Package 'FertBoot'

July 18, 2020

Type Package

Title Fertilizer Response Curve Analysis by Bootstrapping Residuals

Version 0.4.0

Maintainer Ting Fung (Ralph) Ma <tingfung.ma@wisc.edu>

Description
Quantify variability (such as confidence interval) of fertilizer response curves and optimum fertilizer rates using bootstrapping residuals with several popular non-linear and linear models.

Imports stats, nls.multstart, simpleboot

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

NeedsCompilation no

Author Ting Fung (Ralph) Ma [cre, aut],
Hannah Francis [aut],
Matt Ruark [ctb]

Repository CRAN

Date/Publication 2020-07-18 09:32:17 UTC

R topics documented:

boot.CI *Bootstrap confidence intervals of mean*

Description

Bootstrap confidence intervals of mean

Usage

```
boot.CI(x, alpha = 0.05, CI.type = "all")
```

Arguments

`x`	a vector of observation
`alpha`	significance level (default: 0.05)
`CI.type`	type of CI required (default: "all")

Value

`boot.CI` return list of confidence intervals of mean (`CI.percent`: percentile, `CI.BC`:bias-corrected and `CI.BCa`: bias-corrected and accelerated)

Examples

```
set.seed(12)
boot.CI(rnorm(1000, mean=0, sd=1), alpha=0.05, CI.type="per") # example of wrong input for type
boot.CI(rnorm(1000, mean=0, sd=1), alpha=0.05, CI.type="all") # require all type
```

boot.resid.linear.plateau *Linear plateau model estimation by bootstrapping residuals*

Description

`boot.resid.linear.plateau` is the core function that implement bootstrapping residuals on quadratic plateau model, which assumes y ~ a + b * (x - c) * (x <= c). Note that this functions may take minutes up to days. Parallel computing could be considered when necessary. We suggest users start with a smaller `B` and moderate `n.start` to see if the bootstrap models can convergence. In general, increasing `n.start` and `plus_minus` may help convergence. For rigorous statistical inference, `B` should be reach order of thousand.

Usage

```
boot.resid.linear.plateau(
  mod,
  data,
  x.range = data.frame(x = seq(0, 280, by = 40)),
  B = 100 - 1,
  plus_minus = 100,
  n.start = 5000,
  print.progress = TRUE
)
```

Arguments

`mod`	a full model list, probably from `f.linear.plateau()`
`data`	data drame with two columns (`x` and `y`)
`x.range`	vector of data.frame with one column for range of N rate of interested for prediction interval
`B`	bootstrap sample size
`plus_minus`	radius of random initial values (default: `100`)
`n.start`	total number of initial points considered (deafult: `1000`)
`print.progress`	logical flag whehter printing progress

Value

`boot.resid.linear.plateau` returns a list of two elements: `result`: matrix with B rows and columns containing bootstrap sample for parameter (`a,b,c`), optimal N and yield (`max_x,max_y`), log-likelihood (`logLik`) and N values of interest; `x.range`: range of x considered for prediction interval (same as `x.range` in vector form)

Examples

```
set.seed(1)
x <- rep(1:300, each=4)
a <- 8; b <- 0.05; c <- 100
y <- a + b * (x - c) * (x <= c) +
    rnorm(length(x), sd=1)
d <- cbind(x,y)

# a converged example:
ans <- f.linear.plateau(d, start=list(a = 7, b = 0.1, c = 150),
    plus_minus=10, n.start=10, msg=FALSE)


ans.boot <- boot.resid.linear.plateau(ans, d, x.range=seq(0,280,by=40),
    B=99, plus_minus = 1e2, n.start=1000, print.progress=TRUE)
```

boot.resid.quad *Fitting quadratic plateau model using mutiple initial vaues*

Description

boot.resid.quad.plateau is the core function that implement bootstrapping residuals on quadratic plateau model, which assumes y = (a + b * x + c *x^2) * (x <= -0.5*b/c) + (a + -b^2/(4 * c)) * (x > -0.5 * b/c). Note that this functions may take minutes up to days. Parallel computing could be considered when necessary. We suggest users start with a smaller B and moderate n.start to see if the bootstrap models can convergence. In general, increasing n.start and plus_minus may help convergence. For rigorous statistical inference, B should be reach order of thousand.

Usage

```
boot.resid.quad(
  mod,
  data,
  x.range = data.frame(x = seq(0, 280, by = 40)),
  B = 100 - 1,
  plus_minus = 10,
  n.start = 20,
  print.progress = TRUE
)
```

Arguments

mod	a full model list, probably from f.quad.plateau()
data	data drame with two columns (x and y)
x.range	vector of data.frame with one column for range of N rate of interested for prediction interval
B	bootstrap sample size
plus_minus	radius of random initial values (default: 100)
n.start	total number of initial points considered (deafult: 1000)
print.progress	logical flag whehter printing progress

Value

boot.resid.quad.plateau returns a list of two elements: result: matrix with B rows and columns containing bootstrap sample for parameter (a,b,c), optimal N and yield (max_x,max_y), log-likelihood (logLik) and N values of interest; x.range: range of x considered for prediction interval (same as x.range in vector form)

Examples

```
set.seed(1)
x <- rep(1:300, each=5)
a <- 8; b <- 0.05; c <- -1e-3
y <- (a + b * x + c *x^2) * (x <= -0.5*b/c) + (a + -b^2/(4 * c)) * (x > -0.5 * b/c) +
    rnorm(length(x), sd=0.1)
d <- cbind(x,y)

ans <- f.quad.plateau(d, start=list(a = 7, b = 0.02, c = 1e-5),
    plus_minus=10, n.start=10, msg=FALSE)


ans.boot <- boot.resid.quad.plateau(ans, d, x.range=seq(0,280,by=40),
    B=1e1, plus_minus = 1e2, n.start=1e3, print.progress=TRUE)
```

---

boot.resid.quad.plateau

Quadratic plateau model estimation by bootstrapping residuals

---

Description

`boot.resid.quad.plateau` is the core function that implement bootstrapping residuals on quadratic plateau model, which assumes y = (a + b * x + c *x^2) * (x <= -0.5*b/c) + (a + -b^2/(4 * c)) * (x > -0.5 * b/c). Note that this functions may take minutes up to days. Parallel computing could be considered when necessary. We suggest users start with a smaller `B` and moderate `n.start` to see if the bootstrap models can convergence. In general, increasing `n.start` and `plus_minus` may help convergence. For rigorous statistical inference, `B` should be reach order of thousand.

Usage

```
boot.resid.quad.plateau(
  mod,
  data,
  x.range = data.frame(x = seq(0, 280, by = 40)),
  B = 100 - 1,
  plus_minus = 100,
  n.start = 5000,
  print.progress = TRUE
)
```

Arguments

`mod`	a full model list, probably from `f.quad.plateau()`
`data`	data drame with two columns (`x` and `y`)
`x.range`	vector of data.frame with one column for range of N rate of interested for prediction interval
`B`	bootstrap sample size
`plus_minus`	radius of random initial values (default: `100`)
`n.start`	total number of initial points considered (deafult: `1000`)
`print.progress`	logical flag whehter printing progress

Value

`boot.resid.quad.plateau` returns a list of two elements: `result`: matrix with B rows and columns containing bootstrap sample for parameter (`a,b,c`), optimal N and yield (`max_x,max_y`), log-likelihood (`logLik`) and N values of interest; `x.range`: range of x considered for prediction interval (same as `x.range` in vector form)

Examples

```
set.seed(1)
x <- rep(1:300, each=5)
a <- 8; b <- 0.05; c <- -1e-4
y <- (a + b * x + c *x^2) * (x <= -0.5*b/c) + (a + -b^2/(4 * c)) * (x > -0.5 * b/c) +
    rnorm(length(x), sd=0.1)
d <- cbind(x,y)

ans <- f.quad.plateau(d, start=list(a = 7, b = 0.02, c = 1e-5),
    plus_minus=10, n.start=10, msg=FALSE)


boot.resid.quad.plateau(ans, d, x.range=seq(0,280,by=40),
    B=1e2-1, plus_minus = 1e2, n.start=1000, print.progress=TRUE)
```

`compare.two.sample` *Two sample bootstrap test for comparing different in* `sample1` *and* `sample2`*, not necessary with same sample size*

Description

Two sample bootstrap test for comparing different in `sample1` and `sample2`, not necessary with same sample size

Usage

```
compare.two.sample(sample1, sample2, fun = mean, R = 1000)
```

Arguments

sample1	first sample
sample2	second sample
fun	statistic (univariate) to be compared (default: mean)
R	number of resample (deafult: 1000)

Value

compare.two.sample return a list with two components, namely, p.value: two tailed p-value for the bootstrap test object: a "simpleboot" object allowing further analysis using other R packages, such as boot)

Examples

```
set.seed(1203)
# compare median of two expontential r.v.
compare.two.sample(rexp(100, rate=1), rexp(100, rate=2), fun=median, R=1e3)$p.value

f.Q1 <- function(x) quantile(x, probs=0.25)
compare.two.sample(rnorm(100, mean=0), rnorm(200, mean=0.5), fun=f.Q1, R=1e3)$p.value
```

---

f.linear.plateau *Fitting linear plateau model using mutiple initial vaues*

Description

f.linear.plateau fits linear plateau model using multiple initial values. The multiple initial values are randomly sampled in a "cube" of parameter space. More precisely, quadratic plateau model assumes y ~ a + b * (x - c) * (x <= c).

Usage

```
f.linear.plateau(
  d,
  start = list(a = 1, b = 1, c = 1),
  plus_minus = 100,
  n.start = 1000,
  msg = FALSE
)
```

Arguments

d	data drame with two columns (x and y)
start	initial estimate for non-linear least square (default value: list(a = 1,b = 1,c = 1))
plus_minus	radius of random initial values (default: 100)
n.start	total number of initial points considered (deafult: 1000)
msg	logical flag whehter printing progress

Value

f.linear.plateau returns a list of two components (if converged): nls.summary: summary of the fitted model; nls.model: nls object

Examples

```
set.seed(4)
x <- rep(1:300, each=4)
a <- 8; b <- 0.05; c <- 100
y <- a + b * (x - c) * (x <= c) +
    rnorm(length(x), sd=0.1)
d <- cbind(x,y)

# a converged example:
ans <- f.linear.plateau(d, start=list(a = 7, b = 0.1, c = 150),
    plus_minus=10, n.start=10, msg=FALSE)

summary(ans$nls.model)
```

---

f.quad *Fitting quadratic model using mutiple initial vaues*

Description

f.quad fits quadratic model using multiple initial values. The multiple initial values are randomly sampled in a "cube" of parameter space. More precisely, quadratic model assumes y = a+b*x+c*x^2,

Usage

```
f.quad(
  d,
  start = list(a = 1, b = 1, c = 1),
  plus_minus = 1,
```

```
  n.start = 10,
  msg = FALSE
)
```

Arguments

d	data drame with two columns (x and y)
start	initial estimate for non-linear least square (default value: list(a = 1,b = 1,c = 1))
plus_minus	radius of random initial values (default: 100)
n.start	total number of initial points considered (deafult: 1000)
msg	logical flag whehter printing progress

Value

f.quad returns a list of two components (if converged): nls.summary: summary of the fitted model; nls.model: nls object

Examples

```
set.seed(1)
x <- rep(1:300, each=2)
a <- 8; b <- 0.05; c <- -1e-3
y <- a + b*x + c*x^2 + rnorm(length(x), sd=0.1)
d <- cbind(x,y)

# a converged example:
ans <- f.quad(d, start=list(a = 7, b = 0.02, c = 1e-5),
    plus_minus=10, n.start=10, msg=FALSE)

summary(ans$nls.model)
```

---

f.quad.plateau *Fitting quadratic plateau model using mutiple initial vaues*

Description

f.quad.plateau fits quadratic plateau model using multiple initial values. The multiple initial values are randomly sampled in a "cube" of parameter space. More precisely, quadratic plateau model assumes y = (a + b * x + c *x^2) * (x <= -0.5*b/c) + (a + -b^2/(4 * c)) * (x > -0.5 * b/c).

Usage

```
f.quad.plateau(
  d,
  start = list(a = 1, b = 1, c = 1),
  plus_minus = 100,
  n.start = 1000,
  msg = FALSE
)
```

Arguments

`d`	data drame with two columns (`x` and `y`)
`start`	initial estimate for non-linear least square (default value: `list(a = 1,b = 1,c = 1)`)
`plus_minus`	radius of random initial values (default: `100`)
`n.start`	total number of initial points considered (deafult: `1000`)
`msg`	logical flag whehter printing progress

Value

`f.quad.plateau` returns a list of two components (if converged): `nls.summary`: summary of the fitted model; `nls.model`: nls object

Examples

```
set.seed(3)
x <- rep(1:300, each=4)
a <- 8; b <- 0.05; c <- -1e-4
y <- (a + b * x + c *x^2) * (x <= -0.5*b/c) + (a + -b^2/(4 * c)) * (x > -0.5 * b/c) +
    rnorm(length(x), sd=0.1)
d <- cbind(x,y)

# a converged example:
ans <- f.quad.plateau(d, start=list(a = 7, b = 0.02, c = 1e-5),
    plus_minus=10, n.start=10, msg=FALSE)

summary(ans$nls.model)
```

Index