Package: season (via r-universe)

October 8, 2026

Type Package

Title Seasonal Analysis of Health Data

Version 0.3.16.9000

Description Routines for the seasonal analysis of health data, including regression models, time-stratified case-crossover, plotting functions and residual checks, see Barnett and Dobson (2010) ISBN 978-3-642-10748-1. Thanks to Yuming Guo for checking the case-crossover code.

License GPL (>= 3)

URL https://github.com/agbarnett/season, https://agbarnett.github.io/season/

BugReports https://github.com/agbarnett/season/issues

Depends R (>= 4.2.0)

Imports broom, cli, coda, generics, ggplot2 (>= 4.0.3), lifecycle, MASS (>= 7.3.65), rlang, stats, survival (>= 3.8.6), tidyr

Suggests dlnm, dplyr, flextable, knitr, mgcv, nimble, purrr, rmarkdown, spelling, stringr, testthat (>= 3.0.0), tibble, vdiffr, withr

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 8.0.0

Config/pak/sysreqs libicu-dev

Repository https://agbarnett.r-universe.dev

Date/Publication 2026-08-01 05:11:21 UTC

RemoteUrl https://github.com/agbarnett/season

RemoteRef HEAD

RemoteSha 5fe8d32be76292564214b60cff6a2e67e8b6b973

Contents

aaft . . . 3
AFL . . . 4
augment.casecross . . . 5
autoplot.Cosinor . . . 5
autoplot.monthglm . . . 6
autoplot.Monthmean . . . 7
autoplot.nonlintest . . . 8
autoplot.nsCosinor . . . 9
autoplot.peri . . . 10
autoplot.third . . . 11
casecross . . . 11
ciPhase . . . 14
cosinor . . . 15
createAdj . . . 18
CVD . . . 19
CVDdaily . . . 20
exercise . . . 21
glance.casecross . . . 22
indoor . . . 22
invyrfraction . . . 23
monthglm . . . 24
monthmean . . . 26
nonlintest . . . 27
nscosinor . . . 29
peri . . . 31
phasecalc . . . 32
plot.Cosinor . . . 33
plot.monthglm . . . 34
plot.Monthmean . . . 35
plot.nonlintest . . . 36
plot.nsCosinor . . . 37
plot_circle . . . 38
plot_month . . . 39
plotCircle . . . 40
plotCircular . . . 41
plotMonth . . . 42
print.casecross . . . 44
print.monthglm . . . 45
print.Monthmean . . . 45
print.nonlintest . . . 46
print.nsCosinor . . . 47
print.summary.monthglm . . . 48

print.summary.nsCosinor 49
schz 50
seasrescheck 51
sinusoid 52
stillbirth 53
summary.casecross 53
summary.Cosinor 55
summary.monthglm 57
summary.nsCosinor 58
third 59
tidy.casecross 61
wtest 62
yrfraction 63

Index **64**

`aaft` *Amplitude Adjusted Fourier Transform (AAFT)*

Description

Generates random linear surrogate data of a time series with the same second-order properties.

Usage

```
aaft(data, nsur)
```

Arguments

`data`	a vector of equally spaced numeric observations (time series).
`nsur`	the number of surrogates to generate (1 or more).

Details

The AAFT uses phase-scrambling to create a surrogate of the time series that has a similar spectrum (and hence similar second-order statistics). The AAFT is useful for testing for non-linearity in a time series, and is used by `nonlintest`.

Value

a matrix of the `nsur` surrogates.

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

References

Kugiumtzis D (2000) Surrogate data test for nonlinearity including monotonic transformations, *Phys. Rev. E*, vol 62 no. 1, 2000. doi:10.1103/PhysRevE.62.R25

Examples

```
aaft(CVD$cvd, nsur = 1)
surr <- aaft(CVD$cvd, nsur = 1)
plot(CVD$cvd, type = "l")
lines(surr[ ,1], col = "red")
```

AFL *Australian Football League (AFL) Players' Birthdays for the 2009 Season*

Description

The data are: a) the monthly frequencies of birthdays and an expected number based on monthly birth statistics for 1975 to 1991. b) all 617 players' initials and birthdays (excluding non-Australian born players).

Usage

```
AFL
```

Format

A list with the following 5 variables:

- month: integer month (1 to 12)
- players: number of players born in each month (12 observations)
- expected: expected number of players born in each month (12 observations)
- initials: player initials (617 observations)
- dob: date of birth in date format (617 observations; year-month-day format)

Source

Dates of birth from Wikipedia.

Examples

```
barplot(AFL$players, names.arg=month.abb)
```

augment.casecross *Augment*

Description

These objects are imported from other packages. Follow the links below to see their documentation.

generics augment, glance, tidy

Usage

```
## S3 method for class 'casecross'
augment(x, ...)
```

Arguments

x	Model object or other R object with information to append to observations.
...	Addition arguments to augment method.

autoplot.Cosinor *Plot the fitted sinusoid from a* cosinor() *model*

Description

Returns a ggplot of the fitted sinusoid. The y-axis is on the probability scale for `logit` and `cloglog` link functions; the x-axis switches between month, year, and hour depending on the `type` of the model.

Usage

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

Arguments

object	a `Cosinor` object produced by `cosinor()`.
...	unused, for S3 generic compatibility.

Value

a ggplot object.

Author(s)

Nicholas Tierney

See Also

cosinor(), summary.Cosinor(), seasrescheck()

Examples

```
res <- cosinor(
  cvd ~ 1,
  date = "month",
  data = CVD,
  type = "monthly",
  family = poisson(),
  offsetmonth = TRUE
)
autoplot(res)
autoplot(res) + ggplot2::theme_minimal()
```

---

autoplot.monthglm *Plot the monthly estimates from* monthglm()

Description

Returns a ggplot of the per-month coefficient estimates from a monthglm() fit, with a confidence interval per month and (for Poisson/binomial families) a horizontal reference line at the rate / odds ratio of 1.

Usage

```
## S3 method for class 'monthglm'
autoplot(object, alpha = 0.05, ...)
```

Arguments

object	a monthglm object produced by monthglm().
alpha	statistical significance level for the confidence intervals (default 0.05).
...	unused, for S3 generic compatibility.

Value

a ggplot object.

Author(s)

Nicholas Tierney

See Also

monthglm()

Examples

```
mmodel <- monthglm(
  formula = cvd ~ 1,
  data = CVD,
  family = poisson(),
  offsetpop = expression(pop / 100000),
  offsetmonth = TRUE,
  refmonth = 6
)
autoplot(mmodel)
autoplot(mmodel) + ggplot2::labs(x = "Month", y = "Rate ratio")
```

---

autoplot.Monthmean *Plot the monthly mean estimates from* monthmean()

Description

Returns a ggplot of the per-month means computed by monthmean().

Usage

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

Arguments

object	a Monthmean object produced by monthmean().
...	unused, for S3 generic compatibility.

Value

a ggplot object.

Author(s)

Nicholas Tierney

See Also

monthmean()

Examples

```
mmean <- monthmean(
  data = CVD,
  resp = "cvd",
  offsetpop = expression(pop / 100000),
  adjmonth = "average"
)
autoplot(mmean)
autoplot(mmean) + ggplot2::theme_minimal()
```

`autoplot.nonlintest` *Plot the region of the third-order moment outside the test limits*

Description

Returns a ggplot contour plot of the third-order moment region that exceeds the bootstrap test limits computed by `nonlintest()`. Returns `NULL` invisibly with an informational message if no points exceed the limits.

Usage

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

Arguments

`object`	a `nonlintest` object produced by `nonlintest()`.
`...`	unused, for S3 generic compatibility.

Value

a ggplot contour plot, or `NULL` invisibly when no points exceed the test limits.

Author(s)

Nicholas Tierney

See Also

`nonlintest()`

Examples

```
## Not run:
test_res <- nonlintest(data = CVD$cvd, n.lag = 4, n.boot = 1000)
autoplot(test_res)

## End(Not run)
```

`autoplot.nsCosinor` *Plot the trend and seasonal estimates from* `nscosinor()`

Description

Returns a ggplot of the trend and seasonal components estimated by `nscosinor()`, faceted by component and with a ribbon for the confidence interval.

Usage

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

Arguments

`object`	an `nsCosinor` object produced by `nscosinor()`.
`...`	unused, for S3 generic compatibility.

Value

a ggplot object faceted by trend / season(s).

Author(s)

Nicholas Tierney

See Also

`nscosinor()`

Examples

```
## Not run:
res <- nscosinor(
  data = CVD, response = "adj", cycles = 12,
  niters = 200, burnin = 100, tau = c(10, 50)
)
autoplot(res)
autoplot(res) + ggplot2::theme_minimal()

## End(Not run)
```

`autoplot.peri` *Plot the periodogram from* `peri()`

Description

Produce a ggplot of the periodogram in both radians and cycles. The returned ggplot can be extended with + (e.g. + `ggplot2::theme_minimal()`).

Usage

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

Arguments

`object`	a "`peri`" object produced by `peri()`.
`...`	unused, for S3 generic compatibility.

Value

a ggplot object.

Author(s)

Nicholas Tierney

See Also

`peri()`

Examples

```
p <- peri(CVD$cvd)
autoplot(p)
autoplot(p) + ggplot2::theme_minimal()
```

autoplot.third *Plot the third-order moment from* third()

Description

Produce a ggplot contour of the third-order moment over its non-redundant region.

Usage

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

Arguments

object	a "third" object produced by third().
...	unused, for S3 generic compatibility.

Value

a ggplot contour plot.

Author(s)

Nicholas Tierney

See Also

third()

Examples

```
t <- third(CVD$cvd, n.lag = 4)
autoplot(t)
```

casecross *Case–crossover Analysis to Control for Seasonality*

Description

Fits a time-stratified case–crossover to regularly spaced time series data. The function is not suitable for irregularly spaced individual data. The function only uses a time-stratified design, and other designs such as the symmetric bi-directional design, are not available.

Usage

```
casecross(
  formula,
  data,
  exclusion = 2,
  stratalength = 28,
  matchdow = FALSE,
  usefinalwindow = TRUE,
  matchconf = NULL,
  confrange = 0,
  stratamonth = FALSE,
  date_col = "date"
)
```

Arguments

formula	formula. The dependent variable should be an integer count (e.g., daily number of deaths).
data	data set as a data frame.
exclusion	exclusion period (in days) around cases, set to 2 (default). Must be greater than or equal to zero and smaller than stratalength.
stratalength	length of stratum in days, set to 28 (default).
matchdow	match case and control days using day of the week (TRUE/default=FALSE). This matching is in addition to the strata matching.
usefinalwindow	use the last stratum in the time series, which is likely to contain less days than all the other strata (TRUE/default=FALSE).
matchconf	match case and control days using an important confounder (optional; must be in quotes). matchconf is the variable to match on. This matching is in addition to the strata matching. Default is NULL - no confounder is used.
confrange	range of the confounder within which case and control days will be treated as a match (optional). Range = matchconf (on case day) $+/-$ confrange.
stratamonth	use strata based on months, default=FALSE. Instead of a fixed strata size when using stratalength.
date_col	Character. column name for date variable. Default is "date".

Details

The case–crossover method compares "case" days when events occurred (e.g., deaths) with control days to look for differences in exposure that might explain differences in the number of cases. Control days are selected to be nearby to case days, which means that only recent changes in the independent variable(s) are compared. By only comparing recent values, any long-term or seasonal variation in the dependent and independent variable(s) can be eliminated. This elimination depends on the definition of nearby and on the seasonal and long-term patterns in the independent variable(s).

Control and case days are only compared if they are in the same stratum. The stratum is controlled by stratalength, the default value is 28 days, so that cases and controls are compared in four

week sections. Smaller stratum lengths provide a closer control for season, but reduce the available number of controls. Control days that are close to the case day may have similar levels of the independent variable(s). To reduce this correlation it is possible to place an `exclusion` around the cases. The default is 2, which means that the smallest gap between a case and control will be 3 days.

To remove any confounding by day of the week it is possible to additionally match by day of the week (`matchdow`), although this usually reduces the number of available controls. This matching is in addition to the strata matching.

It is possible to additionally match case and control days by an important confounder (`matchconf`) in order to remove its effect. Control days are matched to case days if they are: i) in the same strata, ii) have the same day of the week if `matchdow=TRUE`, iii) have a value of `matchconf` that is within plus/minus `confrange` of the value of `matchconf` on the case day.

If the range is set too narrow then the number of available controls will become too small, which in turn means the number of case days with at least one control day is compromised.

The method uses conditional logistic regression (see `survival::coxph()`), so the parameter estimates are odds ratios.

The code assumes that the data frame contains a date variable (in `Date()` format) called "date".

Value

a list with the following elements:

- call: the original call to the casecross function.
- cox_model: conditional logistic regression model of class `coxph`.
- n_cases: total number of cases.
- n_case_days: number of case days with at least one control day.
- n_control_days: average number of control days per case day.

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

References

Janes, H., Sheppard, L., Lumley, T. (2005) Case-crossover analyses of air pollution exposure data: Referent selection strategies and their implications for bias. *Epidemiology* 16(6), 717–726. doi:10.1097/01.ede.0000181315.18836.9d.

Barnett, A.G., Dobson, A.J. (2010) *Analysing Seasonal Health Data*. Springer. doi:10.1007/9783-642107481

See Also

`summary.casecross`, `coxph`

Examples

```
# cardiovascular disease data
# subset for example
CVDdaily <- subset(CVDdaily, date <= as.Date('1987-12-31'))
# Effect of ozone on CVD death
model1 <- casecross(
  cvd ~ o3mean + tmpd + Mon + Tue + Wed + Thu + Fri + Sat,
  data = CVDdaily
)
summary(model1)
# match on day of the week
model2 <- casecross(cvd ~ o3mean + tmpd, matchdow = TRUE, data = CVDdaily)
summary(model2)
# match on temperature to within a degree
model3 <- casecross(
  cvd ~ o3mean + Mon + Tue + Wed + Thu + Fri + Sat,
  data = CVDdaily,
  matchconf = "tmpd",
  confrange = 1
)
summary(model3)
```

---

ciPhase *Mean and Confidence Interval for Circular Phase*

Description

Calculates the mean and confidence interval for the phase based on a chain of MCMC samples.

Usage

```
ciPhase(theta, alpha = 0.05)
```

Arguments

theta	chain of Markov chain Monte Carlo (MCMC) samples of the phase.
alpha	the confidence level (default = 0.05 for a 95\ interval).

Details

The estimates of the phase are rotated to have a centre of π, the point on the circumference of a unit radius circle that is furthest from zero. The mean and confidence interval are calculated on the rotated values, then the estimates are rotated back.

Value

a tibble with the following columns:

- mean: the estimated mean phase.
- lower: the estimated lower limit of the confidence interval.
- upper: the estimated upper limit of the confidence interval.

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

References

Fisher, N. (1993) *Statistical Analysis of Circular Data*. Cambridge University Press. Page 36.

Barnett, A.G., Dobson, A.J. (2010) *Analysing Seasonal Health Data*. Springer. doi:10.1007/9783-642107481

Examples

```
# 2000 normal samples, centred on zero
theta <- rnorm(n = 2000, mean = 0, sd = pi / 50)
hist(theta, breaks = seq(-pi / 8, pi / 8, pi / 30))
ciPhase(theta)
```

`cosinor` *Cosinor Regression Model for Detecting Seasonality in Yearly Data or Circadian Patterns in Hourly Data*

Description

Fits a cosinor model as part of a generalized linear model.

Usage

```
cosinor(
  formula,
  date,
  data,
  family = stats::gaussian(),
  alpha = 0.05,
  cycles = 1,
  rescheck = FALSE,
  type = c("daily", "weekly", "monthly", "hourly"),
  offsetmonth = FALSE,
  offsetpop = NULL,
  text = TRUE
)
```

Arguments

formula	regression formula.
date	a date variable if type="daily", or an integer between 1 and 53 if type="weekly", or an integer between 1 and 12 if type="monthly", or a POSIXct date if type="hourly".
data	data set as a data frame.
family	a description of the error distribution and link function to be used in the model. Available link functions: stats::gaussian() (default), identity(), log(), logit(), cloglog(). Note, it must have the parentheses.
alpha	Significance level. Default is 0.05.
cycles	number of seasonal cycles per year if type="daily", "weekly" or "monthly"; number of cycles per 24 hours if type="hourly"
rescheck	plot the residual checks (TRUE/FALSE), see seasrescheck().
type	"daily" for daily data (default), or "weekly" for weekly data, or "monthly" for monthly data, or "hourly" for hourly data.
offsetmonth	include an offset to account for the uneven number of days in the month (TRUE/FALSE). Should be used for monthly counts (with family=poisson()).
offsetpop	include an offset for the population (optional), this should be a variable in the data frame. Do not log-transform the offset as the log-transform is applied by the function. This should be an expression, as given in the example below.
text	add explanatory text to the returned phase value (TRUE) or return a number (FALSE). Passed to the invyrfraction() function.

Details

The cosinor model captures a seasonal pattern using a sinusoid. It is therefore suitable for relatively simple seasonal patterns that are symmetric and stationary. The default is to fit an annual seasonal pattern (cycle=1), but other higher frequencies are possible (e.g., twice per year: cycle=2). The model is fitted using a sine and cosine term that together describe the sinusoid. These parameters are added to a generalized linear model, so the model can be fitted to a range of dependent data (e.g., Normal, Poisson, Binomial). Unlike the nscosinor() model, the cosinor model can be applied to unequally spaced data.

Value

Returns an object of class "Cosinor" with the following parts:

- call: the original call to the cosinor function.
- glm: an object of class glm (see glm()).
- fitted: fitted values for intercept and cosinor only (ignoring other independent variables).
- fitted.plus: standard fitted values, including all other independent variables.
- residuals: residuals.
- date: name of the date variable (in Date format when type = daily).

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

References

Barnett, A.G., Dobson, A.J. (2010) *Analysing Seasonal Health Data*. Springer. doi:10.1007/9783-642107481

See Also

summary.Cosinor(), plot.Cosinor()

Examples

```
## cardiovascular disease data (offset based on number of days in...
## ...the month scaled to an average month length)
res <- cosinor(
  cvd ~ 1,
  date = 'month',
  data = CVD,
  type = 'monthly',
  family = poisson(),
  offsetmonth = TRUE
)
summary(res)
seasrescheck(res$residuals) # check the residuals
## stillbirth data
res <- cosinor(
  stillborn ~ 1,
  date = 'dob',
  data = stillbirth,
  family = binomial(link = 'cloglog')
)
summary(res)
plot(res)
## hourly indoor temperature data
res <- cosinor(
  bedroom ~ 1,
  date = 'datetime',
  type = 'hourly',
  data = indoor
)

summary(res)
# to get the p-values for the sine and cosine estimates
summary(res$glm)
```

createAdj *Creates an Adjacency Matrix*

Description

Creates an adjacency matrix in a form suitable for use with nimble

Usage

```
createAdj(matrix, suffix = NULL)
```

Arguments

matrix	square matrix with 1's for neighbours and NA's for non-neighbours.
suffix	string to be appended to "num", "adj" and "weights" object names

Details

Adjacency matrices are used by conditional autoregressive (CAR) models to smooth estimates according to some neighbourhood map. The basic idea is that neighbouring areas have more in common than non-neighbouring areas and so will be positively correlated.

As well as correlations in space it is possible to use CAR models to model similarities in time.

In this case the matrix represents those time points that we wish to assume to be correlated.

Value

A list of:

- num: the total number of neighbours
- adj: the index number of the adjacent neighbours
- weights: weights

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

Examples

```
# Nearest neighbour matrix for 5 time points
x <- c(NA,1,NA,NA,NA)
V <- toeplitz(x)
V
createAdj(V)
```

CVD *Cardiovascular Deaths in Los Angeles, 1987–2000*

Description

Monthly number of deaths from cardiovascular disease in people aged 75 and over in Los Angeles for the years 1987 to 2000.

Usage

```
CVD
```

Format

A data frame with 168 observations on the following 8 variables:

- year: year of death
- month: month of death
- yrmon: a combination of year and month: $year + (month - 1)/12$
- cvd: monthly number of CVD deaths
- tmpd: mean monthly temperature (degrees Fahrenheit)
- pop: Los Angeles population aged 75+ in the year 2000 (this value is constant as only one year was available, but in general the population will change over time)
- ndaysmonth: number of days in each month (used as an offset)
- adj: adjusted number of CVD deaths per month using a standardised month length. Monthly number of CVD deaths multiplied by (365.25/12)/ndaysmonth. So the standard month length is 30.4 days.

Source

From the NMMAPS study.

References

Samet JM, Dominici F, Zeger SL, Schwartz J, Dockery DW (2000). *The National Morbidity, Mortality, and Air Pollution Study, Part I: Methods and Methodologic Issues*. Research Report 94, Health Effects Institute, Cambridge MA.

Examples

```
plot(
  CVD$yrmon,
  CVD$cvd,
  type = 'o',
  xlab = 'Date',
  ylab = 'Number of CVD deaths per month'
  )
```

`CVDdaily` *Daily Cardiovascular Deaths in Los Angeles, 1987–2000*

Description

Daily number of deaths from cardiovascular disease in people aged 75 and over in Los Angeles for the years 1987 to 2000.

Usage

```
CVDdaily
```

Format

A data frame with 5114 observations on the following 16 variables:

- date: date of death in date format (year-month-day)
- cvd: daily number of CVD deaths
- dow: day of the week (character)
- tmpd: daily mean temperature (degrees Fahrenheit)
- o3mean: daily mean ozone (parts per billion)
- o3tmean: daily trimmed mean ozone (parts per billion)
- Mon: indicator variable for Monday
- Tue: indicator variable for Tuesday
- Wed: indicator variable for Wednesday
- Thu: indicator variable for Thursday
- Fri: indicator variable for Friday
- Sat: indicator variable for Saturday
- month: month (integer from 1 to 12)
- winter: indicator variable for winter
- spring: indicator variable for spring
- summer: indicator variable for summer
- autumn: indicator variable for autumn

Source

From the NMMAPS study.

References

Samet JM, Dominici F, Zeger SL, Schwartz J, Dockery DW (2000). *The National Morbidity, Mortality, and Air Pollution Study, Part I: Methods and Methodologic Issues*. Research Report 94, Health Effects Institute, Cambridge MA.

Examples

```
plot(
  CVDdaily$date,
  CVDdaily$cvd,
  type = 'p',
  xlab = 'Date',
  ylab = 'Number of CVD deaths')
```

`exercise` *Exercise Data from Queensland, 2005–2007*

Description

Exercise data in longitudinal format from a physical activity intervention study in Logan, Queensland. Some subjects were lost to follow-up, so all three visits are not available for all subjects.

Usage

```
exercise
```

Format

A data frame with 1302 observations on the following 7 variables:

- id: subject number
- visit: visit number (1, 2 or 3)
- date: date of interview (year-month-day)
- year: year of interview
- month: month of interview
- bmi: body mass index at visit 1 (kg/m^2)
- walking: walking time per week (in minutes) at each visit

Source

From Prof Elizabeth Eakin and colleagues, The University of Queensland, Brisbane.

References

Eakin E, et al (2009) Telephone counselling for physical activity and diet in type 2 diabetes and hypertension, *Am J of Prev Med*, vol 36, pages 142–9

Examples

```
boxplot(exercise$walking ~ exercise$month)
```

glance.casecross *Construct a single row summary "glance" of a model, fit, or other object*

Description

Construct a single row summary "glance" of a model, fit, or other object

Usage

```
## S3 method for class 'casecross'
glance(x, ...)
```

Arguments

x	model or other R object to convert to single-row data frame.
...	other arguments passed to methods.

Examples

```
# subset for example
CVDdaily <- subset(CVDdaily, date <= as.Date('1987-12-31'))
# Effect of ozone on CVD death
model1 <- casecross(
  cvd ~ o3mean + tmpd + Mon + Tue + Wed + Thu + Fri + Sat,
  data = CVDdaily
)
summary(model1)
# one-line summary for each model
glance(model1)
```

indoor *Indoor Temperature Data*

Description

The data are indoor temperatures (in degrees C) for a bedroom and living room in a house in Brisbane, Australia for the dates 10 July 2013 to 3 October 2013. Temperatures were recorded using data loggers which recorded every hour to the nearest 0.5 degrees.

Format

A `data.frame` with the following 3 variables:

- datetime: date and time in `POSIXlt` format
- living: the living room temperature
- bedroom: the bedroom temperature

Source

Adrian G Barnett.

Examples

```
res <- cosinor(
 bedroom ~ 1,
 date = 'datetime',
 type = 'hourly',
 data = indoor
)
summary(res)
```

`invyrfraction` *Inverts a fraction of the year or hour to a useful time scale.*

Description

Returns the day and month (for "daily") or fraction of the month (for "monthly") given a fraction of the year. Assumes a year length of 365.25 days for "daily". When using "monthly" the 1st of January is 1, the 1st of December is 12, and the 31st of December is 12.9. For "hourly" it returns the fraction of the 24-hour clock starting from zero (midnight).

Usage

```
invyrfraction(
  frac,
  type = c("daily", "monthly", "hourly", "weekly"),
  text = TRUE
)

invyrfraction_chr(frac, type = c("daily", "monthly", "hourly", "weekly"))

invyrfraction_num(frac, type = c("daily", "monthly", "hourly", "weekly"))
```

Arguments

`frac`	a vector of fractions of the year, all between 0 and 1.
`type`	"daily" for dates, "monthly" for months, "hourly" for hours, "weekly" for weeks.
`text`	add an explanatory text to the returned value (TRUE) or return a number (FALSE).

Value

the date (day and month for "daily"), fractional month (for "monthly"), or fraction of the 24-hour clock (for "hourly").

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

Examples

```
invyrfraction(c(0, 0.5, 0.99), type = "hourly")
invyrfraction(c(0, 0.5, 0.99), type = "daily")
invyrfraction(c(0, 0.5, 0.99), type = "weekly")
invyrfraction(c(0, 0.5, 0.99), type = "monthly")

# Also provide _chr and _num functions that mean you don't specify arg,
# `text = TRUE` or `text = FALSE`
invyrfraction_num(c(0, 0.5, 0.99), type = "weekly")
invyrfraction_chr(c(0, 0.5, 0.99), type = "weekly")
```

`monthglm` *Fit a GLM with Month*

Description

Fit a generalized linear model with a categorical variable of month.

Usage

```
monthglm(
  formula,
  data,
  family = stats::gaussian(),
  refmonth = 1,
  monthvar = "month",
  offsetmonth = FALSE,
  offsetpop = NULL
)
```

Arguments

`formula`	regression model formula, e.g., `y~x1+x2`, (do not add month to the regression equation, it will be added automatically).
`data`	a data frame. Must contain the column "months" as integer, and year as a 4 digit number.
`family`	a description of the error distribution and link function to be used in the model (default=`gaussian()`). (See `family()` for details of family functions.).
`refmonth`	reference month, must be between 1 and 12 (default=1 for January).
`monthvar`	name of the month variable which is either an integer (1 to 12) or a character or factor (`Jan' to` Dec' or `January' to` December') (default='month').

`offsetmonth` include an offset to account for the uneven number of days in the month (TRUE/FALSE). Should be used for monthly counts (with `family=poisson()`).

`offsetpop` include an offset for the population (optional), this should be a variable in the data frame. Do not log-transform the offset as the log-transform is applied by the function. This should be an expression, as given in the example below.

Details

Month is fitted as a categorical variable as part of a generalized linear model. Other independent variables can be added to the right-hand side of `formula`.

This model is useful for examining non-sinusoidal seasonal patterns. For sinusoidal seasonal patterns see `cosinor()`.

The data frame should contain the integer months and the year as a 4 digit number. These are used to calculate the number of days in each month accounting for leap years.

Value

a list with the following elements:

- call: the original call to the monthglm function.
- fit: GLM model.
- fitted: fitted values.
- residuals: residuals.
- out: details on the monthly estimates.

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

References

Barnett, A.G., Dobson, A.J. (2010) *Analysing Seasonal Health Data*. Springer. doi:10.1007/9783-642107481

See Also

`summary.monthglm`, `plot.monthglm`

Examples

```
model <- monthglm(
  formula = cvd~1,
  data = CVD,
  family = poisson(),
  offsetpop = expression(pop/100000),
  offsetmonth = TRUE
  )
summary(model)
```

`monthmean` *Calculate monthly mean or adjusted monthly mean for count data.*

Description

For time series recorded at monthly intervals it is often useful to examine (and plot) the average in each month. When using count data we should adjust the mean to account for the unequal number of days in the month (e.g., 31 in January and 28 or 29 in February).

Usage

```
monthmean(
  data,
  resp,
  offsetpop = NULL,
  adjmonth = c("none", "thirty", "average")
)
```

Arguments

`data`	Data frame with variables "month" and "year"
`resp`	Response variable in data for which the means will be calculated.
`offsetpop`	optional population, used as an offset (default=NULL).
`adjmonth`	adjust monthly counts and scale to a 30 day month ("thirty") or the average month length ("average") (default="none").

Details

This function assumes that the data set (`data`) contains variables for the year and month called year and month, respectively.

Value

Returns an object of class "Monthmean" with the following parts:

- mean: a vector of length 12 with the monthly means.

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

References

Barnett, A.G., Dobson, A.J. (2010) *Analysing Seasonal Health Data*. Springer. doi:10.1007/9783-642107481

See Also

plot.Monthmean()

Examples

```
# cardiovascular disease data
mmean <- monthmean(
  data=CVD,
  resp='cvd',
  offsetpop = expression(pop/100000),
  adjmonth = 'average'
  )
mmean
plot(mmean)

mmean <- monthmean(
  data=CVD,
  resp='cvd',
  offsetpop = expression(pop/100000),
  adjmonth = 'thirty'
  )
```

---

nonlintest *Test of Non-linearity of a Time Series*

Description

A bootstrap test of non-linearity in a time series using the third-order moment.

Usage

```
nonlintest(data, n.lag, n.boot, alpha = 0.05)
```

Arguments

data	a vector of equally spaced numeric observations (time series).
n.lag	the number of lags tested using the third-order moment, maximum = length of time series.
n.boot	the number of bootstrap replications (suggested minimum of 100; 1000 or more would be better).
alpha	statistical significance level of test (default=0.05).

Details

The test uses `aaft` to create linear surrogates with the same second-order properties, but no (third-order) non-linearity. The third-order moments (`third`) of these linear surrogates and the actual series are then compared from lags 0 up to `n.lag` (excluding the skew at the co-ordinates (0,0)). The bootstrap test works on the overall area outside the limits, and gives an indication of the overall non-linearity. The plot using `region` shows those co-ordinates of the third order moment that exceed the null hypothesis limits, and can be a useful clue for guessing the type of non-linearity. For example, a large value at the co-ordinates (0,1) might be caused by a bi-linear series $X_t = \alpha X_{t-1}\varepsilon_{t-1} + \varepsilon_t$.

Value

Returns an object of class "nonlintest" with the following parts:

- region: the region of the third order moment where the test exceeds the limits (up to `n.lag`).
- n.lag: the maximum lag tested using the third-order moment.
- stats: a list of statistics for the area outside the test limits:
  - outside: the total area outside of limits (summed over the whole third-order moment).
  - stan: the total area outside the limits divided by its standard deviation to give a standardised estimate.
  - median: the median area outside the test limits.
  - upper: the (1-`alpha`)th percentile of the area outside the limits.
  - pvalue: bootstrap p-value of the area outside the limits to test if the series is linear.
  - test: reject the null hypothesis that the series is linear (TRUE/FALSE).

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

References

Barnett AG & Wolff RC (2005) A Time-Domain Test for Some Types of Nonlinearity, *IEEE Transactions on Signal Processing*, vol 53, pages 26–33 doi: 10.1109/TSP.2004.838942.

See Also

`print.nonlintest()` `plot.nonlintest()`

Examples

```
## Not run:
test_res <- nonlintest(data = CVD$cvd, n.lag = 4, n.boot = 1000)

## End(Not run)
```

nscosinor *Non-stationary Cosinor*

Description

Decompose a time series using a non-stationary cosinor for the seasonal pattern.

Usage

```
nscosinor(
  data,
  response,
  cycles,
  niters = 1000,
  burnin = 500,
  tau,
  lambda = 1/12,
  div = 50,
  monthly = TRUE,
  alpha = 0.05,
  year_col = "year",
  month_col = "month"
)
```

Arguments

data	Data frame. Assumes no missing data. Year and month columns are by default assumed to be "year", and "month", respectively. These can be specified with arguments year_col and month_col.
response	character. Response variable.
cycles	vector of cycles in units of time, e.g., for a six and twelve month pattern cycles=c(6,12).
niters	total number of MCMC samples (default=1000).
burnin	number of MCMC samples discarded as a burn-in (default=500).
tau	vector of smoothing parameters, tau[1] for trend, tau[2] for 1st seasonal parameter, tau[3] for 2nd seasonal parameter, etc. Larger values of tau allow more change between observations and hence a greater potential flexibility in the trend and season.
lambda	distance between observations (lambda=1/12 for monthly data, default).
div	divisor at which MCMC sample progress is reported (default=50).
monthly	TRUE for monthly data.
alpha	Statistical significance level used by the confidence intervals.
year_col	character. column referring to year. Default is "year".
month_col	character. column referring to month. Default is "month".

Details

This model is designed to decompose an equally spaced time series into a trend, season(s) and noise. A seasonal estimate is estimated as $s_t = A_t \cos(\omega_t - P_t)$, where t is time, A_t is the non-stationary amplitude, P_t is the non-stationary phase and ω_t is the frequency.

A non-stationary seasonal pattern is one that changes over time, hence this model gives potentially very flexible seasonal estimates.

The frequency of the seasonal estimate(s) are controlled by `cycle`. The cycles should be specified in units of time. If the data is monthly, then setting `lambda=1/12` and `cycles=12` will fit an annual seasonal pattern. If the data is daily, then setting `lambda= 1/365.25` and `cycles=365.25` will fit an annual seasonal pattern. Specifying `cycles= c(182.6,365.25)` will fit two seasonal patterns, one with a twice-annual cycle, and one with an annual cycle.

The estimates are made using a forward and backward sweep of the Kalman filter. Repeated estimates are made using Markov chain Monte Carlo (MCMC). For this reason the model can take a long time to run. To give stable estimates a reasonably long sample should be used (`niters`), and the possibly poor initial estimates should be discarded (`burnin`).

Value

Returns an object of class "nsCosinor" with the following parts:

- call: the original call to the nscosinor function.
- time: the year and month for monthly data.
- trend: mean trend and 95\
- season: mean season(s) and 95\
- oseason: overall season(s) and 95\ be the same as `season` if there is only one seasonal cycle.
- fitted: fitted values and 95\ season(s).
- residuals: residuals based on mean trend and season(s).
- n: the length of the series.
- chains: MCMC chains (of class mcmc) of variance estimates: standard error for overall noise (std.error), standard error for season(s) (std.season), phase(s) and amplitude(s).
- cycles: vector of cycles in units of time.

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

References

Barnett, A.G., Dobson, A.J. (2010) *Analysing Seasonal Health Data*. Springer. doi:10.1007/9783-642107481

Barnett, A.G., Dobson, A.J. (2004) Estimating trends and seasonality in coronary heart disease *Statistics in Medicine*. 23(22) 3505–23.

See Also

`plot.nsCosinor()` `summary.nsCosinor()`

Examples

```
# model to fit an annual pattern to the monthly cardiovascular disease data
f <- c(12)
tau <- c(10,50)
## Not run:
  res12 <- nscosinor(
    data = CVD,
    response = 'adj',
    cycles = f,
    niters = 200,
    burnin = 50,
    tau = tau
    )
summary(res12)
# autoplot replaces plot method - plot(res12)
autoplot(res12)

## End(Not run)
```

---

peri *Periodogram*

Description

Estimated periodogram using the fast Fourier transform (`fft`).

Usage

```
peri(data, adjmean = TRUE, plot = lifecycle::deprecated())
```

Arguments

`data`	a data frame.
`adjmean`	subtract the mean from the series before calculating the periodogram (default=TRUE).
`plot`	**[Deprecated]** Use `autoplot.peri()` on the returned object instead.

Value

an object of class `"peri"` (a tibble) with the columns:

- peri: periodogram, I(ω).
- freq_radians: frequencies in radians, ω.
- freq_cycles: frequencies in cycles of time, $2\pi/\omega$.
- amp: amplitude periodogram.
- phase: phase periodogram.

Pass the result to autoplot() to draw the plot.

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

See Also

autoplot.peri()

Examples

```
p <- peri(CVD$cvd)
autoplot(p)
```

---

phasecalc *Phase from Cosinor Estimates*

Description

Calculate the phase given the estimated sine and cosine values from a cosinor model. Returns the phase in radians, in the range $[0, 2\pi)$. The phase is the peak in the sinusoid. Applies atan2() over a branching workflow for each coordinate. See https://en.wikipedia.org/wiki/Atan2 for more information.

Usage

```
phasecalc(cosine, sine)
```

Arguments

cosine	estimated cosine value from a cosinor model.
sine	estimated sine value from a cosinor model.

Value

the estimated phase in radians.

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

References

Fisher, N.I. (1993) *Statistical Analysis of Circular Data*. Cambridge University Press, Cambridge. Page 31.

Examples

```
# pi/2
phasecalc(cosine=0, sine=1)
```

`plot.Cosinor` *Plot the Results of a Cosinor*

Description

[Deprecated] Soft-deprecated in favour of `autoplot.Cosinor()`, which returns a ggplot object you can extend with +. The base R plot below still works.

Usage

```
## S3 method for class 'Cosinor'
plot(x, ...)
```

Arguments

x	a `Cosinor` object produced by `cosinor()`
...	additional arguments passed to the sinusoid plot.

Details

The code produces the fitted sinusoid based on the intercept and sinusoid. The y-axis is on the scale of probability if the link function is "logit" or "cloglog". If the analysis was based on monthly data then month is shown on the x-axis. If the analysis was based on daily data then time is shown on the x-axis.

Value

connected line plot of fitted sinusoid object produced by cosinor.

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

See Also

`autoplot.Cosinor()`, `cosinor()`, `summary.Cosinor()`, `seasrescheck()`

Examples

```
## cardiovascular disease data (offset based on number of days in...
## ...the month scaled to an average month length)
res <- cosinor(
  cvd ~ 1,
  date = 'month',
  data = CVD,
  type = 'monthly',
  family = poisson(),
  offsetmonth = TRUE
  )
# Recommended:
autoplot(res)
# Still works, but deprecated:
plot(res)
```

`plot.monthglm` *Plot of Monthly Estimates*

Description

[Deprecated] Soft-deprecated in favour of `autoplot.monthglm()`, which returns a ggplot object. The base R `plot()` will still work, but we recommend using `autoplot.monthglm()`.

Usage

```
## S3 method for class 'monthglm'
plot(x, alpha = 0.05, ylim = NULL, xlab = "", ylab = "", ...)
```

Arguments

`x`	a `monthglm` object produced by `monthglm()`.
`alpha`	statistical significance level of confidence intervals.
`ylim`	y coordinates ranges (the default is NULL, and the limits are automatically calculated).
`xlab, ylab`	x and y labels. Defaults is no label: "".
`...`	additional arguments passed to `plot()`.

Value

Plot of the estimated from a generalized linear model with a categorical variable of month.

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

See Also

autoplot.monthglm(), monthglm()

Examples

```
mmodel <- monthglm(
  formula = cvd ~ 1,
  data = CVD,
  family = poisson(),
  offsetpop = expression(pop / 100000),
  offsetmonth = TRUE,
  refmonth = 6
)
# Recommended:
autoplot(mmodel)
# Still works, but deprecated:
plot(mmodel)
```

---

plot.Monthmean *Plot of Monthly Mean Estimates*

Description

[Deprecated] Soft-deprecated in favour of autoplot.Monthmean(), which returns a ggplot object you can extend with +. The base R plot below still works.

Usage

```
## S3 method for class 'Monthmean'
plot(x, ...)
```

Arguments

x	a Monthmean object produced by monthmean().
...	additional arguments passed to the plot.

Value

Connected dot plot of estimated monthly means.

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

See Also

autoplot.Monthmean(), monthmean()

Examples

```
mmean <- monthmean(
  data = CVD,
  resp = 'cvd',
  offsetpop = expression(pop / 100000),
  adjmonth = 'average'
)
mmean
# Recommended:
autoplot(mmean)
# Still works, but deprecated:
plot(mmean)
```

`plot.nonlintest` *Plot the Results of the Non-linear Test*

Description

[Deprecated] Soft-deprecated in favour of `autoplot.nonlintest()`, which returns a ggplot object you can extend with +. The base R-style plot below still works.

Usage

```
## S3 method for class 'nonlintest'
plot(x, plot = lifecycle::deprecated(), ...)
```

Arguments

`x`	a `nonlintest` object produced by `nonlintest()`.
`plot`	**[Deprecated]** display plot (TRUE) or return plot (FALSE).
`...`	additional arguments to `plot()`

Value

ggplot Contour plot of the region of the third-order moment outside the test limits generated by `nonlintest()`.

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

See Also

`autoplot.nonlintest()`, `nonlintest()`

Examples

```
test_res <- nonlintest(data = CVD$cvd, n.lag = 4, n.boot = 1000)
# Recommended:
autoplot(test_res)
# Still works, but deprecated:
plot(test_res)
```

`plot.nsCosinor` *Plot the Results of a Non-stationary Cosinor*

Description

[Deprecated] Soft-deprecated in favour of `autoplot.nsCosinor()`, which is the same ggplot — just nudges the recommended idiom so users can `+ theme_bw()` etc.

Usage

```
## S3 method for class 'nsCosinor'
plot(x, ...)
```

Arguments

`x`	a `nsCosinor` object produced by `nscosinor()`.
`...`	further arguments passed to or from other methods.

Details

The code produces the season(s) and trend estimates.

Value

a plot of class `ggplot`.

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

See Also

`autoplot.nsCosinor()`, `nscosinor()`

Examples

```
# model to fit an annual pattern to the monthly cardiovascular disease data
f <- 12
tau <- c(10,50)
## Not run:
  res12 <- nscosinor(
    data = CVD,
    response = 'adj',
    cycles = f,
    niters = 200,
    burnin = 100,
    tau = tau
    )
# Recommended:
autoplot(res12)
# Still works, but deprecated:
plot(res12)

## End(Not run)
```

`plot_circle` *Circular plot of monthly values (ggplot2)*

Description

Circular plot of a monthly variable. This circular plot can be useful for estimates of an annual seasonal pattern. Darker shades of grey correspond to larger numbers. Pie chart where each segment is one month and the fill is shaded to indicate the value (darker = larger). The first value is assumed to be January. This circular plot can be useful for estimates of an annual seasonal pattern. Darker shades of grey correspond to larger numbers.

Usage

```
plot_circle(months, dp = 1)
```

Arguments

`months`	a length-12 numeric vector of monthly values, January first.
`dp`	decimal places for the per-month label, default 1.

Value

a ggplot object.

Author(s)

Nicholas Tierney

See Also

plotCircle() (deprecated base R version)

Examples

```
plot_circle(months = seq(1, 12, 1), dp = 0)
plot_circle(months = c(1, 3, 5, 7, 9, 11, 12, 10, 8, 6, 4, 2)) +
  ggplot2::labs(title = "Example monthly pattern")
```

---

plot_month *Plot a response by month (ggplot2)*

Description

Plots results by month. Assumes the data frame contains variables called year and month. Faceted or coloured line plot of a monthly response over time. Assumes the data frame contains `year` and `month` columns.

Usage

```
plot_month(data, resp, panels = 12)
```

Arguments

`data`	a data frame containing `year`, `month`, and the response.
`resp`	character; name of the response variable.
`panels`	1 (overlay) or 12 (one facet per month). Default 12.

Value

a ggplot object.

See Also

plotMonth() (deprecated wrapper)

Examples

```
plot_month(data = CVD, resp = "cvd", panels = 12)
plot_month(data = CVD, resp = "cvd", panels = 1) +
  ggplot2::theme_minimal()
```

plotCircle *Circular Plot*

Description

Circular plot of a monthly variable. This circular plot can be useful for estimates of an annual seasonal pattern. Darker shades of grey correspond to larger numbers.

Usage

```
plotCircle(months, dp = 1, ...)
```

Arguments

months	monthly variable to plot, the shades of grey of the 12 segments are proportional to this variable. The first result is assumed to be January, the second February, and so on.
dp	decimal places for statistics, default=1.
...	additional arguments to plot()

Details

[Deprecated] Soft-deprecated in favour of plot_circle(), which returns a ggplot object you can extend with +. The base R plot below still works.

Value

a donut-type plot of a monthly variable

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

See Also

plot_circle()

Examples

```
# Recommended:
plot_circle(months = seq(1, 12, 1), dp = 0)
# Still works, but deprecated:
plotCircle(months = seq(1, 12, 1), dp = 0)
```

plotCircular *Circular Plot Using Segments*

Description

A circular plot useful for visualising monthly or weekly data.

Usage

```
plotCircular(
  data = NULL,
  type = "monthly",
  time = NULL,
  areas = NULL,
  main = NULL,
  xlab = NULL,
  ylab = NULL,
  clockwise = TRUE,
  legend = "bottom",
  spoke.col = NULL,
  pieces.col = NULL
)
```

Arguments

data	Dataset to use for plot as a data.frame.
type	type of data to plot, either "monthly" or "weekly", default:monthly.
time	time variable in the data, typically month or day of the week.
areas	variable(s) to plot, the area of the segments (or petals) are proportional to this variable.
main	title for plot, default:blank
xlab	x axis label, default:blank
ylab	y axis label, default:blank
clockwise	plot in a clockwise direction, default:TRUE.
legend	Where to plot legend, only relevant if two variables. Default is "bottom". See ggplot2::guide_legend() for details, specifically "position". Set to "none" for no legend. Options are: "top", "right", "bottom", "left", or "inside".
spoke.col	spoke colour, default:black.
pieces.col	colours for circular pieces, default:"white" for 1st and "grey" for second variable. Note that a list of available colours may be found with colours().

Details

A circular plot can be useful for spotting the shape of the seasonal pattern. This function can be used to plot any circular patterns, e.g., weekly or monthly. The method assumes that monthly data starts in January and weekly data starts on Monday.

The plots are also called rose diagrams, with the segments then called "petals".

Value

a circular plot in ggplot2 format, also known as "rosebud", and "nightingale" plots.

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

References

Fisher, N.I. (1993) *Statistical Analysis of Circular Data*. Cambridge University Press, Cambridge.

Examples

```
## Monthly numbers of AFL players
AFL.frame = data.frame(AFL[c("month", "players", "expected")])
aplot1 = plotCircular(
 data = AFL.frame,
 time = "month",
 areas = "players",
 legend = "none")
## The observed and expected number of players per month
aplot = plotCircular(
data = AFL.frame,
time = "month",
areas = c("players", "expected"))
## Creating weekly data as an example
weekfreq <- data.frame(day = 1:7, counts = rpois(n = 7, lambda = 5))
wplot = plotCircular(
 data = weekfreq,
 time = "day",
 type = "weekly",
 areas = "counts",
 legend = "none")
```

---

plotMonth *Plot Results by Month*

Description

Plots results by month. Assumes the data frame contains variables called year and month.

Usage

```
plotMonth(data, resp, panels = 12, ...)
```

Arguments

data	a data frame.
resp	response variable to plot.
panels	number of panels to use in plot (1 or 12). 12 gives one panel per month, 1 plots all the months in the same panel.
...	further arguments passed to or from other methods.

Details

[Deprecated] Soft-deprecated in favour of plot_month(), which returns a ggplot object you can extend with +. The existing function still works.

Value

Facetted lineplot of response over time, one facet per month.

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

References

Barnett, A.G., Dobson, A.J. (2010) *Analysing Seasonal Health Data*. Springer. doi:10.1007/9783-642107481

See Also

plot_month()

Examples

```
# Recommended:
plot_month(data = CVD, resp = "cvd", panels = 12)
# Still works, but deprecated:
plotMonth(data = CVD, resp = "cvd", panels = 12)
```

print.casecross *Print the Results of a Case-Crossover Model*

Description

The default print method for a `casecross()` object produced by `casecross()`.

Usage

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

Arguments

x	a `casecross()` object produced by `casecross()`.
...	optional arguments to `print()` or `plot()` methods.

Details

Uses `print.coxph()`.

Value

printed output of `casecross()`.

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

See Also

`casecross()`, `summary.casecross()`, `coxph()`

Examples

```
# subset for example
CVDdaily <- subset(CVDdaily, date <= as.Date('1987-12-31'))
# Effect of ozone on CVD death
model1 <- casecross(
  cvd ~ o3mean + tmpd + Mon + Tue + Wed + Thu + Fri + Sat,
  data = CVDdaily
)
model1
```

print.monthglm *Prints basic results from* monthglm()

Description

Prints basic results from monthglm()

Usage

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

Arguments

x	Object of class monthglm
...	further arguments passed to plot(), or from other methods.

Value

basic summary using the glm method for print()..

Examples

```
mmodel <- monthglm(
  formula = cvd~1,
  data = CVD,
  family = poisson(),
  offsetpop = expression(pop/100000),
  offsetmonth = TRUE
  )
mmodel
```

print.Monthmean *Print the Results from Monthmean*

Description

Print the monthly mean estimates from a Monthmean object produced by monthmean().

Usage

```
## S3 method for class 'Monthmean'
print(x, digits = 1, ...)
```

Arguments

x	a Monthmean object produced by monthmean().
digits	minimal number of significant digits, see print.default().
...	additional arguments passed to print().

Value

a table of values of Month and means

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

See Also

monthmean

Examples

```
mmean <- monthmean(
  data = CVD,
  resp = 'cvd',
  offsetpop = expression(pop / 100000),
  adjmonth = 'average'
)
mmean
```

---

print.nonlintest *Print the Results of the Non-linear Test*

Description

The default print method for a nonlintest object produced by nonlintest().

Usage

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

Arguments

x	a nonlintest object produced by nonlintest().
...	additional arguments to plot()

Value

summary of Results of the Non-linear Test.

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

See Also

nonlintest() plot.nonlintest()

Examples

```
## Not run:
test_res <- nonlintest(data = CVD$cvd, n.lag = 4, n.boot = 1000)
test_res

## End(Not run)
```

---

print.nsCosinor *Print the Results of a Non-stationary Cosinor*

Description

The default print method for a nsCosinor object produced by nscosinor().

Usage

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

Arguments

x	a nsCosinor object produced by nscosinor().
...	further arguments passed to or from other methods.

Details

Prints out the model call, number of MCMC samples, sample size and residual summary statistics.

Value

Prints out the model call, number of MCMC samples, sample size and residual summary statistics.

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

See Also

nscosinor() summary.nsCosinor()

Examples

```
# model to fit an annual pattern to the monthly cardiovascular disease data
f <- 12
tau <- c(10,50)
## Not run:
  res12 <- nscosinor(
    data = CVD,
    response = 'adj',
    cycles = f,
    niters = 250,
    burnin = 100,
    tau = tau
    )
res12
summary(res12)

## End(Not run)
```

`print.summary.monthglm` *Printing a summary of a month.glm*

Description

Printing a summary of a month.glm

Usage

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

Arguments

`x`	a `summary.monthglm` object produced by `summary.monthglm()`.
`...`	further arguments passed to or from other methods.

Value

summary of monthglm results. This includes, number of observations, and a table of the mean, lower, upper, z value, and p value for each of the months.

Examples

```
mmodel <- monthglm(
  formula = cvd~1,
  data = CVD,
  family = poisson(),
```

```
  offsetpop = expression(pop/100000),
  offsetmonth = TRUE
  )
summary(mmodel)
```

`print.summary.nsCosinor` *Print a summary of an* `nscosinor()` *object*

Description

Print a summary of an `nscosinor()` object

Usage

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

Arguments

`x`	a `summary.nsCosinor` object produced by `summary.nsCosinor()`.
`...`	further arguments passed to or from other methods.

Value

summary - Statistics for non-stationary cosinor based on MCMC chains.

Examples

```
# model to fit an annual pattern to the monthly cardiovascular disease data
f <- c(12)
tau <- c(10,50)
## Not run:
  res12 <- nscosinor(
    data = CVD,
    response = 'adj',
    cycles = f,
    niters = 200,
    burnin = 100,
    tau = tau
    )
summary(res12)

## End(Not run)
```

schz *Schizophrenia Births in Australia, 1930–1971*

Description

Monthly number of babies born with schizophrenia in Australia from 1930 to 1971. The national number of births and number of cases are missing for January 1960 are missing.

Usage

```
schz
```

Format

A data frame with 504 observations on the following 6 variables:

- year: year of birth
- month: month of birth
- yrmon: a combination of year and month: $year + (month - 1)/12$
- NBirths: monthly number of births in Australia, used as an offset
- SczBroad: monthly number of schizophrenia births using the broad diagnostic criteria
- SOI: southern oscillation index

Source

From Prof John McGrath and colleagues, The University of Queensland, Brisbane.

Examples

```
plot(
  schz$yrmon,
  schz$SczBroad,
  type = 'o',
  xlab = 'Date',
  ylab = 'Number of schizophrenia births'
)
```

seasrescheck *Seasonal Residual Checks*

Description

Tests the residuals for any remaining seasonality.

Usage

```
seasrescheck(res)
```

Arguments

res	residuals from some time series regression model.

Details

Plots: i) histogram of the residuals, ii) a scatter plot against residual order, iii) the autocovariance, iv) the cumulative periodogram (see cpgram())

Value

four plots: 1. The histogram of the residuals; 2. A scatter plot against residual order; 3. The autocovariance; 4. The cumulative periodogram (see cpgram()).

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

Examples

```
# cardiovascular disease data
# (use an offset of the scaled number of days in a month)
model <- cosinor(
  cvd ~ 1,
  date = 'month',
  data = CVD,
  type = 'monthly',
  family = poisson(),
  offsetmonth = TRUE
  )
seasrescheck(resid(model))
```

`sinusoid` *Plot a Sinusoid*

Description

Plots a sinusoid over 0 to 2π.

Usage

```
sinusoid(amplitude, frequency, phase)
```

Arguments

`amplitude`	the amplitude of the sinusoid (its maximum value).
`frequency`	the frequency of the sinusoid in 0 to 2π (number of cycles).
`phase`	the phase of the sinusoid (location of the peak).

Details

Sinusoidal curves are useful for modelling seasonal data. A sinusoid is plotted using the equation: $A\cos(ft - P), t = 0, \ldots, 2\pi$, where A is the amplitude, f is the frequency, t is time and P is the phase.

Value

tibble of sinusoidal wave of given amplitude, frequency, and phase.

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

References

Barnett, A.G., Dobson, A.J. (2010) *Analysing Seasonal Health Data*. Springer. doi:10.1007/9783-642107481

Examples

```
s_n <- sinusoid(
  amplitude = 1,
  frequency = 1,
  phase = 1
  )
s_n
# plot it
autoplot(s_n)
```

stillbirth *Stillbirths in Queensland, 1998–2000*

Description

Monthly number of stillbirths in Australia from 1998 to 2000. It is a rare event; there are 352 stillbirths out of 60,110 births. To preserve confidentiality the day of birth has been randomly re-ordered.

Usage

```
stillbirth
```

Format

A data frame with 60,110 observations on the following 7 variables:

- dob: date of birth (year-month-day)
- year: year of birth
- month: month of birth
- yrmon: a combination of year and month: $year + (month - 1)/12$
- seifa: SEIFA score, an area level measure of socioeconomic status in quintiles
- gestation: gestation in weeks
- stillborn: stillborn (yes/no); 1=Yes, 0=No

Source

From Queensland Health.

Examples

```
table(stillbirth$month, stillbirth$stillborn)
```

summary.casecross *Summary of the Results of a Case-crossover Model*

Description

The default summary method for an object produced by `casecross()`. Shows the number of control days, the average number of control days per case days, and the parameter estimates.

Usage

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

Arguments

object	a casecross() object produced by casecross().
...	further arguments passed to or from other methods.

Value

The number of control days, the average number of control days per case days, and the parameter estimates.

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

See Also

casecross()

Examples

```
# cardiovascular disease data
# subset for example
CVDdaily <- subset(CVDdaily, date <= as.Date('1987-12-31'))
# Effect of ozone on CVD death
model1 <- casecross(
  cvd ~ o3mean + tmpd + Mon + Tue + Wed + Thu + Fri + Sat,
  data = CVDdaily
)
summary(model1)
# match on day of the week
model2 <- casecross(cvd ~ o3mean + tmpd, matchdow = TRUE, data = CVDdaily)
summary(model2)
# match on temperature to within a degree
model3 <- casecross(
  cvd ~ o3mean + Mon + Tue + Wed + Thu + Fri + Sat,
  data = CVDdaily,
  matchconf = 'tmpd',
  confrange = 1
)
summary(model3)
```

summary.Cosinor *Print and summary methods for a Cosinor*

Description

Print and summary methods for a Cosinor object produced by cosinor(). summary() summarises the sinusoidal seasonal pattern and tests whether there is a statistically significant seasonal or circadian pattern (assuming a smooth sinusoidal pattern). The amplitude describes the average height of the sinusoid, and the phase describes the location of the peak. The scale of the amplitude depends on the link function: for logistic regression the amplitude is given on a probability scale; for Poisson regression the amplitude is given on an absolute scale. print() uses the glm method for print() on the underlying model.

Usage

```
## S3 method for class 'Cosinor'
summary(object, digits = 2, ...)

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

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

Arguments

object	a Cosinor object produced by cosinor().
digits	minimal number of significant digits, see print.default().
...	further arguments passed to or from other methods.
x	a Cosinor or summary.Cosinor object.

Value

summary.Cosinor() returns a list with the following named elements:

- n: sample size.
- amp: estimated amplitude.
- amp.scale: the scale of the estimated amplitude (empty for standard regression; "probability scale" for logistic regression; "absolute scale" for Poisson regression).
- phase: estimated peak phase on a time scale.
- lphase: estimated low phase on a time scale (half a year after/before phase).
- significant: statistically significant sinusoid (TRUE/FALSE).
- alpha: statistical significance level.
- digits: minimal number of significant digits.

- text: add explanatory text to the returned phase value (TRUE) or return a number (FALSE).
- type: type of data (yearly/monthly/weekly/hourly).
- ctable: table of regression coefficients.

`print.Cosinor()` and `print.summary.Cosinor()` are called for their side effect of printing to the console and invisibly return `x`.

summary of output from `cosinor()`.

Methods (by generic)

- `print(summary.Cosinor)`: Print a `summary.Cosinor` object: amplitude, phase, statistical significance, and the regression coefficient table.

Functions

- `print(Cosinor)`: Print basic results from `cosinor()` using the `glm` print method.

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

See Also

`cosinor()` `plot.Cosinor()` `invyrfraction()` `glm()`

Examples

```
## cardiovascular disease data (offset based on number of days in...
## ...the month scaled to an average month length)
res <- cosinor(
  cvd ~ 1,
  date = 'month',
  data = CVD,
  type = 'monthly',
  family = poisson(),
  offsetmonth = TRUE
  )
res
summary(res)


## cardiovascular disease data (offset based on number of days in...
## ...the month scaled to an average month length)
res <- cosinor(
  cvd ~ 1,
  date = 'month',
  data = CVD,
  type = 'monthly',
  family = poisson(),
  offsetmonth = TRUE
  )
```

```
res
summary(res)

## hourly indoor temperature data
res <- cosinor(
  bedroom ~ 1,
  date = 'datetime',
  type = 'hourly',
  data = indoor
)
summary(res)
# to get the p-values for the sine and cosine estimates
summary(res$glm)
```

`summary.monthglm` *Summary for a Monthglm*

Description

The default summary method for an object produced by `monthglm()`.

Usage

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

Arguments

`object`	a `monthglm` object produced by `nscosinor()`.
`...`	further arguments passed to or from other methods.

Details

The estimates are the mean, 95\ p-value (comparing each month to the reference month). If Poisson regression was used then the estimates are shown as rate ratios. If logistic regression was used then the estimates are shown as odds ratios.

Value

a list with the following elements:

- n: sample size.
- month.ests: parameter estimates for the intercept and months.
- month.effect: scale of the monthly effects. `RR` for rate ratios, `OR` for odds ratios, or empty otherwise.

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

See Also

monthglm, plot.monthglm

Examples

```
mmodel <- monthglm(
  formula = cvd~1,
  data = CVD,
  family = poisson(),
  offsetpop = expression(pop/100000),
  offsetmonth = TRUE
  )
summary(mmodel)
```

---

summary.nsCosinor *Summary for a Non-stationary Cosinor*

Description

The default summary method for a nsCosinor object produced by nscosinor().

Usage

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

Arguments

object	a nsCosinor object produced by nscosinor().
...	further arguments passed to or from other methods.

Details

The amplitude describes the average height of each seasonal cycle, and the phase describes the location of the peak. The results for the phase are given in radians (0 to 2π), they can be transformed to the time scale using the invyrfraction() making sure to first divide by 2π.

The larger the standard deviation for the seasonal cycles, the greater the non-stationarity. This is because a larger standard deviation means more change over time.

Value

a list with the following elements:

- cycles: vector of cycles in units of time, e.g., for a six and twelve month pattern cycles=c(6,12).
- niters: total number of MCMC samples.
- burnin: number of MCMC samples discarded as a burn-in.

- tau: vector of smoothing parameters, `tau[1]` for trend, `tau[2]` for 1st seasonal parameter, `tau[3]` for 2nd seasonal parameter, etc.
- stats: summary statistics (mean and confidence interval) for the residual standard deviation, the standard deviation for each seasonal cycle, and the amplitude and phase for each cycle.

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

See Also

`nscosinor()` `plot.nsCosinor()`

Examples

```
# model to fit an annual pattern to the monthly cardiovascular disease data
f <- c(12)
tau <- c(10,50)
## Not run:
  res12 <- nscosinor(
    data = CVD,
    response = 'adj',
    cycles = f,
    niters = 5000,
    burnin = 1000,
    tau = tau
    )
summary(res12)
plot(res12)

## End(Not run)
```

---

third *Third-order Moment*

Description

Estimated third order moment for a time series.

Usage

```
third(
  data,
  n.lag,
  centre = TRUE,
  outmax = TRUE,
  plot = lifecycle::deprecated()
)
```

Arguments

data	a vector of equally spaced numeric observations (time series).
n.lag	the number of lags, maximum = length of time series.
centre	centre series by subtracting mean (default=TRUE).
outmax	display the (x,y) lag co-ordinates for the maximum and minimum values (default=TRUE).
plot	**[Deprecated]** Use autoplot.third() on the returned object instead. See examples.

Details

The third-order moment is the extension of the second-order moment (essentially the autocovariance). The equation for the third order moment at lags (j,k) is: $n^{-1} \sum X_t X_{t+j} X_{t+k}$. The third-order moment is useful for testing for non-linearity in a time series, and is used by nonlintest().

Value

an object of class "third" (a list) with the following elements:

- waxis: the axis from -n.lag to n.lag.
- third: the estimated third order moment in the range -n.lag to n.lag, including the symmetries.
- n.lag: the maximum lag.

Pass the result to autoplot() to draw the contour plot.

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

See Also

autoplot.third()

Examples

```
t <- third(CVD$cvd, n.lag = 12)
autoplot(t)
```

`tidy.casecross` *Tidy a casecross object*

Description

This implements the tidy method for `broom::tidy.coxph()`.

Usage

```
## S3 method for class 'casecross'
tidy(x, exponentiate, conf.int, conf.level, ...)
```

Arguments

`x`	A `casecross` object returned from `survival::coxph()`.
`exponentiate`	Logical indicating whether or not to exponentiate the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to FALSE.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
`conf.level`	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`...`	For `tidy()`, additional arguments passed to `summary(x, ...)`. Otherwise ignored.

Examples

```
# subset for example
CVDdaily <- subset(CVDdaily, date <= as.Date('1987-12-31'))
# Effect of ozone on CVD death
model1 <- casecross(
  cvd ~ o3mean + tmpd + Mon + Tue + Wed + Thu + Fri + Sat,
  data = CVDdaily
)
summary(model1)

# works with "broom" style tidiers:
# data frame of estimate, std. error, p-value for each model term
# similar to "summary", but in a dataframe
tidy(model1)
# exponentiate the coefficient estimates
tidy(model1, exponentiate = TRUE)
# include confidence intervals in output
tidy(model1, conf.int = TRUE)
# change confidence interval amount
tidy(model1, conf.int = TRUE)
```

wtest *Walter and Elwood's Test of Seasonality*

Description

Tests for a seasonal pattern in Binomial data. A test of whether monthly data has a sinusoidal seasonal pattern. The test has low power compared with the `cosinor()` test.

Usage

```
wtest(cases, offset, data, alpha = 0.05)
```

Arguments

`cases`	variable name for cases ("successes").
`offset`	variable name for at-risk population ("trials").
`data`	data frame (optional).
`alpha`	significance level (default=0.05).

Value

a list with the following elements:

- test: test statistic.
- pvalue: p-value.

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

References

Walter, S.D., Elwood, J.M. (1975) A test for seasonality of events with a variable population at risk. *British Journal of Preventive and Social Medicine* 29, 18–21.

Barnett, A.G., Dobson, A.J. (2010) *Analysing Seasonal Health Data*. Springer. doi:10.1007/9783-642107481

Examples

```
# tabulate the total number of births and the number of stillbirths
freqs <- table(stillbirth$month, stillbirth$stillborn)
data <- list()
data$trials <- as.numeric(freqs[, 1] + freqs[, 2])
data$success <- as.numeric(freqs[, 2])
# test for a seasonal pattern in stillbirth
test <- wtest(
  cases = 'success',
```

```
  offset = 'trials',
  data = data
)
```

---

yrfraction *Fraction of the Year*

Description

Calculate the fraction of the year for a date variable (after accounting for leap years) or for month.

Usage

```
yrfraction(date, type = c("daily", "weekly", "monthly"))
```

Arguments

date	a date variable if type="daily", or an integer between 1 and 12 if type="monthly".
type	One of "daily" (default) for dates, "monthly" for months, or "weekly" for weeks.

Details

Returns the fraction of the year in the range [0,1).

Value

the fraction of the year.

Author(s)

Adrian Barnett <a.barnett@qut.edu.au>

Examples

```
# create fractions for the start, middle and end of the year
date <- as.Date(c(0, 181, 364), origin = '1991-01-01')
# create fractions based on these dates
yrfraction(date)
yrfraction(1:12, type = 'monthly')
```

Index

∗ **datasets**
  AFL, 4
  CVD, 19
  CVDdaily, 20
  exercise, 21
  indoor, 22
  schz, 50
  stillbirth, 53

aaft, 3
AFL, 4
atan2(), *32*
augment, *5*
augment.casecross, 5
autoplot(), *31*, *60*
autoplot.Cosinor, 5
autoplot.Cosinor(), *33*
autoplot.monthglm, 6
autoplot.monthglm(), *34, 35*
autoplot.Monthmean, 7
autoplot.Monthmean(), *35*
autoplot.nonlintest, 8
autoplot.nonlintest(), *36*
autoplot.nsCosinor, 9
autoplot.nsCosinor(), *37*
autoplot.peri, 10
autoplot.peri(), *31, 32*
autoplot.third, 11
autoplot.third(), *60*

broom::tidy.coxph(), *61*

casecross, 11
casecross(), *44*, *53, 54*
ciPhase, 14
colours(), *41*
cosinor, 15, *33*
cosinor(), *5, 6*, *25*, *33*, *55, 56*, *62*
coxph(), *44*
cpgram(), *51*

createAdj, 18
CVD, 19
CVDdaily, 20

Date(), *13*

exercise, 21

family(), *24*

ggplot2::guide_legend(), *41*
glance, *5*
glance.casecross, 22
glm(), *16*, *56*

identity(), *16*
indoor, 22
invyrfraction, 23
invyrfraction(), *16*, *56*, *58*
invyrfraction_chr *(*invyrfraction*)*, 23
invyrfraction_num *(*invyrfraction*)*, 23

log(), *16*

monthglm, 24
monthglm(), *6*, *34, 35*, *45*, *57*
monthmean, 26
monthmean(), *7*, *35*, *45, 46*

nonlintest, 27
nonlintest(), *8*, *36*, *46, 47*, *60*
nscosinor, 29
nscosinor(), *9*, *16*, *37*, *47*, *49*, *57–59*

peri, 31
peri(), *10*
phasecalc, 32
plot(), *34*, *36*, *40*, *44–46*
plot.Cosinor, 33
plot.Cosinor(), *17*, *56*
plot.monthglm, 34

plot.Monthmean, 35
plot.Monthmean(), *27*
plot.nonlintest, 36
plot.nonlintest(), *28*, *47*
plot.nsCosinor, 37
plot.nsCosinor(), *30*, *59*
plot_circle, 38
plot_circle(), *40*
plot_month, 39
plot_month(), *43*
plotCircle, 40
plotCircle(), *39*
plotCircular, 41
plotMonth, 42
plotMonth(), *39*
POSIXct, *16*
print(), *44–46*, *55*
print.casecross, 44
print.Cosinor *(*summary.Cosinor*)*, 55
print.coxph(), *44*
print.default(), *46*, *55*
print.monthglm, 45
print.Monthmean, 45
print.nonlintest, 46
print.nonlintest(), *28*
print.nsCosinor, 47
print.summary.Cosinor
*(*summary.Cosinor*)*, 55
print.summary.monthglm, 48
print.summary.nsCosinor, 49

schz, 50
seasrescheck, 51
seasrescheck(), *6*, *16*, *33*
sinusoid, 52
stats::gaussian(), *16*
stillbirth, 53
summary.casecross, 53
summary.casecross(), *44*
summary.Cosinor, 55
summary.Cosinor(), *6*, *17*, *33*
summary.monthglm, 57
summary.monthglm(), *48*
summary.nsCosinor, 58
summary.nsCosinor(), *30*, *47*, *49*
survival::coxph(), *13*, *61*

third, 59
third(), *11*

tidy, *5*
tidy.casecross, 61

wtest, 62

yrfraction, 63