
بسم الله الرحمن الرحيم

آموزش نرم افزار *splus*

چکیده

با توجه به اینکه درسهایی که در رشته ی آمار ارائه می شوند قابلیت قابل توجهی در طرح های پژوهشـــی و تحقیقاتی در زمینه‌های گوناگون دارند، یادگیری فنون وتکنیک های مربوط به این درسها از اهمیت ویژه‌ای برخوردار است، اما نکته‌ی مهمتری که باید به آن توجه کرد بالا بردن سرعت و دقت در نتیجه گیری و استنباط است. خوشبختانه در حال حاضر نرم افزارهای زیادی مانند *spss*، *s_plus*، *sas*، min*itab* و... موجود می باشند که در جهت افزایش سرعت و دقت در نتیجه گیری کمک شایان ذکری به محقق می کنند. از این میان ما به بیان برخی تکنیک ها و قابلیت های نرم افزار *s_plus* پرداخته‌ایم، که می تواند در زمینه ی درســـهای کاربردی رشته‌ی آمار به دانشجویان کمک کند.

ابتدا برخی مفاهیم اساسی که در استفاده ازاین نرم افزار مفید و مؤثر هستند را بیان کرده ایم و سپس چگونگی کارایی این نرم افزار را در درسهایی مانند طرح آزمایشات، رگرسیون، سریهای زمانی و ... شـــرح داده ایم. در آخر در بخش ضـــمیمه تعدادی برنامه‌ی جالب را در زمینه‌ی ریاضیات و آماربرای یادگیری بهتر ارائه داده ایم.

فصل اول

ساختار داده‌ها (1)

بردار

ساده‌ترین نوع ساختار داده در $S+$ بردار است. بردار مجموعه‌ای مرتب از مقادیر عددی، کاراکتری و یا منطقی است.

ساختن بردار

روشهای مختلفی برای ساختن یک بردار وجود دارد. معمولی ترین روش استفاده از تابع $c( )$ می‌باشد. روشی دیگر استفاده از عملگر $(:)$ است، که عملگر دنباله خوانده می‌شود و یک سری داده را به ترتیب می‌گیرد. با استفاده از تابع $rep( )$ نیز می‌توان یک مقدار یا یک بردار را تکرار کرد. با تابع $seq( )$ نیز می‌توان یک بردار تولید کرد. با دستور $scan( )$ نیز می توان تا جایی که بخواهیم مقادیری را وارد یک بردار کنیم. این توابع را به ترتیب به همراه مثال در زیر شرح می‌دهیم.

تابع $c( )$

با استفاده از این تابع می‌توان مقادیر دلخواه را با هم ترکیب کرده، یک بردار ساخت وبا عملگر($<-$) یا ($=$) این بردار را به اسم دلخواهی نسبت داد.

مثال1:

```
x <-c(1,2,3,4)
x↵
[1] 1 2 3 4
y <-c(6,7)
c(x,y,x)↵
[1] 1 2 3 4 6 7 1 2 3 4
```

توجه کنیم که اگر یک بردار دارای اسم نباشد، در قسمت‌های دیگر یک برنامه امکان استفاده از این بردار را نداریم. وهمین طور اگر در معرفی یک متغیر یا یک بردار از حروف کوچک (بزرگ) استفاده کردیم، برای فرا خوانی آن در جریان برنامه نیز باید از حروف کوچک (بزرگ) استفاده کنیم.

نکته1 :

لازم نیست مقداری که یک متغیر می‌پذیرد حتماً عددی باشد، می‌تواند مقدار اسمی نیز باشد. در این صورت هر مقدار اسمی را باید در داخل علامت (" ") قرارداد تا ارزش اسمی داشته باشد.

مثال 2:

```
y < –c("a","b","d")
y↵
[1] a b d
```

عملگر (:)

برای ساختن دنباله‌ای از مقادیر، از این عملگر استفاده می‌کنیم که عملگر دنباله خوانده می‌شود. فرم کلی به صورت زیر است.

انتهای دنباله : ابتدای دنباله

مثال 3:

```
a < –2 : 5
a↵
[1] 2 3 4 5
```

مشاهده می‌کنیم که در خروجی اعداد 2 تا 5 را به ترتیب به ما نشان می‌دهد.

تابع *rep( )*

برای تکرار یک عدد یا یک بردار وقرار دادن آن در یک بردار از این تابع استفاده می‌کنیم. فرم کلی آن به صورت زیر است.

rep (تعداد تکرار ، عبارت)

مثال 4:

```
b < –rep(NA,3)
b↵
[1] NA NA NA
```

اگر تعداد تکرارها خود یک بردار باشد و طول این بردار با طول بردار مقادیر برابر باشد، هر مقدار از بردار مقادیر به تعداد متناظرش در بردار تکرارها، تکرار می‌شود.

مثال 5:

```
e < –rep(1: 4,rep(3,4))
e↵
[1] 111 2 2 2 3 3 3 4 4 4
f < –rep(1: 4,1: 4)
f↵
[1] 1 2 2 3 3 3 4 4 4 4
```

تابع *seq( )*

ازاین تابع می‌توان به جای عملگر دنباله اســتفاده کرد. ولی تفاوتشــان در این اســت که در تابع seq( ) می‌توان طول بردار یا قدر نسبت دنباله وآغاز دنباله را مشـخص کرد. حالتهای مختلف استفاده از این دستوررا با مثال زیر توضیح می‌دهیم.

مثال 6:

```
seq(5)↵
[1] 1 2 3 4 5
```

مانند دستور $1:5$ عمل می‌کند.

```
seq(−1,1,0.5)↵
[1] −1  −0.5 0 0.5 1
```

از عدد -1 تا عدد1 دنباله‌ای با قدر نسبت 0.5 تولید می‌کند.

```
seq(−1,1,length = 5)↵
[1]  −1  −0.5 0 0.5 1
```

5عدد بین اعداد -1 تا 1 را به ما می‌دهد.

```
seq(1,by = 0.05,length = 10)↵
[1] 1.00 1.05 1.10 1.15 1.20 1.25 1.30 1.35 1.40 1.45
```

ده عدد از دنباله‌ای تولید می‌کند که از عدد 1 شروع می‌شود و دارای قدر نسبت 0.05 است.

دستور scan( )

با اجرای این دستور پنجره‌ای باز می‌شود که مرتب از ما داده می‌گیرد و زمانی متوقف می‌شود که دو بار پشت سرهم کلید اینتر(↵) را فشار دهیم.

مثال7:

به دستور زیر توجه کنید.

```
scan( )↵
1: 2↵
2:  3↵
3: ↵ ↵
[1] 2 3
```

با این دســتور نرم افزار مرتب از ما داده می‌گیرد ولی می‌توانیم از همان ابتدا مشــخص کنیم که چند داده از ما بگیرد. به مثال زیر توجه کنید.

```
scan(n = 3)↵
1: 2↵
2: 3↵
3: 4↵
[1] 2 3 4
```

انتخاب زیر مجموعه‌هایی از بردار

انتخاب زیر مجموعه‌ای از یک بردار را با مثال شرح می‌دهیم.

مثال 8:

```
x < –c(5,14,8,9,5)
x↵
[1] 5 14 8 9 5
```

برای نمایش عنصری از یک بردار بنویسید:

[اندیس عنصر] نام بردار

مثلاً دستور زیر عنصر چهارم از بردار x را نمایش می‌دهد.

```
x[4]↵
[1] 9
```

برای انتخاب عنصر دوم و پنجم بردار x عبارت زیر را به کار برید.

```
x[c(2,5)]↵
[1] 14 5
```

اگر بخواهیم همه‌ی عناصر بردار به جز عناصر دوم و پنجم را انتخاب کنیم، قبل از تابع $c(\ )$ از علامت منفی استفاده می‌کنیم.

```
x[–c(2,5)]↵
[1] 5 8 9
```

با استفاده از عبارات منطقی نیز می‌توان زیرمجموعه‌ای از یک بردار را انتخاب کرد.

```
x[x > 8]↵
[1] 14 9
```

گاهی اوقات بعد از مشاهده‌ی بردار متوجه می‌شویم که عددی را اشتباه وارد کرده‌ایم. در این صورت با انتخاب عنصری که آن را اشتباه وارد کرده‌ایم ونسبت دادن مقدار صحیح به آن می‌توان اشتباه را جبران کرد. به مثال‌های زیر توجه کنید.

```
x[1:3] < −2↵
x↵
[1] 2 2 2 9 5
```

می‌خواهیم به جز عنصر اول بقیه‌ی عناصر را مقدار صفر قرار دهد.

```
x[−1] < −0↵
x↵
[1] 5 0 0 0 0
```

مثال 9:

فرض کنید یک بردار داریم مثل همان بردار مثال‌های قبل، می‌خواهیم عدد 12را به عنوان عنصر سوم به این بردار اضافه کنیم .

```
x < −c(x[1:2],12,x[3:5])↵
x↵
[1] 5 14 12 8 9 5
```

چند تابع تعریف شده در $s+$

1) تابع *length*()

این تابع طول برداری که نام آن را در داخل پرانتزها می‌نویسیم به ما می‌دهد. برای بردار مثال 8 داریم:

```
length(x)↵
5
```

2) تابع *rm*()

ساختار داده‌ای را که نام آن به عنوان شناسه‌ی تابع ذکر شود، حذف می‌کند.

3) تابع *q*()

برای خروج از $s+$ از این تابع استفاده می‌شود.

4) تابع *help*()

برای استفاده از راهنمائی‌های $s+$ از این تابع یا عملگر" ؟ " استفاده می‌شود. شناسه‌ی این تابع ممکن است نام یک تابع، یا یک کاراکتر باشد.

```
> help("کاراکتر")                    > help( نام تابع )
> ?"کاراکتر"                         > ? نام تابع
```

5) تابع *ls*()

لیست ساختار داده‌های موجود در زیر فهرست *Data* را نمایش می‌دهد.

6) تابع objects()

لیست ساختار داده‌های موجود در تمام زیر فهرست‌های +s را نمایش می‌دهد.

7) تابع sink()

این تابع خروجی یک برنامه یا عبارت را در یک فایل متن ذخیره می‌کند.

مثال 10:

```
sink("list.dir")↵
ls( )↵
sink( )↵
```

در این مثال خروجی تابع ls( ) در فایل list.dir ذخیره خواهد شد. فرا خوانی مجدد تابع sink( ) به عملیات خاتمه می‌دهد.

نکته 2:

برای مشخص کردن حالت یک بردار از تابع mode( ) استفاده می‌شود. وتابع attributes( ) ویژگی ساختار داده وتابع attr( ) یک ویژگی خاص ساختار داده را نشان می‌دهد .

```
c(1,2,3,4) ↵
mode(x) ↵
[1] 'numeric'
attributes(x) ↵
list( )
attr(x,' length') ↵
[1] 4
```

توجه کنیم که عناصر یک بردار همه باید دارای یک حالت باشند. مثلاً ساختن برداری شامل مقادیر عددی و منطقی غیر ممکن است.

نکته‌ی دیگرکه باید به آن توجه کنیم این است که برای ساختن یک بردار با مقادیر نام گذاری شده از تابع names( ) استفاده می‌کنیم. به مثال زیر دقت کنید.

```
names(x) < –letters[1 : 4]↵
x↵
a b c d
1 2 3 4
```

در اینجا مقادیر بردار x با اولین تا چهارمین حروف لاتین که با بردار $letters[1:4]$ به $names(x)$ نسبت داده شده است، نام گذاری شده اند.

نکته 3 :

از تابع [...] letters برای نمایش حروف لاتین کوچک استفاده می‌شود. اگر به این حروف به ترتیب الفبا، اعداد 1تا 26 را نسبت دهیم، در داخل علامت [ ] می‌توانیم اندیس حروفی را که می‌خواهیم بنویسیم. اگر این دستور به صورت [...]LETTERS به کار رود، حروف بزرگ لاتین را نمایش می‌دهد.

توابع مقدماتی ریاضی در S+

تابع	شرح
abs( )	قدرمطلق یک عدد
ceiling( )	اولین عدد صحیح بزرگتر از عدد
cumsum( )	جمع تجمعی اعداد یک بردار
cumprod( )	ضرب تجمعی اعداد یک بردار
exp( )	تابع نمایی
prod( )	عناصر بردار را در هم ضرب می کند
sum( )	عناصر بردار را با هم جمع می کند
floor( )	جزء صحیح یک عدد
gamma( )	تابع گاما
log( )	لگاریتم در مبنای عدد نپر
log10( )	لگاریتم در مبنای عدد 10
max( )	ماکزیمم یک سری عدد
min( )	می نیمم یک سری عدد
mean( )	میانگین
median( )	میانه
rank( )	اعداد را رتبه بندی می کند
round( )	گرد کردن اعداد
sort( )	مرتب کردن اعداد به صورت صعودی
sqrt( )	ریشه ی دوم عدد

summary( )	یافتن میانه، چارک اول وسوم و min,max
trunc( )	نزدکترین عدد صحیح به عدد
var( )	واریانس
sin( ),cos( ),tan( )	سینوس، کسینوس و تانژانت
*a*sin( ),*a*cos( ),*a*tan( )	توابع مثلثاتی معکوس
sinh( ),cosh( ),tanh( )	توابع مثلثاتی هیپر بولیک
*a*sinh( ),*a*cosh( ),*a*tanh( )	وارون توابع مثلثاتی هیپر بولیک

در ادامه یک سری از توابع مهم دیگری را که در S+ کاربرد دارند معرفی می‌کنیم.

1) تابع *data.frame*( )

اگر بخواهیم در S+ مدلهای طرح آزمایشات یا رگرسیون را براورد کنیم، با این دستور داده‌ها را در کنار هم مرتب می‌کنیم. مثلاً اگر در یک مدل رگرسیون متغیر x یک متغیر کنترلی و y متغیر پاسخ باشد، بعد از وارد کردن اطلاعات برای برازش مدل این دو متغیر را به صورت زیر در کنار هم قرارمی‌دهیم.

$far <- data.frame(x, y)$

در طرح آزمایشات نیز ابتدا داده‌های مربوط به فاکتورها و خود متغیر را وارد می‌کنیم و سپس با این دستور آنها را در کنار هم قرار می‌دهیم تا داده‌های طرح آزمایشات را بسازیم.

مثال 11:

```
x <-c(1,2,3)
y <-c(12,13,16)
data.frame(x, y)↵
x  y
1  12
2  13
3  16
```

2) تابع مشتق *D*( )

برای مشتق گیری از یک عبارت ریاضی از این تابع استفاده می‌کنیم. فرم کلی به صورت زیر است.

D (exp*ression*(تابع ریاضی),"متغیری که نسبت به آن مشتق می‌گیریم")

مثال 12:

```
D(expression(3 * x^2), "x")↵
[1] 3 * (2 * x)
D(expression(exp(x^2)), "x")↵
[1] exp(x^2) * (2 * x)
```

3) تابع integrate( )

از این تابع برای محاسبه‌ی انتگرال توابع ریاضی استفاده می‌شود. که دارای شناسه‌های زیر است.

integrate(تابع, کران پایین انتگرال, کران بالای انتگرال, subdevission = 100)

در قسمت آخر تعداد مستطیل‌هایی که در محاسبه‌ی انتگرال ریمان به کار می روند، را قرار می‌دهیم.

مثال 13:

```
f <–function(x) { y <– exp(x); return(y) }
integrate(f,0,1,1000)↵
[1] 22025.47
```

ابتدا تابع را تعریف کردیم وبعد انتگرال را از صفر تا 1 برای آن محاسبه کردیم. در این مثال تعداد مستطیل‌ها را برای دقت بیشتر 1000 در نظر گرفتیم .

4) تابع polyroot( )

این تابع ریشه‌های چند جمله‌ای $a_k z^k + ... + a_1 z + a_0$ را محاسبه می‌کند. شناسه‌ی این تابع بردار ضرایب چند جمله ای $(a_0, a_1, ..., a_k)$ است .

مثال 14 :

معادله‌ی $z^2 + 5z + 6 = 0$ با استفاده از این تابع به صورت زیر حل می شود.

```
polyroot(c(6,5,1))↵
[1] –2 + 2.58e – 0.26i   –3 – 2.58e – 0.26i
```

عملگرهای منطقی و حسابی

در splus گزاره‌های درست را با کاراکتر T وگزاره‌های نادرست را با کاراکتر F می‌شناسیم. عملگرهای حسابی ومنطقی را به ترتیب درجدول زیر تعریف می‌کنیم .

عملگر	شرح	مثال
+	جمع	> 3 + 2↵ [1]5
-	تفریق	> 3 – 2↵ [1]1
*	ضرب	> 3 * 2↵ [1]6
/	تقسیم	> 3 / 2↵ [1]1.5
^	توان وریشه	> 4^(1/2)↵ [1]2
%|%	خارج قسمت تقسیم	> 24.5%|%3.2↵ [1]7
%%	باقی مانده ی تقسیم	> 24.5%%3.2↵ [1]2.1
%o%	ضرب بیرونی دو بردار	
>	بزرگتر	....
<	کوچکتر	...
<=	کوچکتریا مساوی	...
>=	بزرگتریا مساوی	...
==	مساوی	...
!=	نامساوی	...
&	"و" منطقی	> *T* & *F*↵ *F*
|	"یا"منطقی	> *T* | *T*↵ *T*
!	نقیض	> *x* < –!*T*; *x*↵ *F*

به مثال‌های زیر توجه کنید. که در محیط *command* نوشته و اجرا شده‌اند.

2 >= 3↵
F
!*T*↵
F
T & *F*↵
F
T | *F*↵
T

نکته 4 :

گزاره‌های منطقی بیشتر در حلقه‌ها به عنوان شرط پایانی حلقه استفاده می‌شوند. که در فصل سوم به کاربرد آنها پرداخته‌ایم.

تمرینهای فصل اول

تمرین 1 :

فرض کنید $X_1, X_2, \ldots, X_n$ یک نمونه‌ی تصادفی از یک جامعه باشد برنامه‌ای بنویسید که ضرایب چولگی وبرجستگی که به صورت زیر هستند را محاسبه کند.

ضریب چولگی: $g_1 = \sqrt{n}\frac{\sum_{i=1}^{n}(x_i-\bar{x})^3}{\left(\sum_{i=1}^{n}(x_i-\bar{x})^2\right)^{3/2}}$

ضریب برجستگی: $g_2 = n\frac{\sum_{i=1}^{n}(x_i-\bar{x})^4}{\left(\sum_{i=1}^{n}(x_i-\bar{x})^2\right)^2} - 3$

با استفاده از توابع مقدماتی ریاضی و عملگرهای حسابی برای محاسبه‌ی این دو ضریب دستوراتی را به صورت زیر به کار می‌بریم.

$x <- c(x_1, x_2, ..., x_n)$

$a_1 <- sum((x - mean(x))^3)$

$a_2 <- sum((x - mean(x))^2)$

$a_3 <- sum((x - mean(x))^4)$

$n <- length(x)$

$g_1 <- sqrt(n) * (a_1 / a_2 ^ (3/2))$

$g_2 <- n * (a_3 / a_2 ^ 2) - 3$

فصل دوم

ساختار داده‌ها (2)

ماتریس، آرایه و لیست

ماتریس

برای ساختن ماتریس از تابع matrix( ) استفاده می‌کنیم. بدین منظور مقادیر را با تابع c( ) یا به هر شکل قابل قبولی به صورت بردار در می‌آوریم و با تعیین تعداد سطرها و ستونها آنها را گروه بندی می‌کنیم. اگر تعداد عناصر بردار از تعدادی که ماتریس نیاز دارد (یعنی از تعدادسطرها*تعداد ستونها) کمتر باشد، مقادیر به طور متناوب تکرار می‌شوند تا زمانی که ماتریس کامل شود. فرم کلی به صورت زیر است.

اسم ماتریس <- matrix(داده ها , تعداد سطرها , تعداد ستونها, byrow = F)

نوشتن تعداد ستونها الزامی نیست. byrow = F که حالت قراردادی است مشخص می‌کند که داده‌ها به صورت ستونی در ماتریس گذاشته شوند. یعنی اول تمام سطرهای ستون اول، سپس تمام سطرهای ستون دوم وبالاخره سطرهای ستون آخر پرمی شود. byrow = T به عکس عمل می‌کند.

مثال1:

به تفاوت دو ماتریس زیر توجه کنید.

```
A <-matrix(1:12,3,4)↵
A↵
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12

B <-matrix(1:12,3,byrow=T)↵
B↵
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
```

انتخاب ریز مجموعه‌ای از یک ماتریس

انتخاب یک زیر مجموعه از یک ماتریس همانند انتخاب یک زیر مجموعه از بردار می‌باشد، با این تفاوت که ماتریسها دارای دو بعد هستند. برای انتخاب عنصری که در سطر iام و ستون jام قرار دارد از عبارت زیر استفاده می‌کنیم.

[i,j] نام ماتریس

برای انتخاب سطر i ام ویا ستون j ام به صورت زیر عمل می‌کنیم.

[$i,$] نام ماتریس : انتخاب سطر i ام

[$,j$] نام ماتریس : انتخاب ستون j ام

برای انتخاب چند سطر یا ستون از یک ماتریس می‌توان از تابع $c( )$ استفاده کرد. برای حذف چند سطر یا ستون علامت " - " را قبل از تابع $c( )$ به کار می‌بریم.

مثال2:

در مورد ماتریس مثال قبل داریم:

```
A[1,2]↵
4
A[c(1,2),]↵
1  4  7  10
2  5  8  11

A[2,c(2,3)]↵
5 8

A[,−c(2,3)]↵
1  10
2  11
3  12

A[A > 8]↵
9  10  11  12
```

درقسمت آخرمثال بالا مشاهده می کنیم که می‌توان از عملگرهای منطقی نیز برای انتخاب استفاده کرد. توجه کنید که کدام انتخابها به صورت ماتریس و کدام انتخابها به صورت بردار به ما نشان داده می‌شوند.

ترکیب ماتریسها

برای ترکیب ماتریسها از دو تابع $cbind( ),rbind( )$ استفاده می‌کنیم. تابع $cbind( )$ ماتریسها (بردارها) را به صورت ستونی وتابع $rbind( )$ آنها را به صورت سطری ترکیب می‌کند.

مثال3:

```
rbind(c(1,2,3,4),5:8)↵
1  2  3  4
5  6  7  8

cbind(c(1,2,3,4),5:8)↵
1  5
2  6
3  7
4  8
```

نکته 1 :

به منظوراستفاده از cbind( )، تعداد سطرهای ماتریسها باید برابر باشند و به منظور استفاده از rbind( ) تعداد ستونها باید برابر باشند. در مورد بردارها نیز چنین است.

ویژگی‌های ماتریس

ویژگی‌های یک ماتریس عبارتند از بعد، حالت و طول آن ماتریس که به ترتیب توسط توابع length(),mode(),dim()به دست می‌آیند.

تابعdim() یک بردار به ما می‌دهد که درایه‌ی اول آن تعداد سطرها ودرایه‌ی دوم آن تعداد ستونهای ماتریس است.

مثال4:

ماتریس مثال قبل را در نظر بگیرید.

```
dim(A)↵
[1] 3 4
```

در این مثال داریم : تعداد سطرها = dim(A)[1] و تعداد ستونها = dim(A)[2] .

```
mode(A)↵
[1] "numeric"
```

نکته 2 :

حالت یک ماتریس می‌تواند اسمی باشد، در این صورت هر مقدار اسمی باید در داخل علامت (" ") قرار بگیرد. به مثال زیر توجه کنید.

```
G <-matrix(c("a","b","d","e"),2)↵
     [,1] [,2]
[1,]  a    d
[2,]  b    e
```

تابع length( ) تعداد درایه‌های ماتریس را به ما می‌دهد.

محاسبات ماتریسی

بیشتر محاسبات برداری و ماتریسی در S-plus به صورت عنصر به عنصر صورت می‌گیرد. برای انجام این گونه محاسبات باید ابعاد بردارها با هم وابعاد ماتریسها با هم برابر باشند. در بردارها اگر طول یک بردار از دیگری کوچکتر باشد، عناصر بردار کوچکتر به طور متناوب تکرار می‌شود تا زمانی که طول دو بردار برابر شوند. در جدول زیر چند تابع مورد نیاز برای محاسبات ماتریسی ذکر شده است.

تابع	شرح
diag( )	به دست آوردن قطراصلی ماتریس و ساختن یک ماتریس قطری
eigen( )	به دست آوردن مقادیر وبردارهای ویژه یک ماتریس مربع
kronecker( )	انجام عملیات کرونکر
t( )	محاسبه‌ی ترانهاده‌ی یک ماتریس
solve( )	معکوس یک ماتریس

تابع *eigen*( ) هم مقادیر ویژه وهم بردارهای ویژه را می‌دهد. اگر فقط مقادیر ویژه را خواستیم از *eigen*( )$*values* و اگر فقط بردارهای ویژه را خواستیم از *eigen*( )$*vectors* استفاده می‌کنیم.

مثال 5 :

```
diag(matrix (c(1.2,3.98,5,4.3),2))↵
[1] 1.2 4.3
diag(2)↵
     [,1] [,2]
[1,]    1    0
[2,]    0    1
```

دوکاربرد تابع diag( ) را در بالا مشاهده کردیم، کاربرد توابع دیگر را در ادامه ببینید.

```
eigen(matrix(1:4,2))↵
$values:
[1] 5.37  -.38
$vectors:
          [,1]       [,2]
[1,]  -0.57   -0.91
[2,]  -0.84    0.41

t(matrix(1:4,2))↵
     [,1] [,2]
[1,]    1    2
[2,]    3    4
solve(matrix(1:4,2))↵
     [,1]  [,2]
[1,]   -2   1.5
[2,]    1  -0.5
```

نکته 3 :

در $S+$ تابعی برای محاسبه‌ی دترمینان وجود ندارد، اما می‌توانیم به صورت زیر دترمینان را محاسبه کنیم. که همان حاصل ضرب مقادیر ویژه‌ی ماتریس x است.

```
det<-prod(eigen(x)$values)
```

کار برد عملگرهای حسابی

قبلاًاشاره کردیم که عملیات حسابی روی ماتریسها به صورت عنصر به عنصر صورت می‌گیرد. کاربرد این عملگرها را در مثال‌های زیر مشاهده می‌کنیم.

مثال6:

```
A<-matrix(c(19,8,11,2,18,17,15,19,10),3)
B<-matrix(c(14,13,0,10,11,15,19,3,15),3)
A*B↵
     [,1] [,2] [,3]
[1,]  266   20  285
[2,]  104  198   57
[3,]  110  255  150
```

عملیات حسابی جمع، تفریق وتقسیم نیز مثل عمل ضرب به صورت عنصر به عنصر صورت می‌گیرد.

مثال7:

```
x < –c(3,1,9,5)
y < –c(1,2,0,5)
x + y↵
[1]  4   3   9   10
x / y↵
[1]  3   .5   NA   1
```

مثال8:

```
x < –matrix(c(1,5,4,1,0,9),2)
y < –matrix(c(1,2,1,2,7,2),2,byrow = T)
x^2↵
      [,1]  [,2]  [,3]
[1,]    1    16     0
[2,]   25     1    81
x / 2↵
      [,1]  [,2]  [,3]
[1,]   .5     2     0
[2,]  2.5    .5   4.5
x^y↵
      [,1]  [,2]  [,3]
[1,]    1    16     0
[2,]   25     1    81
```

در ماتریسها عمل ضرب به صورت ماتریسی توسط عملگر %*% صورت می‌گیرد. این عملگر در بردارها که یک نوع ماتریس به حساب می‌آیند نیز کاربرد دارد.

مثال9 :

```
a < –c(1,4)
y < –matrix(c(1,2,2,7,1,2),2)
b < –t(a)%*%y↵
b↵
[1]  9   30   9
```

هر یک از توابع مقدماتی را که در فصـــل قبل معرفی کردیم نیز در مورد ماتریســـها وبردارها کاربرد دارند.

مثال10:

ماتریس مثال قبل را در نظر بگیرید. برای این ماتریس نشان می‌دهیم که می‌توان از توابع ریاضی استفاده کرد.

```
log(y)↵
        [,1]     [,2]     [,3]
[1,]       0   .6931        0
[2,]  0.6931  1.9459   .6931
sum(y)↵
[1] 15
prod(y[1,])↵
[1] 2
sum(y[1:2,2:3])↵
[1] 12
```

نکته 4 :

گاهی اوقات در وارد کردن داده‌ها در یک بردار یا در یک ماتریس ممکن است به یک عدد اشتباهاً به جای ارزش عددی، ارزش اسمی داده باشیم، برای جبران این اشتباه از دستورات *is.numeric*( ) و به دنبال آن *as.numeric*( ) استفاده می‌کنیم. در داخل پرانتزها اسم متغیر که اینجا ماتریس است را قرار می‌دهیم.

مثال11:

```
x < –c('1','2','3')
is.numeric(x)↵
F
as.numeric(x)
[1] 1 2 3
```

نکته 5:

اگر بخواهیم یک ماتریس را به یک بردار تبدیل کنیم از تابع *as.vector*( ) استفاده می‌کنیم.

مثال12 :

```
a < –matrix(1: 4,2)
as.vector(a)
[1] 1 2 3 4
```

نام گذاری ماتریس

از تابع dim names( ) استفاده می‌کنیم واسم ها را به صورت لیست وارد می‌کنیم. برای مثال بالا داریم:

```
dim names(a) < –list(c('d','b'),c('e','f'))
a↵
  e f
```

d 1 3
b 2 4

البته ساختار داده‌ای لیست را بعداً شرح می‌دهیم.

آرایه

اگر ساختمان داده‌ها طوری باشد که در آن داده‌ها در چند بلوک با تعداد سطرها و ستونهای مساوی قرار گرفته باشند، باید داده‌ها را به صورت آرایه وارد کرد. برای ساختن آرایه در $S+$ از تابع $array(\ )$ استفاده می‌کنیم. در این تابع دو شناسه را مشخص می‌کنیم. شناسه‌ی اول بردار مقادیر مشاهده شده‌ی ما می‌باشد وشناسه‌ی دوم برداری است شامل ابعاد آرایه، که در آن به ترتیب تعداد سطرها، تعداد ستونها و تعداد بلوک‌ها را وارد می‌کنیم. فرم کلی به صورت زیر است.

$array($داده‌ها$,\ \dim = c($تعداد سطرها,تعداد ستونها,تعداد بلوک‌ها$))$

نکته 6 :

در آرایه نیز مانند ماتریس داده‌ها به ترتیب ستونی در بلوک‌ها قرار می‌گیرند.

مثال13 :

```
d <-array(c(1:8,11:18),dim=c(2,4,2))↵
,,1
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
,,2
     [,1] [,2] [,3] [,4]
[1,]   11   13   15   17
[2,]   12   14   16   18
```

این تابع ابتدا مقادیر اولین بعد را کامل می‌کند، سپس دومین بعد و به همین ترتیب بعدهای دیگر را کامل می‌کند.

ویژگی‌های آرایه

آرایه نیز مانند ماتریس دارای طول، حالت و بعد می‌باشد که به ترتیب با توابع زیر به دست می‌آیند. $length(\ ),mode(\ ),dim(\ )$، در اینجا تابع $dim(\ )$ دارای سه شناسه می‌باشد. (تعداد بلوک‌ها،ستون،سطر)

مثال 14:

```
dim(d)↵
[1] 2 4 2
```

برای انتخاب کردن درایه‌ی خاصی از آرایه مانند آنچه که در ماتریس و بردار داشتیم، با مشخص کردن شماره‌ی سطر، ستون و بلوک مربوط به آن درایه این کار را انجام می‌دهیم. به مثال زیر توجه کنید.

```
d[1,1,2]↵
[1] 11
```

در مورد آرایه مانند ماتریس تابع length( ) تعداد کل درایه‌های آرایه را به ما می‌دهد.

```
length(d)↵
[1] 16
```

نام گذاری آرایه

برای نام گذاری یک آرایه از تابع dimnames( ) استفاده می‌کنیم و اسم‌های مربوط به سطرها، ستونها و بلوک‌ها را به صورت لیست وارد می‌کنیم.

مثال 15 :

```
dimnames(d) <–list(paste("Test",1:2),NULL,NULL)↵
```

درمثال 15 تابع dimnames( ) نام‌های Test1,Test2 را به سطرهای هر بعد نسبت می‌دهد ولی به ستونها وبلوک‌ها اسمی اختصاص نمی‌دهد. اگر بخواهیم به ستونها و بلوک‌ها اسم اختصاص دهیم، به جای NULL برداری شامل اسم‌ها را قرار می‌دهیم.

لیست

لیست ساختاری است که می‌تواند مقادیری با حالتهای مختلف داشته باشد. به عنوان مثال یک لیست ممکن است شامل مقادیر عددی و کاراکتری باشد، ویا مثلاً مؤلفه‌های لیست توأماً می‌توانند ماتریس، بردار یا هر نوع دیگری از ساختار داده باشد. برای ساختن لیست از تابع list( ) استفاده می‌کنیم. این نوع ساختار داده را می‌توان در مورد داده‌های طرح آزمایشات به کار برد.

مثال 16 :

```
alist < –list(c(0,1,2),1:10)↵
alist↵
[[1]] :
[1] 0 1 2
[[2]] :
[2] 1 2 3 4 5 6 7 8 9 10
```

alist[[1]]↵
[1] 0 1 2

در مثال بالا مشاهده می‌کنیم که برای فراخوانی مؤلفه‌های لیست از علامت [[ ]] استفاده می‌کنیم.

نکته 7 :

در نام گذاریها از دستور paste( ) استفاده کردهایم. این دستور دارای دو شناسه است شناسهی اول عبارت مورد نظر و شناسهی دوم اندیسها است. مثلاً اگر بخواهیم حرفb را با اعداد 1 تا 3 اندیس گذلری کنیم، این دستور را به صورت زیر به کار می بریم.

$paste(\mathrm{b}, 1{:}3)$ ↵
$[1]\ b_1\ b_2\ b_3$

نام گذاری لیست

مؤلفه‌های یک لیست را می‌توان نام گذاری کرد. برای این کار دردستور لیست مؤلفه‌ها را به همراه اسمهایشان قرار می‌دهیم. مثال بعد را به دقت ببینید.

مثال 17 :

b < –list(c("A","B","C"),1:5,"N")↵
c < –list(x = matrix(1:4,2), y = c("A","B"), z = b)↵
c$x↵
```
     [,1] [2,]
[1,]    1    3
[2,]    2    4
```
c$y↵
[1] "A" "B"
c$z[[2]]↵
[1] 1 2 3 4 5

در مثال بالا علاوه بر مشاهده‌ی چگونگی نام گذاری یک لیست می‌بینیم که یک لیست چگونه می‌تواند شامل ساختارهای مختلف داده باشد. در این مثال مؤلفه‌ها ماتریس، بردار و لیست هستند. لیستها هم دارای سه ویژگی طول، حالت واسم هستند. طول لیست تعداد مؤلفه‌های لیست است.

length(c)↵
[1] 3

حالت یک لیست، *list*( ) است. وبا تابع *names*( ) می توان اسم یک لیست را نمایش داد.

names(c)↵
[1] "x" "y" "z"

نکته 8 :

توابع مقدماتی که در فصل قبل معرفی کردیم در مورد ماتریس و آرایه مانند بردارها کاربرد دارند. اما در مورد لیستها کاربرد ندارند.
در آخر چگونگی کار کرد چند تابع مهم که در مورد ساختارهای داده‌های که شرح دادیم به کار می‌روند را بررسی می‌کنیم.

1) توابع apply( ) و sapply( ) و lapply( )

تابع apply( ) با شناسه‌هایش به صورت زیر به کار می‌رود.

apply(اسم متغیر ,بعد ,fun)

این دستور برای ماتریس و آرایه کاربرد دارد. در قسمت بعد سطررا با عدد (1) و ستون را با عدد (2) مشخص می‌کنیم وعملیاتی را که می‌خواهیم روی آنها انجام دهیم، در قسمت fun تعیین می‌کنیم. البته در مورد آرایه در قسمت شناسه‌ی بعد عدد 3 را نیز در مورد بلوکها به کار می‌بریم.
توابع sapply( ) و lapply( ) در مورد لیستها کاربرد دارد ودارای همان شناسه‌های دستور apply( ) می‌باشند، ولی شناسه‌ی بعد را دیگر در نظر نمی‌گیریم. تفاوت این دو دستور در این است که اولی خروجی را به صورت بردار ودومی به صورت لیست به ما می‌دهد.

مثال18 :

```
a <-matrix(1:6,3)
apply(a,1,sum)↵
[1] 5 7 9
b <-list(1:4,6:9)
sapply(b,prod)↵
[1] 24 3024
lapply(b,prod)↵
[[1]] :
[1] 24
[[2]] :
[2] 3024
```

به تفاوت خروجیها در مثال بالا توجه کنید.

2) تابع outer( )

این تابع محاسبات خارجی مانند ضرب خارجی را برای دو بردار و یا دوماتریس انجام می دهد. این تابع سه شناسه دارد .

outer(بردار اول ,بردار دوم ,fun)

شناسه‌ی سوم یک عملگر حسابی یا یک تابع در s + است.

مثال19:

$outer(1:3,1:4,'+')$ ↲

$$\begin{bmatrix}2 & 3 & 4 & 5\\ 3 & 4 & 5 & 6\\ 4 & 5 & 6 & 7\end{bmatrix}$$

$outer(1:3,1:4,'*')$ ↲

$$\begin{bmatrix}1 & 2 & 3 & 4\\ 2 & 4 & 6 & 8\\ 3 & 6 & 9 & 12\end{bmatrix}$$

در این مثال " * " همان کار عملگر (%o%) یعنی ضرب خارجی را انجام می دهد.

مثال 20 :

فرض کنید دو متغیر مستقل هستند، برنامه‌ای بنویسید که احتمال‌های حاشیه‌ای را در هم ضرب کند واحتمال‌های توأم را به دست آورد .

	1	2	3	4	$\pi_{i.}$
1	π_{11}	π_{12}	π_{13}	π_{14}	1/4
2	π_{21}	π_{22}	π_{23}	π_{24}	2/4
3	π_{31}	π_{32}	π_{33}	π_{34}	1/4
$\pi_{.j}$	1/8	3/8	3/8	1/8	

```
x <–outer(c(1/4,2/4,1/4),c(1/8,3/8,3/8,1/8),"*")
```

محاسبات بالا را می‌توانستیم با تابعی دیگر به اسم *kronec*ker( ) انجام دهیم. تفاوتی که با هم دارند در این است که در *outer*( ) خروجی به صورت ماتریس است ولی در *kronec*ker( ) به صورت بردار است. از این دستور به صورت زیر استفاده می‌کنیم .

```
x <–kronecker(c(1/4,2/4,1/4),c(1/8,3/8,3/8,1/8),"*")
```

خروجی دستور اول یعنی دستور outer به صورت زیر است.

```
     [,1]    [,2]    [,3]    [,4]
[1,] 0.03125 0.09375 0.09375 0.03125
[2,] 0.06250 0.18750 0.18750 0.06250
```

[3,] 0.03125 0.09375 0.09375 0.03125

و خروجی دستور دوم یعنی دستور kronecker به صورت زیر است.

```
        [,1]
 [1,] 0.03125
 [2,] 0.09375
 [3,] 0.09375
 [4,] 0.03125
 [5,] 0.06250
 [6,] 0.18750
 [7,] 0.18750
 [8,] 0.06250
 [9,] 0.03125
[10,] 0.09375
[11,] 0.09375
[12,] 0.03125
```

تمرینهای فصل دوم

تمرین 1 :

برنامه‌ای بنویسید برای محاسبه‌ی مجموع مربعات جدول آنالیز واریانس در طرح دو عاملی با تکرارهای مساوی در هر سطح، به طوری که، a = تعداد سطرها، b = تعداد ستونها، n = تعداد تکرارها، x = داده ها، که به صورت آرایه آن را وارد می‌کنیم. تعداد ابعاد این آرایه $a*b$ است که در هر بعد یک سطر و به تعداد تکرارها ستون داریم .

```
Tarh <–function(x,a,b,n){
h <–rep(0,a * b)
for(i in 1:(a * b)) {h[i] <–sum(x[,,i])}
yij. <–matrix(h,a,b,byrow = T)
yi.. <–apply(yij.,1,sum)
y.j. <–apply(yij.,2,sum)
y... <–sum(yij.)
ssT <–sum(x^2) – (y...^2)/(a * b * n)
ssa <–(sum(yi..^2)/(b * n)) – (y...^2)/(a * b * n)
ssb <–(sum(y.j.^2)/(a * n)) – (y...^2)/(a * b * n)
ssab <–(sum(yij.^2)/ n) – (y...^2)/(a * b * n) – ssa – ssb
ssE <–ssT – ssa – ssb – ssab
Fa <–(ssa/(a – 1))/(ssE/(a * b * (n – 1)))
Fb <–(ssb/(b – 1))/(ssE/(a * b * (n – 1)))
Fab <–(ssab/(a – 1) * (b – 1))/(ssE/(a * b * (n – 1)))
return(yij.,yi..,y.j.,y...,ssT,ssa,ssb,ssab,ssE,Fa,Fb,Fab)
}
```

تمرین 2 :

تابعی بنویسید که دو ماتریس را در هم ضرب کند .

```
multiply <– function(mat1,mat2){
if (dim(mat1)[2]!= dim(mat2)[1])return(NA)
n <– dim(mat1)[1]
p <– dim(mat1)[2]
q <– dim(mat2)[2]

mat <–matrix(,n,q)
for(i in 1:n){
for(j in 1:q){
mat[i,j] <–sum(mat1[i,] * mat2[,j])
} }
return(mat)
}
```

فصل سوم
کاربرد حلقه‌ها و دستور if

کاربرد حلقه‌ها

حلقه‌ی for

برای تکرار یک سری دستور از حلقه‌ها استفاده می‌کنیم. فرم کلی حلقه‌ی for به صورت زیر است.

$for(i \quad in \quad a:b) \quad \{ \quad scommand \quad \}$

متغیر i آرگمان حلقه است و در این حلقه از عدد a تاعدد b شمرده می‌شود. ساختار این دستور طوری است که تمامی دستوراتی که در داخل دو تا آکلاد هستند، تازمانی که آرگمان تابع شمرده می‌شود، اجرا می‌شوند و درهر مرحله آرگمان تابع از عدد a تا عدد b یکی یکی شمرده می‌شود. برای شمرده شدن آرگمان حلقه می‌توان از توابع و عملگرهایی که تا کنون یاد گرفته‌ایم به نحوی مناسب استفاده کنیم. مثلاً در بالا می‌توان از دستور $seq(\ )$ به صورت زیر استفاده کرد.

$for(i \quad in \; seq(a,b,1))$

مثال 1:

اعداد 1تا 100 را با استفاده از حلقه جمع ببندید.

```
x<-0
for(i in 1:100) {x<-x+i}
x↵
```

اگر بخواهیم فقط اعداد زوج بین 1 تا 100 با هم جمع شوند دستورات زیر را به کار می‌بریم.

```
X<- 0
for (i in 1:50) {j<- 2*i ; x<-x+ j}
x↵
```

مثال 2 :

با استفاده از حلقه‌ی for مقدار ضریب چولگی را محاسبه کنید.

$x < -c(x_1, x_2, ..., x_n)$

$n < -length(x)$

$a < -0$

$b < -0$

$for(i\ in\ 1:n)\ \{$

$a < -a + (x[i] - mean(x))\text{^}3$

$b < -b + (x[i] - mean(x))\text{^}2$

$\}$

$g_1 < -sqrt(n) * a/(b\text{^}(3/2))$

مثال 3 :

با استفاده از حلقه مربع اعداد 9،2،3و6 را به دست آورید.

$for(i\ in\ c(3,2,9,6))\ \{$

$print(i\text{^}2)$

$\}$

در اینجا از دستور ( ) print که عبارت داخل پرانتز را برای ما عیناً چاپ می‌کند استفاده کرده‌ایم.

حلقه‌های for تو در تو

درحلقه‌های تو درتو یک حلقه درون حلقه‌ی دیگر قرار می‌گیرد. اول آرگمان حلقه‌ی اول شـــمرده می‌شود بعد برای این مقداراز آرگمان حلقه‌ی بیرونی دستورات حلقه‌ی درونی به تعدادی که آرگمان آن شمرده می‌شود یک باربه طور کامل انجام می‌شوند و همین عملیات برای مقادیر دیگر حلقه‌ی بیرونی تکرار می‌شود. فرم کلی به صورت زیر است.

$for(i\ in\ a:b)\ \{for(j\ in\ c:d)\ \{commands$

$\}\ \}$

مثال 4 :

ترا نهاده‌ی ماتریس زیر را به دست آورید.

$$A = \begin{bmatrix} a_{11} & . & . & . & a_{1n} \\ . & . & & & . \\ . & & . & & . \\ . & & & . & . \\ a_{n1} & . & . & . & a_{nn} \end{bmatrix}$$

$A <- matrix(c(a_{11},...,a_{nn}),n,n,byrow=T)$

$B <- matrix(0,n,n)$

$for(i\ in\ 1:n)\ \{for(j\ in\ 1:n)\ \{\ B[i,j] <- A[j,i]\ \}\ \}$

مثال 5 :

ماتریس متقارن متناظر با مثال قبل را که به صورت $B = \frac{(A+t(A))}{2}$ است پیدا کنید. از خط سوم به بعد دستورات مثال قبل را به صورت زیر به کار می‌بریم.

$for(i\ in\ 1:n)\ \{\ for(j\ in\ 1:n)\ \{B[i,j] <- (A[i,j]+A[j,i])/2\ \}\ \}$

حلقه‌ی *repeat* :

این حلقه نیز مانند حلقه‌ی *for* برای تکرار یک سری دستورات به کار می رود. اما فرق آن با حلقه‌ی *for* در این است که در اینجا دستورات حلقه آن قدر تکرار می‌شوند تا به جواب مسئله برسیم یعنی برقراری شرط پایانی در جریان برنامه مشخص می‌شود و شرط پایانی را از همان ابتدا مشخص نمی‌کنیم. فرم کلی به صورت زیر است.

```
repeat{
commands
if(...) {break }
 }
```

که در داخل پرانتزهای مقابل *if* شرط پایانی حلقه را می‌نویسیم.

مثال 6 :

به برنامه‌ی زیر توجه کنید.

```
n <-0
a <-0
repeat{
n <-n+1
a <-n+a
if(a>1000) {break }
 }
```

این برنامه تا زمانی که عدد a کمتر از 1000 باشد دستورات را اجرا می‌کند.

حلقه‌ی *while*

فرم کلی این حلقه به صورت زیر است.

```
while(قید شرطی) {
```

.....
.....
}

درون آکلادها دستوراتی را که در جریان حلقه تکرار می‌شوند، می‌نویسیم. توجه کنیم که شرط پایانی از همان اول حلقه معلوم است. ولی در حلقه‌ی *repeat* این شرط از همان ابتدا معلوم نیست وباید در جریان برنامه ساخته شود.

مثال 7 :

مجموع اعداد زوج بین1 تا 100 را با حلقه *repeat* پیدا می‌کنیم.

$i<-0;\ j<-0;\ repeat\{\ i<-i+1;\ j<-j+(2*i);\ if(i>=50)\ \{break\}\ \}$

دستور *if*

فرم کلی این دستور به صورت زیر است.

if(شرط){ دستور اول }
else{ دستور دوم }

که درآکلادهای مقابل *if* دستوراتی را که در صورت برقرار بودن شرط درون پرانتزها باید اجرا شوند، می‌نویسیم ودر آکلادهای مقابل *else* دستوراتی را که در صورت برقرار نبودن شرط درون پرانتزها باید اجرا شوند، می‌نویسیم. اگر دستورات فقط شامل یک قسمت بودند احتیاجی به آکلادها نیست.

گاهی لازم نیست حتماً دستور *else* را بعد از دستور *if* نوشت. البته اگر بخواهیم در صورت برقرار نبودن شرط درون پرانتزها دستور دیگری اجرا شود *else* لازم است. به جای اینکه دستور را به صورت بالا به کار ببریم، می‌توانیم از دستور زیر استفاده کنیم.

ifelse(شرط، دستور اول، دستور دوم)

در دستور ifelse( ) اگر شرط برقرار باشد، دستور اول اجرا می‌شود و اگر شرط برقرار نباشد دستور دوم اجرا می‌شود.

مثال 8 :

به برنامه‌ی زیر توجه کنید.

$x<-c(0,1,2,3)$
$if(x==0)\ \log(x+1)$
$else\ \log(x)$

این برنامه برای 0 مقدار $\ln(1)$ را محاسبه می‌کند وبرای بقیه‌ی اعداد $\ln(x)$ رامحاسبه می‌کند.

تمرینات فصل سوم

تمرین 1 :

به برنامه‌ی زیر توجه کنید.

```
z <- -0
for(i in 1:4) {
z <- -z + i + (i < 3)
}
z
```

به نحوه‌ی عملکرد این برنامه که در زیر شرح می دهیم توجه کنید .

$i = 1 \Rightarrow z = 0 + 1 + (1 < 3) = 1 + T = 1 + 1 = 2$

$i = 2 \Rightarrow z = 2 + 2 + (2 < 3) = 4 + T = 4 + 1 = 5$

$i = 3 \Rightarrow z = 5 + 3 + (3 < 3) = 8 + F = 8 + 0 = 8$

$i = 4 \Rightarrow z = 8 + 4 + (4 < 3) = 12 + F = 12 + 0 = 12$

نکته‌ی جالب در این اســت که ارزش عددی یک گزاره‌ی درســت 1 و یک گزاره‌ی نا درســت 0 است.

تمرین 2 :

برنامه ای بنویســید که به وســیله‌ی آن بتوان مقدار آماره‌ی D'gostino را برای تشــخیص نرمال بودن یک سری مشاهدات به دست آورد. برای محاسبه‌ی این آماره ابتدا داده‌ها را مرتب می‌کنیم، سپس اگر تعداد مشاهدات زوج باشد قرار می‌دهیم : h= n/2 واگر تعداد آنها فرد باشد h=(n-

1)/2 که n تعداد مشاهدات است. سپس برای i=1,2,…,n تعریف می‌کنیم : din=1/2*(n+1)-i
سپس مقدار d را با فرمول زیر محاسبه می‌کنیم.

$$d = \frac{\sum_{i=1}^{h} \mathrm{din} \times (x_{n-i+1} - x_i)}{n^{3/2} \times \sqrt{n-1} \times s}$$

حال مقدار آماره را به صورت $D = \frac{\sqrt{n} \times (d - 0.282095)}{0.029986}$ محاسبه می‌کنیم.

$x < -c(x_1, x_2, \dots, x_n)$
$x < -sort(x)$
$n < -length(x)$
$S < -sum((x - mean(x))\text{^}2)/(n-1)$
$if\ (n/2 - floor(n/2) == 0)\ \{h < -n/2\}$
$else\{h < -(n-1)/2\}$
$din < -(1/2 * (n+1) - (1:h))$
$d < -0$
$for(i\ in\ 1:h)\ \{d < -d + (din[i] * (x[n-i+1] - x[i]))\}$
$d < -d/((n\text{^}(3/2)) * sqrt((n-1) * S))$
$D < -(sqrt(n) * (d - 0.282095))/0.029986$
D

تمرین 3 :

برنامه‌ای بنویسید که برای داده‌های زیر مقدار آماره‌ی leven را برای آزمون برابری واریانس‌ها به دست آورد. برای محاسبه‌ی این آماره ابتدا قرار می‌دهیم :

$$y_{ij} = |y_{ij} - \bar{y}_{i.}|$$

مقدار آماره را به صورت زیر به دست می‌آوریم. n تعداد کل مشاهدات است.

$$z = \frac{\sum_{i=1}^{k} n_i (\bar{y}_{i.} - \bar{y}_{..})^2 / k - 1}{\sum_{i=1}^{k} \sum_{j=1}^{n_i} (y_{ij} - \bar{y}_{i.})^2 / n - k}$$

مشاهدات

سطح اول	7	7	15	11	9
سطح دوم	12	17	12	18	18
سطح سوم	14	18	18	19	19
سطح چهارم	19	25	22	19	23
سطح پنجم	7	10	11	15	11

$x < -matrix(c(7,7,15,11,9,\ldots,7,10,11,15,11),5,5,byrow = T)$

$n < -0$

$k < -\dim(X)\,[1]$

$for(i\ in\ 1{:}\,\mathrm{k})\{n[i] < -length(x[i,])\}$

$yi. < -0$

$for(j\ in\ 1{:}\,\mathrm{k})\ \{yi.[j] < -mean(x[j,])\}$

$y.. < -mean(yi.)$

$yij < -matrix(0,5,5)$

$for(l\ in\ 1{:}\,\mathrm{k})\{yij[l,] < -(x[l,] - yi.[l])\ \}$

$yij < -abs(yij)$

$a < -sum(n * ((yi. - y..) * 2))/k - 1$

$d < -0$

$for(t\ in\ 1{:}\,k)\{d[t] < -sum((yij[t,] - yi.[t]) \wedge 2)\}$

$d2 < -sum(d)$

$b < -d2/(sum(n) - k)$

$z < -a/b$

z

فصل چهارم

محاسبه ی احتمال، اعداد تصادفی و چندکها

در S+ توابع بسیاری برای محاسبه‌ی چندکها، مقادیر احتمال توزیع‌های احتمال مختلف وتولید اعداد تصادفی وجود دارد. نام هریک از این توابع با یکی از حروف زیرشروع می شود. این حروف به همراه کد توزیع که تابع مربوط به آن توزیع را می سازد و پارامترهای آن توزیع در جدول زیر تنظیم شده اند.

- r – تولید اعداد تصادفی
- p – محاسبه‌ی مقدار احتمال
- d – محاسبه‌ی تابع چگالی
- q – محاسبه‌ی چندک

توزیع	کد توزیع	پارامترهای لازم	پارامترهای اختیاری	مقادیر پیش فرض
نرمال	*norm*	.....	*mean,sd*	0,1
یکنواخت	*unif*	.....	min,max	0,1
نمایی	exp	.....	*rate*	1
بتا	*beta*	*shape1,shape2*	.....	.....
گاما	*gamma*	*shape,rate*	*rate*	...,1
وایبل	*weibull*	*shape,rate*	*rate*	...,1
فیشر	*f*	*df1,df2*	.....	.....
t – استودنت	*t*	*df*	.....	.....
کای دو	*chisq*	*df*	.....	.....
باینومیا ل	*binom*	*size,prob*	.....	.....
پواسن	*pois*	*lambda*	.....	.....
هندسی	*geom*	*prob*	.....	.....
فوق هندسی	*hyper*	*m,n,k*	.....	.....
دو جمله‌ای منفی	*nbinom*	*size,prob*	.....	.....

هر کدام از این توابع را با مثال شرح می دهیم.

توزیع نرمال

rnorm(*n*,*mean*,*sd*)
pnorm(*q*,*mean*,*sd*)
dnorm(*q*,*mean*,*sd*)
qnorm(*p*,*mean*,*sd*)

توجه کنیم که اگر از مقادیر پیش فرض پارامترها استفاده شد، با توزیع نرمال استاندارد سرو کار داریم.

مثال 1:

rnom(100) صد نمونه‌ی تصادفی از توزیع نرمال استاندارد تولید می کند.

dnorm(1.96,5) چگالی نقطه‌ی 1.96 را در توزیع نرمال با میانگین 5 و واریانس 1 تولید می کند.

مثال 2:

فرض کنید $x_1, x_2, ..., x_n$ یک نمونه ی تصادفی از توزیع نرمال با واریانس 1 باشد برنامه ای

$$H_o : \mu = 4$$

بنویسید که مقدار $p-value$ را برای آزمون فرض $H_1 : \mu \neq 4$ به دست آورد.

```
) x1,x2,..., xn  X <–c(
n <–length(X)
x_bar <–mean(X)
z <–sqrt(n)*(x – bar – 4)
p_value <–2*(1– pnorm(abs(z)))
p_value↵
```

توزیع یکنواخت

runif(*n*,min,max)
punif(*q*,min,max)
dunif(*q*,min,max)
qunif(*p*,min,max)

توجه کنیم که اگر دو پارامتر را ننویسیم، مقادیر پیش فرض آنها یعنی $\min = 0$ و $\max = 1$ در نظر گرفته می شوند.

مثال3:

runif(50) پنجاه نمونه‌ی تصادفی از توزیع یکنواخت، $u(0,1)$ تولید می کند.

توزیع نمایی

r exp(*n*,*rate*)
p exp(*q*,*rate*)
d exp(*q*,*rate*)
q exp(*p*,*rate*)

اگر مقدار پارامتر را ننویسیم، مقدار پیش فرض آن یعنی 1 را در نظر می گیرد. در اینجا تابع چگالی به صورت روبه رو در نظر گرفته می شود. $f(x) = \theta\ e^{-\theta x} \quad x = 0,1,...$

توزیع بتا

rbeta(*n*,*shape*1,*shape*2)
pbeta(*q*,*shape*1,*shape*2)
dbeta(*q*,*shape*1,*shape*2)
qbeta(*p*,*shape*1,*shape*2)

در این توزیع برای پارامترها حالت پیش فرضی وجود ندارد.

مثال4:

rbeta(100,1,5) صد نمونه‌ی تصادی از چگالی $f(x) = 5(1-x)^4$ تولید می کند.

توزیع گاما

rgamma(*n*,*shape*,*rate*)
pgamma(*q*,*shape*,*rate*)
dgamma(*q*,*shape*,*rate*)
qgamma(*p*,*shape*,*rate*)

مثال 5:

rgamma(100,α,λ) صد نمونه‌ی تصادفی از چگالی $f(x) = \frac{1}{\Gamma(\alpha)\lambda^{\alpha}} x^{\alpha-1} e^{\frac{-x}{\lambda}}$ تولید می کند.

توزیع وایبل

rweibull(*n*,*shape*,*rate*)
pweibull(*q*,*shape*,*rate*)
dweibull(*q*,*shape*,*rate*)
qweibull(*p*,*shape*,*rate*)

مثال6:

$qweibull(.25,\alpha,\beta)$ چارک اول را از چگالی $f(x)=\alpha\beta^{\alpha}x^{\alpha-1}e^{-\beta x^{\alpha}}$ پیدا می کند.

توزیع فیشر

$rf(n,df1,df2)$
$pf(q,df1,df2)$
$df(q,df1,df2)$
$qf(p,df1,df2)$

مثال7:

آزمون فرض برابری واریانس‌ها را در دو جامعه‌ی نرمال با محاسبه‌ی مقدار p_value انجام دهید.

$X<-c(x_1,x_2,...,x_n)$
$Y<-c(y_1,y_2,...,y_m)$
$n<-length(X)$
$m<-length(Y)$
$F<-\text{var}(X)/\text{var}(Y)$
$p\_value<-1-pf(F,(n-1),(m-1))$
$if(p\_value>0.05)\ \{\ print("H_0 accepted")\ \}\quad else\{\ print("H_0 rejected")\ \}$

توزیع $t-$ استودنت

$rt(n,df)$
$pt(q,df)$
$dt(q,df)$
$qf(p,df)$

توزیع کای دو

$rchisq(n,df)$
$pchisq(q,df)$
$dchisq(q,df)$
$qchisq(p,df)$

مثال8:

مقدار آماره‌ی کای دو و مقدار p_value را در آزمون استقلال برای جدول توافقی زیر به دست آورید.

x \ Y	1	2
1	25	10
2	19	45

```
A<-matrix(c(25,10,19,45),2,byrow=T)
n<-dim(A)[1]; m<-dim(A)[2]
mm<-matrix(0,n,m)
for(i in 1:n){ for(i in 1:m) { mm[i,j]<-(sum(A[i, ])*sum(A[ ,j]))/sum(A) } }
a<-0
for(i in 1:n) { for(j in 1:m) { a<-a+((A[i,j]-mm[i,j])^2)/mm[i,j] } }
p_value<-1-pchisq(a,(n-1)*(m-1))
a
p_value
```

متغیر a در اینجا همان مقدار آماره‌ی آزمون می باشد.

توزیع باینومیال

```
rbinom(n,size,prob)
pbinom(q,size,prob)
dbinom(q,size,prob)
qbinom(p,size,prob)
```

مثال9:

اگر $x \sim binom(20,1/3)$ مقدار تابع توزیع را در نقطه‌ی $x=14$ پیدا کنید.

می دانیم که $F(14)=\sum_{0}^{14}\binom{20}{x}(1/3)^x(2/3)^{20-x}$ که با دستور زیر محاسبه می شود.

```
pbinom(14,20,1/3)
```

توزیع دو جمله‌ای منفی

```
rnbinom(n,size,prob)
pnbinom(q,size,prob)
dnbinom(q,size,prob)
qnbinom(p,size,prob)
```

توجه کنیم که اگر $X \sim nbinom(r,p)$ تابع چگالی احتمال به صورت زیر می باشد.

$$p(X=x)=\binom{x-r-1}{r-1}p^r(1-p)^x$$

توزیع پواسن

rpois(*n*,*lambda*)
ppois(*q*,*lambda*)
dpois(*q*,*lambda*)
qpois(*p*,*lambda*)

مثال10:

ppois(4,2) مقدار تابع توزیع را در نقطه‌ی $X=4$ به ما می دهد. تابع چگالی احتمال X در این مثال به صورت زیر است.

$$p(X=x)=\frac{e^{-2}2^x}{x!}$$

توزیع هندسی

rgeom(*n*,*prob*)
pgeom(*q*,*prob*)
dgeom(*q*,*prob*)
qgeom(*p*,*prod*)

در اینجا منظور از *prob* همان پارامتر توزیع است. در این توزیع $p(X=x)=(1-p)^x p$

توزیع فوق هندسی

rhyper(*n*,*m*,*n*,*k*)
phyper(*q*,*m*,*n*,*k*)
dhyper(*q*,*m*,*n*,*k*)
qhyper(*p*,*m*,*n*,*k*)

دراین توزیع m تعداد کالاهای ویژه، n بقیه‌ی کالاها و k تعداد نمونه اســـت. تابع چگالی به صورت زیر است.

$$p(X=x)=\frac{\binom{m}{x}\binom{n}{k-x}}{\binom{m+n}{k}} \quad x=0,1,\ldots,\min(m,k)$$

توزیع نرمال چند متغیره

rmvnorm(*n*,*mean*,*sigma*)
pmvnorm(*q*,*mean*,*sigma*)
dmvnorm(*q*,*mean*,*sima*)

که در آن q بردار مشاهدات، mean بردار میانگین و sigma ماتریس واریانس کوواریانس است.

برای نمونه گیری از یک توزیع گسسته‌ی دلخواه از تابع $sample( )$ استفاده میکنیم. فرم کلی آن به صورت زیر است.

$$sample(x, size, replace = F, prob)$$

که در آن شناسه‌ی اول برداری شامل مقادیر متغیر تصادفی است که میخواهیم از آن نمونه تولید کنیم. شناسه‌ی دوم تعداد نمونه، شناسه‌ی سوم تعیین میکند که نمونه گیری بدون جایگذاری باشد یا با جایگذاری وشناسه‌ی آخر برداری است که در آن مقادیر تابع چگالی احتمال متغیر را قرار میدهیم.

فصل پنجم

تعریف توابع جدید در s+

s+ یک زبان برنامه نویسی است که به ما امکان می دهد برنامه هایی را که دائماً به کار می بریم در یک تابع جدید ذخیره کنیم. به این ترتیب هر بار که احتیاج به اجرای آن دستورات داشته باشیم در هر جایی از برنامه باشیم می توانیم آن تابع را با نامی که آن را ذخیره کرده ایم فراخوانی کنیم. ساختار عمومی یک تابع به صورت زیر است.

```
اسم تابع <- function(آرگمان های ورودی تابع) {
متن برنامه +
return(آرگمان های خروجی تابع)
}
```

آرگمان های ورودی تابع با یک کاما از هم جدا می شوند و همان متغیرهایی هستند که در متن برنامه از آنها استفاده می‌کنیم. و قسمت متن برنامه شامل دستورات تعریف شده در s+ است که با یک علامت (;) یا یک خط جدید از هم جدا می شوند. در قسمت *return* آرگمان هایی را که تابع باید به عنوان خروجی به ما بدهد می نویسیم. به مثال های ساده‌ی زیر توجه کنید.

مثال 1:

می‌خواهیم تابعی بنویسیم که از این به بعد توسط آن مجذور هر عدد دلخواهی را حساب کند.

```
square <- function(x) {
y <- x^2
return(y)
}
```

این تابع عدد *x* را به عنوان ورودی می گیرد وآن را به توان 2 می رساند وتوان دوم آن را در خروجی به ما نشان می دهد.

```
square(2)↵
4
```

مثال2:

```
a <- function(x, y) {
if (x!= 0 | y!= 0) { z <- x /(x^2 + y^2) }
return(z)
}
```

a(1,2)↵
[1] 0.2

نوع ســـاختار آرگمانهای ورودی تابع در کاربرد آنها در متن برنامه معلوم می‌شـــود و نیازی نیســـت که ما در موقع معرفی آرگمانها ســـاختار آنها را نیز مشـــخص کنیم، فقط ذکر نام آنها به عنوان آرگمانهای ورودی کافی است.

نکاتی در مورد آرگمان های ورودی تابع

آرگمان‌های ورودی تابع به یکی ازدوحالت زیر می باشند.

1)ورودی‌های قطعی وحتمی (entry)

2)ورودی‌های دلخواه (option)، که وارد کردنشان اجباری ندارد.

اگر ما هنگام معرفی آرگمانها به آنها مقدار اولیه‌ای را نســـبت دهیم این ورودی دلخواه به شمار می آید. یعنی در هنگام فراخوانی تابع می‌توان از مقدار پیش فرض آن استفاده کرد ومقداری برای آن بـه عنوان ورودی وارد نکرد. ولی اگر بخواهیم مقداردلخواهی به غیر از مقدار پیش فرض برای آن در نظر بگیریم لازم اســـت که مقدار آن را وارد کنیم. از این رو این ورودی‌هـا را دلخواه می‌نـامیم. ولی اگر برای یـک آرگمـان ورودی مقدار پیش فرض تعیین نکنیم ورودی قطعی به شـــمار می آید و هنگام فراخوانی تابع باید حتماً برای آن مقداری را وارد کنیم .

توجـه کنیـد کـه بـه جز مقـادیر عـددی می‌توان از گزاره‌هـای منطقی T,F نیز بـه عنوان آگمانهای ورودی تابع استفاده کرد.

مثال3:

```
aa<-function(x,y=1) { if( x!=0|y!=0) {z<-x/(x^2+y^2) }; return(z) }
```

در این مثال متغیر y یک ورودی دلخواه اسـت بعداً در اسـتفاده از این تابع می توان به جای آن عددی قرار نداد در این صـورت مقدارپیش فرض آن یعنی مقدار1 برای آن در نظر گرفته می شود.

مثال4:

```
bb<-function(x,unbiased=T) {
n<-length(x)
if(unbiased=T) {y<-sum(x)/(n-1) }
else{y<-sum(x)/n }
return(y)
 }
```

دراین جا متغیر *unbiased* یک ورودی دلخواه منطقی است. حالت پیش فرض آن گزاره‌ی منطقی *T* است. در مثالهای زیراز تابع *bb* استفاده کرده ایم.

```
bb(c(2,4,5))↵
[1] 5.5
bb(c(3,4,5), F)↵
[1] 3.666667
```

مثال 5:

تابعی بنویسید که یک نقطه و پارامترهای چگالی نرمال دو متغیره را بگیرد و مقدار چگالی را در آن نقطه به ما بدهد واگر پارامترها را وارد نکردیم، مقادیر پیش فرض ($\mu_1 = 0, \mu_2 = 0, \sigma_1 = 1, \sigma_2 = 1, \rho = 1$) را در نظر بگیرد.

```
bivariate <−function(x, mu = rep(0,2), sigma = matrix(c(1,0,0,1), 2)) {
a<-(2*pi*prod(eigen(sigma)$values)^0.5)
b<-exp(-0.5*t(x-mu)%*%solve(sigma)%*%(x-mu))
return(b/a)
}
```

نکته 1 :

اگر به جای یکی از آرگمان‌های تابع سه نقطه (...) قرار دهیم، بعداً می توانیم ازهر نوع ورودی با هر مقداری به جای آن استفاده کنیم. به تابع زیر توجه کنید.

```
q <−function(x,...){
.....; return(.....)}
```

نکاتی در مورد آرگمانهای خروجی (*return*)

آرگمان‌های خروجی همان متغیرهای هستند که بعد از انجام محاسبات لازم روی آرگمانهای ورودی در قسمت برنامه‌ی تابع به دست آمده‌اند و می‌خواهیم آنها را در خروجی برنامه ببینیم. تعداد خرو جی ها ممکن است یک یا بیشتر از یکی باشد. هر خروجی می تواند به همراه اسم یا بدون اسم باشد. به مثال زیر توجه کنید.

مثال 6 :

تابعی بنویسید که یک بردار را به عنوان ورودی دریافت کند و در خروجی مجموع مجذور درایه‌های بردار را با نام square و مجموع درایه‌ها را با نام sum نشان دهد.

```
dd<-function(x) {
a<-sum(x)
b<-sum(x^2)
return(sum=a,square=b)
 }
```

در این مثال خروجی ها به همراه اسم داده می شوند.

```
dd(c(1,2,3))↵
$sum
[1] 6
$square
[1] 14
```

در این مثال اگر بخواهیم در خروجی فقط sum نشـان داده شـود، تابع را به صـورت زیر فراخوانی می‌کنیم.

```
dd(c(1,2,3))$sum↵
[1] 6
```

در مورد square نیز چنین است.

مثال 7:

برنامه‌ای بنویسـید که بردار x را به عنوان ورودی بگیرد واگر x ارزش عددی داشـت مقادیر میانگین، واریانس، میانه و مد این بردار را به ترتیب در خروجی به همراه اسم به ما بدهد.

```
ee<-function(x) {
if(is.numeric(x)!=T){
return("The entry is not numeric value") }
a<-mean(x)
b<-var(x)
d<-median(x)
e<-mode(x)
return(mean=a,varianc=b,median=d,mode=e)
 }
```

تابع را برای به دست آوردن میانگین به صورت زیر فراخوانی می‌کنیم.

```
ee(1:3)$mean↵
[1] 2
```

نکته 2 :

در بالا اشـاره کردیم که اگر خروجی ها دارای اسـم باشـند، اگر بخواهیم فقط خروجی خاصـی نشـان داده شـود وبقیه نشـان داده نشـوند از $ به همراه اسـم آن خروجی خاص بعد ازفراخوانی تابع اسـتفاده می کنیم. حال اگر خروجی ها دارای اسـم نباشـند، برای نشـان دادن خروجی خاص بعد از فراخوانی تابع برای مثال به صـورت زیر عمل می کنیم .

```
dd <-function(x) {
a <-sum(x)
b <-sum(x^2)
return(a,b)
 }

dd(c(1,2,3))$a↵
[1] 6
```

نکته 3 :

اگر برنامه‌ی تابع را در محیط script بنویسـیم با یک بار فشـار دادن کلید F10 تابع برای همیشه برای نرم‌افزار شناخته می‌شود و هر کجا احتیاج داشته باشیم می‌توانیم آن را فراخوانی کنیم.

نکته 4 :

اگر بعد از اجرای برنامه با فشـار دادن کلید F10 برنامه اجرا نشـد و متوجه شـدیم که دارای خطا است، برای اینکه بفهمیم ایراد برنامه مربوط به کدام قسمت است، می‌توانیم اجزای مختلف برنـامـه را یکی یکی بـه تنهایی و با ثابت نگه داشـتن بقیه‌ی اجزا اجرا کنیم. به این ترتیب که اگر بخواهیم قسـمتی از برنامه اجرا نشـود از عملگر (#) قبل از آن قسمت استفاده می‌کنیم.

تمرینهای فصل پنج

تمرین 1 :

تابعی بنویسید که دترمینان ماتریس x را محاسبه کند.

```
det <–function(x) { d <–prod(eign(x)$values) ; return(d) }
```

تمرین 2 :

تابعی برای پیدا کردن زاویه‌ی بین دو بردار و نشان دادن آن با واحد درجه بنویسید.

```
ss <–function(x,y){
normx <–sqrt(t(x)%*%x)
normy <–sqrt(t(y)%*%y)
f <–(t(x)%*%y)/(normx*normy)
return(alpha=(a cos(f)*180)/pi)
}
```

تمرین 3 :

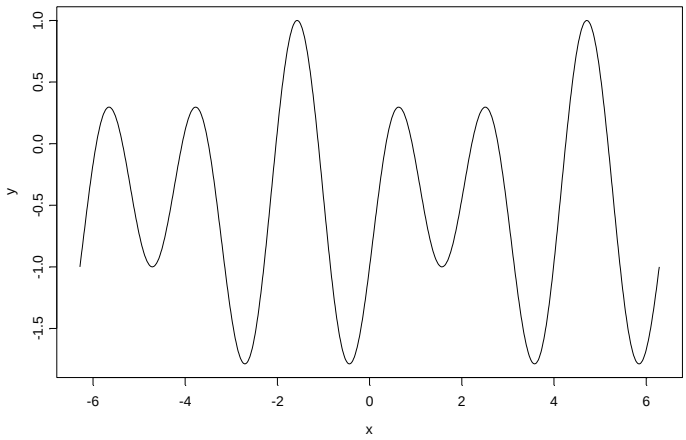
تابعی برای به دست آوردن ماکزیمم یک بردار بنویسید.

```
maxfunc <–function(x){
n <–length(x)
max <–x[1]
for(i in 2:n){
if(x[i]>max) max <–x[i]
}
return(max)
}
```

تمرین 4 :

تابعی بنویسید که نمودار و مقدار ماکزیمم مطلق تابع $y=\sin(3x)-\cos(x)^2$ را در بازه‌ی $I=[-2\pi,2\pi]$ به ما بدهد.

```
> f <–function(x) {y <–sin(3*x)–(cos(x))^2↵
+plot(x,y,type="l") ; return(max(y)) }
> x <–seq(–2*pi,2*pi,length=1000); f(x)↵
[1] 0.9999864
```



مقدار ماکزیمم و نمودار تابع بعد از فراخوانی تابع را مشاهده کردیم. عملکرد دستور plot( ) را در فصل بعد شرح می دهیم.

تمرین 5 :

تابعی بنویسید که دنباله‌ی مجموعهای جزئی سری $\sum_{n=1}^{\infty} \frac{n}{n^3-2}$ را تا جمله‌ی 60 محاسبه کند.

```
> fsigma <−function(n) {y <−n /(n^3 − 2)↵
+ Sigma <−sum(y)↵
+ return(Sigma ) }
> n <−seq(1,60,1); fsigma (n)↵
[1] -0.2761703
```

تمرین6 :

تابعی بنویسید که n عدد صحیح تصادفی بین 1 و عدد a را نمایش دهد.

```
frandom <−function(a, n) {
x <−rnorm(n); x <−floor(1000 * x); y <−x%%(a −1) +1; return(y) }
```

این تابع با توجه به این که باقیمانده‌ی تقسیم یک عدد بر a عددی بین صفر و(1- a) می باشد، نوشته شده است. این تابع در زیر 6 عدد صحیح تصادفی بین 1 و 4 را به ما می‌دهد.

```
frandom(4,6)↵
[1] 1 3 1 1 3 4
```

تمرین7 :

تابعی بنویسید که در مورد تابع درجه‌ی (3) $y = ax^3 + bx^2 + cx + d$ مقادیر اکسترمم‌های مطلق را در بازه‌ی $I = [-15,15]$ و نقطه‌ی عطف را نشان دهد.

```
f <-function(a,b,c,d,x){
y <-a*(x^3)+b*(x^2)+c*x+d
return(y) }
fextermom <-function(a,b,c,d,I=seq(-15,15,length=1000)) {
z <-f(I); ma <-max(z); mi <-min(z); at <-f(a,b,c,d,(-b/(3*a))
return(MAX=ma,MIN=mi,ATF=at) }
```

البته اگر در قسمت return خروجیها را به صورت زیر بنویسیم، به جای مقدار تابع در نقطه‌ی عطف این نقطه را به همراه طول و عرض به ما نشان می‌دهد. (توجه کنید که از حرف لاتین بزرگ C برای این متغیر استفاده کنید.)

```
return (MAX=ma , MIN=mi , ATF=c(-b/(3*a),at))
```

تمرین 8 :

تابعی برای محاسبه‌ی ضریب همبستگی اسپرمن در درس ناپارامتری بنویسید .

```
fspearman <-function(x,y){
n <-length(x)
z <-rank(x)
w <-rank(y)
d <-sum((z-w)^2)
r <-1-(6*d)/(n*(n^2-1))
return(r)
}
```

تمرین9 :

تابعی بنویسید که آماره‌ی ویلکاکسون را برای آزمون فرض زیردر مورد دو جامعه به همراه *p_value* پیدا کند. ($H_o : F_x = F_y$ و $H_1 : F_x \neq F_y$)

```
wilcoxfar <-function(x,y){
m <-length(x); n <-length(y)
z <-c(x,y)
w1 <-sum(rank(z)[(m+1):(m+n)])
E <-(m*(n+m+1))/2
v <-(m*n*(m+n+1))/12
w <-(w1-E)/sqrt(v)
p_value <-2*(1-pnorm(abs(w)))
return(w,p_value)
}
```

تمرین 10 :

تابعی بنویسید که داده‌های یک سری زمانی را گرفته و مقادیر همبستگی و همبستگیهای جزئی را برای آن محاسبه کند.

ابتدا با برنامه‌ی زیر ضـــرایب همبســـتگی یعنی $r_1, r_2, ..., r_k$ را که در آن $k = [n/4]$ می باشد و n تعداد داده ها است، محاسبه می کنیم .

```
R <-function(x){
n <-length(x); k <-floor(n/4); y <-(x-mean(x))
r <-0
for(i in 1:k) { r[i] <-(t(y[1:(n-i)])%*%y[(i+1):n])/(n-1)*var(x) }
return(r) }
```

حال تابع زیر را برای پیدا کردن ضرایب همبستگی جزئی به کار می‌بریم، که ورودی های آن بردار ضرایب همبستگی (r) و بردار مشاهدات (x) است.

```
PR < –function(r, x) {
n < –length(x); k < –floor(n / 4); f < –rep(0, k)
f[1] < –r[1]
for(l in 2 : k) {a < –b < –matrix(0, l, l)
for(i in 1 : l) {for(j in 1 : l) {if (i == j) {a[i, j] < –1}
else {a[i, j] < –r[abs(j – i)] }
b < –a }}
b[, l] < –r[1 : l]
f[l] < – det(b) / det(a) }
return(f) }
```

تمرین 11 :

تابعی که یک بردار را به صورت صعودی مرتب کند بنویسید.

```
fsort < –function(s){
n < –length(s)
t < –0
for(i in 1: (n – 1)) { for(j in n: 2) {
if (s[j] < s[j – 1]) {
t < –s[j]
s[j] < –s[j – 1]
s[j – 1] < –t
}
}}
return(s)
}
```

به فرا خوانی تابع توجه کنید.

```
masort(c(1,4,3,6,7,2))↵
[1] 1 2 3 4 6 7
```

با اندکی تغییر می‌توان تابع را برای حالتی که بردار را به صـــورت نزولی مرتب کند بنویسیم.

فصل ششم

رسم نمودار توابع توسط تابع *plot*

فرم کلی تابع *plot* و شناسه هایش به صورت زیر می باشد.

$plot(x, y, pch = "یک کاراکتر", \text{lty} = \text{یک عدد}, main = "maintitle", sub = "subtitle", xlab = "xAxislab", ylab = "yaxislab", xlim = \text{یک بازه}, ylim = \text{یک بازه}, type = "p")$

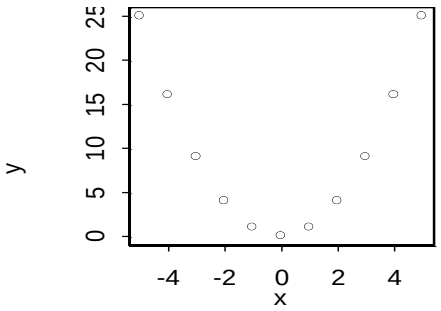
مثال 1:

$x < -5:5$

$y < -x\text{^}2$

$plot(x, y)$

در اینجا دستور *plot* نمودار نقطه‌ای ساده‌ی x را در مقابل y رسم می کند. شکل زیر را ببینید.

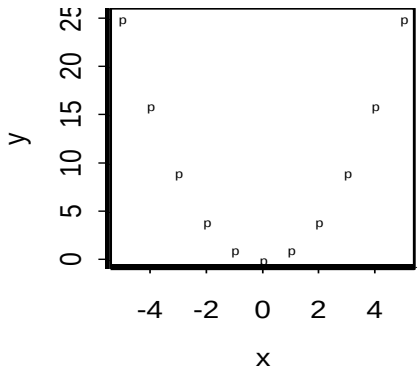


قسمت های مختلف این تابع را درمثال 1 در زیر بررسی می کنیم.

1) pch :

در این قســمت ما تعیین می کنیم که نمودار نقطه‌ای به جای نقطه که حالت پیش فرض در نظر گرفته شده است با چه کاراکتری نمایش داده شود. به مثال زیر توجه کنید.

$plot(x, y, pch = "p")$



2) type :

توسط این قسمت نوع نمودار را به صورت های زیر تعیین می کنیم.

type ="p" :

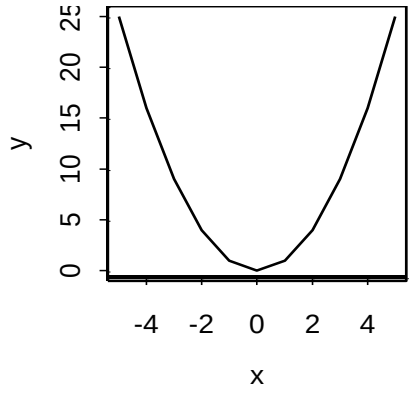
نمودار را به صورت نقطه‌ای رسم می کند که حالت پیش فرض سیستم است.

```
plot(x,y,type ="p")
```

type ="l" :

نمودار را به صورت خطوط پیوسته رسم می کند.

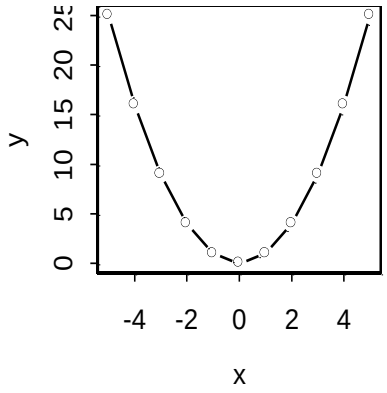
```
plot(x,y,type ="l")
```



type ="b" :

نمودار را توأماً توسط نقطه و خط رسم می کند.

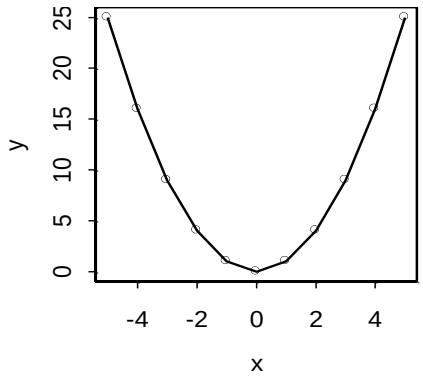
```
plot(x,y,type ="b")
```



type ="o" :

نمودار را توسط نقطه به همراه خطوط پیوسته رسم می کند.

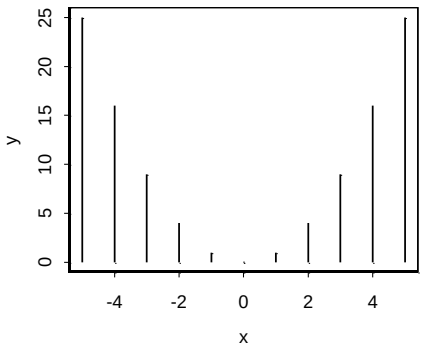
```
plot(x,y,type="o")
```



type="h":

برای رسم نمودار ستونی مورد استفاده قرار می گیرد.

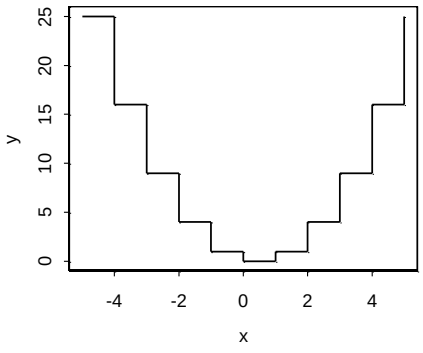
```
plot(x,y,type="h")
```



type="s":

نمودار پله‌ای را رسم می کند.

```
plot(x,y,type="s")
```



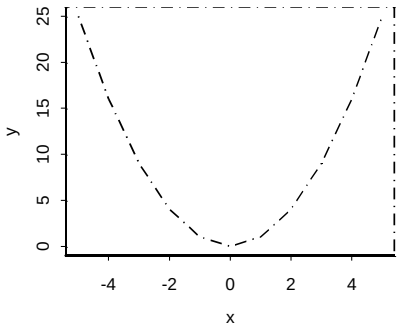
type ="n" :

چیزی را به عنوان نمودار در محورها رسم نمی کند. و یک صفحه‌ی خالی رابه ما نشان می‌دهد.

3) lty :

برای رسم نمودار با حالتهای مختلف نقطه چین استفاده می شود.

```
plot(x,y,type ="l",lty = 3)
```



به جای عدد 3 می توان هر عددی از 2 تا 8 را انتخاب کرد، که هر عدد برای حالت خاصی از نقطه چین کاربرد دارد.

4) col :

توسط این قسمت رنگ نمودار را مشخص می کنیم.

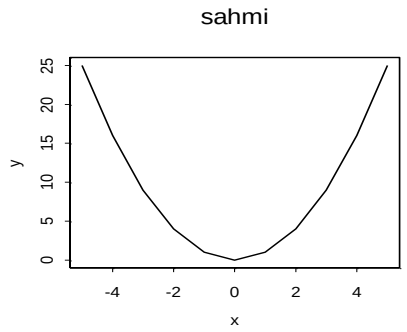
```
plot(x,y,type ="l",lty = 3,col = a)
```

a عددی است که بستگی به سیستم دارد، عددی بین 1 تا 15 یا حتی خارج از این رنج است.

5) subtitle,maintitle:

maintitle عبارت از عنوانی است که ما به نمودار می دهیم و در بالای نمودار می نویسیم و subtitle عنوانی است که ما به نمودار می دهیم و در پایین نمودار می نویسیم.

```
plot(x,y,type ="l",main="sahmi")
```

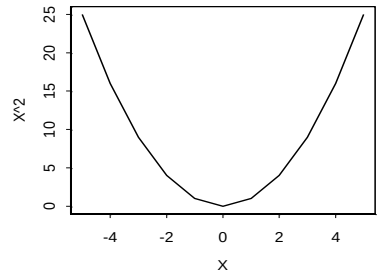


مشـاهده می کنیم که دسـتور بالا نمودار را رسـم می کند و عنوان *sahmi* را به آن می دهد. در این مثال عنوان را می توانسـتیم در *subtitle* بنویسـیم. در این صـورت عنوان در پایین نمودار قرار می گرفت.

6) lable :

هر گـاه بخواهیم بـه متغیرهـای y,x کـه بـه عنوان محورهـای افقی وعمودی در نمودار هسـتند عنوانی را نسـبت دهیم از این قسـمت در دسـتور *plot* اسـتفاده می کنیم. به مثال زیر توجه کنید.

$plot(x,y,type="o",xlab="X",ylab="X^2")$



7) lim :

توسط این دستور می توانیم محدوده‌ی تغییرا ت متغیرهای x,y را روی محورها تغییر دهیم. به مثال زیر توجه کنید.

$plot(x,y,x\lim=c(-8,8))$

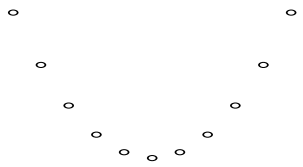
این دســـتور محدوده‌ی تغییرات x را روی محور افقی از حالت اولیه به حالتی که از عدد 8- تا عدد 8 گسـترده شـده اسـت تغییر می دهد. برای تغییر حوزه‌ی تغییرات y نیز می توان از $y\lim$ استفاده کرد.

نکته1:

اگر بخواهیم نموداری رسم کنیم که در آن محورها مشاهده نشوند می توان دستور *plot* را به صورت زیر به کار برد.

$plot(x,y,axes=F)$

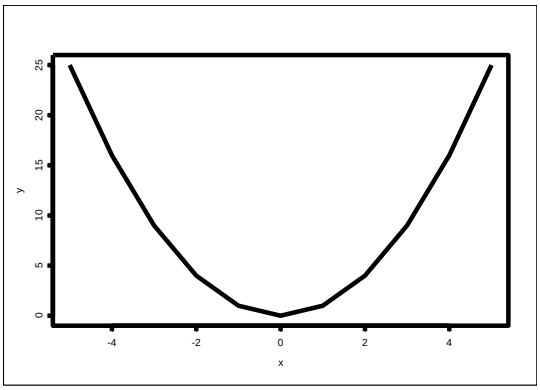
حالت پیش فرض سیستم $axes=T$ است که نمودار را به همراه محورها رسم می کند.



8) *lwd* :

اگر نمودار را با $type="l"$ رسم کنیم می توان پهنای خطوط نمودار را توسط *lwd* که کدهای 1و2و3و... را می تواند اختیار کند تعیین کرد که کد 1 حالت پیش فرض سیستم است.

$plot(x,y,type="l",lwd=10)$



مثال2:

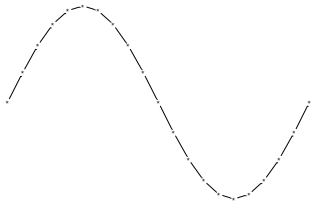
$x<-seq(0,2*pi,length=21)$

$y<-\sin(x)$

$plot(x,y,axes=F,type="b",pch="*",xlab="",ylab="")$

این مثال ابتدا فاصله ی بین 0تا 2π را به 21 قسمت تقسیم می کند ودر متغیر x قرار می دهد و $\sin(x)$را در متغیر y قرار می دهد ونمودار تابع $y=\sin(x)$ در بازه‌ی $[0,2\pi]$ رسم می شود. در اینجا علاوه بر اینکه در دستور *plot* خواسته ایم که محورها

را رسم نکند با دستور *lable* خواسته ایم که عنوان محورها یعنی (x,y)را نیز حذف کند.



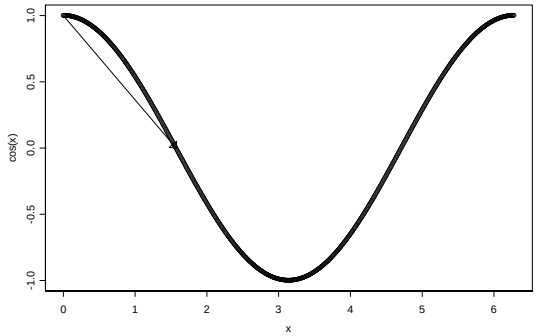
دستوراتی که بعد از رسم نمودار از آنها استفاده می کنیم

1) بعد از دستور *plot* ورسم نمودار اگر بخواهیم از مختصات (x_1,y_1) به مختصات (x_2,y_2) بر روی نمودارفلشی رسم کنیم از دستور arrows استفاده می کنیم . فرم کلی این دستور به صورت زیر است.

$arrows(x_1,y_1,x_2,y_2)$

مثال3 :

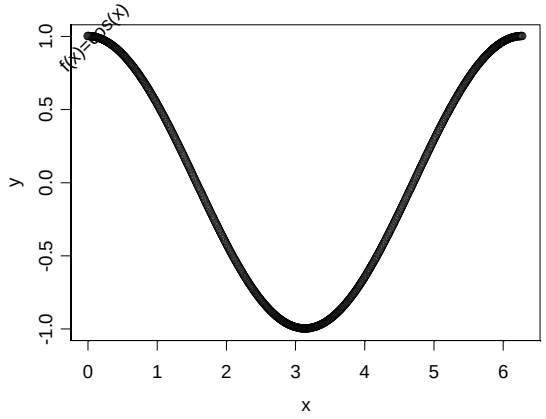
$x<-seq(0,pi,length=1000)$
$plot(x,\cos(x))$
$arrows(0,1,pi/2,0)$



2) اگر بخواهیم بر روی مختصات معینی از نمودار با زاویه‌ای مشخص عنوانی را بنویسیم از دستور text به صورت زیر استفاده می کنیم. در مورد مثال قبل این دستور را به کار برده‌ایم.

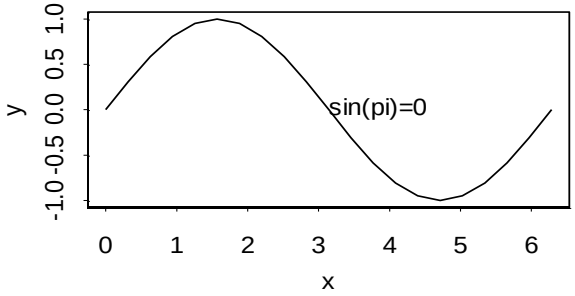
$text(a,b,"عنوان",srt=$ زاویه ی مورد نظر $,adj=0)$

$text(0,1,"f(x)=cos(x)",srt=45,adj=0)$



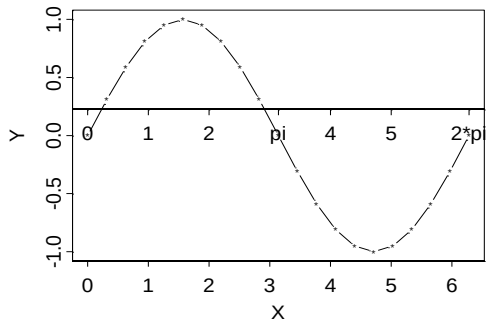
در این دستور در قسمت اول مختصات نقطه را معین می کنیم، در قسمت بعد برچسب ودر آخر با مقادیر 0یا 1 یا 0.5 در قسمت *adj* به ترتیب تعیین می کنیم که می خواهیم این برچسب در طرف راست مختصات مورد نظر یا در طرف چپ ویا در قسمت وسط قرار بگیرد. مثال 2 را در نظر بگیرید:

$text(pi,0.1,"\sin(pi)=0",adj=0)$



3) گاهی اوقات می خواهیم در نموداری که رسم کرده ایم به تعدادی از مقادیر روی محورها عنوانی(*label*) نسبت دهیم این کار را توسط دستور axis، انجام می دهیم. کاربرد آن را در مورد مثال 2 در زیر مشاهده می‌کنید.

$axis(1,at=c(0,1,2,pi,4,5,2*pi),labels=c(0,1,2,"pi",4,5,"2*pi"),pos=0)$



در این دســـتور عدد 1 معلوم می کند کـه می خواهیم *label* هـا را برای مقـادیر روی محور افقی (x) به نمودار اضـــافه کنیم، اگر به جای آن عدد 2 را قرار دهیم محور عمودی y ها و در نمودارهای ســـه بعدی عدد 3 محور ســـوم یعنی محور z ها را در نظر می گیرد. در قســـمت $at = c(\ )$ نقـاطی را کـه می خواهیم برای آنهـا روی محور عنوان تعیین کنیم مشـــخص می کنیم. ودر قســـمت $labels = c(\ )$ عنوانها را تعیین می کنیم. ودرقســـمت *pos* موقعیـت نقطـه‌ای کـه می خواهیم محوری در آن نقطـه روی نمودار رسـم شـود و عنوانها روی آن نمایش داده شـوند را تعیین می کنیم. که در این مثال عدد صفر را نوشته ایم که همان محور افقی x است.

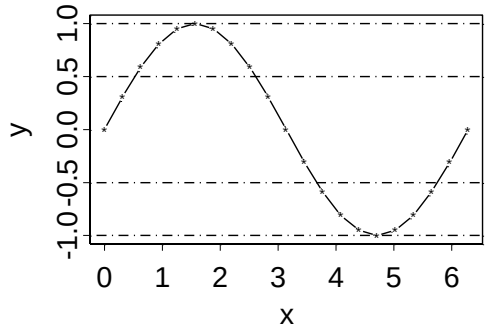
4) دستور abline

این دسـتور برای اضـافه کردن خطهای افقی وعمودی به نمودار اسـتفاده می شـود. در زیر کاربرد این دستور را در مورد مثال 2 می‌بینید.

مثال4:

$abline(h = c(-1,-.5,.5,1), lty = 3)$

در این دسـتور حرف h برای رسـم خطوط افقی وحرف v برای رسـم خطوط عمودی استفاده می شود.



در قسمت *lty* تعیین می کنیم که این خطوط به صورت نقطه چین رسم شوند.

5) دستور par( )

این دستور کاربردهای متفاوتی دارد که در زیر به آنها اشاره می‌کنیم.

اگر بخواهیم برای محورهای نمودار حاشیه تعیین کنیم از این دستور به شکل زیر استفاده می کنیم.

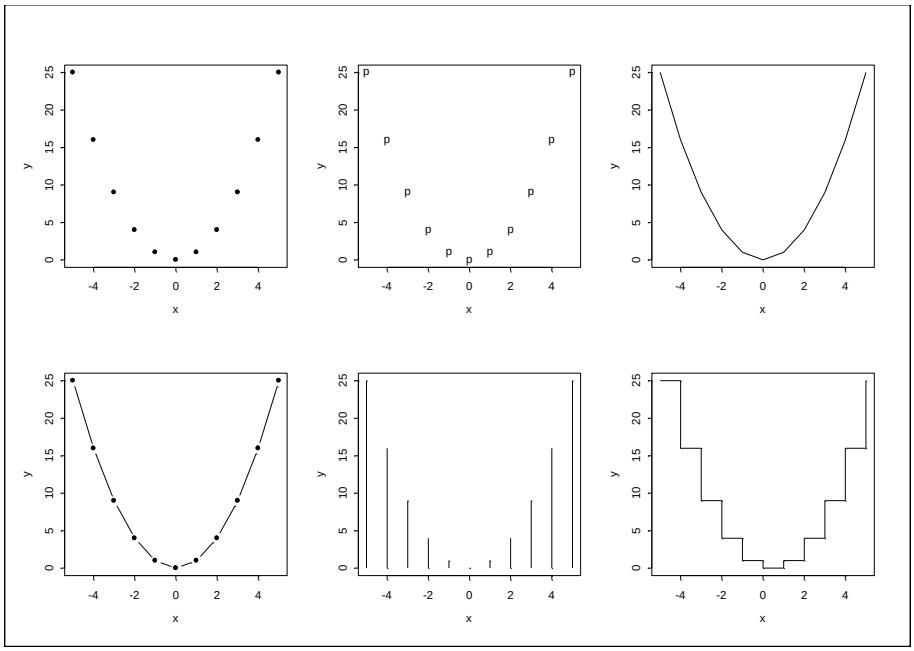
```
par(mar = c(2,2,3,3))
```

این دستور در قسمتهای افقی نمودار 2 سانتیمتر ودر قسمتهای عمودی 3 سانتیمتر حاشیه ایجاد می کند. یک کاربرد دیگر دستور *par* قرار دادن نمودار در یک مستطیل است. برای این منظور به صورت زیر از این دستور استفاده می کنیم.

```
par(pty ="s")
```

محیطی که نمودارها در آن رسم می شوند *Graphsheet* نام دارد. توسط دستور زیر می توانیم این محیط را برای رسم هم زمان چندین نمودار تقسیم بندی کنیم. درمثال زیرمحیط را به دو سطر و سه ستون برای رسم هم زمان 6 نمودار، تقسیم کرده‌ایم.

```
par(mfrow = c(2,3))
```

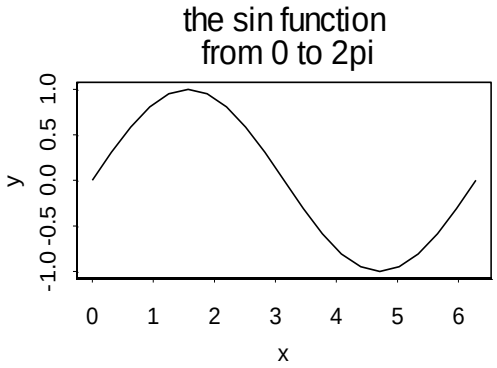


البته بعد از تقسیم بندی، دستور رسم این 6 نمودار را به ترتیب و پشت سر هم اجرا می‌کنیم.

5) دستور title

اگر در دستور *plot* فراموش کردیم که به نمودار عنوان بدهیم می توان بعد از رسم نمودار از دستور *title* به صورت زیر استفاده کرد. به مثال زیردر مورد نمودار sin توجه کنید.

title("*The* sin *function**n from* 0 *to* 2*pi*")

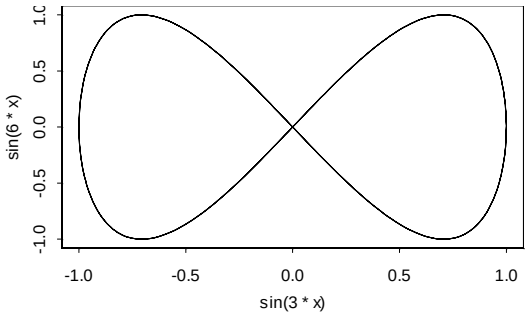


در دستور قبل (*n*\) تعیین می کند که ادامه‌ی متن در یک خط جدید نوشته شود.

مثال5:

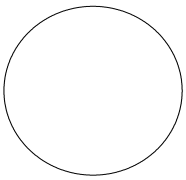
```
par(mfrow = c(2,2))
x <- -seq(0,2 * pi,length = 1000)
plot(sin(3 * x),sin(6 * x),type = "l")
```

نمودار به شکل زیر است.



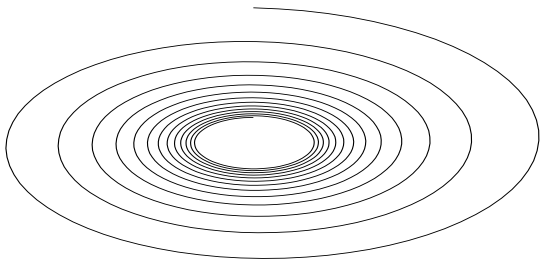
مثال 6:

```
z <- -seq(0,2 * pi,length = 1000)
x <- -sin(z)
y <- -cos(z)
par(pin(2,2))
plot(x,y,type = "l",box = F,axes = F,xlab = "",ylab = "")
title("Acycle")
```



مثال 7:

```
z <- -seq(6 * pi,32 * pi,length = 1000)
x <- -sin(z)/(0.1 * 2)
y <- -cos(z)/(0.1 * 2)
plot(x,y,type = "l",box = F,axes = F,xlab = "",ylab = "",main = "A spiral with 13 winding")
```



6) دستور locator( )

برای اضـافه کردن عنوان به یک نقطه در نمودار استفاده می شود. به مثال زیر توجه کنید.

```
text(locator(1),"max")
```

بعد از اجرای این دستور نشـانه گر موس به شـکل یک علامت + در می آید و در هر جـایی از نمودار کـه آن را ببریم عبـارت max را می نویســـد. عـدد 1 داخـل پرانتز مشخص می کند که در 1 نقطه می خواهیم این عبارت نوشته شود. اگر بخواهیم عنوان را به تعداد بیشتری نقطه نسبت دهیم از عدد دیگری استفاده می‌کنیم.

7) تشخیص نقاط پرت با دستور identify( )

اگر بعد از رسـم نمودار متوجه یک یا چند نقطه‌ی پرت شـدیم و خواسـتیم بدانیم که این نقاط پرت چندمین مشاهده هستند از تابع identify( ) استفاده می کنیم .

```
plot(x,y)
identify(x,y,n = 3)
```

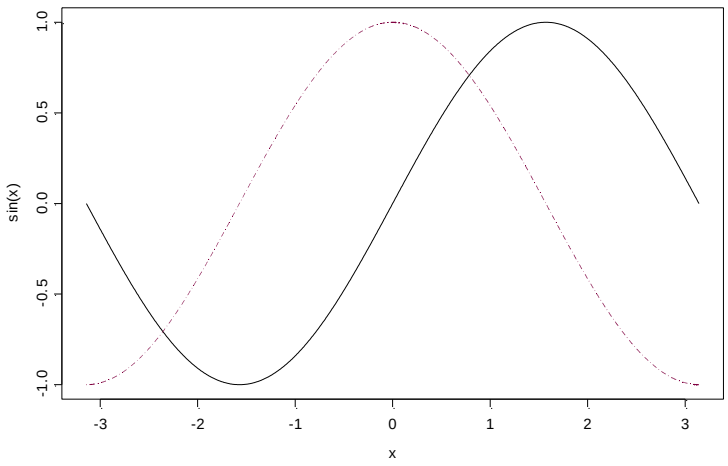
بعداز اجرای این تابع نشانه گر موس به یک علامت + تبدیل می شود، اگر این علامت را روی مشـاهده‌ی پرتی که تشـخیص داده ایم ببریم، مشـخص می کند که کدام مشـاهده است. عدد 3 تعیین می کند که 3 مشاهده‌ی پرت را می خواهیم شناسایی کنیم.

8) دستور lines( )

اگر یک منحنی رسـم کرده باشـیم و بخواهیم منحنی دیگری را در همان نمودار رسـم کنیم از این دســتور اســتفاده می کنیم. یعنی برای رســم هم زمان چند نمودار در یک صفحه به کار میرود. به مثال زیر توجه کنید.

```
x < −seq(−pi,pi,length = 1000)
plot(x,sin(x))
lines(x,cos(x),lty = 3,col = 3)
```

در این مثال ابتدا نمودار $\sin(x)$ رسم می‌شود سپس منحنی $\cos(x)$ در همان صفحه‌ی مختصات رسم می شود وبعد برای تشخیص این دو منحنی از هم حالت *lty* منحنی ها و رنگ آنها را تغییر داده ایم .



در زیر رسم بعضی از نمودارهای خاص مانند نمودار میله‌ای ونمودار دایره‌ای و غیره را شرح می‌دهیم.

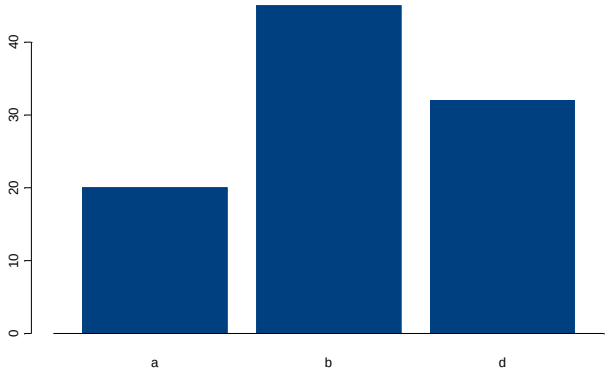
دستور *barplot( )*

برای رسم نمودار میله‌ای از آن استفاده می کنیم. به مثال زیر توجه کنید.

```
x <-c(20,45,32)
barplot(x)
```

اگر بخواهیم در نمودار برای هر ستون اسمی قرار دهیم، دستور بالا را به صورت زیر به کار می بریم.

```
barplot(x,names=c('a','b','d'))
```



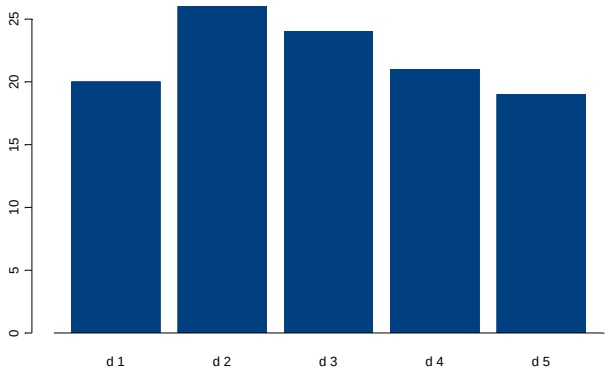
مثال 8 :

ماتریس *digit* را در نظر بگیرید برای اسم گذاری سطرها و ستونها به صورت زیر عمل می کنیم: ابتدا ماتریس را تعریف می کنیم بعد اسم سطرها و ستونها را با تابع dim*names* تعیین می کنیم .

```
digit <–matrix(c(20,26,24,21,19,15,17,16,26,13,30,70,17,20,18),5)
dimnames(digit) <–list(paste("d",1:5),paste("sample",1:3))
```

با دستورات زیر نمودار میله ای را به همراه اسم برای ستون اول این ماتریس رسم می کنیم.

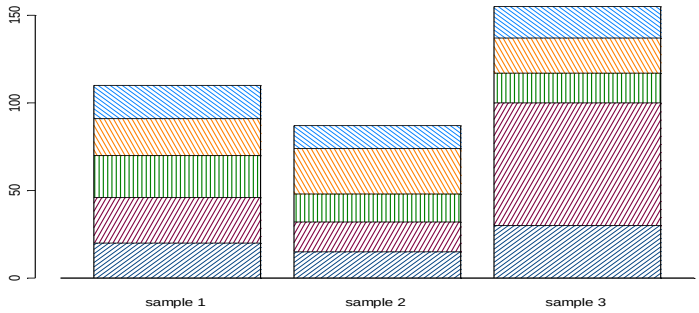
```
digitnames<-dimnames(digit)[1]
samplenames<-dimnames(digit)[2]
barplot(digit[,1],names=digitnames)
```



مثال 9 :

اگر دستور زیر را اجرا کنیم نموداری برای کل ماتریس رسم می شود. به این ترتیب که سطرهای ماتریس را به عنوان محور عمودی و ستونهای ماتریس را به عنوان محور افقی در نظر می‌گیرد.

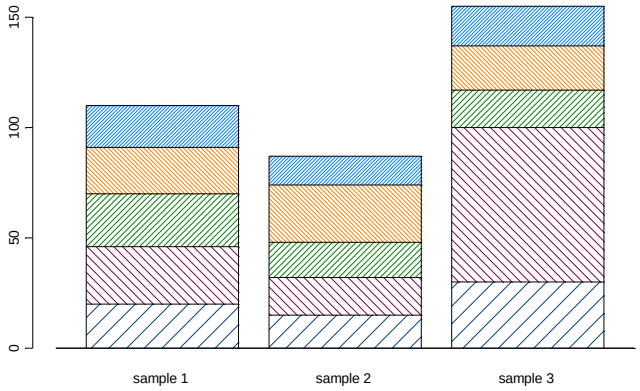
```
barplot(digit,angle=seq(45,135,len=5),density=16,names=samplenames)
```



در این دستور *angle* زا ویه های خطوط را برای هرقسمت تعیین می کند و *density* تعداد خطوط را تعیین می کند. به دستور زیر توجه کنید:

```
barplot(digit,angle=c(45,135),density=(1:5)*5,names=samplenames)
```

فرق دستور بالا با قبلی در این است که درجه ی زا ویه ها را یک در میان با 45 درجه و 135 درجه عوض می کند و تعداد خطوط در ستونها یکسان نمی باشد.

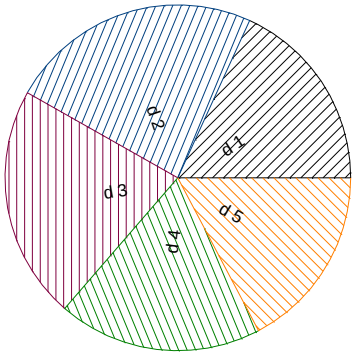


نمودار دایره‌ای

با دستور *pie( )* این نمودار را رسم می کنیم. به مثال زیر توجه کنید.

```
pie(digits[,1],names=digitnames,angle=seq(45,135,len=5),density=10)
```

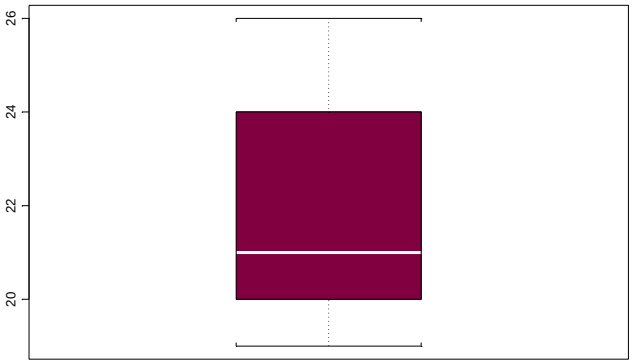
این دستور برای داده های ستون اول ماتریس نمودار دایره ای را رسم می کند.



نمودار جعبه‌ای *boxplot*

به کمک این نمودار می توان فهمید که آیا داده‌ها حول میانه متقارن هســتند ؟ این نمودار دارای 5 مؤلفه است : $\min, Q_1, median, Q_3, \max$. برای دا ده‌های مثال قبل این نموداررا رسم می کنیم.

boxplot(*digit*[,1])



دستور *hist*( ) ***برای رسم هیستوگرام***

این دستور نمودار هیستوگرام را برای داده‌های پیوسته رسم می کند. برای رسم آن باید تعداد طبقات را معلوم نمود که نباید خیلی زیاد ویا خیلی کم باشند، معمولاً تعدادشان را بین 5 تا 20 در نظر می گیرند. فرم کلی به یکی از صورت های زیر است.

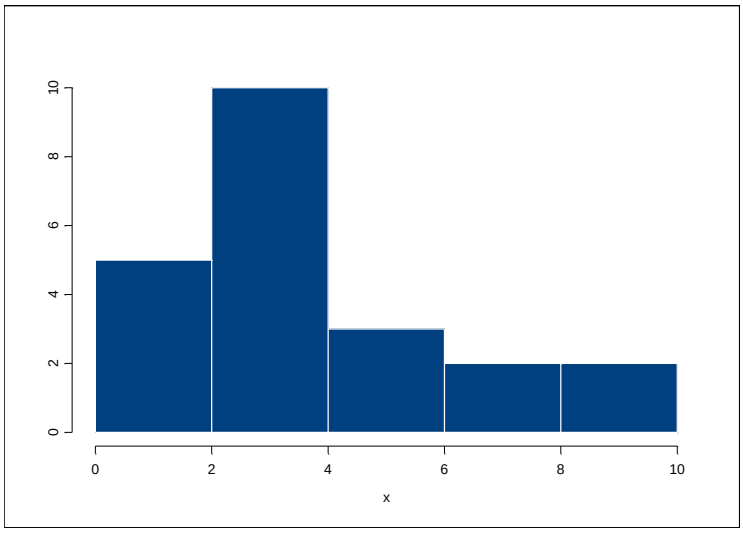
$hist(x, nclass = a)$

$hist(x, breaks = c())$

در فرم اول در قسمت *nclass* با یک عدد تعداد دسته ها را مشخص می کنیم و در فرم دوم در قسمت *breaks* برداری را قرار می دهیم که مقادیر آن مرز دسته ها است. به مثال های زیر توجه کنید.

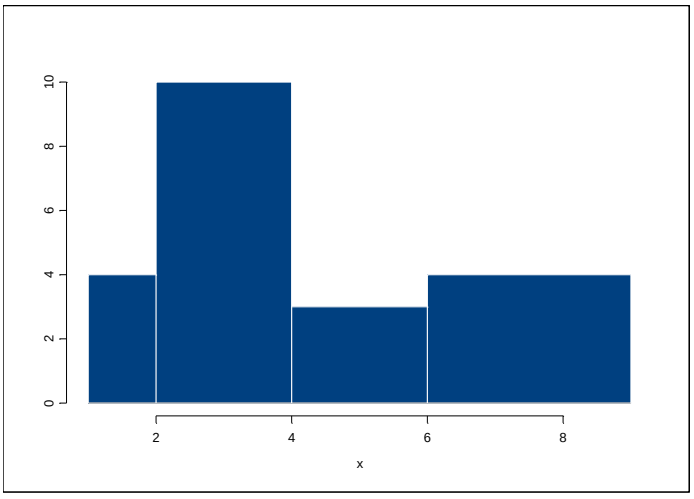
$x <- c(1.3, 2.1, 3.2, 2.5, 5.6, 4.4, 3.4, 3, 7.8, .9, 3.1, 1, 2.3, 4.6, 3.4, 2.2, 1.9, 6.5, 4, 9.2, 8.1, 1.7)$

$hist(x, nclass = 4)$



$x <- c(1.3, 2.1, 3.2, 2.5, 5.6, 4.4, 3.4, 3, 7.8, 3.1, 1, 2.3, 4.6, 3.4, 2.2, 1.9, 6.5, 4, 9, 8.1, 1.7)$

$hist(x, breaks = c(1, 2, 4, 6, 9))$

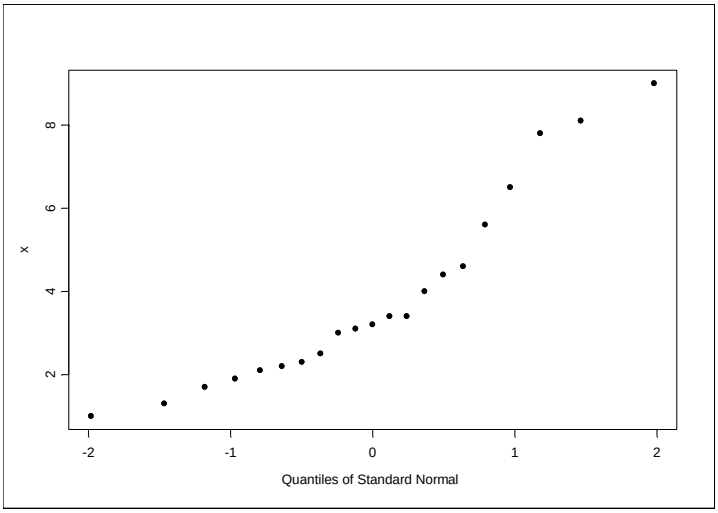


در نمودار اول تعداد مستطیلها یکی بیشتر از تعدادی است که معین کردهایم و در نمودار دوم 1 و 9 را نیز باید به عنوان مزرها ابتدایی و انتهایی باید در نظر گرفت.

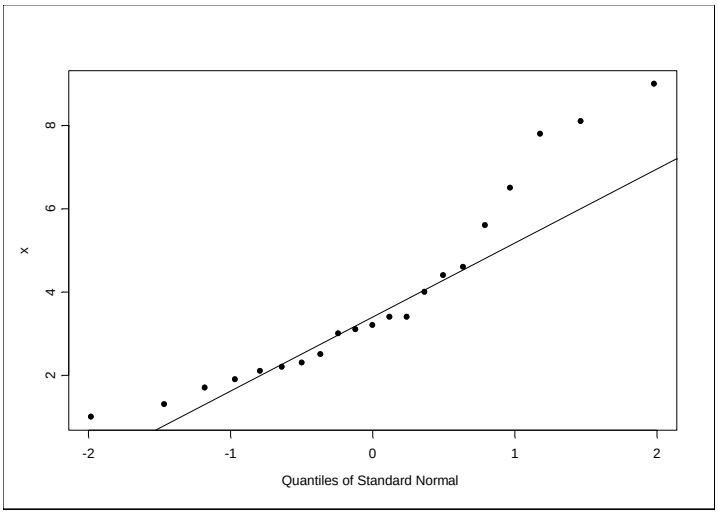
نمودار چندک – چندک برای تشخیص نوع توزیع

اگر توزیع فرض شده برای داده ها نرمال باشد برای رسم این نمودار (نمودار احتمال نرمال) از تابع *qqnorm*( ) استفاده می کنیم. همین طور می توان از دستور *qqline*( ) نیز استفاده کنیم، که یک خط بر اساس توزیع نرمال برازش می دهد، و هر چقدر نقاط نمودار به خط نزدیکتر باشند، دادهها به توزیع نرمال نزدیکتر هستند. به عنوان مثال برای داده های مثال قبل این دو نمودار را رسم می کنیم.

qqnorm(*x*)



qqline(*x*)



برای بررسی سایر توزیعها باید تابعی جهت رسم تعریف کنیم. مثلاً در مورد توزیع گاما با پارامترهای 2و3 به صورت زیر عمل می کنیم.

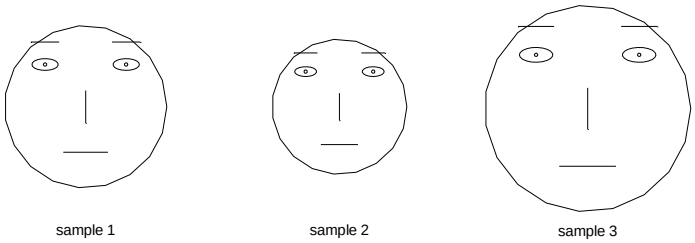
```
x <–c(...) ; x <–sort(x)
n <–length(x)
p <–((1:n)–(3/8))/(n+(1/4)) ; y <–(x–mean(x))/sqrt(var(x))
Q <–qgamma(p,2,3)
plot(y,Q)
```

نمودار چهره نما faces( )

برای نمایش مشاهدات چند متغیره از این نوع نمودار استفاده می شود در این نمودار هر صورتک متناظر بایک مشاهده است و برای مقایسه‌ی دو مشاهده تک تک اعضای صورتک با هم مقایسه می شوند. و تفاوت در اعضای چهره تفاوت در مشاهدات را به ما گزارش می‌دهد.

مثال 10 :

```
x <–digit[1,]
faces(x,labels=dimnames(digit)[[2]],ncol=3)
```

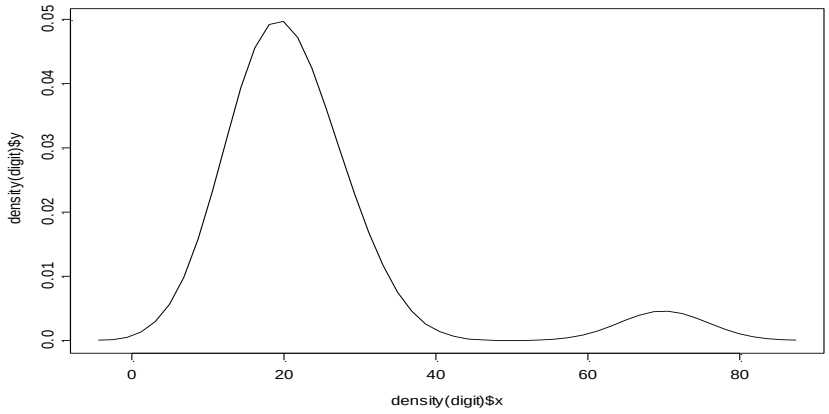


نمودار بالا نشان میدهد که صورتکها فقط از لحاظ اندازه با هم متفاوت هستند.

نمودار چگالی توزیع

در +s تابع ( )density تابع چگالی یک مجموعه داده را به دست می‌آورد. برای رسم نمودار تابع چگالی تابع ( )plot را به صورت زیر به کار می‌بریم.

```
plot(density(digit),type='l')
```



فصل هفتم
تحلیل آماری در مورد میانگینها

الف:تحلیل در مورد میانگین یک جامعه

1- الف : براورد نقطه‌ای

فرض کنیم $X_1, X_2, \ldots, X_n$ نمو نه‌ی تصادفی ما از یک جامعه باشد، دراینجا دو حالت پیش می آید. حالتی که نمونه از یک جامعه‌ی پیوسته گرفته شده باشد وحالتی که نمونه از یک جامعه‌ی گسسته گرفته شـــده باشـــد. درحالت اول دربراورد نقطه‌ای ما باید میانگین جامعه (μ) را براورد کنیم

ودرحالت دوم باید نسبت واقعی تعداد افرادی ازجامعه که دارای صفت ویژه‌ای می باشند (π) رابراورد کنیم.

2- الف : براوردفاصله‌ای

درحالتی که جامعه پیوسته باشد ما به دنبال یک بازه‌ی اطمینان برای میانگین جامعه (μ) هستیم

ودرجامعه‌ای که دارای توزیع باینومیال باشد به دنبال بازه‌ی اطمینانی برای نسبت (π) هستیم.

3- الف : آزمون فرض

زمانی که در مورد پارامتر جامعه ایده‌ی اولیه‌ای داشته باشیم وبخواهیم درستی آن را آزمون کنیم، آزمون فرض راانجام می دهیم. دراینجا فرض های ما اگرجامعه پیوسته باشد درمورد میانگین جامعه ودرجامعه‌ی گسسته در مورد نسبت است.

ب : تحلیل آماری درمورد دو جامعه

تحلیل آماری درمورد دوجامعه نیزصورت می گیرد. دراینجا ما دو نمونه‌ی تصادفی داریم که به طور مستقل ازدو جامعه گرفته شده اند.

نمونه‌ی جامعه‌ی اول : $X_1, X_2, \ldots, X_n$

نمونه‌ی جامعه‌ی دوم: $y_1, y_2, \ldots, y_m$

1- ب : براوردنقطه‌ای

که معمولاً اگردوجامعه پیوسته باشند، براوردی برای تفاضل میانگینها ی دوجامعه یعنی ($\mu_2 - \mu_1$)، به دست می آوریم واگردوجامعه گسسته باشند برای تفاضل نسبتها یعنی$(\pi_2 - \pi_1)$،براورد نقطه‌ای پیدا می کنیم.

2- ب : براورد فاصله‌ای

براوردفاصله‌ای برای ($\mu_2 - \mu_1$) یا برای ($\pi_2 - \pi_1$) به دست می آوریم.

3- ب : آزمون فرض

اگرفرض اولیه‌ای درمورد تفاضل میانگینها یا تفاضل نسبتها داشته باشیم درستی این فرضیات را آزمون می کنیم. درآزمون فرض اگرتوزیع جامعه ها معلوم باشد برای آزمون فرض از آزمونهای پارامتری استفاده می کنیم و درغیراین صورت از آزمونهای ناپارامتری استفاده می کنیم.

آزمونهای پارامتری

اگرتحلیل درمورد یک جامعه باشد وبخواهیم فرض $H_0 : \mu = \mu_0$ را درمقابل فرض مقابلش آزمون کنیم از آماره‌ی t که به صورت زیر است استفاده می کنیم.

$$t = \frac{(\bar{x} - \mu_0)}{\frac{s}{\sqrt{n}}}$$

این آماره دارای توزیع $t - student$ با $n-1$ درجه‌ی آزادی می باشد.

اگر جامعه گسسته بود برای آزمون فرض $H_0 : \pi = \pi_0$ از آماره ی z به صورت زیراستفاده می‌کنیم.

که دارای توزیع نرمال استاندارد است.

$$z = \frac{(\hat{\pi} - \pi_0)}{\sqrt{\frac{\pi_0(1-\pi_0)}{n}}}$$

اگرتحلیل درمورد دوجامعه باشد، با فرض مجهول بودن واریانسها برای آزمون فرض $H_0 : \mu_2 - \mu_1 = \delta$ دربرابر فرض مقابلش ازآماره‌های زیراستفاده می کنیم.

1) اگرواریانسهای دوجامعه برابرباشند ازآماره‌ی زیرکه داری توزیع $t - student$ با $n+m-2$ درجه‌ی آزادی است استفاده می کنیم.

$$t_1 = \frac{\left(\bar{y} - \bar{x} - \delta\right)}{s_p\sqrt{\frac{1}{n} + \frac{1}{m}}}$$

2) اگرواریانسهای دوجامعه برابرنباشند از آماره‌ی زیر که دارای توزیع $t - student$ است اما درجه‌ی آزادی پیچیده‌ای دارد استفاده می کنیم.

$$t_2 = \frac{\left(\bar{y} - \bar{x} - \delta\right)}{\sqrt{\frac{{s_1}^2}{n} + \frac{{s_2}^2}{m}}}$$

آزمونهای ناپارامتری

درتحلیل یک جامعه‌ای ازآماره‌ی من ویتنی یا ویلکاکسون که دو آماره‌ی رتبه‌ای می باشند استفاده می کنیم ودرتحلیل در مورد دوجامعه، از آماره‌ی رتبه‌ای نشانه‌ای ویلکاکسون استفاده می کنیم.

آزمونهای پارامتری وناپارامتری در یک جامعه

برنامه‌ی کامپیوتری برای تحلیل آماری وآزمون فرض پارامتری برای یک جامعه‌ی پیوسته :

دراین حالت ازتابع ( ) $t.test$ استفاده می کنیم. فرم کلی این دستور را در ادامه شرح می‌دهیم.

t.test (نام صفت، mu = عدد دلخواه, alt = "l" or "g", conf.level = عدد دلخواه)

درنام صفت همان بردار مشاهدات یک جامعه را قرار می دهیم. در mu مقدار μ_0 موجود درفرض H_0 را قرار می دهیم. درقسمت alt تعیین می کنیم که فرض مقابل یعنی H_1 به صورت دو طرفه باشد یا به صورت یک طرفه. اگراز کاراکتر "l" استفاده کنیم یعنی می خواهیم فرض مقابل به صورت $H_1 : \mu < \mu_o$ باشد. واگراز کاراکتر "g" استفاده کنیم یعنی می خواهیم فرض مقابل به صورت $H_1 : \mu > \mu_o$ باشد. اگر این قسمت را ازتابع حذف کنیم حالت پیش فرض سیستم را برای فرض مقابل در نظر می گیرد که به صورت $H_1 : \mu \neq \mu_o$ می باشد.

درقسمت conf.level سطح اطمینان مربوط به بازه‌ی اطمینان برای میانگین جامعه (μ) را تعیین می‌کنیم.

برنامه‌ی کامپیوتری برای تحلیل آماری وآزمون فرض پارامتری در یک جامعه‌ی گسسته:

دراین حالت نمونه‌ی ما به صورت زیر می باشد.

$x_1, x_2, ..., x_n \sim b(1,\pi)$ و $X = \sum_i x_i \sim b(n,\pi)$

دراینجا هر x_i فقط دو مقدار می پذیرد. می دانیم که n تعداد نمونه ها و X تعداد موفقیتها می باشد. برای تحلیل آماری از تابع ( ) binom.test به صورت زیر استفاده می کنیم.

binom.test (X, n, p=Л0, alt="l" or "g")

این تابع فقط مقدارآماره‌ی z ومقدار $p-value$ رابرای آزمون فرض پارامتری به ما می دهد. وبرای پیدا کردن فاصله‌ی اطمینان از تابع ( ) $prop.test$ به صورت زیر استفاده می کنیم.

$$prop.test(X, n, conf.level = 0.95, correct = TorF)$$

اگردرقسمت $correct$ از کاراکتر T استفاده کنیم، تعیین می کنیم که برای فاصله‌ی اطمینان تصحیح پیوستگی را اعمال کند و کاراکتر F بر عکس عمل می کند. حالت پیش فرض سیستم F است .

مثال 1 :

فرض کنید از 1000 بار پرتاب یک سکه 476 بار شیر آمده است. درسطح معنی داری 95% بیازمایید آیا سکه سالم است یا نه ؟ یک فاصله‌ی اطمینان 95%برای احتمال شیر آمدن بسازید.
حل : برای قسمت اول ازتابع زیر استفاده می کنیم.

binom . test (476,1000,0.5)

البته حالت پیش فرض سیستم در این تابع برای $p = \pi_0$ مقدار $\pi_0 = 0.5$ می باشد. بنابراین اگر در تابع مقدار .5 را ننویسیم از مقدار پیش فرض آن استفاده می کند. برای ساختن فاصله‌ی اطمینان از تابع زیر استفاده می‌کنیم.

$prop.test(476,1000,conf.level = 0.95,correct = T)$

خروجیها نتیجه می‌دهند که p _ value = 0.1372 و بازه‌ی اطمینان به صورت زیر است.
conf. level = (0.5074863, 0.4447004)

برنامه‌ی کامپیوتری برای تحلیل آماری وآزمون فرض ناپارامتری برای یک جامعه‌ی پیوسته:

برای آزمون ناپارامتری از تابع( ) *wilcox.test* استفاده می کنیم. فرم کلی این تابع به شکل زیر است.

$wilcox.test(x, mu = \text{عدد}, alt = "l" or "g", conf.level = \text{عدد}, exact = TorF, correct = TorF)$

در قسمت *exact* با کاراکتر T تعیین می کنیم که از توزیع دقیق آماره *wilcoxon* استفاده شود. حالت پیش فرض سیستم F می باشد که تعیین می کند از توزیع مجانبی آماره استفاده شود.

آزمونهای پارامتری و ناپارامتری در دو جامعه

برنامه‌ی کامپیوتری برای تحلیل آماری وآزمون فرض پارامتری در دو جامعه‌ی پیوسته:

از تابع( ) *t.test* به صورت زیر استفاده می کنیم .

$t.test(x,y,mu = \text{عدد}, paired = TorF, \text{var}.equal = TorF, alt = "l" or "g", conf.level = \text{عدد})$

درقسمت *paired* با کاراکتر T ما تعیین می کنیم که نمونه ها به صورت زوجی در نظر گرفته شوند. حالت پیش فرض آن F است که تعیین می کند نمونه ها از دو جامعه به صورت زوجی نباشند. ودرقسمت var.*equal* با کاراکتر T تعیین می کنیم فرض برابری واریانسها را می‌پذیریم وحالت پیش فرض آن T است که فرض برابری واریانسها را می پذیرد.

برنامه‌ی کامپیوتری برای تحلیل آماری وآزمون فرض ناپارامتری در دو جامعه :

از تابع زیر استفاده می کنیم. که قسمتهای مختلف آن شبیه دستورات قبلی است.

$wilcox.test(x,y,mu = \text{عدد}, paired = TorF, alt = "l" or "g", conf.level = \text{عدد})$

برنامه‌ی کامپیوتری برای آزمون فرض $H_0 : \sigma_1^2 = \sigma_2^2$ ***در مقابل*** $H_1 : \sigma_1^2 \neq \sigma_2^2$ ***در دو جامعه***

از تابع زیر استفاده می کنیم.

$$\text{var}.test(x,y,conf.level = \text{عدد}, alt = "l" or "g")$$

مثال 2 :

برای داده‌های زیر که به طور مستقل از دو جامعه گرفته شده‌اند آزمون برابری واریانسها را انجام دهید.

X	12	13	22	32	34	44	43	23	18	27	40
y	88	90	78	66	79	85	99	90	56	74	65

```
x<- c ( 12 , … ,40 )
y<- c ( 88, … , 65)
var . test (x , y , conf . level = 0.90)
```

خروجیها به صورت زیر هستند.

```
F = 0.7822, num   df = 10,  denom  df = 10, p-value = 0.7051
Alternative   hypothesis: true ratio of variances is not equal to 1
90 percent confidence interval:
 0.2626248  2.3294549
Sample  estimates:
 Variance  of x   variance of y
          132      168.7636
```

در بررسی دو جامعه ممکن است بخواهیم در مورد ارتباط بین دو متغیر نیز اطلاعاتی به دست آوریم. دو معیاری که برای بررسی ارتباط بین دو متغیر x,y به کار می روند، ضریب همبستگی و رگرسیون هستند. ضریب همبستگی (*correlation*) میزان همبستگی وارتباط را بیان می کند ولی رگرسیون نوع ارتباط را تبیین می کند.

براورد و آزمون فرض در مورد ضریب همبستگی

براورد ضریب همبستگی

ضریب همبستگی دو متغیر x,y به صورت زیر محاسبه می شود.

$$r = \frac{\sum_{1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{1}^{n}(y_i - \bar{y})^2}}$$

این محاسبه را توسط تابع ( ) cor به صورت زیر انجام می دهیم.

$$cor(x,y)$$

اگر این تابع را به صورت $cor(x,y,trim=\alpha)$ به کار ببریم، تعیین می کنیم که ابتدا میانگین پیراسته‌ی داده‌ها را با حذف $\alpha\%$ ازداده‌های ابتدا وانتهای بردارهای x,y پیدا کند بعد ضریب همبستگی میان دو متغیر را حساب کند.

آزمون فرض در مورد ضریب همبستگی

اگر آزمون فرض $H_0:\rho=\rho_0$ در مقابل $H_1:\rho\neq\rho_0$ مد نظر باشد، باید از آماره‌ی زیر که دارای توزیع نرمال استاندارد است استفاده کنیم.

$$z=\frac{0.5\ln(\frac{1-r}{1+r})-0.5\ln(\frac{1-\rho_0}{1+\rho_0})}{\sqrt{\frac{1}{n-3}}}$$

برای محاسبه‌ی این آماره تابعی به صورت زیر می نویسیم.

```
f<-function(x,y,ρh0=0,alt) {
r<-cor(x,y)
z<-(.5*log((1-r)/(1+r))-.5*log((1-ρh0)/(1+ρh0)))/sqrt(1/(n-3))
ifelse(alt=T,return(statistic=z,2*(1-pnorm(abs(z)))),return(statistic=z,1-pnorm(abs(z))))
}
```

اگر آزمون فرض $H_0:\rho=0$ درمقابل $H_1:\rho\neq0$ مد نظر باشد از تابع زیر استفاده می کنیم .

```
cor.test(x,y,alt="l"or"g",method="p"or"s"or"k")
```

درقسمت *method* با کاراکتر p ضریب همبستگی *pearson* وبا کاراکتر sضریب همبستگی *spearman* وبا کاراکتر k ضریب همبستگی*kenedal* محاسبه می شود. در خروجی مقدار آماره ومقدار $p-value$ و براورد نمونه‌ای ضریب همبستگی را می بینیم.

مثال 3 :

برای داده‌های مثال قبل آزمون کنید که آیا ضریب همبستگی بین این دو متغیر صفر است یا نه ؟

Cor. test (x , y, method= “p”)

خروجی به صورت زیر است.

```
data :  x and y
t = 0.0704,  df  = 9, p-value = 0.9454
alternative hypothesis: true coef is not equal to 0
sample estimates:
      cor
```

0.02344993

تحلیل آماری برای چند جامعه‌ی گسسته

فرض کنیم چند جامعه داریم ومی خواهیم وجود یک صفت مشخص را در آنها مورد بررسی قرار دهیم. دراینجا اگر x_i را تعداد افرادی از جامعه‌ی i ام تعریف کنیم که دارای این صفت هستند دراین صورت داریم:

$$x_i \sim b(n_i, p_i) \quad , \quad i = 1...k$$

که در آن n_i اندازه‌ی جامعه ی i ام و p_i نسبت تعداد افرادی است که در جامعه‌ی i دارای این صفت مشخص هستند. براورد ها به صورت زیر می با شند.

$$\hat{p}_i = \frac{x_i}{n_i} \quad , i = 1,2,...,k$$

می خواهیم در مورد این جامعه ها فرض ($H_0 : p_1 = p_2 = ... = p_k$) را درمقابل (خلاف: H_1) آزمون کنیم. از تابع ( ) *prop.test* به صورت زیر استفاده می کنیم .

```
x <- c(x1, x2,..., xk)
n <- c(n1, n2,..., nk)
prop.test(x,n)↵
```

اگر درمورد هر p_i یک فرض اولیه داشته باشیم و بخواهیم فرض $H_0 : p_i = p_{i0}, \ i = 1,2,...,k$ را آزمون کنیم، از این تابع به صورت زیر استفاده می کنیم .

```
x <- c(x1, x2,..., xk)
n <- c(n1, n2,..., nk)
p <- c(p1, p2,..., pk)
prop.test(x,n,p)↵
```

مثال 4 :

جدول زیر تعداد افرادی که در سه شهر الف ، ب ، ج از شوینده های نوع A, B استفاده می کنند را نشان می دهد. می خواهیم آزمون برابری نسبتها را برای سه شهر برای نسبت افرادی که از شوینده‌ی نوع A استفاده می کنند انجام دهیم.

	A	***B***	n_i
الف	232	168	400

ب	260	240	500
ج	197	303	400

$x < -c(232,260,197)$

$n < -c(400,500,400)$

$prop.test(x,n)$

خروجی این تابع مقدار $p-value=.03$ وهمچنین براورد نسبت ها ومقدار آماره‌ی آزمون $X-square=6.47$ را می دهد. که با در نظر گرفتن مقدار $\alpha=0.05$، فرض صفر را رد می کنیم.

تمرینهای فصل هفتم

تمرین 1 :

برای داده‌های زیر که مربوط به معدل دیپلم و معدل دانشگاه دانشجویان یک کلاس است با توجه به اینکه نمونه‌ها به صورت زوجی گرفته شده‌اند آزمون کنید که آیا تفاضل میانگینها برابر 8 است یا بیشتر از 8 است؟ فرض کنید که داده‌ها از توزیع نرمال باشند.

معدل دیپلم	7.87 , 17 , 17.32 , 17.24 , 18.4 , 17.5 , 15 , 15 , 15.53 , 17.20 , 16.7 , 17.41 , 13.5 , 18 , 16.89 , 16 , 18.7 , 14 , 17 , 19.36 , 19.4 , 16.5 , 19 , 14 , 19.48 , 19.26 , 18.66 , 19.33 , 18.5 , 17 , 15 , 19.07 , 18.87
معدل دانشگاه	15 , 15 , 14 , 13.5 , 15.6 , 14 , 13 , 12.5 , 13.25 , 15.72 , 12.86 , 14 , 19.25 , 14.5 , 14.8 , 14.58 , 14.3 , 13 , 14 , 17.36 , 16 , 12 , 14.8 , 15.1 , 13.6 , 17.21 , 16.5 , 17.42 , 16 , 14.5 , 13.3 , 13.15 , 15 , 14.9

ابتدا آزمون برابری واریانسها را انجام می‌دهیم.

X<- c(17.87 , 17 , … , 18.87)

Y<- c (15 , 15 , … ,14.9)

Var. test (x , y , conf .level=0.9)

با توجه به این که در خروجی مقدار pvalue = 0.3242 بنابراین فرض برابری واریانسها را می‌پذیریم. با این فرض به سراغ آزمون زیر می‌رویم.

t.test (x , y , mu= 8 , paired=T, var. equal=T, alt="l")

در خروجی داریم :

-16.1033 = مقدار آماره

33= درجه‌ی آزادی

pvalue=1

در سطح معنی داری 5 درصد، با توجه به مقدار pvalue دلیلی برای رد فرض صفر نداریم.

فصل هشتم

آزمونهای استقلال در جداول توافقی

برای آزمون استقلال از توابع زیر می توان استفاده کرد.

1)*chisq.test*(...)
2)*fisher.test*(...)
3)*mantelhaentest*(...)
4)*mcnemar.test*(...)

آماره‌ی chisq

این آماره برای آزمون استقلال دو متغیر در مورد داده‌های گسسته به کار می‌رود.

مثال1 :

برای داده های جدول زیر آزمون استقلال را انجام می دهیم. که در آن متغیرستونی عقیده‌ی فکری ومتغیر سطری میزان سواد افراد است.

	بنیاد گرا	میانه رو	آزاد اندیش
کمتر از دبیرستان	178	138	108
دبیرستان	570	648	442
دانشگاه	138	252	252

می خواهیم فرض $H_0 : \pi_{i,j} = \pi_{i.} * \pi_{.j}$ را در برابر فرض مقابلش آزمون کنیم.

$$\pi_{i.} = \sum_j \pi_{ij}$$

$$\pi_{.j} = \sum_i \pi_{ij}$$

$$x < -matrix(c(178,570,\dots,252),3) \text{ ↲}$$
$$chisq.test(x) \text{ ↲}$$

در خروجی این برنامه(69.16=مقدار آماره) و($p-value=0$) و(4= درجه ی آزادی آماره) را می‌بینیم. که با توجه به مقدار $p-value=0$ و $\alpha=.05$ فرض استقلال دو متغیر را رد می کنیم.
درچهار دستور بالا می توان آماره را با در نظر گرفتن ضریب تصحیح محاسبه کرد. به عنوان مثال برای مثال بالا می توانستیم از دستور زیراستفاده کنیم.

$$chisq.test(x, correct = T)$$

درقسمت *correct* با کاراکتر T تعیین می کنیم که برای محاسبه‌ی آماره، ضریب تصحیح را نیز در نظر بگیرد. وآماره را به صورت زیر محاسبه کند.

$$\chi^2 = \sum_{i,j} \frac{((n_{ij} - m_{ij}) - 0.5)^2}{n_{ij}}$$

آماره‌ی fisher

این آماره نیز برای آزمون استقلال به کارمی‌رود. اما در مورد جداول پیشایندی کاربرد دارد که در آنها جمع های سطری وستونی ثابت هستند.

مثال2 :

برای مثال معروف فیشر آزمون استقلال را انجام دهید.

	milk	**tea**	$n_{i.}$
milk	3	1	4
tea	1	3	4
$n_{.j}$	4	4	

```
x <–matrix(c(3,1,1,3),2)
fisher.test(x)
```

خروجی این تابع مقدار $p-value = 0.486$ را نشان می دهد. اگر $\alpha = .05$، فرض استقلال پذیرفته خواهد شد.

آماره ی mcnemar**:**

اگر نمونه های ما از دو جامعه‌ی گسسته به صورت زوج شده گرفته شده باشند ونتایج به صورت جدول زیر تنظیم شوند. وبخواهیم در مورد این جدول آزمون فرض $H_0 : \pi_{1.} = \pi_{.1}$ را در مقابل $H_1 : \pi_{1.} \neq \pi_{.1}$ انجام دهیم از این آماره که دارای توزیع کای دو با یک درجه‌ی آزادی می باشد استفاده می کنیم که به صورت زیر است.

$$z_0 = \frac{n_{12} - n_{21}}{(n_{12} + n_{21})^{1/2}} \quad \text{و} \quad z_0^{\,2} = \frac{(n_{12} - n_{21})^2}{(n_{12} + n_{21})}$$

توجه کنیم که فرضیات بالا معادل با $H_0 : \pi_{12} = \pi_{21}$ در مقابل $H_1 : \pi_{12} \neq \pi_{21}$ می باشند. این آزمون به آزمون تقارن نیز معروف است.

	موافق	مخالف	$\pi_{i.}$
مخالف	$\pi_{11}, (n_{11})$	$\pi_{12}, (n_{12})$	$\pi_{1.}$
موافق	$\pi_{21}, (n_{21})$	$\pi_{22}, (n_{22})$	$\pi_{2.}$
	$\pi_{.1}$	$\pi_{.2}$	

```
x <–matrix(c(n11,n12,n21,n22),2,byrow = T)
mcnemar.test(x,correct = T)
```

مثال 3 :

برای داده‌های جدول زیر آزمون تقارن را انجام دهید.

	موافق	مخالف
موافق	12	22
مخالف	42	34

$x < -matrix(c(12,42,22,34), 2)$ ↲

$mcnemar.test(x)$ ↲

خروجی به صورت زیر است.

McNemar's chi-square = 5.6406, df = 1, p-value = 0.0175

آماره‌ی *mantelhaen*

این آماره برای آزمون اســتقلال شــرطی در جداول توافقی به کارمی رود. به این ترتیب که آزمون اســتقلال را برای دو متغیر گســســته، در ســطوح متفاوت یک متغیر دیگر انجام می‌دهد. فرمول محاسباتی این آماره به صورت زیر است .

$$CMH = \frac{(\sum_k (n_{iik} - M_{iik}))^2}{\sum_k \text{var}(n_{iik})}$$

مثال 4:

برای داده های جدول زیر در مراکز 1 تا 8 آزمون استقلال دو متغیر دیگر را انجام دهید.

center	treatment	success	failur
1	Drag	11	25
	contor	10	27
2	Drag	16	4
	contor	22	10
3	Drag	14	5
	contor	7	12
4	Drag	2	14
	contor	1	16
5	Drag	6	1
	contor	0	12
6	Drag	1	10
	contor	0	10
7	Drag	1	4
	contor	1	8

8	Drag contor	4 6	2 1

چون باید آزمون استقلال دو متغیر را در سطوح مختلف یک متغیر دیگر انجام دهیم، ابتدا داده‌ها را به صورت یک آرایه با 8 بلوک وارد می‌کنیم. و سپس از آماره‌ی mantelhaen استفاده می‌کنیم.

$$x < -array\big(c(11,10,25,27,\dots,4,6,2,1), dim = c(2,2,8)\big) \text{ ↲}$$
$$mantelhaen(x) \text{ ↲}$$

خروجی این دستور به صورت زیر است. که شامل مقدار آماره و مقدار p_value می‌باشد.

Mantel-Haenszel chi-square = 7.7447, df = 1, p-value = 0.0054

با توجه با مقدار p_value، در سطح معنی داری 5 درصد فرض استقلال دو متغیر در مراکز مختلف رد می‌شود.

فصل نهم

رگرسیون

1) رگرسیون خطی ساده

در براورد مدلهای رگرسیونی ما با یک متغیر کنترلی (x) و متغیر پاسخ (y) که به ازای هر مقدار از متغیر x متغیری تصادفی است رو به رو هستیم. تشخیص هر مدل از روی نوع رابطه‌ای که متغیر y با متغیر x دارد صورت می‌گیرد. اگر این رابطه خطی باشد مدل خطی زیر باید به داده‌ها برازش داده شود. که به آن مدل رگرسیون خطی ساده می‌گویند.

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \qquad i = 1,2,3, \dots$$

بعد از تشخیص مدل نوبت به براورد پارامترهای مدل یعنی β_1 و β_0 , انجام آزمون فرض $H_0 : \beta_1 = 0$، براورد واریانس ضرایب، به دست آوردن مقادیر برازش شده و باقیمانده‌ها، تعیین مناسبت مدل و سرانجام پیدا کردن داده‌های پرت می‌باشد.

برای انجام دادن مراحل بالا در نرم افزار $S+$ ابتدا داده‌ها را در قالب بردارهای x و y وارد می‌کنیم. برای سادگی از دستور $scan$ برای وارد کردن داده‌ها استفاده می‌کنیم. البته می‌توان از دستور c( ) نیز استفاده کرد. سپس توسط دستور $data.frame$ این داده‌ها را در کنارهم قرار می‌دهیم و آن را به متغیری با نام دلخواه که از این به بعد به عنوان بانک اطلاعاتی ما به حساب می‌آید نسبت می‌دهیم. در آخر توسط دستور lm ( ) که فرم کلی آن را برای این مدل در زیر نشان داده‌ایم اطلاعات مختصری در مورد مدل مانند براورد ضرایب رگرسیون را به دست می‌آوریم .

$lm(y\sim x, \text{بانک اطلاعاتی})$

دقت کنید که در دستور بالا ابتدا متغیر پاسخ را با یک علامت مد به متغیر کنترلی ربط می‌دهیم سپس یک علامت کاما و بعد متغیری را که بانک اطلاعاتی را در آن قرار داده‌ایم می‌نویسیم. اگر اطلاعات مفصلی را در رابطه با مدل بخواهیم از دستور $summary$ به صورت زیر استفاده می‌کنیم.

$r < -lm(y\sim x, \text{بانک اطلاعاتی})$

$summary(r)$

به این طریق ما علاوه بر براورد ضرایب، انحراف استاندارد و کوواریانس بین آنها، باقیمانده‌ها و مقدار p_value برای آزمون فرض مورد نظر را به دست می‌آوریم. برای به دست آوردن $\hat{y}_i$ ها می توان از یکی از دو دستور زیر استفاده کنیم، که هر دو مثل هم عمل می‌کنند. توجه کنید که نام r کاملا دلخواه است.

$fitted(r)$

$predict(r)$

برای به دست آوردن باقیمانده‌ها از دستور $resid(r)$ استفاده می‌کنیم و برای تشخیص مناسبت مدل می‌توان توسط دستور $plot(\ )$ نمودار باقیمانده‌ها در مقابل براوردها را رسم کرد و تحلیل‌های لازم را انجام داد.
در آخر برای تشخیص داده‌های پرت از دستور زیر استفاده می‌کنیم.

$$lm.influence(r)$$

این دستور خروجی های زیر را برای تشخیص نقاط پرت به ما می‌دهد.

1) coefficient، در این قسمت براورد ضرایب را با حذف تک تک ردیف‌های مشاهدات برای بقیه‌ی مشاهدات به دست می‌آورد، داده‌ای پرت است که براوردهای مربوط به آن ردیف با بقیه متفاوت باشد.

2) sigma، براورد انحراف استانداردها را با حذف تک تک مشاهدات به ما می‌دهد.

3) عناصر روی قطر اصلی ماتریس hat که در محاسبات مربوط به رگرسیون بسیار مهم است. داده‌ای پرت است که عنصر متناظر با آن در اینجا با بقیه متفاوت باشد.

بعد از براورد مدل اگر بخواهیم برای مقادیر جدیدی از متغیر کنترلی متغیر پاسخ را پیش‌بینی کنیم ابتدا آن مقادیر جدید را در یک بردار با نام دلخواه مثلاً b قرار می‌دهیم و سپس دستور زیر را به کار می‌بریم.

$$predict(r, b)$$

مثال 1 :

برای داده‌های مثال 2-1 کتاب مونتگمری تمامی مراحل بالا را انجام می‌دهیم.

y: مقاومت برشی	**2158.7, 1687.15, 2316, 2061.3, 2207.5, 1708.3, 1784.7, 2575, 2357.9, 2265.7, 2156.2, 2399.55 1779.8, 2336.75, 1765.3, 2053.5, 2414.4, 2200.5, 2654.2, 1753.7**
X:سن عامل	**15.5, 23.75, 8, 17, 5.5, 19, 24, 2.5, 7.5, 11, 13, 3.75, 25, 9.7, 22, 18, 6, 12.5, 2, 21.5**

x<-scan( ) ↲
1: 15.5 ↲
2: 23.75 ↲
⋮
20: 21.5 ↲↲
21: ↲
$y < -scan(\ )$ ↲
1: 2158.70 ↲
2: 1678.50 ↲

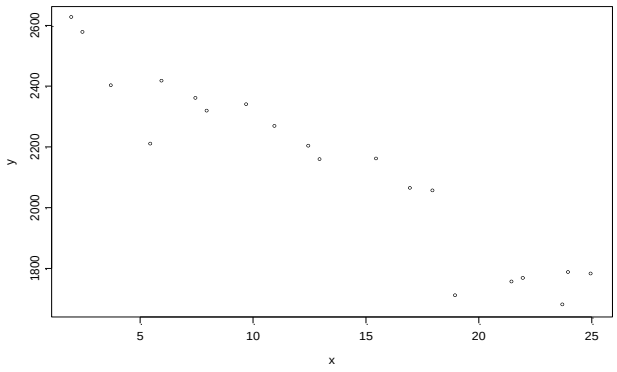
⋮
20: 1753.70 ↲↲
21:↲
$far < -data.frame(x, y)$ ↲

ابتدا داده‌ها را وارد کردیم و آنها را توسط دستور $data.frame$ در کنار هم قرار دادیم. از این به بعد متغیر far به عنوان بانک اطلاعاتی ما به حساب می آید. در ادامه از دستوراتی که شرح داده‌ایم استفاده می‌کنیم.

برای تشخیص نوع رابطه‌ی بین این دو متغیر اولین گام رسم نمودار است، که با دستور $plot$ در ادامه آن را رسم کرده‌ایم.

با توجه به نمودار زیر یک رابطه‌ی خطی با شیب منفی بین این دو متغیر وجود دارد. با تشخیص این رابطه به براورد پارامترهای مدل رگرسیون خطی می‌پردازیم.

$plot(x, y)$ ↲



$reg < -lm(y \sim x, far)$ ↲
$regres < -summary(reg)$ ↲
$regres$ ↲

دستورات بالا خروجی‌هایی به صورت زیر به ما نشان می‌دهند.

Coefficients:

	Value	Std. Error	t value	Pr(>|t|)
(Intercept)	2628.0397	44.2465	59.3955	0.0000
x	-37.1699	2.8932	-12.8473	0.0000

010 F-statistic: 165.1 on 1 and 18 degrees of freedom, the p-value is 1.67e-

Correlation of Coefficients:

(Intercept)

x -0. 8737

در قسمت **Coefficients** ما مقادیر براورد ضرایب رگرسیون، انحراف استاندارد هر کدام و همچنین آماره‌های $t\_student$ به همراه مقادیر p_value برای آزمونهای t در مورد هر کدام از ضرایب داده شده است. مقدار همبستگی بین β_0 و β_1 را در قسمت Correlation of Coefficients**s**می‌بینیم. در این خروجی‌ها همچنین مقدار آماره‌ی F به همراه p_value برای آزمون فرض $H_0: \beta_1 = 0$ داده شده است که با توجه به مقدار خیلی کوچک p_value فرض H_0 رد می‌شود.

در ادامه باقیمانده‌ها و $\hat{y}_i$ ها را به دست می‌آوریم و سپس برای تشخیص کفایت مدل نمودار آنها را در مقابل هم رسم می‌کنیم.

$re < -resid(reg)$ ↲

re ↲

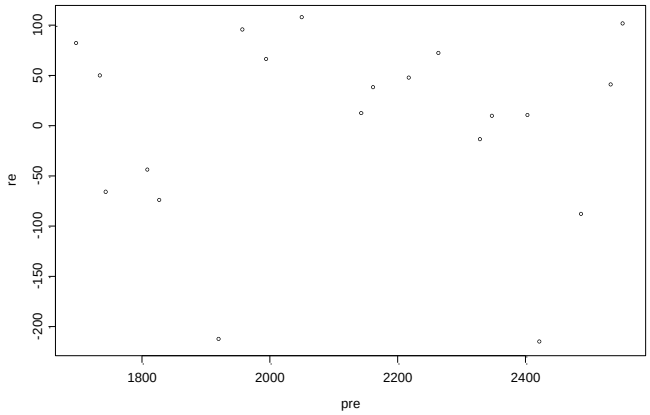
```
        1         2         3         4         5         6         7         8         9
10
106.7931  -67.1056  -14.68086  65.14786  -216.1055  -213.5124  48.73686  39.88492  8.63421
46.52871
       11        12        13        14        15        16        17        18        19
   20
 11.36843  -89.10276  81.00672  71.11639  -45.00286  94.51771  9.379424  37.0835  100.5  -
75.18778
```

$pre < -predict(reg)$ ↲

pre ↲

```
  1         2         3         4         5         6         7         8         9        10
2051.907  1745.256  2330.681  1996.152  2423.606  1921.812  1735.963  2535.115  2349.266
2219.171
  11        12        13        14        15        16        17        18        19        20
 2144.832  2488.653  1698.793  2265.634  1810.303  1958.982  2405.021  2163.417  2553.7
1828.888
```

$plot(pre, re)$ ↲



با توجه به پراکندگی تصادفی نقاط در اطراف یک خط افقی می‌توان مناسبت مدل را نتیجه گرفت. اگر مناسبت مدل تأیید نمی‌شد ممکن بود دلیل آن وجود داده‌ی پرت باشد. در ادامه از دستوری که قبلاً معرفی کردیم برای تشخیص داده‌های پرت استفاده می‌کنیم.

$lm.influence(reg)$ ↵

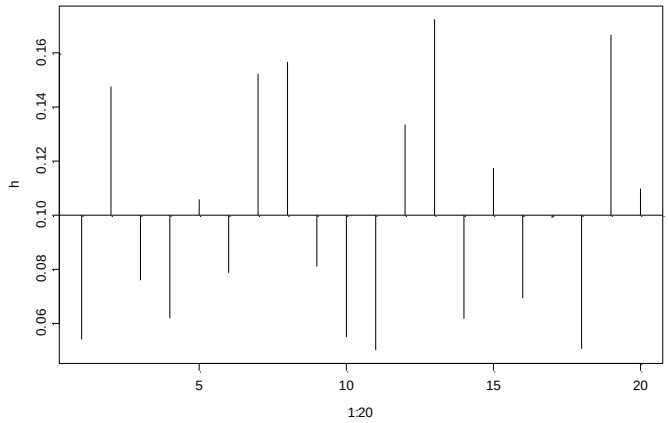
```
coefficients$ :
      (Intercept)        x
1    2619.671   -37.08162
2    2616.335   -36.10416
3    2623.855   -36.92227
4    2621.999   -37.09144
5    2656.820   -38.54795
6    2618.229   -35.67979
7    2626.769   -37.39432
8    2612.814   -36.34116
9    2620.854   -36.79297
10   2618.372   -36.75165
11   2621.680   -36.85733
12   2638.646   -37.71036
13   2631.057   -37.86470
14   2615.066   -36.60526
15   2619.583   -36.45353
16   2623.033   -37.28780
17   2620.396   -36.76472
18   2619.951   -36.82991
19   2605.462   -35.93149
20   2618.292   -36.23234
:$sigma
     1        2        3        4        5        6        7        8        9       10
93.68214 95.76093 97.40823 96.06021 80.83884 81.1148 96.66337 96.73297 97.40482 96.68943
    11       12       13       14       15       16       17       18       19       20
97.39811 94.92351 95.15312 95.68699 96.72016 94.50531 97.39374 96.97065 95.36482 95.46297

:$hat
[1] 0.05412893  0.14750959  0.07598722  0.06195725  0.10586587  0.07872092
[7] 0.15225968  0.15663134  0.08105925  0.05504393  0.05011875  0.13350221
[13] 0.17238964  0.06179345  0.11742196  0.06943538  0.09898644  0.05067227
[19] 0.16667373  0.10984216
```

با توجه به خروجی‌های بالا نمی‌توان گفت که در داده‌ها داده‌ی پرت موجود است. برای تشخیص داده‌های پرت یک روش دیگر رسم نموداری است که در مورد این مثال با دستورات زیر رسم می شود.

$h < -lm.influence(reg)\$hat$ ↵

$plot(1:20, h, type = \text{n})$ ↲
$abline(h = mean(h))$ ↲
$segments(1:20, h, 1:20, mean(h))$ ↲



داده‌ای که شاخک مربوط به آن در نمودار بالا از بقیه خیلی دور باشد داده‌ی پرت محسوب می‌شود. چون در این نمودار شاخکی که از بقیه خیلی دور باشد وجود ندارد پس می‌توانیم بگوییم داده‌ی پرتی وجود ندارد. که نتیجه‌ای مشابه نتیجه‌ی قبلی است. با توجه به تحلیلهای بالا مدل برازش داده شده به شکل زیر است.

$$y_i = \ 2628.0397 + \ -37.1699 \quad x_i \qquad i = 1,2,3, \dots$$

2) رگرسیون خطی چند گانه

در رگرسیون خطی چند گانه ما با متغیرهای کنترلی $X_1, X_2, \dots, X_k$ ومتغیر پاسخ y که به ازای هر مقدار ثابت از بردار $X_1, X_2, \dots, X_k$ یک متغیر تصادفی است، رو به رو هستیم و می‌خواهیم مدل خطی زیر را براورد کنیم.

$$y_i = \ \beta_0 + \beta_1 \, x_{1i} + \cdots + \beta_k x_{ki} + \varepsilon_i \qquad i = 1,2,3, \dots$$

دستورات مربوط به این نوع رگرسیون کاملاً مانند رگرسیون خطی ساده است. فقط باید توجه کنیم که در اینجا در دستور $lm( )$ همه‌ی متغیرهای کنترلی $X_1, X_2, \dots, X_k$ باید در نظر گرفته شوند. در حالت چندگانه این دستور را به صورت زیر به کار می‌بریم.

$lm(y \sim x_1 + x_2 + \cdots + x_k \text{, بانک اطلاعاتی})$

مثال 2 :

برای داده‌های جدول زیر مدل رگرسیون خطی چند گانه را برازش دهید.

X_1	70و65و63و72و81و83و66و75و80و75و79و76و76و69و75و87

x_2	8.3و8.6و8.8و10.5و10.7و10.8و11و11و11.1و11.2و11.3و11.4و11.4و11.7و12و26
y	10.3و10.3و10.2و16.4و18.8و19.7و15.6و18.2و22.6و15.9و24.2و21و21.4و21.3و19.1و77

$y <- scan(\ \ )$ ↲

1: 10.3 ↲

2: 10.3 ↲

⋮

16: 77 ↲

$x1 <- scan(\ \ )$ ↲

1: 70 ↲

⋮

16: 87 ↲

$x2 <- scan(\ \ )$ ↲

1: 8.3 ↲

⋮

16: 26 ↲

$far <- data.frame(y, x1, x2)$ ↲

$a1 <- lm(y \sim x1 + x2, far)$ ↲

$a2 <- summary(a1)$ ↲

$a2$ ↲

خروجیها را در زیر مشاهده می‌کنید.

Coefficients:

	Value	Std. Error	t value	Pr(>|t|)
(Intercept)	-34.5399	6.7412	-5.1237	0.0002
x 1	0.1814	0.1042	1.7410	0.1053
x2	3.6474	0.1754	20.7926	0.0000

F-statistic: 380.9 on 2 and 13 degrees of freedom, the p-value is 2.888e-012

Correlation of Coefficients:

	(Intercept)	x1
x1	-0.9676	
x2	0.3989	-0.6088

خروجیهای بالا مانند خروجیهای رگرسیون خطی ساده است. که شامل براورد ضرایبβ_0، β_1 وβ_2، انحراف استاندارد این براوردها، آماره‌های $t\_student$، مقدار آماره‌ی F و همچنین در قسمت آخر همبستگی بین این براوردها را داریم. حال به محاسبه‌ی باقیمانده‌ها و $\hat{y}_i$ ها و همچنین پیدا کردن داده‌های پرت می‌پردازیم.

$r <- resid(a1)$ ↲

r ↲

1	2	3	4	5	6	7	8	9	10
1.867251	1.680083	1.21342	-0.4199417	-0.3821445	-0.9097139	-1.955185	-0.9878997	2.140293	-4.017388

11	12	13	14	15	16
3.192217	0.1717117	0.5717117	0.6473692	-3.735339	0.9235557

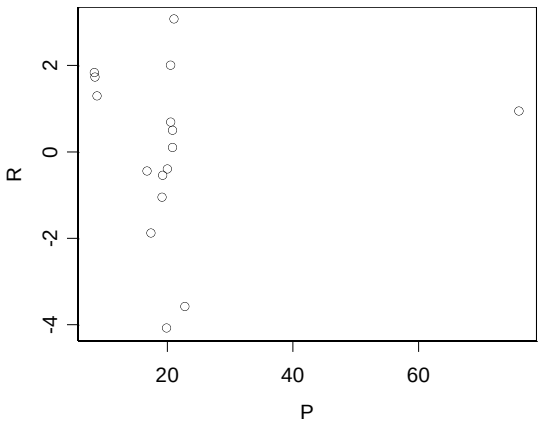
$p < -fitted(a1)$ ↲

p ↲

1	2	3	4	5	6	7	8	9	10
8.432749	8.619917	8.98658	16.81994	19.18214	19.90971	17.55518	19.1879	20.45971	19.91739

11	12	13	14	15	16
21.00778	20.82829	20.82829	20.65263	22.83534	76.07644

$plot(p, r)$ ↲



با توجه به نمودار بالا می‌توان به وجود داده‌ی پرت پی برد. که ممکن اســـت با حذف آن مدل برای بقیه‌ی داده‌ها مناسب باشد. در ادامه می‌خواهیم داده‌ی پرت را پیدا کنیم.

$lm.influence(a1)$ ↲

hat $:

[1] 0.11122896 0.19680176 0.26960263 0.07200670 0.19555223
0.26940325
[7] 0.20931297 0.06704944 0.14868628 0.06520721 0.11736614 0.06960489
[13] 0.06960489 0.13576136 0.06315247 0.93965882

sigma$:

1	2	3	4	5	6	7	8	9	10
2.166007	2.173854	2.202387	2.236663	2.236822	2.219032	2.148396	2.220659	2.137777	1.892018

11	12	13	14	15	16
2.014049	2.239611	2.233657	2.231163	1.943551	1.959729

coefficients $:

	(Intercept)	x1	x2
1	-35.34952	0.1868522	3.670956
2	-36.94430	0.2128484	3.641559

3	-36.97084	0.2149654	3.632580
4	-34.39690	0.1799707	3.646813
5	-35.00735	0.1896910	3.637142
6	-36.11788	0.2086492	3.615293
7	-31.45592	0.1357893	3.687873
8	-34.61744	0.1842118	3.641859
9	-32.53714	0.1459091	3.689217
10	-34.76708	0.1907079	3.630501
11	-32.26309	0.1405529	3.694046
12	-34.50682	0.1806697	3.648365
13	-34.42977	0.1789388	3.650521
14	-35.19781	0.1912261	3.637107
15	-34.42583	0.1823735	3.652913
16	-29.85308	0.2620290	2.644269

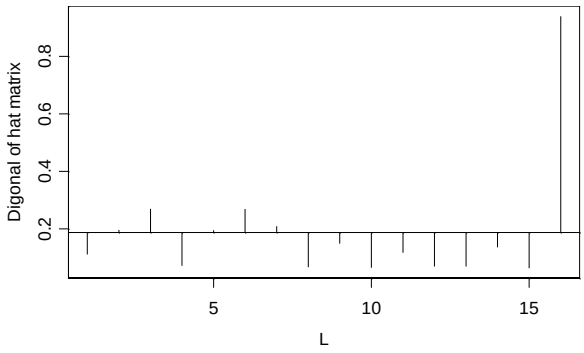
با توجه به خروجیهای بالا می‌توان نتیجه گرفت که داده‌ی شانزدهم داده‌ی پرت است. می‌توان نموداری را که برای پیدا کردن داده‌ی پرت در مثال قبل رسم کردیم، برای این مثال نیز رسم کنیم. در این مثال دستورات مربوط به رسم این نمودار به صورت زیر است.

$h < -lm.influence(a1)\$hat$ ↲

$plot(1:16, h, type = \text{n})$ ↲

$abline(h = mean(h))$ ↲

$segments(1:16, h, 1:16, mean(h))$ ↲



نمودار نیز نشان می‌دهد که داده‌ی شانزدهم داده‌ی پرت است.

پیش‌بینی

بعد از براورد مدل ما می‌توانیم پیش‌بینی انجام دهیم. به این صورت که ابتدا مقادیر متغیر کنترلی را که می‌خواهیم برای آنها پیش‌بینی انجام دهیم در یک بانک اطلاعاتی جدید قرار می‌دهیم و بعد با دستور $predict$ مقادیر پیش‌بینی‌ها را به دست می‌آوریم. مثالا اگر بانک اطلاعاتی جدید ما در متغیری با نام new قرار گرفته باشد و بخواهیم علاوه بر مقادیر پیش‌بینی شده براورد انحراف استانداردها را نیز داشته باشیم از دستور $predict$ به شکل زیر استفاده می‌کنیم. توجه کنید که

بانک اطلاعاتی در این حالت شامل مقادیر جدید متغیرهای کنترلی است که با دستور data.frame در کنار هم قرار گرفته‌اند.

$$predict(a1, new, se.fit = T)$$

3) رگرسیون لجستیک

مدلهای رگرسیون معمولی در مواردی که متغیر پاسخ، متغیری کیفی با دو سطح باشد قابل استفاده نیستند. در این گونه موارد از مدل رگرسیون لجستیک استفاده می‌کنیم. این مدل به صورت زیر است.

$$\log\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k + \varepsilon$$

در این مدل $log\left(\frac{\pi}{1-\pi}\right)$ " لگ بختها " نامیده می‌شود که π مقدارمورد انتظارمتغیر پاسخ، یا همان احتمال موفقیت است. برای براورد پارامترهای این مدل از تابع $glm()$ استفاده می‌کنیم. مانند مدل خطی ابتدا مقادیرمتغیرهای کنترلی و همچنین مقادیر مربوط به متغیر پاسخ که شامل کدهای صفر و یک است را وارد می‌کنیم و در ادامه از دستور $glm()$ به صورت زیر استفاده می‌کنیم.

$$glm(y{\sim}x1 + x2 + \cdots + xk, family = binomial)$$

در این دستور متغیر y همان متغیر پاسخ است و در قسمت $family$ حالت $binomial$ مشخص می‌کند که پارامترهای مدل رگرسیون لجستیک را براورد کند. این دستور خروجی کوتاهی به ما می‌دهد. اگر خروجی مفصل‌تری بخواهیم می‌توانیم دستور $summary$ را به همان شکلی که در مدل خطی به کار بردیم در اینجا نیز به کار ببریم.

نکته 1 :

اگر داده‌ها به همراه وزن یا همان فراوانی در دست ما باشند، ابتدا وزنها را نیز در یک متغیر وارد می‌کنیم و سپس این متغیر را در دستور $glm()$ در قسمت $weights$ قرار می‌دهیم. در این صورت با حالت زیر روبه رو هستیم.

$$glm(y{\sim}x1 + x2 + \cdots + xk, weigths = \text{متغیر وزنها}, family = binomial)$$

برای به دست آوردن باقیمانده‌ها، $\hat{y}_i$ ها، نقاط پرت و بررسی کفایت مدل مانند مدل‌های خطی عمل می‌کنیم. در ادامه دو مثال برای یادگیری بهتر دستورات بالا ذکر می‌کنیم.

مثال 3 :

برای داده‌های مثال 6-6 کتاب مونتگمری که نتایج مربوط به آزمون آتش 25 موشک ضد هوایی زمین به هوا با سرعت متغیر را ارائه می‌دهد، مدل لجستیک را برازش می‌دهیم. نتیجه‌ی هر آزمون آتش یا اصابت $(y_i = 1)$ است.

سرعت هدف	400, 220, 490, 410, 500, 270, 200, 470, 480, 310, 240, 490, 420, 330, 280, 210, 300, 400, 230, 430, 460, 220, 250, 200, 390

اصابت یا عدم اصابت	0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0

$x < -scan( )$ ↲
$1:400$ ↲
⋮
$25:390$ ↲↲
$y < -scan( )$ ↲
$1:0$ ↲
⋮
$25:0$ ↲↲
$a < -glm(y \sim x, family = binomial)$ ↲
$summary(a)$ ↲

خروجی‌ها به صورت زیر هستند.

Coefficients:

	Value	Std. Error	t value
(Intercept)	5.56027780	2.040337835	2.725175
X	-0.01566193	0.005589564	-2.801994

Residual Deviance: 22.82128 on 23 degrees of freedom
Correlation of Coefficients:
(Intercept)
x -0.9661769

با توجه به خروجیهای بالا که مشابه خروجیهای مدل خطی است مدل براورد شده‌ی ما به شکل زیر است.

$$\text{Ln}\left(\frac{\widehat{\pi}}{1-\pi}\right) = 5.5603 - 0.01566\, x$$

مثال 4 :

برای بررسی اثر نژاد و نوعی دارو(Azt) بر روی بیماری ایدز نمونه‌ای به صورت جدول 1 داریم. مدل رگرسیون لجستیک را برای این داده‌ها برازش دهید.
در این مثال ما با دو متغیر کنترلی نژاد و دارو رو به رو هستیم. و متغیر پاسخ ما به همراه وزنها که همان فراوانی‌های مشاهده شده است داده شده‌اند. داده‌ها را برای راحتی کار در جدولی دسته بندی کرده‌ایم

نژاد	دارو	بیماری(خیر)	بیماری(بلی)
سفید	بلی	14	93
سفید	خیر	32	81
سیاه	بلی	11	52

سیاه	خیر	12	43

جدول 1

در این مدل x_1 را متغیر کنترلی نژاد و x_2 را متغیر کنترلی دارو در نظر گرفته‌ایم و متغیر پاسخ ما بیماری است. ابتدا داده‌ها را به صورت جدول زیر دسته بندی می‌کنیم.

نژاد	Azt		
1	1	1	14
1	1	0	93
1	0	1	32
1	0	0	81
0	1	1	11
0	1	0	52
0	0	1	12
0	0	0	43

داده‌ها را به صورت زیروارد می‌کنیم .

$Race < -c(1,1,1,1,0,0,0,0)$

$Azt < -c(1,1,0,0,1,1,0,0)$

$w < -c(14,93,32,81,11,52,12,43)$

$y < -c(1,0,1,0,1,0,1,0)$

حال به برازش مدل می‌پردازیم. برای برازش مدل از تابع $glm( )$ استفاده می‌کنیم.

$aa < -glm(y \sim Race + Azt, weigths = w, family = binomial)$ ↵

$summary(aa)$ ↵

این دستورات خروجی شامل براورد ضرایب، انحراف استانداردهای آنها، مقادیر t_value برای آزمون معنی‌دار بودن هر یک از ضرایب و مقادیر کوواریانس بین ضرایب را به ما می‌دهد. که به صورت زیرهستند.

Coefficients:

	Value	Std. Error	t value
(Intercept)	3.140185e-016	1.224745	2.563950e-016
Rase	-3.140185e-016	1.414214	-2.220446e-016
Azt	-3.140185e-016	1.414214	-2.220446e-016

Correlation of Coefficients:

	(Intercept)	Rase
Rase	-0.5773503	

Azt -0.5773503 0.0000000

بنابراین مدل براورد شده به صورت زیر می‌باشد.

$$\text{Ln}\left(\widehat{\frac{\pi}{1-\pi}}\right) = (3.140185e-016) - (3.140185e-016)x1 - (3.140185e-016)x2$$

برای پیدا کردن باقیمانده‌های مدل و $\hat{y}_i$ ها مانند قبل از توابع زیر استفاده می‌کنیم .

$aa\$resid$

$aa\$fitted$

برای پیدا کردن داده‌های دور افتاده از دستور $lm.influence(aa)$ مانند قبل استفاده می‌کنیم. در این مثال داده‌ی پرت نداریم .

نکته 2 :

برای برازش مدلهای رگرسیون می‌توان از منوی $statistic$ نرم افزار نیز استفاده کرد. برای این منظور مسیر زیر را طی می‌کنیم.

$statistic$ منوی $\longrightarrow regresion$

تمرینها فصل نهم

تمرین 1 :

داده‌های مثال 3-3 کتاب رگرسیون بازرگان لاری را در نظر بگیرید، که در آن متغیر پاسخ (y) تعداد بازدید کنندگان، x_1 متوسط مسافتی که اسکی می‌کنند و x_2 ظرفیت بالا بر، بر حسب شمار اسکی بازان است. آزمون فرضهای زیر را انجام دهید.

الف : $\{H_0: \beta_1 = \beta_2 = 0$ ب: $H_0: \beta_2 = 0\}$

X_1	10.5	2.5	13.1	4	14.7	3.6	7.1	22.5	6.4	17
X_2	2200	1000	3250	1475	3800	1200	1900	5575	4200	18520
y	19929	5839	23696	9881	30011	7241	11634	45684	36476	12068

الف:

$x1 < -c(10.5, \dots, 17)$

$x2 < -c(2200, \dots, 18520)$

$y < -c(19929, \dots, 12068)$

$far < -data.frame(x1, x2, y)$

$ff < -lm(y \sim x1 + x2, far)$

$summary(ff)$

خروجیها به صورت زیر هستند.

Coefficients:

Value Std. Error t value Pr (>|t|)

(Intercept)	-1806.8237	1397.5264	-1.2929	0.2371
x1	1131.0285	615.7636	1.8368	0.1089
x2	4.0015	2.7077	1.4778	0.1830

Residual standard error: 1661 on 7 degrees of freedom

Multiple R-Squared: 0.9883

F-statistic: 294.4 on 2 and 7 degrees of freedom, the p-value is 1.758e-007

با توجه به مقدار کم pvalue و مقدار آماره‌ی F فرض صفر رد می‌شود. مدل برازش داده شده‌ی ما به شکل زیر است:

$$y_i = -1806.8237 + \ 1131.0285\ x_{1i} + \ 4.0015\ x_{2i} \qquad i = 1,2,3,$$

با توجه به خروجیهای بالا مقدار $\text{SSE} = (1661^2 \times 7) = 19312447$ و ضریب تعیین چندگانه $R^2 = 0.9883$

ب:

برای آزمون این فرض ابتدا مدل کاهش یافته را برازش می‌دهیم و با در نظرگرفتن مدل کامل و مدل کاهش یافته از آماره‌ی (مدل کامل)MSE/(تفاوت دو مدل)MSE= F^* استفاده می‌کنیم.

$x1 < -c(10.5, \dots, 17)$
$y < -c(19929, \dots, 12068)$
$far2 < -data.frame(x1, y)$
$ff2 < -lm(y \sim x1, far)$
$summary(ff2)$

خروجیها به شکل زیر هستند.

Coefficients:

	Value	Std. Error	t value	Pr (>|t |)
(Intercept)	-363.9804	1071.3768	-0.3397	0.7428
x1	2032.5326	89.9149	22.6051	0.0000

Residual standard error: 1779 on 8 degrees of freedom
Multiple R-Squared: 0.9846
F-statistic: 511 on 1 and 8 degrees of freedom, the p-value is 1.553e-008

خروجیهای مدل کاهش یافته نشـــان می‌دهند که $(1779^2 \times 8) = 25318728$ =مدل کاش یافتهSSE.
حالا آماره‌ی F^* را محاسبه می‌کنیم.

$$F^* = \left(\frac{(25318728 - 19312447)/1}{19312447/7}\right) = 2.17704$$

اگر ناحیه‌ی بحرانی را $F^* > F_{0.05}(1,7) = 5.59$ در نظر بگیریم دلیلی برای رد فرض صـــفر نداریم.

فصل دهم

طرح آزمایشات

1) طرح یک عاملی تصادفی ساده

اگر بخواهیم آزمون برابری میانگینها را برای چند ($k \geq 2$ و k) جامعه انجام دهیم از توابعی که مربوط به طرح آزمایشات است استفاده می‌کنیم. نظیر این را برای دو جامعه با تابع *t.test* انجام می‌دادیم. در اینجا ما قصد داریم فرض زیر، که فرض برابری میانگین‌های سطوح تیمار ما می‌باشد را آزمون کنیم.

$$\begin{cases} H_0: \mu_1 = \cdots = \mu_k \\ H_1: H_0 \text{ خلاف فرض} \end{cases}$$

مدل ما در اینجا به صورت زیر است.

$$y_{ij} = \mu + \tau_i + \varepsilon_{ij} \qquad \begin{cases} i = 1,2, \dots, k \\ j = 1,2, \dots, n_i \end{cases}$$

قدم اول شناساندن داده‌ها برای نرم افزار است. فرض کنیم که داده‌ها در جدولی به شکل زیر دسته بندی شده باشند.

سطح اول	سطح دوم	...	سطح k ام
y_{11}	y_{21}	...	y_{k1}

y_{12}	y_{22}	…	y_{k2}
⋮	⋮	⋮	⋮
y_{1n_1}	y_{2n_2}	…	y_{kn_k}

ابتدا داده‌ها را به ترتیب ستونی در متغیری قرار می‌دهیم. سپس متغیری را که ارزش عددی ندارد و فقط به عنوان فاکتور برای معرفی کردن اندیس داده‌ها به همان ترتیبی که آنها را وارد کرده‌ایم (ترتیب ستونی) به کار می‌رود، معرفی می‌کنیم. و در آخر این دو متغیر را با دستور $data.frame( )$ در کنار هم قرار می‌دهیم. به این ترتیب داده‌ها برای نرم افزار شناخته می‌شوند. این مراحل را در مورد جدول بالا به صورت زیر انجام می‌دهیم.

$y<-scan( )$ ↲

$1: y_{11}$ ↲

$2: y_{12}$ ↲

⋮

$\sum_{i=1}^{k} n_i : y_{kn_k}$ ↲↲

$a<-factor(rep(1:k, c(n_1, n_2, \dots, n_k)))$ ↲

$far<-data.frame(a, y)$ ↲

و اگر طرح متعادل باشد متغیر a را می‌توان به صورت زیر تعریف کرد .

$a<-factor(rep(1:k, rep(n, k)))$

متغیر a را با دستور $factor$ معرفی کردیم تا ارزش عددی نداشته باشد و فقط به عنوان معرف فاکتورها در نظر گرفته شود.

نکته 1 :

اگر طرح متعادل و تعداد تکرارها در سطوح متفاوت مساوی باشد می‌توانیم داده‌ها را به ترتیب سطری وارد کنیم. در این صورت باید متغیر a را به صورت زیر تعریف کنیم. (n: تعداد تکرارها)

$a<-factor(rep(1:k, n))$

قبل از هر کاری برای بررسی نرمال بودن و متقارن بودن داده‌ها می‌توان از دو دستوری که هر کدام برای رسم یک نوع نمودار که ما را در مورد مناسب بودن داده‌ها برای مدل بندی راهنمایی می‌کنند، استفاده کنیم. این دو دستور را در زیر آورده‌ایم.

$plot.factor(far)$

$plot.design(far)$

حال برای به دست آوردن جدول آنالیز واریانس از دستور $aov(\ )$ به شکل زیر استفاده می‌کنیم.

$de<-aov(y\sim a,far)$ ↵

de ↵

این دستور خروجی مختصری را به ما می‌دهد. اگر خروجی مفصل‌تری را بخواهیم باید از دستور $summary$ استفاده کنیم.

$summary\ (de)$ ↵

در آخر برای به دست آوردن باقیمانده‌ها و $\hat{y}_{ij}$ ها به ترتیب از دستورات زیر استفاده می‌کنیم.

$de\$resid$

$de\$fitted$

نکته 2 :

اگر در مدل مد نظر ما میانگین اولیه μ وجود نداشته باشد. یعنی مدل $y_{ij}=\tau_i+\varepsilon_{ij}$ $\begin{cases}i=1,2,\dots,k\\ j=1,2,\dots,n_i\end{cases}$ مد نظر باشد باید دستور $aov(\ )$ را به صورت زیر به کار ببریم.

$aov(y\sim -1+a,far)$

در ادامه مثالی را برای کاربرد دستورات بالا ارائه می‌دهیم.

مثال 1 :

با توجه به مشاهدات مربوط به مثال 1.3 کتاب طرح آزمایشهای مونتگمری که مربوط به مقاومت کششی الیاف مصنوعی جدید با درصدهای متفاوتی از پنبه است، مدل $y_{ij}=\mu+\tau_i+\varepsilon_{ij}$ $\begin{cases}i=1,2,\dots,5\\ j=1,2,\dots,5\end{cases}$ را در نظر گرفته و به تحلیل و تجزیه‌ی آن می پردازیم. داده‌های مشاهده شده‌ی مربوط به این مثال را برای راحتی کار در اینجا آورده‌ایم.

درصد پنبه	مشاهدات				
15	7	7	15	11	9
20	12	17	12	18	18
25	14	18	18	19	19
30	19	25	22	19	23
35	7	10	11	15	11

$y<-scan(\ )$ ↵

$1:7$ ↵

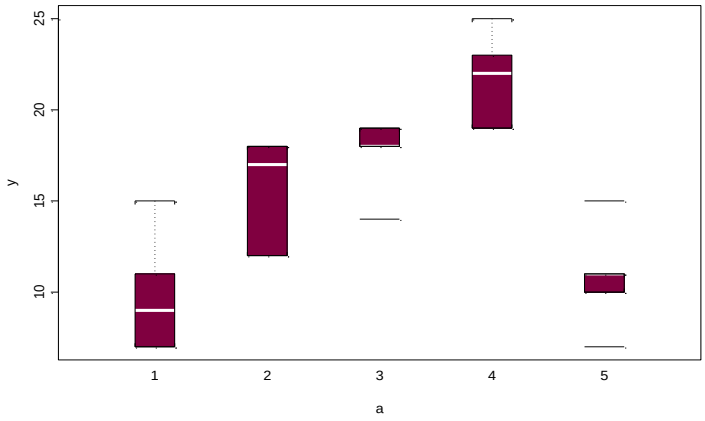
$2:7$ ↵

⋮

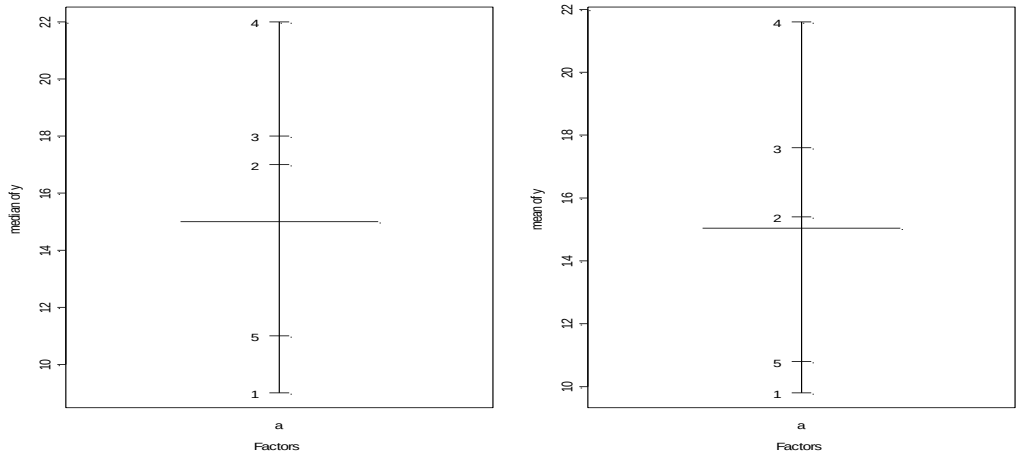
$25:11$ ↵↵

$a<-factor(rep(1:5,rep(5,5)))$ ↵

$far <- data.frame(a, y)$ ↲
$plot.factor(far)$ ↲



$par(mfrow = c(1,2))$ ↲
$plot.design(far)$ ↲
$plot.design(far, fun = \text{"median"})$ ↲



نمودار اول، نمودار جعبه‌ای را برای پنج سطح متفاوت پنبه رسم کرده است، که نشان می‌دهد داده‌ها تقریباً در این پنج سطح متقارن هستند. نمودار دوم که یکی بر اساس میانگین و دیگری بر اساس میانه رسم شده است به ترتیب میانگین و میانه‌ی داده‌های پنج سطح متفاوت پنبه را نشان می‌دهند. چون در این دو نمودار مشاهده می‌کنیم که تقریباً میانگین‌ها و میانه‌ها با هم برابر هستند بنابراین در اینجا هم نتیجه می‌گیریم که داده‌ها در سطوح متفاوت پنبه متقارن هستند. پس داده‌ها به داده‌های نرمال نزدیک هستند. بنابراین به ادامه‌ی محاسبات می‌پردازیم.

$des <- aov(y \sim a, far)$ ↲
$summary(des)$ ↲

خروجی این دستورات جدول آنالیز واریانسی است که در زیر آمده است.

	Df	Sum of Sq	Mean Sq	F Value	Pr(F)
a	4	475.76	118.94	14.75682	9.127937e-006
Residuals	20	161.20	8.06		

با توجه به جدول بالا در سطح معنی داری یک در صد ($\alpha = 0.01$) با توجه به مقدار کم p_value و مقدار F جدول فرض H_0 را رد می‌کنیم و نتیجه می‌گیریم که میانگین تیمارها متفاوت است. باقیمانده‌ها و $\hat{y}_i$ ها نیز به صورت زیر هستند.

$g < -des\$resid$ ↲

g ↲

```
      1      2      3      4      5      6      7      8      9     10     11     12     13     14     15     16     17     18
     19
-2.8   -2.8   5.2    1.2   -0.8   -3.4   1.6    -3.4   2.6    2.6   -3.6   0.4    0.4    1.4    1.4   -2.6   3.4   0.4   -2.6
     20     21     22     23     24     25
1.4    -3.8   -0.8   0.2    4.2    0.2
```

$v < -des\$fitted$ ↲

v ↲

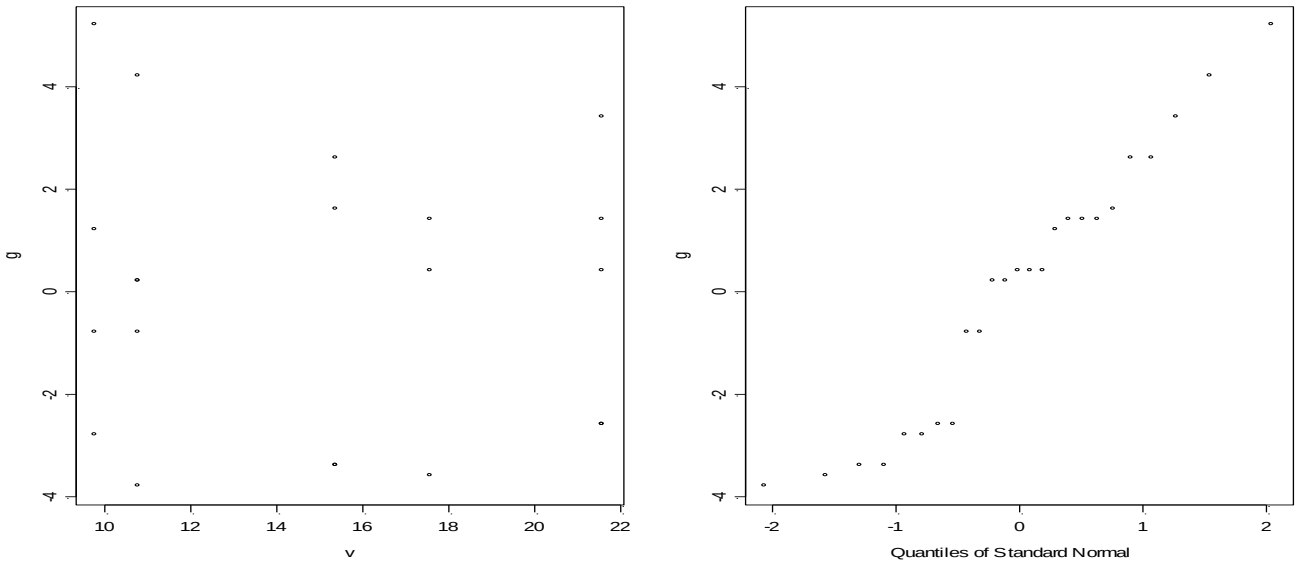
```
 1    2    3    4    5     6     7     8     9     10    11    12    13    14    15    16    17
 9.8  9.8  9.8  9.8  9.8  15.4  15.4  15.4  15.4  15.4  17.6  17.6  17.6  17.6  17.6  21.6  21.6
         18    19     20    21     22    23     24    25
 21.6  21.6  21.6  10.8  10.8  10.8  10.8  10.8
```

برای تشخیص مناسبت مدل از نمودارهای زیر استفاده می‌کنیم. که اولی نمودار مقادیر برازش شده در مقابل باقیماندهها و دیگری نمودار احتمال نرمال است، که هر دو دلالت بر مناسب بودن مدل می‌کنند.

$par(mfrow = c(1,2))$ ↲

$plot(v, g)$ ↲

$qqnorm(g)$ ↲



یک آزمون ناپارامتری برای آنالیز واریانس یک طرفه آزمون کروسکال والیس است. که توسط تابع kruskal . test ( ) انجام می‌شود. در مورد مثال بالا به صورت زیر است.

$kruskal.test(y, a)$ ↵

خروجی این دستور به صورت زیر است. که مقدار آماره و $p - value$ را در آن مشاهده می‌کنیم. بنابراین در سطح معنی داری 5 درصد فرض صفر را رد می‌کنیم.

$Kruskal - Wallis\ rank\ sum\ test$

$data:\ y\ and\ a$

$Kruskal - Wallis\ chi - square\ =\ 19.0637, df\ =\ 4, p - value$
$=\ 0.0008$

$alternative\ hypothesis: two.sided$

2) طرح دو عاملی تصادفی ساده‌ی متعادل

فرض کنیم عامل اول دارای a سطح و عامل دوم دارای b سطح باشد. و تعداد تکراها n باشد. در این صورت می‌خواهیم مدل زیر را برازش دهیم.

$$y_{ijk} = \tau_i + \beta_j + \tau\beta_{ij} + \varepsilon_{ijk} \qquad \begin{cases} i = 1,2,\dots,a \\ j = 1,2,\dots,b \\ k = 1,2,\dots,n \end{cases}$$

فرض کنیم که داده‌ها در جدولی مانند جدول بعد جمع آوری شده باشند.

عامل اول / عامل دوم	1	2	...	a
1	y_{111} ⋮	y_{211} ⋮	...	y_{a11} ⋮

	y_{11n}	y_{21n}		y_{a1n}
2	y_{121} ⋮ y_{12n}	y_{221} ⋮ y_{22n}	....	y_{a21} ⋮ y_{a2n}
⋮	⋮	⋮	....	⋮
b	y_{1b1} ⋮ y_{1bn}	y_{2b1} ⋮ y_{2bn}	...	y_{ab1} ⋮ y_{abn}

مانند طرح یک عاملی ابتدا داده‌ها را با دستو $scan(\,)$ وارد می‌کنیم. سپس متغیرهایی را که فاکتورها را با دستور $factor(\,)$ در آنها قرار می‌دهیم معرفی می‌کنیم. چون در این حالت با دو فاکتور روبه‌رو هستیم پس باید دو متغیر فاکتور را معرفی کنیم، که با توجه به ترتیب وارد کردن داده‌ها، یکی اندیسهای عامل اول و دیگری اندیس‌های عامل دوم را مشخص کند. برای داده‌های داخل جدول اگر آنهارا به ترتیب سطری وارد کنیم، باید دستورات زیر را به کار ببریم.

$y < -scan(\,)$ ↲

$1: y_{111}$ ↲

$2: y_{112}$ ↲

⋮

$n: y_{11n}$ ↲

$n+1: y_{211}$ ↲

⋮

$a < -factor\left(rep\left(rep(1:a, rep(n,a)), b\right)\right)$ ↲

$b < -factor\left(rep\left(1:b, rep(a*n, b)\right)\right)$ ↲

$far < -data.frame(y, a, b)$ ↲

نکته 3 :

به نحوه‌ی معرفی دو متغیر a وb دقت کنید. طوری آنها را معرفی کرده‌ایم که a معرف اندیسهای مربوط به عامل اول و b معرف اندیسهای مربوط به عامل دوم باشد، وقتی که داده‌ها را به ترتیب سطری وارد می‌کنیم. اگر داده‌ها را به ترتیب ستونی وارد کنیم این دو متغیر را به شکل زیر تعریف می‌کنیم.

$a < -factor\left(rep\left(1:a, rep(n*b, a)\right)\right)$ ↲

$b < -factor(rep\left(rep\left(1:b, rep(n,b)\right), a\right))$ ↲

در هر صورت این دو متغیر باید به گونه‌ای تعریف شوند که اندیس داده‌ها را به ترتیبی که آنها را وارد کرده‌ایم برای ما مشخص کنند.

برای به دست آوردن جدول آنالیز واریانس از همان دستور $aov(\ )$ به صورت زیر استفاده می‌کنیم.

$desi < -aov(y \sim a + b + a{:}\, b, far)$ ↵

$summary(desi)$ ↵

نکته 4 :

در دستور $aov(\ )$ ، a یعنی اثر عامل اول (τ_i)، b یعنی اثر عامل دوم (β_j)، a:b یعنی اثر متقابل این دو عامل ($\tau\beta_{ij}$) و علامت "+" قبل از هر اثر به این معنی است که ما می‌خواهیم این اثر در مدل برازش شده باشد. اگر بخواهیم مدلی را برازش دهیم که فاقد اثر خاصی باشد، در این دستور باید قبل از آن اثر به جای "+" از علامت "-" استفاده کنیم. در حالتی که مدلی شامل تمامی اثرهای اصلی و تمامی اثرهای متقابل رابخواهیم می‌توانیم این دستوررا به شکل زیر به کار ببریم.

$aov(y \sim a * b, far)$

همچنین اگر مدلی کامل، بدون میانگین اولیه را بخواهیم برازش دهیم مانند طرح یک عاملی از این دستور به شکل زیر استفاده می‌کنیم.

$aov(y \sim -1 + a * b, far)$

بنابراین برای در نظر گرفتن اثرهای متقابل بین فاکتورها از علامت " **:** " و از علامت " * " بین فاکتورها برای در نظر گرفتن تمامی اثرها در مدل استفاده می‌کنیم.

برای رسم نمودارها و همچنین به دست آوردن باقیمانده‌ها و $\hat{y}_i$ ها از همان دستوراتی که در طرح یک عاملی استفاده می‌کردیم، در اینجا نیز استفاده می‌کنیم.

قبل از اجرای طرح برای آگاهی از وجود اثر متقابل در مدل می‌توانیم از نموداری که توسط دستور زیر رسم می‌شود استفاده کنیم.

$interaction.plot(a, b, y)$ ↵

مثال 2 :

برای داده‌های مثال 1.7 کتاب مونتگمری که مربوط به طول عمر مؤثر باطری‌ها تحت دو عامل دما و نوع مواد است و در جدول زیر جمع‌آوری شده‌اند جدول آنالیز واریانس مربوط به مدل زیر را به دست آورید.

$$y_{ijk} = \tau_i + \beta_j + \tau\beta_{ij} + \varepsilon_{ijk} \quad \begin{cases} i = 1,2,3 \\ j = 1,2,3 \\ k = 1,2,3,4 \end{cases}$$

نوع مواد \ دما	15	70	125
1	130 155 74 180	34 40 80 75	20 70 82 58
2	150 188	136 122	25 70

	159 126	106 115	58 45
3	138 110 168 160	174 120 150 139	96 104 82 160

$y < -scan( )$ ↲

$1: 130$ ↲

$2: 155$ ↲

⋮

$4: 180$ ↲

$5: 34$ ↲

⋮

$48: 160$ ↲↲

$a < -factor\left(rep\left(rep(1: 3, rep(4,3)), 3\right)\right)$ ↲

$b < -factor(rep(1: 3, rep(12,3)))$ ↲

$far < -data.frame(y, a, b)$ ↲

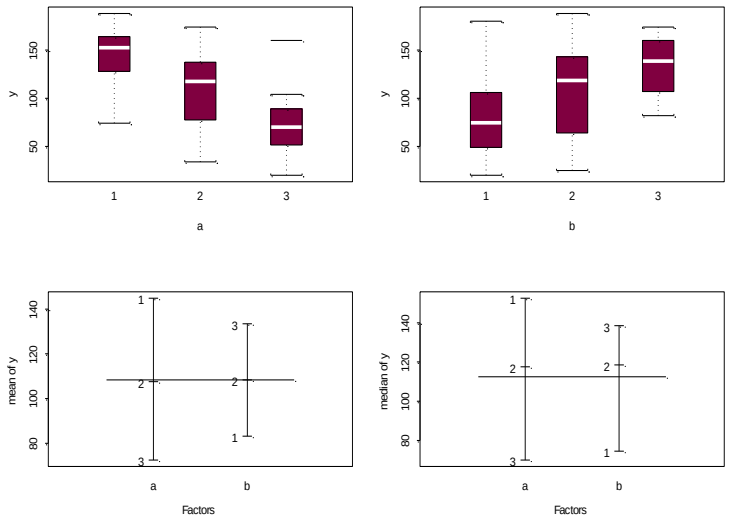
حال به رســـم نمودارهای مهم می‌پردازیم. توجه کنیم که متغیر far بانک اطلاعاتی ما محســـوب می‌شود.

$par(mfrow = c(2,2))$ ↲

$plot.factor(far)$ ↲

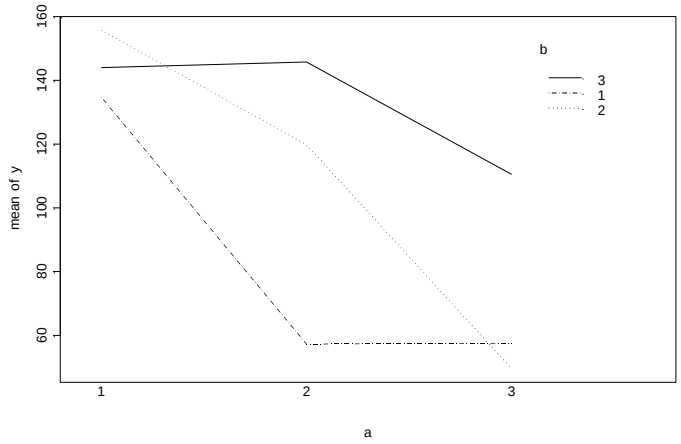
$plot.design(far)$ ↲

$plot.design(far, fun = \text{median})$ ↲



با توجه به نمودارهای بالا می‌توان نتیجه گرفت، که داده‌ها تقریباً نرمال هستند و می‌توان مدل طرح مورد نظر را به آنها برازش داد. با توجه به نمودار زیر وجود اثر متقابل در مدل معنی‌دار است.

$interaction.plot(a, b, y)$ ↲



به سراغ جدول آنالیز واریانس می‌رویم.

$desi < -aov(y \sim a + b + a{:}\, b, far)$ ↲

$summary(desi)$ ↲

	D f	Sum of Sq	Mean Sq	F Value	Pr (F)
b	2	15150.39	7575.19	9.91385	0.00059105
a	2	31402.06	15701.03	20.54834	0.00000377
b: a	4	11080.44	2770.11	3.62532	0.01724889
Residuals	27	20630.75	764.10		

با توجه به مقادیر p_value در جدول بالا وجود تمامی اثرات در مدل معنی‌دار است. باقیمانده‌ها و $\hat{y}_i$ ها و نمودار آنها در مقابل هم را به صورت زیر به دست می‌آوریم.

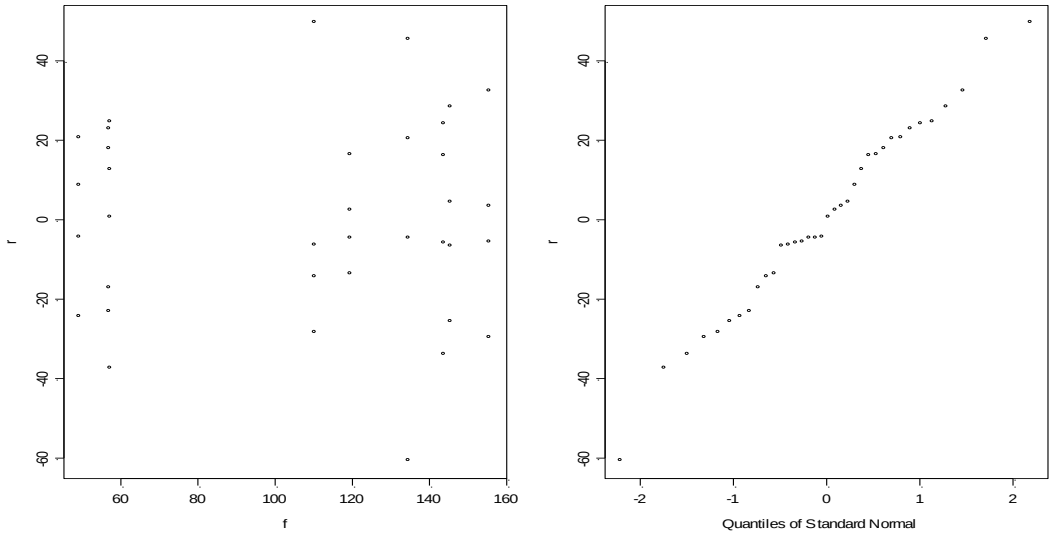
$r < -desi\$resid$ ↲

$f < -desi\$fitted$

$par(mfrow = c(1,2))$ ↲

$plot(f, r)$ ↲

$qqnorm(r)$ ↲



برای اجرای طرح سه عاملی و غیره باید ابتدا متغیرهای فاکتور را تعریف کنیم و سپس از دستور $aov()$ استفاده کنیم. مثلاً در مورد طرح سه عاملی اگر متغیرهای فاکتور را با حروف a , b و d نامگذاری کنیم، برای برازش مدل کامل که شامل همه‌ی اثرها است از این دستور می‌توان به شکل‌های زیر استفاده کرد.

1) $aov(y \sim a + b + a:b + a:d + b:d + a:b:d,$ بانک اطلاعاتی$)$

2) $aov(y \sim a * b * d,$ بانک اطلاعاتی$)$

نکته 5 :

اگر بخواهیم مدلی با اثرهای مربوط به طرح آشیانه‌ای $(nested.design)$ را برازش دهیم، باید در دستور $aov()$ به جای علامت " **:** " از علامت " / " برای اثرهای مربوط به این نوع مدل‌ها استفاده کنیم. مثلاً به کاربرد این دستور برای برازش مدلی که در ادامه معرفی کرده‌ایم و در آن عامل b در عامل a آشیانه شده است توجه کنید.

$$y_{ijk} = \tau_i + \beta_j(\tau_i) + \varepsilon_{ijk} \quad \begin{cases} i = 1,2,\dots,a \\ j = 1,2,\dots,b \\ k = 1,2,\dots,n \end{cases}$$

$aov(y \sim a + a/b,$ بانک اطلاعاتی$)$

3) طرح بلوک کاملاً تصادفی متعادل

در اینجا می‌خواهیم مدل $y_{ijk} = \tau_i + \beta_j + \varepsilon_{ijk}$ $\begin{cases} i = 1,\dots,t \\ j = 1,\dots,b \\ k = 1,\dots,n \end{cases}$ که در آن τ_i اثر تیمار و β_j اثر بلوک می‌باشد را برازش دهیم. t تعداد تیمارها و b تعداد بلوکها است. نحوه‌ی وارد کردن داده‌ها و دستورات مربوط به این مدل مانند طرح دو عاملی است. فقط باید دقت کنیم که در این مدل اثر

متقابل نداریم. بنابراین در دستور $aov(\ )$ اثر متقابل را در نظر نمی‌گیریم. برای یادگیری بهتر به ذکر مثالی می‌پردازیم.

مثال 3 :

برای داده های جدول زیرمدل طرح بلوکی را برازش دهید.

	تیمار اول	تیمار دوم	تیمارسوم	تیمار چهارم
بلوک اول	89	88	97	94
بلوک دوم	84	77	92	79
بلوک سوم	81	87	87	85
بلوک چهارم	87	92	89	84
بلوک پنجم	79	81	80	88

ابتدا داده‌ها را به ترتیب سطری وارد می‌کنیم.

$y < -scan(\ )$ ↲

$1:89$ ↲

⋮

$20:88$ ↲↲

$treat < -factor\big(rep(LETTERS[1:4],5)\big)$ ↲

$block < -factor\left(rep\big(paste(\mathsf{b},1:5),rep(4,5)\big)\right)$ ↲

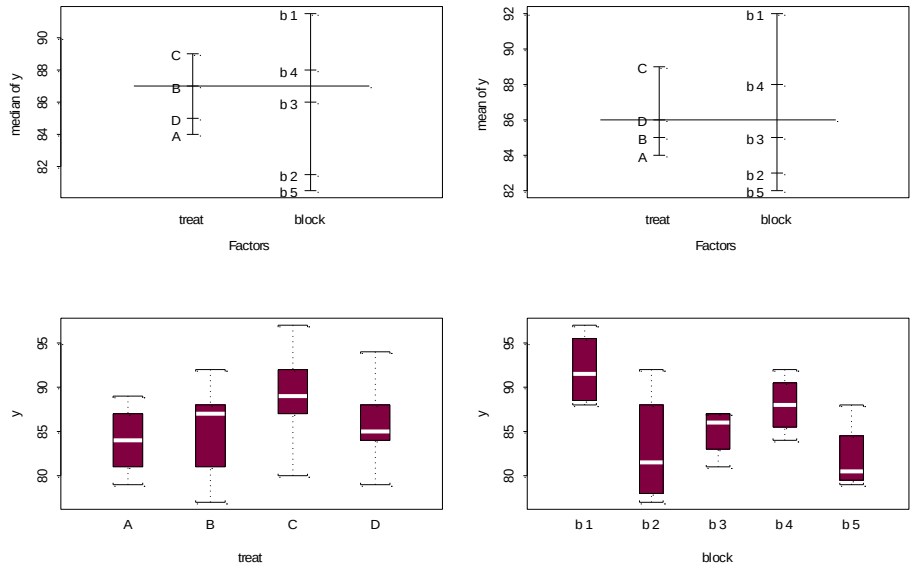
$far < -data.frame(block, treat, y)$ ↲

$par\big(mfrow = c(2,2)\big)$ ↲

$plot.design(far)$ ↲

$plot.design(far, fun = 'median')$ ↲

$plot.factor(far)$ ↲



با توجه به نمودارهای بالا داده‌ها متقارن و نرمال هستند. پس به برازش مدل می‌پردازیم.

$aa < -aov(y \sim treat + block, far)$ ↵

$summary(aa)$ ↵

	Df	Sum of Sq	Mean Sq	F Value	Pr(F)
treat	3	70	23.33333	1.238938	0.3386581
block	4	264	66.00000	3.504425	0.0407462
residuals	12	226	18.83333		

در سطح معنی‌داری $\alpha = 0.05$ ، با توجه به مقادیر p_value اثر تیمار در مدل معنی‌دار نیست ولی اثر بلوک در مدل معنی‌دار است.

برای به دست آوردن باقیمانده‌ها و $\hat{y}_{ijk}$ ها نیز از همان دستورات قبلی استفاده می‌کنیم.

نکته 6 :

یک آزمون ناپارامتری برای طرح بلوکی توسط تابع friedman.test ( ) انجام می‌گیرد. در مورد مثال قبل داریم :

$friedman.test(y, treat, block)$ ↵

در خروجیهای این دستور مقدار آماره به همراه مقدار $p - value$ را مشاهده می‌کنیم.

$Friedman\ rank\ sum\ test$

$data:\ y\ and\ treat\ and\ block$

$Friedman\ chi - square\ =\ 3.4898, df\ =\ 3, p - value\ =\ 0.3221$

$alternative\ hypothesis: two.sided$

4) طرح فاکتوریل 2^k

برای اجرای این نوع طرح با توجه به این که هر کدام از عامل ها از دو سطح تشکیل شده‌اند ابتدا ترتیب تصادفی اجرای واحدهای آزمایش را تعیین می‌کنیم. برای این کار ابتدا لیستی را به صورت زیر تعریف می‌کنیم و آن را به یک متغیر دلخواه نسبت می‌دهیم.

$f < -list(f1 = c(\text{"1L","1H"}), \text{f2=c("2}L\text{" ,"2}H\text{"}), \ldots, fk = c(\text{"}kL\text{" ,"}kH\text{"}))$

حرف L را برای سطح پایین هر عامل و حرف H را برای سطح بالای هر عامل در نظر گرفته‌ایم. حال توسط تابع $fac.design( )$ متغیر f را در کنار برداری شامل k تا عدد 2 قرار می‌دهیم و ترتیب تصادفی اجرای واحدها را به دست می‌آوریم. سپس مشاهدات را به ترتیبی که داریم توسط دستور scan( ) وارد می‌کنیم. به چگونگی کاربرد تابع $fac.design( )$ در زیر دقت کنید.

$d < -fac.design(c(2,2, \ldots,2), f)$

d ↵

می‌توانیم به جای دستور c( ) از دستور rep(2,k) استفاده کنیم. توجه کنید که متغیر d دلخواه است. بعد از اینکه داده‌ها را به ترتیبی که متغیر d تعیین می‌کند وارد کردیم، آنها را توسط

دســـتور( )data.frame در کنار هم قرار می‌دهیم و برای به دســـت آوردن جدول آنالیز واریانس مانند قبل از دستور $aov(\ )$ استفاده می‌کنیم.

مثال 4 :

برای داده‌های مثال 2.9 کتاب مونتگمری جدول آنالیز واریانس را برای مدل طرح 2^k که شـــامل تمامی اثرات است به دست آورید.

شماره اجرا	عاملA	عاملB	عامل C	عامل D	مشاهدات
1	-	-	-	-	45
2	+	-	-	-	71
3	-	+	-	-	48
4	+	+	-	-	65
5	-	-	+	-	68
6	+	-	+	-	60
7	-	+	+	-	80
8	+	+	+	-	65
9	-	-	-	+	43
10	+	-	-	+	100
11	-	+	-	+	45
12	+	+	-	+	104
13	-	-	+	+	75
14	+	-	+	+	86
15	-	+	+	+	70
16	+	+	+	+	96

$f <-list(f1 = c(\text{ "1L","1H"}),\text{f2=c("2}L\text{" ,"2H"}), f3 = c(\text{ "3L",3H"}), f4 = c(\text{ "4L","4H"}))$

$d <- fac.design(rep(2,4), f)$

d ↵

خروجی این دستورات ترتیب اجرای واحدها را به ما نشان می‌دهد. که دقیقاً مانند جدول بالا است.

```
   f1 f2 f3 f4                x f1 f2 f3 f4
1  1L 2L 3L 4L             1  45 1L 2L 3L 4L
2  1H 2L 3L 4L             2  71 1H 2L 3L 4L
3  1L 2H 3L 4L             3  48 1L 2H 3L 4L
4  1H 2H 3L 4L             4  65 1H 2H 3L 4L
5  1L 2L 3H 4L             5  68 1L 2L 3H 4L
6  1H 2L 3H 4L             6  60 1H 2L 3H 4L
7  1L 2H 3H 4L             7  80 1L 2H 3H 4L
8  1H 2H 3H 4L             8  65 1H 2H 3H 4L
9  1L 2L 3L 4H             9  43 1L 2L 3L 4H
10 1H 2L 3L 4H            10 100 1H 2L 3L 4H
11 1L 2H 3L 4H            11  45 1L 2H 3L 4H
12 1H 2H 3L 4H            12 104 1H 2H 3L 4H
```

13 1L 2L 3H 4H 13 75 1L 2L 3H 4H
14 1H 2L 3H 4H 14 86 1H 2L 3H 4H
15 1L 2H 3H 4H 15 70 1L 2H 3H 4H
16 1H 2H 3H 4H 16 96 1H 2H 3H 4H

حال داده‌ها را به ترتیب وارد می‌کنیم.

$x < -scan(\,)$ ↲
$1:45$ ↲
⋮
$16:96$ ↲
$far < -data.frame(d, x)$
$a < -aov(x \sim f1 * f2 * f3 * f4, far)$

$summary(a)$ ↲

	Df	$Sum\ of\ Sq$	$Mean\ Sq$
$f1$	1	1870.562	1870.562
$f2$	1	39.063	39.063
$f3$	1	390.063	390.063
$f4$	1	855.562	855.562
$f1:f2$	1	0.062	0.062
$f1:f3$	1	1314.062	1314.062
$f2:f3$	1	22.562	22.562
$f1:f4$	1	1105.562	1105.562
$f2:f4$	1	0.563	0.563
$f3:f4$	1	5.063	5.063
$f1:f2:f3$	1	14.063	14.063
$f1:f2:f4$	1	68.063	68.063
$f1:f3:f4$	1	10.562	10.562
$f2:f3:f4$	1	27.562	27.562
$f1:f2:f3:f4$	1	7.562	7.562

متأسفانه در جدول آنالیز واریانس مقدار آماره‌های F داده نشده است. بنابراین برای بررسی معنی‌داری هرکدام از اثرها باید خودمان این آماره‌ها را محاسبه کنیم.

تمرینهای فصل دهم

تمرین 1 :

برای داده‌های زیر که مربوط به مثال 4.7 کتاب مونتگمری اســـت، جدول آنالیز واریانس را برای مدل زیر به دست آورید.

$$Y_{ijlk} = \tau_i + \beta_j + \gamma_l + \tau\beta_{ij} + \gamma\beta_{jl} + \alpha\beta\gamma_{ijl} + \varepsilon_{ijlk} \qquad \begin{cases} i = 1,2,3 \\ j = 1,2 \\ l = 1,2 \\ k = 1,2,3,4 \end{cases}$$

درصد گاز کربنیک (A)	فشار دستگاه (B)			
	25 پوند بر اینچ		30 پوند بر اینچ	
	سرعت نقاله (C)		سرعت نقاله (C)	
	200	250	200	250
10	-3	-1	-1	1
	-1	0	0	1
12	0	2	2	6
	1	1	3	5
14	5	7	7	10
	4	6	9	11

```
y < −c(−3, −1, −1,1, … ,4,6,9,11)
a < −factor(rep(1: 3, rep(8,3)))
b < −factor(rep(rep(1: 2, rep(2,2)), 6)
C < −factor(rep(1: 2,12))
far < −data. frame(y, a, b, C)
a1 < −aov(y~a ∗ b ∗ C − a: C, far)
summary(a1)
```

جدول آنالیز واریانس به صورت زیر است.

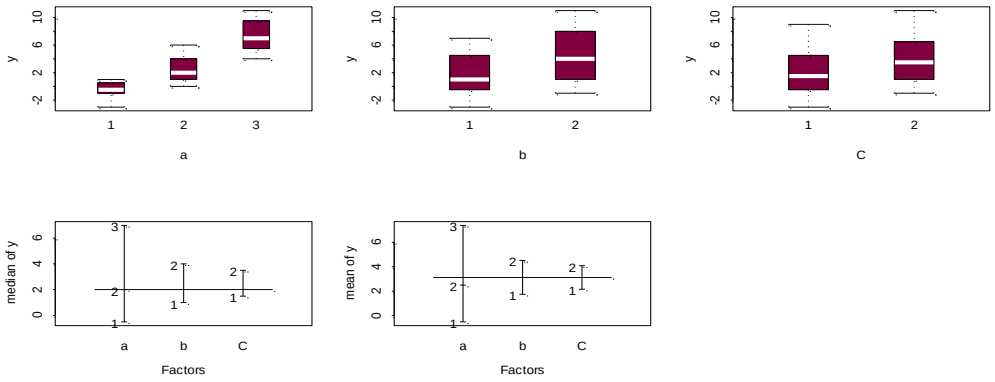
		Df	Sum of Sq	Mean Sq	F Value	Pr(F)
a	2	252.7500	126.3750	194.7798	0.0000000	
b	1	45.3750	45.3750	69.9358	0.0000008	
C	1	22.0417	22.0417	33.9725	0.0000438	
a: b	2	5.2500	2.6250	4.0459	0.0410475	
b: C	1	1.0417	1.0417	1.6055	0.2257978	
a: b: C	2	1.0833	0.5417	0.8349	0.4544297	
Residuals	14	9.0833	0.6488			

می‌توانیم دراینجا هم نمودارهای مهم را رسم کنیم.

```
plot. factor(far)
plot. design(far)
```

$plot.design(far, fun =' median')$

نمودارها به شکل زیر هستند.



تمرین 2 :

برای داده‌های زیر که مربوط به مثال 1.13 کتاب مونتگمری است مدل زیر را برازش دهید.

$$Y_{ijk} = \tau_i + \beta_j(\tau_i) + \varepsilon_{ijk} \qquad \begin{cases} i = 1,2,3 \\ j = 1,2,3,4 \\ k = 1,2,3 \end{cases}$$

	تهیه کننده‌ی 1				تهیه کننده‌ی 2				تهیه کننده‌ی 3			
دسته‌ها	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
	1	-2	-2	1	1	0	-1	0	2	-2	1	3
	-1	-3	0	4	-2	4	0	3	4	0	-1	2
	0	-4	1	0	-3	2	-2	2	0	2	2	1

$y < -c(1, -2, -2,1,1, \dots ,2,2,1)$

$a < -factor\left(rep\left(rep\left(1{:}\,3, rep(4,3)\right), 3\right)\right)$

$b < -factor\left(rep(1{:}\,4,9)\right)$

$far < -data.frame(y, a, b)$

$a < -aov(y \sim a + a/b, far)$

$summary(a)$ ↲

جدول آنالیز واریانس که خروجی این دستورات است به شکل زیر است.

	Df	Sum of Sq	Mean Sq	F Value	Pr(F)
a	2	15.05556	7.527778	2.852632	0.07736313
b %in% a	9	69.91667	7.768519	2.943860	0.01667416
Residuals	24	63.33333	2.638889		