RjpWiki アーカイブス

【回帰、分散分析関数一覧 (06.06.25) 】

R は回帰分析関連の関数を多数持ち、統計解析機能の中心的位置を占める。以下では、線形 (重) 回帰モデル、一般化線形モデル、非線型回帰モデル、そしていくつかの現代的手法用の関数を紹介する。現代的な統計理論では、分散分析も線形回帰モデルとして処理することが普通であるため、分散分析関連の関数もここで一緒に紹介するのが適当である。

1　線形回帰モデル

線形モデルを当てはめる lm() 関数 lm() は線形モデルの当てはめに使われる。回帰分析、および一元配置分散・共分散分析を行える (後者に付いては aov() 関数の方がより広範囲なインタフェイスを与える)。

```
lm(formula, data, subset, weights, na.action,
   method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE,
   singular.ok = TRUE, contrasts = NULL, offset, ...)
```

1.1　回帰診断 lm.influence()

回帰当てはめの善し悪しを検査するための広範囲な診断手順に使われる基本的な量を計算する。

```
influence(model, ...)
## クラス "lm" に対する S3 メソッド:
influence(model, do.coef = TRUE, ...)
## クラス "glm" に対する S3 メソッド:
influence(model, do.coef = TRUE, ...)
lm.influence(model, do.coef = TRUE)
```

1.2　線形モデルの当てはめ関数 lm.fit()

線形モデルを当てはめる lm() から呼び出される、基本計算エンジンである。経験あるユーザでなければ直接使うべきではない。

```
lm.fit (x, y,    offset=NULL, method="qr",
        tol=1e-7, singular.ok=TRUE, ...)
lm.wfit(x, y, w, offset=NULL, method="qr",
        tol=1e-7, singular.ok=TRUE, ...)
```

1.3 lm オブジェクトの診断図 plot.lm()

lm オブジェクトの診断図を描く。現在 4 種類のプロット (which で選択できる) が提供されている : 当てはめ値に対する残差プロット、当てはめ値に対する sqrt(|residuals|) の Scale-Location プロット、正規 Q-Q プロット、行ラベルに対する Cook の距離のプロットである。

```
## クラス "lm" に対する S3 メソッド :
plot(x, which = 1:4,
     caption = c("Residuals vs Fitted", "Normal Q-Q plot",
                 "Scale-Location plot", "Cook's distance plot"),
     panel = points, sub.caption = deparse(x$call), main = "",
     ask = prod(par("mfcol")) < length(which) && dev.interactive(),
     ..., id.n = 3, labels.id = names(residuals(x)), cex.id = 0.75)
```

1.4 線形モデルによる予測 predict.lm()

線形モデルオブジェクトに基づく予測メソッドである。

```
## クラス "lm" に対する S3 メソッド
predict(object, newdata, se.fit = FALSE, scale = NULL, df = Inf,
        interval = c("none", "confidence", "prediction"),
        level = 0.95, type = c("response", "terms"),
        terms = NULL, na.action = na.pass, ...)
```

1.5 線形モデル当てはめの要約 summary.lm()

クラス "lm" に対する summary メソッドである。

```
## クラス "lm" に対する S3 メソッド
summary(object, correlation = FALSE, symbolic.cor = FALSE, ...)
## クラス "lm" に対する S3 メソッド
print(x, digits = max(3, getOption("digits") - 3),
      symbolic.cor = x$symbolic.cor,
     signif.stars = getOption("show.signif.stars"), ...)
```

2 一般化線形回帰モデル

2.1 一般化線形モデルの当てはめ glm()

線形予測子と誤差分布のシンボリックな記述を指定して、一般化線形モデルを当てはめる。

```
glm(formula, family = gaussian, data, weights, subset,
    na.action, start = NULL, etastart, mustart,
    offset, control = glm.control(...), model = TRUE,
    method = "glm.fit", x = FALSE, y = TRUE, contrasts = NULL, ...)
glm.fit(x, y, weights = rep(1, nobs),
        start = NULL, etastart = NULL, mustart = NULL,
        offset = rep(0, nobs), family = gaussian(),
        control = glm.control(), intercept = TRUE)
## クラス "glm" に対する S3 メソッド
weights(object, type = c("prior", "working"), ...)
```

2.2 モデルファミリ family()

ファミリオブジェクトは glm() 関数等でモデルの詳細を指定する便利な手段を提供する。そのようなモデルを使った当てはめがどのように行われるかに付いては glm() のヘルプドキュメントを参照せよ。

```
family(object, ...)
binomial(link = "logit")
gaussian(link ="identity")
Gamma(link = "inverse")
inverse.gaussian(link = "1/mu^2")
poisson(link = "log")
quasi(link = "identity", variance = "constant")
quasibinomial(link = "logit")
quasipoisson(link = "log")
```

2.3 GLM ファミリに対するリンクを作る make.link()

この関数は glm() 中の family() 関数とともに用いられる。リンクを与えると、リンク関数、逆リンク関数、導関数 d mu/d eta、そして領域チェックのための関数が返される。

```
make.link(link)
```

2.4 巾乗リンクオブジェクトを作る power()

リンク関数 eta = mú lambda に基づくリンクオブジェクトを作る。

```
power(lambda = 1)
```

2.5 一般化線形モデルによる予測 predict.glm()

一般化線形モデル当てはめオブジェクトから予測値を得たり、オプションとしてこれらの予測値の標準偏差を推定する。

```
## クラス "glm" に対する S3 メソッド
predict(object, newdata = NULL,
        type = c("link", "response", "terms"),
        se.fit = FALSE, dispersion = NULL, terms = NULL,
        na.action = na.pass, ...)
```

2.6 一般化線形モデル当てはめ結果の要約 glm.summaries()

これらの関数はクラス "glm" もしくは "summary.glm" のオブジェクトのメソッド関数である。

```
## クラス "glm" に対する S3 メソッド
summary(object, dispersion = NULL, correlation = FALSE,
        symbolic.cor = FALSE, ...)
## クラス "summary.glm" に対する S3 メソッド
print(x, digits = max(3, getOption("digits") - 3),
      symbolic.cor = x$symbolic.cor,
      signif.stars = getOption("show.signif.stars"), ...)
```

2.7 GLM 当てはめを制御する glm.control()

glm() による当てはめに対するユーザインタフェイスを与える補助関数である。典型的には、glm() または glm.fit() を呼び出すときにだけ使われる。

```
glm.control(epsilon=1e-8, maxit=25, trace=FALSE)
```

3 分散分析表

3.1 分散分析モデルの当てはめ aov()

lm() を各層に適用し、分散分析モデルを当てはめる。

```
aov(formula, data = NULL, projections = FALSE, qr = TRUE,
    contrasts = NULL, ...)
```

3.2 分散分析表を作る anova()

一つまたは複数の当てはめオブジェクトから分散 (または逸脱度) 分析表を計算する。

```
anova(object, ...)
```

3.3 分散分析モデルの要約 summary.aov()

分散分析モデルの要約を与える。

```
## クラス aov に対する S3 メソッド
summary(object, intercept = FALSE, split,
        expand.split = TRUE, keep.zero.df = TRUE, ...)
## クラス aovlist に対する S3 メソッド
summary(object, ...)
```

3.4 線形モデル当てはめ結果に対する分散分析 anova.lm()

一つ、もしくは複数の線形モデル当てはめ結果に対する分散分析表を計算する。

```
## クラス "lm" に対する S3 メソッド
anova(object, ...)
anova.lmlist(object, ..., scale = 0, test = "F")
```

3.5 GLM の分散分析統計量 stat.anova()

これは anova(..., test != NULL) に対する lm() と glm() メソッドであり、普通のユーザが使うべきではない。

```
stat.anova(table, test = c("Chisq", "F", "Cp"), scale, df.scale, n)
```

3.6 GLM モデル当てはめによる逸脱度分析表 anova.glm()

一つもしくは複数の一般化線形モデル当てはめ結果に対する逸脱度分析表を計算する。

```
## クラス "glm" に対する S3 メソッド
anova(object, ..., dispersion = NULL, test = NULL)
```

3.7 多変量分散分析 manova()

多変量分散分析に対するクラス属性を与える。

```
manova(...)
```

3.8 多変量分散分析に対する要約メソッド summary.manova()

多変量分散分析に対する要約メソッド関数である。

```
## クラス "manova" に対する S3 メソッド
summary(object,
        test = c("Pillai", "Wilks", "Hotelling-Lawley", "Roy"),
        intercept = FALSE, ...)
```

3.9 Tukey の Honest Significant Difference 法 TukeyHSD()

```
TukeyHSD(x, which, ordered = FALSE, conf.level = 0.95, ...)
```

4 非線形回帰モデル

4.1 非線形モデルの当てはめ nls()

非線形モデルパラメータを非線形最小自乗法で決定し、クラス ”nls” のオブジェクトを返す。

```
nls(formula, data = parent.frame(), start, control = nls.control(),
    algorithm = "default", trace = FALSE, subset,
    weights, na.action, model = FALSE)
```

4.2 nls オブジェクトからモデル式を取り出す formula.nls()

object を当てはめるために使われるモデルを返す。

```
## クラス "nls" に対する S3 メソッド
formula(x, ...)
```

4.3 非線形最小自乗当てはめの制御 nls.control()

nls() 関数による非線形最小自乗法アルゴリズムの幾つかの特性をユーザが設定できるようにする。

```
nls.control(maxiter=50, tol=1e-05, minFactor=1/1024)
```

4.4 nlsModel オブジェクトを作る nlsModel()

nlsModel オブジェクトを構成する。このオブジェクトはリスト形式の幾つかの関数の関数閉包である。この閉包は非線形モデル式、式に対するデータ値、そしてパラメータとそれらの値を含む。

```
nlsModel(form, data, start)
```

4.5 ”profile.nls” オブジェクトのプロット plot.profile.nls()

プロファイル関数の一連のプロットを表示し、nls() 関数で当てはめられ profile.nls() でプロファイルされた非線形回帰モデル中のパラメータに対する信頼区間を補間する。

```
## クラス "profile.nls" に対する S3 メソッド
plot(x, levels, conf= c(99, 95, 90, 80, 50)/100,
     nseg = 50, absVal =TRUE, ...)
```

4.6 非線形回帰モデルによる予測 predict.nls()

フレーム newdata 中の回帰関数を評価して得られる予測値を計算する。もし論理値 se.fit が TRUE なら、予測の標準偏差が計算される。もし数値引数 scale が設定 (オプション df 付き) されていれば、それが標準偏差の計算の過程で残差標準偏差として使われる。さもなければ、これは当てはめモデルから取り出される。intervals は指定された水準 level で信頼区間、予測 (許容) 区間のどちらを計算するかを指示する。現在のところ se.fit と interval は無視される。

```
## クラス "nls" に対する S3 メソッド
predict(object, newdata , se.fit = FALSE, scale = NULL, df = Inf,
        interval=c("none","confidence","prediction"),level=0.95,...)
```

4.7 nls オブジェクトのプロファイル profile.nls()

nls オブジェクトの fitted で表される解の近傍での対数尤度関数の挙動を調べる。

```
## クラス "nls" に対する S3 メソッド
profile(fitted, which=1:npar, maxpts=100, alphamax=0.01,
        delta.t=cutoff/5, ...)
```

4.8 非線型回帰オブジェクトに対するプロファイラを構成する profiler.nls()

クラス ”nls” の当てはめオブジェクトに対するプロファイラを構成する。

```
## クラス "nls" に対する S3 メソッド
profiler(fitted, ...)
```

4.9 対数線形モデル loglin()

比率の当てはめの繰り返しにより、対数線形モデルを多元分類分割表に当てはめる。

```
loglin(table, margin, start = rep(1, length(table)), fit = FALSE,
       eps = 0.1, iter = 20, param = FALSE, print = TRUE)
```

5 漸近回帰モデル

しばしば用いられる非線型回帰モデルを扱う関数を紹介する。これらはモデル曲線が漸近線を持ち、それがパラメータ値に関係することから、漸近回帰モデル (asymptotic regression model) と呼ばれる。こうしたモデルではモデルパラメータの適切な初期推定値を見出すことが困難な問題となるが、自己開始モデル (SS, Self-Start model) と呼ばれるメカニズムは、こうした初期推定値を計算する手順自身をモデルに含むことを可能にする。

5.1 漸近回帰モデル NLSstAsymptotic()

xy データに $b_0 + b_1$ (1-exp(-exp(lrc) x) の形の漸近回帰モデルを当てはめる。これはより複雑なモデルの初期推定値を決定するのに使うことができる。

```
NLSstAsymptotic(xy)
```

5.2 漸近回帰関数とグラディエントの評価 SSasymp()

この「自己開始」モデルは漸近回帰関数とそのグラディエントを評価する。これは与えられたデータセットに対しパラメータ Asym, R0, lrc の初期推定値を評価する「初期属性」を持つ。

```
SSasymp(input, Asym, R0, lrc)
```

5.3 オフセットを持つ漸近回帰モデル SSasympOff()

もう一つのパラメータ化を持つ漸近回帰関数とそのグラディエントを評価する。これはパラメータ Asym, lrc, c0 の初期推定値を計算する initial 属性を持つ。

```
SSasympOff(input, Asym, lrc, c0)
```

5.4 原点を通る漸近回帰モデル SSasympOrig()

原点を通る漸近回帰モデルとそのグラディエントを評価する。与えられたデータに対してパラメータ Asym と lrc の初期推定値を計算する initial 属性を持つ。

```
SSasympOrig(input, Asym, lrc)
```

5.5 二重指数モデル SSbiexp()

この自己開始モデルは二重指数モデルとそのグラディエントを評価する。これはパラメータ A1, lrc1, A2 そして lrc2 の初期推定値を生成する ”initial” 属性を持つ。

```
SSbiexp(input, A1, lrc1, A2, lrc2)
```

5.6 自己開始型非線型モデル selfStart()

自己開始型非線型モデル (SS, self-start non-linear model) を構成する。

```
selfStart(model, initial, parameters, template)
```

5.7 非線型回帰モデルのパラメータ初期値を求める getInitial()

非線型回帰モデルに対する初期パラメータ推定値を求める。もし data がパラメータ化されたデータフレームか、pframe オブジェクトなら、その parameter 属性が返される。さもなければ、オブジェクトが評価可能な”initial” 属性を持つ selfStart オブジェクトへの呼び出しを含むかどうかが検査される。

```
getInitial(object, data, ...)
```

5.8 一次のコンパートメントモデル SSfol()

この「自己開始」モデルは一次のコンパートメント関数とそのグラディエントを評価する。パラメータ lKe, lKa, lCl の初期推定値を計算する ”initial” 属性を持つ。

```
SSfol(Dose, input, lKe, lKa, lCl)
```

5.9 4パラメータのロジスティックモデル SSfpl()

4パラメータロジスティック関数とそのグラディエントを評価する。与えられたデータセットに対しパラメータ A, B, xmid そして scal の初期推定値を評価する ”initial” 属性を持つ。

```
SSfpl(input, A, B, xmid, scal)
```

5.10 ゴンペルツ成長曲線モデル SSgompertz()

この自己開始モデルはゴンペルツ成長曲線モデルとそのグラディエントを評価する。パラメータ Asym, b2, b3 の初期推定値をつくり出すための ”initial”属性を持つ。

```
SSgompertz(x, Asym, b2, b3)
```

5.11 ロジスティックモデル SSlogis()

自己開始モデルであり、ロジスティック関数とそのグラディエントを評価する。与えられたデータセットに対しパラメータ Asym, xmid, scal の初期推定値を評価する initial 属性を持つ。

```
SSlogis(input, Asym, xmid, scal)
```

5.12 Michaelis-Menten モデル SSmicmen()

Michaelis-Menten モデルとそのグラディエントを評価する。与えられたデータセットに対しパラメータ Vm, K の初期推定値を評価する initial 属性を持つ。

```
SSmicmen(input, Vm, K)
```

5.13 ワイブル成長モデル SSweibull()

ワイブル成長モデルとそのグラディエントを評価する自己開始モデルである。与えられたデータセットに対しパラメータ Asym, Drop, lrc そして pwr の初期推定値を評価する ”initial” 属性を持つ。

```
SSweibull(x, Asym, Drop, lrc, pwr)
```

5.14 漸近回帰モデルのその他の補助関数

- sortedXyData(x, y, data) : x, y 座標データから関数を表す散布図データを計算する。自己開始モデルに対する initial 関数の内部で使われる。
- NLSstClosestX(xy, yval) : 関数を表す散布図データ xy (sortedXyData() 関数の返り値オブジェクト) から、線形補間により値が yval となる x 値を計算する。
- NLSstLfAsymptote(xy) : sortedXyData() 関数の返り値オブジェクトから、関数の左側水平漸近値の初期推定を行う。自己開始モデルに対する initial 関数の内部で使われる。
- NLSstRtAsymptote(xy) : sortedXyData() 関数の返り値オブジェクトから、関数の右側水平漸近値の初期推定を行う。自己開始モデルに対する initial 関数の内部で使われる。

6 回帰モデル用補助関数

6.1 モデルに一つの項を加える、取り去る add1()

モデルに付け加えることのできる、また取り去ることのできる全ての単一項を求め、対応するモデルを当てはめ、そして当てはめ結果の差異を計算する。

```
add1(object, scope, ...)
## 既定の S3 メソッド:
add1(object, scope, scale = 0, test = c("none", "Chisq"),
     k = 2, trace = FALSE, ...)
## クラス "lm" に対する S3 メソッド
add1(object, scope, scale = 0, test = c("none", "Chisq", "F"),
     x = NULL, k = 2, ...)
## クラス "glm" に対する S3 メソッド
add1(object, scope, scale = 0, test = c("none", "Chisq", "F"),
     x = NULL, k = 2, ...)
## 既定の S3 メソッド:
drop1(object, scope, scale = 0, test = c("none", "Chisq"),
      k = 2, trace = FALSE, ...)
## クラス "lm" に対する S3 メソッド
drop1(object, scope, scale = 0, all.cols = TRUE,
      test=c("none", "Chisq", "F"),k = 2, ...)
## クラス "glm" に対する S3 メソッド
drop1(object, scope, scale = 0, test = c("none", "Chisq", "F"),
      k = 2, ...)
```

6.2 エイリアス alias()

モデル式指定された線形モデル中のエイリアス (別名、alias)、つまり線形従属項、を見付ける。

```
alias(object, ...)
## クラス "formula" に対する S3 メソッド
alias(object, data, ...)
## クラス "lm" に対する S3 メソッド
alias(object, complete = TRUE, partial = FALSE,
      partial.pattern = FALSE, ...)
```

6.3 モデルのケースと変数名 case.names(), variable.names()

当てはめモデルの (欠けていない) ケース名や、(除去されていない) 変数名を返す簡単なユーティリティである。

```
## クラス "lm" に対する S3 メソッド
case.names(object, full = FALSE, ...)
## クラス "lm" に対する S3 メソッド
variable.names(object, full = FALSE, ...)
```

6.4 対比行列の作成

対比行列 (contrast matrix) を返す。

```
contr.helmert(n, contrasts = TRUE)
contr.poly(n, scores = 1:n, contrasts = TRUE)
contr.sum(n, contrasts = TRUE)
contr.treatment(n, base = 1, contrasts = TRUE)
```

6.5 対比の設定・閲覧 contrasts()

ある因子に関連する対比を設定したり、閲覧したりする。

```
contrasts(x, contrasts = TRUE)
contrasts(x, how.many) <- value
```

6.6 因子に対比を設定する C()

因子に対比属性 ”contrasts” を設定する。

```
C(object, contr, how.many, ...)
```

6.7 本来のコーディングで回帰係数を取り出す dummy.coef()

この関数はコード化されたものではなく、本来の名前で係数を取り出す。

```
dummy.coef(object, ...)
## クラス "lm" に対する S3 メソッド
dummy.coef(object, use.na = FALSE, ...)
## クラス "aovlist" に対する S3 メソッド
dummy.coef(object, use.na = FALSE, ...)
```

6.8 当てはめモデルからの影響 effects()

当てはめモデル、普通線形モデル、から (直交化された) 効果を返す。これは総称的関数であるが、現在クラス "lm" と "glm" を継承するオブジェクトに対するメソッドだけを持つ。

```
effects(object, ...)
## クラス "lm" に対する S3 メソッド
effects(object, set.sign=FALSE, ...)
```

6.9 取り除きによる回帰診断 influence.measures()

この一連の関数は Belsley, Kuh & Welsch [?], Cook & Weisberg [?] 等で論じられている、線形モデルと一般化線形モデルに対する幾つかの (一時に一つ取り除く) 回帰診断を計算するのに使うことができる。

```
influence.measures(model)
rstandard(model, ...)
## クラス "lm" に対する S3 メソッド
rstandard(model, infl = lm.influence(model, do.coef=FALSE),
          sd = sqrt(deviance(model)/df.residual(model)), ...)
## クラス "glm" に対する S3 メソッド
rstandard(model, infl = lm.influence(model, do.coef=FALSE), ...)
rstudent(model, ...)
## クラス "lm" に対する S3 メソッド
rstudent(model, infl = lm.influence(model, do.coef=FALSE),
         res = infl$wt.res, ...)
## クラス "glm" に対する S3 メソッド
rstudent(model, infl = influence(model, do.coef=FALSE), ...)
dffits(model, infl = , res = )
dfbeta(model, ...)
## クラス "lm" に対する S3 メソッド
    dfbeta(model, infl = lm.influence(model, do.coef=TRUE), ...)
    dfbetas(model, ...)
## クラス "lm" に対する S3 メソッド
dfbetas(model, infl = lm.influence(model, do.coef=TRUE), ...)
covratio(model, infl = lm.influence(model, do.coef=FALSE),
         res = weighted.residuals(model))
cooks.distance(model, ...)
## クラス "lm" に対する S3 メソッド
cooks.distance(model, infl = lm.influence(model, do.coef=FALSE),
               res = weighted.residuals(model),
               sd = sqrt(deviance(model)/df.residual(model)),
               hat = infl$hat, ...)
```

```
## クラス "lm" に対する S3 メソッド
cooks.distance(model, infl = influence(model, do.coef=FALSE),
               res = infl$pear.res,
               dispersion = summary(model)$dispersion,
               hat = infl$hat, ...)
hatvalues(model, ...)
## クラス "lm" に対する S3 メソッド
hatvalues(model, infl = lm.influence(model, do.coef=FALSE), ...)
hat(x, intercept = TRUE)
```

6.10 回帰モデルの対数尤度を取り出す logLik()

この関数は総称的である。特定のクラスのオブジェクトに対するメソッド関数を書くことができる。この関数に対するメソッドを既に持つ関数には glm(), lm(), nls(), gls(),lme() そしてパッケージ nlme 中の関数がある。

```
## クラス "lm" に対する S3 メソッド
logLik(object, REML = FALSE, ...)
```

6.11 モデルフレーム model.frame()

総称的関数 model.frame() とそのメソッドは formula を使用するのに必要な変数と任意の追加引数 ... を含むデータフレームを返す。

```
## 既定の S3 メソッド
model.frame(formula, data = NULL,
            subset = NULL, na.action = na.fail,
            drop.unused.levels = FALSE, xlev = NULL, ...)
## クラス "aovlist" に対する S3 メソッド
model.frame(formula, data = NULL, ...)
## クラス "glm" に対する S3 メソッド
model.frame(formula, ...)
## クラス "lm" に対する S3 メソッド
model.frame(formula, ...)
```

6.12 計画行列 model.matrix()

計画行列 (design matrix) を作る。

```
## 既定の S3 メソッド:
model.matrix(object, data = environment(object),
             contrasts.arg = NULL, xlev = NULL, ...)
```

6.13 データの線形モデル空間への射影 proj()

データを線形モデルの項の空間へ射影した行列もしくは行列のリストを返す。aov() モデルで最もしばしば使われる。

```
## クラス "aov" に対する S3 メソッド
proj(object, onedf = FALSE, unweighted.scale = FALSE, ...)
## クラス "aovlist" に対する S3 メソッド
proj(object, onedf = FALSE, unweighted.scale = FALSE, ...)
## 既定の S3 メソッド
proj(object, onedf = TRUE, ...)
## クラス "lm" に対する S3 メソッド
proj(object, onedf = FALSE, unweighted.scale = FALSE, ...)
```

6.14 当てはめモデルの共分散行列を計算する vcov()

当てはめモデルオブジェクトの主要パラメータの分散共分散行列を計算する。

```
vcov(object, ...)
```

6.15 モデル式 formula()

総称的関数 formula() とその個別のメソッドは、他のオブジェクトに含まれているモデル式を取り出す。as.formula() はほとんど同一であるが object が既に”formula” 属性を継承している際に、属性を保存する。env 引数の既定値はモデル式が環境を欠いている場合にだけ使われる。

```
formula(x, ...)
as.formula(object, env = parent.frame())
```

6.16 パターンによる因子水準 gl()

パターンを与えて因子の水準を作成する。

```
gl(n, k, length = n*k, labels = 1:n, ordered = FALSE)
```

6.17 モデル式にオフセット項を含める offset()

オフセットとは一般化線形モデルに於けるような線形予測子に加えられる項で、係数は既知の係数 1 を持ち、推定されることは無い。

```
offset(object)
```

6.18 そのまま関数 I()

I() はオブジェクトが、文字通りの意味で解釈されるべきことを指示する様にクラスを変更する。

```
I(x)
```

6.19 モデル項 terms()

関数 terms() は総称的関数で、様々な R データオブジェクトから ”terms” オブジェクトを取り出すことができる。

```
terms(x, ...)
```

6.20 射影追跡回帰 ppr()

射影追跡回帰モデル (projection pursuit regression model) を当てはめる。

```
ppr(x, ...)
## クラス "formula" に対する S3 メソッド
ppr(formula, data, weights, subset, na.action,
    contrasts = NULL, ..., model = FALSE)
## 既定の S3 メソッド
ppr(x, y, weights = rep(1,n),
    ww = rep(1,q), nterms, max.terms = nterms, optlevel = 2,
    sm.method = c("supsmu", "spline", "gcvspline"),
    bass = 0, span = 0, df = 5, gcvpen = 1, ...)
```

6.21 モデルフレームの変数のタイプを検査する .checkMFClasses()

.checkMFClasses() は予測メソッドで使われる変数が、当てはめで使われたそれらとタイプが一致するかどうかを検査する。.MFclass() はこのために変数を類別する。

```
.checkMFClasses(cl, m, ordNotOK = FALSE)
.MFclass(x)
.getXlevels(Terms, m)
```

6.22 モデルの係数を取り出す coef()

coef() はモデリング関数が返すオブジェクトからモデル係数を取り出すための総称的関数である。coefficients() はその別称である。

```
coef(object, ...)
coefficients(object, ...)
```

6.23 モデルパラメータの信頼区間 confint()

当てはめモデルの一つもしくは複数のパラメータに対する信頼区間を計算する。クラス "lm" を継承するオブジェクトに対するメソッドを持つ。

```
confint(object, parm, level = 0.95, ...)
```

6.24 モデルの逸脱度 deviance()

当てはめモデルの逸脱度 (deviance) を返す。

```
deviance(object, ...)
```

6.25 モデルフレームに新しい変数を加える expand.model.frame()

新しい変数を、それが指定モデルのモデル式の一部であるかのように評価する。同じ na.action と subset 引数が適用されることが保証され、例えば sin(x) を予測子に使ったモデルに対し x がモデルから復元できることを保証する。

```
expand.model.frame(model, extras, envir=environment(formula(model)),
                   na.expand = FALSE)
```

6.26 AIC 規準 AIC()

式 -2 log(-尤度) + k x npar により、尤度が計算できるような一つもしくは複数の当てはめモデルオブジェクトに対し赤池の情報量規準を計算する総称的関数である。ここで npar は当てはめモデル中のパラメータ数で、k=2 なら通常の AIC、k=log n (n は観測値数) ならばいわゆる BIC または SBC (Schwarz のベイズ規準) となる。

```
AIC(object, ..., k = 2)
```

6.27 当てはめモデルの AIC 規準を計算 extractAIC()

当てはめパラメトリックモデルに対する (一般化) 赤池情報量規準 AIC を計算する。

```
extractAIC(fit, scale, k = 2, ...)
```

6.28 モデル当てはめ値を取り出す fitted()

fitted() はモデリング関数が返すオブジェクトから当てはめ値を取り出す総称的関数である。fitted.values() はその別称である。モデリング関数が返す全てのオブジェクトクラスは fitted メソッドを持つべきである。(総称的関数は fitted() であり fitted.values() でないことを注意。) メソッドは欠損値の除外を補整する napredict メソッドを利用することができる。既定では lm と glm メソッドが利用している。

```
fitted(object, ...)
fitted.values(object, ...)
```

6.29 モデルが空かどうか検査する is.empty.model()

R のモデル式表記は切片項も説明変数もないモデルを許す。これは内部的に特殊な扱いを必要とする。is.empty.model() はオブジェクトが空のモデルかどうかを検査する。

```
is.empty.model(x)
```

6.30 model.tables を作る make.tables.aovproj()

これらは model.tables() (のメソッド関数) の支援関数であり、おそらくそれ以外の使い途は無い。

```
make.tables.aovproj(proj.cols, mf.cols, prjs, mf,
                    fun = "mean", prt = FALSE, ...)
make.tables.aovprojlist(proj.cols, strata.cols, model.cols, projections,
                        model, eff, fun = "mean", prt = FALSE, ...)
```

6.31 モデルフレームから成分を取り出す model.extract()

model.frame の目的変数、オフセット、部分集合、重み、もしくはオプション引数として引き渡されたモデルフレームの他の特殊な成分を返す。

```
model.extract(frame, component)
model.offset(x)
model.response(data, type = "any")
model.weights(x)
```

6.32 分散分析モデルの結果から表を計算する model.tables()

モデル当てはめ、特に複雑な aov 当てはめ、に対する要約表を計算する。

```
model.tables(x, ...)
## クラス "aov" に対する S3 メソッド
model.tables(x, type = "effects", se = FALSE, cterms, ...)
## クラス "aovlist" に対する S3 メソッド
    model.tables(x, type = "effects", se = FALSE, ...)
```

6.33 予測 predict()

predict() は様々なモデル当てはめ関数の結果から、予測を行う総称的関数である。この関数は最初の引数のクラスに依存する、特定のメソッド関数を呼び出す。

```
predict (object, ...)
```

6.34 プロファイリングモデルに対する総称的関数 profile()

fitted で表される解の近くでの目的関数の挙動を調べる。これ以上の詳細はメソッド関数のドキュメントを見よ。

```
profile(fitted, ...)
```

6.35 非線形モデルに対するプロファイラオブジェクト profiler()

モデルオブジェクト fitted に対するプロファイラオブジェクトを作る。

```
profiler(fitted, ...)
```

6.36 モデルの射影 proj()

データの線形モデル項の上への射影を与える行列もしくは行列のリストを返す。これは最もしばしば aov モデルで使われる。

```
proj(object, ...)
## クラス "aov" に対する S3 メソッド
proj(object, onedf = FALSE, unweighted.scale = FALSE, ...)
## クラス "aovlist" に対する S3 メソッド
proj(object, onedf = FALSE, unweighted.scale = FALSE, ...)
## 既定の S3 メソッド
proj(object, onedf = TRUE, ...)
## クラス "lm" に対する S3 メソッド
proj(object, onedf = FALSE, unweighted.scale = FALSE, ...)
```

6.37 モデルの残差を取り出す residuals()

モデリング関数が返すオブジェクトから、モデル残差を取り出す総称的関数である。省略形 resid() は residuals() の別名である。これはユーザが、オブジェクトのスロットを直接参照すること無く、アクセス関数でオブジェクト成分にアクセスすることを推奨することを企図している。モデリング関数が返す全てのオブジェクトクラスは residuals メソッドを持つべきである。(メソッドは residuals() であり resid() ではないことを注意しよう。) 既定のメソッドがそうであるように、メソッドは欠損値の除外を補整するために naresid メソッドを使うことができる。

```
residuals(object, ...)
resid(object, ...)
```

6.38 モデル項の対比の標準誤差 se.contrast()

aov オブジェクト中の一つ、もしくは複数の対比の標準誤差を返す。

```
se.contrast(object, ...)
## クラス "aov" に対する S3 メソッド
se.contrast(object, contrast.obj,
            coef = contr.helmert(ncol(contrast))[, 1],
            data = NULL, ...)
```

6.39 変数増減による AIC モデル選択 step()

変数をステップ毎に増減し、AIC 規準でモデルを選択する。

```
step(object, scope, scale = 0,
     direction = c("both", "backward", "forward"),
     trace = 1, keep = NULL, steps = 1000, k = 2, ...)
```

6.40 モデルの更新と再当てはめ update()

update() はモデルを更新し (既定では) 再当てはめする。これは、オブジェクト中に保管された呼出し記録を取り出し、それを更新し、(既定では) その呼出しを評価する。例えばデータフレームが強制変換されてしまったときなど、update() を一つの引数だけで呼び出すことが役に立つこともある。

```
update(object, ...)
## 既定の S3 method:
update(object, formula., ..., evaluate = TRUE)
```

6.41 モデルの更新 update.formula()

update.formula() はモデル式を更新するのに使われる。これは典型的には項を加えたり、取り除いたりすることを意味するが、しかし更新はもっと一般的であり得る。

```
## クラス "formula" に対する S3 メソッド
update(old, new, ...)
```

6.42 片側モデル式への変換 asOneSidedFormula()

名前、表現式、数値 そして文字列を片側モデル式に変換する。もし object がモデル式なら、それは片側式でなければならず、そしてそのまま返される。

```
asOneSidedFormula(object)
```

6.43 term オブジェクトを修正する delete.response()

delete.response() は同じモデルに対する目的変数を持たない terms オブジェクトを返す。drop.terms() はモデルの右辺から変数を取り除く。同じことを (keep.response=TRUE 付きで) 行う "[.terms" メソッドもある。reformulate() は文字列からモデル式を作る。

```
delete.response(termobj)
reformulate(termlabels, response = NULL)
drop.terms(termobj, dropx = NULL, keep.response = FALSE)
```

6.44 モデル式に追加・除去可能な項を計算 factor.scope()

add.scope() と drop.scope() はモデルから、項の階層性を保ちながら、個別に追加・除去可能な項を計算する。

```
add.scope(terms1, terms2)
drop.scope(terms1, terms2)
factor.scope(factor, scope)
```

6.45 モデル式から term オブジェクトを作る terms.formula()

この関数はモデル式をと幾つかのオプションを引数に取り、term オブジェクトを作る。term オブジェクトは model.matrix を作るのに使うことができる。

```
## クラス "formula" に対する S3 メソッド
terms(x, specials = NULL, abb = NULL, data = NULL, neg.out = TRUE,
      keep.order = FALSE, simplify = FALSE, ...)
```

6.46 線形モデル当てはめ要約 lm.summaries()

これら全ての関数はクラス ”lm” のオブジェクトに対するメソッドである。線形モデル当てはめ情報を取り出す。

```
## クラス lm に対する S3 メソッド
family(object, ...)
## クラス lm に対する S3 メソッド
formula(x, ...)
## クラス lm に対する S3 メソッド
residuals(object, type=c("working","response","deviance","pearson","partial"), ...)
weights(object, ...)
```

6.47 安全な予測用の補助関数 makepredictcall()

model.frame.default() を支援する補助関数 makepredictcall() は、poly や ns といった項を持つモデルから予測を行う際の正しい行列を作る。

```
makepredictcall(var, call)
```

6.48 直交多項式 poly(), polym()

指定された点集合 x 上の、次数 1 から degree までの直交多項式を返す。これらは全て次数 0 の定数値多項式とも直交する。

```
poly(x, ..., degree = 1, coefs = NULL)
polym(..., degree = 1)
## クラス "poly" に対するメソッド
predict(object, newdata, ...)
```

6.49 因子へのコード化 factor()

関数 factor() はベクトルを因子にコード化する (名前カテゴリと列挙タイプもまた因子として使用される)。もし ordered = TRUE なら因子水準は順序を持つと仮定される。S との互換性のために、関数 ordered() もある。is.factor(), is.ordered(), as.factor() そして as.ordered() はこれらのクラスの所属関数と強制変換関数である。

```
factor(x, levels = sort(unique.default(x), na.last = TRUE),
       labels = levels, exclude = NA, ordered = is.ordered(x))
ordered(x, ...)
is.factor(x)
is.ordered(x)
as.factor(x)
as.ordered(x)
```