RjpWiki アーカイブス

【R の統計解析関数 Tips (06.06.27) 】

R に関する Tips で欠けているもの、そしておそらく一番困難 (統計学本来の知識が不可欠) なものは R の本来の目的である統計解析機能に関する Tips でしょう。ここには気がついた有用な Tips をメモ代わりに紹介します。

1 lm 関数を用いた線形モデルの当てはめ結果の要約 (r-help の S. Graves の記事、2003.12.31)

```
> df1 <- data.frame(x=1:6, y=rep(1:3, 2))
> fit <- lm(y~x, df1)
> Sum <- summary(fit)
# 当てはめオブジェクト fit の要約 (普通ユーザーはこの情報だけで十分)
> Sum

Call:
lm(formula = y ~ x, data = df1)

Residuals:
      1       2       3       4       5       6
-0.4286  0.3429  1.1143 -1.1143 -0.3429  0.4286

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.2000     0.8177   1.468    0.216
x             0.2286     0.2100   1.089    0.338

Residual standard error: 0.8783 on 4 degrees of freedom
Multiple R-Squared: 0.2286,     Adjusted R-squared: 0.03571
F-statistic: 1.185 on 1 and 4 DF,  p-value: 0.3375
# 当てはめオブジェクトの要約が持つ情報 (リスト形式になっている) 一覧
> attributes(Sum)
$names
[1] "call"          "terms"         "residuals"     "coefficients"
[5] "sigma"         "df"            "r.squared"     "adj.r.squared"
[9] "fstatistic"    "cov.unscaled"
# 回帰係数と関連情報を表形式で得る (summary(fit) が表示する情報の一部!)
```

```
> coefficients(Sum)    # coefficients 関数は線形回帰当てはめオブジェクト
                       から回帰係数を取り出す

            Estimate Std. Error  t value  Pr(>|t|)
(Intercept) 1.2000000  0.8176622 1.467599 0.2161194
x           0.2285714  0.2099563 1.088662 0.3375019

> Sum$coefficients  # これでも良い
             Estimate Std. Error  t value  Pr(>|t|)
(Intercept) 1.2000000  0.8176622 1.467599 0.2161194
x           0.2285714  0.2099563 1.088662 0.3375019
```

参考までに述べると lm 関数が返す当てはめオブジェクトは実際は次のような大量の情報を所有している。これらから本当に必要な情報を抜き出すために summary 関数等が必要になる。もちろんこれらはユーザーには直接必要なものではないが、当てはめ結果を用いたその後の二次的解析等で用いられる。自分でそうした二次解析用の関数を書くためには、これらの内部情報を適当に取り出し利用することになる。S/R の最大のメリットの一つが、多様な二次的解析を可能にするこうした解析結果の詳細情報の保持であるが、一方でともかく結果だけが欲しいユーザーにとっつきにくい印象を与えるのは致し方ない。

```
> str(fit)  # str 関数で当てはめオブジェクトの内容を見る
List of 12
 $ coefficients : Named num [1:2] 1.200 0.229
  ..- attr(*, "names")= chr [1:2] "(Intercept)" "x"
 $ residuals    : Named num [1:6] -0.429  0.343  1.114 -1.114 -0.343 ...
  ..- attr(*, "names")= chr [1:6] "1" "2" "3" "4" ...
 $ effects      : Named num [1:6] -4.89898  0.95618  1.25970 -0.87467 -0.00904 ...
  ..- attr(*, "names")= chr [1:6] "(Intercept)" "x" "" "" ...
 $ rank         : int 2
 $ fitted.values: Named num [1:6] 1.43 1.66 1.89 2.11 2.34 ...
  ..- attr(*, "names")= chr [1:6] "1" "2" "3" "4" ...
 $ assign       : int [1:2] 0 1
 $ qr           :List of 5
  ..$ qr   : num [1:6, 1:2] -2.449  0.408  0.408  0.408  0.408 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:6] "1" "2" "3" "4" ...
  .. .. ..$ : chr [1:2] "(Intercept)" "x"
  .. ..- attr(*, "assign")= int [1:2] 0 1
  ..$ qraux: num [1:2] 1.41 1.19
  ..$ pivot: int [1:2] 1 2
  ..$ tol  : num 1e-07
  ..$ rank : int 2
  ..- attr(*, "class")= chr "qr"
 $ df.residual  : int 4
 $ xlevels      : list()
```

```
 $ call         : language lm(formula = y ~ x, data = df1)
 $ terms        :Classes 'terms', 'formula' length 3 y ~ x
  .. ..- attr(*, "variables")= language list(y, x)
  .. ..- attr(*, "factors")= int [1:2, 1] 0 1
  .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. ..$ : chr [1:2] "y" "x"
  .. .. .. ..$ : chr "x"
  .. ..- attr(*, "term.labels")= chr "x"
  .. ..- attr(*, "order")= int 1
  .. ..- attr(*, "intercept")= int 1
  .. ..- attr(*, "response")= int 1
  .. ..- attr(*, ".Environment")=length 33 <environment>
  .. ..- attr(*, "predvars")= language list(y, x)
 $ model        :'data.frame':  6 obs. of  2 variables:
  ..$ y: int [1:6] 1 2 3 1 2 3
  ..$ x: int [1:6] 1 2 3 4 5 6
  ..- attr(*, "terms")=Classes 'terms', 'formula' length 3 y ~ x
  .. .. ..- attr(*, "variables")= language list(y, x)
  .. .. ..- attr(*, "factors")= int [1:2, 1] 0 1
  .. .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. .. ..$ : chr [1:2] "y" "x"
  .. .. .. .. ..$ : chr "x"
  .. .. ..- attr(*, "term.labels")= chr "x"
  .. .. ..- attr(*, "order")= int 1
  .. .. ..- attr(*, "intercept")= int 1
  .. .. ..- attr(*, "response")= int 1
  .. .. ..- attr(*, ".Environment")=length 33 <environment>
  .. .. ..- attr(*, "predvars")= language list(y, x)
 - attr(*, "class")= chr "lm"
> str(Sum)  # Sum の持つ内部情報の詳細を見る
List of 11
 $ call         : language lm(formula = y ~ x, data = df1)
 $ terms        :Classes 'terms', 'formula' length 3 y ~ x
  .. ..- attr(*, "variables")= language list(y, x)
  .. ..- attr(*, "factors")= int [1:2, 1] 0 1
  .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. ..$ : chr [1:2] "y" "x"
  .. .. .. ..$ : chr "x"
  .. ..- attr(*, "term.labels")= chr "x"
  .. ..- attr(*, "order")= int 1
  .. ..- attr(*, "intercept")= int 1
  .. ..- attr(*, "response")= int 1
  .. ..- attr(*, ".Environment")=length 34 <environment>
  .. ..- attr(*, "predvars")= language list(y, x)
```

```
 $ residuals    : Named num [1:6] -0.429  0.343  1.114 -1.114 -0.343 ...
  ..- attr(*, "names")= chr [1:6] "1" "2" "3" "4" ...
 $ coefficients : num [1:2, 1:4] 1.200 0.229 0.818 0.210 1.468 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:2] "(Intercept)" "x"
  .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
 $ aliased      : Named logi [1:2] FALSE FALSE
  ..- attr(*, "names")= chr [1:2] "(Intercept)" "x"
 $ sigma        : num 0.878
 $ df           : int [1:3] 2 4 2
 $ r.squared    : num 0.229
 $ adj.r.squared: num 0.0357
 $ fstatistic   : Named num [1:3] 1.19 1.00 4.00
  ..- attr(*, "names")= chr [1:3] "value" "numdf" "dendf"
 $ cov.unscaled : num [1:2, 1:2]  0.8667 -0.2000 -0.2000  0.0571
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:2] "(Intercept)" "x"
  .. ..$ : chr [1:2] "(Intercept)" "x"
 - attr(*, "class")= chr "summary.lm"
# 当てはめオブジェクト fit に基づく二次的解析の例 (分散分析)
> aovfit <- aov(fit)
> aovfit
Call:
   aov(formula = fit)

Terms:
                        x Residuals
Sum of Squares  0.9142857 3.0857143
Deg. of Freedom         1         4

Residual standard error: 0.87831
Estimated effects may be unbalanced

> summary(aovfit)
            Df  Sum Sq Mean Sq F value Pr(>F)
x            1 0.91429 0.91429  1.1852 0.3375
Residuals    4 3.08571 0.77143

> str(aovfit)
List of 12
 $ coefficients : Named num [1:2] 1.200 0.229
  ..- attr(*, "names")= chr [1:2] "(Intercept)" "x"
 $ residuals    : Named num [1:6] -0.429  0.343  1.114 -1.114 -0.343 ...
  ..- attr(*, "names")= chr [1:6] "1" "2" "3" "4" ...
```

```
$ effects      : Named num [1:6] -4.89898  0.95618  1.25970 -0.87467  -0.00904 ...
..- attr(*, "names")= chr [1:6] "(Intercept)" "x" "" "" ...
$ rank         : int 2
$ fitted.values: Named num [1:6] 1.43 1.66 1.89 2.11 2.34 ...
 ..- attr(*, "names")= chr [1:6] "1" "2" "3" "4" ...
$ assign       : int [1:2] 0 1
$ qr           :List of 5
 ..$ qr   : num [1:6, 1:2] -2.449  0.408  0.408  0.408  0.408 ...
 .. ..- attr(*, "dimnames")=List of 2
 .. .. ..$ : chr [1:6] "1" "2" "3" "4" ...
 .. .. ..$ : chr [1:2] "(Intercept)" "x"
 .. ..- attr(*, "assign")= int [1:2] 0 1
 ..$ qraux: num [1:2] 1.41 1.19
 ..$ pivot: int [1:2] 1 2
 ..$ tol  : num 1e-07
 ..$ rank : int 2
 ..- attr(*, "class")= chr "qr"
$ df.residual  : int 4
$ xlevels      : list()
$ call         : language aov(formula = fit)
$ terms        :Classes 'terms', 'formula' length 3 y ~ x
 .. ..- attr(*, "variables")= language list(y, x)
 .. ..- attr(*, "factors")= int [1:2, 1] 0 1
 .. .. ..- attr(*, "dimnames")=List of 2
 .. .. .. ..$ : chr [1:2] "y" "x"
 .. .. .. ..$ : chr "x"
 .. ..- attr(*, "term.labels")= chr "x"
 .. ..- attr(*, "order")= int 1
 .. ..- attr(*, "intercept")= int 1
 .. ..- attr(*, "response")= int 1
 .. ..- attr(*, ".Environment")=length 35 <environment>
 .. ..- attr(*, "predvars")= language list(y, x)
$ model        :'data.frame':  6 obs. of  2 variables:
 ..$ y: int [1:6] 1 2 3 1 2 3
 ..$ x: int [1:6] 1 2 3 4 5 6
 ..- attr(*, "terms")=Classes 'terms', 'formula' length 3 y ~ x
 .. .. ..- attr(*, "variables")= language list(y, x)
 .. .. ..- attr(*, "factors")= int [1:2, 1] 0 1
 .. .. .. ..- attr(*, "dimnames")=List of 2
 .. .. .. .. ..$ : chr [1:2] "y" "x"
 .. .. .. .. ..$ : chr "x"
 .. .. ..- attr(*, "term.labels")= chr "x"
 .. .. ..- attr(*, "order")= int 1
 .. .. ..- attr(*, "intercept")= int 1
```

```
  .. .. ..- attr(*, "response")= int 1
  .. .. ..- attr(*, ".Environment")=length 35 <environment>
  .. .. ..- attr(*, "predvars")= language list(y, x)
 - attr(*, "class")= chr [1:2] "aov" "lm"
```

2　線形モデル当てはめ関数におけるモデル公式 (2004.01.05)

lm (線形回帰)、aov (分散分析)、glm (一般化線形回帰) 等の線形モデル当てはめ関数では、目的変数と (複数の) 説明変数・因子の組合せからなる当てはめモデル式をモデル公式 (model formula) と呼ばれる簡略記法で表現する。基本的な記法は

`y ~ ...`	左辺の目的変数を、右辺の説明変数の線形式で線形回帰
`+ A`	説明変数 `A` を加える
`- A`	説明変数 `A` を除く
`+ 0` 又は `- 1`	切片 (定数) 項の除外
`A:B`	二つの説明変数 `A, B` の交互作用項 (`A` と `B` の積からなる説明変数)
`A * B`	同上
`(...)*(...)`	右括弧内の変数と左括弧内の変数の掛け合わせ (交互作用) すべての組合せを表す. ただし `A^2, A^2*B` 等はそれぞれ `A, A*B` と解釈
`(A+B)^2`	`(A+B)*(A+B)` と同じ。`A + B + A:B` の意味
`A %in% B`	今一意味がわからない?

より一般に A:B:C や (A+B+C)3 といった高次交互作用項も同様に表現される。y を目的変数、A, B, C 等を説明変数・因子とすると

`y ~ A`	`y` を `A` と切片 (定数) 項で単回帰
`y ~ A - 1`	切片項のない、原点を通る単回帰
`y ~ A + 0`	同上
`0 + y ~ A`	同上
`y ~ A + B`	`y` を `A, B` と切片 (定数) 項で重回帰
`y ~ A + B - 1`	`y` を `A, B` で重回帰。切片 (定数) 項は含まない
`y ~ A + B + A:B`	`A,B` とそれらの二次交互作用項で重回帰
`y ~ (A+B)^2`	同上
`y ~ (A+B)^2 - 1`	`A,B` とそれらの二次交互作用項で重回帰。切片 (定数) 項は含まない
`y ~ (A+B+C)^2`	`A,B,C` とそれらのすべての二次交互作用項で重回帰
`y ~ A + B + C + A:B + A:C + B:C`	同上
`y ~ (A+B+C)^2 - B:C`	`y ~ A + B + C + A:B + A:C` の意味
`y ~ (A+B+C)^3`	`A,B,C` とそれらのすべての二次及び三次交互作用項で重回帰
`y ~ A + A %in% B`	`y ~ A + A:B` の意味

モデル式には説明変数の関数を使うことが出来る。

```
y ~ A + log(B)            y を A と B の対数値で説明
log(y) ~ exp(A) + log(B)  log(y) を exp(A) と log(B) で説明
y ~ A/B                   y を A と B の比で説明
```

モデル式で特別の意味を持つオペレータ (例えば +,-,*,^) を、この意味の説明変数の変形と区別するため特殊記号 I() を用いる。I() の中では +,-,*,^ 等は数値演算子と解釈される。

```
y ~ A + I(B+C)  y を A と B+C (B と C の和) で説明
y ~ A + I(B*C)  y を A と B*C (B と C の積) で説明、A + B:C と同じ
y ~ A + I(B^2)  y を A と B^2 (B の自乗値)  で説明
```

3 非線形最小自乗法 nls() 関数

非線形モデルの非線形最小自乗推定専用の関数で、独自の最適化アルゴリズム (二種類) を備える。クラス nls のオブジェクトを返す。

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

```
引数
formula     変量とパラメータを含む非線形モデル公式
data        オプションのデータフレームで、その中で formula
start       初期推定値の名前付きリスト、もしくは名前付き数値ベクトル。
control     制御仕様を決めるオプションのリスト。設定可能な制御値の名前と、
            それらの効果に付いては nls.control を見よ。
algorithm   使用されるアルゴリズムを指定する文字列。 既定の手法は
            Gauss-Newton アルゴリズム。他の選択肢は "plinear" で
            Golub-Pereyra による局所線形最小自乗法のアルゴリズム。
trace       論理値で、繰り返しの過程の記録を表示するかどうかを指定する。
            既定値は FALSE。もし TRUE ならば、各繰り返し毎に残差平方と
            パラメータ値が 出力される。もし "plinear" アルゴリズムが
            使われると、非線形パラメータの後に線形パラメータの条件付き
            推定値が出力される。
subset      オプションのベクトルで、当てはめの過程で用いられる観測値の
            部分集合を指定する。
weights     オプションの数値ベクトルで、(固定された) 重みを与える。もしこれが
            あれば、目的値は重み付き自乗和となる。現在移植されていない。
na.action   データが NA を含む時に、どうするかを指示する関数。
```

nls を人工的な ”残差がゼロ” のデータに使ってはならない。 nls 関数は、現在のパラメータ推定値の精度と残差平方和を比較するのに relative-offset 収束判定基準を使う。 これは、丸め誤差の二つの成分を比較することにより、y = f(x, theta) + eps の形 (var(eps) ¿ 0) のデータに対してうまく働く。 もし nls を人工的なデータに適用する際は、以下の例に示したように、ノイズを加えて欲しい。nls オブジェクトは典型的には当てはめモデルオブジェクトである。これは総称的関数 coef, formula, resid, print, summary, AIC, fitted そして vcov に対するメソッドを持つ。

返り値：次の成分からなるリスト
`m` モデルを含む `nlsModel` オブジェクト
`data` データ引数として `nls` に引き渡された公式。 実際のデータ値は `m` 成分の環境中にある。

4 例 1

データフレーム DNase は成分 Run (順序つき因子, 二因子)、conc (数値)、density (数値) を持つ。

```
data( DNase )
DNase1 <- DNase[ DNase$Run == 1, ] # データの一部を取り出す
## 自動スタートモデルを使う。SSlogis はロジスティックモデル関数を作る R 関数
fm1DNase1 <- nls( density ~ SSlogis( log(conc), Asym, xmid, scal ), DNase1 )

summary( fm1DNase1 ) # 当てはめ結果の要約
Formula: density ~ SSlogis(log(conc), Asym, xmid, scal) # 非線形モデル公式
Parameters: # パラメータ推定値、標準誤差、t 値、p 値
     Estimate Std. Error t value Pr(>|t|)
Asym  2.34518    0.07815   30.01 2.17e-13 ***
xmid  1.48309    0.08135   18.23 1.22e-10 ***
scal  1.04146    0.03227   32.27 8.51e-14 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 0.01919 on 13 degrees of freedom
Correlation of Parameter Estimates: 推定パラメータの相関値
       Asym   xmid
xmid 0.9868
scal 0.9008 0.9063
## 条件付き線形性を使う
fm2DNase1 <- nls( density ~ 1/(1 + exp(( xmid - log(conc) )/scal ) ),
                  data = DNase1,
                  start = list( xmid = 0, scal = 1 ),
                  alg = "plinear", trace = TRUE )
0.7139315 : 0.000000 1.000000 1.453853  # 当てはめ途中結果のレポート
0.1445295 : 1.640243 1.390186 2.461754  # 残差平方和と推定値
```

```
0.008302151 : 1.620899 1.054228 2.478388
0.004794192 : 1.485226 1.043709 2.347334
0.004789569 : 1.483130 1.041468 2.345218
0.004789569 : 1.483090 1.041455 2.345180

summary( fm2DNase1 ) # 当てはめ結果の要約
Formula: density ~ 1/(1 + exp((xmid - log(conc))/scal)) # 非線形モデル公式
Parameters: # パラメータ推定値、標準誤差、t 値、p 値
     Estimate Std. Error t value Pr(>|t|)
xmid  1.48309    0.08135   18.23 1.22e-10 ***
scal  1.04145    0.03227   32.27 8.51e-14 ***
.lin  2.34518    0.07815   30.01 2.17e-13 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 0.01919 on 13 degrees of freedom
Correlation of Parameter Estimates: 推定パラメータの相関値
       xmid   scal
scal 0.9063
.lin 0.9868 0.9008
## 条件付き線形性を使わない
fm3DNase1 <- nls( density ~ Asym/(1 + exp(( xmid - log(conc) )/scal ) ),
                  data = DNase1,
                  start = list( Asym = 3, xmid = 0, scal = 1 ),
                  trace = TRUE )
14.32279 :  3 0 1 # 当てはめ途中結果のレポート
0.4542698 :  2.1152456 0.8410193 1.2000640 # 残差平方和と推定値
0.05869602 :  2.446376 1.747516 1.189515
0.005663523 :  2.294087 1.412198 1.020463
0.004791528 :  2.341429 1.479688 1.040758
0.004789569 :  2.345135 1.483047 1.041439
0.004789569 :  2.345179 1.483089 1.041454

summary(fm3DNase1) # 当てはめ結果の要約
Formula: density ~ Asym/(1 + exp((xmid - log(conc))/scal)) # 非線形モデル公式
Parameters: # パラメータ推定値、標準誤差、t 値、p 値
     Estimate Std. Error t value Pr(>|t|)
Asym  2.34518    0.07815   30.01 2.17e-13 ***
xmid  1.48309    0.08135   18.23 1.22e-10 ***
scal  1.04145    0.03227   32.27 8.51e-14 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
Residual standard error: 0.01919 on 13 degrees of freedom
Correlation of Parameter Estimates: 推定パラメータの相関値
       Asym   xmid
```

```
 xmid 0.9868
 scal 0.9008 0.9063


\section{例２ 重み付き非線形回帰 (非線形モデルを関数で与える)}

データ Puromycin は成分 conc (数値), rate (数値), state (因子、二水準) を持つデータフレーム

\begin{breakbox}
\begin{verbatim}
 data(Puromycin)
 Treated <- Puromycin[Puromycin$state == "treated", ]
 weighted.MM <- function(resp, conc, Vm, K) # 非線形モデルを与える関数の定義
 { ## 目的: 白本の p.451 -- RHS にある Michaelis-Menten モデルに対する nls() の重み付き版
    pred <- (Vm * conc)/(K + conc)
    (resp - pred) / sqrt(pred)  # sqrt(pred) で割り重みとする
 }
 # Vm, K がモデルパラメータ、rate および conc はデータフレームの成分変数名
 Pur.wt <- nls(~weighted.MM(rate, conc, Vm, K), data = Treated,
               start = list(Vm = 200, K = 0.1), trace = TRUE)
 112.5978 :  200.0   0.1
 17.33824 :  205.67588840   0.04692873
 14.6097 :  206.33087396   0.05387279
 14.59694 :  206.79883508   0.05457132
 14.59690 :  206.83291286   0.05460917
 14.59690 :  206.83468191   0.05461109

 >   summary(Pur.wt)
 Formula: 0 ~ weighted.MM(rate, conc, Vm, K) # 関数 weighted.MM の値を目的変数とする
 Parameters:
     Estimate Std. Error t value Pr(>|t|)
 Vm 2.068e+02  9.225e+00  22.421 7.00e-10 ***
 K  5.461e-02  7.979e-03   6.845 4.49e-05 ***
 ---
 Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
 Residual standard error: 1.208 on 10 degrees of freedom
 Correlation of Parameter Estimates:
       Vm
 K 0.7543
```

5 一次元分布に対する最尤推定。 MASS パッケージ中の fitdistr 関数 (2004.2.16)

書式:
```
fitdistr(x, densfun, start, ...)
```

引数

x	数値ベクトル
densfun	密度関数名を与える文字列か関数 (第一引数で評価される)。文字列で指定できる分布は '"beta"', '"cauchy"', '"chi-squared"', '"exponential"', '"f"', '"gamma"', '"log-normal"', '"lognormal"', '"logistic"', '"negative binomial"', '"normal"', '"t"', '"uniform"' そして '"weibull"'。
start	初期値を含む最適化されるべきパラメータの名前つきリスト。名前で指定できる分布の幾つかについては省略可能。
...	'densfun' もしくは 'optim' に引き渡される追加引数。'lower' もしくは 'upper' (の一方もしくは双方) で、パラメータの上・下界を指定できる。もし 'densfun' (もしくは名前で指定可能な密度関数) の引数が含まれれば、その値が固定される。

対数尤度の数値微分値を用いた直接的最大化が実行される。以下の名前指定分布では、もし 'start' が省略されたり、部分的にしか与えられない時は、合理的な初期値が設定される。'cauchy', 'gamma', 'logistic', 'negative binomial' ('mu' と 'size' がパラメータ), 't', 'uniform', 'weibull'. 再尤推定量が陽に書き下せる時は、それを計算するだけで、数値的最適化を行なうわけではない。

返り値:クラス '"fitdistr"' のオブジェクトで次の二つの成分を持つ

estimate	推定パラメータ
sd	推定標準偏差

6 例

```
> set.seed(123)  # 乱数種
> x <- rgamma(100, shape = 5, rate = 0.1)
> fitdistr(x, "gamma")
     shape         rate
  6.48592894   0.13649147
 (0.89425721) (0.01956555)
> fitdistr(x, dgamma, list(shape = 1, rate = 0.1), lower = 0.01)
      shape         rate
  2.156761461   0.010000000
```

```
  (0.277607887) (0.001433778)
 > set.seed(123) # 乱数種
 > x2 <- rt(250, df = 9)                # t 分布に従う乱数を 250 個発生
 > fitdistr(x2, "t", df = 9)            # 最尤推定による自由度 9 の t 分布の当てはめ
         m             s                # 二つのパラメータの推定値と推定標準偏差
   -0.01080467    1.04402833
  ( 0.07222206) ( 0.05433544)
 > fitdistr(x2, "t")                              # 自由度も未値パラメータとする
         m             s             df          # 三つのパラメータの推定値と推定標準偏差
   -0.009754149    1.006270714    6.632753559
  ( 0.071474570) ( 0.077107965) ( 2.716093153)
 > mydt <- function(x, m, s, df) dt((x - m)/s, df)/s # 密度関数を独自定義
 > fitdistr(x2, mydt, list(m = 0, s = 1), df = 9, lower = c(-Inf, 0))
         m             s           # 二つのパラメータの推定値と推定標準偏差
   -0.01069635    1.04409435
  ( 0.07222562) ( 0.05434249)
 > set.seed(123) # 乱数種
 > x3 <- rweibull(100, shape = 4, scale = 100) # ワイブル分布に従う乱数を 100 個発生
 > fitdistr(x3, "weibull")  # ワイブル分布の当てはめ
      shape        scale           # 二つのパラメータの推定値と推定標準偏差
    4.0807404   99.9812022
  ( 0.3127017) ( 2.5816389)
 > set.seed(123) # 乱数種
 > x4 <- rnegbin(500, mu = 5, theta = 4) # 負の二項分布に従う乱数 500 個発生
 > fitdistr(x4, "Negative Binomial")  # 負の二項分布を当てはめ
      size         mu                 # 二つのパラメータの推定値と推定標準偏差
   4.2178713   4.9439803
  (0.5048242) (0.1465538)
 Warning messages: # 関連警告
 1: NaNs produced in: dgamma(x, shape, scale, log)
 2: NaNs produced in: dgamma(x, shape, scale, log)
 3: NaNs produced in: dgamma(x, shape, scale, log)
 4: bounds can only be used with method L-BFGS-B in:
      optim(start, mylogfn, x =  x, hessian = TRUE, ...)
 5: bounds can only be used with method L-BFGS-B in:
      optim(start, myfn, x = x, hessian = TRUE, ...)
 6: NaNs produced in: dweibull(x, shape, scale, log)
 7: NaNs produced in: dweibull(x, shape, scale, log)
```