

[連載]フリーソフトによるデータ解析・マイニング 第5回

Rでのデータの視覚化(2)

6 散布図

散布図は変数間の関係を考察するために、個体を2~3次元空間に配置したグラフである。

6.1 2次元散布図

2次元散布図というのは、一つの変数を横軸、もう一つの変数を縦軸として2次元平面に個体のデータを配置したグラフを指す。Rで散布図の作成はplotという関数を用いる。例えば、irisデータの第1列を横軸、第3列を縦軸とし、関数plotを用いてコマンド

```
>plot(iris[,1],iris[,3])
```

を実行すると、図11のような散布図が作成される。

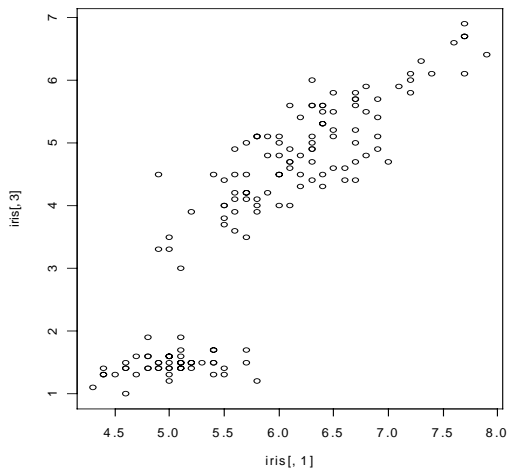


図11 ラベルなしの散布図

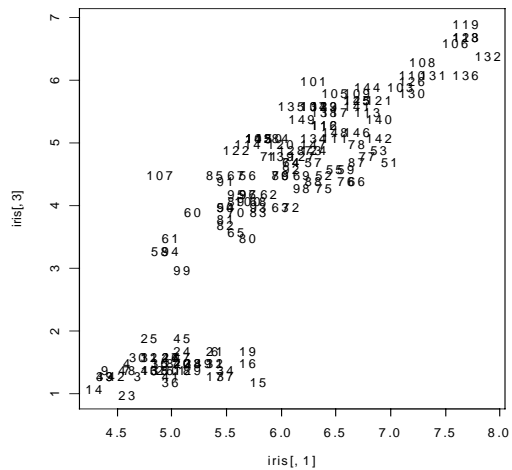


図12 個体番号が付いた散布図

このような散布図はラベルがないので、どの個体がどの位置に配置されているかに関する情報が読み取れない。次のように2行のコマンドを実行すると個体の番号が付けられた図12のような散布図が返される。

```
>plot(iris[,1],iris[,3],type="n")  
>text(iris[,1],iris[,3])
```

関数 `plot` に用いた `type` は散布図のマークの種類を指定する引数で、`type="n"` は散布図のマークを描かない。関数 `text` は散布図のラベルなどを加える関数で、`plot` の引数 `type="n"` と `text(x,y)` の組み合わせで、データの番号をラベルとして付ける。

データの番号ではなく、好きなラベルを各個体につけることも可能である。好きなラベルを付けるためには、ラベルを表す文字列ベクトルを事前に準備しておくことと便利である。

用いた `iris` データの第 5 列はラベルのデータであるが、その表記が長いので、それを用いると散布図が非常に見にくくなる。そこで新たにラベルを作成することにする。ラベルデータの作成は、一つ一つのラベルをキーボードで入力することが最も素朴な方法である。しかし、`iris` の個体数は 150 もあり、すべてをキーボードで入力することはやや面倒であるので、`rep` という関数を用いて作成する。`rep` を用いた次のコマンドを実行すると、はじめの 50 のラベルは S、51 から 100 までのラベルは C、101 から 150 までのラベルは V であるラベルベクトルが `iris.label` という名前で作成される。

```
>iris.label<-rep(c("S","C","V"),rep(50,3))
```

次の 2 行のコマンドにより作成された散布図を図 13 に示す。

```
>plot(iris[,1],iris[,3],type="n")  
>text(iris[,1],iris[,3],iris.label)
```

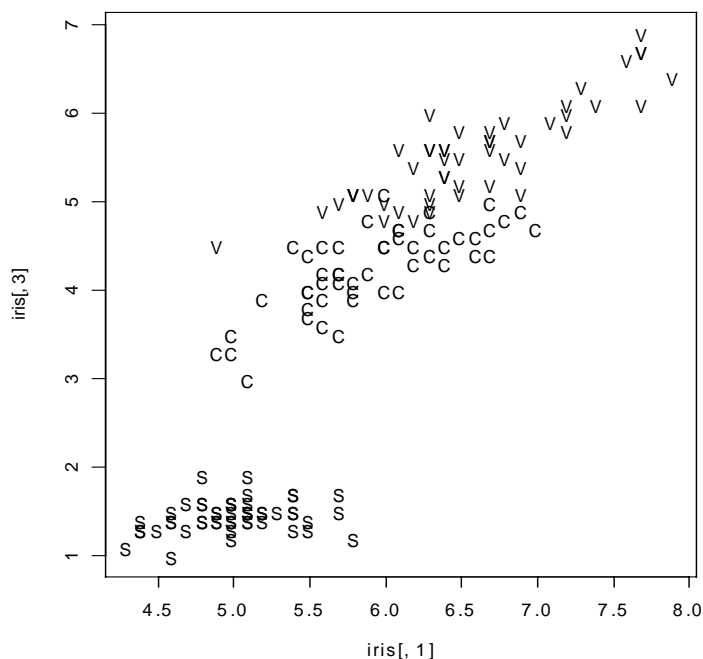


図 13 ラベルが付いた散布図

図 13 から三種類のアヤメにおける各種類の特徴が読み取られる。花卉の長さ(縦軸)が最も短いのは *setosa* で、その次が *versicolor*、*virginica* の順で、花卉の長さでその品種がある程度識別できる。例えば、花卉の長さが 2 を超えなければその品種は *setosa* であるなど。

関数 `plot` の書き式と主な引数及びその機能を表 3 に示す。

表 3 関数 `plot` の書き式と主な引数

<code>plot(x, y, xlim=range(x), ylim=range(y), type="p", main, xlab, ylab.....)</code>	
<code>x</code>	横軸のデータ
<code>y</code>	縦軸のデータ
<code>xlim</code>	横軸の範囲
<code>ylim</code>	縦軸の範囲
<code>type</code>	"p" は、点を描く "l" は、点を通する線のみを描く "b" は、点を描き、線で点を連結 "c" は、点を飛ばして線を描く "h" は、各点から横軸までの垂直線を描く "n" は、枠(軸)のみを描く
<code>main</code>	散布図の上部にタイトルを付ける
<code>xlab</code>	x 軸(横軸)の名前を付ける
<code>ylab</code>	y 軸(縦軸)の名前を付ける
<code>cex</code>	点のサイズ、指定しない場合は 1、

表 3 の引数を組み合わせた次のコマンドを実行すると図 14 のような図が作成される。

```
>plot(iris[,1],iris[,3],type="p",xlab="Length of Sepal", ylab="Length of Petal",cex=2, col="red")
```

図 4 はどの点がどの種類であることを識別することができない。次のように *iris* の種類別に色付けすることもできる。次のコマンドを実行すると、図 15 のような色が付いた散布図が作成される。

```
>plot(iris[,1],iris[,3],pch = 21, cex=2,bg = c(2, 3, 4)[unclass(iris$Species)])
```

ここの引数 `pch = 21` は、円形マークを色で塗りつぶすこと、`bg = c(2, 3, 4)` はマークを 3 種類(赤、緑、青)の色で塗りつぶすこと、`[unclass(iris$Species)]` は、塗りつぶす色はデータ *iris* の中 *Species* 列の品種の順であることの指定である。Iris データは 3 種類であるので、個体 1 ~ 50 の *setosa* 品種は色コード 2 の赤色(`red`)、51 ~ 100 の *versicolor* 品種は色コード 3 の緑色(`green`)、101 ~ 150 の *virginica* 品種は色コード 4 の青色(`blue`)で塗りつぶす。

軸の文字、ラベル、軸の名前の文字サイズを同時にコントロールするには、`plot` を実行する前に関数 `par` に引数 `cex` を次のように用いる。

```
>par(cex=0.8)
```

引数 `cex` のパラメータ `0.8` は標準サイズの `0.8` 倍の大きさを文字を表示することを意味する。このパラメータは自由に調整することが可能である。関数 `par` はグラフ画面の設定を行う関数で、数十個の引数がある。その主な引数を表 4 に示す。

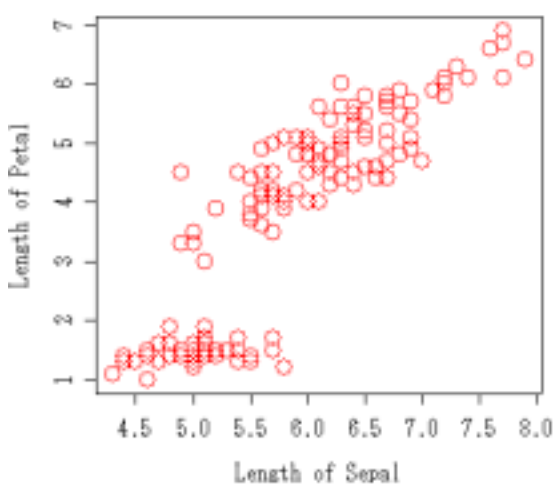


図 14 複数の引数を用いた散布図の例

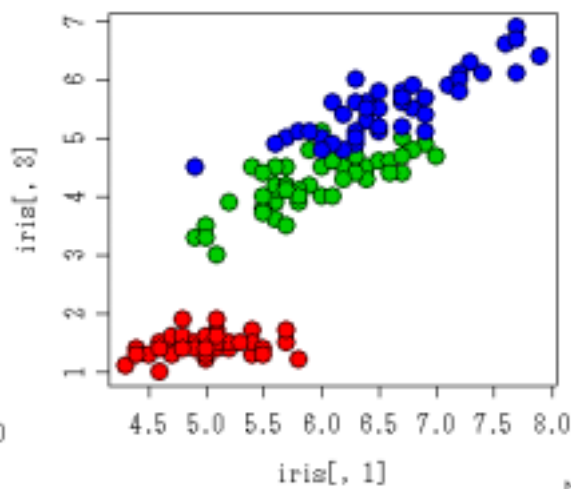


図 15 種類ごと着色した散布図

表 4 `par` の書き式と主な引数

	<code>par(cex=,pch=, mfrow=, col=,.....)</code>
<code>cex=</code>	文字、あるいはマークのサイズを指定する。初期値は 1 である。
<code>pch=n</code>	数値でマーク(絵記号)を指定。0 は <code>○</code> 、1 は <code>●</code> など図 16 を参照。
<code>pch = "文字"</code>	<code>"</code> で囲まれた文字を点の代わりにプロットする。
<code>mfrow=c(m,n)</code>	一つの画面に <code>m</code> 行 <code>n</code> 列の図を行順に描く、初期値は <code>c(1,1)</code> である。
<code>mfcol=c(m,n)</code>	一つの画面に <code>m</code> 行 <code>n</code> 列の図を列順に描く、初期値は <code>c(1,1)</code> である。
<code>bg=</code>	マークなどを塗りつぶすのに使用するの色を指定する。
<code>col=</code>	軸とマークの色を指定する。色の表記は表 1 を参照。文字列で色を示す場合は <code>" "</code> で囲むが、数値で示す場合は <code>" "</code> で囲まない。例えば、赤で散布図を作成する場合は、 <code>col="red"</code> 、あるいは <code>col=2</code> 。初期値は 1 (黒) である。
<code>lty=</code>	線のタイプを指定する。1 は実線、2 は点線など
<code>lwd=</code>	線の太さを指定する。初期値は 1。数値が大きくなると線が太くなる。

図 16 に引数 `pch` の数値とマークの対応関係を示す。例えば、`pch=1` の時は `●` となる。

引数pchの数値とマークの対応

○	△	+	×	◇
1	2	3	4	5
▽	⊠	✱	⊕	⊗
6	7	8	9	10
⊠	⊞	⊠	⊠	■
11	12	13	14	15
●	▲	◆	●	●
16	17	18	19	20
○	□	◇	△	▽
21	22	23	24	25

図 16 引数 pch の数値とマークの対応関係

6.2 対散布図(散布図行列)

用いる変数が 2 以上で、かつ変数がそれほど多くない量的データ場合は、本格的な解析を行う前に、すべての変数を組み合わせた散布図について考察を行うことで、データ間の関連性を視覚的に把握することができる。1 つの画面に複数の変数を組み合わせた散布図を対散布図、あるいは散布図行列という。対散布図の作成は、関数 `pairs` を用いる。`iris` のデータを用いて対散布図の作成について説明する。`iris` データの四つの変数を用いた最もシンプルな対散布図は次のコマンドで作成できる。

```
>pairs(iris[1:4])
```

このコマンドで作成された散布図では、品種を区別することができない。コマンド

```
>pairs(iris[1:4], pch = 21, bg = c("red", "green3", "blue")[unclass(iris$Species)])
```

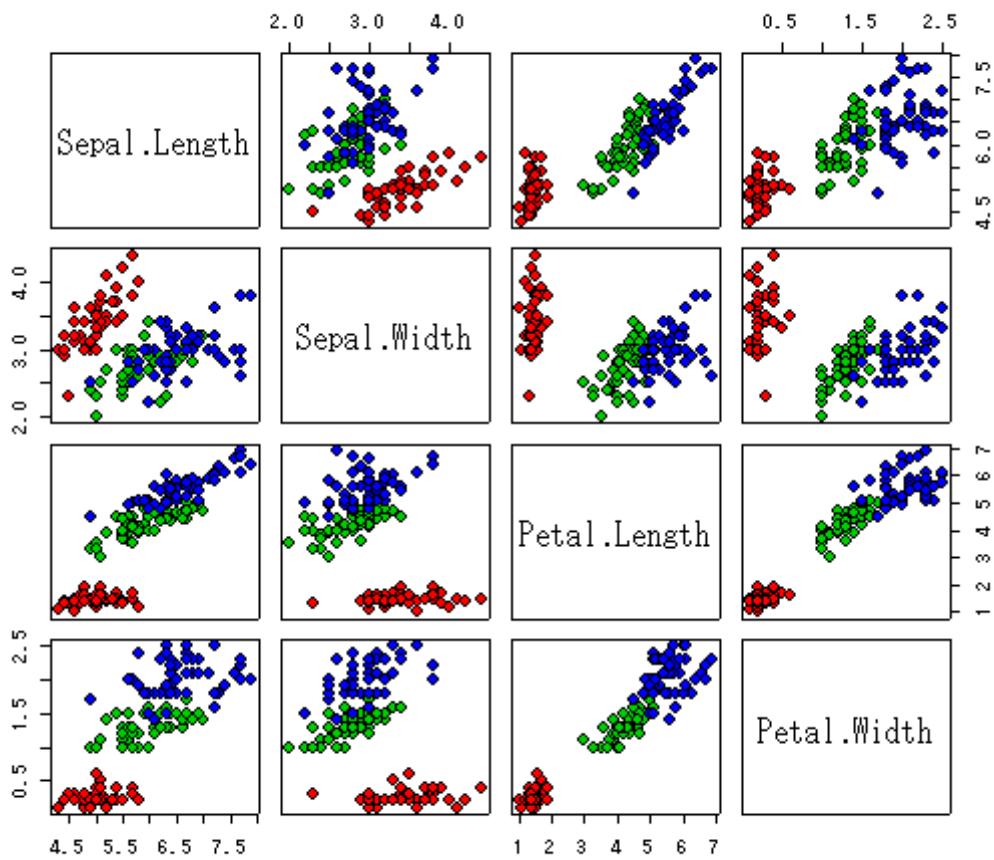


図 17 iris データの対散布図

で種類ごとに色付けされた対散布図が作成される。pch = 21 は円形マーク、bg = c("red", "green3", "blue")は色の指定、[unclass(iris\$Species)]は、色付けは iris データの Species で示された順に行うことを指定している。

7. 3次元グラフ

3 変数を用いて、3 次元空間でデータを配置した 3 次元散布図の作成について説明する。パッケージを用いると図 18 に示すような 3 次元グラフを作成することができる。

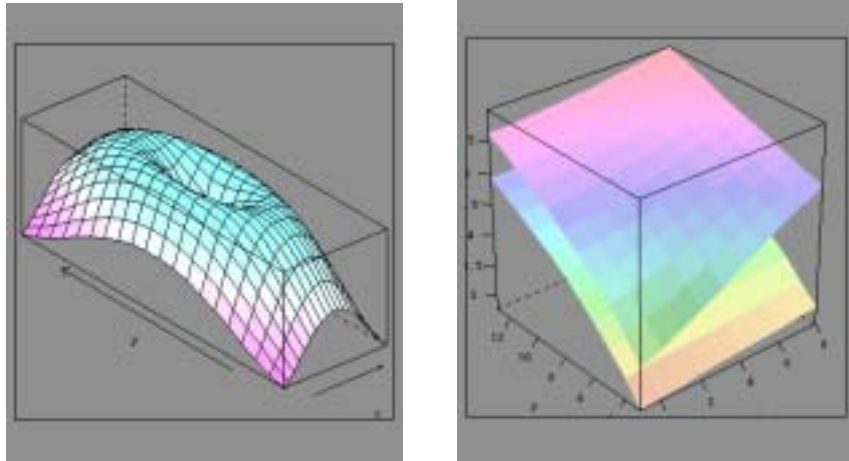


図 18 3次元グラフのサンプル

パッケージ `lattice`、`scatterplot3d` には 3次元散布図を描く関数が含まれている。

`Package` のメニューから `lattice` を選択し、[OK]をクリックするか、コンソールから `library(lattice)` と入力し、[Enter]キーを押すか、いずれかの方法でパッケージ `lattice` を R 読み込む。

`lattice` を読み込むと 3次元散布図作成の関数 `cloud` の作業環境が整ったことになる。その次に必要となるのはデータである。説明の便利のためここでも `iris` データを用いる。次のコマンドを実行すると図 19 のような図が返される。

```
>library(lattice)
>cloud(iris[,1]~iris[,2]*iris[,3],col="blue",pch=19,data = iris)
```

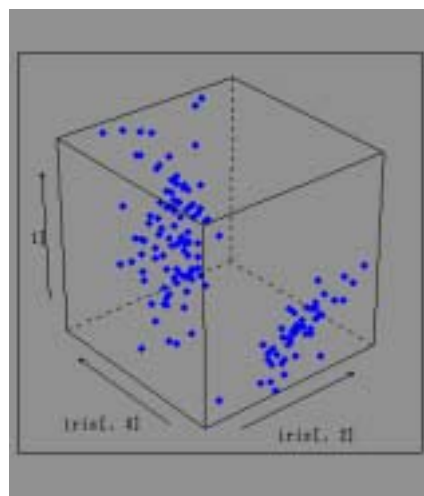


図 19 iris の 3次元散布図 1

cloud の最も簡潔な書き式は `cloud(formula, data)` である。formula は data のなかのどの変数をどの軸に対応させるかを指定する引数である。その書き式は `z ~ x * y` である。z は縦軸、x,y は両横軸である。"~"はチルタと呼ぶ。用いるデータの列の名前が付いている場合は、列の名前で次のように指定してもよい。次のコマンドを実行すると図 20 のような品種別に色分けした散布図が得られる。

```
>cloud(Sepal.Length ~ Petal.Length * Petal.Width, data = iris, groups = Species)
```

引数 groups は自動的に色付けを行う際に、グループに関するベクトルを指定するのに用いる。図 20 と同じにするには上記のコマンドに引数 `pch=c(16,17,18),col=c(1,2,4)` を付け加え、マークの種類と色を指定する必要がある。

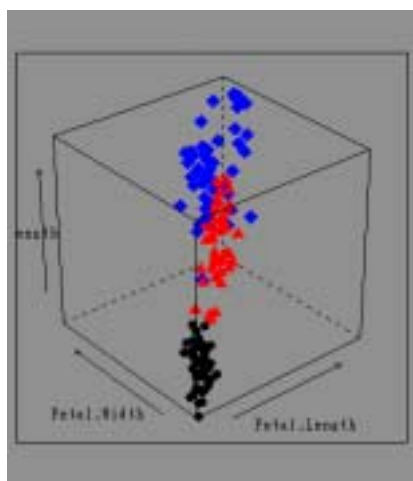


図 20 iris の 3 次元散布図 2

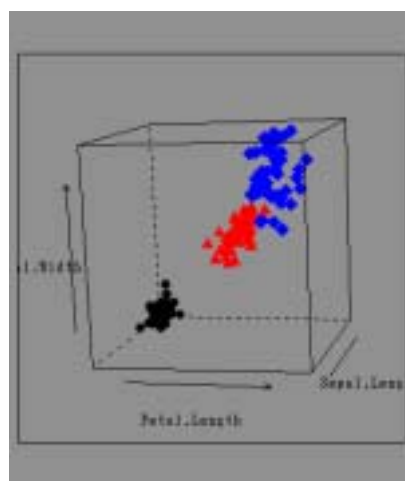


図 21 iris の 3 次元散布図 3

3 次元の回転角度は引数 `screen = list(z =, x=,y =)` を用いて調整することが可能である。上記のコマンドに引数 `screen = list(z = 0, x=10,y =-15)` を付け加えると図 21 のように回転した 3 次元散布図が作成される。

パッケージ graphics 中の関数 `persp` を用いて 3 次元のグラフを作成することもできる。次のコマンドで $z = y^2 - x^2$ 、 $-5 \leq x \leq 5$ 、 $-5 \leq y \leq 5$ の馬鞍形の 3 次元グラフが作成される。

```
>x <- seq(-5, 5, length=50)
>y <- x
>f <- function(x,y) {r <- (y^2)-(x^2)}
>z <- outer(x, y, f)
>persp(x, y, z,theta = 60,phi = 25,expand = 0.8,shade=.01,col = "lightblue")
```

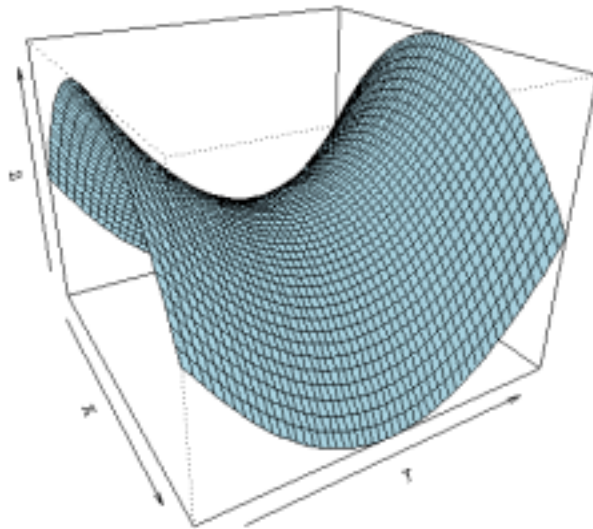


図 22 馬鞍形の 3 次元グラフ

R にはニュージーランドのオークランドの Maunga Whau 火山のデータ `volcano` がある。このデータセットは 10m x 10m 格子ごとの Maunga Whau の標高地形データ情報で、87 行 61 列のマトリックスで、北から南の方向の格子は行に、西から東の方向の格子は列に対応している。このデータと関数 `persp` 用いて次のコマンドで図 23 に示す 3 次元グラフを作成することができる。

```
>data(volcano)
>z <- 5 * volcano      # 見やすくするため高さを 5 倍に誇張
>x <- 10 * (1:nrow(z)) # 南北方向の 10 メートルの格子の辺
>y <- 10 * (1:ncol(z)) # 東西方向の 10 メートルの格子の辺
>par(bg = "skyblue")  #背景の色設定
>persp(x, y, z, theta = 115, phi = 20, col = "lightgreen", scale = FALSE,ltheta = -120,
shade = 0.75, border = NA, box = FALSE)
```

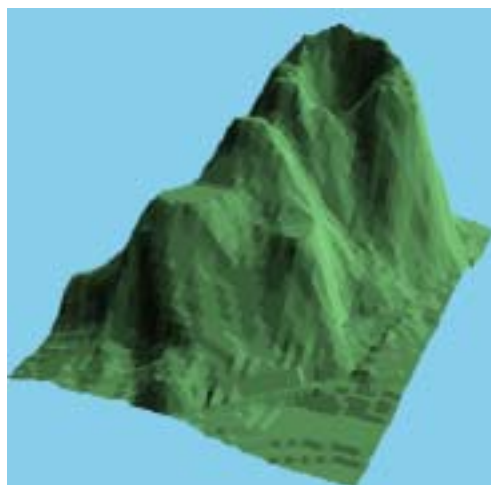


図 23 volcano データの 3 次元グラフ

7. 等高線グラフ

パッケージ `graphics` には関数 `contour`、`filled.contour`、`image`、パッケージ `lattice` の中の関数 `levelplot` など等高線のグラフを作成することができる。次にニュージーランド Maunga Whau 火山のデータ `volcano` を用いた等高線のグラフ作成の例を示す。

```
>contour(volcano,main="contour(volcano)の等高線グラフ")
>image(volcano,main="image(volcano)の等高線グラフ")
```

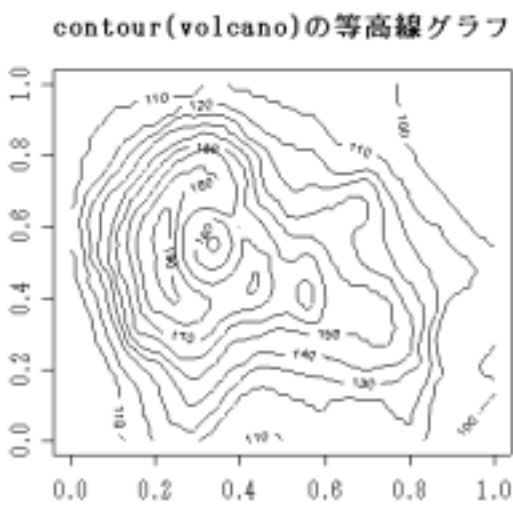


図 24 contour(volcano)によるグラフ

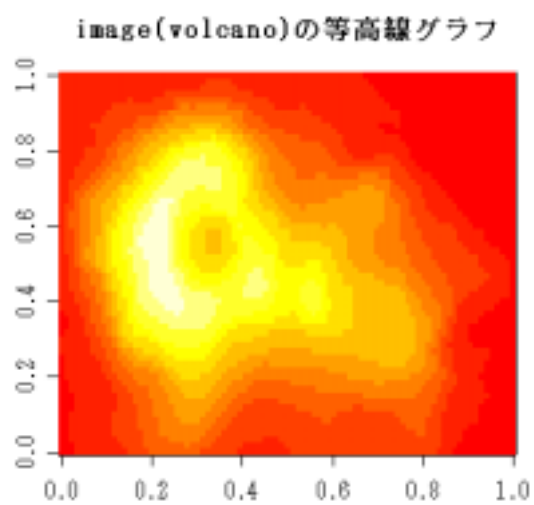


図 25 image(volcano)によるグラフ

```
>image(volcano,main="image(volcano)と\n contour(volcano,add=T)\n を組み合わせた等\n 高線グラフ")
>contour(volcano,add=T)
```

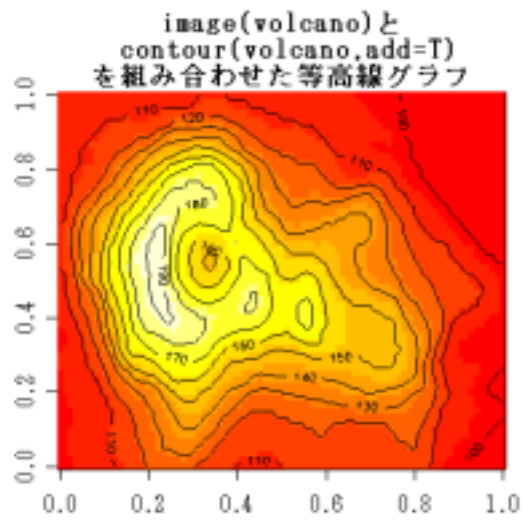


図 26 関数 image と contour によるグラフ

```
>filled.contour(volcano, color = terrain.colors)
>levelplot(volcano,main="levelplot(volcano)の等高線グラフ")
```

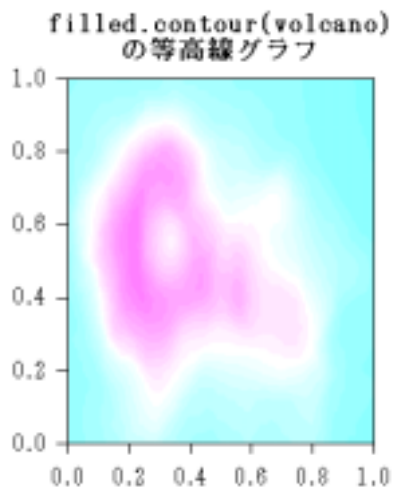


図 27 関数 filled.contour によるグラフ

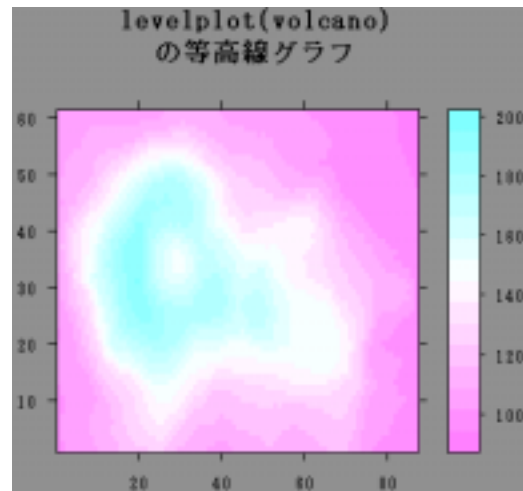


図 28 関数 levelplot によるグラフ