Package ‘OEFPIL’

March 5, 2021

Type Package

Title OEPFIL

Version 0.1.0

Description Package for optimal estimation of function parameters by Iterated Linearization.

License GPL (>=2)

Depends R(>= 3.5.0)

Imports Deriv, MASS, ggplot2, matrixcalc, minpack.lm, plyr

URL https://github.com/stazam/OEFPIL-

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Author Stanislav Zámečník [aut,cre],
Vojtěch Šindlář [aut,cre] (xsindlarv@math.muni.cz),
Zdeňka Geršlová [aut,cre] (gerslovaz@math.muni.cz),
Gejza Wimmer [aut] (Author of teoretical backround, wimmer@math.muni.cz)
TACR [fnd] (This software OEFPIL was financed by the Technology Agency of the Czech Republic within the ZÉTA Programme, https://www.tacr.cz)

Maintainer Stanislav Zámečník <xzamecniks@math.muni.cz>

R topics documented:

coef.OEFPIL *Extract model coefficients from OEFPIL*

Description

Function which extracts the estimated model coefficients from an object of class "OEFPIL".

Usage

```
## S3 method for class 'OEFPIL'
   coef(object)
```

Arguments

object	an object of class "OEFPIL" (a result of a call to OEFPIL).

Value

A named vector of estimated model coefficients extracted from an "OEFPIL" object.

See Also

OEFPIL

Examples

```
##Creating a data file (using steam data from MASS library)
library(MASS)
steamdata <- steam
colnames(steamdata) <- c("x","y")
startsteam <- list(b1 = 5, b2 = 8, b3 = 200)
k <- nrow(steamdata)
CM <- diag(rep(0.1,2*k))

##Creating an OEFPIL object
st1 <- OEFPIL(steamdata, y ~ b1 * 10^(b2 * x/ (b3 + x)), startsteam, CM, useNLS = F)

##Use of coef function
coef(st1)
```

confBands.OEFPIL *Calculate confidence bands for an object of class 'OEFPIL'*

Description

Function calculates pointwise confidence bands of estimated function from 'OEFPIL'.

Usage

```
## S3 method for class 'OEFPIL'
   confBands(object, xx, signif.level = 0.05)
```

Arguments

`xx`	numerical vector of points, where confidence bands will be calculated.
`signif.level`	numerical value or vector of significance levels for confidence intervals (default value is 0.05)
`object`	object of class 'OEFPIL'.

Details

We can add one numerical value or vector of numerical values of significance levels for confidence intervals.

Value

Matrix with named columns of estimated pointwise confidence bands of estimated function from on object `OEFPIL`. And also points where the bands are calculated.

See Also

`OEFPIL`

Examples

```
##-- Continuing the coef.OEFPIL(.) example:

##Use of confBands function with default parameters
(a <- confBands(st1))

#vector of numerical values
(b <- confBands(st1,signif.level = c(0.01,0.05)))
```

`confint.OEFPIL` *Confidence intervals for OEFPIL parameters*

Description

Function computes confidence intervals for the parameters counted by `OEFPIL` function.

Usage

```
## S3 method for class 'OEFPIL'
   confint(object, signif.level = object$contents$signif.level)
```

Arguments

`object`	an object of class `"OEFPIL"` (a result of a call to `OEFPIL`).
`signif.level`	a numerical value or a vector of significance levels for confidence intervals. If missing, a value from the input `"OEFPIL"` object is used.

Details

The confidence intervals are computing under normality assumption.

Value

A matrix of estimated confidence intervals for model coefficients from an "OEFPIL" object. The matrix contains lower and upper confidence limits (columns) for each parameter (rows).

See Also

OEFPIL

Examples

```
##-- Continuing the coef.OEFPIL(.) example:

##Use of confint function
#one numerical value
confint(st1)

#vector of numerical values
confint(st1, signif.level = c(0.01,0.05,0.1))
```

---

curvplot.OEFPIL   *Plot of estimated curve for OEFPIL object*

Description

Function for plotting the estimated curve with pointwise confidence bands for an object of class "OEFPIL".

Usage

```
curvplot.OEFPIL(object, signif.level, xx)
```

Arguments

`object`	an object of class "OEFPIL" (a result of a call to OEFPIL).
`signif.level`	a numeric value or a vector of significance levels for pointwise confidence bands. If missing, the estimated curve is plotted without confidence bands.
`xx`	a sequence of x-coordinates of points for computing and plotting confidence bands. If missing, the default sequence `seq(from = min(x), to = max(x), length.out = 301)` is used.

Value

A ggplot graph of the estimated curve with pointwise confidence bands. The result can be edit using other ggplot components as usually.

See Also

OEFPIL, paramplot.OEFPIL and plot.OEFPIL.

Examples

```
library(MASS)
library(ggplot2)

##Creating a data file
steamdata <- steam
colnames(steamdata) <- c("x","y")
n <- nrow(steamdata)
CM1 <- diag(rep(10,2*n))
CM2 <- diag(c(rep(12,n), rep(14,n)))

##Creating OEFPIL objects
st1 <- OEFPIL(steamdata, y ~ b1 * 10^(b2 * x/ (b3 + x)), list(b1 = 5, b2 = 8, b3 = 200),
             CM1, useNLS = F)
st2 <- OEFPIL(steamdata, y ~ b1 * 10^(b2 * x/ (b3 + x)), list(b1 = 5, b2 = 8, b3 = 200),
             CM2, useNLS = F)

##Use of curvplot.OEFPIL function on an object of class 'OEFPIL'
curvplot.OEFPIL(st1, signif.level = 0.05)

##Use of curvplot.OEFPIL function on an object of class 'OEFPIL' with different arguments
curvplot.OEFPIL(st2, signif.level = c(0.01,0.05), xx = seq(0,110,1))

##Use of curvplot.OEFPIL function with additional arguments as for ggplot2
curvplot.OEFPIL(st1, signif.level = 0.05) + labs(x = "New x label") + labs(title = "New graph title")
```

NanoIndent.OEFPIL *Estimation of parameters in nanoindentation*

Description

Fitting the unloading curve in nanoindentation process by power law function with parameters estimated by iterated linearization algorithm (OEFPIL). The special case of OEFPIL function customized for using in nanoindentation (see 'Details').

Usage

```
NanoIndent.OEFPIL(data, alpha.start, m.start, hp.start, unload.data = F, ucut = 0.98,
                  lcut = 0.4, CM, uh = 0.5, uF = 0.001, max.iter = 100, see.iter.val = F,
                  save.file.name, th = .Machine$double.eps ^ (2 / 3), signif.level = 0.05,
                          useNLS = T)
```

Arguments

data	an object of type `data.frame` with 2 named columns or `list` with 2 elements.
alpha.start	a starting value of the fitting constant alpha.
m.start	a starting value of the exponent m.
hp.start	a starting value of the permanent indentation depth hp.
unload.data	a logical value (default `FALSE`) indicating the structure of `data`. If `TRUE`, an input data contains only unloading part of the curve. If `FALSE`, an input data contains complete loading, hold and unloading parts of an indentation process.

`ucut`	a numerical value, indicating the upper bound of cut off.
`lcut`	a numerical value, indicating the lower bound of cut off.
`CM`	a covariance matrix of the input `data`. See 'Details' for more information.
`uh`	standard deviation of depth.
`uF`	standard deviation of load.
`max.iter`	maximum number of iterations.
`see.iter.val`	logical. If `TRUE`, all the partial results of the OEFPIL algorithm are displayed and saved. The default value is `FALSE`.
`save.file.name`	a name of the file for saving results. If missing, no output file is saved.
`th`	a numerical value, indicating threshold necessary for the iteration stoppage.
`signif.level`	a significance level for the confidence interval.
`useNLS`	logical. If `TRUE` (the default value), function will set up starting parameters calculated by `nlsLM` function (nonlinear least square estimation).

Details

In this special case of the `OEFPIL` function, the dependence of parameters is fixed in the form: $F = \alpha * (h - h_p)^m$, where `F` is load and `h` depth measured within a nanoindentation process. It is possible to set own starting values of the parameters, in the other case these values are calculated by the algorithm and printing into the console.

A selection of the part of the unloading curve fitted by a power law function is provided with `lcut` and `ucut` arguments. The default values 0.4 and 0.98 corresponds to the range 40 - 98% F_max (maximum force) as recommended in ISO 14577 standard.

The `CM` has to be a `2n` covariance matrix (where `n` is length of `data`) of following structure: first `n` elements of the diagonal correspond to the variance of depth and other to the variance of load. If argument `CM` is missing, the input covariance matrix is set to a diagonal matrix with variance of depth and load (calculated from `uh` and `uF`) on the diagonal. If standard deviations are missing too, the default values (`uh`=0.5, `uF`=0.001) are used.

The estimations and confidence intervals are computed under normality assumption (see `OEFPIL` 'Details').

Value

Returns an object of class `"OEFPIL"`. It is a list containing at least the following components

`name_Est`	estimations of model parameters.
`name_upgraded.start.val`	modified starting values of estimating parameters (result from `nlsLM` function).
`cov.m_Est`	estimated covariance matrix of parameters.
`it_num`	number of iterations.
`CI_parameters`	a list of confidence intervals for estimated parameters (a significance level is based on `signif.level` argument).
`logs`	warnings or messages of events, which happen during the run of the algorithm.
`...`	for other components specification see `OEFPIL`.
`contents`	a list of outputs as original values of data and other characteristics, which are usable in plotting or other operations with model results.

If `useNLS` argument is set to `FALSE`, the `name_upgraded.start.val` are the same as `start.values` (no `nlsLM` procedure for starting value fitting is performed).

References

ISO/IEC: *14577-1:2015 Metallic materials – Instrumented indentation test for hardness and materials parameters – Part 1: Test method* (ISO/IEC, Internation Organisation for Standardisation, 2015).

Anna Charvátová Campbell, Petr Grolich, Radek Šlesinger. *Niget: Nanoindentation general evaluation tool*. SoftwareX (2019), **Vol. 9**: 248–254. https://doi.org/10.1016/j.softx.2019.03.001.

Köning, R., Wimmer, G. and Witkovský, V. *Ellipse fitting by nonlinear constraints to demodulate quadrature homodyne interferometer signals and to determine the statistical uncertainty of the interferometric phase*. Measurement Science and Technology (2014).

See Also

OEFPIL

Examples

```
##Use of NanoIndent function for data file "silicaBerk.txt" (a part of the OEFPIL package)
signif.level = 0.05
output.form.NI <- NanoIndent.OEFPIL(silicaBerk, unload.data = TRUE, ucut = 0.98, lcut = 0.2,
uh = 0.5, uF = 0.001, signif.level = signif.level, useNLS = TRUE)

##The output is an object of class 'OEFPIL', supplementary functions for this class are available
##Use of summary function
summary(output.form.NI)

##Plot of estimated unloading curve
plot(output.form.NI, signif.level = signif.level)
```

OEFPIL — *Optimal Estimation of Parameters by Iterated Linearization*

Description

Function for computing optimal estimate of parameters of a nonlinear function by iterated linearization (using Taylor expansion). The model considers measurements errors in both (dependent and independent) variables.

Usage

```
OEFPIL(data, form, start.val, CM, max.iter = 100, see.iter.val = F,
       save.file.name, th, signif.level, useNLS = T)
```

Arguments

`data`	a data file can be any object of type `data.frame` with 2 named columns or `list` with 2 elements.
`form`	an object of class `formula` (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.

`start.val`	a named list of starting values of estimating parameters.
`CM`	a covariance matrix of `data` (See 'Details' for the information about required structure.).
`max.iter`	maximum number of iterations.
`see.iter.val`	logical. If `TRUE`, all the partial results of the algorithm are displayed and saved. The default value is `FALSE`.
`save.file.name`	a name of the file for saving results. If missing, no output file is saved.
`th`	a numerical value, indicating threshold necessary for the iteration stoppage. The default value is `.Machine$double.eps ^ (2 / 3)`.
`signif.level`	a significance level for the confidence interval. If missing, the default value 0.05 is used.
`useNLS`	logical. If `TRUE` (the default value), function will set up starting parameters calculated by `nlsLM` function (nonlinear least square estimation).

Details

Models for OEPFIL function are specified symbolically. A typical model has the form `y ~ f(x,a_1,...,a_n)`, where

- `y` is the (numerical) response vector
- `x` is the predictor
- terms `a_1,...,a_n` are parameters of specified model.

Function `f` is known nonlinear function with continuous second partial derivatives with respect to `x` and parameters `a_1,...a_n` (for more details see *Kubáček*).

All calculations are performed assuming normality of a response vector and measurements errors.

In the `data` entry of type `data.frame`, both columns must be named as variables in formula. The same holds for elements of `list`.

A choice of `start.val` is important for the convergence of the algorithm. If the `OEFPIL` algorithm does not converge, starting values modified by `nlsLM` function (`useNLS = TRUE`) are recommended (see Example 3).

The `CM` has to be a `2n` covariance matrix (where `n` is length of `data`) of following structure: first `n` elements of the diagonal correspond to the variance of independent variable (x) and other to the variance of dependent variable (y). If argument `CM` is missing, the input covariance matrix is set to a diagonal variance matrix with sample variance on the main diagonal.

Value

Returns an object of class `"OEFPIL"`. It is a list containing the following components

`name_Est`	estimations of model parameters.
`name_upgraded.start.val`	modified starting values of estimating parameters (result from `nlsLM` function).
`cov.m_Est`	estimated covariance matrix of parameters.
`cov.m_nlsLM`	a covariance matrix of starting values of parameters from `nlsLM` function (if `useNLS` was set to `TRUE`).
`it_num`	number of iterations.

`name_previous.step`
: the parameter values from the previous iterative step.

`CI_parameters`
: a list of confidence intervals for estimated parameters (a significance level is based on `signif.level` argument).

`logs`
: warnings or messages of events, which happen during the run of the algorithm.

`contents`
: a list of outputs as original values of data and other characteristics, which are usable in plotting or other operations with model results.

If `useNLS` argument is set to `FALSE`, the `name_upgraded.start.val` are the same as `start.values` (no `nlsLM` procedure for starting value fitting is performed).

Note

The symbol `pi` is reserved for the Ludolf's constant. So naming one of the model´s parameters by this symbol results in constant entry of the model.

References

Kubáček, L. and Kubáčková, L.*Statistika a metrologie* (2000). Univerzita Palackého v Olomouci.

Köning, R., Wimmer, G. and Witkovský, V. *Ellipse fitting by nonlinear constraints to demodulate quadrature homodyne interferometer signals and to determine the statistical uncertainty of the interferometric phase*. Measurement Science and Technology (2014).

See Also

`NanoIndent.OEFPIL` and function `nlsLM` from `minipack.lm` package for nonlinear least square algorithms.

Examples

```
##Example 1 - Use of OEFPIL function for steam data from MASS library
library(MASS)
steamdata <- steam
colnames(steamdata) <- c("x","y")
k <- nrow(steamdata)
CM <- diag(rep(5,2*k))

st1 <- OEFPIL(steamdata, y ~ b1 * 10 ^ (b2 * x/ (b3 + x)),
 list(b1 = 5, b2 = 8, b3 = 200), CM, useNLS = F)

## Displaying results using summary function
summary(st1)

## Plot of estimated function
plot(st1, signif.level = signif.level)

##Example 2 - Use of OEFPIL for nanoindentation data "silica2098.RData" (which is part of the OEFPIL package)
colnames(silica2098) <- c('x','y')

## Preparing arguments for OEFPIL function
max.iter = 100
see.iter.val = F
#th = 0.001
signif.level = 0.05
useNLS = T
```

```
## Creating a list with starting values for parameters
start.val <- list(alpha=0.1, m=1.5, hp=0.9)
names(start.val) <- c("alpha", "m", "hp")

## Imputed formula
form <- y ~ alpha * (x - hp) ^ m
k <- length(silica2098[,1])
CM <- diag(c(rep(0.5^2,k),rep(0.001^2,k)))

## Use of OEFPIL function with defined arguments
output.form <- OEFPIL(silica2098, form, start.val, CM = CM, max.iter = max.iter, see.iter.val = see.iter.val, s

## Displaying results with summary (the result is the same as in NanoIndent.OEFPIL function)
summary(output.form)

##Example 3 - shows sensitivity of algorithm to starting values of parameters
startsteam <- list(b1 = 0.1, b2 = 5, b3 = 200)
st3 <- OEFPIL(steamdata, y ~ b1 * 10^(b2 * x/ (b3 + x)), startsteam,
              CM1, useNLS = F)

##With choice useNLS = T, starting values are upgraded by nls function and algorithm converges
st3 <- OEFPIL(steamdata, y ~ b1 * 10^(b2 * x/ (b3 + x)), startsteam,
              CM1, useNLS = T)
print(st3)
```

`ortresiduals.OEFPIL` *Orthogonal residuals from an OEFPIL object*

Description

Function for calculating orthogonal residuals of an "OEFPIL" object (i.e. the shortest Euclidean distance between data points and the estimated function from OEFPIL).

Usage

```
ortresiduals.OEFPIL(object, min.c)
```

Arguments

object	an object of class "OEFPIL" (a result of a call to OEFPIL).
min.c	a numeric value, for defining minimization interval for the optimize function (if not defined, default value 0.05 * range(x) is used). Must be positive.

Value

Returns an object of type list containing following components

x.ores	a numerical vector of x coordinates of points, where the minimal distance is realized.
o.resid	a numerical vector of orthogonal residuals (minimal Euclidean distances between data points and estimated function).
SSort	the orthogonal sum of squares.

Note

The value `min.c` should not be too small. In that case the minimization interval is too narrow and the result can be misleading (see Example 3).

See Also

`OEFPIL`

Examples

```
##-- Continuing the coef.OEFPIL(.) example:

##Example 1 Use ortresiduals.OEFPIL function on the OEFPIL object, with specified value 'min.c'
ortresiduals.OEFPIL(st1,5)

##Example 2 Use ortresiduals.OEFPIL function without value 'min.c' (defaut.value = 0.05 * range(x))
ortresiduals.OEFPIL(st1)

##Example 3 Choice of too narrow interval. Misleading result!
ortresiduals.OEFPIL(st1,0.5)
```

`paramplot.OEFPIL` *Plot parameters of an OEFPIL object*

Description

Function for plotting the estimated values of the parameters with error bars (plus minus standard deviation) using `ggplot` for an object (or list of objects) of class `"OEFPIL"`.

Usage

```
paramplot.OEFPIL(object)
```

Arguments

`object`	an object or a `list` of objects of class `"OEFPIL"` (a result of a call to `OEFPIL`).

Details

The input list has to be without `NaN`, `NA`, `Inf` or `-Inf` values in the estimated parameters or covariance matrix in the source `"OEFPIL"` object. In that case the function returns a warning message and no graph is plotted (see Example 3).

Value

A ggplot graph of the estimated parameter values with error bars. The result can be edit using other ggplot components as usually.

Note

Due to possible large differences in units of estimated parameters, the `scale` argument for facetting in the `ggplot` graph is set to `"free"`. It should be taken into account when interpreting the results.

See Also

OEFPIL, curvplot.OEFPIL and plot.OEFPIL.

Examples

```
##-- Continuing the coef.OEFPIL(.) example:

CM2 <- diag(c(rep(0.2^2,n), rep(0.1^2,n)))
st2 <- OEFPIL(steamdata, y ~ b1 * 10^(b2 * x/ (b3 + x)), list(b1 = 5, b2 = 8, b3 = 200),
             CM2, useNLS = F)

##Example 1 - Use of paramplot.OEFPIL function on an object of class 'OEFPIL'
paramplot.OEFPIL(st2)

##Example 2 - Use of paramplot.OEFPIL function on a list of objects of class 'OEFPIL'
paramplot.OEFPIL(list(st1,st2))

##Example 3 - Use of paramplot.OEFPIL function on an object with NaN values (i. e. OEPFIL function does not conve
startsteam <- list(b1 = 0.1, b2 = 5, b3 = 200)
st3 <- OEFPIL(steamdata, y ~ b1 * 10^(b2 * x/ (b3 + x)), startsteam,
              CM1, useNLS = F)
paramplot.OEFPIL(st3)
```

---

plot.OEFPIL *Plot the estimate from an OEPFIL object*

Description

Plot of the iterated linearization estimate of a function from an "OEFPIL" object with pointwise confidence bands.

Usage

```
## S3 method for class 'OEFPIL'
    plot(object, xx, signif.level,...)
```

Arguments

`object`	an object of class "OEFPIL" (a result of a call to OEFPIL).
`xx`	a sequence of x-coordinates of points for computing and plotting confidence bands. If missing, the default sequence `seq(from = min(x),to = max(x),length.out = 301)` is used.
`signif.level`	a numerical value or a vector of significance levels for confidence bands.
`...`	additional arguments (same as in plot function) affecting the plot.

Details

If the `signif.level` argument is missing, the value is set to 0.05, but the confidence bands are not plotted. The confidence bands are computing under normality assumption.

Value

Returns an object of type list containing at least the following components

- `xx` a numerical vector of points where bands are calculated.
- `yy` a numerical vector with values of estimated function in `xx`

  .
- `PointwiseCB` a matrix of pointwise confidence bands at points `xx`

  .

See Also

OEFPIL

Examples

```
##-- Continuing the coef.OEFPIL(.) example:

##Use of plot function with default parameters, signif.level is not set up...No confidence bands are plotted
plot(st1)

##Use of plot function with different parameters
plot(st1, seq(0,113,0.1), signif.level = c(0.01,0.05), main = "Graph of estimated function")

##Return values of plot function
(a <- plot(st1, signif.level = 0.05))
```

---

print.OEFPIL *Print function for an object of class 'OEFPIL'*

Description

Function prints the information about an object of class `"OEFPIL"`. print(object)

Usage

```
## S3 method for class 'OEFPIL'
print(object)
```

Arguments

`object` an object of class `"OEFPIL"` (a result of a call to `OEFPIL`).

Details

Function prints the formula and estimated parameters of the model from an `OEFPIL` object.

See Also

OEFPIL

Examples

```
##-- Continuing the coef.OEFPIL(.) example:

##Use of print function
print(st1)
```

---

silica2098 *Nanoindentation measurements data*

Description

Load and depth measured with Berkovich diamond tip on fused silica material. Data contains only cutted part of unloading indentation curve (20 - 98%, as recommended in ISO 14577 standard). The relationship between *Fload* and *Depth h* is given by equation $Fload = \alpha(h - h_p)^m$, where *h_p* is permanent indentation depth after removal of the load, α is fitting constant related to the indenter geometry and power law exponent *m* should be from (1,2) interval.

Usage

```
silica2098
```

Format

The data frame contains two columns:

Depth nanoindentation depth, in nanometers

Fload load, in milinewtons

Source

Czech Metrology Institute, Brno, Czech Republic.

References

ISO/IEC: *14577-1:2015 Metallic materials – Instrumented indentation test for hardness and materials parameters – Part 1: Test method* (ISO/IEC, Internation Organisation for Standardisation, 2015).

Examples

```
attach(silica2098)
plot(Depth, Fload, main = 'Graph of nanoindentation data', xlab = 'Depth (nm)', ylab = 'Load (mN)',
col = 'darkgreen', cex = 1)
```

summary.OEFPIL — *Summary from an OEFPIL object*

Description

Function for fast and clean output of all basic information of an "OEFPIL" object.

Usage

```
## S3 method for class 'OEFPIL'
   summary(object, signif.level = object$contents$signif.level, print = T)
```

Arguments

object	an object of class "OEFPIL" (a result of a call to OEFPIL).
signif.level	a significance level for the confidence interval. If missing, a value from the input OEFPIL object is used.
print	print out result summaries in the console (default TRUE)

Value

Returns an object of type list containing following components

param_Est	the (numerical) vector of estimated model parameters.
sd	standard deviations for estimated model parameters.
cov.m_Est	the covariance matrix of estimated model parameters.
it_num	number of iterations.
CI_parameters	the matrix of lower and upper bounds for confidence intervals.

See Also

OEFPIL

Examples

```
##-- Continuing the coef.OEFPIL(.) example:

##Use of summary function with default parameters
summary(st1)

##Use of summary function with different parameters
summary(st1, signif.level = 0.01, print = F)
```

vcov.OEFPIL *Covariance matrix from an OEFPIL object*

Description

Function for extracting the estimated covariance matrix from an object of class "OEFPIL".

Usage

```
## S3 method for class 'OEFPIL'
   vcov(object)
```

Arguments

object	an object of class "OEFPIL" (a result of a call to OEFPIL).

Value

A matrix of the estimated covariances between the parameter estimates from an "OEFPIL" object.

See Also

OEFPIL

Examples

```
##-- Continuing the coef.OEFPIL(.) example:

##Use of vcov function
vcov(st1)
```

Index