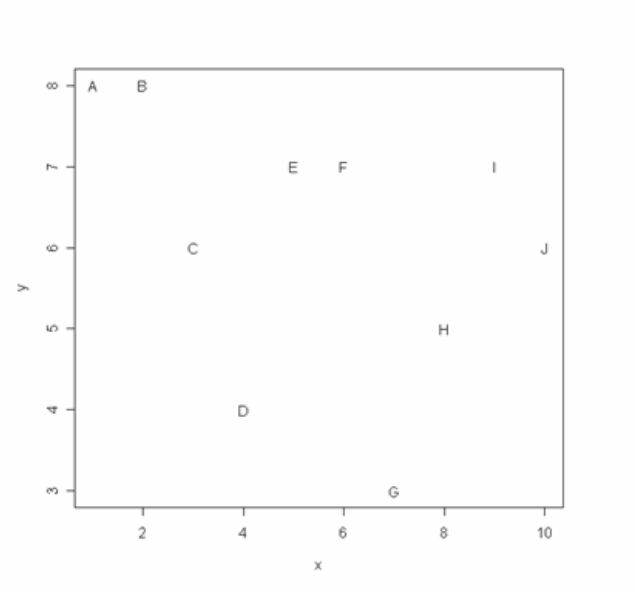
مثال :

```
> x <- 1:10
> y <- rbinom(10,10,0.5)
> plot(x,y,type="S")
> text(x,y,c(1:10))
```



مثال : یکی از بیشترین موارد استفاده از تابع `text()` به صورت زیر است.

```
> plot(x,y,type="n")
> text(x,y,c("A","B","C","D","E","F","G","H","I","J"))
```

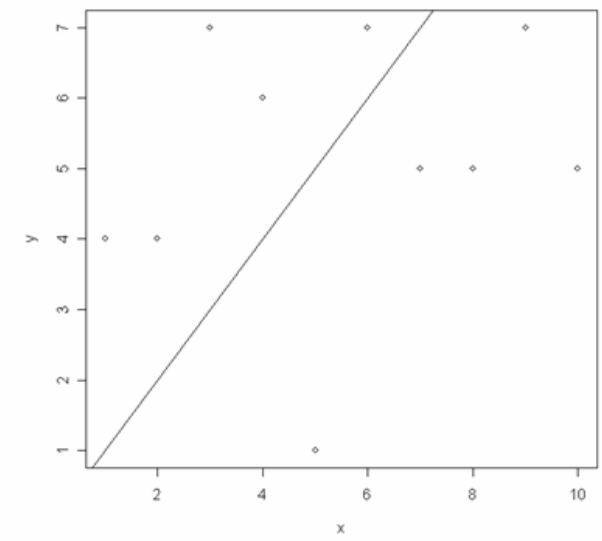


2.2. تابع `abline()`

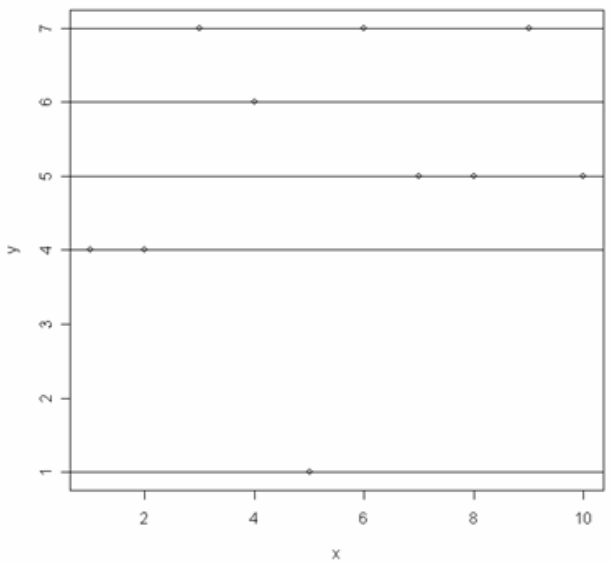
برای رسم یک خط با شیب و عرض از مبداء دلخواه روی یک نمودار از تابع `abline()` استفاده مـی شـود. در صورتی که از این تابع به شکل `abline(a,b)` استفاده کنیم خطی با عرض از مبداء `a` و شیب `b` روی نمـودار رسم خواهد شد. اگر از این تابع به شکل `abline(v=x)` یا `abline(h=y)` استفاده کنیم از هر نقطه نمـودار به ترتیب خطوط عمود بر محور x یا y رسم خواهد شد.

مثال :

```
> x <- 1:10
> y <- rpois(10, 5)
> plot(x,y)
> abline(0,1)
```



```
> plot(x,y)
> abline(h=y)
```

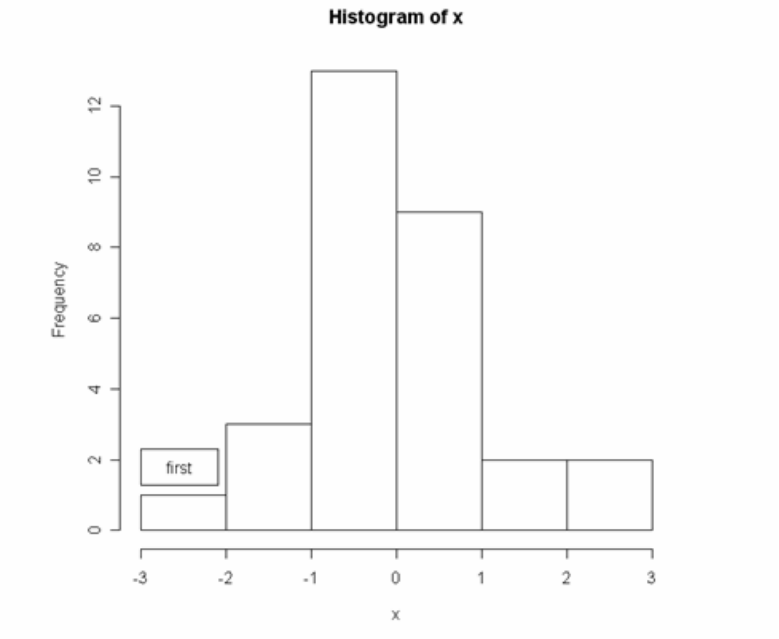


2.3. تابع `legend()`

از تابع `legend()` برای وارد کردن یک نوشته درقسمتی از نمودار استفاده می کنیم. در این تابع بایـد مختـصات نقطه مورد نظر و نوشته مورد نظر را وارد کنیم.

مثال :

```
> x <- rnorm(30)
> hist(x)
> legend(-3,2.3,"first     ")
```



در این تابع می توان آرگومان هایی نیز وارد کرد. برخی از این آرگومان ها به شرح زیرند.

`fill=` : با وارد کردن نام یا شماره رنگ مربعی با آن رنگ وارد نمودار می کند.

`lty=` : با وارد کردن شماره خط مورد نظر خطی با آن نوع روی نمودار رسم می کند.

`lwd=` : با وارد کردن اندازه ضخامت خطی با آن ضخامت روی نمودار رسم می کند.

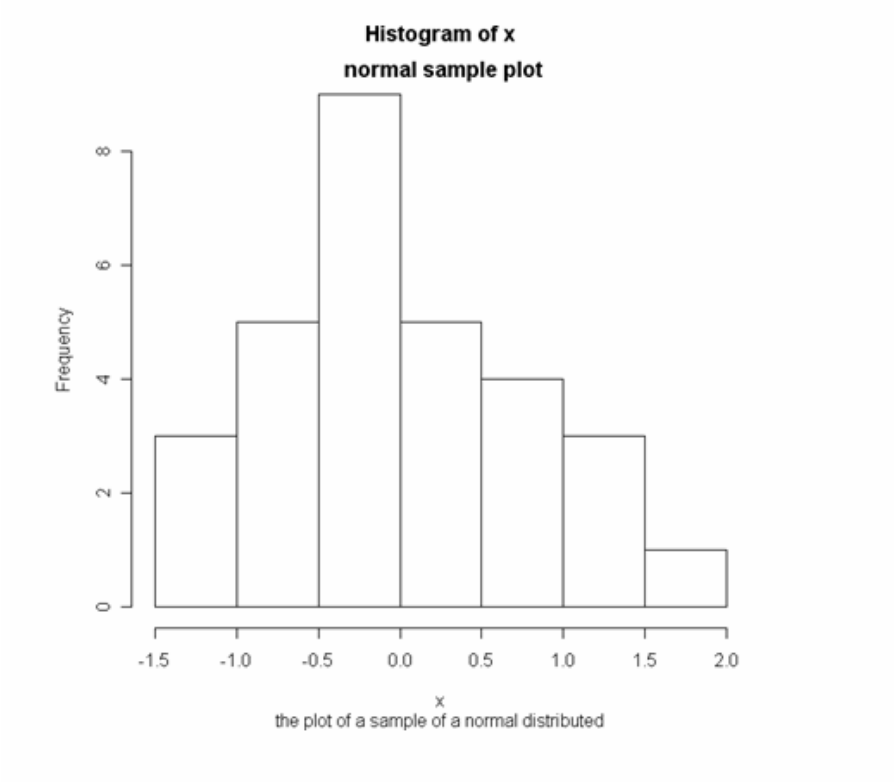
`pch=` : با وارد کردن شماره یا کاراکتر مورد نظر آن کاراکتر را وارد نمودار می کند.

2.4. تابع `title()`

از این تابع برای وارد کردن عنوان در بالای نمودار و زیرعنوان در پایین نمودار استفاده می شود.

مثال :

```
> hist(x)
> title("\n\n\n normal sample plot"," the plot of a sample of a
normal distributed ")
```



3. دستورات ترسیم تعاملی

دستورات ترسیم تعاملی به ما این امکان را می دهند تا از طریق موس اطلاعاتی را از نمودار دریافت کرده یا به آن وارد کنیم. توابع ترسیم تعاملی به شرح زیرند.

3.1. تابع `locator()`

از این تابع برای تعیین مختصات نقاط توسط موس استفاده می کنیم. با قرار دادن این تابع در توابع دیگر می توان از مقداری که این تابع به عنوان مختصات نقطه کلیک شده توسط موس بر می گرداند استفاده کرد. ساختار کلی ایـن تابع به شکل زیر است.

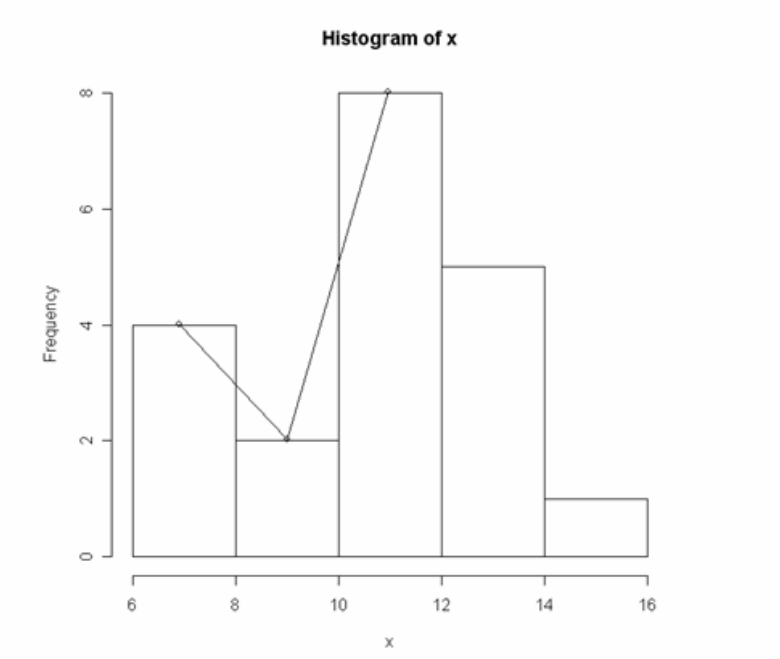
```
locator(n,type)
```

آرگومان `n` مشخص می کند که چند نقطه توسط موس کلیک شود، و آرگومان `type` همانند آرگومان `type` در توابع ترسیم سطح بالا عمل می کند و می تواند مقادیر " `n` "، " `l` "، " `p` "، و یا " `o` " را اختیار کند.

مثال :

```
> x <- rbinom(20, 20, 0.5)
> hist(x)
> locator(3,"o")
$x
[1]  6.916213  8.992383 10.951034

$y
[1] 4.000000 2.033199 8.003223
```

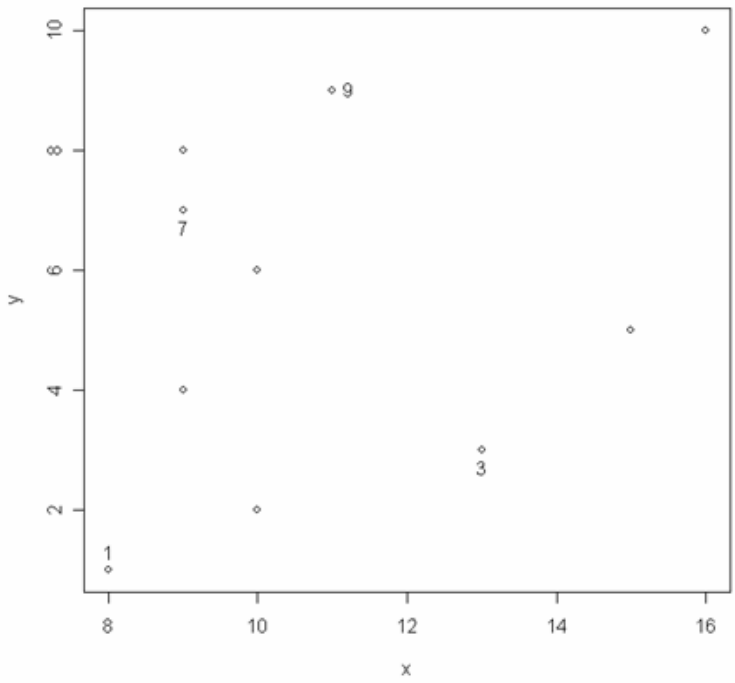


3.2. تابع `identify()`

با استفاده از این تابع می توان ترتیب هر نقطه از نمودار را با کلیک موس روی آن نقطه نمایش داد.

مثال :

```
> x <- rpois(10,10)
> y <- 1:10
> plot(x,y)
> identify(x,y)
```



4. تغییر پارامترهای توابع

برای تغییر پارامترهای توابع از تابع par() استفاده می شود. در صورت بکار گیری این تابع بدون آرگومان لیستی از پارامترهای قابل تغییر توابع ترسیم به نمایش در خواهد آمد. از تابع par() قبل از رسم نمودار استفاده می شود. برای کسب اطلاعات دقیق در مورد این تابع می توان با استفاده از دستور ?par() از فایل help کمک گرفت.

آزمون های آماری

1. آزمون t'student

آزمون t'student برای انجام آزمون در مورد میانگین یک یا دو جامعه مورد استفاده قرار مـی گیـرد. بـرای انجـام آزمون t از تابع `t.test()` استفاده می کنیم. ساختار کلی این تابع به شکل زیر است.

```
t.test(x,y,alternative,mu,paired,var.equal,conf.level)
```

در ساختار فوق برای انجام آزمون یک نمونه ای فقط یک بردار عددی در قسمت x به عنوان بردار مشاهدات وارد می کنیم و برای انجام آزمون دو نمونه ای علاوه بر بردار x یک بردار عـددی نیـز در قـسمت y بـه عنـوان بـردار مـشاهدات دوم وارد مـی کنـیم. در قـسمت `altervative` فـرض مقابـل را بـه صـورت " `graeter` "، " `less` " ، و یا " `two.sided` " برای انجام آزمون یک طرفه یا دو طرفه به صورت دلخواه وارد می کنـیم. در قسمت `mu` مقدار میانگین تحت فرض صفر را وارد می کنیم. در قسمت `paired` اگر مقدار `TRUE` را وارد کنیم آزمون به صورت جفتی انجام می شود که در این حالت باید اندازه دو بردار وارد شده یکسان باشد. در صورتی که مقدار `paired` را `FALSE` وارد کنیم آزمون به صورت جفتی انجام نمی شـود. در قـسمت `var.equal` اگـر مقدار `TRUE` را وارد کنیم آزمون با شرط برابری واریانس ها انجام می شود و اگر مقدار `FALSE` را وارد کنـیم آزمون با شرط نا برابری واریانس ها انجام می شود. در قسمت `conf.level` نیز باید سطح اطمینان مورد نظر را وارد کنیم. در ساختار تابع گزینه ها دارای مقادیر پیش فرض می باشند. مقادیر پیش فرض گزینه ها به صورت زیر است.

```
t.test(x,y = NULL,alternative = c("two.sided", "less", "greater"),
mu = 0,paired = FALSE,var.equal = FALSE,conf.level = 0.95)
```

مثال (آزمون یک نمونه ای) :

```
> x <- rnorm(10)
> t.test(x)

        One Sample t-test

data:  x
t = -0.7416, df = 9, p-value = 0.4772
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 -1.0486421  0.5308475
sample estimates:
 mean of x
-0.2588973


> x <- rnorm(10,5,25); mean(x)
[1] 1.971701
```

```
> t.test(x,mu=5,alternatinve="less")

        One Sample t-test

data:  x
t = -0.5815, df = 9, p-value = 0.5752
alternative hypothesis: true mean is not equal to 5
95 percent confidence interval:
 -9.809701 13.753104
sample estimates:
mean of x
 1.971701
```

مثال (آزمون دو نمونه ای) :

```
> x <- rnorm(30); y <- rnorm(25)
> t.test(x,y,var.equal=T)

        Two Sample t-test

data:  x and y
t = 0.8449, df = 53, p-value = 0.402
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.3073231  0.7546665
sample estimates:
mean of x mean of y
0.3012040 0.0775323

> t.test(x,y,var.equal=F)

        Welch Two Sample t-test

data:  x and y
t = 0.855, df = 52.835, p-value = 0.3964
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.3010520  0.7483953
sample estimates:
mean of x mean of y
0.3012040 0.0775323
```

مثال (آزمون جفتی) :

```
> x <- rpois(15,5); y <- rpois(15,6)
> t.test(x,y,paired=T)

        Paired t-test

data:  x and y
t = -1.9335, df = 14, p-value = 0.07365
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.9373174  0.2039841
sample estimates:
mean of the differences
              -1.866667
```

2. آزمون F

آزمون F برای انجام آزمون در مورد واریانس دو جامعه نرمال مورد استفاده قرار می گیرد. برای انجام آزمـون F از تابع `var.test()` استفاده می کنیم. ساختار کلی این تابع به شکل زیر است.

```
var.test(x,y,ratio,alternative,conf.level)
```

در این تابع در `x` و `y` بردارهای مشاهدات را وارد می کنیم. در قسمت `ratio` نسبت مورد نظر تحت فر ض صفر را وارد می کنیم. در قسمت های `alternative` و `conf.level` نیز به ترتیب مقادیر فرض مقابـل و سـطح اطمینان مورد نظر را وارد می کنیم. مقدار پیش فرض گزینه های این تابع به صورت زیر است.

```
var.test(x,y,ratio = 1,alternative = c("two.sided", "less",
"greater"),conf.level = 0.95)
```

مثال:

```
> x <- rnorm(30,0,1)
> y <- rnorm(30,0,5)
> var.test(x,y)

        F test to compare two variances

data:  x and y
F = 0.0237, num df = 29, denom df = 29, p-value < 2.2e-16
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.01128097 0.04979629
sample estimates:
ratio of variances
        0.02370128

> var.test(x,y,ratio=0.02)

        F test to compare two variances

data:  x and y
F = 1.1851, num df = 29, denom df = 29, p-value = 0.6505
alternative hypothesis: true ratio of variances is not equal to 0.02
95 percent confidence interval:
 0.01128097 0.04979629
sample estimates:
ratio of variances
        0.02370128
```

3. آزمون دو جمله ای

آزمون دو جمله ای برای انجام آزمون در مورد نسبت موفقیت مورد استفاده قرار می گیرد. بـرای انجـام آزمـون دو جمله ای از تابع `binom.test()` استفاده می کنیم. ساختار کلی این تابع به شکل زیر است.

```
binom.test(x,n,p,alternative,conf.level)
```

در این تابع `x` تعداد موفقیـت، `n` تعـداد آزمـایش هـا، و `p` احتمـال موفقیـت تحـت فـرض صـفر اسـت. مقـادیر `alternative` و `conf.level` نیز به ترتیب فرض مقابل و سطح اطمینان مورد نظـر هـستند. مقـدار پـیش فرض گزینه های فوق به صورت زیر است.

```
binom.test(x,n,p = 0.5,alternative = c("two.sided", "less",
"greater"),conf.level = 0.95)
```

مثال :

```
> binom.test(48,90,0.5)

        Exact binomial test

data:  48 and 90
number of successes = 48, number of trials = 90, p-value = 0.5984
alternative hypothesis: true probability of success is not equal to
0.5
95 percent confidence interval:
 0.4251273 0.6392843
sample estimates:
probability of success
             0.5333333
```

4. آزمون chi-square

آزمون chi-square برای انجام آزمون در مورد نیکویی برازش و یا بررسی استقلال در جـداول پیـشایندی مـورد استفاده قرار می گیرد. برای انجام آزمون chi-square از تابع `chisq.test()` استفاده می کنیم. ساختار کلـی این تابع به شکل زیر است.

```
chisq.test(x,y,correct,p,rescale.p,simulate.p.value,B)
```

در ساختار فوق `x` یک بردار یا یک ماتریس و `y` یک بردار است. در صورتی که بخواهیم آزمـون نیکـویی بـرازش انجام دهیم تنها در قسمت `x` بردار فراوانی ها را وارد می کنیم و نیازی نیست در قسمت `y` چیزی وارد شـود. اگـر بخواهیم آزمون استقلال در جداول پیشایندی انجام دهیم یا جدول را به صورت یـک مـاتریس در قـسمت `x` وارد

می کنیم و یا بردارهای جمع هـای حاشـیه ای را در قـسمت x و y وارد مـی کنـیم. در قـسمت correct در صورتی که بخواهیم تصحیح پیوستگی انجام شود مقدار TRUE ، و در غیر این صورت مقـدار FALSE را وارد می کنـیم. در قـسمت p بـردار احتمـال هـا را بـرای انجـام آزمـون نیکـویی بـرازش وارد مـی کنـیم. در قـسمت rescale.p در صورتی که جمع احتمال های بردار p برابر یک نباشد مقدار TRUE را وارد می کنیم تا بردار از نو مقدار دهی شود به صورتی که جمع احتمـالات برابـر یـک شـود. در قـسمت simulate.p.value در صورتی که بخواهیم p-value از روش مونت-کارلو شبیه سازی شود مقدار TRUE را وارد می کنیم. در قـسمت B نیز تعداد اجراهای روش مونت-کارلو را مشخص می کنیم. مقدار پیش فرض گزینه های فوق بـه صـورت زیـر است.

```
chisq.test(x,y = NULL,correct = TRUE,p = rep(1/length(x),length(x)),
rescale.p = FALSE,simulate.p.value = FALSE,B = 2000)
```

مثال (آزمون نیکویی برازش) :

```
> x <- c(15,18,12,17,11,19)
> prob <- rep(1/6,6)
> chisq.test(x,p=prob)

        Chi-squared test for given probabilities

data:  x
X-squared = 3.4783, df = 5, p-value = 0.6267
```

مثال (آزمون استقلال) :

```
> x <- matrix(c(10,15,14,10,12,15),2,3)
> x
     [,1] [,2] [,3]
[1,]   10   14   12
[2,]   15   10   15
> chisq.test(x)

        Pearson's Chi-squared test

data:  x
X-squared = 1.7944, df = 2, p-value = 0.4077
```

آزمون های ناپارامتری

1. آزمون ویلکاکسون

برای انجام آزمون ناپارامتری در مورد میانه یک یا دو جامعه از آزمون یک نمونه ای یـا دو نمونـه ای ویلکاکـسون استفاده می شود. آزمون دو نمونه ای ویلکاکسون به نام آزمون من-ویتنی نیز شناخته می شود. برای انجـام آزمـون ویلکاکسون از تابع `wilcox.test()` استفاده می کنیم. ساختار کلی این تابع به شکل زیر است.

```
wilcox.test(x,y,alternative,mu,paired,exact,correct,conf.int,
conf.level)
```

در ساختار فوق در قسمت `x` بردار مشاهدات را وارد می کنیم و در قسمت `y` در صورتی کـه بخـواهیم آزمـون دو نمونه ای انجام دهیم بردار مشاهدات نمونه دوم را وارد می کنیم. در قسمت `alternative` فرض مقابل را بـه صورت `" graeter "` ، `" less "` ، و یا `" two.sided "` برای انجام آزمون یک طرفه یا دو طرفه به صورت دلخواه وارد می کنیم. در قسمت `mu` مقدار میانه را تحت فرض صفر وارد می کنیم. در قسمت `paired` اگر مقدار `TRUE` را وارد کنیم آزمون به صورت جفتی انجام می شود که در این حالت باید اندازه دو بردار وارد شده یکسان باشد. در صورتی که مقدار `paired` را `FALSE` وارد کنیم آزمون به صورت جفتی انجام نمی شود. در قـسمت `exact` اگر مقدار `TRUE` را وارد کنیم مقدار p-value به صورت دقیق محاسبه می شود، در غیر ایـن صـورت p-value با تقریب محاسبه می گردد. در قسمت `correct` در صورتی که بخـواهیم تـصحیح پیوسـتگی انجـام شود مقدار `TRUE` ، و در غیر این صورت مقدار `FALSE` را وارد می کنیم. در قسمت `conf.int` در صورتی که بخواهیم فاصله اطمینان محاسبه شود مقدار `TRUE` و در غیر این صورت مقدار `FALSE` را وارد می کنیم. در قسمت `conf.level` نیز باید سطح اطمینان مورد نظر را وارد کنیم. مقـدار پـیش فـرض گزینـه هـای فـوق بـه صورت زیر است.

```
wilcox.test(x,y = NULL,alternative = c("two.sided", "less",
"greater"),mu = 0,paired = FALSE,exact = NULL,correct = TRUE,
conf.int = FALSE,conf.level = 0.95)
```

مثال (آزمون یک نمونه ای) :

```
> x <- sample(c(-5:5),20,rep=T)
> x
 [1]  1 -2  2  0 -5  3 -2 -5  5  1 -3  1 -1 -2 -3 -1  3  3 -3 -5
> wilcox.test(x,exact=F)

        Wilcoxon signed rank test with continuity correction

data:  x
V = 71.5, p-value = 0.351
alternative hypothesis: true location is not equal to 0
```

مثال (آزمون من-ویتنی) :

```
> x <- sample(c(5:15),20,rep=T)
> y <- sample(c(0:20),20,rep=T)
> x
 [1] 15 13 13 14 13  6  7 10 12 14  6  9  7 15 14  8 11 14  5 12
> y
 [1] 15  3 20 11 15  0 12 19  7 10 14  6 19  0  5  6  3 11 13 13
> wilcox.test(x,y,exact=F)

        Wilcoxon rank sum test with continuity correction

data:  x and y
W = 218, p-value = 0.6347
alternative hypothesis: true location shift is not equal to 0
```

مثال (آزمون جفتی) :

```
> wilcox.test(x,y,exact=F,paired=T)

        Wilcoxon signed rank test with continuity correction

data:  x and y
V = 107.5, p-value = 0.6288
alternative hypothesis: true location shift is not equal to 0
```

2. آزمون کولموگروف-اسمیرونوف

برای انجام آزمون ناپارامتری در مورد توزیع یک یا دو متغیر از آزمون کولموگروف-اسمیرونوف استفاده می شود. برای انجام آزمون کولموگروف-اسمیرونوف از تابع `ks.test()` استفاده می کنیم. ساختار کلی این تابع به شکل زیر است.

```
ks.test(x,y,...,alternative,exact)
```

در تابع فوق در قسمت x بردار مشاهدات را وارد می کنیم. در قسمت y در صورتی که بخواهیم آزمون یک نمونه ای انجام دهیم نام توزیع پیوسته مورد نظر و در صورتی که بخواهیم آزمون دو نمونه ای انجام دهیم بردار مشاهدات دوم را وارد می کنیم. در قسمت بعد از y پارامترهای توزیع وارد شده در قسمت y برای آزمون یک نمونه ای را وارد می کنیم. در قسمت های `alternative` و `exact` نیز به ترتیب فرض مقابل و اینکه آیا p-value به صورت دقیق یا تقریبی محاسبه شود را مشخص می کنیم. مقدار پیش فرض گزینه های فوق به صورت زیر است.

```
ks.test(x,y,...,alternative = c("two.sided", "less", "greater"),
exact = NULL)
```

مثال (آزمون یک نمونه ای) :

```
> x <- rnorm(30)
> ks.test(x,"pnorm")

        One-sample Kolmogorov-Smirnov test

data:  x
D = 0.1434, p-value = 0.5221
alternative hypothesis: two-sided
```

مثال (آزمون دو نمونه ای) :

```
> x <- rnorm(30)
> y <- rnorm(25)
> ks.test(x,y)

        Two-sample Kolmogorov-Smirnov test

data:  x and y
D = 0.2333, p-value = 0.3848
alternative hypothesis: two-sided
```

3. آزمون شاپیرو-ویلک

آزمون شاپیرو-ویلک برای بررسی نرمال بودن توزیع مشاهدات به صورت ناپارامتری مورد استفاده قرار مـی گیـرد. برای انجام آزمون شاپیرو-ویلک از تابع `shapiro.test()` استفاده می شود. ساختار کلی این تابع به شکل زیر است.

```
shapiro.test(x)
```

همانطور که مشاهده می شود تنها موردی که باید در تابع وارد کرد بردار مشاهدات است.

مثال :

```
> x <- rnorm(25)
> shapiro.test(x)

        Shapiro-Wilk normality test

data:  x
W = 0.9317, p-value = 0.09503
```

4. آزمون کروسکال-والیس

آزمون کروسکال-والیس برای انجام مقایسه در مورد بیش از دو جامعه به عنوان آنالیز واریانس یک طرفه ناپارامتری مورد استفاده قرار می گیرد. برای انجام آزمون کروسکال-والیس از تابع `kruskal.test()` استفاده می شود. ساختار کلی این تابع به شکل زیر است.

```
kruskal.test(x,g)
```

در این تابع در قسمت x بردار مشاهدات و در قسمت g فاکتور مشخص کننده گروه ها را وارد می کنیم.

مثال :

```
> fact <- rep(1:4,each=5)
> fact <- factor(fact)
> x <- sample(20:30,20,rep=T)
> kruskal.test(x,fact)

        Kruskal-Wallis rank sum test

data:  x and fact
Kruskal-Wallis chi-squared = 0.707, df = 3, p-value = 0.8715
```

رگرسیون

1. مدل های خطی

برای انجام یک تحلیل رگرسیونی در R ابتدا باید مدل را تعیین و معرفی کنیم. برای معرفی مدل های خطی به طور معمول از توابع lm() یا glm() استفاده می شود. تابع lm() دارای کاربردهای وسیعی بوده و به طور کلی در بسیاری از تحلیل های خطی از جمله آنالیز واریانس برای معرفی مدل مورد استفاده قرار می گیرد. برای تعریف یک مدل خطی در تابع lm() از علامت " ~ " در میان نام متغیر وابسته و متغیرهای مستقل استفاده می کنیم. در صورتی که مدل دارای عرض از مبداء صفر باشد از " -1 " برای نشان دادن این مطلب در مدل استفاده می کنیم. مثلا یک مدل خطی ساده با متغیر مستقل x و متغیر وابسته y و عرض از مبداء صفر را به صورت (y~x -1) نشان می دهیم. پس از معرفی مدل با استفاده از تابع lm() از دو تابع summary() و plot() برای انجام تحلیل ها و رسم نمودارهای مدل استفاده می کنیم. این دو تابع زمانی که با تابع lm() مورد استفاده قرار می گیرند با این تابع سازش یافته و خروجی خود را با آن منطبق می سازند. از خروجی تابع summary() می توان در توابعی نظیر resid() ، coef() استفاده کرد. تابع resid() برای مشاهده باقیمانده ها و تابع coef() برای مشاهده ضرایب مورد استفاده قرار می گیرند. همچنین از خروجی تابع lm() نیز علاوه بر توابع summary() و plot() ، می توان در توابع predict() و fitted() استفاده کرد. تابع predict() برای مشاهده مقادیر پیش بینی شده و تابع fitted() برای مشاهده مقادیر برازانده شده مورد استفاده قرار می گیرند. البته بدون استفاده از این توابع نیز می توان مقادیر مورد نظر را با فراخوانی به صورت اجزای یک لیست مشاهده کرد.

مثال (مدل خطی ساده) :

```
> x <- sample(20:50, 20)
> y <- 5+2*x+rnorm(20)
> lm(y~x)

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
      6.034        1.968

> result <- summary(lm(y~x))
> result

Call:
lm(formula = y ~ x)

Residuals:
     Min       1Q   Median       3Q      Max
-2.06729 -0.37419  0.04529  0.60234  2.38873
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.03444    1.11164   5.428 3.71e-05 ***
x            1.96844    0.03124  63.009  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.069 on 18 degrees of freedom
Multiple R-Squared: 0.9955,     Adjusted R-squared: 0.9952
F-statistic:  3970 on 1 and 18 DF,  p-value: < 2.2e-16

> coef(result) ### or use result[['coef']]
            Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 6.034437  1.1116414  5.428402 3.708729e-05
x           1.968442  0.0312408 63.008679 1.444519e-22
> resid(result) ### or use result[['resid']]
          1           2           3           4           5
6
-0.99868842  0.57877341 -0.19974547 -0.34595725  0.06239822 -
0.14850556
          7           8           9          10          11
12
 0.22635293  0.77872693  1.44455478 -0.45889197 -2.06728728
0.02818797
         13          14          15          16          17
18
-1.59416327  0.99571160  0.08357862 -1.45554332  0.67302312 -
0.09639819
         19          20
 0.10514744  2.38872574
> fitted(lm(y~x)) ### or use lm(y~x)[['fitted']]
        1         2         3         4         5         6         7
8
 79.20855  89.24124 105.29355  57.13663  71.18240  47.10393  99.27394
69.17586
        9        10        11        12        13        14        15
16
 49.11047  81.21509  53.12355  63.15624  65.16278  45.09740  61.14970
83.22163
       17        18        19        20
 91.24778  93.25432  95.26086 101.28048
> predict(lm(y~x)) ### or use lm(y~x)[['predict']]
        1         2         3         4         5         6         7
8
 79.20855  89.24124 105.29355  57.13663  71.18240  47.10393  99.27394
69.17586
        9        10        11        12        13        14        15
16
 49.11047  81.21509  53.12355  63.15624  65.16278  45.09740  61.14970
83.22163
       17        18        19        20
 91.24778  93.25432  95.26086 101.28048
```

تنها تفاوت انجام یک تحلیل رگرسیونی خطی ساده و یک تحلیل رگرسیونی خطی چندگانه در R در نحوه معرفی مدل در تابع lm() است. برای معرفی یک مدل خطی چندگانه در تابع lm() کافیست متغیرهای مستقل را با علامت " + " در کنار هم وارد کنیم. بقیه موارد درست شبیه مدل خطی ساده انجام می شود.

مثال (مدل خطی چندگانه) :

```
> x1 <- sample(20:50, 15)
> x2 <- sample(100:200, 15)
> x3 <- sample(35:80, 15)
> y <- 20+10*x1+36*x2+x3
> result <- summary(lm(y~x1+x2+x3))
> result

Call:
lm(formula = y ~ x1 + x2 + x3)

Residuals:
       Min         1Q     Median         3Q        Max
-2.226e-13 -1.601e-13 -2.370e-14  5.958e-14  8.123e-13

Coefficients:
             Estimate Std. Error   t value Pr(>|t|)
(Intercept) 2.000e+01  6.659e-13 3.004e+13   <2e-16 ***
x1          1.000e+01  7.354e-15 1.360e+15   <2e-16 ***
x2          3.600e+01  2.900e-15 1.241e+16   <2e-16 ***
x3          1.000e+00  5.857e-15 1.707e+14   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.88e-13 on 11 degrees of freedom
Multiple R-Squared:     1,      Adjusted R-squared:     1
F-statistic: 5.694e+31 on 3 and 11 DF,  p-value: < 2.2e-16

> result[['coef']] ### or use coef(result)
            Estimate   Std. Error      t value      Pr(>|t|)
(Intercept)       20 6.658508e-13 3.003676e+13 6.996206e-144
x1                10 7.353855e-15 1.359831e+15 4.272794e-162
x2                36 2.900086e-15 1.241342e+16 1.164768e-172
x3                 1 5.856779e-15 1.707423e+14 3.493768e-152
> result[['resid']] ### or use resid(result)
             1              2              3              4              5
 8.122656e-13 -1.768020e-13 -2.370040e-14 -2.225954e-13  1.712964e-13
             6              7              8              9             10
-9.655112e-14 -1.597916e-13 -1.997304e-14  2.331408e-14  1.000102e-13
            11             12             13             14             15
-1.085188e-13 -1.228068e-14 -2.220154e-13 -1.605037e-13  9.584586e-14
```

2. رسم نمودار خط رگرسیون

برای رسم نمودار خط رگرسیون ابتدا نمودار پراکندگی متغیر مستقل در مقابل متغیر وابسته را رسم کرده و سپس با استفاده از تابع `abline()` خط رگرسیون را روی نمودار پراکندگی رسم می کنیم.

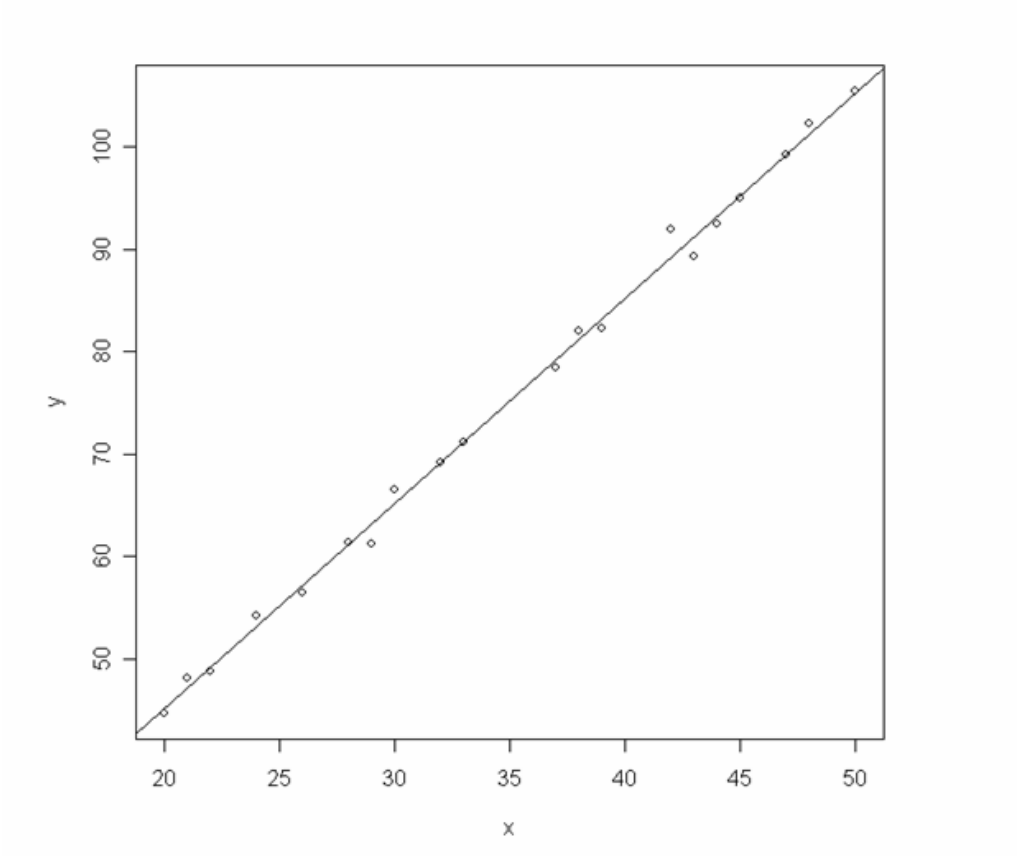
مثال :

```
> x <- sample(20:50, 20)
> y <- 5+2*x+rnorm(20)
> lm(y~x)

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
      4.967        2.007

> plot(x,y)
> abline(lm(y~x))
```

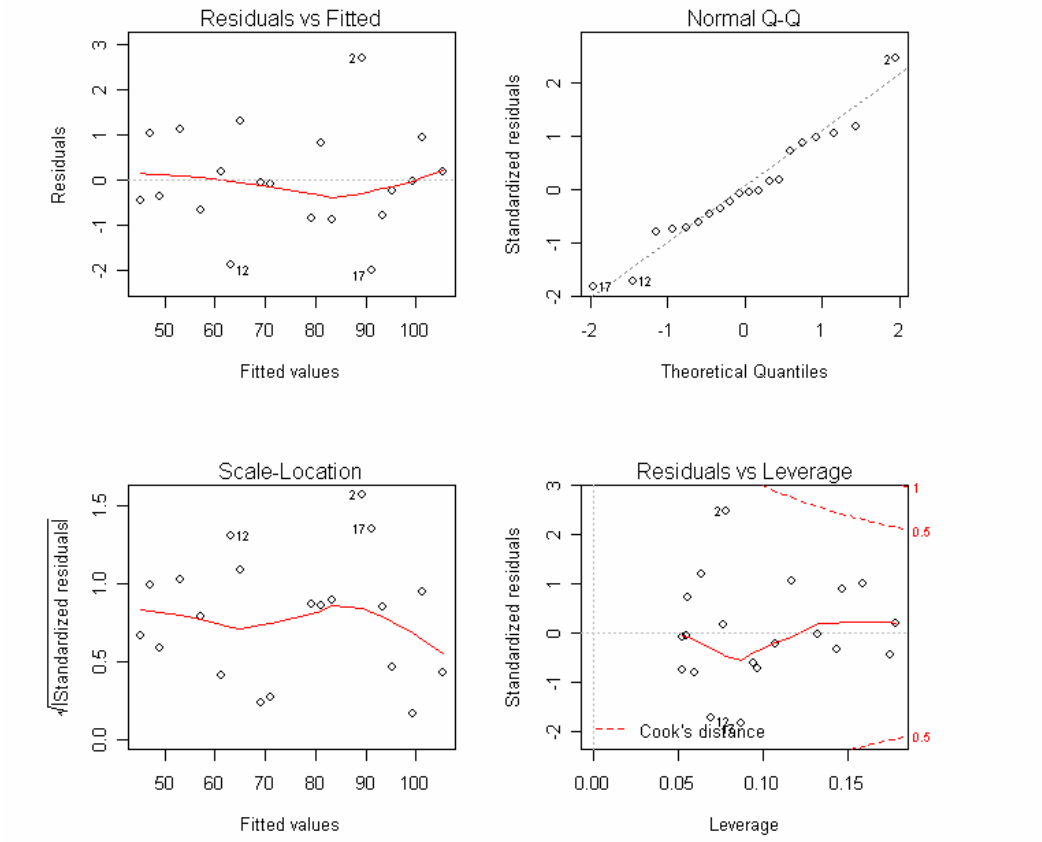


3. بررسی فرض های مدل

برای بررسی برقرار بودن فرض های مدل با استفاده از روش نمـوداری از تـابع `plot()` بـه همـراه تـابع `lm()` استفاده می شود. تابع `plot()` چهار نمودار رسم می کند و آنها را یکی پس از دیگری نمایش مـی دهـد. بـرای نمایش هر چهار نمودار در یک صفحه از تابع `par()` به شکل `par(mfrow=c(2,2))` قبل از تابع `plot()` استفاده می کنیم.

مثال :

```
> par(mfrow=c(2,2))
> plot(lm(y~x))
```



4. آزمون فرض

جدولی که در خروجی تابع summary() ظاهر می شود مقادیر عرض از مبداء و ضرایب را با صفر مقایسه می کند و آزمونی که انجام می دهد بررسی فرض صفر بودن این مقادیر است. در صورتی که بخواهیم عرض از مبداء و ضرایب را با مقادیر مورد نظر خود مقایسه کنیم و آزمونی که در تابع summary() انجام می شود بر اساس مقادیر مورد نظر ما باشد باید زمانی که مدل را در تابع lm() تعیین می کنیم فرض مورد نظر خود را در مدل اعمال کنیم. در تابع lm() عملگرهای ریاضی کاربردهای متفاوتی دارند. برای اینکه عملگرهای ریاضی به کار برده شده در مدل دارای معانی همیشگی خود در محاسبات ریاضی باشند از تابع I() استفاده می کنیم و عملگرها را با استفاده از این تابع به کار می گیریم.

مثال : می خواهیم آزمون کنیم آیا عرض از مبداء برابر 1 است.

```
> x <- sample(10:30,15)
> y <- 1+3*x+rnorm(15)
> lm(y~x)

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
      1.744        2.964

> summary(lm(I(y-1)~x))

Call:
lm(formula = I(y - 1) ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-1.0105 -0.5707 -0.2755  0.4279  1.5420

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.74427    0.72543   1.026    0.324
x            2.96391    0.03616  81.959   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8138 on 13 degrees of freedom
Multiple R-Squared: 0.9981,     Adjusted R-squared: 0.9979
F-statistic:  6717 on 1 and 13 DF,  p-value: < 2.2e-16
```

مثال : در مثال قبل می خواهیم آزمون کنیم آیا ضریب x برابر 3 است.

```
> summary(lm(I(y-3*x)~x))

Call:
lm(formula = I(y - 3 * x) ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-1.0105 -0.5707 -0.2755  0.4279  1.5420

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.74427    0.72543   2.404   0.0318 *
x           -0.03609    0.03616  -0.998   0.3365
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8138 on 13 degrees of freedom
Multiple R-Squared: 0.07117,    Adjusted R-squared: -0.0002754
F-statistic: 0.9961 on 1 and 13 DF,  p-value: 0.3365
```

مثال : در مثال قبل می خواهیم هر دو فرض را هم زمان آزمون کنیم.

```
> summary(lm(I(y-1-3*x)~x))

Call:
lm(formula = I(y - 1 - 3 * x) ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-1.0105 -0.5707 -0.2755  0.4279  1.5420

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.74427    0.72543   1.026    0.324
x           -0.03609    0.03616  -0.998    0.336

Residual standard error: 0.8138 on 13 degrees of freedom
Multiple R-Squared: 0.07117,    Adjusted R-squared: -0.0002754
F-statistic: 0.9961 on 1 and 13 DF,  p-value: 0.3365
```

5. فاصله اطمینان

می توان برای عرض از مبداء و ضرایب فاصله اطمینان تشکیل داد. برای محاسبه فاصله اطمینان از تابع confint() استفاده می کنیم.

مثال :

```
> lm(y~x)

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
      1.744        2.964

> confint(lm(y~x))
                2.5 %   97.5 %
(Intercept) 0.1770625 3.311473
x           2.8857802 3.042033
```

6. مقایسه مدل ها

برای انجام آزمون در مورد اینکه آیا دو مدل با هم تفاوت معنی داری دارند یا نه از تابع anova() به همراه تابع lm() استفاده می کنیم. این تابع برای مقایسه دو مدل یک جدول آنالیز واریانس تشکیل می دهد.

مثال :

```
> x1 <- sample(10:30,10); x2 <- sample(100:130,10)
> y <- 5+9*x1+x2+rnorm(10)
> m1 <- lm(y~x1+x2); m2 <- lm(y~x2)
> anova(m1,m2)
Analysis of Variance Table

Model 1: y ~ x1 + x2
Model 2: y ~ x2
  Res.Df      RSS Df Sum of Sq     F    Pr(>F)
1      7      3.9
2      8  28532.3 -1  -28528.4 51829 8.329e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> m3 <- lm(y~x1)
> anova(m1,m3)
Analysis of Variance Table

Model 1: y ~ x1 + x2
Model 2: y ~ x1
  Res.Df     RSS Df Sum of Sq      F    Pr(>F)
1      7    3.85
2      8  423.04 -1   -419.19 761.56 2.106e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

طرح آزمایش ها

1. آنالیز واریانس یک طرفه

برای انجام آنالیز واریانس به دو روش می توان عمل کرد. روش اول آن است که ابتدا مدل را بـا اسـتفاده از تـابع `lm()` معرفی کرده و سپس با استفاده از تابع `anova()` به آنالیز واریانس بپردازیم. روش دیگر آن است که ابتدا مدل را با استفاده از تابع `aov()` معرفی کرده و سپس با استفاده از تابع `summary()` به آنالیز واریانس بپردازیم. نتیجه حاصل از هر دو روش یکسان خواهد بود، اما شیوه متداول استفاده از روش دوم است. معرفی مدل در هر دو روش به یک شکل انجام می شود. برای معرفی مدل در آنالیز واریانس یک طرفه از یک بردار پاسخ و یک فـاکتور برای نشان دادن تیمارها استفاده می شود.

مثال (روش اول) :

```
> resp <-
c(21,25,30,25,29,24,29,20,27,22,28,26,29,21,28,21,26,20,20,29)
> fact <- rep(1:5,each=4)
> fact <- factor(fact)
> anova(lm(resp~fact))
Analysis of Variance Table

Response: resp
          Df  Sum Sq Mean Sq F value Pr(>F)
fact       4  10.000   2.500  0.1589 0.9559
Residuals 15 236.000  15.733
```

مثال (روش دوم) :

```
> summary(aov(resp~fact))
            Df  Sum Sq Mean Sq F value Pr(>F)
fact         4  10.000   2.500  0.1589 0.9559
Residuals   15 236.000  15.733
```

برای مشاهده باقیمانده ها و مقادیر برازانده شده مقادیر `resid` و `fitted` را همانند یک لیست از مدل فراخوانی می کنیم.

مثال :

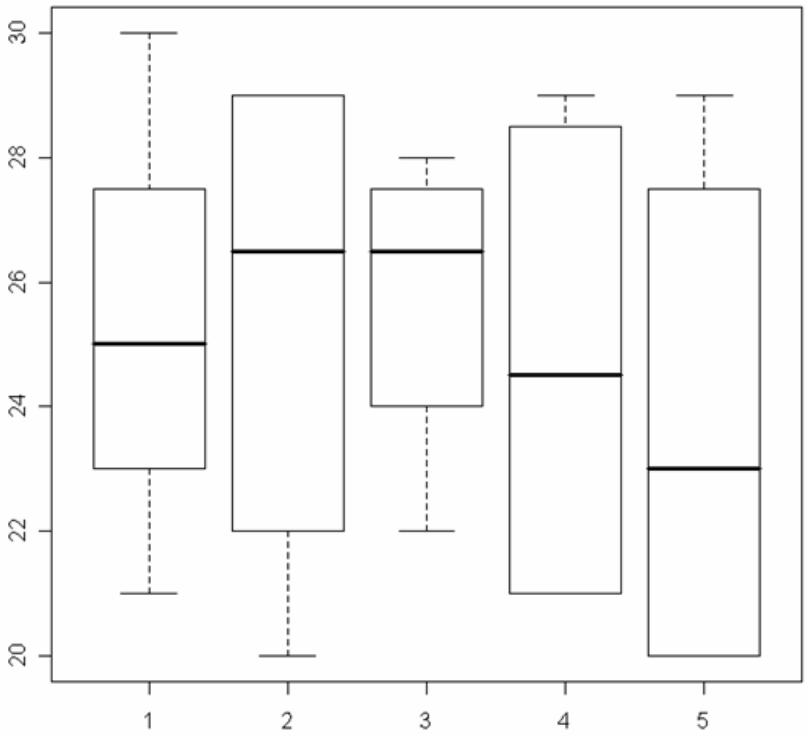
```
> mod <- aov(resp~fact)
> mod[['resid']]
    1     2     3     4     5     6     7     8     9    10    11
12    13
-4.25 -0.25  4.75 -0.25  3.50 -1.50  3.50 -5.50  1.25 -3.75  2.25
0.25  4.25
   14    15    16    17    18    19    20
-3.75  3.25 -3.75  2.25 -3.75 -3.75  5.25
```

```
> mod[['fitted']]
    1     2     3     4     5     6     7     8     9    10    11
12    13
25.25 25.25 25.25 25.25 25.50 25.50 25.50 25.50 25.75 25.75 25.75
25.75 24.75
   14    15    16    17    18    19    20
24.75 24.75 24.75 23.75 23.75 23.75 23.75
```

برای رسم نمودار جعبه ای از تابع `boxplot()` استفاده می کنیم.

مثال :

```
> boxplot(resp~fact)
```

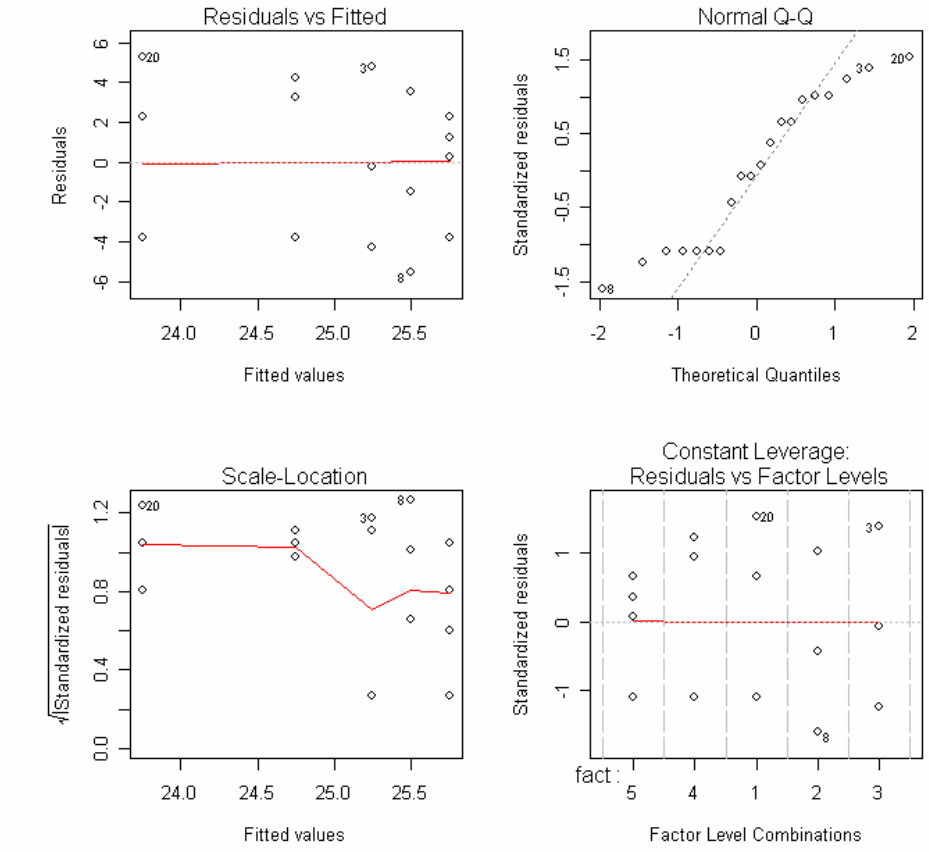


2. بررسی فرض های مدل

برای بررسی فرض های مدل از طریق رسم نمودار از تابع `plot()` به همراه `aov()` استفاده می کنیم. تابع `plot()` چهار نمودار رسم می کند و آنها را یکی پس از دیگری نمایش می دهد. برای نمایش هر چهار نمودار در یک صفحه از تابع `par()` به شکل `par(mfrow=c(2,2))` قبل از تابع `plot()` استفاده می کنیم.

مثال :

```
> par(mfrow=c(2,2))
> plot(aov(resp~fact))
```



3. آزمون توکی

برای انجام آزمون توکی و انجام مقایسه های چندگانه از تابع TukeyHSD() استفاده می شود.

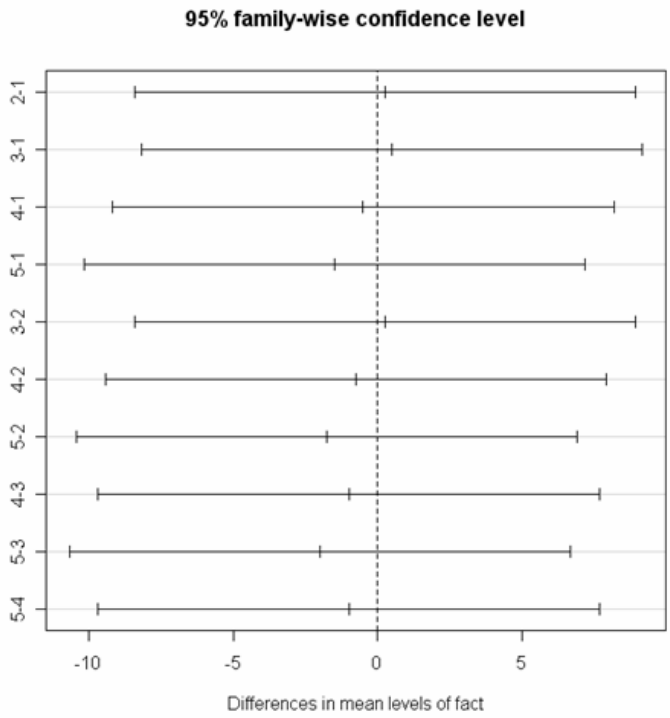
مثال :

```
> TukeyHSD(aov(resp~fact))
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = resp ~ fact)

$fact
     diff       lwr     upr     p adj
2-1  0.25  -8.41088 8.91088 0.9999839
3-1  0.50  -8.16088 9.16088 0.9997455
4-1 -0.50  -9.16088 8.16088 0.9997455
5-1 -1.50 -10.16088 7.16088 0.9821931
3-2  0.25  -8.41088 8.91088 0.9999839
4-2 -0.75  -9.41088 7.91088 0.9987412
5-2 -1.75 -10.41088 6.91088 0.9688634
4-3 -1.00  -9.66088 7.66088 0.9961482
5-3 -2.00 -10.66088 6.66088 0.9502519
5-4 -1.00  -9.66088 7.66088 0.9961482


> plot(TukeyHSD(aov(resp~fact)))
```



4. طرح بلوک های تصادفی و مربع لاتین

برای اجرای یک طرح بلوک های تصادفی یا مربع لاتین و یا حتی طرح هایی با مولفه های بیشتر کافیست به ازای هر مولفه فاکتوری که معرفی کننده آن مولفه باشد در معرفی مدل وارد کنیم. برای اضافه کردن مولفه ها بـه مـدل از علامت " + " استفاده می شود.

مثال (طرح بلوک های تصادفی) :

```
> resp <- c(21,25,30,25,29,24,29,20,27,22,28,26,29,21,28,21)
> fact <- factor(rep(1:4,each=4))
> block <- factor(rep(1:4,times=4))
> summary(aov(resp~fact+block))
            Df Sum Sq Mean Sq F value  Pr(>F)
fact         3  2.188   0.729  0.0825 0.96789
block        3 95.687  31.896  3.6080 0.05862 .
Residuals    9 79.563   8.840
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

مثال (طرح مربع لاتین) :

```
> latin <-
c("A","B","C","D","D","A","B","C","C","D","A","B","B","C","D","A")
> latin <- factor(latin)
> summary(aov(resp~fact+block+latin))
            Df Sum Sq Mean Sq F value  Pr(>F)
fact         3  2.188   0.729  0.0933 0.96098
block        3 95.687  31.896  4.0827 0.06744 .
latin        3 32.688  10.896  1.3947 0.33256
Residuals    6 46.875   7.813
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

5. طرح های عاملی

تفاوت طرح های عاملی با طرح های قبل نیز در نحوه معرفی مدل است. برای اجرای طرح های عاملی کافیست به ازای هر عامل فاکتوری را برای معرفی آن عامل در مدل وارد کنیم. برای وارد کردن عامـل هـا در مـدل از علامـت " * " استفاده می شود.

مثال (طرح دو عاملی) :

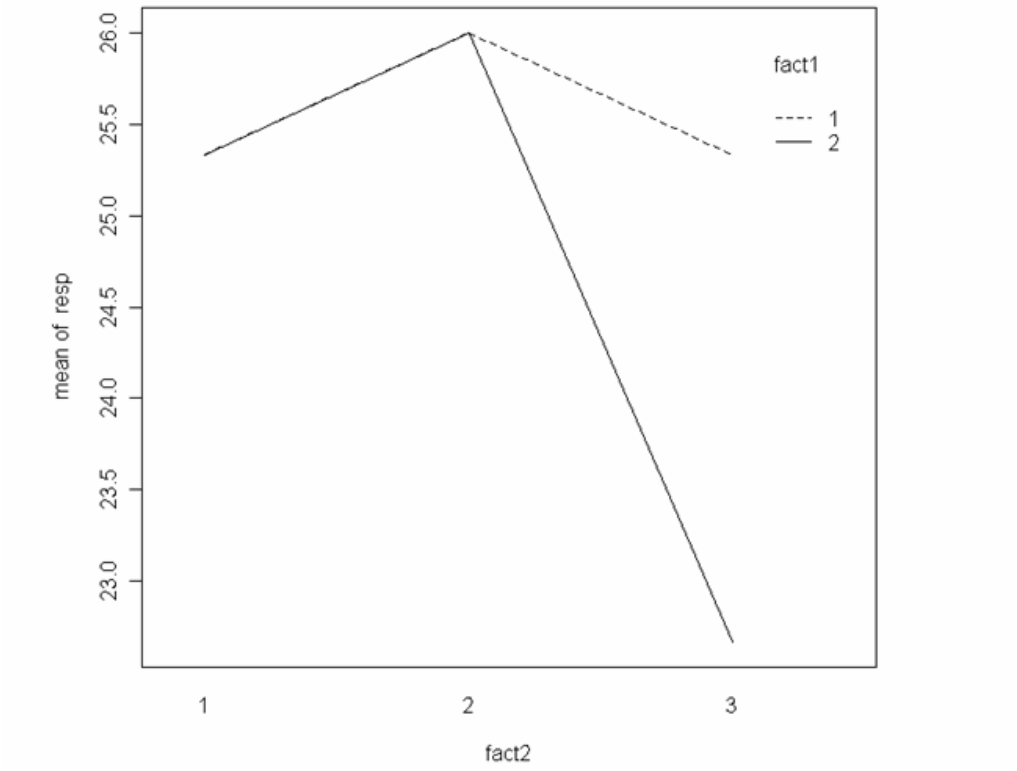
```
> resp <- c(21,25,30,25,29,24,29,20,27,22,28,26,29,21,28,21,25,22)
> fact1 <- factor(rep(1:2,each=9))
> fact2 <- factor(rep(1:3,each=3,times=2))
> summary(aov(resp~fact1*fact2))
            Df  Sum Sq Mean Sq F value Pr(>F)
fact1        1   3.556   3.556  0.2591 0.6200
fact2        2  12.444   6.222  0.4534 0.6459
fact1:fact2  2   7.111   3.556  0.2591 0.7759
Residuals   12 164.667  13.722
```

6. نمودار اثر متقابل

برای رسم نمودار اثر متقابل از تابع `interaction.plot()` استفاده می شود. در ایـن تـابع ابتـدا دو فـاکتور معرفی کننده عوامل، سپس متغیر پاسخ، و پس از آن نوع نمایش نمودار را مشخص می کنیم.

مثال :

```
> interaction.plot(fact2,fact1,resp,type="l")
```



منابع

1. A short introduction to R, European Educational Programme in Epidemiology, 2005

2. An Introduction to R: Software for StatisticalModelling & Computing, Petra Kuhnert and Bill Venables

3. An Introduction to R; Notes on R: A Programming Environment for Data Analysis and Graphics, W. N. Venables, D. M. Smith and the R Development Core Team

4. An R companion to " Experimental Design ", Vikneswaran

5. R for Statistics, By Example: A Workshop for the Ecological Society America, Hank Stevens

6. simpleR; Using R for Introductory Statistics, John Verzani

7. Statistics with R, Vincent Zoonekynd