# Rによる作図のアドバイス

080530 玉木一郎
（森林生態生理学研究分野）

# はじめに

今回は主にパッケージgraphicsとgrDevices*を用いた作図の仕方について紹介します。話の流れは以下の通りです。応用編と言いつつ、割と基礎的な話になってしまいました...。

- Rによる作図の概要
- 各作図関数の紹介
- フォントの指定方法と図の出力方法

*これらのパッケージは起動時に自動的に読み込まれるのでlibrary()しなくて良いです。

# Rが行っていること

Rでの作図は基本的に**作図関数**と**作図デバイス**によって行われます。普段作図している人も後者はあまり意識していないかもしれません。

- **作図関数**：散布図やヒストグラム等を描かせる命令
- **作図デバイス**：図が出力されるウィンドウやファイル

# 高水準・低水準作図関数

作図関数には**高水準作図関数**と**低水準作図関数**の2つ*があり、これらを組み合わせて自分の好みにあった作図を行うことができます。

- **高水準作図関数**：一回のコマンドで一つの図を完成させることができる関数（`plot()`や`hist()`）。
- **低水準作図関数**：高水準作図関数で作成した図に点や凡例などを描き足す関数（`points()`や`legend()`）。

*対話的～というものもありますが今回は略。

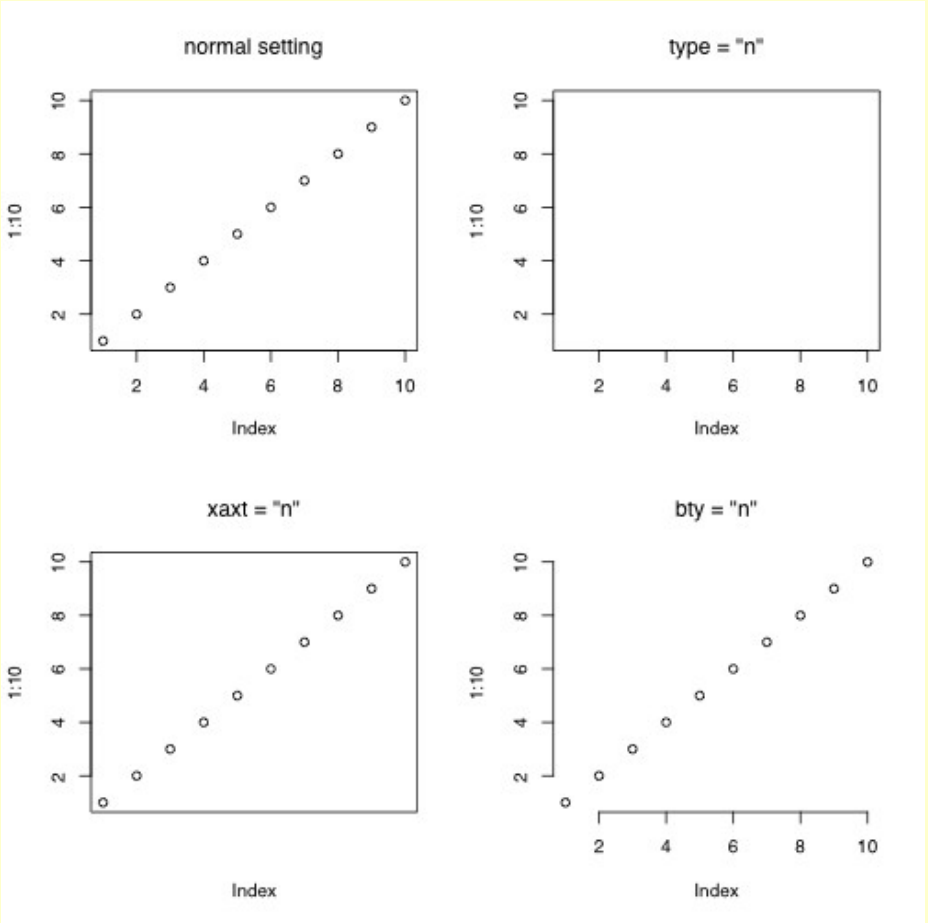
# Rによる作図の流れ

1. 高水準作図関数で基本的な図を描く。
2. 低水準作図関数で軸や点、線、文字列などを描き込む*。

*高水準作図関数でも引数に`add = T`とすれば重ね描きはできます。

# 基礎となる図を描く

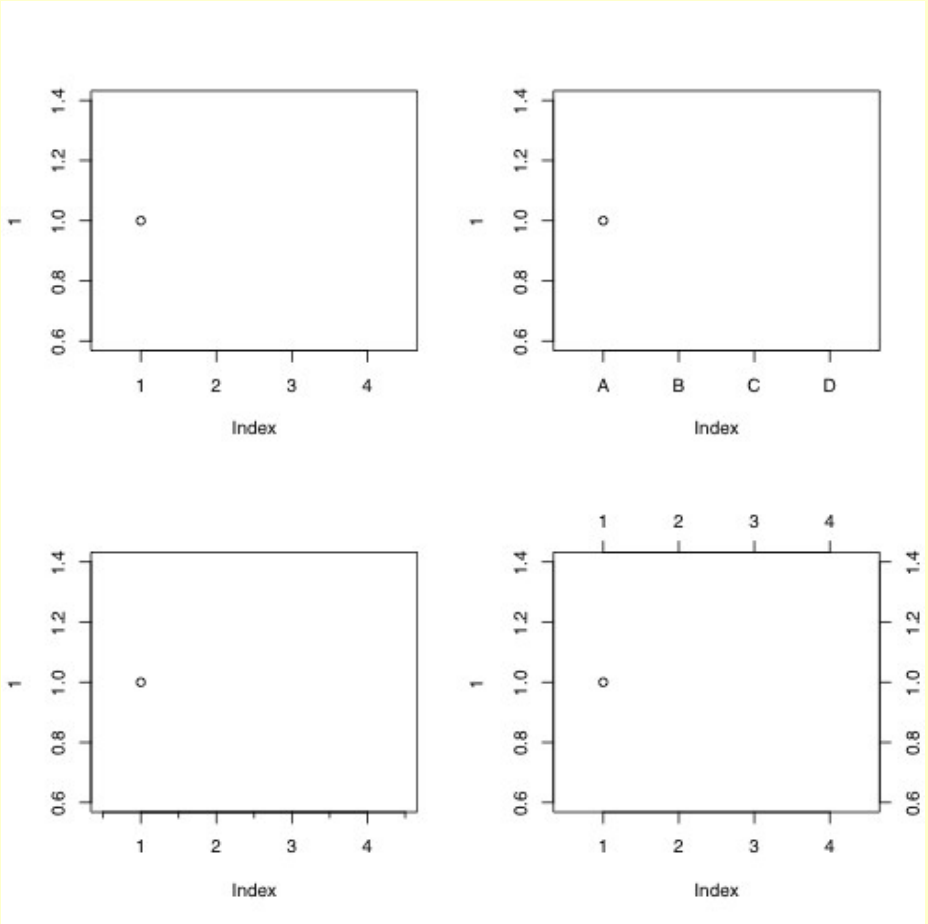
まずは高水準作図関数で基礎となる図を描きます。このとき、点や軸が気に入らない場合は表示しないでおきます。




```r
### 標準
plot(1:10)

### 点をプロットしない
plot(1:10, type = "n")

### x軸を表示しない
plot(1:10, xaxt = "n")

### 枠を表示しない
plot(1:10, bty = "n")
```

# 軸を描き足す

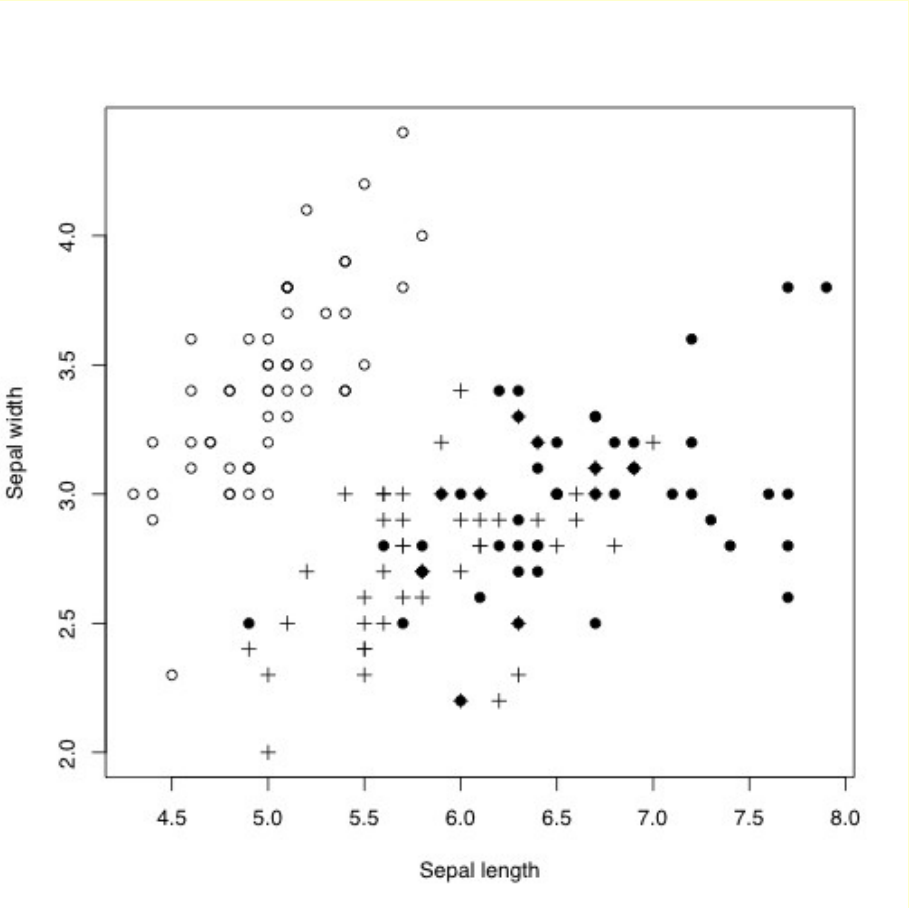
axis()で軸を描き足します。かなり自由に操作できます。




```
### 普通に描いた場合
plot(1, xlim = c(0.5, 4.5))

### 軸を消して描き換える
plot(1, xlim = c(0.5, 4.5), xaxt = "n") # 軸を消す
axis(side = 1,    # 軸を描く場所（下）
     at = 1:4,    # 目盛の場所
     labels = c("A", "B", "C", "D")) #ラベル

### 二重目盛にする
plot(1, xlim = c(0.5, 4.5))
axis(side = 1,
     at = seq(0.5, 4.5, by = 1),
     tcl = -0.25, # 目盛の長さ（デフォルト=-0.5）
     labels = F)   # ラベルは描かない

### 反対側にも軸をつける
plot(1, xlim = c(0.5, 4.5))
axis(side = 3, at = 1:4)                        # 上
axis(side = 4, at = seq(0.6, 1.4, by = 0.2)) # 左
```

# 点を描き足す

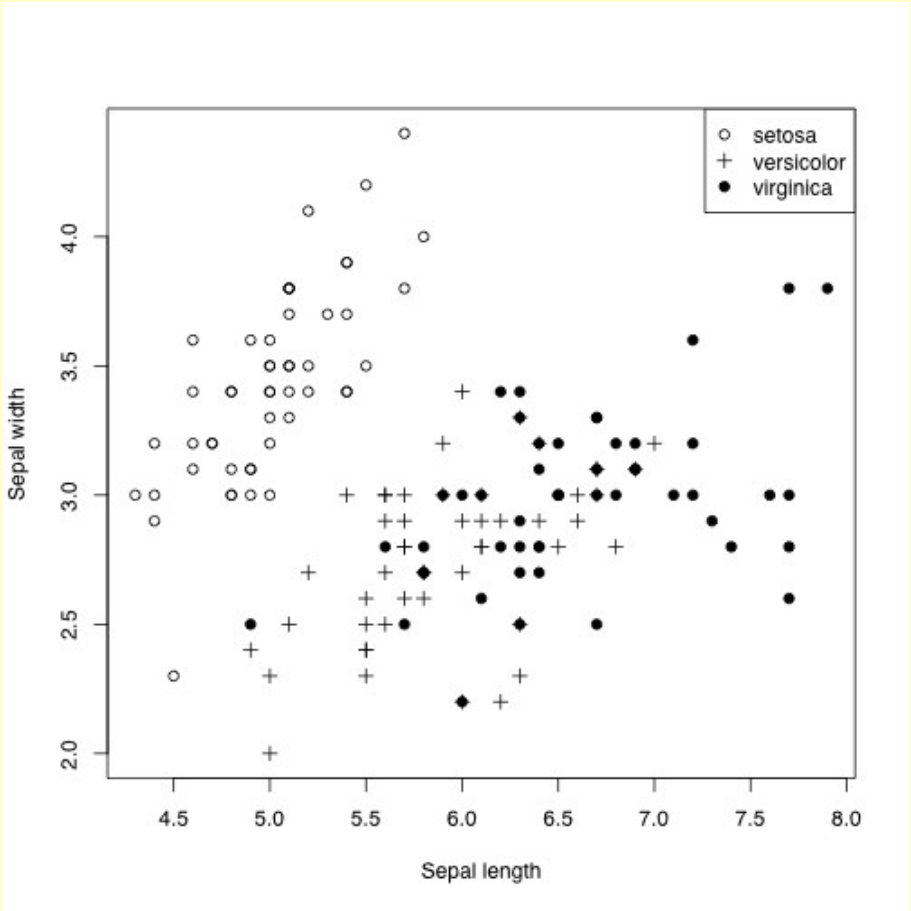
points()で点を描き足します。この程度だとplot()だけで描けないこともないのですが、まあ練習ということで。




```
### 組込みデータirisを使用
d <- iris

### まず枠だけ描く
plot(d$Sepal.Length, d$Sepal.Width, type = "n",
     xlab = "Sepal length", ylab = "Sepal width")

### Species毎に追加する
points(d$Sepal.Length[d$Species == "setosa"],
       d$Sepal.Width[d$Species == "setosa"],
       pch = 1) # 白丸

points(d$Sepal.Length[d$Species == "versicolor"],
       d$Sepal.Width[d$Species == "versicolor"],
       pch = 3) # 十字

points(d$Sepal.Length[d$Species == "virginica"],
       d$Sepal.Width[d$Species == "virginica"],
       pch = 16) # 黒丸
```

# 凡例を描き足す

legend()で凡例を先ほどのpoints()の例に続けて描き足します。

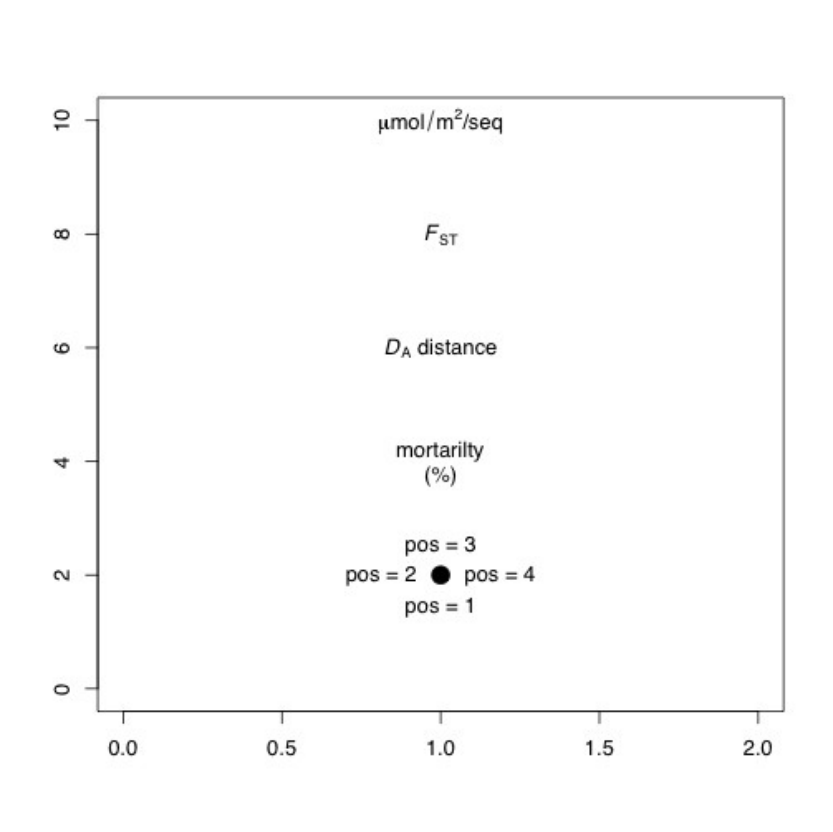



```
### 先ほどのpoints()の例に続けて
legend("topright", pch = c(1, 3, 16),
       legend = levels(d$Species))
```

通常はlegend()の最初の引数にx、yを指定して凡例を示す場所を決めるのですが、ここではtoprightを使いました。きれいに収めてくれるのでお勧めです。他にはbottomrightなどがあります。詳しくは?legendを参照して下さい。

# 文字を描き足す

text()では**文字**を、expression()を使うと**数式**を描き込むことができます。適当な文字列を例示しておくので参考にして下さい*。

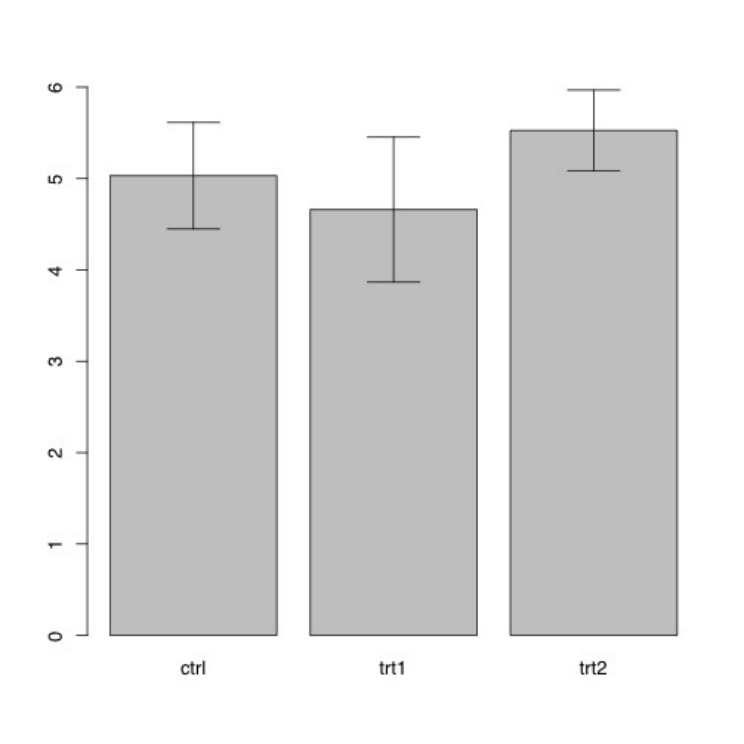



```
plot(1, xlim = c(0, 2), ylim = c(0, 10),
     type = "n", xlab = NA, ylab = NA)

text(1, 10,
     expression(paste(mu, mol/m^2, "/seq")))

text(1, 8,
     expression(italic(F)[ST]))

text(1, 6,
     expression(paste(italic(D)[A], " distance")))

text(1, 4,
     "mortarilty\n(%)") # \nで改行

points(1, 2, pch = 16, cex = 2)
text(1, 2, "pos = 1", pos = 1, offset = 1)
text(1, 2, "pos = 2", pos = 2, offset = 1)
text(1, 2, "pos = 3", pos = 3, offset = 1)
text(1, 2, "pos = 4", pos = 4, offset = 1)
```

*\（バックスラッシュ）は¥（円マーク）で読み替えて下さい。

# 線を描き足す

線を描き足す方法には様々な方法があり（points(..., type=“l”)やlines()、abline()、arrows()など）、状況に応じて適切なものを選択すると良いでしょう。ここではたまに聞かれるエラーバーの描き方を示しておきます。




```
### 組込みデータPlantGrowthを使用
d <- PlantGrowth
mean.weight <- tapply(d$weight, d$group, mean)
sd.weight <- tapply(d$weight, d$group, sd)
barplot(mean.weight, ylim = c(0, 6))

### 平均+-標準偏差のエラーバーを追加
arrows(c(0.7, 1.9, 3.1), mean.weight - sd.weight,
       c(0.7, 1.9, 3.1), mean.weight + sd.weight,
       angle = 90, code = 3)
```

# グラフィックパラメタの変更

par()でグラフィックパラメタを変更する際は、元々のパラメタを適当な変数に保存しておいて、作図終了後にもとに戻しておくのが良いです。

```
oldpar <- par(no.readonly = T)
  # 現在のパラメタを変数oldparに退避

par(mfrow = c(2, 1)) # 画面を2行1列に分割
hist(rnorm(100))
hist(rnorm(100))

par(oldpar) # 作業前のパラメタに戻す
hist(rnorm(100)) # 1行1列に戻っていることを確認
```

# 図中のフォントの指定 1

Rを使っているうちに（最も？）気になるのがフォントの指定です（気にならない人は気にしなくて良いです）。フォントの指定は**OSや出力するデバイス**によって異なります*。今回はWindowsユーザ向けに説明します。

- `windows()`、`png()`、`win.metafile()`など
  → 設定ファイルRdevgaで指定**。
- `pdf()`、`ps()`など
  → 引数`family`で指定。
  例えば`family = “Helvetica”`や`family = “Palatino”`

*どの環境でも半角英数は○○、全角は□□という分けた指定は基本的にできません。

**設定ファイルRdevgaについては安部君の補足資料を参照して下さい。

# 図中のフォントの指定 2

## ～pdf()やps()などに出力する際の注意！～

図中で和文フォントを用いた場合、当たり前ですがfamilyに和文フォントを指定しないと文字化けします。Japan1やJapan1Ryumin、Japan1GothicBBBがそれですが、これらのフォントを持っていない場合はOS付属のフォントMS明朝などに置き換えて表示されます（普通は持っていないと思います）。英文フォントはいろいろありますが、和文フォントに関してはあまり選択肢はありません。?postscriptFontsを参照して下さい。

# ファイルへの出力

1. **最初からデバイスを指定して図を描いてデバイスを終了する方法**と、2. **図を描いた後でdev.copy()でファイルに出力する方法**があります。どっちでも良いのですが、微妙に主力ファイルが異なる場合もあるので注意して下さい。

```
### 最初からpng()を指定
png(file = “test1.png”)
 # 背景を透明にしたい場合はbg = “transparent”
plot(1:10)
dev.off()

### 後でdev.copy()する
plot(1:10)
dev.copy(device = png, file = “test2.png”)
 # 最初から背景が透明になっている
dev.off()


### dev.copy()の亜種？、便利
plot(1:10)
dev.copy2pdf(file = “test3.pdf”, family = “Palatino”)
 # これはdev.off()しなくて良い
```

# 終わりに

Rでは通常の`graphics`と`grDevice`パッケージで大体の図を描くことができます。今回は紹介してませんが**地形図**（`contour()`）も描けます。さらにパッケージ`maptools`ではシェープファイルから**地図**を、`lattice`を使えば**多変量データを一括**で作図することもできます。他にも作図用パッケージはたくさんあるので**Rjpwiki**などで模索してみて下さい。