R 初心者ゼミ 2009in 森林総合研究所

飯島勇人[*†‡§]

2009 年 11 月 25 日

目次

[*] 山梨県森林総合研究所森林研究部森林保護科技師

[†] 連絡先: hayato.iijima@gmail.com または iijima-akks@pref.yamanashi.lg.jp

[‡] 本ゼミのサポートページ: `http://www7.atwiki.jp/hayatoiijima/pages/34.html`

[§] この文章は LaTeX で作成しました。わーどなどではけっしてありません。

1 本講義の目的

本講義では、R を使ってみなさんがある程度のデータ解析をできるようになることを目的としています。R は近年急速に利用者が増えているデータ解析ソフトです。その特徴としては、以下のようなものがあげられます。

- あらゆる OS で使用できる。
- 無料。
- 世界中の統計学者が開発に携わっており、開発速度が速く、信頼度も高い。
- データの整形・加工、作図、統計処理が全て行える。

ということで、R が使えるようになると、データ打ち込みと作図はゑくせる、統計は SPSS、JMP..... いやいや、こっちの研究室になったら作図は Delta Graph、統計は世にも恐ろしいゑくせる統計..... といったように、やることによってソフトをいちいち乗り換えたりする必要がなくなります。

本講義では、R を用いた

- データ操作（2 章）
- 作図（3 章）
- 統計解析方法（4 章）

ということを解説します。ですが、今回は時間の制約もあるので、全部は紹介しません。

1.1 使うデータ

- 樹木の生残、成長に養分処理と明るさが与える影響を調べたデータ
- syori は養分処理
- FL は調査プロットの番号
- ID は個体の番号
- omosa は個体の総重量、tijou は個体の地上部重量、ne は個体の根の重量、miki は個体の幹の重量、ha は個体の葉の重量
- light は個体の光環境
- surv は個体の生残（1 が生残、0 が死亡）

```
   syori   FL   ID omosa tijou   ne miki   ha     light surv
1 futsuu C510  C-1  2.12  1.99 0.13 0.91 1.08 0.2647800    1
2 futsuu C510 C-10  3.20  2.54 0.66 0.89 1.65 0.1329878    0
......
```

2 データ操作

この章では R にデータを読み込んで、整形・加工する方法を学びます。R での作業の全ての基礎となります。

2.1 R にデータを読み込ませる方法

2.1.1 データファイルの作り方

- ゑくせるなどにデータを打ち込む。
- 各列に個体番号、環境条件などを入れる。
- 1 行目にはデータのラベルを入力する。
- 日本語、特殊文字（% や○×など）は**絶対に使わない**（最近の R は日本語も使えるみたいですが)。入力していいのは、数字、半角英文字、空欄のみ（生残なら 0 と 1 で区別する)。
- 基本的に空のセルを作らない（欠損値がある場合は空欄でもよいが、NA を入力しておいた方がいい)。
- 違った形式（並べ方が異なる）のデータを同一ファイル内に混在させない。
- 入力が終わったら、保存する際に「csv（カンマ区切り）*.csv」を選択し、csv ファイルとして保存する。

2.1.2 用意したファイルを置く場所

R へファイルを読み込ませるには、ゑくせるなどで作った csv ファイルをどこか好きな場所において、「ファイル」→「ディレクトリの変更」で、データファイルがあるフォルダを選択します。

または、ファイルを作らず、ゑくせるなどで使う部分だけコピーし、以下で述べるコマンドで貼り付けるという技もあります（あまりおすすめしませんが)。

2.2 R でのデータの扱い方

R でデータを操作する基本単位は、**オブジェクト**です。オブジェクトとは、「適当な文字」です。自分で適当につけた文字にデータをくっつけて、データをいじります。通常、R を起動する、あるいは命令を出していない状態では、R には、

起動時の R

```
>
```

と表示されているはずです。これは、R が「命令を出してください（準備オッケーです)」と言っていることを示します。では、これが出ている状態で、オブジェクトがどういうものか、試してみましょう。

データの読み込み例

```
> d <- read.csv("data.csv") #read.csv("ファイルネーム") という書き方
```

コピペ派の人

```
> d <- read.table("clipboard")
```

- 以上のように打ち込むことで、d **という文字**に data.csv のデータを**くっつける**ことができます。
- つまり、d が**オブジェクト**です。
- 本当に読み込めたのか、

  `> d`

  で確認します。さっきのデータが表示されるはずです。

- オブジェクトは、数字のみ、あるいは数字が頭に来る文字以外なら何でも構いません。例えば、
  `> test <- read.csv("data.csv")`
  としてみてると、test に data.csv のデータが付与されています。
- 最初の方のデータだけ見たいときは、`> head(d,3)` と打ってみて下さい。これは、d の最初の 3 行だけ表示してくださいというコマンドです。データを読み込んだときは、`> head()` でデータを確認する癖をつけて下さい。

2.2.1 作業・データの保存

作業内容（読み込んだデータや生成したオブジェクトなど）やコマンドの入力履歴をそのまま保存しておくことが出来ます。作業内容は、「ファイル」→「作業スペースの保存」で.RData という拡張子のファイルとして保存されます。.RData には、読み込んだデータそのものが含まれています（Linux とは異なるようです）。.Rhistory には、入力したコマンドの履歴だけが含まれています。

2.2.2 こまごましたこと

- キーボードの上矢印（↑）を押すと一つ前に打ったコマンドが、下矢印（↓）を打つと先のコマンドが表示されます。
- コマンドを入力する際、いきなり R に文字を打ち込むのではなく、**必ずメモ帳などでテキストの形でコマンドを下書き**して、それをコピペすることで行ってください。テキストを残しておくと間違っていた際の修正や、あとから自分が何をしたのか振り返るのに便利です。
  また、#という記号の後は R の中で無視されます（コメントアウト）。そのため、コード中に日本語で何をする操作なのか書き込んでコメントアウトしておくと、やはり後で振り返るのに便利です。
- 1:10 というように数字の間に:をはさむと、左の数字から右の数字まで 1 ずつ変化する数列を生成できます（等差数列）。

2.3 R におけるデータの種類

単一の値の種類

名称	R での表記	変更方法	例
空値	NA		NA
論理値	logical	as.logical()	TRUE
整数	numeric	as.numeric()	5.3
文字列	character	as.character()	"Tekito"
要因	factor	as.factor()	

NA はしばしば他の関数を無効化してしまうので注意が必要です（後述）。また、要因は外見は文字列とほぼ同じですが、要因は明確なカテゴリーとして扱われます。外見が数字でも文字列として扱われているケースがあるので注意が必要です。

値の集まりの構造

名称	R での表記	変更方法
ベクトル	`c`	c()
行列	`matrix`	matrix() または as.matrix()
データフレーム	`data.frame`	data.frame() または as.data.frame()
リスト	`list`	list() または as.list()

ベクトル 最も基本の単位。一つながりのデータ。あらゆるデータを含めるが、**同一ベクトル内で異なった形式の値を混在させることはできない**。むりくり異なった形式の値を混在させると、一番ランクの低い形式に合わされる。例えば、

```
> test <- 1:4
> test[1] + test[2] #test の 1 番目と 2 番目の要素を足しなさい
[1] 3
> test2 <- c(1, 2, "Tekito", TRUE)
> test2
[1] "1"      "2"      "Tekito" "TRUE"
> test2[1] + test2[2] #test2 の 1 番目と 2 番目の要素を足しなさい
Error in test2[1] + test2[2] : non-numeric argument to binary operator
```

となり、文字列が入っている場合は数字も文字列扱いになってしまう。

行列 ベクトルがいくつも集まったもの。numeric 以外は含めない。

データフレーム ベクトルの集まり。R にデータを読ませるときは大概この形式で、実用上最も多用する。行や列のラベルあるいは番号で行や列を取り出せる。行数の違う列が混在することはできない。行列でいろんなデータを扱えるようにしたもの、と考えられる。

リスト 行数の違うデータだろうがなんだろうが無理やり階層化して含めてしまう（最終奥義）。データの長さや形式がごちゃごちゃなものをとりあえず一つのオブジェクトにまとめたいときに有用。

とりあえずゑくせるなどに打ち込んだデータはデータフレームとして扱うことがほとんどで、これが基本形です。

2.4 データフレームの一部（ベクトル）へのアクセス

データフレームをいじるためによく使う関数を紹介します。なお、これらの関数は覚える必要はなく、必要なときにこのレジュメを見る、と言った程度で十分です。

以下では、データフレームのうち、特定の列（縦方向に並んでいるデータ）にアクセスする方法を示します。データフレームのうち特定の列、あるいは行はベクトルです。

- `$`または `[ , ]`：データフレームのうち、特定の列や行を指定する（とっても大事！）。
- `mean()`: 列の平均値

- median(): 列の中央値
- sd(): 列の標準偏差
- sum(): 列の値の合計
- min(): 列の最小値
- max(): 列の最大値
- length(): 列のデータ数
- table(): 列の各カテゴリーに含まれるデータ数を示す。2 カテゴリーに拡張した場合は xtabs()
- apply(): 全ての列または行に上記の関数を適用させる
- tapply(): ある列に関し、別の列の**カテゴリーごと**に、上記の関数の計算をさせる

2.4.1 データフレームの一部の行または列を取り出す

```
> d <- read.csv("data.csv")
> head(d, 2)
> syori
Error: object "syori" not found
> d$syori #今度はデータが表示される。
> d[, 1] #syori は 1 列目なので、これでも同じ
> d[2, 1] #2 列 1 行の部分の値が取り出される。
```

データフレーム名を省略することも出来ます。

```
> attach(d) #"d"を R に読み込ませる。
> syori #こんどはラベル名だけでデータが表示される。
> detach(d) #attach を解除
```

ただし、attach() してしまうと何のデータセットが attach() されているか忘れやすく、またスクリプトを再利用する際にも混乱を招くので、利用はあまりおすすめしません。

あるラベルだけ捨てたいとき。

```
> d2 <- d[, -3] #3 列目を捨てなさいという命令。
> head(d2, 2) #tijou がなくなっている。
```

データフレームに、新しい列を加える。

```
> d$Nonphoto <- d$ne + d$miki #幹重量と根の重量の合計値を、Nonphoto というラベルで d に付与。
> head(d, 2) #Nonphoto という新しい列ができている。
```

2.4.2 その他の関数

ある列の平均値。

```
> mean(d$tijou)
```

ある列の中央値。

```
> median(d$tijou)
```

ある列の標準偏差。

```
> sd(d$tijou)
```

ある列の値の合計値。

```
> sum(d$tijou)
```

ある列の最小値。

```
> min(d$tijou)
```

ある列の最大値。

```
> max(d$tijou)
```

ある列のデータ数。

```
> length(d$tijou)
```

ある列のカテゴリーに含まれるデータ数。

```
> table(d$syori)
> xtabs(~ syori + ID, d) #2 カテゴリーに拡張
```

行または列ごとに関数を適用させる

#apply(**対象のデータフレーム，各行なら** 1**、各列なら** 2**，関数**)

```
> apply(d[, -(1:3)], 2, max) #各列の最大値
```

カテゴリーデータごとに平均値やデータ数が見たい場合。

```
> tapply(d$tijou, d$syori, mean) #syori ごとの tijou の平均値
> tapply(d$tijou, d$syori, length) #syori ごとの tijou のデータ数
```

2.5 データフレーム全体へのアクセス

以下では、データフレーム同士、あるいはデータフレーム全体に適用できる関数を紹介します。

- head(): データフレームの先頭数行を表示
- subset(): 条件に従うデータフレームのみを選ぶ
- split(): いくつかのカテゴリーにデータフレームを分割
- cbind(): 2 つのデータフレームを、横方向で結合
- rbind(): 2 つのデータフレームを、縦方向で結合
- summary(): データフレームの各列の、基礎統計量を表示
- cor(): データフレームの列間の相関係数を表示する。デフォルトでは pearson の積率相関係数だが、引数に kendall や spearman を取ることで、これらの順位相関係数でも出せる。
- merge(): 2 つのデータフレームを、両データフレームに共通な列に合わせる形で、横方向で結合
- rownames(): データフレームの行名を見る。列名は colnames()。

データフレームの最初だけ見る。

```
> d <- read.csv("data.csv")
```

```
> head(d,2) #"d"の上から 2 行分（ラベル除く）が表示される。
```

ある条件に従うデータだけ見るとき（tijou が 3.0 より大きいデータだけ）。

```
> head(d, 10) #全データのうち、上から 10 行分のみが表示される。
> d2 <- subset(d, tijou>3.0)
> head(d2, 10) #tijou を見ると、3.0 より大きい値しかない。
#条件が二つある場合は&で結合
> d2 <- subset(d, tijou>=3.0&tijou<5.0) #&を使うと条件をつなげられる

#カテゴリーの場合も同様に（syori が futsuu のデータのみ）
> d2 <- subset(d, syori=="futsuu")
> d2 #futsuu のデータしか入っていない
#futsuu だけを除く
> d2 <- subset(d, syori!="futsuu")
> d2 #futsuu だけが抜けている
```

あるカテゴリーごとにデータフレームを分割

```
> d3 <- split(d, d$syori)
> d3 #d3 に syori ごとのデータフレームが入っている
> d3[[1]] #1 カテゴリー目のデータのみが取り出せる
> d3$futsuu #これでも同じ。カテゴリー名を使用
```

2 つのデータフレームの結合（横方向）

```
> d3$futsuu #futsuu だけのデータフレーム
> d4 <- cbind(d3$futsuu, d3$futsuu)
```

2 つのデータフレームの結合（縦方向）

```
> d3$futsuu #futsuu だけのデータフレーム
> d3$zeitaku #zeitaku だけのデータフレーム
> d5 <- rbind(d3$futsuu, d3$zeitaku)
```

各列の基礎的な統計量の表示。

```
> summary(d)
```

列間の相関係数の算出。

```
> d0 <- d[ , -c(1:3)]
> cor(d0) #pearson の積率相関係数
> cor(d0, method="kendall") #kendall の順位相関係数の場合

> colnames(d) #列名が表示される
> month.name[1:10] #R で用意されている文字列
> colnames(d) <- month.name[1:10] #列名を変えることもできる
```

ちなみに、ここで挙げたすべてのコマンドに関しては、R に読み込まれた d というオブジェクトを操作しているだけなので、元のファイル data.csv は一切改変されていません。失敗を恐れずいろいろ試して見て下さい。

逆に、R でいじったオブジェクトを csv ファイルとして出力したいときは、`write.csv()` を使います。例えば d というオブジェクトを csv ファイルとして出力したければ、

```
> write.csv(d, file="つけたいファイル名.csv")
```

とすると、ホームディレクトリに csv ファイルができます。

2.5.1 NA（欠損値）の扱い方

空欄（欠損値）のことを、R では **NA** と表記します。こいつが少々曲者です。NA を含むデータフレーム（あるいはベクトル）に上記の関数を適用しても、**ほとんどは NA が返ってきます**（つまり実行できない！）。

NA に適用できる関数としては、以下のものがあります。

- `na.omit()`: NA を含む行を削除する。
- `is.na()`: NA であれば TRUE、そうでなければ FALSE を返す。
- `ifelse()`: NA 用の関数ではないが、NA を処理するのに使える条件分岐関数。

```
> d2 <- read.csv("data0.csv")
> summary(d2) #surv の列には NA が 2 つあることがわかる
> d2$surv <- ifelse(is.na(d2$surv), 0, d2$surv)
#ifelse(条件，TRUE の場合の行動，FALSE の場合の行動)。今回は NA を 0 に置換。
> summary(d2) #NA がなくなっている
```

このほかにも、R には様々なデータ操作用関数が用意されています。

3 作図

とりあえず**データ解析において最も重要といってもいいのが作図**です。データが何を語っているのか？を作図を通して把握しましょう。作図を経ないで行った統計解析は、しばしば失敗、ぬか喜びの元になります。

R でのグラフの作り方は、**高水準作図** → **低水準作図**という順番に行います。

3.1 高水準作図

高水準作図は、**グラフの種類を決定する**作図です。グラフの種類としては、

- 散布図
- ヒストグラム
- 箱ひげ図
- 対散布図
- 棒グラフ
- 円グラフ

などがあります。

ヒストグラム hist()

```
> hist(d$omosa) #omosa のヒストグラム
```

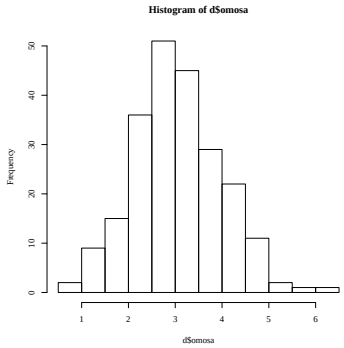

図 1　ヒストグラム

箱ひげ図 boxplot()

```
> boxplot(omosa ~ syori, d) #syori ごとの omosa
```

一部例外を除き R では（y 軸の値 ~ x 軸の値，データフレーム
\index{でーたふれーむ@データフレーム}名）という書き方
をします。覚えておいて下さい。

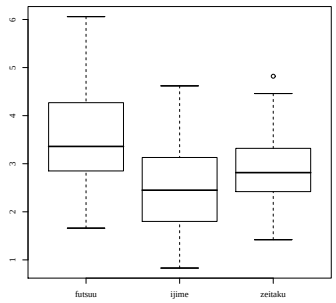

図 2　箱ひげ図

散布図 plot()

```
> plot(ha ~ omosa, d)
#カテゴリーごとにシンボルを変えて描画したい場合、
> plot(ha ~ omosa, pch=as.numeric(syori)-1, d)
#さらに色まで変えちゃおう
> plot(ha ~ omosa, pch=as.numeric(syori)-1,
+       col=as.numeric(syori), d)

#カテゴリーごとに独立して図を描きたいなら
> library(lattice)
> xyplot(miki ~ omosa | syori, d)
```

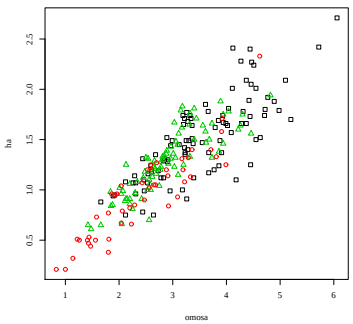

図 3　散布図

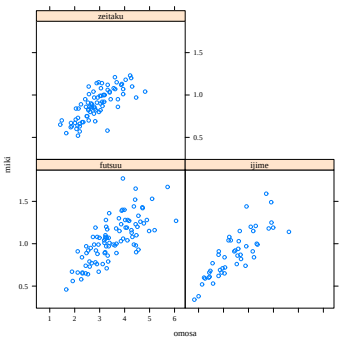

図 4　散布図（カテゴリー別）

対散布図 pairs()

```
> pairs(d)
#散布図と同様、カテゴリーごとの描画もできる
> pairs(d, pch=as.numeric(d$syori)-1, col=as.numeric(d$syori))
```

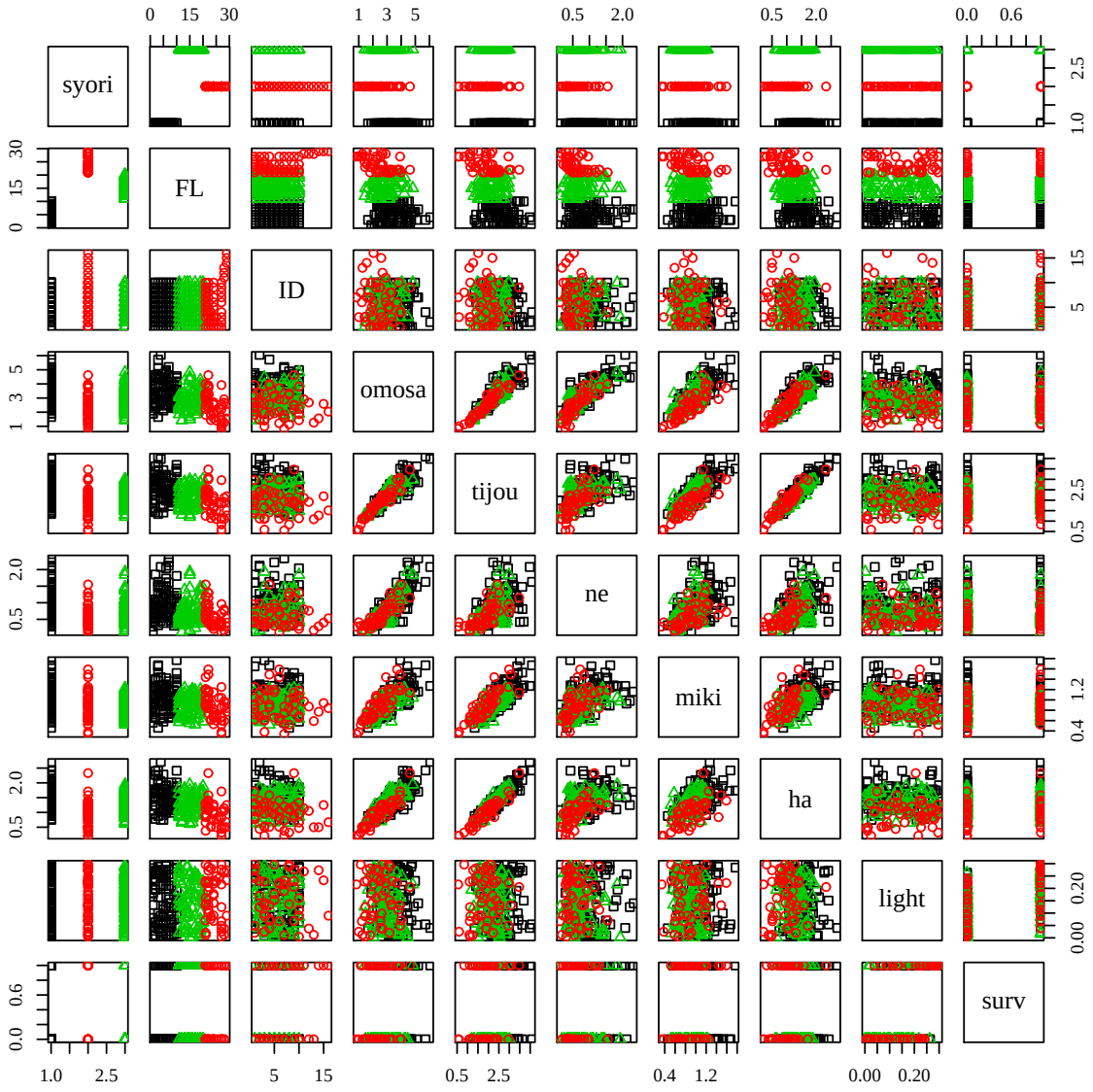

図 5　対散布図

棒グラフ barplot()

```
> barplot(tapply(d$tijou, d$syori, mean))
```

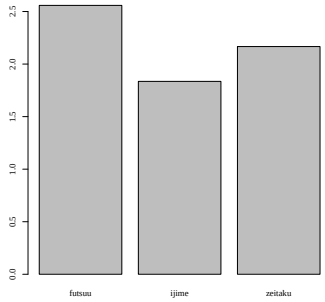

図 6　棒グラフ

円グラフ pie()

```
> pie(table(d$syori))
```

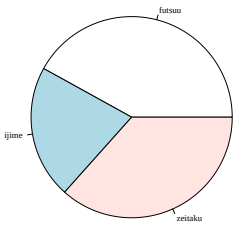

図 7　円グラフ

ヒストグラムはある一つのデータの分布を見るとき、散布図は 2 つの変量間の関係を見るとき（拡張版が対散布図)、カテゴリー間での差を見るときは箱ひげ図が基本です。まずデータを取ったら、これらのいずれかの形式でデータの傾向を見る癖をつけましょう。棒グラフや円グラフは、状況に応じて使ってください。

高水準作図で勝手に出力されるタイトルや文字、目盛幅などが思い通りにいかないことはよくあります。そのような時、以下の低水準作図が役に立ちます。

3.2　低水準作図

低水準作図は、上記のように**できているグラフの飾り付けをする**作図です。具体的には、

- タイトルやラベル
- 軸の値の幅や目盛
- 描くシンボルの形や色
- 回帰直線
- 凡例
- フォントの大きさ

などの調整ができます。

3.2.1 高水準作図の中で実行する低水準作図

タイトルやラベル、軸の値の幅や目盛、描くシンボルの形や色、は高水準作図の命令の中で実行する必要があります。

散布図中における命令の例を示します。ほとんどは他の高水準作図でも応用できます。

タイトル・軸ラベル

```
> plot(..., main="タイトルにしたい文字", xlab="x 軸のラベル", ylab="y 軸のラベル", ...)
```

実際例は、

```
> plot(miki ~ omosa, main="Test", xlab="Total mass (g)",
+                    ylab="Stem mass (g)", d)
```

main がタイトル、xlab が x 軸、ylab が y 軸のラベルであり、それぞれ”” の中に文字を入力する。

軸の値の範囲・目盛

```
> plot(..., xlim=c(x 軸の最小値, x 軸の最大値), ylim=c(y 軸の最小値, y 軸の最大値), ...)
```

実際例は、

```
> plot(miki ~ omosa, xlim=c(0, 7), ylim=c(0, 2), d)
```

xlim、ylim でそれぞれ x 軸、y 軸の値の最大値と最小値を調整します。また、これだけでは飽き足らず、自分の好きなところに目盛を打ちたい場合、

```
> plot(..., xaxt="n", yaxt="n", ...) #xaxt、yaxt="n"で x 軸、y 軸の目盛を描かない
> axis(1, at=c(自分で打ちたい目盛の値)) #x 軸の描画
> axis(2, at=c(自分で打ちたい目盛の値)) #y 軸の描画
```

実際例は、

```
> plot(miki ~ omosa, xlim=c(0, 7), ylim=c(0, 2), xaxt="n", yaxt="n", d)
> axis(1, at=c(0, 2, 4, 6))
> axis(2, at=c(0, 1, 2))
```

で、自分の好きなところに目盛を打ちます。axis(の直後の数字は、1 が図の下（つまり x 軸）、2 が図の左（つまり y 軸）に軸を描くことを意味します。ちなみに 3 で図の上、4 で図の右に軸を描くこともできます。

シンボルの形・色

```
> plot(x,y, pch=*) #pch=*のところの数字を変えると、いろいろなシンボルが出てくる
> plot(x,y, col=*) #col=*のところに数字や色の名前（"black"、"blue"など）を入れると、色が変わる
```

ちなみに、Hmisc パッケージをダウンロードしておいて、

```
> library(Hmisc)
> show.pch() #使える記号と数値の対応表が、
> colors() #使える色の一覧が、
> show.col() #数字で指定できる色と数字の対応表が表示されます。
```

実際例は、

```
> plot(miki ~ omosa, pch=1, d)
> plot(miki ~ omosa, col="red", d)
```

3.2.2 低水準作図単独で実行

一方**回帰直線、凡例、フォントの大きさ特殊フォント**は独立した低水準作図で描く必要があります。

回帰直線

ここはいきなり実例で行きます。

```
> plot(ha ~ omosa, d) #「葉重-個体重」の関係
> test <- lm(ha ~ omosa, d) #回帰直線の切片と傾きを調べる
> co <- test$coefficients #切片と傾きを co に付与
> curve(co[1] + co[2] * x, add=TRUE,
+ from=min(d$omosa), to=max(d$omosa)) #回帰直線を描画
```

curve() は線を描く命令で、x 軸の値として x という文字を入れることが決まっています。co[1] + co[2] * x は予測値そのものですね。add=TRUE というのは、前の図に重ねて線を引くということを宣言するものです。で、x の値の範囲を from と to で決めているわけです。

凡例の書き方

```
> legend("場所を指定"または x と y の座標，凡例に描きたい文字やシンボル)

実際例は、
> plot(miki ~ omosa, d)
> legend("topleft", pch=1, col="black", "Stem mass (g)")
#場所の指定方法: "top", "topleft", "bottomright"など
#x と y 座標で指定するなら
> legend(0.6, 1.8, pch=1, col="black", "Stem mass (g)")
```

フォントの大きさ

```
> par(ps=**)
# **の数字を変えればよい。なお、これは作図前に行う。
```

ラベルで特殊・飾り文字

```
#字体の変更（例えばイタリック）
> expression(italic("ここに斜体にしたい文字"))

#数式の記述。
> expression(paste(mu,"mol ",m^-2,s^-1))
> expression(paste(CO[2]," concentration (%)"))
```

と書けば、それぞれ μmol $m^{-2}s^{-1}$、CO_2 concentration (%) と表示されます。実際には xlab=expression(... といった形で使います。記述部分で" "で囲まれた部分が通常の文字列部分、囲まれていない部分が**数式処理をしている部分**です。paste はこれらを結合する意味を持ちます。

3.2.3 図の保存の仕方

- Windows: できた図は、図の上で右クリックし、**ビットマップにコピー**を選び、コピーします。そして、ペイントなどに貼り付けて、ファイルとして保存します。

- Windows: 同じ要領で、**メタファイルにコピー**というものがあります。これを実行し、Word などを立ち上げると、図を貼り付けることができます。メタファイルはあまり一般的な形式ではありませんが、拡大縮小に伴う崩れがないのが特徴です（図として保存してもかまいません）。
- その他（Windows でも可）: とりあえず作図します。そして、

```
> dev.copy(ファイル形式, file="好きなファイルネーム.ファイル形式")
> dev.off() #上のコマンドの後すぐに行うこと！
```

とすることで、作業ディレクトリに図ができています。ファイルの形式としては、png、gif、jpg、pdf、eps などいろいろなものが使えます。汎用性の高さを考えると、とりあえず pdf にしておくとよいでしょう。

3.3 連続図

よく、

1. x 軸は全部同じだけど、y 軸だけが異なる図が複数有り、それをきれいに並べて表示したい
2. x 軸も y 軸も同じだけど、（何でもいいんですけど）年度とか種とかが違う複数の図を書きたい

というケースがあると思います。ゑくせるでも頑張ればできます。ですが、ゑくせるで作業したことがある人は分かると思いますが、2 つ以上の図をきれいに並べるためには、マウスで微妙な大きさをあわせなければならないので、とても時間がかかります。やってられません。R なら、フォーマットを整えた図を作るのも簡単です。

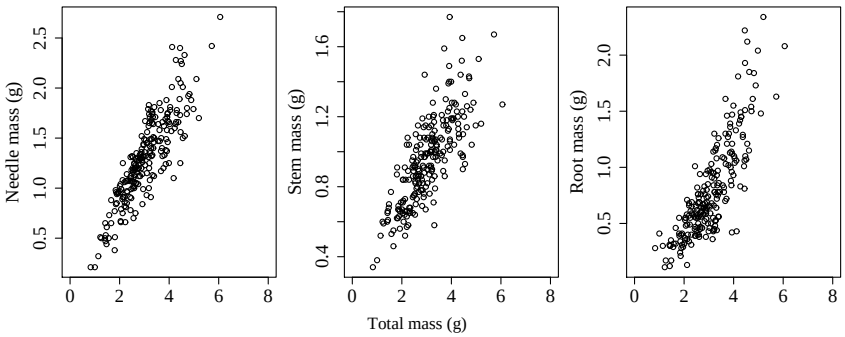

図 8　個別の散布図

3.3.1 図の数とレイアウト

まず、図を縦に何個、横に何個（何 × 何）で配置するかを決めましょう。mfrow=c(,) を使います。実際に使うときは、

```
> par(mfrow=c(縦に並べる図の数, 横に並べる図の数))
```

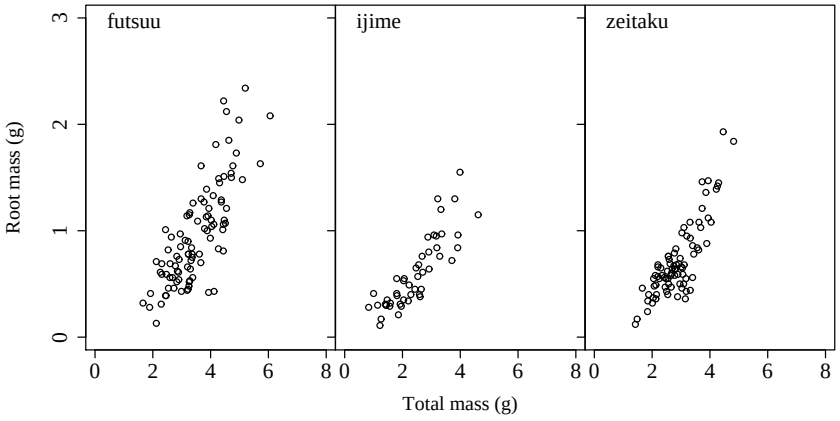

図 9　個別の散布図 2

というようにします。

3.3.2　図の間の間隔の調整

複数図を作る場合、図の間の間隔を調整する必要があります。

1. x 軸は全部同じだけど、y 軸だけが異なる図が複数有り、それをきれいに並べて表示したい → **縦方向は隙間なし、横方向はラベルが入るように少しあける**
2. x 軸も y 軸も同じだけど、年度とか種とかが違う複数の図を書きたい → **縦方向も横方向も隙間なし**(隙間を入れることもできますが)。

図の間の間隔を調整するには、`par(mar=c( , , , ))` を使います。`mar()` には、左から順に、図の下、左、上、右に、どれほどの隙間をとるか設定します。

3.3.3　複数の図に対するラベル

さらに、複数の図を描いた場合、x 軸や y 軸のラベルを、それぞれの図につけるのではなく、複数の図全体に対してラベルをつけたいことがあると思います。このためには、

- 複数の図の外に、x 軸と y 軸のラベル用のスペースを作る。
- 複数の図の x 軸と y 軸を描く。

というようにします。なぜこんなことをする必要があるかというと、例えば、描く図の数が 2 つの場合、各図の x 軸にラベルを設定すると、複数の図の真ん中にラベルを描くことができません。

まず、複数の図全体に関して、図の外の余白を設定します。これには、`oma()` を使います。`mar()` との違いは、

- `mar()`: 各図に関して、図の外の余白を設定する。
- `oma()`: **複数の図群に関して**、図群の外の余白を設定する。

ということです。

次に、複数の図を描いた後で、x 軸と y 軸を描きます。

```
> mtext(1, line=4, outer=T, text="x 軸のラベル")
> mtext(2, line=4, outer=T, text="y 軸のラベル")
```

mtext() の最初の数字は 1 が図の下（x 軸）、2 が図の左（y 軸）にラベルを描くことを、line の数字は、図からどれだけラベルを離して描くか、outer は、図の外にラベルを描くことを認めるか（T は TRUE）を示しています。

3.3.4 その他

複数の図を並べるときに、x 軸や y 軸のうち、図に共通な部分は、軸の値の間隔もそろえるのが一般的です。xlim=c(,) や ylim=c(,) を使ってそろえておきましょう。

3.3.5 y 軸の変数だけが図間で異なる複数の図

全データに関し、個体重に対して左から葉重、幹重、根重の関係の図を描いて見ましょう。

```
> d <- read.csv("data.csv")
> par(mfrow=c(1, 3))
> par(mar=c(0, 5, 0, 0), oma=c(5, 0, 1, 1), ps=20)
> summary(d) #値の範囲を決めるため
> plot(ha ~ omosa, xlim=c(0, 8), xlab="", ylab="Needle mass (g)", d)
> plot(miki ~ omosa, xlim=c(0, 8), xlab="", ylab="Stem mass (g)", d)
> plot(ne ~ omosa, xlim=c(0, 8), xlab="", ylab="Root mass (g)", d)
> mtext(1, line=3, outer=T, text="Total mass (g)", cex=0.6)
```

ちなみに、mtext は設定してあるフォントサイズよりも大きめになっているので、cex=0.6 することでサイズをあわせています（cex は拡大率を指定する）。。

3.3.6 x 軸も y 軸も同じ変数である複数の図

syori ごとに、「個体重–根重」を描いてみます。

```
> d <- read.csv("data.csv")
> test <- split(d, d$syori)
> par(mfrow=c(1, 3), mar=c(0, 0, 0, 0), oma=c(6, 6, 1, 1), ps=20)
> summary(d) #値の範囲を決めるため
> for (i in 1:3) {
+       plot(ne ~ omosa, xlim=c(0, 8), ylim=c(0, 3),
+       xlab="", ylab="", yaxt="n", test[[i]])
+       if (i == 1) { axis(2, at=0:3) }
+       else { axis(2, at=0:3, label=FALSE) }
+       legend("topleft", legend=unique(test[[i]]$syori), bty="n")
+       }
```

```
> mtext(1, line=3, outer=T, text="Total mass (g)", cex=0.6)
> mtext(2, line=4, outer=T, text="Root mass (g)", cex=0.6)
```

4 R で統計解析

4.1 データ解析と統計的モデリング

我々はなぜデータ解析をするのでしょうか？それは、自分と他人に現象をわかりやすくするためです。たとえば森林を見て、「ここではこんなことが起きていそうだ」と感じても、言葉だけでそれを説明するのは難しいですし、ましてや相手を納得させるのは不可能でしょう。そこで我々がよく行うのは、現象を客観性が高い「数値」で取り出し、その数値の塊を加工して示すことです。このため、統計解析は研究の重要な部分を占めているといえます。

統計解析と一口に言っても、さまざまな種類のものがあります。ヒストグラムなどは記述統計と呼ばれるもので、このような数値の集約も現象を解釈する上で有用です。一方、今回扱うのは統計的モデリングという方法です。ある現象を簡単に説明できるモデルを構築することで、見ている現象の因果関係や、定量的な関係を明らかにすることができます。

4.1.1 本日の題材

統計的モデリングを説明するために、今回は「植物種 A の生残に養分処理と光環境が影響しているか？」という問題を検討します。養分処理は 3 処理（futsuu、ijime、zeitaku）、光環境は相対照度の連続値です。

```
> d <- read.csv("surv.csv")
> head(d, 2) #データの上*行を頭出しします
> plot(Surv ~ Light, col=as.numeric(Nutrient), d)
```

データでは生残が 1、死亡が 0 と入力してあります。図を見ると、明るい環境では 1 が多くなり、明るいほど生残率は高いように見えます。一方、養分処理は明確な違いを与えていないように見えます。これを、統計的モデリングではっきりさせます。

4.1.2 統計モデルの部品

統計的モデリングは、**決定論的モデル（Deterministic model）**と**確率論的モデル（Stochastic model）**の 2 つに分けることができます。統計的モデリングを正しく行うためには、この 2 つをきちんと意識することが重要です。

- 決定論的モデル: 説明変数と応答変数の関係性を示したもの。自分がどう現象を捉えているかといってもいい。
- 確率論的モデル: 得られる現象がどのような確率的変動をもって生じるかに関する仮説。

例えば、上記のデータをモデリングするとき、決定論的モデルとしては以下のようなものが考えられます。

決定論的モデルの例

個体の生残 ← 光条件 + 養分条件 #実はこれは正確ではない

これは非常に単純な線形モデルといえます。決定論的モデルは自分がどのように現象を表現するかですから、必ずしも線形である必要はありません。例えば、光合成曲線は二次関数を使っています。

一方、確率論的モデルとしては以下のようなものが考えられます。

確率論的モデルの例

個体の生残 ~ Binomial(p), p は生起確率

?????なんのこっちゃという感じですよね。今回は、個体の生残は二項分布という確率分布に従って発生すると考えています。この、確率論的モデルについて、以下でもう少し詳しく述べます。

4.1.3 確率論的モデル

先ほど、自分がどのように現象を捉えているか表現したのが決定論的モデルだと説明しました。では、決定論的モデルだけで全ての現象を説明できるのでしょうか？

答えは No です。現象を観測し、誤差が全く生じないことはありえません。誤差が生じることで、**決定論的モデルから予測される値と観測値の間にはずれが生じます**。そのため、観測される現象がどのような誤差を持って生じるかを考慮しなくてはなりません。誤差は、

- 観測誤差
- 個体差、注目していない条件など、観測していない要因

といったように、いくらでも生じえます。この、「観測される現象がどのような誤差を持って生じるか」を確率分布で表現したものが、確率論的モデルです。つまり、**観測される現象は決定論的モデルと確率論的モデルの両方が作用して観測される**といえます。

確率論的モデルには、各種の確率分布を用いることができます。身近なものとしては、正規分布やポアソン分布、二項分布などが挙げられます（図 10）。確率分布の形は、**確率分布のパラメータの値によって変化します**。

- 正規分布：平均、分散（2 パラメータ）。−∞〜∞の値を取り得る。
- 二項分布：平均（=生起確率、1 パラメータ）。0〜1 までの値（確率）を取り得る。
- ポアソン分布：平均（1 パラメータ）。0 以上の整数を取り得る。

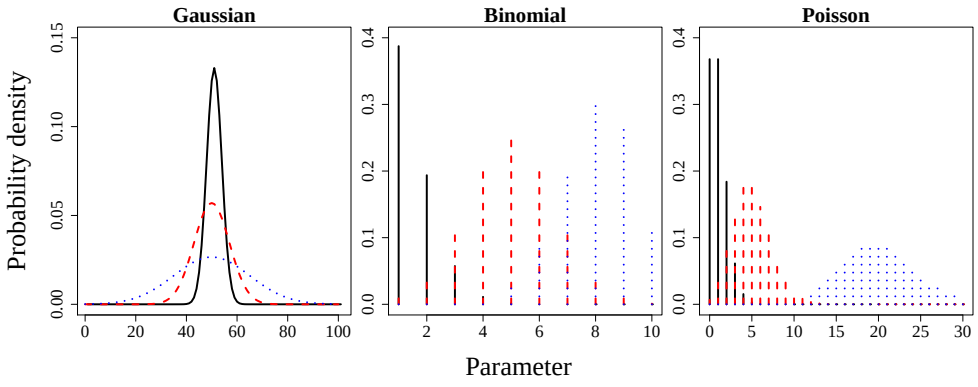

図 10　確率分布とパラメータによる形の違い

図中の色の違いは、正規分布においては分散のみを、二項分布とポアソン分布については平均値を変化させたもの。

このように確率分布の形はパラメータによって変化しますから、観測された現象（データ）に最も適合する形になるときの確率分布のパラメータが、最も可能性の高いパラメータとなります。

4.1.4 決定論と確率論の結合

我々が興味があるのは、決定論的モデルによる予測が観測された現象とどのような関係にあるのか（定量性）ということです。これは、決定論的モデルに組み込んだ要因の係数を推定するという問題として考えることができます。

では、係数の推定はどのような基準で行えばよいのでしょうか？観測される現象を表現する確率分布の形は、そのパラメータによって決定されます。そのため、**観測される現象に関する確率分布のパラメータを、決定論的モデルによって、観測される現象に最も当てはまるように決定**すればいいことになります。決定論的モデルの挙動は、そのモデルに組み込まれた要因の係数や切片によって決まりますから、係数や切片を観測された現象が最もよく再現できるように決定する、と考えることができます。

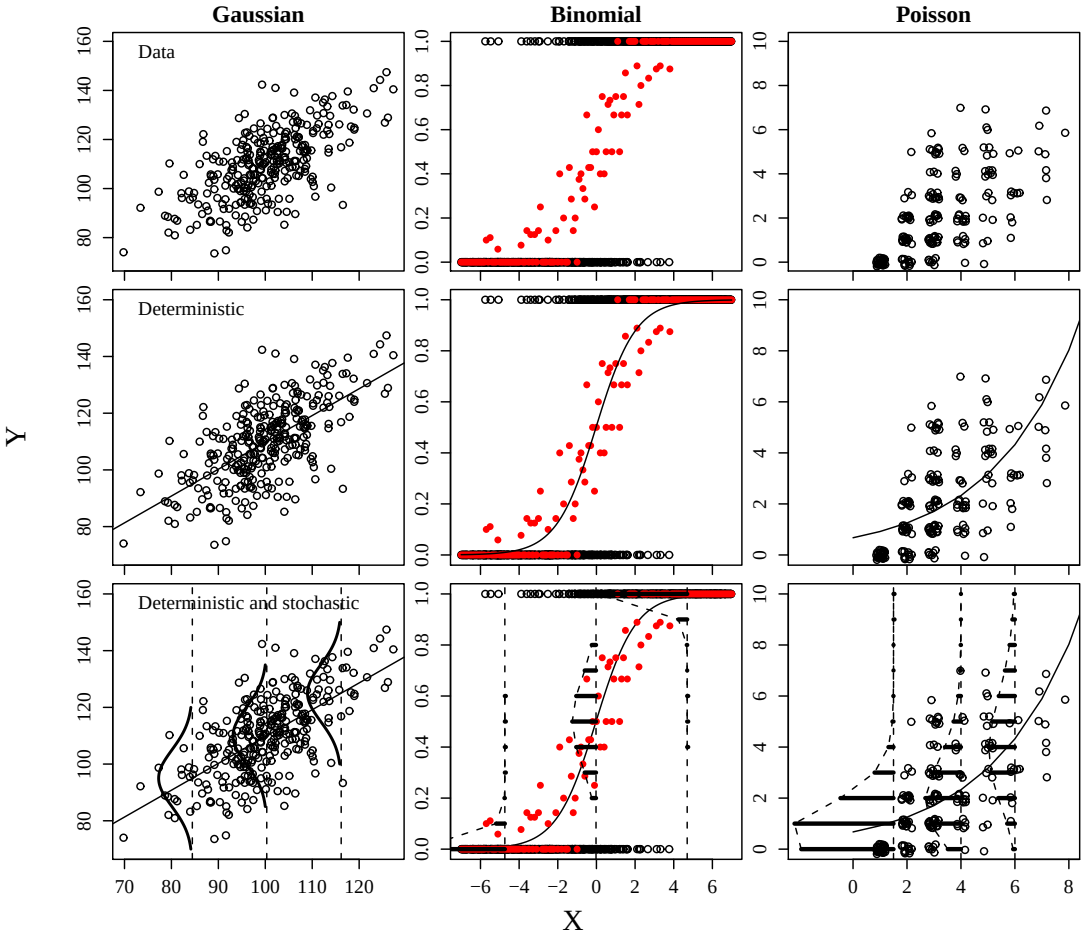

図 11　実データと決定論および確率論的モデルの関係

上段が実データ、中段が決定論的モデルによる予測、下段が確率論的モデルを加えた図。下段では、確率論的モデルによって予測される値の発生 確率が、太線で示されている。二項分布の図では、X 軸の値が 0.1 毎に平均した Y の値を、赤丸で示している。

決定論的モデルでの予測を決定する際、確率論的モデルに従って誤差が発生すると仮定して決定しています。そのため、確率論的モデルを表示させると、決定論的モデルによる予測がどのような基準で決定されているのか見て取ることができます。

以上を踏まえると、統計的モデリングは以下の手順から構築されます。

- 観測される（説明したい）現象が、何らかの確率分布に従って生じると仮定。すなわち、確率論的モデルの決定。
- 考えている要因から観測される現象を説明するモデルを作成。すなわち、決定論的モデルの決定。
- 観測される現象を最もよく表現できるように、確率分布のパラメータ、すなわち決定論的モデルの係数を推定する。

「観測される現象を最もよく表現できる」ということを評価する指標として、尤度（*Likelihood*）を用いることができます。尤度については次節で説明します。

4.1.5 尤度

注！

- ここの文章は粕谷（1997）の最尤法の説明をほぼそのまま使わせてもらっています。これよりエレガントな説明なんてとてもできないので.....

尤度とは、「**あるデータが得られたときに、ある確率分布の元でデータをどれだけ尤もらしく表現できているかを示すもの**」です。なんだかイメージしにくいと思うので、実例を挙げて紹介します。

n 回コイントスをして表か裏かを調べ、r 回表、$(n-r)$ 回裏が出たとしたときに、表が出る確率は以下の二項分布で定義できます。この $L(p|r)$ が、**尤度**です。尤度は、データによく当てはまっているほど**値が大きくなります**。

$$
\begin{aligned}
L(p|r) &= \binom{\mathrm{n}}{r} p^r (1-p)^{(n-r)} \\
&= {}_n\mathrm{C}_r p^r (1-p)^{(n-r)}
\end{aligned}
$$

例えば 3 回コイントスをして 2 回表だったとします。このデータだけから「このコインの表が出る確率は？」と問われたら、ほとんどの人が（表が出た回数）/（全試行回数）で $2/3 = 0.6666...$ と答えるでしょう。「コインの表が出る確率」は、二項分布のパラメータである生起確率 p と同じものです。

では、$p = 2/3$ にすると、本当に尤度は最大になるのでしょうか？図を描いて見ましょう。

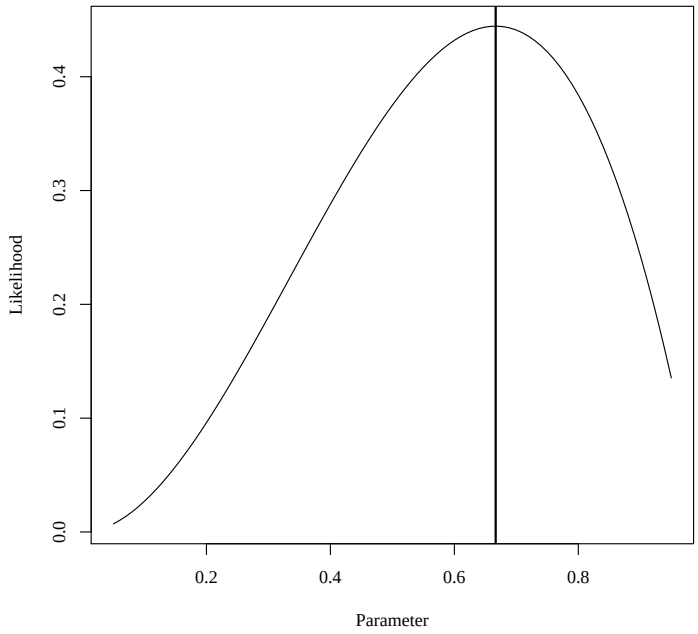

図 12　パラメータを変化させたときの二項分布の尤度

作図コード

```
> Lbinom <- function(p, r, n) {choose(n, r) * p^r * (1 - p)^(n - r)}
> pr <- 5:95/100
#3 回コイントスして 2 回表が出た場合
> plot(Lbinom(pr, 2, 3) ~ pr, xlab="Parameter", ylab="Likelihood", type="l")
> abline(v = 2/3, lwd=2)
```

この図からわかるように、$p = 2/3$ が尤度を最も大きくする値のようですね。では、3 回コイントスして 2 回表が出た場合に、p を 2/3 にすると何故尤度が最も大きくなるのでしょうか？

尤度に関わらず、ある関数の変曲点はその関数を微分することで求めることができます。今回は「3 回コイントスして 2 回表が出た」というデータが得られているので、そのときの尤度を微分して、尤度が最大となる p を求めることができます。

$$\begin{aligned} L(p|r) &= {}_3\mathrm{C}_2 p^2 (1-p)^{(3-2)} \\ &= 3 \times p^2 (1-p) \end{aligned}$$

この尤度を微分すると、

$$\begin{aligned} L(p|r)' &= 6p - 9p^2 = 0 \\ &= p(6 - 9p) = 0 \\ &= p = 2/3 \text{ or } 0 \end{aligned}$$

となり、2/3 が導かれました。この、

- $p = 2/3$ を求める方法が**最尤法**（尤度という基準で確率分布のパラメータをデータに尤も当てはまるように決める）
- $p = 2/3$ が**最尤推定値**。

です。後で出てくる deviance とか AIC も、尤度を基にして算出されます。

実際の推定の際は、p を決定論的モデルで表現することになります。その場合、推定する係数の数だけ偏微分をして求めることになります。また、今日は説明しませんが（以下に解説だけ示します)、計算が簡単になること、最尤推定値は変わらないことから、実際の計算では尤度の対数を取った対数尤度を用いることがほとんどです。

4.1.6 対数尤度

ちょっとの世界に立ち入ったことがある人だと、尤度（*Likelihood*）よりも、対数尤度（*Log Likelihood*）という単語のほうになじみがあるかもしれません。対数尤度とは、**尤度の対数をとったもの**です。実際の線形モデルにおける最尤法は、対数尤度を対象として実行されることがほとんどです。

なぜかというと、使うのは尤度でも対数尤度でもどちらでもいいのですが、計算は対数尤度でないと最尤推定値が求められないことが多いからです。例えば正規分布を例にとって見ましょう。

$y = y_1, y_2, ..., y_{100}$ という 100 個のデータがあったときに、y_1 が得られる確率（尤度）は、

$$\frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{1}{2\sigma^2}(y_1-\mu)^2)$$

と定義できます。同じように y_2、y_{100} なども得られますが、これらのデータ y が得られる確率を最も大きくするためには、（y_1 が得られる確率）×（y_2 が得られる確率）×.....×（y_{100} が得られる確率）を最も大きくする必要があります。この尤度を $L(\mu, \sigma^2|y)$ とすると、

$$\begin{aligned} L(\mu,\sigma^2|y) &= \prod_{i=1}^{100}\frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{1}{2\sigma^2}(y_i-\mu)^2) \\ &= (\frac{1}{\sqrt{2\pi}\sigma})^{100}\exp(-\sum_{i=1}^{100}\frac{(y_i-\mu)^2}{2\sigma^2}) \end{aligned}$$

となり、どうやって微分するんだこれ？という式になってしまいます。そこで対数をとります。対数をとった先ほどの尤度を $\log L(\mu, \sigma^2|y)$ とすると、

$$\begin{aligned} \log L(\mu,\sigma^2|y) &= \log((\frac{1}{\sqrt{2\pi}\sigma})^{100}\exp(-\sum_{i=1}^{100}\frac{(y_i-\mu)^2}{2\sigma^2})) \\ &= -100\times\log(\sqrt{2\pi}\sigma) - \sum_{i=1}^{100}\frac{(y_i-\mu)^2}{2\sigma^2} \end{aligned}$$

となり、結局のところ $(y-\mu)^2/2\sigma^2$ の部分を微分すれば最尤推定値が求まることになります。対数尤度にすると計算が楽になることが多いので、ほとんどの場合対数尤度を用います。

4.2 GLM&モデル選択

4.2.1 GLM がなぜ必要か

GLM とは Generalized Linear Model の略です。Generalized がつくのは、これまでの仮説検定では（ほぼ）正規分布しか扱えなかったのに対し、GLM では複数の確率分布を扱えるようになっているためです。

この、「複数の確率分布を扱える」ということが重要です。なぜなら、現実には**正規分布に従わない**データがよく得られるからです。これまでは変数変換などで、データを正規分布にあわせようとしてきました。しかし、変数変換してもあわない場合はありますし、変数変換すると存在しないはずの交互作用項が有意になったりするため（二元配置以上の ANOVA で苦しめられた人は多いと思います)、GLM が必要になります。

4.2.2 こんなときは GLM だ！

- まあとりあえず基本というか＾＾；）
- ある現象に別の変数が影響を与えているかどうかを検討する

4.2.3 今回のデータを統計モデルで考える

最初に見たように、今回は「植物種 A の生残に養分処理と光環境が影響しているか？」という問題を検討しています。この場合の統計モデルはどのようになるでしょうか？

まず確率論的モデルですが、今回観測された現象（説明したい現象）は生残率になります。このような、事象が起こる、起こらないという現象は二項分布で比較的よく近似できるので、今回は二項分布とします。二項分布のパラメータは、生起確率（事象が起こる確率）p の 1 つのみです。

つぎに決定論的モデルですが、予想していたのは「光および養分条件によって植物の生残は影響を受けるのでは？」ということでした。しかし、単にそれらを線形に結合しただけでは、予測される値は（生起確率 p を予測するのに）0 から 1 の間に収まりません。

そこで、決定論的モデルをちょっと変更します。要因を線形結合したもの（線形予測子）を X とすると、$\exp(X)/(1+\exp(X))$ または $1/(1+\exp(-X))$ とすれば、X は必ず 0 から 1 に収まります。これを図示してみましょう（図 13)。

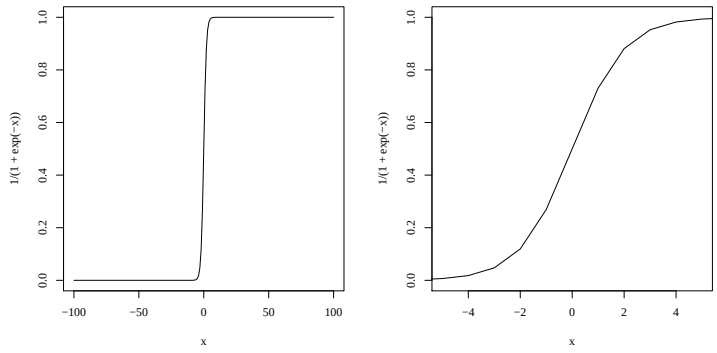

図 13　ロジスティック変換による値の挙動

作図コード

```
> x <- -100:100
> par(mfrow=c(1, 2))
> plot(x, 1/(1 + exp(-x)), type="l")
> plot(x, 1/(1 + exp(-x)), xlim=c(-5, 5), type="l")
```

ちなみに、これを数式で表現すると以下のようになります。

$$\begin{aligned} p &= 1/(1+\exp(-X)) \\ 1/p &= 1+\exp(-X) \\ (1-p)/p &= \exp(-X) \\ \log((1-p)/p) &= -X \\ \log(p/(1-p)) &= X \end{aligned}$$

こう見ると、生起確率 p をロジット変換したものを要因の線形結合（線形予測子）で予測する、と捉えることもできると思います。

以上を踏まえると、

- 確率論的モデル：Binomial$(p), p$ は生起確率
- 決定論的モデル：$p = 1/(1+\exp(-(\text{光環境} + \text{養分条件} + \text{切片})))$（逆リンク関数＋線形予測子）または $\log(p/(1-p)) = \text{光環境} + \text{養分条件} + \text{切片}$

となります。

4.2.4 自力で GLM

決定論的モデルと確率論的モデルが決定されました。では、これらを使って実際にどのように最尤推定を行うのでしょうか？その過程を理解するために、自分の手で計算をして見ましょう。単純化のために、この場合は、光が生残に影響しているか？というモデルについて検討します。

（ゑくせるで作業していただきます。survML.xls を開いてください）

さて、手計算をしていただいてわかったと思いますが、最尤推定値を求めるためにはいろんな切片や傾きの値を試し、（対数）尤度が最大となる値を探索する必要があります。切片や傾きを変化させると、対数尤度は以下のように変化します。

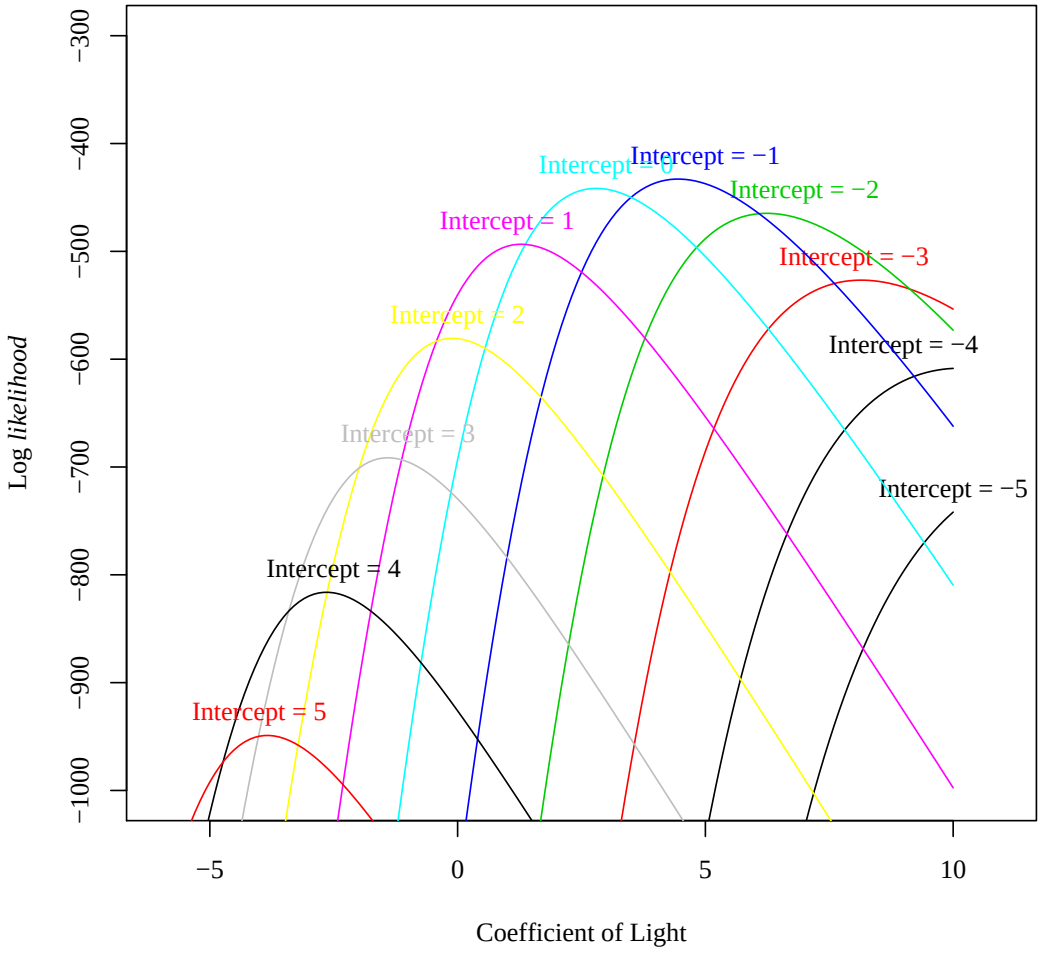

図 14　パラメータを変化させたときの対数尤度の挙動

で、最適な傾きと切片を手作業で探索するのは大変なんで、そこは R の glm() 関数に任せましょう。

4.2.5 GLM で指定しなければならないこと

GLM は統計モデルであり、先ほどのような統計的モデリングの過程を明示してやる必要があります。R での方法は以下の通りです。

```
> result <- glm(y ~ x1 + ..., family = 誤差構造（リンク関数）, データ名）
```

GLM で指定しなくてはならないことは以下の 3 つです。データは y（応答変数。現象そのもの）と x（説明変数。モデルの構成要素）に分かれていますが、それを意識しながら以下を読んでください。

- y の誤差構造: y の確率分布、すなわち確率論的モデル。
- 線形予測子: モデル式（$\beta_1 x_1 + \beta_2 x_2 + ...$）。
- リンク関数: 確率論的モデルのパラメータと線形予測子をつなぐもの。線形予測子に逆リンク関数を作用させたものが決定論的モデルと考えるとわかりやすい。

誤差構造

glm() で比較的よく用いられる誤差構造（確率論的モデル）としては、以下のものがあげられます。

- gaussian: いわゆる正規分布。平均と分散の 2 パラメータで定義され、連続値。
- binomial: 二項分布。**生死データ**などで使う。平均（生起確率といっていいか）1 パラメータで定義。
- poisson: ポアソン分布。**カウントデータ**（1 個、2 個...）で使う。平均 1 パラメータで定義。

リンク関数

各誤差構造に対し、通常用いられるリンク関数は以下のようです。

- gaussian の場合: identify（特に変換しない）
- binomial の場合: logit
- poisson の場合: log

なお、各誤差構造に対して示したリンク関数は標準のリンク関数であり、通常は **R 上では入力する必要はありません**。

4.2.6 GLM の結果の見方

では GLM での解析を行ってみましょう。GLM の結果を見るためには以下のようにします。

GLM の実行例

```
> res <- glm(Surv ~ Light + Nutrient, family=binomial, d)
> summary(res)
```

```
Call:
glm(formula = Surv ~ Light + Nutrient, family = binomial, data = d)
```

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5992   0.2709   0.3792   0.6678   1.4863

Coefficients:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)     -0.5564     0.1870  -2.975  0.00293 **
Light            3.9648     0.3110  12.749  < 2e-16 ***
Nutrientijime   -0.1874     0.2113  -0.887  0.37510
Nutrientzeitaku -0.2710     0.2082  -1.301  0.19312
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1071.25  on 999  degrees of freedom
Residual deviance:  860.14  on 996  degrees of freedom
AIC: 868.14

Number of Fisher Scoring iterations: 5
```

見るべきものとしては、

- Coefficients: Estimate が推定された係数
- deviance: Null deviance が切片項のみの deviance、Residual deviance が説明変数を投入したときの deviance

が挙げられます。deviance とはあてはまりの悪さを示し、これが小さいほどデータによくあてはまっていることを示しています。

Coefficients の見かた

- 連続変数（Light）: 生残率に正の効果
- カテゴリー変数（Nutrient）: アルファベットで最初のカテゴリーの係数を 0 としたときの、他のカテゴリーの影響の仕方が表示されている。今回は処理間の差はあまりなさそうである。

4.2.7 AIC によるモデル選択

AIC は Akaike Information Criterion の略で、

$$\mathrm{AIC} = -2 \times \text{maximum log likelihood} + 2p$$

と定義されます。**AIC が小さいほどよいモデル**とされます。

- maximum log likelihood とは最大化対数尤度と訳され、先ほど紹介した尤度の対数をとったものです。尤度同様、想定した確率分布の元でのデータへの当てはまりがよいほど大きい値を取ります（ちなみに-2 をかけたものが deviance)。
- p はパラメータ数、つまりモデルに投入する説明変数の数です。

これらからわかるように、AIC は**当てはまりがよく、かつ出来るだけ説明変数が少ないモデルを選ぶ**ようになっています。

AIC によるモデル選択を行うためには、パッケージ MASS に含まれる stepAIC() を使います。

```
> library(MASS)
> stepAIC(res)
Start:  AIC=868.14
Surv ~ Light + Nutrient

           Df Deviance     AIC
- Nutrient  2   861.92  865.92
<none>          860.14  868.14
- Light     1  1070.95 1076.95

Step:  AIC=865.92
Surv ~ Light

        Df Deviance     AIC
<none>       861.92  865.92
- Light  1  1071.25 1073.25

Call:  glm(formula = Surv ~ Light, family = binomial, data = d)

Coefficients:
(Intercept)        Light
    -0.6943       3.9292

Degrees of Freedom: 999 Total (i.e. Null);  998 Residual
Null Deviance:     1071
Residual Deviance: 861.9  AIC: 865.9
```

ここで決定されたモデルに含まれる変数は意味のある変数であり、含まれなかった変数はそうではない変数、と考えるわけです。つまり、今回だと Nutrient は効いていない変数と判断されたわけです。

こうして、光のみが植物の生残に影響しているという結果が得られました。ここで、ちょっと先ほどの survML.xls に戻ってみましょう。ここに推定結果の切片と光の係数を入れ、対数尤度を-2 倍して deviance を計算すると.....？

4.2.8 結果の図示

統計をかけたら、その結果を図示するのもやっぱり大事なことです。統計解析が正しいかを判断するために。作図方法ですが、

- 結果を付与したオブジェクトから各変数の係数を取り出す
- 得られた確率の平均値を、リンク関数の逆関数である逆リンク関数に与え、`curve()` に放り込んで描画する

という手順を取ります。回帰直線と一緒です。

ただし推定された係数は、リンク関数が作用している状態での推定値ですから、推定された説明したい現象（ここでは生残確率 y）を取り出すためには、推定値に逆リンク関数を作用させて値を変換する必要があります。

- identify: 特に変換する必要なし
- logit: library(faraway) にある `ilogit()` に推定値を入れる（ロジスティック変換）
- log: `exp()` に推定値を入れる

では、先ほどの結果を図示してみましょう。

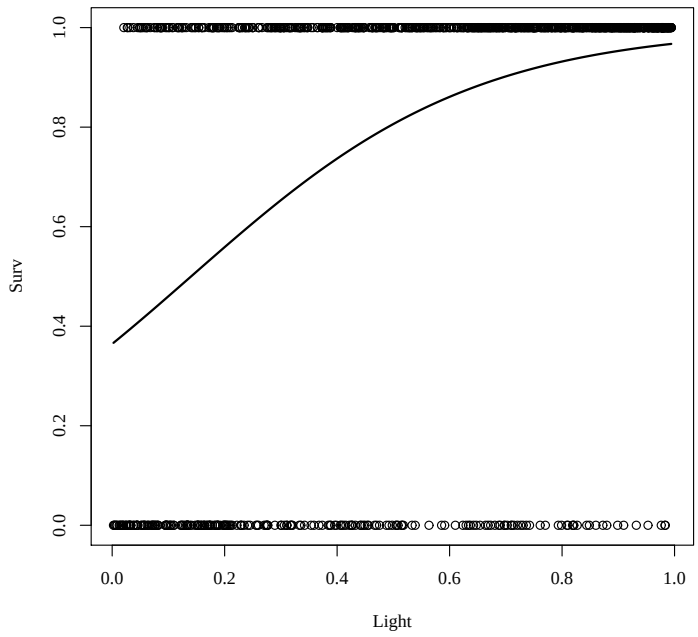

図 15　GLM の当てはめ結果

GLM（binomial）における作図コード

```
> res <- glm(surv ~ light + syori, family=binomial(logit), d)
> resB <- stepAIC(res)
> co <- resB$coefficients
> library(faraway) #関数 ilogit() を含む faraway を呼び出す
> plot(Surv ~ Light, d) #まず図を作ります
> curve(ilogit(co[1] + co[2] * x), lwd=2, add=TRUE)

#ilogit() に頼らない場合。自分で関数を作っちゃうと楽です。
> logistic <- function(x) {exp(x) / (1 + exp(x)) }
> plot(surv ~ light, d)
> curve(logistic(co[1] + co[2] * x), lwd=2, add=TRUE)
```

ポイントは curve() に放り込む値です。今回は Light しか影響していなかったので、切片と Light を足した値を、逆リンク関数であるロジスティック変換（ilogit()）して曲線を描いています。

4.3 GLMM&モデル選択

4.3.1 GLMM がなぜ必要か

GLMM とは Generalized Linear Mixed Model の略です。Mixed は、GLM で変量効果（Random effect）を含むモデルなのでついています。

GLM では、興味のある現象に影響していると考えられる要因を決定論的モデルに取り入れ、それに確率論的モデルによる変動があると考えて、現象を説明するモデルを構築しました。

しかし、決定論的モデルに組み込む要因はあくまで我々が意識的に取り上げるものであり、それだけでデータが説明し切れることはまずありません。我々が測定しない要因でデータに影響する要因を Random effect と呼び、これはいたるところに潜んでいます。

Random effect の例〜世の中 Random effect だらけ！

- 個体差（同じ処理区なのに.....）
- ブロック間差
- その場所に固有な何か
- 時間（年）変動

今回のデータは、

- Nutrient 区ごとにいくつかの Plot が反復として取られている
- 各 Plot から 100 個体のデータが得られている

となっています。GLMM を使えば、**プロット間の値のばらつきを考慮しながら処理区間差を見る**ことが、**プロットを Random effect にする**ことで可能になります。

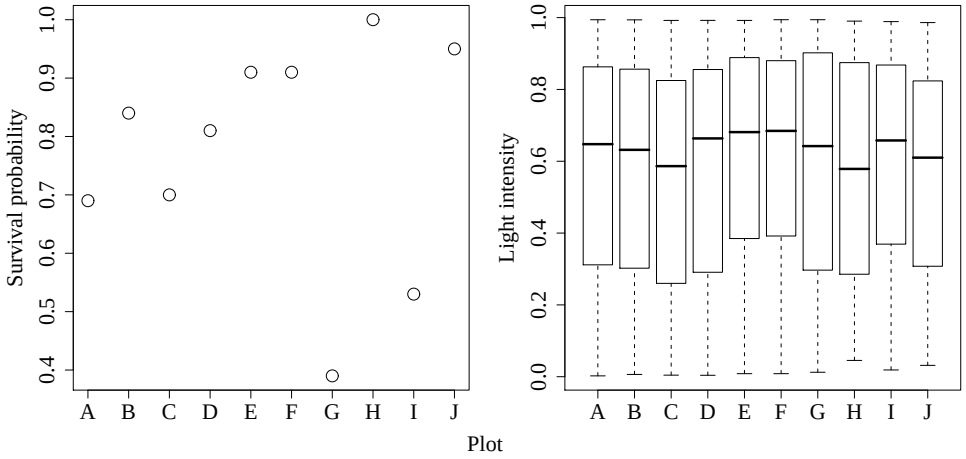

図 16　Random effect が必要なケース

Plot 間で生残率の差がかなりあるが、主要な要因である光環境はプロット間で差がない。この場合、観測できていない要因によってプロット間差が発生していると考えられる。

作図コード

```
> par(mfrow=c(1, 2), mar=c(1, 5, 1, 1), oma=c(4, 0, 0, 0), ps=20)
> plot(1:10, tapply(d$Surv, d$Plot, mean), xlim=c(1, 10),
+      xlab="", ylab="Survival probability", xaxt="n", cex=2.0)
> axis(1, at=1:10, labels=LETTERS[1:10])
> plot(Light ~ Plot, xlab="", ylab="Light intensity", d)
> mtext(1, line=2, outer=TRUE, text="Plot")
```

4.3.2　Random effect **とは？**

GLMM では、Random effect を、平均 0、分散**である**正規分布に従う変数として扱います**。つまり、平均値は変わらないが、Random effect によってデータのばらつきだけが変化するような変数を 1 個導入してやるわけです。これは式の形で書くのであれば、

y（応答変数） = x1**（説明変数** 1**）**+ x2**（説明変数** 2**）**+ ... + RE**（**Random effect**）**

という形で考えることが出来ます。

今回のデータでは、Plot が Random effect になりますが、Plot は特に差を見たい対象ではありません。この Plot 一つ一つに関して**厳密に**推定値を定めることは可能ですが、それは Plot の数だけパラメータ数を増やすことであり、あてはまりがよくなるのは当たり前です。しかし、Plot は特に興味のある対象ではないので、その推定値を求めてもあまり意味がありませんし、モデルの一般性は大きく低下します（だから AIC などはパラメータ数と当てはまりのバランスをとる）。

そこで、Random effect は、何かの確率変数に従って生じる値とします。こうすることで、各 Plot にはある程度のばらつきがあることを反映させながら、パラメータとしては 1 つ増えるだけですみます。

確率分布に従って生じる値として考えるメリットは他にもあります。Random effect のように興味がない要因についてはあまり考慮しないでデータを取ることが多いので、しばしばその効果をデータから推定することは難しいです（ある Plot だけ点数が少ない、とか）。そのような状況で各 Plot に関する推定値を厳密に決定しようとすると、しばしば現実的にはありえない値を推定してしまいます（例えば生残率のデータで、もともと 1 個体しかいなくてそれが生残した場合、生残率は 100% になってしまう）。

しかし、ある確率分布に従うという縛りの中で推定すれば、確率分布の形の縛りにより、Plot の中では最も生残率が高い場所として推定されますが、極端に大きな値をとりにくくなります。つまり、Random effect は不確実性を伴うパラメータを扱う方法として、優れた点が多くあるといえます。

このような Random effect の考え方自体は決して新しいものではなく、例えば検定の枠組みでも Nested ANOVA などといった形の混合モデルで扱われてきました。しかし、検定の枠組みで変量効果を扱うことは一般的に難しく、あまり取り入れられることがありませんでした。しかし、その制約は GLMM によって取り払われた今、データを得る過程で含まれると考えられる Random effect を考慮することが一般的になりつつあります。

4.3.3 こんなときは GLMM だ！

- 影響を検討したい変数以外に、応答変数に明らかに影響を与えている要因がある。
- その要因に関して、要因として評価はしたくないが、無視もできない。
- 実質的に、ほとんどのデータ解析がこれに該当する。

4.3.4 R における GLMM の実行関数

- `glmmML()`: パッケージ glmmML に収録。library(glmmML) で使える。
  長所: stepAIC() など既存の関数が適用しやすい。安定している。
  短所: 扱える誤差構造は binomial、poisson のみ。入れられる Random effect は 1 個だけ
  基本形: #RE は Random effect、d はデータフレーム名
  `glmmML(y ~ x1 + ..., cluster=RE, family=**, d)`
- `lmer()`: パッケージ lme4 に収録。library(lme4) で使える。
  長所: 扱える誤差構造が gaussian、binomial、poisson など多い。Random effect は複数指定可能（しかも切片と傾きどちらにも投入できる）。
  短所: 関数の仕様が特殊なため、既存の関数が適用できないことが多い。
  基本形:
  `lmer(y ~ x1 + ... + (1 | RE), family=**, d)`

それぞれの解析例を示します。

glmmML() の使用例

```
> library(glmmML)
> res <- glmmML(Surv ~ Light + Nutrient, cluster=Plot, family=binomial, d)
> summary(res)
```

```
Call:  glmmML(formula = Surv ~ Light + Nutrient, family = binomial, data = d, cluster = Plot)
```

```
                  coef se(coef)      z Pr(>|z|)
(Intercept)     -0.7076   0.6830 -1.036    0.300
Light            5.8076   0.4552 12.758    0.000
Nutrientijime   -0.3141   0.2590 -1.213    0.225
Nutrientzeitaku -0.4025   0.2533 -1.589    0.112

Scale parameter in mixing distribution:  2.001 gaussian
Std. Error:                              0.5614

Residual deviance: 628.6 on 995 degrees of freedom  AIC: 638.6
```

GLM と大きな差はありません。Null deviance が出力されないぐらいが違いでしょうか。また、光の推定値が GLM と大きく異なっていることに注目してください。

lmer() の使用例

```
> library(lme4)
> res2 <- lmer(Surv ~ Light + Nutrient + (1 | Plot), family=binomial, d)
> res2
```

```
Generalized linear mixed model fit by the Laplace approximation
Formula: Surv ~ Light + Nutrient + (1 | Plot)
   Data: d
   AIC   BIC logLik deviance
 638.6 663.2 -314.3    628.6
Random effects:
 Groups Name        Variance Std.Dev.
 Plot   (Intercept) 4.0055   2.0014
Number of obs: 1000, groups: Plot, 10

Fixed effects:
               Estimate Std. Error z value Pr(>|z|)
(Intercept)     -0.7076     0.6797  -1.041    0.298
Light            5.8076     0.4496  12.918   <2e-16 ***
Nutrientijime   -0.3141     0.2581  -1.217    0.224
Nutrientzeitaku -0.4026     0.2524  -1.595    0.111
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Correlation of Fixed Effects:
            (Intr) Light  Ntrntj
Light       -0.214
Nutrientijm -0.153 -0.127
Nutrientztk -0.152 -0.140  0.496
```

こちらは大分様子が違います。まずモデルの当てはまりを示す AIC などが示されており、その下に Random effect がどの程度のばらつきを説明しているかが表示され、その下に説明変数に関する情報が載せられています。結果のオブジェクトを `fixef()` に入れると説明変数が、`ranef()` に入れると Random effect の値が、`AIC(logLik( ))` に入れると AIC が表示されます。

今回のように Random effect の効果が強い場合、推定結果は見ていただいたように結構変わります（図 17）。

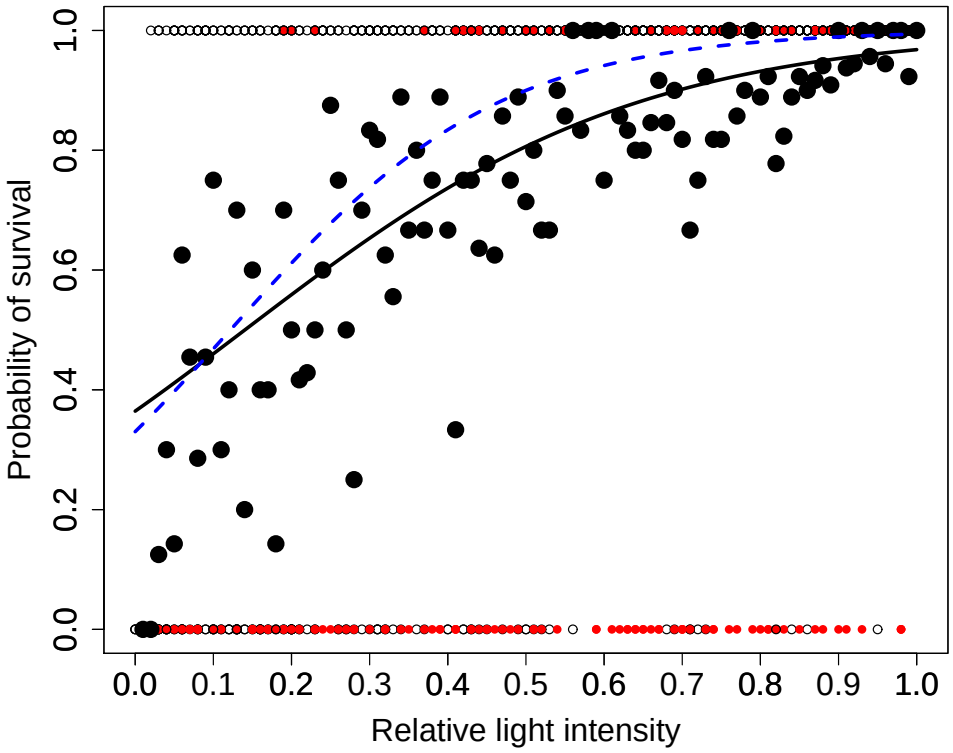

図 17　GLM と GLMM の推定結果の違い

小さい丸が実測値（塗りつぶしの丸は生残率が低いプロット）、大きい丸が光環境別の平均値。GLM の推定結果は黒い曲線、GLMM の推定結果は青い破線で示している。GLM の推定結果は Plot のばらつきを考慮していないため、全体に合わせるような推定結果となっている。

4.3.5　GLMM におけるモデル選択

`glmmML()` については `stepAIC()` を、`lmer()` については `dredge()` を使います。`dredge()` はパッケージ MuMIn に含まれているのですが、MuMIn は通常のパッケージのインストール項目には含まれていませんの

で、http://r-forge.r-project.org/projects/mumin/ からダウンロードしてください。

上記のページにアクセスし、Download という緑色のアイコンをクリックし、Windows の方でしたら Windows binary(.zip) をクリックし、zip ファイルをダウンロードしてください。

次に R の「パッケージ」→「ローカルにある zip ファイルからのパッケージのインストール」を選び、先ほどのダウンロードした zip ファイルを選択してください。

4.3.6 使い方

結果のオブジェクトを、それぞれの関数に入れるだけです。まずは glmmML() の場合。

```
> library(glmmML)
> res <- glmmML(Surv ~ Light + Nutrient, cluster=Plot, family=binomial, d)
> library(MASS)
> stepAIC(res)
Start:  AIC=638.62
Surv ~ Light + Nutrient

           Df    AIC
- Nutrient  2 637.43
<none>        638.62
- Light     1 914.57

Step:  AIC=637.43
Surv ~ Light

[glmmml] fail = 1
Max. No. of iterations reached without convergence        Df    AIC
<none>     637.43
- Light  1 979.39

Call:  glmmML(formula = Surv ~ Light, family = binomial, data = d, cluster = Plot)



               coef se(coef)      z Pr(>|z|)
(Intercept) -0.9087   0.6674 -1.362    0.173
Light        5.7181   0.4467 12.800    0.000

Scale parameter in mixing distribution:  1.987 gaussian
Std. Error:                              0.5563

Residual deviance: 631.4 on 997 degrees of freedom  AIC: 637.4
```

```
Warning messages:
1: In glmmML.fit(X, Y, weights, cluster.weights, start.coef, start.sigma,  :

2: In glmmML(formula = Surv ~ 1, family = binomial, data = d, cluster = Plot) :
  'vmmin' did not converge. Increase 'maxit'?
```

あれ？なんかエラーが出てしまいましたね。これに関しては後ほど。続いて lmer() の場合です。

```
> res2 <- lmer(Surv ~ Light + Nutrient + (1 | Plot), family=binomial, d)
> dredge(res2)
Model selection table
   (Intr) Light Nutrnt k  Dev.   AIC  AICc   delta weight
2 -0.9082 5.718        3 631.4 637.4 637.4   0.000  0.649
4 -0.7076 5.808      1 5 628.6 638.6 638.7   1.232  0.351
1  1.6620              2 906.9 910.9 910.9 273.500  0.000
3  1.6870            1 4 906.6 914.6 914.6 277.200  0.000
Random terms: 1 | Plot
```

dredge() は、すべての組み合わせのモデルの AIC を検討し、AIC が小さいモデルほど上に表示します。カテゴリー変数については、選択されている場合には 1 が示されています。

4.3.7 GLMM のトラブルシュート

glmmML()、lmer() ともによくエラーが発生します。数学的には解けないモデルなので、数値探索によって解析的に解を出しています。そのため、うまく収束しないとエラーが発生します。そのときの対処方法です。

- 本当にそのモデルでよい？: これが結局一番大事です。自分の考えているモデルが（生物学的に）正しいのか、Random effect として取り込めるものが他にないか（逆に必要ないか）など、モデルそのものについてよく考えましょう。

以下は、glmmML() のみ有効です。

- 引数 start.sigma: 解析を始める際の初期値を変えます。デフォルトは 0.5 ですので、値を大きくしたり小さくしたりして収束するか試してください。
- 引数 maxit: 収束させるまでに繰り返す計算の回数を決定します。デフォルトは 200 回です（少なっ！）。計算回数を増やすことによって収束するかも知れません（私自身はそういう経験はないですが）。

4.4 検定

今日はやりません。後ほど、必要がありましたら参照してください。

4.4.1 こんなときは検定だ！

- すでに検討したい要因が絞られていて、その要因の影響の差を検討したい。
- 厳密な制御環境にあり、検討したい要因以外の影響を考慮する必要がない。
- 指導者（あるいは Editor）に検定しろって言われて逆らえない^^;)（このケースが一番多いか？）

検定の原理に関しては逐一説明しませんが、基本的には

- 差を見ようとする集団間で差がないと仮定する。
- 両集団からランダムサンプリングしてきて差をとった値（あるいはもう少しこねくり回した値。いわゆる検定統計量）が既存の理論分布（正規分布またはそれに類するものとすることが多い）に**従うと仮定する**。
- 実際の測定値から検定統計量を計算したときに、両集団に差がないと仮定したときの検定統計量の分布においてどこに位置するかを調べ、すごくはしっこ（いわゆる 95 % 点よりも端。つまり珍しいゾーン）だったら、「両集団に差がないという仮定が間違っていた」と結論する。

ということです。

以下に各種検定を行う方法を示します。その前に、以下のことをしておいてください。

```
> d <- read.csv("data.csv")
> d2 <- subset(d, syori!="ijime")
```

4.4.2 2 つのグループの差の検定

t 検定

得られたデータが相互に独立であり、連続数である場合に `t.test()` を使う。

```
> t.test(omosa ~ syori, d2, var.equal=TRUE)
```

`var.equal` を TRUE にしておけば通常の T 検定、FALSE の場合（こちらがデフォルト）は、いわゆる Welch の検定となる。 また、`paired = TRUE` とすることで、対応のある *t* 検定を実行できる。

U 検定

取っているデータが間隔、カテゴリーデータである、あるいははずれ値があるような場合 `wilcox.test()` を使う（分散が大きく異なるときは適さない）。データを全て順位化して用いるので、はずれ値の影響を受けにくい。U 検定は数学的には Wilcoxon の順位和検定と呼ばれる検定と同様であり、R の中では Wilcoxon の順位和検定が行われる。

```
> wilcox.test(omosa ~ syori, d2)
```

4.4.3 3 つ以上のグループ間の差の検定

ANOVA（いわゆる分散分析）

連続数のデータであり、グループの母分散が等しい（等分散性）場合は場合は aov() を使う。ただし、ANOVA は各群の母分散が等しい（等分散性）ことを仮定するので、それを確かめるために、bartlett.test() によって、等分散性の検定を事前に行っている。

```
> bartlett.test(omosa ~ syori, d) #等分散性の検定
> result <- aov(omosa ~ syori, d)
> summary(result) #結果の完全な取出しには summary()
```

Kruskal-Wallis 検定

間隔、カテゴリーのデータで対応のない 3 つ以上のグループ間で中央値に違いがあるかどうかを検定するときは、kruskal.test() を使う。

```
> result <- kruskal.test(omosa ~ syori, d)
```

4.4.4 多重比較（3 群以上のグループ間のどことどこに差があるのか？）

パラメトリックな多重比較法として最も一般的に使われる Tukey 法の場合。

Tukey 法による多重比較

```
> result <- aov(omosa ~ syori, d)
> TukeyHSD(result)
```

ノンパラ検定の場合によく使われる Stee-Dwass 検定については関数は用意されていないので、群馬大学の青木先生が作ったスクリプトを使います。

比率の検定

何対何という比率が、カテゴリー間で異なるか検討するときに用いる。R の中では、chisq.test() を使う。2×2 の表なら Fisher の正確確率検定 fisher.test() を用いる。

```
> e <- matrix(c(100, 20, 70, 50), ncol=2, byrow=T)
> rownames(e) <- c(``Engineer'', ``Agriculture'')
> colnames(e) <- c(``Male'', ``Female'')
#今回は 2 × 2 の表なので、fisher.test() を使ってみる。
> fisher.test(e)
#2 × 2 以外の表なら chisq.test()。
> chisq.test(e)
```

4.4.5 要因が 2 つ以上の場合の検定

2 変数の相関の有無

cor.test() を使う。

```
> cor.test(d$omosa, d$miki)
```

回帰（1 変数から、もう 1 変数を予測できるか？）

lm() を使う。

```
> result <- lm(miki ~ omosa, d)
> summary(result)
```

重回帰（複数の変数から、1 変数を予測できるか？）

lm() を使う。当然だが応答変数の分布形は正規分布限定。

```
> result <- lm(miki ~ omosa + ha, d)
> summary(result)
```

混合モデルの分散分析

変量効果と考えられる要因が 1 つ含まれる場合の分散分析。Nested-ANOVA と言われることもある。今回のデータでは FL が変量効果。library(car) に含まれる Anova と lm() を使う。

```
> library(car)
> result <- Anova(lm(miki ~ omosa, d), error=lm(miki ~ FL, d))
> result
```

ANOVA（N 元配置の分散分析）

library(car) に含まれる Anova と lm() を使う。

```
> library(car)
> result <- Anova(lm(miki ~ omosa * syori, d), type="II")
> result
```

式の中の*は単独項と交互作用項を含める記号。:なら交互作用項のみが投入される。

ただし、3 次元以上の分散分析は大概呪われた（解釈不能な）結果を返します。また、変数変換によって応答変数を無理やり正規分布にして解析すると、存在しないはずの交互作用項が検出されることがあります（Kasuya 2004）。これらのような複雑系の結果の解析をされる方は GLM に移行することを強くお勧めします。

検定の結果は、基本的に p の値を見れば判断できます。

5 R の情報源

いろいろありますが、まず最低限抑えておく必要があるのが、

- 「生態学の統計モデリング」:http://hosho.ees.hokudai.ac.jp/~kubo/ce/EesLecture2007.html
  北大地環研の久保先生の統計講義資料（全 7 回)。データをどう扱うか？ということに関して R を使って非常に丁寧に解説している。そのうち岩波書店からまとめたものが出版される？
- 「統計学：R を用いた入門書」: Michael J. Crawley（訳：野間口謙太郎・菊池泰樹）
  R の入門書として非常に評価が高い Crawley 本の邦訳。R の操作を学びつつ、統計学の基礎も習得できる。
- 「The R Tips」: 船尾暢男　著（九天社)、ISBN-10: 486167039X
  R のコマンドのレファレンスとして非常に有用。現在はもう販売されていないが、ネット上にも同じもの（http://cse.naro.affrc.go.jp/takezawa/r-tips/r.html）がある。近々再版される見込み。
- 「データ解析環境 R」: 船尾暢男・高浪洋平　著（工学社)、ISBN-10: 4777511847
  R を一から一人で始めるのに適した入門書。それほど細かいことは書いていないが、とりあえず R を一通り動かしたい人にお勧め。
- 「RjpWiki」: http://www.okada.jp.org/RWiki/index.php?RjpWiki
  日本における R の総本山のようなサイト。R のインストール方法、基本からかなりマニアックなコマンドの使い方、（辛辣な返答が多いが）質問掲示板などがある。
- よき相談者^^;)。

です。これに加えて、以下も有用です。

- 「R による統計処理」: http://aoki2.si.gunma-u.ac.jp/R/
  群馬大学の青木先生による、R の使い方ページ。便利な統計関数が多数そろう。
- 「R による統計解析の基礎」: http://phi.ypu.jp/statlib/stat.pdf
  群馬大学の中澤先生による、R による統計解析に関する PDF ファイル。より医学関係のデータ解析例に特化した内容の PDF ファイル、「R による保健医療データ解析演習」（http://phi.ypu.jp/msb/medstatbook.pdf）も有用。
- 「The R Project」: http://www.r-project.org/
  R の総本山。

索引