# Package: season (via r-universe)

September 15, 2024

**Type** Package

**Title** Seasonal Analysis of Health Data

**Version** 0.3.15

**Author** Adrian Barnett and Peter Baker and Oliver Hughes

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

**Depends** R (>= 3.0.1), ggplot2 (>= 0.9.3), MASS, survival

**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)

**Encoding** UTF-8

**LazyData** true

**Suggests** knitr, rmarkdown, mgcv, dlnm, coda, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**RoxygenNote** 7.1.2

**Config/testthat/edition** 3

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

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

**RemoteRef** HEAD

**RemoteSha** 6a707fccc0bdef472446b758e548e259f48e7b10

## Contents

---

`season-package` *season: Tools for Uncovering and Estimating Seasonal Patterns.*

---

## Description

The package contains graphical methods for displaying seasonal data and regression models for detecting and estimating seasonal patterns.

## Details

The regression models can be applied to normal, Poisson or binomial dependent data distributions. Tools are available for both time series data (equally spaced in time) and survey data (unequally spaced in time).

Sinusoidal (parametric) seasonal patterns are available (`cosinor`, `nscosinor`), as well as models for monthly data (`monthglm`), and the case-crossover method to control for seasonality (`casecross`).

`season` aims to fill an important gap in the software by providing a range of tools for analysing seasonal data. The examples are based on health data, but the functions are equally applicable to any data with a seasonal pattern.

## Author(s)

Adrian Barnett <a.barnett@qut.edu.au>
Peter Baker
Oliver Hughes

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

## References

Barnett, A.G., Dobson, A.J. (2010) *Analysing Seasonal Health Data*. Springer.

---

`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

`surrogates` 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

### Examples

```
data(CVD)
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

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

---

`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 = FALSE,
  matchconf = "",
  confrange = 0,
  stratamonth = FALSE
)
```

## 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. |
| `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`. |

## 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 `coxph` and so the parameter estimates are odds ratios.)

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

## Value

| | |
|---|---|
| `call` | the original call to the casecross function. |
| `c.model` | conditional logistic regression model of class `coxph`. |
| `ncases` | total number of cases. |
| `ncasedays` | number of case days with at least one control day. |
| `ncontroldayss` | 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.

Barnett, A.G., Dobson, A.J. (2010) *Analysing Seasonal Health Data*. Springer.

## See Also

`summary.casecross`, `coxph`

## Examples

```
# cardiovascular disease data
data(CVDdaily)
CVDdaily = subset(CVDdaily, date<=as.Date('1987-12-31')) # subset for example
# 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% confidence interval). |

### Details

The estimates of the phase are rotated to have a centre of $\pi$, 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

| | |
|---|---|
| `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.

### Examples

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

`cipolygon` *Function to Draw CI Polygon*

## Description

Internal function to draw a confidence interval for multiple times as a grey area. For internal use only.

## Usage

```
cipolygon(time, lower, upper)
```

## Arguments

| | |
|---|---|
| `time` | x-axis. |
| `lower` | lower limit of the confidence level. |
| `upper` | upper limit of the confidence level. |

## Author(s)

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

`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 = gaussian(),
  alpha = 0.05,
  cycles = 1,
  rescheck = FALSE,
  type = "daily",
  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: identity, log, logit, cloglog. Note, it must have the parentheses. |
| `alpha` | significance level, set to 0.05 (default). |
| `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 (type="monthly") (with `family=poisson()`). |
| `offsetpop` | include an offset for the population (optional), this should be a variable in the data frame. Do not log-transform this offset, as the transform is applied by the code. |
| `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.

### 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)
data(CVD)
res = cosinor(cvd~1, date='month', data=CVD, type='monthly',
              family=poisson(), offsetmonth=TRUE)
summary(res)
seasrescheck(res$residuals) # check the residuals
## stillbirth data
data(stillbirth)
res = cosinor(stillborn~1, date='dob', data=stillbirth,
              family=binomial(link='cloglog'))
summary(res)
plot(res)
## hourly indoor temperature data
res = cosinor(bedroom~1, date='datetime', type='hourly', data=indoor)
summary(res)
# to get the p-values for the sine and cosine estimates
summary(res$glm)
```

---

## createAdj — *Creates an Adjacency Matrix*

---

### Description

Creates an adjacency matrix in a form suitable for using in BRugs or WinBUGS.

### Usage

```
createAdj(matrix, filename = "Adj.txt", suffix = NULL)
```

### Arguments

| | |
|---|---|
| matrix | square matrix with 1's for neighbours and NA's for non-neighbours. |
| filename | filename that the adjacency matrix file will be written to (default='Adj.txt'). |
| 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

Creates a text file named `filename` that contains the total number of neighbours (num), the index number of the adjacent neighbours (adj) and the 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))
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 (of course) 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

```
data(CVD)
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

```
data(CVDdaily)
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

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

---

# flagleap *Count the Number of Days in the Month*

---

## Description

Counts the number of days per month given a range of dates. Used to adjust monthly count data for the at-risk time period. For internal use only.

## Usage

```
flagleap(data, report = TRUE, matchin = FALSE)
```

## Arguments

| | |
|---|---|
| `data` | data. |
| `report` | produce a brief report on the range of time used (default=TRUE). |
| `matchin` | expand the result to match the start and end dates, otherwise only dates in the data will be returned (default=FALSE). |

## Details

The data should contain the numeric variable called 'year' as a 4 digit year (e.g., 1973).

## Value

| | |
|---|---|
| `year` | year (4 digits). |
| `month` | month (2 digits). |
| `ndaysmonth` | number of days in the month (either 28, 29, 30 or 31). |

## Author(s)

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

---

`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

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

## invyrfraction — *Inverse Fraction of the Year or Hour*

### Description

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

### Usage

```
invyrfraction(frac, type = "daily", text = TRUE)
```

### Arguments

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

### Details

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).

### Value

| | |
|---|---|
| `daym` | date (day and month for `daily`) or fractional month (for `monthly`) or fractional 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='daily')
invyrfraction(c(0, 0.5, 0.99), type='monthly')
invyrfraction(c(0, 0.5, 0.99), type='hourly')
```

## kalfil *Forward and Backward Sweep of the Kalman Filter*

### Description

Internal function to do a forward and backward sweep of the Kalman filter, used by `nscosinor`. For internal use only.

### Usage

```
kalfil(data, f, vartheta, w, tau, lambda, cmean)
```

### Arguments

| | |
|---|---|
| `data` | a data frame. |
| `f` | vector of cycles in units of time. |
| `vartheta` | variance for noise. |
| `w` | variance of seasonal component. |
| `tau` | controls flexibility of trend and season. |
| `lambda` | distance between observations. |
| `cmean` | used to give a vague prior for the starting values. |

### Author(s)

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

## monthglm *Fit a GLM with Month*

### Description

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

### Usage

```
monthglm(
  formula,
  data,
  family = 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. |
| `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. |

## 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

| | |
|---|---|
| `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.

## See Also

`summary.monthglm`, `plot.monthglm`

### Examples

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

---

## monthmean *Monthly Means*

---

### Description

Calculate the monthly mean or adjusted monthly mean for count data.

### Usage

```
monthmean(data, resp, offsetpop = NULL, adjmonth = FALSE)
```

### Arguments

| | |
|---|---|
| data | data set as a data frame. |
| resp | response variable in the data set 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=FALSE). |

### Details

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).

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.

## See Also

`plot.Monthmean`

## Examples

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

---

`nochars` *Remove Letters and Characters from a String*

---

## Description

Remove letters and characters from a string to leave only numbers. Removes all letters (upper and lower case) and the characters ".", "(" and ")". For internal use only.

## Arguments

| | |
|---|---|
| `text` | text string. |

## Author(s)

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

---

`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 following 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

## See Also

`print.nonlintest`, `plot.nonlintest`

## Examples

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

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
)
```

## Arguments

| | |
|---|---|
| `data` | a data frame. |
| `response` | 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. |

## 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 $\omega_t$ 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% confidence interval. |
| `season` | mean season(s) and 95% confidence interval(s). |
| `oseason` | overall season(s) and 95% confidence interval(s). This will be the same as `season` if there is only one seasonal cycle. |
| `fitted` | fitted values and 95% confidence interval, based on trend + 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.

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

```
data(CVD)
# 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)
```

---

`nscosinor.initial` *Initial Values for Non-stationary Cosinor*

---

## Description

Creates initial values for the non-stationary cosinor decomposition `nscosinor`. For internal use only.

## Arguments

| | |
|---|---|
| `data` | a data frame. |
| `response` | response variable. |
| `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). |
| `n.season` | number of seasons. |

## Author(s)

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

peri *Periodogram*

## Description

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

## Usage

```
peri(data, adjmean = TRUE, plot = TRUE)
```

## Arguments

| | |
|---|---|
| `data` | a data frame. |
| `adjmean` | subtract the mean from the series before calculating the periodogram (default=TRUE). |
| `plot` | plot the estimated periodogram (default=TRUE). |

## Value

| | |
|---|---|
| `peri` | periodogram, I($\omega$). |
| `f` | frequencies in radians, $\omega$. |
| `c` | frequencies in cycles of time, $2\pi/\omega$. |
| `amp` | amplitude periodogram. |
| `phase` | phase periodogram. |

## Author(s)

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

## Examples

```
data(CVD)
p = peri(CVD$cvd)
```

`phasecalc` *Phase from Cosinor Estimates*

## Description

Calculate the phase given the estimated sine and cosine values from a cosinor model.

## Usage

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

## Arguments

| | |
|---|---|
| `cosine` | estimated cosine value from a cosinor model. |
| `sine` | estimated sine value from a cosinor model. |

## Details

Returns the phase in radians, in the range $[0, 2\pi)$. The phase is the peak in the sinusoid.

## Value

| | |
|---|---|
| `phaser` | 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

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

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

### Description

Plots the fitted sinusoid from a `Cosinor` object produced by `cosinor`.

### 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.

### Author(s)

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

### See Also

`cosinor`, `summary.Cosinor`, `seasrescheck`

## plot.monthglm *Plot of Monthly Estimates*

### Description

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

### Usage

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

### 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). |
| ... | additional arguments passed to the plot. |

### Author(s)

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

### See Also

monthglm

### Examples

```
data(CVD)
mmodel = monthglm(formula=cvd~1, data=CVD, family=poisson(),
                  offsetpop=expression(pop/100000), offsetmonth=TRUE, refmonth=6)
plot(mmodel)
```

---

## plot.Monthmean *Plot of Monthly Mean Estimates*

---

### Description

Plots estimated monthly means.

### Usage

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

### Arguments

| | |
|---|---|
| x | a Monthmean object produced by monthmean. |
| ... | additional arguments passed to the plot. |

### Author(s)

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

### See Also

monthmean

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

### Description

Creates a contour plot of the region of the third-order moment outside the test limits generated by `nonlintest`.

### Usage

```
## S3 method for class 'nonlintest'
plot(x, plot = TRUE, ...)
```

### Arguments

| | |
|---|---|
| `x` | a `nonlintest` object produced by `nonlintest`. |
| `plot` | display plot (TRUE) or return plot (FALSE) |
| `...` | additional arguments to `plot` |

### Author(s)

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

### See Also

`nonlintest`

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

### Description

Plots the trend and season(s) from a `nsCosinor` object produced by `nscosinor`.

### 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

| | |
|---|---|
| `gplot` | A plot of class `ggplot` |

### Author(s)

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

### See Also

`nscosinor`

---

## `plotCircle` *Circular Plot*

---

### Description

Circular plot of a monthly variable.

### 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

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

### Author(s)

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

### Examples

```
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(
  area1,
  area2 = NULL,
  spokes = NULL,
  scale = 0.8,
  labels,
  stats = TRUE,
  dp = 1,
  clockwise = TRUE,
  spoke.col = "black",
  lines = FALSE,
  centrecirc = 0.03,
  main = "",
  xlab = "",
  ylab = "",
  pieces.col = c("white", "gray"),
  length = FALSE,
  legend = TRUE,
  auto.legend = list(x = "bottomright", fill = NULL, labels = NULL, title = ""),
  ...
)
```

## Arguments

| | |
|---|---|
| area1 | variable to plot, the area of the segments (or petals) are proportional to this variable. |
| area2 | 2nd variable to plot (optional), the area of the segments are plotted in grey. |
| spokes | spokes that overlay segments, for example standard errors (optional). |
| scale | scale the overall size of the segments (default:0.8). |
| labels | optional labels to appear at the ends of the segments (there should be as many labels as there are area1). |
| stats | put area values at the ends of the segments, default:TRUE. |
| dp | decimal places for statistics, default:1. |
| clockwise | plot in a clockwise direction, default:TRUE. |
| spoke.col | spoke colour, default:black. |

| | |
|---|---|
| `lines` | add dotted lines to separate petals, default:FALSE. |
| `centrecirc` | controls the size of the circle at the centre of the plot, default:0.03. |
| `main` | title for plot, default:blank |
| `xlab` | x axis label, default:blank |
| `ylab` | y axis label, default:blank |
| `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 'colors()' |
| `length` | make the length of the segments proportional to the dependent variable, default:FALSE |
| `legend` | whether to include legend or not, default:TRUE when plotting two variables |
| `auto.legend` | list of parameters for legend, see `legend` |
| `...` | additional arguments to `plot` and/or `legend`. See `par` for more details |

## 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 number of segments will be the length of the variable `area1`.

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

## Author(s)

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

## References

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

## Examples

```
# months (dummy data)
plotCircular(area1=seq(1,12,1), scale=0.7, labels=month.abb, dp=0)
# weeks (random data)
daysoftheweek = c('Monday','Tuesday','Wednesday','Thursday','Friday',
'Saturday','Sunday')
weekfreq = table(round(runif(100, min=1, max=7)))
plotCircular(area1=weekfreq, labels=daysoftheweek, dp=0)
# Observed number of AFL players with expected values
data(AFL)
plotCircular(area1=AFL$players, area2=AFL$expected, scale=0.72,
  labels=month.abb, dp=0, lines=TRUE, legend=FALSE)
plotCircular(area1=AFL$players, area2=AFL$expected, scale=0.72,
  labels=month.abb, dp=0, lines=TRUE, pieces.col=c("green","red"),
  auto.legend=list(labels=c("Obs","Exp"), title="# players"),
  main="Observed and Expected AFL players")
```

## plotMonth *Plot Results by Month*

### Description

Plots results by 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

Assumes the data frame contains variables called year and month.

### Author(s)

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

### References

Barnett, A.G., Dobson, A.J. (2010) *Analysing Seasonal Health Data*. Springer.

### Examples

```
data(CVD)
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`.

### Author(s)

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

### See Also

`casecross`, `summary.casecross`, `coxph`

## print.Cosinor — *Print the Results of a Cosinor*

### Description

The default print method for a `Cosinor` object produced by `cosinor`.

### Usage

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

### Arguments

| | |
|---|---|
| `x` | a `Cosinor` object produced by `cosinor`. |
| `...` | optional arguments to `print` or `plot` methods. |

### Details

Uses `print.glm`.

### Author(s)

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

### See Also

`cosinor`, `summary.Cosinor`, `glm`

---

## print.monthglm — *Print* monthglm

### Description

Print `monthglm`

### Usage

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

### Arguments

| | |
|---|---|
| x | Object of class `monthglm` |
| ... | further arguments passed to or from other methods. |

---

## print.Monthmean — *Print the Results from Monthmean*

### Description

Print the monthly means 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 the print. |

### Details

The code prints the monthly mean estimates.

### Author(s)

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

### See Also

monthmean

---

## 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 |

### Author(s)

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

### See Also

nonlintest, plot.nonlintest

## 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.

### Author(s)

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

### See Also

`nscosinor`, `summary.nsCosinor`

## print.summary.Cosinor *printing a summary of a Cosinor*

### Description

printing a summary of a Cosinor

### Usage

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

### Arguments

| | |
|---|---|
| `x` | a `summary.Cosinor` object produced by `summary.Cosinor` |
| `...` | optional arguments to `print` or `plot` methods. |

`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. |

`print.summary.nscosinor` *printing a summary of an nscosinor*

## Description

printing a summary of an nscosinor

## 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. |

## `rinvgamma` *Random Inverse Gamma Distribution*

### Description

Internal function to simulate a value from an inverse Gamma distribution, used by `nscosinor`. See the MCMCpack library. For internal use only.

### Arguments

| | |
|---|---|
| `n` | number of observations. |
| `shape` | Gamma shape parameter. |
| `scale` | Gamma scale parameter (default=1). |

## `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

```
data(schz)
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)

### 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)
data(CVD)
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 $2\pi$.

### Usage

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

### Arguments

| | |
|---|---|
| `amplitude` | the amplitude of the sinsuoid (its maximum value). |
| `frequency` | the frequency of the sinusoid in 0 to $2\pi$ (number of cycles). |
| `phase` | the phase of the sinusoid (location of the peak). |
| `...` | additional arguments passed to the plot. |

### Details

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

### Author(s)

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

### References

Barnett, A.G., Dobson, A.J. (2010) *Analysing Seasonal Health Data*. Springer.

### Examples

```
sinusoid(amplitude=1, frequency=1, phase=1)
```

---

## 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

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

---

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

---

### Description

The default summary method for a `casecross` object produced by `casecross`.

### Usage

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

### Arguments

| | |
|---|---|
| `object` | a `casecross` object produced by `casecross`. |
| `...` | further arguments passed to or from other methods. |

### Details

Shows 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`

`summary.Cosinor` *Summary for a Cosinor*

## Description

The default print method for a `Cosinor` object produced by `cosinor`.

## Usage

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

## 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. |

## Details

Summarises the sinusoidal seasonal pattern and tests whether there is 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.

## Value

| | |
|---|---|
| `n` | sample size. |
| `amp` | estimated amplitude. |
| `amp.scale` | the scale of the estimated amplitude (empty for standard regression; 'probability scale' for logistic regession; '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. |

## Author(s)

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

## See Also

cosinor, plot.Cosinor, invyrfraction

---

summary.monthglm *Summary for a Monthglm*

---

## Description

The default summary method for a monthglm 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% confidence interval, Z-value and associated 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

| | |
|---|---|
| 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. |
| x | object of class monthglm |
| object | object of class monthglm |
| list() | further arguments passed to or from other methods. |

## Author(s)

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

## See Also

monthglm, plot.monthglm

## 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 $2\pi$), they can be transformed to the time scale using the `invyrfraction` making sure to first divide by $2\pi$.

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

| | |
|---|---|
| `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`

---

`third` *Third-order Moment*

---

## Description

Estimated third order moment for a time series.

## Usage

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

## 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` | contour plot of the third order moment (default=TRUE). |

## 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

| | |
|---|---|
| `waxis` | The axis –`n.lag` to `n.lag`. |
| `third` | The estimated third order moment in the range –n.lag to n.lag, including the symmetries. |

## Author(s)

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

## Examples

```
data(CVD)
third(CVD$cvd, n.lag=12)
```

## wtest — *Walter and Elwood's Test of Seasonality*

### Description

Tests for a seasonal pattern in Binomial data.

### 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). |

### Details

A test of whether monthly data has a sinusoidal seasonal pattern. The test has low power compared with the `cosinor` test.

### Value

| | |
|---|---|
| `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.

### Examples

```
data(stillbirth)
# 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 = "daily")
```

### Arguments

| | |
|---|---|
| date | a date variable if type='daily', or an integer between 1 and 12 if type='monthly'. |
| type | 'daily' for dates, or 'monthly' for months. |

### Details

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

### Value

| | |
|---|---|
| yrfrac | 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