クレジット:
UTokyo Online Education 統計データ解析 I 2017 小池祐太

東京大学 THE UNIVERSITY OF TOKYO

CC BY NC ND

統計データ解析Ⅰ (平成２９年度)

東京大学大学院数理科学研究科
統計データ解析教育研究グループ

村田　昇 (早稲田大学, 東京大学)
吉田朋広 (東京大学)
小池祐太 (首都大学東京, 東京大学)

第 4 章 データのプロット

記述統計量と並んでデータ全体の特徴や傾向を把握するために効果的な方法は, データを可視化することである. R の基本パッケージ graphics に用意されている作図機能はきわめて多彩であり，これらを適切に組み合わせることによって様々な種類のグラフを描くことができる．以下では，いくつかの代表的な描画関数を取り上げて解説する．

描画関連の関数は色，線種や線の太さ，あるいは図中の文字の大きさなどを指定するために，多彩なオプションを用意しているので，必要に応じて関数 help()(ヘルプの表示) と example()(例題の表示) を利用して欲しい．

4.1. 基本的な描画

描画において基本となるのは関数 plot() である．

関数 sin() のように 1 変数の関数として定義されているものは，定義域を指定してやればそのまま表示することができる．関数を追加するにはオプション add とともに関数 curve() を用いれば良い．

また，関数 plot() に同じ長さの二つのベクトルを与えると，同じ番号の要素からなる点の組 (x, y) をプロットして，その**散布図**を描くことができる．

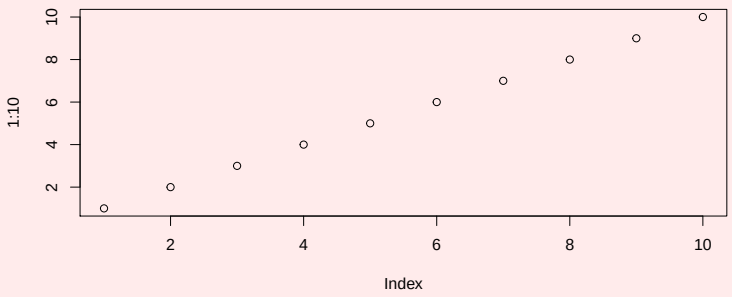
プロットの種類 (点や線) を指定するにはオプション type を用いる．'p' で点 (point)，'l' で点列を順に結んだ線 (line) が描かれる．なお，オプションに与える文字列は '(シングルクォート) か "(ダブルクォート) で囲む必要がある．

オプション col で"色の名前"を指定することにより点や線の色を変えることができる．R で指定することのできる色の名前は関数 colors() で照会することができる．

関数 plot() で描いた図中に更に線を追加するには関数 lines() を，点を追加するには関数 points() を用いる．

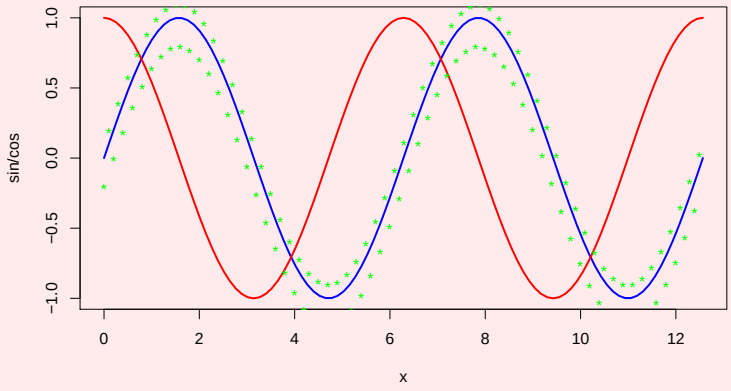
これ以外にも関数 plot() は様々なオプションを指定することができるので, help(plot) および help(plot.default) を参照して欲しい．

```
> ### ベクトルのプロット
> plot(1:10)
```

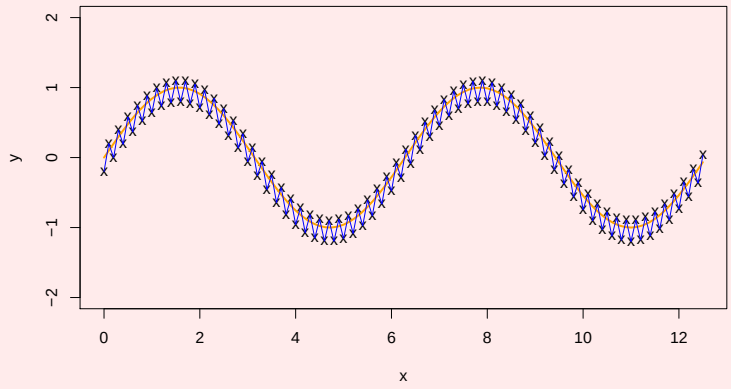



```
> ### 擬似データの作成
> x <- seq(0, 4*pi, by=0.1)
```

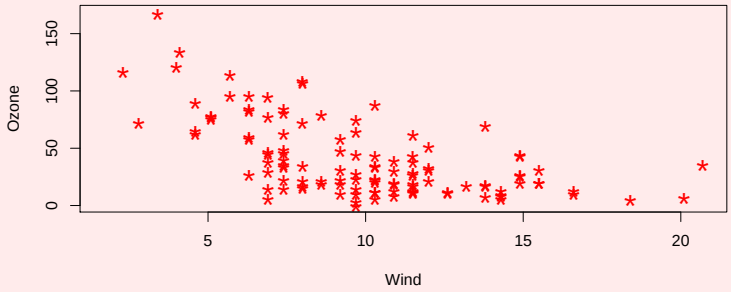
```
> y <- sin(x) + rep_len(c(-0.2, 0.1), length(x))
> ### 関数の描画
> plot(sin, 0, 4*pi,
+      col="blue", # グラフの線の色
+      lwd=2, # グラフの線の太さ
+      ylab="sin/cos" # y 軸のラベル
+      )
> curve(cos,
+       add=TRUE, # グラフを上書き
+       col="red", lwd=2)
> points(x, y, col="green", pch="*") # 点を追加．pch は点の形を指定
```




```
> ### (x,y) データの描画
> plot(x, y, type="p", pch="x", ylim=c(-2,2)) # ylim で値域を指定
> curve(sin, add=TRUE, col="orange", lwd=2)
> lines(x, y, col="blue") # 折れ線を追加
```



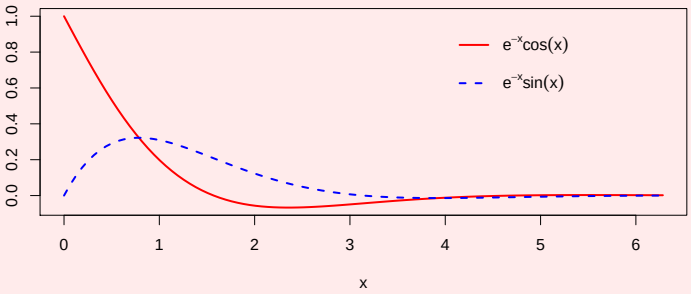

```
> ### データフレームを用いた散布図 (airquality を利用)
> plot(Ozone ~ Wind, data=airquality, pch="*", col="red", cex=2) # cex は点の大きさの倍率を指定
```



(plot3.r)

関数 legend() によってグラフに凡例を追加することができる. なお, 以下の例で見るように, R には数式を扱う機能がある. 詳細は help(plotmath) を参照してほしい.

```
> f <- function(x) exp(-x) * cos(x)
> plot(f, 0, 2 * pi, col = "red", lwd = 2, ylab = "")
> g <- function(x) exp(-x) * sin(x)
> curve(g, lty = 2, # グラフの線の形式. 2 はダッシュ線に対応
+       add = TRUE, col = "blue", lwd = 2)
> legend(4, # 凡例の左上の x 座標
+        1, # 凡例の左上の y 座標
+        legend = c(expression(e^{-x}*cos(x)), expression(e^{-x}*sin(x))), # 凡例
+        lty = c(1, 2), lwd = 2, col = c("red", "blue"), # ここのパラメーターは通常グラフと合わせる
+        bty = "n", # 凡例の枠線の形式 (オプション). "n"は書かない
+        y.intersp = 2 # 行間の指定 (オプション)
+        )
```



(legend.r)

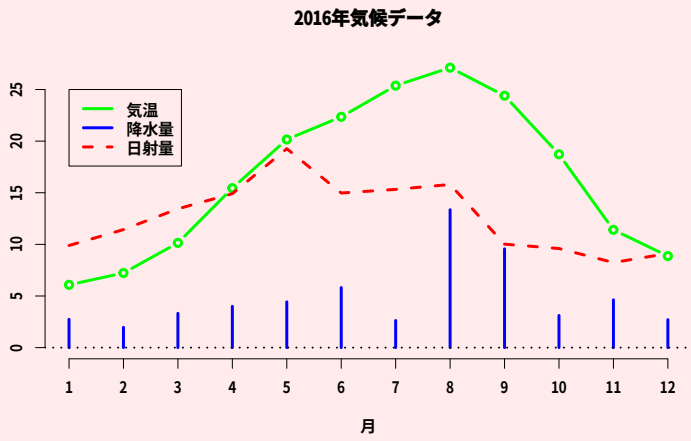
なお, OS によっては日本語を含む図を描画すると文字化けする場合がある. その場合, 関数 par() のオプション family に適当なフォントファミリーを指定することで文字化けを回避できる場合がある. 例えば, Mac OS のデフォルトの設定では日本語を含む図は文字化けしてしまうが, 以下のコマンドをコンソール上で実行することで文字化けを回避できる.

```
> par(family = "HiraginoSans-W4")
```

(family.r)

上の例ではフォントファミリーとしてヒラギノ角ゴシック W4 を指定している (数字を変えると太さが変わる).

```
> ### 東京都の 2016 年の気候データによる例
> ### 気象庁のホームページより取得
> ### http://www.data.jma.go.jp/gmd/risk/obsdl/index.php
> ### 東京都の 2016 年の各日の平均気温 (℃)・降水量 (mm)・全天日射量 (MJ/u)・
> ### 平均風速 (m/s) を記録したデータセット kikou2016.csv
> kikou <- read.csv("kikou2016.csv", fileEncoding = "sjis")
> ## 月ごとの平均をプロットする
> (x <- aggregate(kikou[ ,-c(1,2)], by = list(月 = kikou$月),
+                 FUN = "mean")) # 月ごとの平均を計算
   月       気温     降水量      日射量       風速
1   1  6.080645  2.741935  9.891290 2.393548
2   2  7.227586  1.965517 11.431034 2.889655
3   3 10.141935  3.322581 13.443226 2.812903
4   4 15.446667  4.000000 14.909667 3.263333
5   5 20.161290  4.435484 19.268065 3.383871
6   6 22.353333  5.816667 14.974000 2.926667
7   7 25.374194  2.629032 15.326129 2.674194
8   8 27.116129 13.354839 15.801935 3.096774
9   9 24.400000  9.566667 10.021000 2.436667
10 10 18.722581  3.112903  9.597742 2.441935
11 11 11.406667  4.633333  8.243000 2.466667
12 12  8.864516  2.709677  9.112581 2.641935
> plot(x$気温, type = "b", lwd = 3, col = "green", ylim = c(0, max(x$気温)),
+      xlab = "月", ylab = "", main = "2016 年気候データ", # グラフタイトル
+      axes = FALSE) # 軸を書かない
> axis(1, 1:12, 1:12) # x 軸の作成
> axis(2) # y 軸の作成
> lines(x$降水量, type = "h", lwd = 3, col = "blue")
> lines(x$日射量, lwd = 3, lty = 2, col = "red")
> abline(0, 0, lwd = 2, lty = "dotted") #  y=0 の線を引く
> legend(1, 25, legend = c("気温","降水量","日射量"),
+        col = c("green", "blue", "red"), lwd = 3,
+        lty = c(1, 1, 2))
```



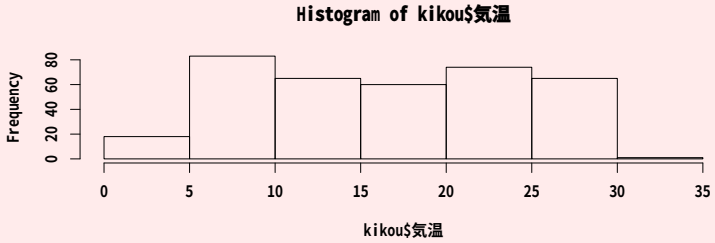
(plot-kion.r)

作成したグラフは保存することができる. RStudio の機能を使う場合, 右下ペインの「Plots」タブの「Export」をクリックすると, 形式やサイズを指定して保存できる (もしくはクリップボードにコピーもできる). コマンドで実行することも可能であるが, それについての詳細は `help(png)` や `help(dev.copy)` を参照してほしい.

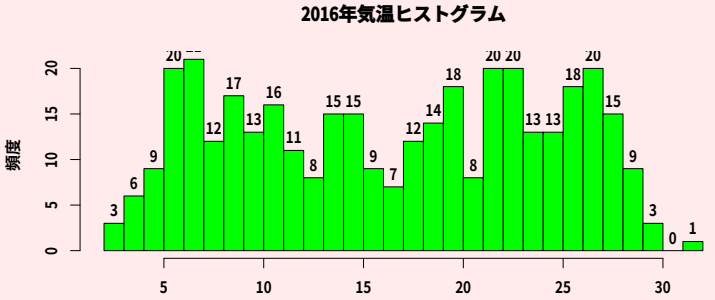
4.2. ヒストグラム

データの頻度分布を表すヒストグラムを描画するには関数 hist() を用いる．これ以外にも凝ったヒストグラムを書くための関数がいくつか用意されているが，これらについては help.search("histogram") を参照して欲しい．

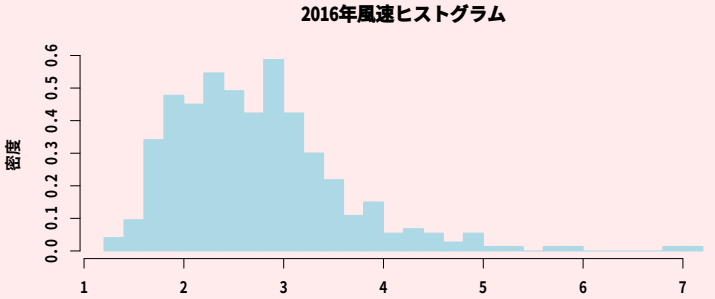
```
> ### 気候データによる例
> ### 基本的なヒストグラムの描画
> kikou <- read.csv("kikou2016.csv", fileEncoding = "sjis")
> hist(kikou$気温)
```



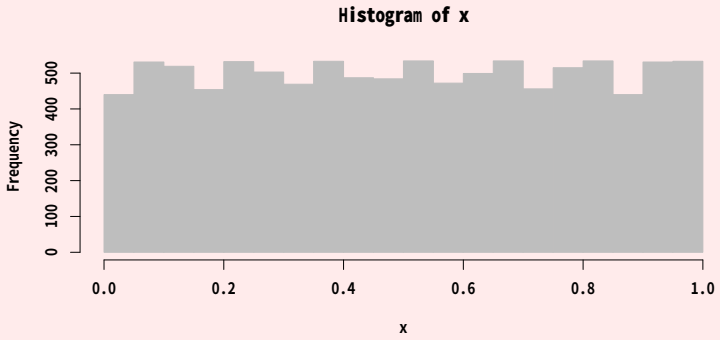

```
> hist(kikou$気温,
+      xlab = "", ylab = "頻度",
+      breaks=25, # ビンの数を約 25 に設定
+      labels = TRUE, # 各ビンの度数を表示
+      col = "green", main="2016 年気温ヒストグラム")
```




```
> hist(kikou$風速, freq = FALSE, # 全体に対する割合で表示
+      xlab = "", ylab = "密度", breaks=25, col = "lightblue",
+      border = "lightblue", # 長方形の境界の色
+      main="2016 年風速ヒストグラム")
```



```
> ### Weyl の一様分布定理の確認
> ### a が無理数のとき，数列 a, 2a, 3a, ... の小数部分は
> ### 区間 (0,1) 上に均一に現れる
> a <- pi # 無理数
> n <- 10000
> x <- (1:n) * a
> x <- x - floor(x) # 小数部分の計算 (floor はいわゆる Gauss 記号)
> hist(x, breaks = 20, col = "gray", border = "gray")
```

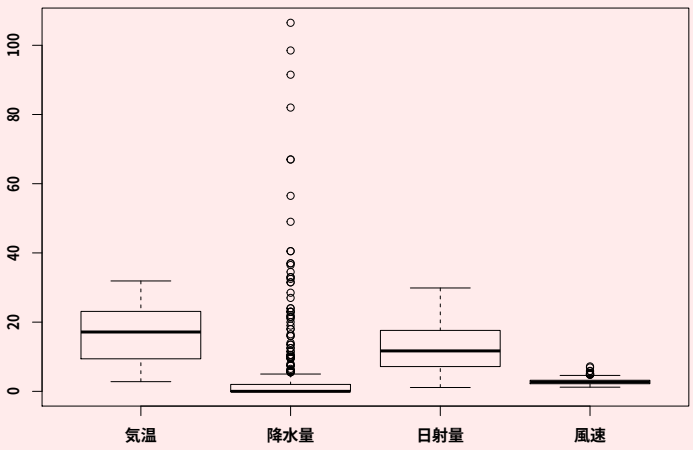


(hist3.r)

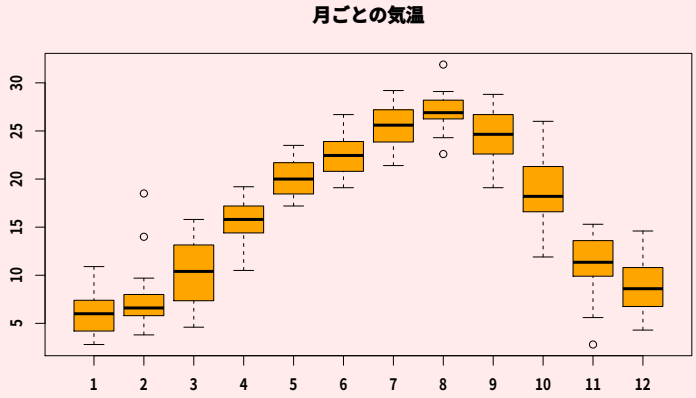
4.3. 箱ひげ図

複数のデータの分布を比較する際, 観測数が大きく異なるなどヒストグラムでの比較が難しい場合がある. 複数のデータの分布の違いを簡便に見るには箱ひげ図 (boxplot) が良く用いられるが, これは関数 `boxplot()` で描くことができる.

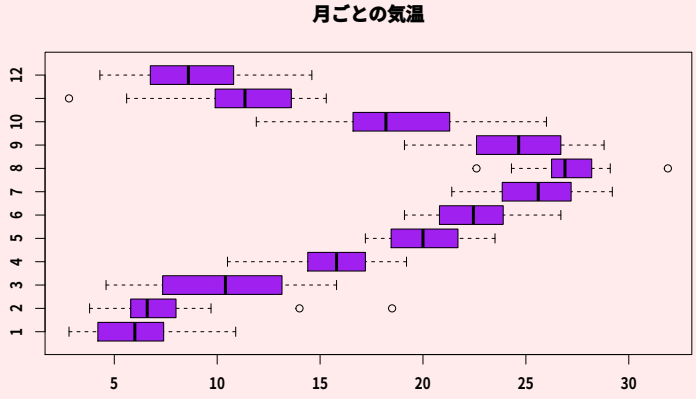
```
> ### データフレームを用いた表示例 (気候データを利用)
> kikou <- read.csv("kikou2016.csv", fileEncoding = "sjis")
> boxplot(kikou[ ,-c(1,2)]) # 月日は除く
```




```
> ### 月ごとに気温を分類した場合
> boxplot(気温 ~ 月, data=kikou, col="orange", main = "月ごとの気温")
```




```
> boxplot(気温 ~ 月, data=kikou, col="purple", main = "月ごとの気温", horizontal=TRUE)
```

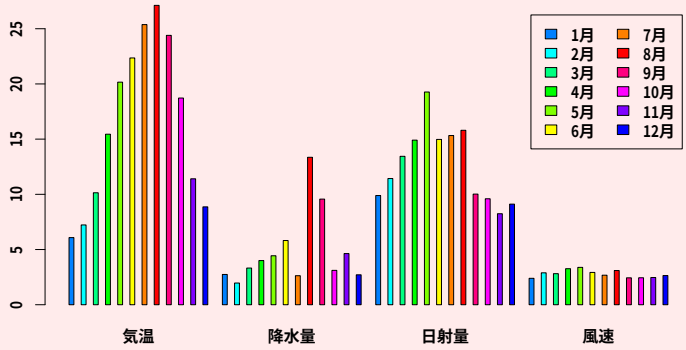


(boxplot3.r)

4.4. 棒グラフ

関数 barplot() によって棒グラフを作成できる. barplot() の第 1 引数はベクトルまたは行列でなければならないことに注意すること.

```
> ### 気候データによる例
> kikou <- read.csv("kikou2016.csv", fileEncoding = "sjis")
> ## 月ごとに各変数の平均を計算
> x <- aggregate(kikou[ ,-c(1,2)], by = list(月 = kikou$月),
+                FUN = "mean")
> # 棒グラフの作成
> barplot(as.matrix(x[ ,-1]), # 第 1 引数はベクトル/行列でなければならない
+         col = rainbow(12)[c(8:1,12:9)], # 12 色に色分け
+         beside = TRUE, # 棒グラフを横に並べる
+         space = c(1.5, 3), # 棒グラフ間・変数間のスペースを指定
+         legend.text = paste0(x[ ,1], "月"), # 凡例の指定
+         args.legend = list(ncol = 2) # 凡例を 2 列にして表示
+         )
```

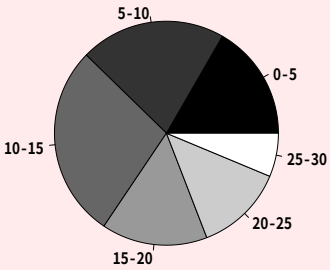


(barplot.r)

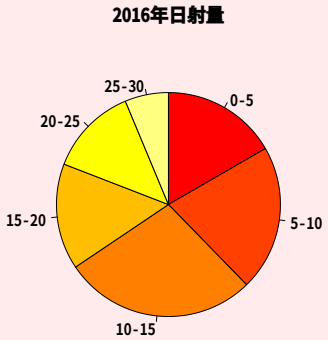
4.5. 円グラフ

円グラフは関数 pie() で描くことができる.

```
> ### 関数 pie による円グラフの作図
> kikou <- read.csv("kikou2016.csv", fileEncoding = "sjis")
> z <- hist(kikou$日射量, breaks=5, plot=FALSE) # 5 つ程度に分類
> x <- z$count
> y <- z$breaks
> names(x) <- paste(y[-length(y)], y[-1], sep="-")
> pie(x, col=gray(seq(0,1,length=length(x))))
```




```
> pie(x, clockwise=TRUE, col=heat.colors(length(x)), main = "2016 年日射量")
```

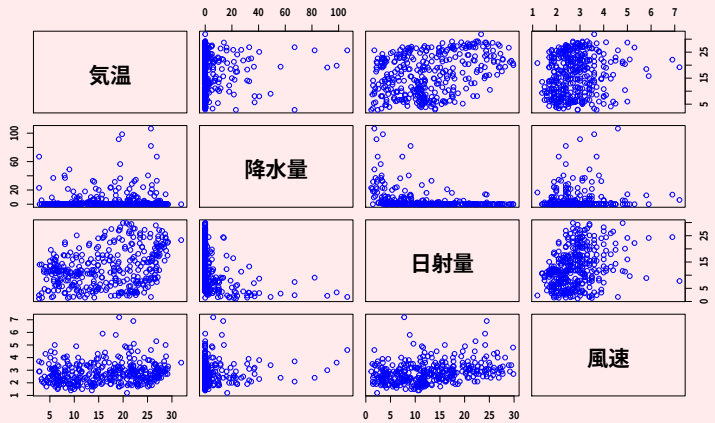


(pie.r)

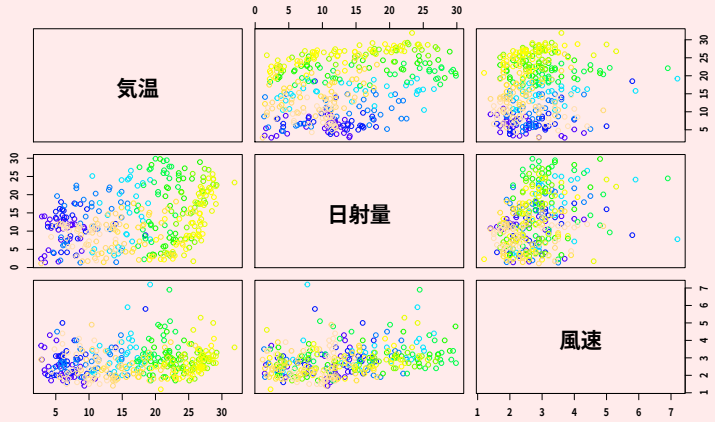
4.6. 散布図行列

多次元データの変数間の関係を概観するために，2 つの変数間の散布図を複数行列状に並べた図を用いることがある．これは関数 pairs() によって作成することができる (関数 plot() でも同じことができる).

```
> ### 関数 pairs による散布図の作図
> kikou <- read.csv("kikou2016.csv", fileEncoding = "sjis")
> pairs(kikou[,-c(1,2)], col="blue")
> # plot(kikou[,-c(1,2)], col="blue") でも同じ図が描ける
```




```
> pairs(~ 気温 + 日射量 + 風速, data = kikou,  # 表示する項目を指定
+       col=topo.colors(12)[kikou$月]) # 月毎に異なる色で表示
```

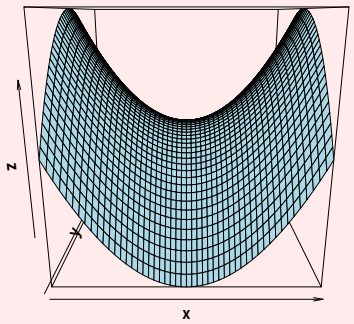


(pairs.r)

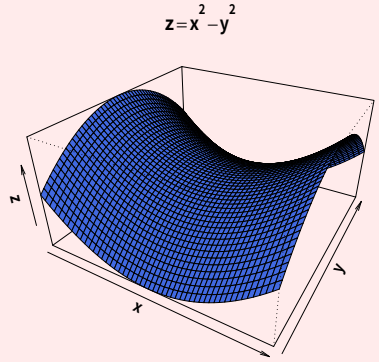
4.7. 3 次元のグラフ

3 次元のグラフを 2 次元に射影した俯瞰図は，関数 persp() を用いて描くことができる．視線の方向はオプション theta と phi で極座標を指定することによって制御することができる．パッケージ scatterplot3d には, 3 次元の散布図を書くための関数 scatterplot3d() が用意されている．

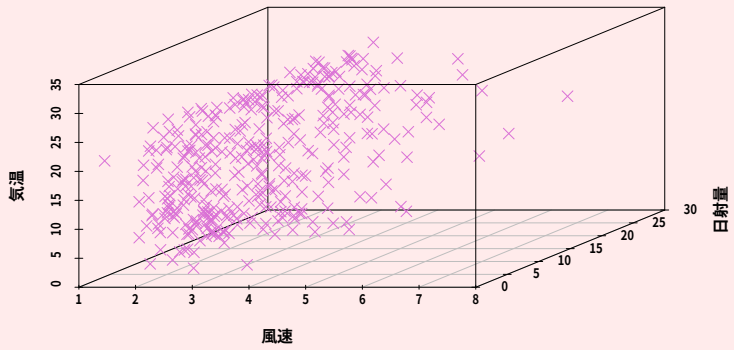
```
> ### 関数 persp による 2 変数関数の俯瞰図
> f <- function(x,y) x^2 - y^2
> x <- seq(-3, 3, length=51) # x 座標の定義域の分割
> y <- seq(-3, 3, length=51) # y 座標の定義域の分割
> z <- outer(x, y, f) # z 座標の計算
> persp(x, y, z, col="lightblue")
```




```
> persp(x, y, z, theta=30, phi=30, expand=0.5, col="royalblue",
+       main = expression(z==x^2-y^2))
```




```
> ### 3次元散布図 (パッケージ scatterplot3d を利用)
> install.packages("scatterplot3d") # パッケージのインストール
> library(scatterplot3d) # パッケージのロード
> kikou <- read.csv("kikou2016.csv", fileEncoding = "sjis")
> dat <- subset(kikou, select = c("風速", "日射量", "気温"))
> scatterplot3d(dat, pch = 4, color = "orchid", cex.symbols = 1.5)
```



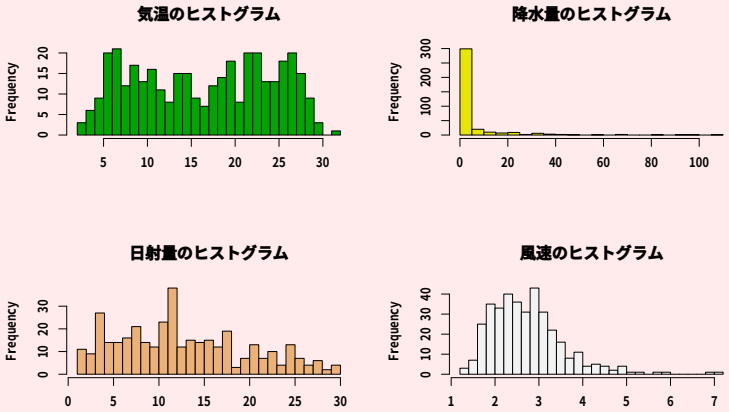
(plot3d.r)

4.8. プロット環境の設定

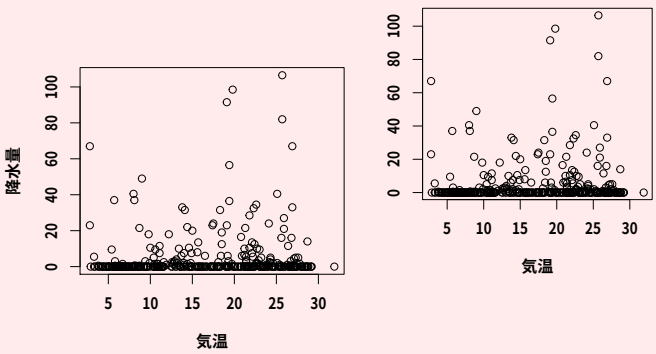
プロットの際の線の種類や色, 点の形等のデフォルト値は関数 par() で設定できる. 設定可能なグラフィックスパラメーターは help(par) で確認できる. 特に, 以下の例のように, 関数 par() によってプロット環境の設定 (複数図の配置, 余白の設定) ができる.

```
> ### 複数図の配置
> ## 気候データの各変数のヒストグラムを 1 つの画面に配置
> kikou <- read.csv("kikou2016.csv", fileEncoding = "sjis")
> op <- par(mfrow = c(2,2)) # 画面を 2行2列 に分割し, 行ごとにプロットしていく
> # par(mfcol = c(2,2)) で列ごとにプロットできる
> cl <- terrain.colors(4) # 色を用意
```

```
> nam <- colnames(kikou)[-(1:2)] # ヒストグラムを作成する変数名
> ## 第 1 変数のヒストグラムの作成
> hist(kikou[ ,nam[1]], col = cl[1], breaks = 25, xlab = "",
+      main = paste0(nam[1], "のヒストグラム"))
> ## 第 2 変数のヒストグラムの作成
> hist(kikou[ ,nam[2]], col = cl[2], breaks = 25, xlab = "",
+      main = paste0(nam[2], "のヒストグラム"))
> ## 残りは for 文で作成
> for(i in 3:4){
+   hist(kikou[ ,nam[i]], col = cl[i], breaks = 25, xlab = "",
+        main = paste0(nam[i], "のヒストグラム"))
+ }
> par(op) # 設定解除
```




```
> ### 余白の設定
> op0 <- par(mfrow = c(1,2))
> plot(kikou[ ,3:4])
> op <- par(mar = c(9,2,1,6))
> ## 下・左・上・右の順で余白を設定
> ## デフォルトは par(mar = c(5,4,4,2)+0.1)
> plot(kikou[ ,3:4])
> par(op0); par(op) # 設定解除
```



(par.r)

4.9. その他

データの可視化は, データ解析において基本的かつ有効な方法であるのみならず, 分析結果を他の人々に説明する際の資料としても必須のものである. そのため, R の

グラフィック機能を拡張するためのパッケージも多数開発されている．その中でも，近年利用が広まっているものにパッケージ ggplot2 がある．ggplot2 は，統一的な文法で系統的に美しいグラフを描くことを目的として開発されているパッケージである．基本設計は確定しているが，細かい部分は現在も頻繁に開発が進められている．用意されている関数の細かな情報については，

http://docs.ggplot2.org/

に詳しい例題とともにまとめられている．また，良く使われる関数については，簡潔に纏めた 2 頁のシート

http://www.rstudio.com/wp-content/uploads/2015/12/ggplot2-cheatsheet-2.0.pdf

が用意されているので，興味に応じて参照してほしい．

4.10. 参考文献

1. 金明哲著「R によるデータサイエンス (第 2 版)」(第 4 章), 森北出版 (2017 年).
2. U. リゲス著, 石田基広訳「R の基礎とプログラミング技法」(第 8 章), 丸善出版 (2012 年).