Package 'Slick'

September 11, 2025

Type Package

Title Interactive Visualization of MSE Results

Version 1.0.0

Maintainer Adrian Hordyk <adrian@bluematterscience.com>

Description A framework for visualizing and exploring results of a Management
Strategy Evaluation (MSE). The publication quality figures and tables
can be developed directly from the R console, or interactively explored
with the 'Slick' App. For more details, see the `Slick` website
<https://slick.bluematterscience.com>.

URL https://slick.bluematterscience.com/,
https://github.com/Blue-Matter/Slick

License GPL-2

Depends R (>= 4.1.0)

Imports cli, dplyr, DT, ggplot2, ggrepel, golem, graphics, grDevices,
methods, scales, shiny, stats, tibble, utils

Suggests bookdown, colourpicker, colorspace, cowplot, esquisse,
flextable, fresh, httr, kableExtra, knitr, MSEtool, openMSE,
RColorBrewer, shiny.i18n, shinyalert, shinyBS, shinycssloaders,
shinydashboard, shinydashboardPlus, shinyhelper, shinyjs,
shinyWidgets, rmarkdown, testthat (>= 3.0.0), waiter

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.3

Collate 'RunSlick.R' 'Slick_tools.R' 'a_classunions.R' 'a_imports.R'
'aa_generics.R' 'app_config.R' 'app_server.R' 'app_ui.R'
'class_Boxplot.R' 'class_Check.R' 'class_Kobe.R' 'class_MPs.R'
'class_OMs.R' 'class_Quilt.R' 'class_Tradeoff.R'
'class_Timeseries.R' 'class_Spider.R' 'class_Slick.R'
'fct_FilterSlick.R' 'fct_Update.R'
'fct_check_required_packages.R' 'fct_check_slick.R'
'fct_colors.R' 'fct_dashboardHeader2.R'

'fct_load_casestudies.R' 'fct_misc.R' 'fct_plotBoxplot.R'
'fct_plotKobe.R' 'fct_plotQuilt.R' 'fct_plotSpider.R'
'fct_plotTimeseries.R' 'fct_plotTradeoff.R' 'fct_translator.R'
'make_Slick_old.R' 'man_functions.R' 'misc.R' 'mod_About.R'
'mod_Boxplot.R' 'mod_Boxplot_OM.R' 'mod_Boxplot_overall.R'
'mod_Global_Filters.R' 'mod_Home.R' 'mod_Kobe.R'
'mod_Kobe_overall.R' 'mod_Kobe_time.R' 'mod_MP_Color.R'
'mod_MP_Info.R' 'mod_Metadata.R' 'mod_OM_Info.R'
'mod_PM_Info.R' 'mod_Page_Filter.R' 'mod_Quilt.R'
'mod_Report_Add.R' 'mod_Report_Add_Button.R'
'mod_Report_Page.R' 'mod_Report_Page_plot.R' 'mod_Resources.R'
'mod_Sidebar.R' 'mod_Spider.R' 'mod_Spider_MP.R'
'mod_Spider_OM.R' 'mod_Spider_overall.R' 'mod_Timeseries.R'
'mod_Timeseries_byMP.R' 'mod_Timeseries_byOM.R'
'mod_Timeseries_overall.R' 'mod_Tradeoff.R'
'mod_filter_selection.R' 'mod_filter_selection_OM.R'
'mod_subtitle.R' 'mod_toplink.R' 'utils_Table.R' 'utils_util.R'

Contents

App *Run the Slick App in the local browser*

Description

An online hosted version of the Slick App is available here

Usage

```
App(...)
```

Arguments

...   Additional arguments. To load a object directly in the App, use App(slick=myslick) where myslick is your Slick-class() object.

Details

App() runs the Slick Shiny Application

Value

None

Boxplot   *Methods for Creating, Accessing and Assigning* Boxplot *objects*

Description

The Boxplot function is used both to create and modify an [Boxplot-class()](#) object. and to access
and assign Boxplot for an object of class [Slick-class()](#). See Details.

Usage

```
Boxplot(
  Code = "",
  Label = "",
  Description = "",
  Value = array(),
  Preset = list(),
  Defaults = list("overall", "boxplot")
)

Boxplot(Slick) <- value

## S4 method for signature 'missing'
Boxplot()

## S4 method for signature 'character_list'
Boxplot(
  Code = "",
  Label = "",
  Description = "",
  Value = array(),
  Preset = list(),
  Defaults = list("overall", "boxplot")
)

## S4 method for signature 'Slick'
Boxplot(Code)

## S4 replacement method for signature 'Slick'
Boxplot(Slick) <- value
```

Arguments

Code	A *short* code for the Performance Indicators for this object. A character string length nPI or a named list for multi-language support. See Details
Label	A short label for the Performance Indicators for this object. Used to label axes on charts. Can be longer than Code but recommended to keep short as possible so it shows clearly in plots and tables. A character string length nPI or a named list for multi-language support. See Details
Description	A description for the Performance Indicators for this object. Can include Markdown, see Examples. A character string length nPI or a named list for multi-language support. See Details
Value	A numeric array with the stochastic performance indicator values for each simulation (sim), operating model (OM), management procedure (MP), and performance indicator (PI). Dimensions: c(nsim, nOM, nMP, and nPI).
Preset	An optional named list for the preset buttons in the App(). The name of the list element will appear as a button in the App().
Defaults	A list object with default selections for the Boxplot
Slick	A Slick-class() object
value	A Boxplot-class() object

Details

Objects of class Boxplot are created with Boxplot()

Use plotBoxplot() to create the boxplot from the console.

Use the Code(), Label(), Description(), Value(), Preset() functions to access and assign the values for an existing Boxplot object, see Examples

Multi-Language Support:

Text with multi-language supported can be provided as a named list. Available languages:

- en: English (default)
- es: Spanish
- fr: French

Note:

Character strings in Code, Label, and Description must all be same length as the number of performance indicators (nPIs) in 'Value

Defaults:

Defaults is used to select the plot options that are selected in the Boxplot. It is a list of length 2, with the following requirements for the list elements:

1. A character string. Options: 'overall' (default) or 'byom'
2. A character string. Options: 'boxplot' (default), 'violin', or 'both'

If unrecognized values are entered, the defaults will be used.

Functions

- Boxplot(missing): Create an empty Boxplot object
- Boxplot(character_list): Create a populated Boxplot object
- Boxplot(Slick): Return Boxplot from a Slick-class() object
- Boxplot(Slick) <- value: Assign a Boxplot-class() object to a Slick-class() object

See Also

Code(), Label(), Description(), Metadata(), Value(), Preset(), Defaults(), plotBoxplot()

Examples

```
# Generate dummy values
nsim <- 10
nOM <- 2
nMP <- 4
nPI <- 3

values <- array(NA, dim=c(nsim, nOM, nMP, nPI))
pi_means <- runif(nPI, 5, 50)
for (om in 1:nOM) {
  for (mp in 1:nMP) {
    for (pi in 1:nPI) {
      values[,om, mp, pi] <- rlnorm(nsim, log(pi_means[pi]), 0.4)
    }
  }
}

# Create and populate Object
boxplot <- Boxplot(Code=c('PI1', 'PI2', 'PI3'),
                   Label=c('Performance Indicator 1',
                           'Performance Indicator 2',
                           'Performance Indicator 3'),
                   Description = c('This is the description for PI 1',
                                   'This is the description for PI 2',
                                   'This is the description for PI 3'),
                   Value=values)

# Check
Check(boxplot)

# Add to `Slick` object
slick <- Slick()
Boxplot(slick) <- boxplot

# Plots
plotBoxplot(slick)

plotBoxplot(slick, type='violin')

plotBoxplot(slick, byOM=TRUE)
```

```
plotBoxplot(slick, 2, type='both', byOM=TRUE)
```

Boxplot-class   Boxplot *S4 Object Class*

Description

Objects of class `Boxplot` are used to store information for the Boxplot and Violin plots. Like all S4 objects in `Slick`, slots in this object can be accessed and assigned using functions corresponding to slot name. See `Boxplot()` and the the `See Also` section below.

Details

Objects of class `Boxplot` are created with `Boxplot()`

Multi-Language Support:

Text with multi-language supported can be provided as a named list. Available languages:

- `en`: English (default)
- `es`: Spanish
- `fr`: French

Note:

Character strings in `Code`, `Label`, and `Description` must all be same length as the number of performance indicators (nPIs) in `Value`

Slots

`Code` A *short* code for the Performance Indicators for this object. A character string length `nPI` or a named list for multi-language support. See `Details`

`Label` A short label for the Performance Indicators for this object. Used to label axes on charts. Can be longer than `Code` but recommended to keep short as possible so it shows clearly in plots and tables. A character string length `nPI` or a named list for multi-language support. See `Details`

`Description` A description for the Performance Indicators for this object. Can include Markdown, see `Examples`. A character string length `nPI` or a named list for multi-language support. See `Details`

`Value` A numeric array with the stochastic performance indicator values for each simulation (sim), operating model (OM), management procedure (MP), and performance indicator (PI). Dimensions: c(`nsim`, `nOM`, `nMP`, and `nPI`).

`Preset` An optional named list for the preset buttons in the `App()`. The name of the list element will appear as a button in the `App()`.

`Defaults` A list object with default selections for the Boxplot. See `Boxplot()`

See Also

Boxplot(), Code(), Label(), Description(), Metadata(), Value(), Preset()

Examples

```
# Generate dummy values
nsim <- 10
nOM <- 2
nMP <- 4
nPI <- 3

values <- array(NA, dim=c(nsim, nOM, nMP, nPI))
pi_means <- runif(nPI, 5, 50)
for (om in 1:nOM) {
  for (mp in 1:nMP) {
    for (pi in 1:nPI) {
      values[,om, mp, pi] <- rlnorm(nsim, log(pi_means[pi]), 0.4)
    }
  }
}

# Create and populate Object
boxplot <- Boxplot(Code=c('PI1', 'PI2', 'PI3'),
                   Label=c('Performance Indicator 1',
                           'Performance Indicator 2',
                           'Performance Indicator 3'),
                   Description = c('This is the description for PI 1',
                                   'This is the description for PI 2',
                                   'This is the description for PI 3'),
                   Value=values)

# Check
Check(boxplot)

# Add to `Slick` object
slick <- Slick()
Boxplot(slick) <- boxplot

# Plots
plotBoxplot(slick)

plotBoxplot(slick, type='violin')

plotBoxplot(slick, byOM=TRUE)

plotBoxplot(slick, 2, type='both', byOM=TRUE)
```

Check   *Check an object for errors or issues*

Description

Checks S4 objects to check for warnings and errors

Usage

```
Check(object, ...)

## S4 method for signature 'Boxplot'
Check(object, req_dims = c(NA, NA, NA, NA))

## S4 method for signature 'Kobe'
Check(object)

## S4 method for signature 'MPs'
Check(object)

## S4 method for signature 'OMs'
Check(object)

## S4 method for signature 'Quilt'
Check(object)

## S4 method for signature 'Tradeoff'
Check(object)

## S4 method for signature 'Timeseries'
Check(object)

## S4 method for signature 'Spider'
Check(object)

## S4 method for signature 'Slick'
Check(object)
```

Arguments

object	An object of class: Slick-class(), MPs-class(), OMs-class() or the six chart types: Boxplot-class(), Kobe-class(), Quilt-class(), Spider-class(), Timeseries-class(), and Tradeoff-class().
...	Additional arguments
req_dims	Required dimensions for Value slot. Used internally

Value

Prints messages to the console

Methods (by class)

- Check(Boxplot): Check Boxplot-class() objects for errors
- Check(Kobe): Check Kobe-class() objects for errors
- Check(MPs): Check MPs-class() objects for errors
- Check(OMs): Check OMs-class() objects for errors
- Check(Quilt): Check Quilt-class() objects for errors
- Check(Tradeoff): Check Tradeoff-class() objects for errors
- Check(Timeseries): Check Timeseries-class() objects for errors
- Check(Spider): Check Spider-class() objects for errors
- Check(Slick): Check Slick-class() objects for errors

Examples

```
slick <- Slick()
Check(slick)
```

Code	*Access or assign* Code, Label, *and* Description *for a valid object class*

Description

Access or assign Code, Label, and Description for a valid object class

Usage

```
Code(object, lang = "en")

Code(object) <- value

Description(object, lang = "en")

Description(object) <- value

Label(object, lang = "en")

Label(object) <- value

## S4 method for signature 'Boxplot'
Code(object, lang = "en")

## S4 replacement method for signature 'Boxplot'
Code(object) <- value
```

```
## S4 method for signature 'Boxplot'
Description(object, lang = "en")

## S4 method for signature 'Boxplot'
Label(object, lang = "en")

## S4 replacement method for signature 'Boxplot'
Label(object) <- value

## S4 replacement method for signature 'Boxplot'
Description(object) <- value

## S4 method for signature 'Kobe'
Code(object, lang = "en")

## S4 replacement method for signature 'Kobe'
Code(object) <- value

## S4 method for signature 'Kobe'
Description(object, lang = "en")

## S4 replacement method for signature 'Kobe'
Description(object) <- value

## S4 method for signature 'Kobe'
Label(object, lang = "en")

## S4 replacement method for signature 'Kobe'
Label(object) <- value

## S4 method for signature 'MPs'
Code(object, lang = "en")

## S4 replacement method for signature 'MPs'
Code(object) <- value

## S4 method for signature 'MPs'
Label(object, lang = "en")

## S4 replacement method for signature 'MPs'
Label(object) <- value

## S4 method for signature 'MPs'
Description(object, lang = "en")

## S4 replacement method for signature 'MPs'
Description(object) <- value
```

```
## S4 method for signature 'Quilt'
Code(object, lang = "en")

## S4 replacement method for signature 'Quilt'
Code(object) <- value

## S4 method for signature 'Quilt'
Description(object, lang = "en")

## S4 replacement method for signature 'Quilt'
Description(object) <- value

## S4 method for signature 'Quilt'
Label(object, lang = "en")

## S4 replacement method for signature 'Quilt'
Label(object) <- value

## S4 method for signature 'Tradeoff'
Code(object, lang = "en")

## S4 replacement method for signature 'Tradeoff'
Code(object) <- value

## S4 method for signature 'Tradeoff'
Description(object, lang = "en")

## S4 replacement method for signature 'Tradeoff'
Description(object) <- value

## S4 method for signature 'Tradeoff'
Label(object, lang = "en")

## S4 replacement method for signature 'Tradeoff'
Label(object) <- value

## S4 method for signature 'Timeseries'
Code(object, lang = "en")

## S4 replacement method for signature 'Timeseries'
Code(object) <- value

## S4 method for signature 'Timeseries'
Description(object, lang = "en")

## S4 replacement method for signature 'Timeseries'
Description(object) <- value
```

```
## S4 method for signature 'Timeseries'
Label(object, lang = "en")

## S4 replacement method for signature 'Timeseries'
Label(object) <- value

## S4 method for signature 'Spider'
Code(object, lang = "en")

## S4 replacement method for signature 'Spider'
Code(object) <- value

## S4 method for signature 'Spider'
Description(object, lang = "en")

## S4 replacement method for signature 'Spider'
Description(object) <- value

## S4 method for signature 'Spider'
Label(object, lang = "en")

## S4 replacement method for signature 'Spider'
Label(object) <- value
```

Arguments

object	An object of class `MPs-class()`, `Boxplot-class()`, `Kobe-class()`, `Quilt-class()`, `Spider-class()`, `Timeseries-class()`, or `Tradeoff-class()`
lang	Optional text string specifying the language (if available). Either 'en', 'es', or 'fr' for English, Spanish, or French respectively
value	A character vector or a named list for multi-language support

Details

`Code`, `Label`, and `Description` must all be equal length.

Multi-Language Support:

Text with multi-language supported can be provided as a named list. Available languages:

- en: English (default)
- es: Spanish
- fr: French

Value

Returns character string or named list with the contents of the `Code`, `Label`, or `Description` slot of `object`

Methods (by class)

- `Code(Boxplot)`: Return Code from a [`Boxplot-class()`](Boxplot-class()) object
- `Code(Boxplot) <- value`: Assign Code to a [`Boxplot-class()`](Boxplot-class()) object
- `Description(Boxplot)`: Return Description from a [`Boxplot-class()`](Boxplot-class()) object
- `Label(Boxplot)`: Return Label from a [`Boxplot-class()`](Boxplot-class()) object
- `Label(Boxplot) <- value`: Assign Label to a [`Boxplot-class()`](Boxplot-class()) object
- `Description(Boxplot) <- value`: Assign Description to a [`Boxplot-class()`](Boxplot-class()) object
- `Code(Kobe)`: Return Code from a [`Kobe-class()`](Kobe-class()) object
- `Code(Kobe) <- value`: Assign Code to a [`Kobe-class()`](Kobe-class()) object
- `Description(Kobe)`: Return Description from a [`Kobe-class()`](Kobe-class()) object
- `Description(Kobe) <- value`: Assign Description to a [`Kobe-class()`](Kobe-class()) object
- `Label(Kobe)`: Return Label from a [`Kobe-class()`](Kobe-class()) object
- `Label(Kobe) <- value`: Assign Label to a [`Kobe-class()`](Kobe-class()) object
- `Code(MPs)`: Return Code from a [`MPs-class()`](MPs-class()) object
- `Code(MPs) <- value`: Assign Code to a [`MPs-class()`](MPs-class()) object
- `Label(MPs)`: Return Label from a [`MPs-class()`](MPs-class()) object
- `Label(MPs) <- value`: Assign Label to a [`MPs-class()`](MPs-class()) object
- `Description(MPs)`: Return Description from a [`MPs-class()`](MPs-class()) object
- `Description(MPs) <- value`: Assign Description to a [`MPs-class()`](MPs-class()) object
- `Code(Quilt)`: Return Code from a [`Quilt-class()`](Quilt-class()) object
- `Code(Quilt) <- value`: Assign Code to a [`Quilt-class()`](Quilt-class()) object
- `Description(Quilt)`: Return Description from a [`Quilt-class()`](Quilt-class()) object
- `Description(Quilt) <- value`: Assign Description to a [`Quilt-class()`](Quilt-class()) object
- `Label(Quilt)`: Return Label from a [`Quilt-class()`](Quilt-class()) object
- `Label(Quilt) <- value`: Assign Label to a [`Quilt-class()`](Quilt-class()) object
- `Code(Tradeoff)`: Return Code from a [`Tradeoff-class()`](Tradeoff-class()) object
- `Code(Tradeoff) <- value`: Assign Code to a [`Tradeoff-class()`](Tradeoff-class()) object
- `Description(Tradeoff)`: Return Description from a [`Tradeoff-class()`](Tradeoff-class()) object
- `Description(Tradeoff) <- value`: Assign Description to a [`Tradeoff-class()`](Tradeoff-class()) object
- `Label(Tradeoff)`: Return Label from a [`Tradeoff-class()`](Tradeoff-class()) object
- `Label(Tradeoff) <- value`: Assign Label to a [`Tradeoff-class()`](Tradeoff-class()) object
- `Code(Timeseries)`: Return Code from a [`Timeseries-class()`](Timeseries-class()) object
- `Code(Timeseries) <- value`: Assign Code to a [`Timeseries-class()`](Timeseries-class()) object
- `Description(Timeseries)`: Return Description from a [`Timeseries-class()`](Timeseries-class()) object
- `Description(Timeseries) <- value`: Assign Description to a [`Timeseries-class()`](Timeseries-class()) object
- `Label(Timeseries)`: Return Label from a [`Timeseries-class()`](Timeseries-class()) object

- Label(Timeseries) <- value: Assign Label to a [Timeseries-class()](#) object

- Code(Spider): Return Code from a [Spider-class()](#) object

- Code(Spider) <- value: Assign Code to a [Spider-class()](#) object

- Description(Spider): Return Description from a [Spider-class()](#) object

- Description(Spider) <- value: Assign Description to a [Spider-class()](#) object

- Label(Spider): Return Label from a [Spider-class()](#) object

- Label(Spider) <- value: Assign Label to a [Spider-class()](#) object

See Also

[Label()](#), [Description()](#), [MPs-class()](#), [Boxplot-class()](#), [Kobe-class()](#), [Quilt-class()](#), [Spider-class()](#), [Timeseries-class()](#), [Tradeoff-class()](#)

Color   *Access or assign* Color *for* MPs *and* Quilt *objects*

Description

Access or assign Color for MPs and Quilt objects

Usage

```
Color(object)

Color(object) <- value

## S4 method for signature 'MPs'
Color(object)

## S4 replacement method for signature 'MPs'
Color(object) <- value

## S4 method for signature 'Quilt'
Color(object)

## S4 replacement method for signature 'Quilt'
Color(object) <- value
```

Arguments

object	An [MPs-class()](#) or [Quilt-class()](#) object
value	A character vector formatted to match the class of object. See the documentation for corresponding object class for more details.

Value

Returns a character string with the contents of the `Color` slot of `object`

Methods (by class)

- `Color(MPs)`: Return Color from a [`MPs-class()`](#) object
- `Color(MPs) <- value`: Preset Assign Color to a [`MPs-class()`](#) object
- `Color(Quilt)`: Return Color from a [`Quilt-class()`](#) object
- `Color(Quilt) <- value`: Preset Assign Color to a [`Quilt-class()`](#) object

Defaults   *Access or assign the default selections for the plots in the* App()

Description

Access or assign the default selections for the plots in the App()

Usage

```
Defaults(object)

Defaults(object) <- value

## S4 method for signature 'Boxplot'
Defaults(object)

## S4 replacement method for signature 'Boxplot'
Defaults(object) <- value

## S4 method for signature 'Kobe'
Defaults(object)

## S4 replacement method for signature 'Kobe'
Defaults(object) <- value
```

Arguments

object	A [Boxplot()](#) or [Kobe()](#) object
value	A list of default selections for the [App()](#)

Value

Returns a list object with default selections from the `Defaults` slot of [Boxplot()](#) and [Kobe()](#) objects.

Methods (by class)

- `Defaults(Boxplot)`: Defaults Code from a [`Boxplot-class()`](#) object

- `Defaults(Boxplot) <- value`: Assign Defaults to a [`Boxplot-class()`](#) object

- `Defaults(Kobe)`: Defaults Code from a [`Kobe-class()`](#) object

- `Defaults(Kobe) <- value`: Assign Defaults to a [`Kobe-class()`](#) object

`default_mp_colors` *Set Default Colors for the MPs*

Description

Requires the `colorspace` package

Usage

```
default_mp_colors(nMP)
```

Arguments

nMP   The number of management procedures

Value

A character vector of length nMP with color hex codes

Examples

```
cols <- default_mp_colors(4)
cols
```

Design *Return the Design matrix from an* OMs *object*

Description

Return the Design matrix from an OMs object

Usage

```
Design(object)

Design(object) <- value

## S4 method for signature 'OMs'
Design(object)

## S4 replacement method for signature 'OMs'
Design(object) <- value

## S4 method for signature 'Slick'
Design(object)

## S4 replacement method for signature 'Slick'
Design(object) <- value
```

Arguments

object	A [Slick-class()] object
value	A data.frame with the Design matrix

Value

The Design matrix from an OMs object in a Slick object

Methods (by class)

- Design(OMs): Return the operating model Design matrix from a [OMs-class()] object
- Design(OMs) <- value: Assign the operating model Design matrix to a [OMs-class()] object
- Design(Slick): Access the operating model Design matrix from a [Slick-class()] object
- Design(Slick) <- value: Assign the operating model Design matrix to a [Slick-class()] object

---

Factors	*Return the Factors matrix from an* OMs *object*

Description

Return the Factors matrix from an OMs object

Usage

```
Factors(object, lang = "en")

Factors(object) <- value

## S4 method for signature 'OMs'
Factors(object, lang = "en")

## S4 replacement method for signature 'OMs'
Factors(object) <- value

## S4 method for signature 'Slick'
Factors(object, lang = "en")

## S4 replacement method for signature 'Slick'
Factors(object) <- value
```

Arguments

object	A [Slick-class()](#) object
lang	Optional text string specifying the language (if available). Either 'en', 'es', or 'fr' for English, Spanish, or French respectively
value	A data.frame with the Factors

Value

The Design matrix from an `OMs` object in a `Slick` object

Methods (by class)

- Factors(OMs): Return the operating model Factors table from a [OMs-class()](#) object

- Factors(OMs) <- value: Assign the operating model Factors table to a [OMs-class()](#) object

- Factors(Slick): Access the operating model Factors table from a [Slick-class()](#) object

- Factors(Slick) <- value: Assign the operating model Factors table to a [Slick-class()](#) object

---

FilterSlick   *FilterSlick*

Description

Filter a Slick Object

Usage

```
FilterSlick(slick = NULL, MPs = NULL, OMs = NULL, PIs = NULL, plot = NULL)
```

Arguments

`slick`	An object of class `Slick`
`MPs`	Numeric values of the MPs to keep. Default NULL keeps all MPs.
`OMs`	Numeric values of the OMs to keep (rows of `OM@Design`). Default NULL keeps all OMs.
`PIs`	Numeric values of the PIs in `plot` to keep. Default NULL keeps all PIs.
`plot`	The plot to filter the PIs. One of: `Timeseries`, `Boxplot`, `Kobe`, `Quilt`, `Spider`, or`Tradeoff`

Details

Filter a Slick Object by management procedures (MPs), operating models (OMs), and performance indicators (PIs) for a given plot

Value

A filtered Slick Object

Examples

```
slick <- Slick() # a completed slick object
boxplot_OM_1 <- FilterSlick(slick, OMs=1, plot='boxplot')
```

---

get_casestudies	*Get Case Studies from the SlickLibrary GitHub Repo*

Description

Get Case Studies from the SlickLibrary GitHub Repo

Usage

```
get_casestudies()

download_casestudy(
  name,
  case_studies = NULL,
  dir = NULL,
  silent = FALSE,
  object = TRUE,
  delete = object
)
```

Arguments

`name`	The name of the case study to download. Name column from `get_casestudies()`
`case_studies`	optional. Dataframe returned by `get_casestudies()`
`dir`	Optional. Directory to save the file. Defaults to a temporary directory
`silent`	Logical. Print out messages?
`object`	Logical. Return the Slick object? Default downloads Slick object to a temporary location, loads and returns the Slick object, and then deletes downloaded file.
`delete`	Logical. Delete the downloaded file after function finishes? Only useful if `object = TRUE`

Value

A data.frame for `get_casestudies` and a `Slick` object for `download_casestudy`

The downloaded `Slick` object if `object==TRUE`, otherwise nothing.

Functions

- `download_casestudy()`: download a case study file

Examples

```
case_studies <- get_casestudies()
slick <- download_casestudy(case_studies$Name[1])
```

Kobe *Methods for Creating, Accessing and Assigning* Kobe *objects*

Description

The Kobe function is used both to create and modify an Kobe-class() object. and to access and assign Kobe for an object of class Slick-class(). See `Details`.

Usage

```
Kobe(
  Code = "",
  Label = "",
  Description = "",
  Time = numeric(),
  TimeLab = "Year",
  Value = array(),
  Preset = list(),
  Target = 1,
  Limit = NULL,
```

```
  Defaults = list(),
  TimeTerminal = numeric()
)

Kobe(Slick) <- value

## S4 method for signature 'missing'
Kobe()

## S4 method for signature 'character'
Kobe(
  Code = "",
  Label = "",
  Description = "",
  Time = numeric(),
  TimeLab = "Year",
  Value = array(),
  Preset = list(),
  Target = 1,
  Limit = NULL,
  Defaults = list(),
  TimeTerminal = numeric()
)

## S4 method for signature 'list'
Kobe(
  Code = "",
  Label = "",
  Description = "",
  Time = numeric(),
  TimeLab = "Year",
  Value = array(),
  Preset = list(),
  Target = 1,
  Limit = NULL,
  Defaults = list(),
  TimeTerminal = numeric()
)

## S4 method for signature 'Slick'
Kobe(Code)

## S4 replacement method for signature 'Slick'
Kobe(Slick) <- value
```

Arguments

Code   A *short* code for the Performance Indicators for this object. A character string
                    length nPI or a named list for multi-language support. See `Details`

Label	A short label for the Performance Indicators for this object. Used to label axes on charts. Can be longer than `Code` but recommended to keep short as possible so it shows clearly in plots and tables. A character string length `nPI` or a named list for multi-language support. See `Details`
Description	A description for the Performance Indicators for this object. Can include Markdown, see `Examples`. A character string length `nPI` or a named list for multi-language support. See `Details`
Time	A numeric vector with values for the projection time-steps. Must match length `nTS` in `Value`
TimeLab	Character string length 1. Name of the time step (e.g., 'Year'). Will be used as the label in the `Kobe  Time` plot. Use a named list for multiple languages.
Value	A numeric array with the stochastic performance indicator values for each simulation (sim), operating model (OM), management procedure (MP), performance indicator (PI), and projection time-steps (nTS). Dimensions: c(`nsim`, `nOM`, `nMP`, `nPI`, `nTS`)
Preset	An optional named list for the preset buttons in the [App()](). The name of the list element will appear as a button in the [App()]().
Target	Numeric vector length `nPI` with the target value for the PIs. Defines the color quadrants on the Kobe plot. Defaults to c(1,1).
Limit	Numeric vector length `nPI` with the limit value for the PIs. Shows as red line on Kobe plot. NULL to ignore.
Defaults	A list object with default selections for the Kobe See [Kobe()]()
TimeTerminal	Optional. By default the `Kobe` plot shows the terminal projection year. `TimeTerminal` can be used to override this. Use a numeric value indicating the time (must match a value in `Time`) to use for the Kobe plot
Slick	A [Slick-class()]() object
value	A [Kobe-class()]() object

Details

Objects of class Kobe are created with Kobe()

The Kobe plot typically shows B/BMSY (or something similar) on the x-axis, and F/FMSY (or something similar) on the y-axis.

Performance Indicators:

The first PI will be on the x-axis (usually B/BMSY or something similar) and the second on the y-axis (e.g., F/FMSY)

Multi-Language Support:

Text with multi-language supported can be provided as a named list. Available languages:

- en: English (default)
- es: Spanish
- fr: French

Note:

Character strings in `Code`, `Label`, and `Description` must all be same length as the number of performance indicators (nPIs) in `Value`

Functions

- Kobe(missing): Create an empty Kobe object
- Kobe(character): Create a populated Kobe object
- Kobe(list): Create a populated Kobe object
- Kobe(Slick): Return Kobe from a Slick-class() object
- Kobe(Slick) <- value: Assign a Kobe-class() object to a Slick-class() object

See Also

Code(), Label(), Description(), Time(), [TimeLab(), Value(), Preset(), Target() and Limit()

Examples

```
# Generate dummy values
nsim <- 10
nOM <- 2
nMP <- 4
nPI <- 2
nTS <- 30

values <- array(NA, dim=c(nsim, nOM, nMP, nPI, nTS))

pi_means <- c(1,1)

for (om in 1:nOM) {
  for (mp in 1:nMP) {
    for (pi in 1:nPI) {
      values[,om, mp, pi,] <- pi_means[pi] *
        matrix(
        cumprod(c(rlnorm(nTS*nsim, 0, 0.05))),
        nrow=nsim)
    }
  }
}

# Create and populate Object
kobe <- Kobe(Code=c('B/BMSY', 'F/FMSY'),
             Label=c('B/BMSY',
                     'F/FMSY'),
             Description = c('This is the description for PI 1',
                             'This is the description for PI 2'),
             Value=values
)

# Add values for projection time steps
Time(kobe) <- seq(2025, by=1, length.out=nTS)

# Check
Check(kobe)
```

```
# Add to `Slick` object
slick <- Slick()
Kobe(slick) <- kobe

# Plots
plotKobe(slick)

plotKobe(slick, Time=TRUE)
```

Kobe-class   *S4 class Kobe*

Description

Slots can be accessed and assigned using functions corresponding to slot name. See See Also section below.

Details

Objects of class Kobe are created with Kobe()

Performance Indicators:

By default, the first PI will be on the x-axis (usually B/BMSY or something similar) and the second on the y-axis (e.g., F/FMSY). The Slick App() provides drop down menus for selecting other PIs.

Multi-Language Support:

Text with multi-language supported can be provided as a named list. Available languages:

- en: English (default)
- es: Spanish
- fr: French

Note:

Character strings in Code, Label, and Description must all be same length as the number of performance indicators (nPIs) in Value

Slots

Code A *short* code for the Performance Indicators for this object. A character string length nPI or a named list for multi-language support. See Details

Label A short label for the Performance Indicators for this object. Used to label axes on charts. Can be longer than Code but recommended to keep short as possible so it shows clearly in plots and tables. A character string length nPI or a named list for multi-language support. See Details

Description A description for the Performance Indicators for this object. Can include Markdown, see Examples. A character string length nPI or a named list for multi-language support. See Details

Time A numeric vector with values for the projection time-steps. Must match length nTS in Value

TimeLab Character string length 1. Name of the time step (e.g., 'Year'). Will be used as the label in the Kobe Time plot. Use a named list for multiple languages.

Value A numeric array with the stochastic performance indicator values for each simulation (sim), operating model (OM), management procedure (MP), performance indicator (PI), and projection time-steps (nTS) Dimensions: c(nsim, nOM, nMP, nPI, nTS)

Preset An optional named list for the preset buttons in the [App()](). The name of the list element will appear as a button in the [App()]().

Target Numeric vector length nPI with the target value for the PIs. Defines the color quadrants on the Kobe plot. Defaults to 1.

Limit Numeric vector length nPI with the limit value for the two PIs. Shows as red line on Kobe plot. NULL to ignore.

Defaults A list object with default selections for the Kobe See [Kobe()]()

TimeTerminal Optional. By default the Kobe plot shows the terminal projection year. TimeTerminal can be used to override this. Use a numeric value indicating the time (must match a value in Time) to use for the Kobe plot

See Also

[Kobe()](), [Code()](), [Label()](), [Description()](), [Value()](), [Preset()]()

Examples

```
# Generate dummy values
nsim <- 10
nOM <- 2
nMP <- 4
nPI <- 2
nTS <- 30

values <- array(NA, dim=c(nsim, nOM, nMP, nPI, nTS))

pi_means <- c(1,1)

for (om in 1:nOM) {
  for (mp in 1:nMP) {
    for (pi in 1:nPI) {
      values[,om, mp, pi,] <- pi_means[pi] *
        matrix(
        cumprod(c(rlnorm(nTS*nsim, 0, 0.05))),
        nrow=nsim)
    }
  }
}

# Create and populate Object
kobe <- Kobe(Code=c('B/BMSY', 'F/FMSY'),
            Label=c('B/BMSY',
                    'F/FMSY'),
```

```
                Description = c('This is the description for PI 1',
                                'This is the description for PI 2'),
                Value=values
)

# Add values for projection time steps
Time(kobe) <- seq(2025, by=1, length.out=nTS)

# Check
Check(kobe)

# Add to `Slick` object
slick <- Slick()
Kobe(slick) <- kobe

# Plots
plotKobe(slick)

plotKobe(slick, Time=TRUE)
```

Limit	*Access or assign* Target *and* Limit *for object of class* Kobe *or* Timeseries

---

Description

Access or assign `Target` and `Limit` for object of class `Kobe` or `Timeseries`

Usage

```
Limit(object)

Limit(object) <- value

Target(object)

Target(object) <- value

## S4 method for signature 'Kobe'
Limit(object)

## S4 replacement method for signature 'Kobe'
Limit(object) <- value

## S4 method for signature 'Kobe'
Target(object)

## S4 replacement method for signature 'Kobe'
```

```
Target(object) <- value

## S4 method for signature 'Timeseries'
Limit(object)

## S4 replacement method for signature 'Timeseries'
Limit(object) <- value

## S4 method for signature 'Timeseries'
Target(object)

## S4 replacement method for signature 'Timeseries'
Target(object) <- value
```

Arguments

object	A Kobe-class() or Timeseries() class object
value	Value to assign to Target

Value

Returns a numeric vector with the contents of the Target or Limit slots of Kobe() and Timeseries() objects

Methods (by class)

- Limit(Kobe): Return Limit from a Kobe-class() object

- Limit(Kobe) <- value: Assign Limit to a Kobe-class() object

- Target(Kobe): Return Target from a Kobe-class() object

- Target(Kobe) <- value: Assign Target to a Kobe-class() object

- Limit(Timeseries): Return Limit from a Timeseries-class() object

- Limit(Timeseries) <- value: Assign Limit to a Timeseries-class() object

- Target(Timeseries): Return Target from a Timeseries-class() object

- Target(Timeseries) <- value: Assign Target to a Timeseries-class() object

Functions

- Target(object) <- value: Assign value to object@Target

Make_Slick   *Make an example Slick object*

Description

A function that creates an example Slick data object

Usage

```
Make_Slick(
  name = "Unnamed Slick object",
  OM = NULL,
  MPs = c("DCAC", "AvC", "Fratio", "FMSYref", "FMSYref50", "matlenlim"),
  MP_Desc = NULL,
  PMs = c("AAVE", "AAVY", "LTY", "P10", "P50", "P100", "PNOF", "STY", "Yield"),
  Design = as.data.frame(cbind(rbind(as.matrix(expand.grid(1:2, 1:3, 1:2)), matrix(1,
    nrow = 5, ncol = 3)), c(rep(1, 12), 2:6))),
  SN = list(Factor_Labels = c("Natural Mortality", "Resilience", "Stock Depletion",
   "Robustness"), Labels = list(c("M=0.2", "M=0.3"), c("h=0.5", "h=0.7", "h=0.9"),
    c("Dep=0.1", "Dep=0.3"), c("Ref_Case", "L50=0.5", "Vmaxlen=0.1", "Cobs=0.5",
   "Perr=0.5", "AC=0.95")), Codes = list(c("M2", "M3"), c("h5", "h7", "h9"), c("D1",
   "D3"), c("Ref_case", "mat_low", "dome", "h_Cerr", "h_Perr", "h_AC")), Description =
   list(c("M=0.2", "M=0.3"), c("h=0.5", "h=0.7", "h=0.9"), c("Dep=0.1", "Dep=0.3"),
    c("Reference Case",
     "L50=0.5", "Vmaxlen=0.1", "Cobs=0.5", "Perr=0.5",
     "AC=0.95"))),
  mods = list(function(OM, lev) {
     if (lev == 1) OM@M <- c(0.2, 0.2)
     if (lev ==
     2) OM@M <- c(0.3, 0.3)
      OM
  }, function(OM, lev) {
     if (lev == 1) OM@h <-
     c(0.5, 0.5)
     if (lev == 2) OM@h = c(0.7, 0.7)
     if (lev == 3) OM@h = c(0.9,
     0.9)
      OM
  }, function(OM, lev) {
     if (lev == 1) OM@D <- c(0.1, 0.1)
      if
     (lev == 2) OM@D <- c(0.3, 0.3)
      OM
  }, function(OM, lev) {
     if (lev == 2)
     OM@L50 <- c(0.5, 0.5)
     if (lev == 3) OM@Vmaxlen = c(0.1, 0.1)
```

```
     if (lev == 4)
   OM@Cobs = c(0.5, 0.5)
    if (lev == 5) OM@Perr = c(0.5, 0.5)
    if (lev == 6)
   OM@AC = c(0.9, 0.9)
     OM
 }),
  nsim = 48,
  MSElist = NULL,
  fstYr = NULL,
  returnMSEs = FALSE
)
```

Arguments

name	Character string that is the object name (abbreviated for use in menus etc)
OM	An operating model object (class 'OM')
MPs	A vector of methods (character string) of class MP
MP_Desc	A vector method descriptions (character string) nMPs long
PMs	A vector of performance metrics of class PM
Design	A design matrix of OM runs
SN	A list of Labels, Codes and Descriptions of the factor levels. Each list item is a factor containing a vector of factor levels.
mods	A nested list of mods
nsim	Integer, the number of simulations
MSElist	An optional list of prerun MSEs
fstYr	An optional numeric value for first projection year. Otherwise current year is used
returnMSEs	Logical, rather than the Slick object should the list of MSEs be returned?

Value

An object of class Slick

Author(s)

T. Carruthers

Metadata	*Return* Code, Label, Description *and other information from an object*

Description

Return Code, Label, Description and other information from an object

Usage

```
Metadata(object, lang = "en")

Metadata(object) <- value

## S4 method for signature 'Boxplot'
Metadata(object, lang = "en")

## S4 replacement method for signature 'Boxplot'
Metadata(object) <- value

## S4 method for signature 'Kobe'
Metadata(object, lang = "en")

## S4 replacement method for signature 'Kobe'
Metadata(object) <- value

## S4 method for signature 'MPs'
Metadata(object, lang = "en")

## S4 replacement method for signature 'MPs'
Metadata(object) <- value

## S4 method for signature 'Quilt'
Metadata(object, lang = "en")

## S4 replacement method for signature 'Quilt'
Metadata(object) <- value

## S4 method for signature 'Tradeoff'
Metadata(object, lang = "en")

## S4 replacement method for signature 'Tradeoff'
Metadata(object) <- value

## S4 method for signature 'Timeseries'
Metadata(object, lang = "en")
```

```
## S4 replacement method for signature 'Timeseries'
Metadata(object) <- value

## S4 method for signature 'Spider'
Metadata(object, lang = "en")

## S4 replacement method for signature 'Spider'
Metadata(object) <- value

## S4 method for signature 'Slick'
Metadata(object, lang = "en")
```

Arguments

object	A `Slick-class()`, `MPs-class()`, `Boxplot-class()`, `Kobe-class()`, `Quilt-class()`, `Spider-class()`, `Timeseries-class()`, or `Tradeoff-class()` object
lang	Optional text string specifying the language (if available). Either 'en', 'es', or 'fr' for English, Spanish, or French respectively
value	Replacement value for `Metadata()` in the corresponding `object`. See help documentation for the relevant object class for details.

Value

A `data.frame`

A `data.frame`

Methods (by class)

- `Metadata(Boxplot)`: Return Metadata for `Boxplot-class()` objects
- `Metadata(Boxplot) <- value`: Assign Metadata for `Boxplot-class()` objects
- `Metadata(Kobe)`: Return Metadata for `Kobe-class()` objects
- `Metadata(Kobe) <- value`: Assign Metadata for `Kobe-class()` objects
- `Metadata(MPs)`: Return Metadata for `MPs-class()` objects
- `Metadata(MPs) <- value`: Assign Metadata for `MPs-class()` objects
- `Metadata(Quilt)`: Return Metadata for `Quilt-class()` objects
- `Metadata(Quilt) <- value`: Assign Metadata for `Quilt-class()` objects
- `Metadata(Tradeoff)`: Return Metadata for `Tradeoff-class()` objects
- `Metadata(Tradeoff) <- value`: Assign Metadata for `Tradeoff-class()` objects
- `Metadata(Timeseries)`: Return Metadata for `Timeseries-class()` objects
- `Metadata(Timeseries) <- value`: Assign Metadata for `Timeseries-class()` objects
- `Metadata(Spider)`: Return Metadata for `Spider-class()` objects
- `Metadata(Spider) <- value`: Assign Metadata for `Spider-class()` objects
- `Metadata(Slick)`: Return `Author`, `Email`, and `Institution` from `Slick()` objects

MinValue *Assign and access* MinValue *and* MaxValue *for a* Quilt *object*

Description

Assign and access MinValue and MaxValue for a Quilt object

Usage

```
MinValue(Quilt)

MinValue(Quilt) <- value

MaxValue(Quilt)

MaxValue(Quilt) <- value
```

Arguments

Quilt   A Quilt-class() object

value   A numeric vector with minimum or maximum values for the Performance Indicators.

Value

A numeric value

Functions

- MinValue(): Return MinValue from a Quilt-class() object

- MinValue(Quilt) <- value: Assign MinValue to a Quilt-class() object

- MaxValue(): Return MaxValue from a Quilt-class() object

- MaxValue(Quilt) <- value: Assign MaxValue to a Quilt-class() object

See Also

Quilt()

MPs   *Methods for Creating, Accessing and Assigning* MPs *objects*

Description

The MPs function is used both to create and modify an `MPs-class()` object. and to access and assign MPs for an object of class `Slick-class()`. See `Details`.

Usage

```
MPs(Code = "", Label = "", Description = "", Color = "", Preset = list())

MPs(object) <- value

## S4 method for signature 'missing'
MPs()

## S4 method for signature 'character_list'
MPs(Code = "", Label = "", Description = "", Color = "", Preset = list())

## S4 method for signature 'Slick'
MPs(Code)

## S4 replacement method for signature 'Slick'
MPs(object) <- value
```

Arguments

Code	A *short* code for the Management Procedures in this `Slick` object. A character string length nMP or a named list for multi-language support. See `Details`
Label	A short label for the Management Procedures in this `Slick` object. Can be longer than `Code` but recommended to keep short as possible so it shows clearly in plots and tables. A character string length nMP or a named list for multi-language support. See `Details`
Description	A description for the Management Procedures in this `Slick` object. Can include Markdown, see `Examples`. A character string length nMP or a named list for multi-language support. See `Details`
Color	A character vector of colors for the MPs.
Preset	An optional named list for the preset buttons in the `App()`. The name of the list element will appear as a button in the `App()`. Use `Code()`, `Label()`, `Description()`, and `Preset()` to access and assign the values for an existing MPs object, see `Examples`.
object	A `Slick-class()` object
value	A `MPs-class()` object

Details

Objects of class MPs are created with `MPs()`

Functions

- `MPs(missing)`: Create an empty MPs object

- `MPs(character_list)`: Create a populated MPs object

- `MPs(Slick)`: Return an `MPs-class()` object from a `Slick()` object

- `MPs(Slick) <- value`: Assign an `MPs-class()` object to a `Slick()` object

See Also

`Code()`, `Label()`, `Description()`, `Color()`, `Metadata()`, `Preset()`

Examples

```
myMPs <- MPs()
Code(myMPs) <- c('MP1', 'MP2', 'MP3')
Label(myMPs) <- c('Management Procedure 1',
                  'Management Procedure 2',
                  'Management Procedure 3')
Description(myMPs) <- c('This is the description for Management Procedure 1',
                        'This is the description for Management Procedure 2',
                        'This is the description for Management Procedure 3')

Preset(myMPs) <- list(All=1:3, FirstTwo=1:2)


myMPs

# Multi-language
Description(myMPs) <- list(en=c('This is the English description for Management Procedure 1',
                           'This is the English description for Management Procedure 2',
                          'This is the English description for Management Procedure 3'),
                      es=c("This is the Spanish description for Management Procedure 1",
                          "This is the Spanish description for Management Procedure 2",
                         "This is the Spanish description for Management Procedure 3"),
                     fr=c("This is the French description for Management Procedure 1",
                         "This is the French description for Management Procedure 2",
                         "This is the French description for Management Procedure 3")
                         )

Metadata(myMPs)
Metadata(myMPs, 'es')
Metadata(myMPs, 'fr')
```

MPs-class   MPs *S4 class and functions*

Description

An object of class MPs contains information about the management procedures (MPs) in a `Slick-class()` object. Like all S4 objects in Slick, slots in this object can be accessed and assigned using functions corresponding to slot name. See `MPs()` and the the See Also section below.

Details

Objects of class MPs are created with MPs()

Multi-Language Support:

Text with multi-language supported can be provided as a named list. Available languages:

- en: English (default)
- es: Spanish
- fr: French

Note:

Character strings in Code, Label, and Description must all be same length as the number of management procedures (nMPs) in the plot objects Boxplot, Kobe, Quilt, Spider, Timeseries, and Tradeoff.

Slots

Code A *short* code for the Management Procedures in this Slick object. A character string length nMP or a named list for multi-language support. See Details *Required*

Label A short label for the Management Procedures in this Slick object. Can be longer than Code but recommended to keep short as possible so it shows clearly in plots and tables. A character string length nMP or a named list for multi-language support. See Details *Required*

Description A description for the Management Procedures in this Slick object. Can include Markdown, see Examples. A character string length nMP or a named list for multi-language support. See Details

Color A character vector of colors for the MPs. Defaults will be used if not populated

Preset An optional named list for the preset buttons in the `App()`. The name of the list element will appear as a button in the `App()`.

See Also

`Code()`, `Label()`, `Description()`, `Preset()`

Examples

```
myMPs <- MPs()
Code(myMPs) <- c('MP1', 'MP2', 'MP3')
Label(myMPs) <- c('Management Procedure 1',
                  'Management Procedure 2',
                  'Management Procedure 3')
Description(myMPs) <- c('This is the description for Management Procedure 1',
                        'This is the description for Management Procedure 2',
                        'This is the description for Management Procedure 3')


Preset(myMPs) <- list(All=1:3, FirstTwo=1:2)



myMPs

# Multi-language
Description(myMPs) <- list(en=c('This is the English description for Management Procedure 1',
                              'This is the English description for Management Procedure 2',
                              'This is the English description for Management Procedure 3'),
                          es=c("This is the Spanish description for Management Procedure 1",
                              "This is the Spanish description for Management Procedure 2",
                              "This is the Spanish description for Management Procedure 3"),
                          fr=c("This is the French description for Management Procedure 1",
                              "This is the French description for Management Procedure 2",
                              "This is the French description for Management Procedure 3")
                          )

Metadata(myMPs)
Metadata(myMPs, 'es')
Metadata(myMPs, 'fr')
```

NewSlick   *Creates a blank Slick object*

Description

Creates a blank Slick object

Usage

```
NewSlick(
  name = "Unnamed Slick Object",
  nPerf = list(nD = 5, nS = 6, nP = 7),
  nMPs = 5,
  nsim = 10,
  nProjYr = 50,
  nStateVar = 2,
  nHistYr = 55,
  Design = expand.grid(1:2, 1:2)
)
```

Arguments

name	Character string that is the object name (shortened for use in menus etc.)
nPerf	An integer vector of the number of deterministic (nD), stochastic (nS) and projected (nP) performance metrics
nMPs	Integer, the number of management options (aka management procedures).
nsim	Integer, the number of simulations (stochastic replicates per state of nature)
nProjYr	Integer, the number of projected years
nStateVar	Integer, the number of state variables
nHistYr	Integer, the number of historical years for state variables
Design	A design matrix of factor levels SN, factor

Value

An object of class Slick

Author(s)

T. Carruthers

OMs   *Methods for Creating, Accessing and Assigning* OMs *objects*

Description

The OMs function is used both to create and modify an OMs-class() object. and to access and assign OMs for an object of class Slick-class(). See Details.

Usage

```
OMs(Factors = data.frame(), Design = data.frame(), Preset = list())

OMs(object) <- value

## S4 method for signature 'missing'
OMs()

## S4 method for signature 'dataframe_list'
OMs(Factors = data.frame(), Design = data.frame(), Preset = list())

## S4 method for signature 'Slick'
OMs(Factors)

## S4 replacement method for signature 'Slick'
OMs(object) <- value
```

Arguments

Factors	A data.frame with column headings Factor, Level, and Description. See Details
Design	A data.frame with nFactor columns (i.e., length(unique(Factors$Factor))), and nOM rows. See Details
Preset	An optional named list for the preset buttons in the App(). The name of the list element will appear as a button in the App().. See Details and 'Examples
object	A Slick-class() object
value	A OMs-class() object

Details

Factors:

Factors can be accessed and assigned using Factors(myslick) and Factors(myslick) <- data.frame() respectively.

The Factor column should be character strings with the name of each factor, while the Level column is a numeric or character value with the level for the corresponding factor.

The Description column is a description for each row, i.e., a unique factor and level. See Examples.

Design:

The Design matrix is nOM rows and nFactor columns. The values in each column should either be numeric values indicating the levels for the corresponding factor, or the actual level values (i.e., Factors$Level) that correspond to each OM. See Examples.

Preset:

For OMs objects, Preset should be a named list, where each list element represents a different preset button to be shown in the app by the name of the list element, and each named list element should be a list of length nFactors, with the list elements for each Factor containing numeric values indicating the levels to include for that factor. See Examples

Use Factors(), Design(), and Preset() to access and assign the values for an existing OMs object, see Examples.

Functions

- OMs(missing): Create an empty OMs object

- OMs(dataframe_list): Create a populated OMs object

- OMs(Slick): Return an OMs-class() object from a Slick() object

- OMs(Slick) <- value: Assign an OMs-class() object to a Slick() object

See Also

OMs-class(), Factors(), Design(), Preset()

Examples

```
# Create Object
oms <- OMs()

# Specify Factors
Factors(oms) <- data.frame(Factor='M',
                           Level=c('Base', 'Low M', 'High M'),
                           Description=c('Base Case',
                                                 'Lower Natural Mortality',
                                                 'Higher Natural Mortality')
)

Factors(oms)

# OM Design

Design(oms) <- data.frame(M=c('Base', 'Low M', 'High M'))

# Add names for OMs
rownames(Design(oms)) <- c('Base Case', 'Less Productive', 'More Productive')

Design(oms)

# Preset

Preset(oms) <- list('Base Case'=list(1),
                    'Low M' = list(2),
                    'High M' = list(3),
                    'All'= list(1:3)
)

# Create Slick Object
myslick <- Slick()

# Add OMs to Slick Object
OMs(myslick) <- oms
```

OMs-class   OMs *S4 class and functions*

Description

An object of class `OMs` contains information about the operating models (MPs) in the [Slick()](#)
object. Like all S4 objects in `Slick`, slots in this object can be accessed and assigned using functions
corresponding to slot name. See [OMs()](#) and the the See Also section below.

Details

Multi-Language Support:

Use a named list to use multi-languages in `Factors`

Factors:

Factors can be accessed and assigned using Factors(myslick) and Factors(myslick) <-data.frame() respectively.

The Factor column should be character strings with the name of each factor, while the Level column is a numeric or character value with the level for the corresponding factor.

The Description column is a description for each row, i.e., a unique factor and level. See Examples.

Design:

The Design matrix is nOM rows and nFactor columns. The values in each column should either be numeric values indicating the levels for the corresponding factor, or the actual level values (i.e., Factors$Level) that correspond to each OM. See Examples.

Preset:

For OMs objects, Preset should be a named list, where each list element represents a different preset button to be shown in the app by the name of the list element, and each named list element should be a list of length nFactors, with the list elements for each Factor containing numeric values indicating the levels to include for that factor. See Examples

Slots

Factors A data.frame with column headings Factor, Level, and Description. See Details

Design A data.frame with nFactor columns (i.e., length(unique(Factors$Factor))), and nOM rows. See Details

Preset An optional named list for the preset buttons in the App(). The name of the list element will appear as a button in the App().. See Details and Examples

See Also

OMs(), Factors(), Design(), Preset()

Examples

```
# Create Object
oms <- OMs()

# Specify Factors
Factors(oms) <- data.frame(Factor='M',
                           Level=c('Base', 'Low M', 'High M'),
                           Description=c('Base Case',
                                         'Lower Natural Mortality',
                                         'Higher Natural Mortality')
)

Factors(oms)

# OM Design

Design(oms) <- data.frame(M=c('Base', 'Low M', 'High M'))
```

```
# Add names for OMs
rownames(Design(oms)) <- c('Base Case', 'Less Productive', 'More Productive')

Design(oms)

# Preset

Preset(oms) <- list('Base Case'=list(1),
                    'Low M' = list(2),
                    'High M' = list(3),
                    'All'= list(1:3)
)

# Create Slick Object
myslick <- Slick()

# Add OMs to Slick Object
OMs(myslick) <- oms
```

---

plotBoxplot   *Plot* Boxplot

Description

Plots boxplot, violin plot, or a combined box+violin plot for information stored in a Boxplot object

Usage

```
plotBoxplot(
  slick,
  PI = NULL,
  type = c("boxplot", "violin", "both", "all"),
  byOM = FALSE,
  ncol = 4,
  MP_label = "Code",
  PI_label = "Code"
)
```

Arguments

slick	A `Slick-class()` object
PI	Numeric value indicating the Performance Indicator(s) to plot from the `Boxplot-class` object. If NULL, it will facet_wrap all PIs
type	Character string specifying the plot type.
byOM	Logical. Facet the plots by operating model? PI must be a single value
ncol	Numeric. Number of columns

MP_label	Label to use for the MPs. Either Code or Label. Description works as well, but you probably don't want to do that.
PI_label	Label to use for the PIs. Either Code or Label. Description works as well, but you probably don't want to do that.

Value

A ggplot2 object, or a list of ggplot2 objects

Examples

```
# Generate dummy values
nsim <- 10
nOM <- 2
nMP <- 4
nPI <- 3

values <- array(NA, dim=c(nsim, nOM, nMP, nPI))
pi_means <- runif(nPI, 5, 50)
for (om in 1:nOM) {
  for (mp in 1:nMP) {
    for (pi in 1:nPI) {
      values[,om, mp, pi] <- rlnorm(nsim, log(pi_means[pi]), 0.4)
    }
  }
}

# Create and populate Object
boxplot <- Boxplot(Code=c('PI1', 'PI2', 'PI3'),
                   Label=c('Performance Indicator 1',
                           'Performance Indicator 2',
                           'Performance Indicator 3'),
                 Description = c('This is the description for PI 1',
                                 'This is the description for PI 2',
                                 'This is the description for PI 3'),
                   Value=values)

# Check
Check(boxplot)

# Add to `Slick` object
slick <- Slick()
Boxplot(slick) <- boxplot

# Plots
plotBoxplot(slick)

plotBoxplot(slick, type='violin')

plotBoxplot(slick, byOM=TRUE)

plotBoxplot(slick, 2, type='both', byOM=TRUE)
```

plotKobe   *Plot* Kobe

Description

Plots a Kobe plot for a given projection year, or a Kobe Time plot.

Usage

```
plotKobe(
  slick,
  xPI = 1,
  yPI = 2,
  Time = FALSE,
  BLcol = "#F8DC7A",
  TLcol = "#D8775D",
  TRcol = "#FDBD56",
  BRcol = "#67C18B",
  axis_label = "Code",
  percentile = 0.9,
  axis.text.size = 14,
  axis.title.size = 16,
  strip.text.size = 16,
  strip.text.color = "#D6501C",
  incMP_label = TRUE,
  mp.text.size = 7,
  mp.point.size = 4,
  mp.init.point.size = 2,
  xmax = 2,
  ymax = 2,
  hist_traj = FALSE,
  ncol = 4,
  lang = "en",
  MP_label = "Code"
)
```

Arguments

slick	A [Slick-class()](#) object
xPI	Numeric value specifying the performance indicator for the x-axis
yPI	Numeric value specifying the performance indicator for the y-axis
Time	Logical. Kobe Time plot?
BLcol	Color for the bottom left quadrant
TLcol	Color for the top left quadrant

TRcol	Color for the top right quadrant
BRcol	Color for the bottom right quadrant
axis_label	Label to use for the axes. Either `Code` or `Label`. `Description` works as well, but you probably don't want to do that.
percentile	Numeric value specifying the percentile for the x and y percentile bars. Use NULL to remove percentile lines.
axis.text.size	Font size for axis text
axis.title.size	
	Font size for axis title
strip.text.size	
	Font size for facet strip text
strip.text.color	
	Color for facet strip text
incMP_label	Logical. Include MP labels?
mp.text.size	Font size for MP labels
mp.point.size	Point size for MP labels
mp.init.point.size	
	Point size for start of trajectory. If `hist_traj==TRUE`
xmax	Maximum value for the x-axis. Values greater than xmax will be shown at xmax
ymax	Maximum value for the yx-axis. Values greater than ymax will be shown at ymax
hist_traj	Logical. Plot the historical trajectories?
ncol	Numeric. Number of columns for Kobe Time
lang	Optional. Language (if supported in Slick Object). Either 'en', 'es', 'fr'
MP_label	Label to use for the MPs. Either `Code` or `Label`. `Description` works as well, but you probably don't want to do that.

Details

By default `plotKobe` shows the terminal projection year. `TimeTerminal(Kobe)` can be used to override this. Use a numeric value indicating the time (must match a value in `Time(Kobe)`) to use for the Kobe plot.

Value

A `ggplot2` object

See Also

[Kobe()](), [Kobe-class()]()

Examples

```
# Generate dummy values
nsim <- 10
nOM <- 2
nMP <- 4
nPI <- 2
nTS <- 30

values <- array(NA, dim=c(nsim, nOM, nMP, nPI, nTS))

pi_means <- c(1,1)

for (om in 1:nOM) {
  for (mp in 1:nMP) {
    for (pi in 1:nPI) {
      values[,om, mp, pi,] <- pi_means[pi] *
        matrix(
        cumprod(c(rlnorm(nTS*nsim, 0, 0.05))),
        nrow=nsim)
    }
  }
}

# Create and populate Object
kobe <- Kobe(Code=c('B/BMSY', 'F/FMSY'),
             Label=c('B/BMSY',
                     'F/FMSY'),
             Description = c('This is the description for PI 1',
                             'This is the description for PI 2'),
             Value=values
)

# Add values for projection time steps
Time(kobe) <- seq(2025, by=1, length.out=nTS)

# Check
Check(kobe)

# Add to `Slick` object
slick <- Slick()
Kobe(slick) <- kobe

# Plots
plotKobe(slick)

plotKobe(slick, Time=TRUE)
```

---

plotQuilt   *Plot* Quilt

Description

Create a Quilt plot (unless shading==FALSE in which case it's just a table)

Usage

```
plotQuilt(
  slick,
  MP_label = "Code",
  minmax = FALSE,
  shading = TRUE,
  kable = FALSE,
  signif = 2,
  alpha = 0.5
)
```

Arguments

slick	A [Slick-class()] object
MP_label	Label to use for the MPs. Either `Code` or `Label`. `Description` works as well, but you probably don't want to do that.
minmax	Logical. Color shading from min to max values in each column? If TRUE, ignores `MinValue(quilt)` and `MaxValue(quilt)`
shading	Logical. Color shading for the columns?
kable	Logical. Return a kable object?
signif	Numeric Number of significant figures
alpha	Numeric value. Transparency for color shading

Details

The columns are color shaded from light (lowest values) to dark (highest values).

Colors are set in `Color(quilt)`.

The color shading has 10 steps, from `MinValue(quilt)` to `MaxValue(quilt)` for each Performance Indicator. If those values are missing (NA) for a given PI, colors are shaded from lowest to highest values. If minmax==TRUE, `MinValue(quilt)` and `MaxValue(quilt)` are ignored.

Value

A `DT::datatable` or a `knitr::kable` object

See Also

[Quilt()], [Quilt-class()]

Examples

```
# Generate dummy values
nOM <- 2
nMP <- 4
nPI <- 4

values <- array(NA, dim=c(nOM, nMP, nPI))

pi_means <- runif(nPI, 5, 50)
for (om in 1:nOM) {
  for (mp in 1:nMP) {
    for (pi in 1:nPI) {
      values[om, mp, pi] <- rlnorm(1,log(pi_means[pi]), 0.4)
    }
  }
}

# Create and populate Object
quilt <- Quilt(Code=c('PI1', 'PI2', 'PI3', 'PI4'),
               Label=c('Performance Indicator 1',
                       'Performance Indicator 2',
                       'Performance Indicator 3',
                       'Performance Indicator 4'),
               Description = c('This is the description for PI 1',
                               'This is the description for PI 2',
                               'This is the description for PI 3',
                               'This is the description for PI 4'),
               Value=values)

# Check
Check(quilt)

# Add to `Slick` object
slick <- Slick()
Quilt(slick) <- quilt

# Plots
plotQuilt(slick)
```

---

plotSpider	*Plot* Spider

Description

A Spider or Radar plot

Usage

```
plotSpider(
  slick,
  byOM = FALSE,
  byMP = FALSE,
  incMean = TRUE,
  incMax = TRUE,
  relScale = FALSE,
  col.om.title = "#D6501C",
  cex.om.title = 2,
  col.Mean = "white",
  bg.Mean = "black",
  grid.fill = "#f2f3f5",
  grid.line = "black",
  fill = byMP | all(byOM),
  inc.grid = TRUE,
  PI.labels = !(byMP | all(byOM)),
  MP_label = "Code",
  mp.lwd = 3,
  alpha = 0.3,
  ncol = 4,
  PI.mean.cex = 2.2,
  incMPtitle = byMP,
  mplab.cex = 2.2,
  max.pt.cex = 2,
  max.pt.col = "darkred"
)
```

Arguments

slick	A [Slick-class()](#) object
byOM	Logical Plot by OM? Otherwise mean over OMs
byMP	Logical. Plot by MP? Otherwise all MPs together on one plot
incMean	Logical. Include mean PI score in center?
incMax	Logical. Include colored points indicating maximum PI values?
relScale	Logical. Scale PI values between minimum (0) and maximum (1)?
col.om.title	Color of the OM names/labels
cex.om.title	Size of OM names/labels
col.Mean	Color of mean value text
bg.Mean	Background color of mean value text
grid.fill	Color of fill for the spider grid
grid.line	Color of lines for the spider grid
fill	Logical Fill the spider plot?
inc.grid	Logical. Include the grid?

PI.labels	Logical Show PI labels?
MP_label	Label to use for the MPs. Either `Code` or `Label`. `Description` works as well, but you probably don't want to do that.
mp.lwd	Line width
alpha	Alpha value for the fill
ncol	Number of columns
PI.mean.cex	Size of PI mean score text
incMPtitle	Logical. Include MP label?
mplab.cex	MP label size
max.pt.cex	Max value point size
max.pt.col	Max value point color

Value

A `ggplot2` object

Examples

```
# Generate dummy values
nOM <- 2
nMP <- 4
nPI <- 4

values <- array(NA, dim=c(nOM, nMP, nPI))

# Note: PI values must be between 0 and 1, with 1 indicating better performance
pi_means <- runif(nPI, 0, 1)
for (om in 1:nOM) {
  for (mp in 1:nMP) {
    for (pi in 1:nPI) {
      values[om, mp, pi] <- runif(1, pi_means[pi])
    }
  }
}

# Create and populate Object
spider <- Spider(Code=c('PI1', 'PI2', 'PI3', 'PI4'),
              Label=c('Performance Indicator 1',
                      'Performance Indicator 2',
                      'Performance Indicator 3',
                      'Performance Indicator 4'),
              Description = c('This is the description for PI 1',
                              'This is the description for PI 2',
                              'This is the description for PI 3',
                              'This is the description for PI 4'),
              Value=values)

# Check
Check(spider)
```

```
# Add to `Slick` object
slick <- Slick()
Spider(slick) <- spider

# Plots
plotSpider(slick)

plotSpider(slick, fill=TRUE)

plotSpider(slick, byMP=TRUE)

plotSpider(slick, byOM=TRUE)
```

---

plotTimeseries   *Plot* Timeseries

Description

Plot the historical and projected values for a performance indicator.

Usage

```
plotTimeseries(
  slick,
  PI = 1,
  byMP = FALSE,
  byOM = FALSE,
  includeHist = TRUE,
  ncol = 4,
  col_line = "darkgray",
  includeQuants = TRUE,
  includeLabels = TRUE,
  MeanMed = c("mean", "median"),
  fill_ribbon1 = "#ededed",
  col_ribbon1 = "#ededed",
  quants1 = c(0.25, 0.75),
  alpha1 = 0.3,
  fill_ribbon2 = "white",
  col_ribbon2 = "#c9c9c9",
  linetype_ribbon2 = "dashed",
  quants2 = c(0.1, 0.9),
  alpha2 = 0.1,
  MP_label = "Code",
  col_title = "#D6501C",
  size.title = 18,
  size.axis.title = 18,
```

```
    size.axis.text = 16,
    size.mp.label = 6,
    linewidth.median.line = 0.5,
    targ_color = "green",
    targ_name = "Target",
    lim_color = "red",
    lim_name = "Limit",
    inc_y_label = TRUE,
    sims = NULL,
    lang = "en"
)
```

Arguments

slick	A [Slick-class()](#) object
PI	A numeric value specifying the performance indicator to plot
byMP	Logical. Facet by MP? Defaults to FALSE, where all MPs are shown on the same plot
byOM	Logical. Facet by OM? Defaults to FALSE where values are calculated as mean across OMs
includeHist	Logical. Include the historical period in the projections?
ncol	Numeric. Number of columns if faceting by MP or OM
col_line	Color for the median line (historical)
includeQuants	Logical. Include quantile shading for the projections?
includeLabels	Logical. Include MP labels?
MeanMed	Character. Plot the 'mean' (default) or 'median'.
fill_ribbon1	Fill color for the inner ribbon
col_ribbon1	Color of the line for inner ribbon
quants1	Quantiles for the inner ribbon. Numeric length 2
alpha1	Alpha for the colored MPs inner shading
fill_ribbon2	Fill color for the outer ribbon
col_ribbon2	Color of the line for outer ribbon
linetype_ribbon2	
	Line type for outer ribbon
quants2	Quantiles for the outer ribbon. Numeric length 2.
alpha2	Alpha for the colored MPs outer shading
MP_label	Label to use for the MPs. Either Code or Label. Description works as well, but you probably don't want to do that.
col_title	Color of the MP title (if byMP==TRUE)
size.title	Numeric length 1. Size for plot title
size.axis.title	
	Numeric length 1. Size for axis title

size.axis.text	Numeric length 1. Size for axis text
size.mp.label	Numeric length 1. Size of MP labels. Set to NULL for no MP labels
linewidth.median.line	
	Width of the mean/median line
targ_color	Color for the target line (if it exists in `Target(Timeseries(slick))`)
targ_name	Label for the target line
lim_color	Color for the limit line (if it exists in `Limit(Timeseries(slick))`)
lim_name	Label for the limit line
inc_y_label	Include the label for the y-axis?
sims	Optional. Numeric values indicating the simulations to include. Defaults to all.
lang	Optional. Language (if supported in Slick Object). Either 'en', 'es', 'fr'

Details

If `byOM==FALSE` the results are shown as the mean across operating models.

Value

A `ggplot2` object

See Also

[`Timeseries()`](), [`Timeseries-class()`]()

Examples

```
# Generate dummy values
nsim <- 10
nOM <- 2
nMP <- 4
nPI <- 3
nHistTS <- 50
nProjTS <- 30
nTS <- nHistTS + nProjTS

set.seed(101)

values <- array(NA, dim=c(nsim, nOM, nMP, nPI, nTS))

pi_means <- c(1,1, 1000)

for (om in 1:nOM) {
  for (pi in 1:nPI) {
    # PI identical for historical
    histVals <- matrix(
      pi_means[pi] *
        cumprod(c(rlnorm(nHistTS*nsim, 0, 0.05))),
      nrow=nsim, ncol=nHistTS, byrow=TRUE)
    histVals <- replicate(nMP, histVals)
```

```
    values[,om, , pi,1:nHistTS] <- aperm(histVals, c(1,3,2))

    for (mp in 1:nMP) {
      values[,om, mp, pi,(nHistTS+1):nTS] <- matrix(
        pi_means[pi] *
          cumprod(c(rlnorm(nProjTS*nsim, 0, 0.05))),
        nrow=nsim, ncol=nProjTS, byrow=FALSE)
    }
  }
}

# Create and populate Object
timeseries <- Timeseries(Code=c('B/BMSY', 'F/FMSY', 'TAC'),
                          Label=c('B/BMSY',
                                  'F/FMSY',
                                  'TAC'),
                          Description = c('This is the description for PI 1',
                                          'This is the description for PI 2',
                                          'This is the description for PI 3'),
                          Value=values
)

# Last historical time step
TimeNow(timeseries) <- 2024

# Add values for time steps
Time(timeseries) <- c(rev(seq(TimeNow(timeseries), by=-1, length.out=nHistTS)),
                      seq(TimeNow(timeseries)+1, by=1, length.out=nProjTS))

# Check
Check(timeseries)

# Add to `Slick` object
slick <- Slick()
Timeseries(slick) <- timeseries

# Plots
plotTimeseries(slick)
plotTimeseries(slick, 2)
plotTimeseries(slick, 3)

plotTimeseries(slick, byMP=TRUE)

plotTimeseries(slick, byOM=TRUE)

# Custom Reference Points
RefPoints(timeseries) <- list(
  list(Name=c('0.5 BMSY', 'BMSY', '1.5 BMSY'),
       Value=c(0.5, 1, 1.5),
       Color=c('red', 'orange', 'green')),
  list(Name=c('0.8 FMSY', 'FMSY'),
       Value=c(0.8,1),
       Color=c('orange', 'red')),
```

```
  list(Name='Target Catch',
       Value=1200,
       Color='blue')
)

Timeseries(slick) <- timeseries
plotTimeseries(slick)
plotTimeseries(slick, 2)
plotTimeseries(slick, 3)
```

---

plotTradeoff *Plot* Tradeoff

Description

Plot Tradeoff

Usage

```
plotTradeoff(
  slick,
  xPI = NULL,
  yPI = NULL,
  MP_label = "Code",
  lab_size = 6,
  point_size = 2,
  size.axis.title = 14,
  size.axis.text = 12
)
```

Arguments

slick	A [Slick-class()](#) object
xPI	Numeric value indicating the PI to plot on the x-axis. Multiple values are accepted. Recycled if xPI<yPI
yPI	Numeric value indicating the PI to plot on the y-axis. Multiple values are accepted. Recycled if yPI<xPI
MP_label	Label to use for the MPs. Either Code or Label. Description works as well, but you probably don't want to do that.
lab_size	Size of the MP labels
point_size	Size of the points
size.axis.title	
	Size of axis title
size.axis.text	Size of axis text

Value

A `ggplot2` object

Examples

```
# Generate dummy values
nOM <- 2
nMP <- 4
nPI <- 4

values <- array(NA, dim=c(nOM, nMP, nPI))

pi_means <- runif(nPI, 5, 50)
for (om in 1:nOM) {
  for (mp in 1:nMP) {
    for (pi in 1:nPI) {
      values[om, mp, pi] <- rlnorm(1,log(pi_means[pi]), 0.4)
    }
  }
}

# Create and populate Object
tradeoff <- Tradeoff(Code=c('PI1', 'PI2', 'PI3', 'PI4'),
               Label=c('Performance Indicator 1',
                       'Performance Indicator 2',
                       'Performance Indicator 3',
                       'Performance Indicator 4'),
             Description = c('This is the description for PI 1',
                              'This is the description for PI 2',
                              'This is the description for PI 3',
                              'This is the description for PI 4'),
               Value=values)

# Check
Check(tradeoff)

# Add to `Slick` object
slick <- Slick()
Tradeoff(slick) <- tradeoff

# Plots
plotTradeoff(slick)

plotTradeoff(slick, c(1,1,2), c(2,3,3))
```

PMnorm   *Normalize performance metric values in a Slick object*

Description

A function that converts deterministic or stochastic performance metrics to the range 0-100 and optionally inverts these

Usage

```
PMnorm(obj, det = TRUE, inv = NULL)
```

Arguments

obj	An object of class 'slick'
det	logical, should the normalization be applied to the deterministic performance metrics (or, if false the stochastic ones)
inv	A logical vector nPM long. If true, the PM will be inverted (100-value).

Value

An object of class Slick

Author(s)

T. Carruthers

Preset   *Assign or access* Preset *for a valid object class*

Description

Assign or access Preset for a valid object class

Usage

```
Preset(object)

Preset(object) <- value

## S4 method for signature 'Boxplot'
Preset(object)

## S4 replacement method for signature 'Boxplot'
Preset(object) <- value

## S4 method for signature 'Kobe'
Preset(object)

## S4 replacement method for signature 'Kobe'
Preset(object) <- value
```

```
## S4 method for signature 'MPs'
Preset(object)

## S4 replacement method for signature 'MPs'
Preset(object) <- value

## S4 method for signature 'OMs'
Preset(object)

## S4 replacement method for signature 'OMs'
Preset(object) <- value

## S4 method for signature 'Quilt'
Preset(object)

## S4 replacement method for signature 'Quilt'
Preset(object) <- value

## S4 method for signature 'Tradeoff'
Preset(object)

## S4 replacement method for signature 'Tradeoff'
Preset(object) <- value

## S4 method for signature 'Timeseries'
Preset(object)

## S4 replacement method for signature 'Timeseries'
Preset(object) <- value

## S4 method for signature 'Spider'
Preset(object)

## S4 replacement method for signature 'Spider'
Preset(object) <- value
```

Arguments

object   An object of class Boxplot-class(), Kobe-class(), Quilt-class(), Spider-class(),
               Timeseries-class(), or Tradeoff-class()

value   A list, formatted to match the class of object. See the documentation for
               corresponding object class for more details.

Value

Returns a list object from the Preset slot in object

Methods (by class)

- `Preset(Boxplot)`: Return Preset from a [`Boxplot-class()`](#) object
- `Preset(Boxplot) <- value`: Assign Preset slot from a [`Boxplot-class()`](#) object
- `Preset(Kobe)`: Return Preset from a [`Kobe-class()`](#) object
- `Preset(Kobe) <- value`: Assign Preset to a [`Kobe-class()`](#) object
- `Preset(MPs)`: Return Preset from a [`MPs-class()`](#) object
- `Preset(MPs) <- value`: Assign Preset to a [`MPs-class()`](#) object
- `Preset(OMs)`: Return Preset from a [`OMs-class()`](#) object
- `Preset(OMs) <- value`: Assign Preset to an OMs-class() object
- `Preset(Quilt)`: Return Preset from a [`Quilt-class()`](#) object
- `Preset(Quilt) <- value`: Assign Preset to a [`Quilt-class()`](#) object
- `Preset(Tradeoff)`: Return Preset from a [`Tradeoff-class()`](#) object
- `Preset(Tradeoff) <- value`: Assign Preset to a [`Tradeoff-class()`](#) object
- `Preset(Timeseries)`: Return Preset from a [`Timeseries-class()`](#) object
- `Preset(Timeseries) <- value`: Assign Preset to a [`Timeseries-class()`](#) object
- `Preset(Spider)`: Return Preset from a [`Spider-class()`](#) object
- `Preset(Spider) <- value`: Assign Preset slot from a [`Spider-class()`](#) object

Quilt   *Methods for Creating, Accessing and Assigning* Quilt *objects*

Description

The `Quilt` function is used both to create and modify an [`Quilt-class()`](#) object. and to access and assign Quilt for an object of class [`Slick-class()`](#). See Details.

Usage

```
Quilt(
  Code = "",
  Label = "",
  Description = "",
  Value = array(),
  Preset = list(),
  Color = c("darkblue", "lightblue"),
  MinValue = as.numeric(NA),
  MaxValue = as.numeric(NA)
)

Quilt(Slick) <- value

## S4 method for signature 'missing'
```

```
Quilt()

## S4 method for signature 'character_list'
Quilt(
  Code = "",
  Label = "",
  Description = "",
  Value = array(),
  Preset = list(),
  Color = c("darkblue", "lightblue"),
  MinValue = as.numeric(NA),
  MaxValue = as.numeric(NA)
)

## S4 method for signature 'Slick'
Quilt(Code)

## S4 replacement method for signature 'Slick'
Quilt(Slick) <- value
```

Arguments

Code	A *short* code for the Performance Indicators for this object. A character string length nPI or a named list for multi-language support. See Details
Label	A short label for the Performance Indicators for this object. Used to label axes on charts. Can be longer than Code but recommended to keep short as possible so it shows clearly in plots and tables. A character string length nPI or a named list for multi-language support. See Details
Description	A description for the Performance Indicators for this object. Can include Markdown, see Examples. A character string length nPI or a named list for multi-language support. See Details
Value	A numeric array with the stochastic performance indicator values for each operating model (OM), management procedure (MP), and performance indicator (PI). Dimensions: c(nOM, nMP, and nPI)
Preset	An optional named list for the preset buttons in the [App()]. The name of the list element will appear as a button in the [App()].
Color	A character vector length 2 of colors for the maximum and minimum values in the chart.
MinValue	Numeric vector length nPI with the minimum possible value for the respective PIs. Defaults to minimum PI value in Value (averaged across OMs in some cases)
MaxValue	Numeric vector length nPI with the maximum possible value (i.e., best performance) for the respective PIs. Defaults to maximum PI value in Value (averaged across OMs in some cases).
Slick	A [Slick-class()] object
value	A [Quilt-class()] object

Details

Objects of class `Quilt` are created with `Quilt()`

Use the `Code()`, `Label()`, `Description()`, `Value()`, `Preset()`, `Color()`, `MinValue()`, and `MaxValue()` functions to access and assign the values for an existing `Quilt` object, see `Examples`

Multi-Language Support:

Text with multi-language supported can be provided as a named list. Available languages:

- `en`: English (default)
- `es`: Spanish
- `fr`: French

Note:

Character strings in `Code`, `Label`, and `Description` must all be same length as the number of performance indicators (`nPIs`) in `Value`

Functions

- `Quilt(missing)`: Create an empty `Quilt` object
- `Quilt(character_list)`: Create a populated `Quilt` object
- `Quilt(Slick)`: Return `Quilt` from a `Slick-class()` object
- `Quilt(Slick) <- value`: Assign a `Quilt-class()` object to a `Slick-class()` object

See Also

`Code()`, `Label()`, `Description()`, `Color()`, `Metadata()`, `Preset()`, `Color()`, `MinValue()`, `MaxValue()`

Examples

```
# Generate dummy values
nOM <- 2
nMP <- 4
nPI <- 4

values <- array(NA, dim=c(nOM, nMP, nPI))

pi_means <- runif(nPI, 5, 50)
for (om in 1:nOM) {
  for (mp in 1:nMP) {
    for (pi in 1:nPI) {
      values[om, mp, pi] <- rlnorm(1,log(pi_means[pi]), 0.4)
    }
  }
}

# Create and populate Object
quilt <- Quilt(Code=c('PI1', 'PI2', 'PI3', 'PI4'),
               Label=c('Performance Indicator 1',
                       'Performance Indicator 2',
                       'Performance Indicator 3',
```

```
                           'Performance Indicator 4'),
              Description = c('This is the description for PI 1',
                              'This is the description for PI 2',
                              'This is the description for PI 3',
                              'This is the description for PI 4'),
              Value=values)

# Check
Check(quilt)

# Add to `Slick` object
slick <- Slick()
Quilt(slick) <- quilt

# Plots
plotQuilt(slick)
```

Quilt-class   *S4 class* `Quilt`

Description

Objects of class `Quilt` are used to store information for the Quilt chart. Like all S4 objects in `Slick`, slots in this object can be accessed and assigned using functions corresponding to slot name. See [Quilt()](#) and the the See Also section below.

Details

Objects of class `Quilt` are created with `Quilt()`

Multi-Language Support:

Text with multi-language supported can be provided as a named list. Available languages:

- en: English (default)
- es: Spanish
- fr: French

Note:

Character strings in `Code`, `Label`, and `Description` must all be same length as the number of performance indicators (nPIs) in `Value`

Slots

Code A *short* code for the Performance Indicators for this object. A character string length `nPI` or a named list for multi-language support. See `Details`

Label A short label for the Performance Indicators for this object. Used to label axes on charts. Can be longer than `Code` but recommended to keep short as possible so it shows clearly in plots and tables. A character string length `nPI` or a named list for multi-language support. See `Details`

Description A description for the Performance Indicators for this object. Can include Markdown, see `Examples`. A character string length `nPI` or a named list for multi-language support. See `Details`

Value A numeric array with the stochastic performance indicator values for each operating model (OM), management procedure (MP), and performance indicator (PI). Dimensions: c(nOM, nMP, and nPI)

Preset An optional named list for the preset buttons in the `App()`. The name of the list element will appear as a button in the `App()`.

Color A character vector length 2 of colors for the maximum and minimum values in the chart.

MinValue Numeric vector length `nPI` with the minimum possible value for the respective PIs. Defaults to minimum PI value in `Value` (averaged across OMs in some cases)

MaxValue Numeric vector length `nPI` with the maximum possible value (i.e., best performance) for the respective PIs. Defaults to maximum PI value in `Value` (averaged across OMs in some cases).

See Also

[Quilt()](), [Code()](), [Label()](), [Description()](), [Value()](), [Preset()]()

Examples

```
# Generate dummy values
nOM <- 2
nMP <- 4
nPI <- 4

values <- array(NA, dim=c(nOM, nMP, nPI))

pi_means <- runif(nPI, 5, 50)
for (om in 1:nOM) {
  for (mp in 1:nMP) {
    for (pi in 1:nPI) {
      values[om, mp, pi] <- rlnorm(1,log(pi_means[pi]), 0.4)
    }
  }
}

# Create and populate Object
quilt <- Quilt(Code=c('PI1', 'PI2', 'PI3', 'PI4'),
               Label=c('Performance Indicator 1',
                       'Performance Indicator 2',
                       'Performance Indicator 3',
                       'Performance Indicator 4'),
               Description = c('This is the description for PI 1',
                               'This is the description for PI 2',
```

```
                                     'This is the description for PI 3',
                                     'This is the description for PI 4'),
                   Value=values)

# Check
Check(quilt)

# Add to `Slick` object
slick <- Slick()
Quilt(slick) <- quilt

# Plots
plotQuilt(slick)
```

RefPoints   *Assign or access* RefPoints *for a valid object class*

Description

Assign or access RefPoints for a valid object class

Usage

```
RefPoints(object)

RefPoints(object) <- value

## S4 method for signature 'Timeseries'
RefPoints(object)

## S4 replacement method for signature 'Timeseries'
RefPoints(object) <- value
```

Arguments

object	An object of class [Timeseries-class()](#)
value	A list, formatted to match the class of object. See the documentation for corresponding object class for more details.

Value

Returns a list object with the contents of the RefPoints slot of [Timeseries()](#) objects

Methods (by class)

- RefPoints(Timeseries): Return RefPoints from a `Timeseries-class()` object

- RefPoints(Timeseries) <- value: Assign RefPoints to a `Timeseries-class()` object

---

show   *Generic show method*

Description

A copy of `methods::show()`

Usage

```
show(object)

## S4 method for signature 'Boxplot'
show(object)

## S4 method for signature 'CheckList'
show(object)

## S4 method for signature 'Kobe'
show(object)

## S4 method for signature 'MPs'
show(object)

## S4 method for signature 'OMs'
show(object)

## S4 method for signature 'Quilt'
show(object)

## S4 method for signature 'Tradeoff'
show(object)

## S4 method for signature 'Timeseries'
show(object)

## S4 method for signature 'Spider'
show(object)

## S4 method for signature 'Slick'
show(object)
```

Arguments

object   Object to print to console

Value

show returns an invisible NULL

Methods (by class)

- show(Boxplot): Print a `Boxplot-class()` object
- show(CheckList): Print a CheckList object
- show(Kobe): Print a `Kobe-class()` object
- show(MPs): Print a `MPs-class()` object
- show(OMs): Print a `OMs-class()` object
- show(Quilt): Print a `Quilt-class()` object
- show(Tradeoff): Print a `Tradeoff-class()` object
- show(Timeseries): Print a `Timeseries-class()` object
- show(Spider): Print a `Spider-class()` object
- show(Slick): Print a `Slick-class()` object

Slick-class   *Create a* Slick *class object*

Description

The Slick class is the main object class used in the Slick package. It contains sub-objects for the management procedures `MPs()`, operating models `OMs()`, and the six chart types: `Boxplot()`, `Kobe()`, `Quilt()`, `Spider()`, `Timeseries()`, and `Tradeoff()`, as well as metadata information for the Slick object such as Title, Author, and Introduction.

Usage

```
Slick(
  Title = "",
  Subtitle = "",
  Date = Sys.Date(),
  Author = "",
  Email = "",
  Institution = "",
  Introduction = "",
  MPs = NULL,
  OMs = NULL,
  Boxplot = NULL,
  Kobe = NULL,
```

```
  Quilt = NULL,
  Spider = NULL,
  Timeseries = NULL,
  Tradeoff = NULL
)

Title(object, lang = "en", markdown = FALSE)

Title(object) <- value

Subtitle(object, lang = "en", markdown = FALSE)

Subtitle(object) <- value

Date(object)

Date(object) <- value

Author(object, markdown = FALSE)

Author(object) <- value

Email(object, markdown = FALSE)

Email(object) <- value

Institution(object, lang = "en", markdown = FALSE)

Institution(object) <- value

Introduction(object, lang = "en", markdown = FALSE)

Introduction(object) <- value
```

Arguments

Title	Title for the `Slick` object. A character string. For multiple languages, use a named list with names: en, es, fr for the three supported languages.
Subtitle	Subtitle for the `Slick` object. A character string or a named list with languages: en, es, fr
Date	Date the Slick object was created. Text in format 'YYYY-MM-DD' or class Date e.g., `Sys.Date()`
Author	A character vector with Author(s) names. The length of the vector should equal the number of authors.
Email	A character vector with email addresses for the author(s). Must be same length as `Author`. Can include Markdown.
Institution	A character vector with institution details for the author(s). Must be same length as `Author`. Can include Markdown.

Introduction	Introduction text for the Slick object. Supports all markdown formatting. Character string, must be length 1. For multiple languages, use a named list with names: en, es, fr for the three supported languages.
MPs	An object of class [MPs-class()](#)
OMs	An object of class [OMs-class()](#)
Boxplot	An object of class [Boxplot-class()](#)
Kobe	An object of class [Kobe-class()](#)
Quilt	An object of class [Quilt-class()](#)
Spider	An object of class [Spider-class()](#)
Timeseries	An object of class [Timeseries-class()](#)
Tradeoff	An object of class [Tradeoff-class()](#)
object	A [Slick-class()](#) object
lang	Optional text string specifying the language (if available). Either 'en', 'es', or 'fr' for English, Spanish, or French respectively
markdown	Logical. Process markdown?
value	The value to assign to the object. See Slots for format of the relevant object class

Details

Objects of class Slick are created with Slick().

Like all S4 objects in Slick, slots in this object can be accessed and assigned using functions corresponding to slot name. See Usage and Functions section.

Multi-Language Support:

Text with multi-language supported can be provided as a named list. Available languages:

- en: English (default)
- es: Spanish
- fr: French

All functions with the exception of Date support Markdown.

Value

A Slick object

Functions

- Slick(): Create a [Slick-class()](#) object
- Title(): Access Title, Multi-language support
- Title(object) <- value: Assign Title, Multi-language support
- Subtitle(): Access Subtitle, Multi-language support
- Subtitle(object) <- value: Assign Subtitle, Multi-language support
- Date(): Access Date

- `Date(object) <- value`: Assign `Date`
- `Author()`: Access `Author`
- `Author(object) <- value`: Assign `Author`
- `Email()`: Access `Email`
- `Email(object) <- value`: Assign `Email`
- `Institution()`: Access `Institution`
- `Institution(object) <- value`: Assign `Institution`
- `Introduction()`: Access `Introduction`
- `Introduction(object) <- value`: Assign `Introduction`, can include Markdown. See `Examples`

Slots

`Title` Title for the `Slick` object. A character string. For multiple languages, use a named list with names: en, es, fr for the three supported languages.

`Subtitle` Subtitle for the `Slick` object. A character string or a named list with languages: en, es, fr

`Date` Date the Slick object was created. Text in format 'YYYY-MM-DD' or class `Date` e.g., `Sys.Date()`

`Author` A character vector with Author(s) names. The length of the vector should equal the number of authors.

`Email` A character vector with email addresses for the author(s). Must be same length as `Author`. Can include Markdown.

`Institution` A character vector with institution details for the author(s). Must be same length as `Author`. Can include Markdown.

`Introduction` Introduction text for the `Slick` object. Supports all markdown formatting. Character string, must be length 1. For multiple languages, use a named list with names: en, es, fr for the three supported languages.

`MPs` An object of class MPs-class()

`OMs` An object of class OMs-class()

`Boxplot` An object of class Boxplot-class()

`Kobe` An object of class Kobe-class()

`Quilt` An object of class Quilt-class()

`Spider` An object of class Spider-class()

`Timeseries` An object of class Timeseries-class()

`Tradeoff` An object of class Tradeoff-class()

See Also

[MPs()](MPs()), [OMs()](OMs()), Boxplot(), Kobe(), Quilt(), Spider(), Timeseries(), Tradeoff(), Check(), Title(), Subtitle(), Date(), Author(), Email(), Institution(), Introduction()

Examples

```
# Assign values to a new `Slick` object
slick <- Slick()

Title(slick) <- 'An Example Slick Object'
Subtitle(slick) <- ""
Date(slick) <- Sys.Date()
Author(slick) <- 'Adrian Hordyk'
Email(slick) <- "[mailto:adrian@bluematterscience.com](mailto:adrian@bluematterscience.com)"
Institution(slick) <- "[Blue Matter Science](bluematterscience.com)"

Introduction(slick) <- "This is the Introduction text"

# Access values from `Slick` object
Title(slick)
Subtitle(slick)
Date(slick)
Author(slick)
Email(slick)
Institution(slick)
Introduction(slick)
```

Spider   *Methods for Creating, Accessing and Assigning* Spider *objects*

Description

The Spider function is used both to create and modify an Spider-class() object. and to access
and assign Spider for an object of class Slick-class(). See Details.

Usage

```
Spider(
  Code = "",
  Label = "",
  Description = "",
  Value = array(),
  Preset = list()
)

Spider(Slick) <- value

## S4 method for signature 'missing'
Spider()

## S4 method for signature 'character'
Spider(
```

```
  Code = "",
  Label = "",
  Description = "",
  Value = array(),
  Preset = list()
)

## S4 method for signature 'list'
Spider(
  Code = "",
  Label = "",
  Description = "",
  Value = array(),
  Preset = list()
)

## S4 method for signature 'Slick'
Spider(Code)

## S4 replacement method for signature 'Slick'
Spider(Slick) <- value
```

Arguments

Code	A *short* code for the Performance Indicators for this object. A character string length nPI or a named list for multi-language support. See `Details`
Label	A short label for the Performance Indicators for this object. Used to label axes on charts. Can be longer than `Code` but recommended to keep short as possible so it shows clearly in plots and tables. A character string length nPI or a named list for multi-language support. See `Details`
Description	A description for the Performance Indicators for this object. Can include Markdown, see `Examples`. A character string length nPI or a named list for multi-language support. See `Details`
Value	A numeric array with the stochastic performance indicator values for each operating model (OM), management procedure (MP), and performance indicator (PI). Dimensions: c(nOM, nMP, and nPI). All PI values must range between 0 and 1 or 0 and 100. If all values are <= 1, they will be multiplied by 100 in the plot.
Preset	An optional named list for the preset buttons in the [App()]. The name of the list element will appear as a button in the [App()].
Slick	A [Slick-class()] object
value	A [Spider-class()] object

Details

Objects of class Spider are created with Spider()

Use the [Code()], [Label()], [Description()], [Value()], [Preset()] functions to access and assign the values for an existing Spider object, see `Examples`

Multi-Language Support:

Text with multi-language supported can be provided as a named list. Available languages:

- en: English (default)
- es: Spanish
- fr: French

Note:

Character strings in `Code`, `Label`, and `Description` must all be same length as the number of performance indicators (nPIs) in 'Value

Functions

- `Spider(missing)`: Create an empty `Spider` object
- `Spider(character)`: Create a populated `Spider` object
- `Spider(list)`: Create a populated `Spider` object
- `Spider(Slick)`: Return Spider from a Slick-class() object
- `Spider(Slick) <- value`: Assign a Spider-class() object to a Slick-class() object

See Also

Code(), Label(), Description(), Metadata(), Value(), Preset()

Examples

```
# Generate dummy values
nOM <- 2
nMP <- 4
nPI <- 4

values <- array(NA, dim=c(nOM, nMP, nPI))

# Note: PI values must be between 0 and 1, with 1 indicating better performance
pi_means <- runif(nPI, 0, 1)
for (om in 1:nOM) {
  for (mp in 1:nMP) {
    for (pi in 1:nPI) {
      values[om, mp, pi] <- runif(1, pi_means[pi])
    }
  }
}

# Create and populate Object
spider <- Spider(Code=c('PI1', 'PI2', 'PI3', 'PI4'),
              Label=c('Performance Indicator 1',
                      'Performance Indicator 2',
                      'Performance Indicator 3',
                      'Performance Indicator 4'),
              Description = c('This is the description for PI 1',
                              'This is the description for PI 2',
```

```
                                      'This is the description for PI 3',
                                      'This is the description for PI 4'),
                 Value=values)

# Check
Check(spider)

# Add to `Slick` object
slick <- Slick()
Spider(slick) <- spider

# Plots
plotSpider(slick)

plotSpider(slick, fill=TRUE)

plotSpider(slick, byMP=TRUE)

plotSpider(slick, byOM=TRUE)
```

Spider-class   *S4 class* Spider

Description

Objects of class Spider are used to store information for the Spider plots. Like all S4 objects in Slick, slots in this object can be accessed and assigned using functions corresponding to slot name. See Spider and the the See Also section below.

Details

Objects of class Spider are created with Spider()

Multi-Language Support:

Text with multi-language supported can be provided as a named list. Available languages:

- en: English (default)
- es: Spanish
- fr: French

Note:

Character strings in Code, Label, and Description must all be same length as the number of performance indicators (nPIs) in Value

Slots

Code A *short* code for the Performance Indicators for this object. A character string length nPI or a named list for multi-language support. See `Details`

Label A short label for the Performance Indicators for this object. Used to label axes on charts. Can be longer than `Code` but recommended to keep short as possible so it shows clearly in plots and tables. A character string length nPI or a named list for multi-language support. See `Details`

Description A description for the Performance Indicators for this object. Can include Markdown, see `Examples`. A character string length nPI or a named list for multi-language support. See `Details`

Value A numeric array with the stochastic performance indicator values for each operating model (OM), management procedure (MP), and performance indicator (PI). Dimensions: c(nOM, nMP, and nPI). All PI values must range between 0 and 1 or 0 and 100. If all values are <= 1, they will be multiplied by 100 in the plot. Dimensions: c(nOM, nMP, and nPI)

Preset An optional named list for the preset buttons in the `App()`. The name of the list element will appear as a button in the `App()`.

See Also

Spider, `Code()`, `Label()`, `Description()`, `Value()`, `Preset()`

Examples

```
# Generate dummy values
nOM <- 2
nMP <- 4
nPI <- 4

values <- array(NA, dim=c(nOM, nMP, nPI))

# Note: PI values must be between 0 and 1, with 1 indicating better performance
pi_means <- runif(nPI, 0, 1)
for (om in 1:nOM) {
  for (mp in 1:nMP) {
    for (pi in 1:nPI) {
      values[om, mp, pi] <- runif(1, pi_means[pi])
    }
  }
}

# Create and populate Object
spider <- Spider(Code=c('PI1', 'PI2', 'PI3', 'PI4'),
              Label=c('Performance Indicator 1',
                      'Performance Indicator 2',
                      'Performance Indicator 3',
                      'Performance Indicator 4'),
              Description = c('This is the description for PI 1',
                              'This is the description for PI 2',
                              'This is the description for PI 3',
```

```
                                    'This is the description for PI 4'),
                 Value=values)

# Check
Check(spider)

# Add to `Slick` object
slick <- Slick()
Spider(slick) <- spider

# Plots
plotSpider(slick)

plotSpider(slick, fill=TRUE)

plotSpider(slick, byMP=TRUE)

plotSpider(slick, byOM=TRUE)
```

Time   *Access or assign* Time *for object of class* Kobe *or* Timeseries

---

Description

Access or assign Time for object of class Kobe or Timeseries

Usage

```
Time(object)

Time(object) <- value

## S4 method for signature 'Kobe'
Time(object)

## S4 replacement method for signature 'Kobe'
Time(object) <- value

## S4 method for signature 'Timeseries'
Time(object)

## S4 replacement method for signature 'Timeseries'
Time(object) <- value
```

Arguments

object	A [Kobe-class()] or [Timeseries()] class object
value	Value to assign to Time

Value

Returns a numeric vector with values from the Time slot in [Kobe()](#) and [Timeseries()](#) objects

Methods (by class)

- Time(Kobe): Return Time from a [Kobe-class()](#) object
- Time(Kobe) <- value: Assign Time to a [Kobe-class()](#) object
- Time(Timeseries): Return Time from a [Timeseries-class()](#) object
- Time(Timeseries) <- value: Assign Time to a [Timeseries-class()](#) object

Functions

- Time(object) <- value: Assign value to object@Time

TimeLab   *Access or assign* TimeLab *in a* Kobe *or* Timeseries *object*

Description

Access or assign TimeLab in a Kobe or Timeseries object

Usage

```
TimeLab(object, lang = "en")

TimeLab(object) <- value

## S4 method for signature 'Kobe'
TimeLab(object, lang = "en")

## S4 replacement method for signature 'Kobe'
TimeLab(object) <- value

## S4 method for signature 'Timeseries'
TimeLab(object, lang = "en")

## S4 replacement method for signature 'Timeseries'
TimeLab(object) <- value
```

Arguments

object	A [Kobe-class()](#) or [Timeseries](#) object
lang	Optional text string specifying the language (if available). Either 'en', 'es', or 'fr' for English, Spanish, or French respectively
value	A character string to assign to TimeLab in object.

Value

Returns a character string from the TimeLab slot in Kobe() and Timeseries() objects

Methods (by class)

- TimeLab(Kobe): Return TimeLab from a Kobe-class() object

- TimeLab(Kobe) <- value: Assign TimeLab to a Kobe-class() object

- TimeLab(Timeseries): Return TimeLab from a Timeseries-class() object

- TimeLab(Timeseries) <- value: Assign TimeLab to a Timeseries-class() object

---

TimeNow   *Access and return* TimeNow *from a* Timeseries-class() *object*

Description

Access and return TimeNow from a Timeseries-class() object

Usage

```
TimeNow(Timeseries)

TimeNow(Timeseries) <- value
```

Arguments

Timeseries   A Timeseries-class() object

value   A character string label for the time axis. Use a named list for multiple languages

Value

A numeric value

Functions

- TimeNow(Timeseries) <- value: Assign TimeNow to a Timeseries-class() object

---

Timeseries   *Methods for Creating, Accessing and Assigning* Timeseries *objects*

Description

An object of class Timeseries contains information for the Time Series chart. The Timeseries function is used both to create and modify an Timeseries-class() object, and to access and assign Timeseries for an object of class Slick-class(). See Details.

Usage

```
Timeseries(
  Code = "",
  Label = "",
  Description = "",
  Time = numeric(),
  TimeNow = numeric(),
  TimeLab = "Year",
  Value = array(),
  Preset = list(),
  Target = NULL,
  Limit = NULL,
  RefPoints = list()
)

Timeseries(Slick) <- value

## S4 method for signature 'missing'
Timeseries()

## S4 method for signature 'character'
Timeseries(
  Code = "",
  Label = "",
  Description = "",
  Time = numeric(),
  TimeNow = numeric(),
  TimeLab = "Year",
  Value = array(),
  Preset = list(),
  Target = NULL,
  Limit = NULL,
  RefPoints = list()
)

## S4 method for signature 'list'
Timeseries(
```

```
  Code = "",
  Label = "",
  Description = "",
  Time = numeric(),
  TimeNow = numeric(),
  TimeLab = "Year",
  Value = array(),
  Preset = list(),
  Target = NULL,
  Limit = NULL,
  RefPoints = list()
)

## S4 method for signature 'Slick'
Timeseries(Code)

## S4 replacement method for signature 'Slick'
Timeseries(Slick) <- value
```

Arguments

Code	A *short* code for the Performance Indicators for this object. A character string length nPI or a named list for multi-language support. See `Details`
Label	A short label for the Performance Indicators for this object. Used to label axes on charts. Can be longer than `Code` but recommended to keep short as possible so it shows clearly in plots and tables. A character string length nPI or a named list for multi-language support. See `Details`
Description	A description for the Performance Indicators for this object. Can include Markdown, see `Examples`. A character string length nPI or a named list for multi-language support. See `Details`
Time	A numeric vector with values for the historical and projection time-steps. Must match length nTS in `Value`
TimeNow	A numeric value matching the last historical timestep in `Time`
TimeLab	Character string length 1. Name of the time step (e.g., 'Year'). Will be used as the label in the `Timeseries` plot. Use a named list for multiple languages.
Value	A numeric array with the stochastic performance indicator values for each simulation (sim), operating model (OM), management procedure (MP), performance indicator (PI), and historical + projection timestep (nTS). Dimensions: c(nsim, nOM, nMP, nPI, nTS)
Preset	An optional named list for the preset buttons in the `App()`. The name of the list element will appear as a button in the `App()`.
Target	Numeric vector length nPI with the target value for the PIs.
Limit	Numeric vector length nPI with the limit value for the PIs.
RefPoints	List for setting custom Reference Points. Overrides `Target` and `Limit`. See `Details`
Slick	A `Slick-class()` object
value	A `Timeseries-class()` object

Details

Use `plotTimeseries()` to create the boxplot from the console.

Note:

Character strings in `Code`, `Label`, and `Description` must all be same length as the number of performance indicators (nPIs) in `Value`

Objects of class `Timeseries` are created with `Timeseries()`

Multi-Language Support:

Text with multi-language supported can be provided as a named list. Available languages:

- `en`: English (default)
- `es`: Spanish
- `fr`: French

Custom Reference Points with `RefPoints`:

`RefPoints` provides more options than the default `Target` and `Limit` reference points. It can be used to control the name and color of the reference point lines, or to add additional reference point lines to the `Timeseries` plot.

Note: If `RefPoints` is included, `Target` and `Limit` are ignored.

`RefPoints` must be a list of length <= length(Code) (i.e., the number of performance indicators). Each element in `RefPoints` should be a named list:

- `Name` character vector with name(s) of reference point(s)
- `Value` numeric vector length `Name` with value(s) for the reference point(s)
- `Color` character vector length `Name` with color(s) for the reference point(s)

See `Examples`

Accessing Slots:

Use the `Code()`, `Label()`, `Description()`, `Value()`, `Preset()` functions to access and assign the values for an existing `Timeseries` object, see `Examples`

Functions

- `Timeseries(missing)`: Create an empty `Timeseries` object
- `Timeseries(character)`: Create a populated `Timeseries` object
- `Timeseries(list)`: Create a populated `Timeseries` object
- `Timeseries(Slick)`: Return `Timeseries` from a `Slick-class()` object
- `Timeseries(Slick) <- value`: Assign a `Timeseries-class()` object to a `Slick-class()` object

See Also

`Timeseries-class()`, `Code()`, `Label()`, `Description()`, `Metadata()`, `Value()`, `Preset()`, `plotTimeseries()`

`Code()`, `Label()`, `Description()`, `Metadata()`, `Value()`, `Preset()`

Examples

```
# Generate dummy values
nsim <- 10
nOM <- 2
nMP <- 4
nPI <- 3
nHistTS <- 50
nProjTS <- 30
nTS <- nHistTS + nProjTS

set.seed(101)

values <- array(NA, dim=c(nsim, nOM, nMP, nPI, nTS))

pi_means <- c(1,1, 1000)

for (om in 1:nOM) {
  for (pi in 1:nPI) {
    # PI identical for historical
    histVals <- matrix(
      pi_means[pi] *
        cumprod(c(rlnorm(nHistTS*nsim, 0, 0.05))),
      nrow=nsim, ncol=nHistTS, byrow=TRUE)
    histVals <- replicate(nMP, histVals)
    values[,om, , pi,1:nHistTS] <- aperm(histVals, c(1,3,2))

    for (mp in 1:nMP) {
      values[,om, mp, pi,(nHistTS+1):nTS] <- matrix(
        pi_means[pi] *
          cumprod(c(rlnorm(nProjTS*nsim, 0, 0.05))),
        nrow=nsim, ncol=nProjTS, byrow=FALSE)
    }
  }
}

# Create and populate Object
timeseries <- Timeseries(Code=c('B/BMSY', 'F/FMSY', 'TAC'),
                         Label=c('B/BMSY',
                                 'F/FMSY',
                                 'TAC'),
                         Description = c('This is the description for PI 1',
                                         'This is the description for PI 2',
                                         'This is the description for PI 3'),
                         Value=values
)

# Last historical time step
TimeNow(timeseries) <- 2024

# Add values for time steps
Time(timeseries) <- c(rev(seq(TimeNow(timeseries), by=-1, length.out=nHistTS)),
                      seq(TimeNow(timeseries)+1, by=1, length.out=nProjTS))
```

```
# Check
Check(timeseries)

# Add to `Slick` object
slick <- Slick()
Timeseries(slick) <- timeseries

# Plots
plotTimeseries(slick)
plotTimeseries(slick, 2)
plotTimeseries(slick, 3)

plotTimeseries(slick, byMP=TRUE)

plotTimeseries(slick, byOM=TRUE)

# Custom Reference Points
RefPoints(timeseries) <- list(
  list(Name=c('0.5 BMSY', 'BMSY', '1.5 BMSY'),
       Value=c(0.5, 1, 1.5),
       Color=c('red', 'orange', 'green')),
  list(Name=c('0.8 FMSY', 'FMSY'),
       Value=c(0.8,1),
       Color=c('orange', 'red')),
  list(Name='Target Catch',
       Value=1200,
       Color='blue')
)

Timeseries(slick) <- timeseries
plotTimeseries(slick)
plotTimeseries(slick, 2)
plotTimeseries(slick, 3)
```

Timeseries-class   *S4 class* Timeseries

Description

Objects of class Timeseries are used to store information for the Time Series plots. Like all S4 objects in Slick, slots in this object can be accessed and assigned using functions corresponding to slot name. See Timeseries() and the the See Also section below.

Details

Objects of class Timeseries are created with Timeseries()

Multi-Language Support:

Text with multi-language supported can be provided as a named list. Available languages:

- en: English (default)
- es: Spanish
- fr: French

Note:

Character strings in `Code`, `Label`, and `Description` must all be same length as the number of performance indicators (nPIs) in `Value`

Slots

Code A *short* code for the Performance Indicators for this object. A character string length `nPI` or a named list for multi-language support. See `Details`

Label A short label for the Performance Indicators for this object. Used to label axes on charts. Can be longer than `Code` but recommended to keep short as possible so it shows clearly in plots and tables. A character string length `nPI` or a named list for multi-language support. See `Details`

Description A description for the Performance Indicators for this object. Can include Markdown, see `Examples`. A character string length `nPI` or a named list for multi-language support. See `Details`

Time A numeric vector with values for the historical and projection time-steps. Must match length `nTS` in `Value`. Can also be class `Date`, `POSIXct` or `POSIXt`

TimeNow A numeric value matching the last historical timestep in `Time`

TimeLab Character string length 1. Name of the time step (e.g., 'Year'). Will be used as the label in the plots. Use a named list for multiple languages.

Value A numeric array with the stochastic performance indicator values for each simulation (sim), operating model (OM), management procedure (MP), performance indicator (PI), and historical + projection timestep (nTS). Dimensions: c(nsim, nOM, nMP, nPI, nTS)

Preset An optional named list for the preset buttons in the [App()]. The name of the list element will appear as a button in the [App()].

Target Numeric vector length `nPI` with the target value for the PIs.

Limit Numeric vector length `nPI` with the limit value for the PIs.

RefPoints List for setting custom Reference Points. Overrides `Target` and `Limit`. See `Details` section in `Timeseries()`.

See Also

[Timeseries()], [Code()], [Label()], [Description()], [Value()], [Preset()]

Examples

```
# Generate dummy values
nsim <- 10
nOM <- 2
nMP <- 4
```

```
nPI <- 3
nHistTS <- 50
nProjTS <- 30
nTS <- nHistTS + nProjTS

set.seed(101)

values <- array(NA, dim=c(nsim, nOM, nMP, nPI, nTS))

pi_means <- c(1,1, 1000)

for (om in 1:nOM) {
  for (pi in 1:nPI) {
    # PI identical for historical
    histVals <- matrix(
      pi_means[pi] *
        cumprod(c(rlnorm(nHistTS*nsim, 0, 0.05))),
      nrow=nsim, ncol=nHistTS, byrow=TRUE)
    histVals <- replicate(nMP, histVals)
    values[,om, , pi,1:nHistTS] <- aperm(histVals, c(1,3,2))

    for (mp in 1:nMP) {
      values[,om, mp, pi,(nHistTS+1):nTS] <- matrix(
        pi_means[pi] *
          cumprod(c(rlnorm(nProjTS*nsim, 0, 0.05))),
        nrow=nsim, ncol=nProjTS, byrow=FALSE)
    }
  }
}

# Create and populate Object
timeseries <- Timeseries(Code=c('B/BMSY', 'F/FMSY', 'TAC'),
                          Label=c('B/BMSY',
                                  'F/FMSY',
                                  'TAC'),
                          Description = c('This is the description for PI 1',
                                          'This is the description for PI 2',
                                          'This is the description for PI 3'),
                          Value=values
)

# Last historical time step
TimeNow(timeseries) <- 2024

# Add values for time steps
Time(timeseries) <- c(rev(seq(TimeNow(timeseries), by=-1, length.out=nHistTS)),
                      seq(TimeNow(timeseries)+1, by=1, length.out=nProjTS))

# Check
Check(timeseries)

# Add to `Slick` object
slick <- Slick()
```

```
Timeseries(slick) <- timeseries

# Plots
plotTimeseries(slick)
plotTimeseries(slick, 2)
plotTimeseries(slick, 3)

plotTimeseries(slick, byMP=TRUE)

plotTimeseries(slick, byOM=TRUE)

# Custom Reference Points
RefPoints(timeseries) <- list(
  list(Name=c('0.5 BMSY', 'BMSY', '1.5 BMSY'),
       Value=c(0.5, 1, 1.5),
       Color=c('red', 'orange', 'green')),
  list(Name=c('0.8 FMSY', 'FMSY'),
       Value=c(0.8,1),
       Color=c('orange', 'red')),
  list(Name='Target Catch',
       Value=1200,
       Color='blue')
)

Timeseries(slick) <- timeseries
plotTimeseries(slick)
plotTimeseries(slick, 2)
plotTimeseries(slick, 3)
```

TimeTerminal	*Assign or access* TimeTerminal *for a valid object class*

Description

Assign or access TimeTerminal for a valid object class

Usage

```
TimeTerminal(object)

TimeTerminal(object) <- value

## S4 method for signature 'Kobe'
TimeTerminal(object)

## S4 replacement method for signature 'Kobe'
TimeTerminal(object) <- value
```

Arguments

`object`	An object of class [Kobe-class()]()
`value`	A numeric value with a value matching one in Kobe@Time [Kobe-class()]() for details.

Value

Returns a numeric values from the `TimeTerminal` slot in [Kobe()]() objects

Methods (by class)

- `TimeTerminal(Kobe)`: Return `TimeTerminal` from a [Kobe-class()]() object

- `TimeTerminal(Kobe) <- value`: Assign `TimeTerminal` to a [Kobe-class()]() object

Tradeoff   *Methods for Creating, Accessing and Assigning* Tradeoff *objects*

Description

The `Tradeoff` function is used both to create and modify an [Tradeoff-class()]() object. and to access and assign Tradeoff for an object of class [Slick-class()](). See `Details`.

Usage

```
Tradeoff(
  Code = "",
  Label = "",
  Description = "",
  Value = array(),
  Preset = list()
)

Tradeoff(Slick) <- value

## S4 method for signature 'missing'
Tradeoff()

## S4 method for signature 'character'
Tradeoff(
  Code = "",
  Label = "",
  Description = "",
  Value = array(),
  Preset = list()
)
```

```
## S4 method for signature 'list'
Tradeoff(
  Code = "",
  Label = "",
  Description = "",
  Value = array(),
  Preset = list()
)

## S4 method for signature 'Slick'
Tradeoff(Code)

## S4 replacement method for signature 'Slick'
Tradeoff(Slick) <- value
```

Arguments

`Code`	A *short* code for the Performance Indicators for this object. A character string length nPI or a named list for multi-language support. See `Details`
`Label`	A short label for the Performance Indicators for this object. Used to label axes on charts. Can be longer than `Code` but recommended to keep short as possible so it shows clearly in plots and tables. A character string length nPI or a named list for multi-language support. See `Details`
`Description`	A description for the Performance Indicators for this object. Can include Markdown, see `Examples`. A character string length nPI or a named list for multi-language support. See `Details`
`Value`	A numeric array with the stochastic performance indicator values for each operating model (OM), management procedure (MP), and performance indicator (PI) Dimensions: c(nOM, nMP, nPI)
`Preset`	An optional named list for the preset buttons in the `App()`. The name of the list element will appear as a button in the `App()`.
`Slick`	A `Slick-class()` object
`value`	A `Tradeoff-class()` object

Details

Objects of class `Tradeoff` are created with `Tradeoff()`

Multi-Language Support:

Text with multi-language supported can be provided as a named list. Available languages:

- en: English (default)
- es: Spanish
- fr: French

Note:

Character strings in `Code`, `Label`, and `Description` must all be same length as the number of performance indicators (nPIs) in `Value`

Functions

- Tradeoff(missing): Create an empty Tradeoff object

- Tradeoff(character): Create a populated Tradeoff object

- Tradeoff(list): Create a populated Tradeoff object

- Tradeoff(Slick): Return Tradeoff from a Slick-class() object

- Tradeoff(Slick) <- value: Assign a Tradeoff-class() object to a Slick-class() object

See Also

Code(), Label(), Description(), Metadata(), Value(), Preset()

Examples

```
# Generate dummy values
nOM <- 2
nMP <- 4
nPI <- 4

values <- array(NA, dim=c(nOM, nMP, nPI))

pi_means <- runif(nPI, 5, 50)
for (om in 1:nOM) {
  for (mp in 1:nMP) {
    for (pi in 1:nPI) {
      values[om, mp, pi] <- rlnorm(1,log(pi_means[pi]), 0.4)
    }
  }
}

# Create and populate Object
tradeoff <- Tradeoff(Code=c('PI1', 'PI2', 'PI3', 'PI4'),
              Label=c('Performance Indicator 1',
                      'Performance Indicator 2',
                      'Performance Indicator 3',
                      'Performance Indicator 4'),
              Description = c('This is the description for PI 1',
                              'This is the description for PI 2',
                              'This is the description for PI 3',
                              'This is the description for PI 4'),
              Value=values)

# Check
Check(tradeoff)

# Add to `Slick` object
slick <- Slick()
Tradeoff(slick) <- tradeoff

# Plots
plotTradeoff(slick)
```

```
plotTradeoff(slick, c(1,1,2), c(2,3,3))
```

Tradeoff-class *S4 class* Tradeoff

Description

Objects of class Tradeoff are used to store information for the Tradeoff plot. Like all S4 objects in Slick, slots in this object can be accessed and assigned using functions corresponding to slot name. See Tradeoff() and the the See Also section below.

Details

Objects of class Tradeoff are created with Tradeoff()

Multi-Language Support:

Text with multi-language supported can be provided as a named list. Available languages:

- en: English (default)
- es: Spanish
- fr: French

Note:

Character strings in Code, Label, and Description must all be same length as the number of performance indicators (nPIs) in Value

Slots

Code A *short* code for the Performance Indicators for this object. A character string length nPI or a named list for multi-language support. See Details

Label A short label for the Performance Indicators for this object. Used to label axes on charts. Can be longer than Code but recommended to keep short as possible so it shows clearly in plots and tables. A character string length nPI or a named list for multi-language support. See Details

Description A description for the Performance Indicators for this object. Can include Markdown, see Examples. A character string length nPI or a named list for multi-language support. See Details

Value A 3 dimensional numeric array with the stochastic performance indicator values for operating model (OM), management procedure (MP), and performance indicator (PI). Dimensions: c(nOM, nMP, and nPI)

Preset An optional named list for the preset buttons in the App(). The name of the list element will appear as a button in the App().

See Also

Tradeoff(), Code(), Label(), Description(), Value(), Preset()

Examples

```
# Generate dummy values
nOM <- 2
nMP <- 4
nPI <- 4

values <- array(NA, dim=c(nOM, nMP, nPI))

pi_means <- runif(nPI, 5, 50)
for (om in 1:nOM) {
  for (mp in 1:nMP) {
    for (pi in 1:nPI) {
      values[om, mp, pi] <- rlnorm(1,log(pi_means[pi]), 0.4)
    }
  }
}

# Create and populate Object
tradeoff <- Tradeoff(Code=c('PI1', 'PI2', 'PI3', 'PI4'),
              Label=c('Performance Indicator 1',
                      'Performance Indicator 2',
                      'Performance Indicator 3',
                      'Performance Indicator 4'),
              Description = c('This is the description for PI 1',
                              'This is the description for PI 2',
                              'This is the description for PI 3',
                              'This is the description for PI 4'),
              Value=values)

# Check
Check(tradeoff)

# Add to `Slick` object
slick <- Slick()
Tradeoff(slick) <- tradeoff

# Plots
plotTradeoff(slick)

plotTradeoff(slick, c(1,1,2), c(2,3,3))
```

Update   *Updates an old object of class* Slick *to new S4 class* Slick

---

Description

Updates an old object of class Slick to new S4 class Slick

Usage

```
Update(slick)
```

Arguments

slick   An S3 object of class `Slick`

Value

An S4 object of class `Slick`

Examples

```
slick <- Slick() # a dummy old Slick object
slick <- Update(slick) # updated to S4 object
```

Value   *Assign or access* Value *for a valid object class*

Description

Assign or access `Value` for a valid object class

Usage

```
Value(object)

Value(object) <- value

## S4 method for signature 'Boxplot'
Value(object)

## S4 replacement method for signature 'Boxplot'
Value(object) <- value

## S4 method for signature 'Kobe'
Value(object)

## S4 replacement method for signature 'Kobe'
Value(object) <- value

## S4 method for signature 'Quilt'
Value(object)

## S4 replacement method for signature 'Quilt'
Value(object) <- value

## S4 method for signature 'Tradeoff'
Value(object)
```

```
## S4 replacement method for signature 'Tradeoff'
Value(object) <- value

## S4 method for signature 'Timeseries'
Value(object)

## S4 replacement method for signature 'Timeseries'
Value(object) <- value

## S4 method for signature 'Spider'
Value(object)

## S4 replacement method for signature 'Spider'
Value(object) <- value
```

Arguments

`object`	An object of class [Boxplot-class()](), [Kobe-class()](), [Quilt-class()](), [Spider-class()](), [Timeseries-class()](), or [Tradeoff-class()]()
`value`	An `array`, formatted to match the class of `object`. See the documentation for corresponding `object` class for more details.

Value

Returns a numeric array with the contents of the `Value` slot of `object`

Methods (by class)

- `Value(Boxplot)`: Return Value from a [Boxplot-class()]() object
- `Value(Boxplot) <- value`: Assign Value to a [Boxplot-class()]() object
- `Value(Kobe)`: Return Value from a [Kobe-class()]() object
- `Value(Kobe) <- value`: Assign Value to a [Kobe-class()]() object
- `Value(Quilt)`: Return Value from a [Quilt-class()]() object
- `Value(Quilt) <- value`: Assign Value to a [Quilt-class()]() object
- `Value(Tradeoff)`: Return Value from a [Tradeoff-class()]() object
- `Value(Tradeoff) <- value`: Assign Value to a [Tradeoff-class()]() object
- `Value(Timeseries)`: Return Value from a [Timeseries-class()]() object
- `Value(Timeseries) <- value`: Assign Value to a [Timeseries-class()]() object
- `Value(Spider)`: Return Value from a [Spider-class()]() object
- `Value(Spider) <- value`: Assign Value to a [Spider-class()]() object

Index