Package 'CSwR'

June 1, 2026

Title Companion to the Book ``Computational Statistics with R''

Version 0.1.3

Description Provides data sets and functions used in the book
``Computational Statistics with R'' (<https://cswr.nrhstat.org>).

Depends R (>= 3.5.0)

Imports bench, ggplot2, rlang

Suggests covr, spelling, testthat (>= 2.1.0)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Language en-US

URL https://jolars.github.io/CSwR-Package/, https://cswr.nrhstat.org/

BugReports https://github.com/jolars/CSwR-Package/issues

License MIT + file LICENSE

NeedsCompilation no

Author Niels Richard Hansen [aut] (ORCID:
<https://orcid.org/0000-0003-3883-365X>),
Johan Larsson [aut, cre] (ORCID:
<https://orcid.org/0000-0002-4029-5945>)

Maintainer Johan Larsson <johan@jolars.co>

Repository CRAN

Date/Publication 2026-06-01 08:40:20 UTC

Contents

`angle` *Torsion angles*

Description

The phi and psi torsion angles between peptide planes for the human protein 1HMP.

Usage

```
angle
```

Format

A data frame with 419 rows and 5 columns:

`chain` a character. The chain (either A or B) of the protein

`AA` a character. The three-letter amino acid name

`pos` an integer. Position in the amino acid sequence

`phi` numeric. The phi angle

`psi` numeric. The psi angle

Source

https://warwick.ac.uk/fac/sci/moac/people/students/peter_cock/r/ramachandran

autoplot.tracer *Plot results from a trace*

Description

Plots the value of a traced object on a log-scale against runtime.

Usage

```
## S3 method for class 'tracer'
autoplot(object, y, ...)

## S3 method for class 'trace'
autoplot(object, y, log = TRUE, ...)
```

Arguments

object	a trace or tracer object
y	the name of the traced object to plot
...	additional arguments passed to
log	logical. Should the y-axis be on a log-scale. Default is TRUE.

Value

a ggplot object

Examples

```
tr <- tracer("i", Delta = 0)

for (i in 1:3) {
  tr$tracer()
}

ggplot2::autoplot(tr, i)
```

force_all *Force evaluation of all arguments to a function*

Description

Forces the evaluation of all arguments in the calling environment of this function.

Usage

```
force_all(...)
```

Arguments

...	arguments captured by ... are also evaluated if passed via ...

Details

Use `force_all()` as syntactic sugar to force evaluation of all arguments to a function and thereby circumvent lazy evaluation. If called from within a function with `...` as formal argument, use `force_all(...)` to force evaluation of arguments captured by `...`.

Value

NULL invisibly

See Also

`force()`

Examples

```
affine <- function(a, b) {
  function(x) a * x + b
}

a <- 1
b <- 1

affine_11 <- affine(a, b)

a <- 2
b <- 2

affine_11(1) # Gives 4 and not 2 due to lazy evaluation

affine_forced <- function(a, b) {
  force_all()
  function(x) a * x + b
}

a <- 1
b <- 1

affine_11 <- affine_forced(a, b)

a <- 2
b <- 2

affine_11(1) # Gives 2
```

greenland *South west Greenland temperature data*

Description

Monthly average temperatures in degree Celsius in Nuuk and Qaqortoq from 1873 to 2013.

Usage

```
greenland
```

Format

A data frame with 1692 rows and 5 columns:

Year Year (numeric)

Month Month (integer, 1 to 12)

Temp_nuuk Monthly mean temperature in Nuuk

Temp_Qaqortoq Monthly mean temperature in Qaqortoq

Temp_diff Temperature difference, `Temp_nuuk - Temp_Qaqortoq`

Source

https://crudata.uea.ac.uk/cru/data/greenland/

news *Social media news sharing*

Description

A dataset containing features about articles published by Mashable in a period of two years from January 7, 2013, to January 7, 2015. The purpose of collecting the data was to predict the number of shares of the news articles on social networks. Compared to the source, seven features are excluded. Two (url and a timestamp) are not relevant predictors and five are redundant or almost redundant, leading to collinearity.

Usage

```
news
```

Format

A data frame with 39,644 rows and 54 columns:

n_tokens_title Number of words in the title

n_tokens_content Number of words in the content

n_unique_tokens Rate of unique words in the content

num_hrefs Number of links

num_self_hrefs Number of links to other articles published by Mashable

num_imgs Number of images

num_videos Number of videos

average_token_length Average length of the words in the content

num_keywords Number of keywords in the metadata

data_channel_is_lifestyle Is data channel 'Lifestyle'?

data_channel_is_entertainment Is data channel 'Entertainment'?

data_channel_is_bus Is data channel 'Business'?

data_channel_is_socmed Is data channel 'Social Media'?

data_channel_is_tech Is data channel 'Tech'?

data_channel_is_world Is data channel 'World'?

kw_min_min Worst keyword (min. shares)

kw_max_min Worst keyword (max. shares)

kw_avg_min Worst keyword (avg. shares)

kw_max_max Best keyword (max. shares)

kw_avg_max Best keyword (avg. shares)

kw_min_avg Avg. keyword (min. shares)

kw_max_avg Avg. keyword (max. shares)

kw_avg_avg Avg. keyword (avg. shares)

self_reference_min_shares Min. shares of referenced articles in Mashable

self_reference_avg_sharess Avg. shares of referenced articles in Mashable

weekday_is_monday Was the article published on a Monday?

weekday_is_tuesday Was the article published on a Tuesday?

weekday_is_wednesday Was the article published on a Wednesday?

weekday_is_thursday Was the article published on a Thursday?

weekday_is_friday Was the article published on a Friday?

weekday_is_saturday Was the article published on a Saturday?

weekday_is_sunday Was the article published on a Sunday?

LDA_00 Closeness to LDA topic 0

LDA_01 Closeness to LDA topic 1

LDA_02 Closeness to LDA topic 2

LDA_03 Closeness to LDA topic 3
LDA_04 Closeness to LDA topic 4
global_subjectivity Text subjectivity
global_sentiment_polarity Text sentiment polarity
global_rate_positive_words Rate of positive words in the content
global_rate_negative_words Rate of negative words in the content
rate_positive_words Rate of positive words among non-neutral tokens
rate_negative_words Rate of negative words among non-neutral tokens
avg_positive_polarity Avg. polarity of positive words
min_positive_polarity Min. polarity of positive words
max_positive_polarity Max. polarity of positive words
avg_negative_polarity Avg. polarity of negative words
min_negative_polarity Min. polarity of negative words
max_negative_polarity Max. polarity of negative words
title_subjectivity Title subjectivity
title_sentiment_polarity Title polarity
abs_title_subjectivity Absolute subjectivity level
abs_title_sentiment_polarity Absolute polarity level
shares Number of shares (target)

Source

https://archive.ics.uci.edu/dataset/332/online+news+popularity

---

nuuk *Nuuk temperature data*

Description

Annual summaries of monthly mean temperatures in Nuuk from 1867 to 2013.

Usage

```
nuuk
```

Format

A data frame with 147 rows and 6 columns:

Year Year (numeric)
Temperature Annual mean temperature
Median Annual median of monthly mean temperatures
High Annual maximum of monthly mean temperatures
Low Annual minimum of monthly mean temperatures
Range High - Low

Source

https://crudata.uea.ac.uk/cru/data/greenland/

---

plotter *Constructor of plotter expression*

Description

The plotter function returns a quoted expression that adds points to a current plot. For use with tracer and terminator objects to iteratively update plots during long running function evaluations.

Usage

```
plotter(y, col = "black", lty = "solid", pch = 1)
```

Arguments

y	a name of a symbol to plot
col	point and line color
lty	line type
pch	plot symbol

Value

a quoted expression

Examples

```
plotter("i")
```

---

print.terminator *Print terminator information*

Description

Print terminator information

Usage

```
## S3 method for class 'terminator'
print(x, ...)
```

Arguments

x	a terminator object
...	other arguments (currently ignored)

Value

The list of objects from the terminator's evaluation environment, returned invisibly.

Examples

```
term <- terminator(quote(i >= 3), print = FALSE)

for (i in 1:5) {
  if (term$terminator()) {
    break
  }
}

print(term, all.names = TRUE)
```

---

rng_stream *Random number stream based on caching*

Description

A random number stream uses a vectorized random number generator to generate a cache of random numbers that can then be used sequentially. Whenever the cache runs empty new numbers are generated automatically.

Usage

```
rng_stream(m, rng, ...)
```

Arguments

m	initial cache size
rng	a random number generator
...	additional arguments passed to the random number generator

Value

A function that extracts random numbers from the cache and fills the cache whenever it runs empty.

Examples

```
runif_stream <- rng_stream(10, runif, min = -1, max = 1)
runif_stream()
```

summary.tracer *Summarize and print trace information*

Description

Summarize and print trace information

Usage

```
## S3 method for class 'tracer'
summary(object, ...)

## S3 method for class 'tracer'
print(x, ...)
```

Arguments

object	a tracer object
...	other arguments (currently ignored)
x	a tracer object

Value

summary returns a data frame (of class trace) with columns containing the values of the traced objects, and if time is traced an additional column, .time, containing the cumulative runtime in seconds.

Examples

```
tr <- tracer("i", Delta = 0)

for (i in 1:3) {
  tr$tracer()
}

summary(tr)
print(tr)
```

`terminator` *Constructor of a terminator object*

Description

Terminator objects are used to write termination conditions that can be tested via a callback function within another function during its evaluation.

Usage

```
terminator(cond = FALSE, Delta = 1, print = TRUE, plotter = NULL, ...)
```

Arguments

`cond`	a termination condition. Either an expression or call that evaluates to a logical.
`Delta`	an integer specifying how often the termination condition is evaluated. `Delta = 0` means never, and otherwise the condition is evaluated every `Delta`-th iteration. `Delta = 1` is the default.
`print`	a variable name to print or a logical. If FALSE, nothing is printed. If a variable name is given, that variable (if it exists) will be printed every `Delta`-th iteration. If TRUE (the default) the last variable in `cond` is printed every `Delta`-th iteration.
`plotter`	an expression, possibly created by the `plotter` function.
`...`	other arguments passed to `format` for printing and `plot.window`

Details

Terminator objects are similar to tracer objects but serve a different purpose. Like tracer objects, they can be used to trace, print and plot values of a variable within the evaluation environment of another function during its evaluation. The primary purpose of a terminator object is, however, to evaluate a termination condition, which can trigger termination of a loop. Terminator objects do not save trace information and do not trace runtime.

Value

A terminator object containing the functions `terminator` and `clear`.

Examples

```
term <- terminator(quote(i >= 3), print = FALSE)

for (i in 1:5) {
  if (term$terminator()) {
    break
  }
}

i
```

`tracer` *Constructor of a tracer object*

Description

Tracer objects can collect, print and summarize trace information from the evaluation environment of other functions during their evaluation.

Usage

```
tracer(
  objects = NULL,
  Delta = 1,
  save = TRUE,
  time = TRUE,
  expr = NULL,
  plotter = NULL,
  ...
)
```

Arguments

`objects`	a character vector of names of the objects that are to be traced. The objects are searched for in the calling environment of the `tracer` function. Objects created by the `expr` argument can also be traced.
`Delta`	an integer specifying if and how often trace information is printed. `Delta = 0` means never, and otherwise trace information is printed every `Delta`-th iteration. `Delta = 1` is the default.
`save`	a logical value. Determines if the trace information is stored.
`time`	a logical value. Determines if runtime information in seconds is traced.
`expr`	an expression that will be evaluated in an environment that has the calling environment of the `tracer` function as parent.
`plotter`	an expression, possibly created by the `plotter` function.
`...`	other arguments passed to `format` for printing and `plot.window`

Details

The function `tracer` constructs a tracer object containing a `tracer`, a `get` and a `clear` function. A call of the objects `tracer` function can be inserted in other functions and used to collect, print and plot trace information about the internals of that function during its evaluation. The objects `get` function can access that information afterwards, and its `clear` function deletes all stored values in the tracer object.

A tracer object can trace time (in seconds) between `tracer` calls, which are measured by the `hires_time` function from the bench package. There are `print` and `summary` methods available for summarizing the trace information. A call of the `tracer` function can be manually inserted into the body of the function that is to be traced, it can be inserted using `base::trace`, or it can be passed as an argument to any function with a callback argument.

Value

A tracer object containing the functions `tracer`, `get` and `clear`.

Examples

```
test_tracer <- tracer(c("m", "m_sq"), expr = quote(m_sq <- m^2))

test <- function(n, cb = NULL) {
  for(i in 1:n) {
    m <- 2 * i
    Sys.sleep(0.1)
    if (!is.null(cb)) cb()
  }
}

test(10, test_tracer$tracer)
summary(test_tracer)
```

---

vegetables *Vegetable sale*

Description

Weekly sale of frozen vegetables in different stores.

Usage

```
vegetables
```

Format

A data frame with 1066 rows and 3 columns:

`sale` a numeric vector. Number of items sold in a week

`normalSale` a numeric vector. Estimated normal sale that week

`store` a character vector. Id of the different stores

Source

Obtained by package author.

`[.tracer` *Subsetting tracer objects*

Description

Subsetting tracer objects

Usage

```
## S3 method for class 'tracer'
x[i, j, ..., drop = TRUE]
```

Arguments

`x`	a tracer object.
`i`	the indices of the trace information to extract.
`j`	currently ignored.
`...`	other arguments passed on to `get`.
`drop`	simplify a list with one element to a vector if TRUE.

Value

a list

Examples

```
tr <- tracer("i", Delta = 0)

for (i in 1:3) {
  tr$tracer()
}

tr[1]
```

Index