Package ‘Wats’

December 5, 2016

Title Wrap Around Time Series Graphics
Description Wrap-around Time Series (WATS) plots for interrupted time series designs with seasonal patterns.
Version 0.10.3
Date 2015-11-11
Author Will Beasley [aut, cre], Joe Rodgers [aut], Matthew Schuelke [ctb], Ronnie Coleman [ctb], Mark Joseph Lachowicz [ctb]
Maintainer Will Beasley <wibeasley@hotmail.com>
URL https://github.com/OuhsCbbmc/Wats
BugReports https://github.com/OuhsCbbmc/Wats/issues
Depends R (>= 3.0.0), stats
Imports colorspace, ggplot2, grid, lubridate, plyr, RColorBrewer, testit, zoo
Suggests BayesSingleSub, boot, devtools, knitr, scales, testthat
License MIT + file LICENSE
LazyData TRUE
VignetteBuilder knitr
RoxygenNote 5.0.0
NeedsCompilation no
Repository CRAN
Date/Publication 2016-12-05 18:28:47

R topics documented:

Wats-package .......................................................... 2
AnnotateData .......................................................... 3
AugmentCycleData ....................................................... 4
CartesianPeriodic ....................................................... 5
CartesianRolling ......................................................... 6
Description

Wrap-around Time Series (WATS) Plots for Interrupted Time Series Designs with Seasonal Patterns

Note

The release version is available through CRAN by running `install.packages('wats')`. The most recent development version is available through GitHub by running `devtools::install_github(repo = 'OuhsBbmc/Wats')` (make sure `devtools` is already installed). If you’re having trouble with the package, please install the development version. If this doesn’t solve your problem, please create a new issue, or email Will.

Author(s)

William Howard Beasley – Assistant Professor of Research, University of Oklahoma Health Sciences Center, Dept of Pediatrics, Biomedical and Behavioral Methodology Core (BBMC)

Joseph Lee Rodgers – Director, Quantitative Methods, Department of Psychology and Human Development, Peabody College, 230 Appleton Pl #552, Hobbs 202C, Vanderbilt University, Nashville, TN 37203.

Matthew Schuelke – Senior Statistician, Grants Development Office Saint Louis University

References


Examples

```R
## Not run:
# Install/update Wats with the release version from CRAN.
install.packages('wats')

# Install/update Wats with the development version from GitHub
#install.packages('devtools') #Uncomment if 'devtools' isn't installed already.
devtools::install_github('OuhsBbmc/Wats')
```
Description
Finds midpoints and bands for the within and between cycles. This is the second of two functions that needs to be called to produce WATS Plots. `AugmentZZZ` is the first.

Usage
AnnotateData(dsLinear, dvName, centerFunction, spreadFunction, 
cycleTallyName = "CycleTally", stageIDName = "StageID", 
stageProgressName = "StageProgress", 
proportionThroughCycleName = "ProportionThroughCycle", 
proportionIDName = "ProportionID", 
terminalPointInCycleName = "TerminalPointInCycle")

Arguments
- **dsLinear**  The data.frame to containing the detailed data.
- **dvName**  The name of the dependent/criterion variable.
- **centerFunction**  A function to calculate the center of a subsample.
- **spreadFunction**  A function to calculate the bands of a subsample.
- **cycleTallyName**  The variable name indicating how many cycles have been completed.
- **stageIDName**  The variable name indicating the stage. In a typical interrupted time series, these values are 1 before the interruption and 2 after.
- **stageProgressName**  The variable name indicating the stage in a decimal form. This is mostly for internal uses.
- **proportionThroughCycleName**  The variable name indicating how far the point is through a cycle. For example, 0 degrees would be 0, 180 degrees would be 0.5, 359 degrees would be 0.9972, and 360 degrees would be 0.
- **proportionIDName**  The variable name indicating the ordinal position through a cycle.
- **terminalPointInCycleName**  The variable name indicating the last point within a given cycle.

Value
Returns a data.frame with additional variables «Say what they are». 
AugmentCycleData

Examples

```r
library(Wats)
dsLinear <- CountyMonthBirthRate2005Version
dSLinear <- dsLinear[dsLinear$CountyName=="oklahoma", ]
dSLinear <- AugmentYearDataWithMonthResolution(dsLinear=dsLinear, dateName="Date")

hSpread <- function( scores ) { return( quantile(x=scores, probs=c(.25, .75)) ) } 
portfolio <- AnnotateData(
  dsLinear = dsLinear, 
  dvName = "BirthRate",
  centerFunction = median, 
  spreadFunction = hSpread 
)

head(portfolio$dsStageCycle)
head(portfolio$dsLinear)
head(portfolio$dsPeriodic)
```

AugmentCycleData  Calculates variables necessary for WATS Plots

Description

Calculates variables necessary for WATS Plots. This the first of two functions that needs to be called to produce WATS Plots. AnnotateData is the second.

Usage

`AugmentYearDataWithMonthResolution( dsLinear, dateName )`
`AugmentYearDataWithSecondResolution( dsLinear, dateName )`

Arguments

dSLinear The data.frame to containing the detailed data.
dateName The variable name in dsLinear containing the date or datetime value.

Value

Returns a data.frame with additional variables: CycleTally, ProportionThroughCycle, ProportionID, and TerminalPointInCycle.

Examples

```r
library(Wats)
dSLinear <- CountyMonthBirthRate2005Version
dSLinear <- dsLinear[dsLinear$CountyName=="oklahoma", ]
dSLinear <- AugmentYearDataWithMonthResolution(dsLinear=dsLinear, dateName="Date")
head(dsSLinear)
```
**CartesianPeriodic**  

**Linear Plot with Periodic Elements**

**Description**

Shows the interrupted time series in Cartesian coordinates and its a periodic/cyclic components.

**Usage**

\[
\text{CartesianPeriodic}(\text{dsLinear, dsPeriodic, xName, yName, stageIDName, periodicLowerName = "PositionLower", periodicUpperName = "PositionUpper", paletteDark = NULL, paletteLight = NULL, changePoints = NULL, changePointLabels = NULL, drawPeriodicBand = TRUE, jaggedPointSize = 2, jaggedLineSize = 0.5, bandAlphaDark = 0.4, bandAlphaLight = 0.15, changeLineAlpha = 0.5, changeLineSize = 3, title = NULL, xTitle = NULL, yTitle = NULL)}
\]

**Arguments**

- **dsLinear**  
  The data.frame to containing the simple linear data. There should be one record per observation.

- **dsPeriodic**  
  The data.frame to containing the reoccurring/periodic bands. There should be one record per observation per stage. If there are three stages, this data.frame should have three times as many rows as dsLinear.

- **xName**  
  The variable name containing the date.

- **yName**  
  The variable name containing the dependent/criterion variable.

- **stageIDName**  
  The variable name indicating which stage the record belongs to. For example, before the first interruption, the StageID is 1, and is 2 afterwards.

- **periodicLowerName**  
  The variable name showing the lower bound of a stage’s periodic estimate.

- **periodicUpperName**  
  The variable name showing the upper bound of a stage’s periodic estimate.

- **paletteDark**  
  A vector of colors used for the dark/heavy graphical elements. The vector should have one color for each StageID value. If no vector is specified, a default will be chosen, based on the number of stages.

- **paletteLight**  
  A vector of colors used for the light graphical elements. The vector should have one color for each StageID value. If no vector is specified, a default will be chosen, based on the number of stages.

- **changePoints**  
  A vector of values indicate the interruptions between stages. It typically works best as a Date or a POSIXct class.

- **changePointLabels**  
  The text plotted above each interruption.

- **drawPeriodicBand**  
  A boolean value indicating if the bands should be plotted (whose values are take from the periodicLowerName and periodicUpperName).
jaggedPointSize
The size of the observed data points.
jaggedLineSize
The size of the line connecting the observed data points.
bandAlphaDark
The amount of transparency of the band appropriate for a stage’s x values.
bandAlphaLight
The amount of transparency of the band comparison stages for a given x value.
changeLineAlpha
The amount of transparency marking each interruption.
changeLineSize
The width of a line marking an interruption.
title
The string describing the plot.
xTitle
The string describing the x-axis.
yTitle
The string describing the y-axis.

Value
Returns a ggplot2 graphing object

Examples

```r
library(Hwats) # Load the package
c changemonth <- base::as.Date("1996-02-15")
d slinear <- CountyMonthBirthRate2005Version
d slinear <- d slinear[CountyName="oklahoma", ]
d slinear <- AugmentYearDataWithMonthResolution(d slinear=d slinear, dateName="Date")
hSpread <- function( scores ) { return( quantile(x=scores, probs=c(.25, .75)) ) }
portfolio <- AnnotateData(  
  d slinear, 
  dvName = "BirthRate", 
  centerFunction = median, 
  spreadFunction = hSpread 
)

CartesianPeriodic(  
  portfolio$dslinear,  
  portfolio$dsperiodic,  
  xName = "Date",  
  yName = "BirthRate",  
  stageIDName = "StageID",  
  changePoints = changemonth,  
  changePointLabels = "Bombing Effect"  
)
```

---

**CartesianRolling**

**Linear Plot with Rolling Summaries**

---

**Description**

Shows the interrupted time series in Cartesian coordinates without a periodic/cyclic components.
Usage

CartesianRolling(dsLinear, xName, yName, stageIDName, 
rollingLowerName = "RollingLower", rollingCenterName = "RollingCenter", 
rollingUpperName = "RollingUpper", paletteDark = NULL, 
paletteLight = NULL, colorSparse = grDevices::adjustcolor("tan1", 0.5), 
changePoints = NULL, changePointLabels = NULL, drawJaggedLine = TRUE, 
drawRollingLine = TRUE, drawRollingBand = TRUE, 
drawSparseLineAndPoints = TRUE, jaggedPointSize = 2, 
jaggedLineSize = 0.5, rollingLineSize = 1, sparsePointSize = 4, 
sparseLineSize = 0.5, bandAlpha = 0.4, changeLineAlpha = 0.5, 
changeLineSize = 3, title = NULL, xTitle = NULL, yTitle = NULL)

Arguments

dSLinear The data.frame to containing the data.
xName The variable name containing the date.
yName The variable name containing the dependent/criterion variable.
stageIDName The variable name indicating which stage the record belongs to. For example, 
before the first interruption, the StageID is 1, and is 2 afterwards.
rollingLowerName The variable name showing the lower bound of the rolling estimate.
rollingCenterName The variable name showing the rolling estimate.
rollingUpperName The variable name showing the upper bound of the rolling estimate.
paletteDark A vector of colors used for the dark/heavy graphical elements. The vector should 
have one color for each StageID value. If no vector is specified, a default will 
be chosen, based on the number of stages.
paletteLight A vector of colors used for the light graphical elements. The vector should have 
one color for each StageID value. If no vector is specified, a default will be 
chosen, based on the number of stages.
colorSparse The color of the 'slowest' trend line, which plots only one value per cycle.
changePoints A vector of values indicate the interruptions between stages. It typically works 
best as a Date or a POSIXct class.
changePointLabels The text plotted above each interruption.
drawJaggedLine A boolean value indicating if a line should be plotted that connects the observed 
data points.
drawRollingLine A boolean value indicating if a line should be plotted that connects the rolling 
estimates specified by rollingCenterName.
drawRollingBand A boolean value indicating if a band should be plotted that envelopes the rolling 
estimates (whose values are take from the rollingLowerName and rollingUpperName.
CartesianRolling

drawSparseLineAndPoints
   A boolean value indicating if the sparse line and points should be plotted.

jaggedPointSize
   The size of the observed data points.

jaggedLineSize
   The size of the line connecting the observed data points.

rollingLineSize
   The size of the line connecting the rolling estimates.

sparsePointSize
   The size of the sparse estimates.

sparseLineSize
   The size of the line connecting the sparse estimates.

bandAlpha
   The amount of transparency of the rolling estimate band.

changeLineAlpha
   The amount of transparency marking each interruption.

changeLineSize
   The width of a line marking an interruption.

title
   The string describing the plot.

xTitle
   The string describing the x-axis.

yTitle
   The string describing the y-axis.

Value

Returns a ggplot2 graphing object

Examples

library(Wats) #Load the package
changeMonth <- base::as.Date("1996-02-15")
dsLinear <- CountyMonthBirthRate2005Version
dsLinear <- dsLinear[dsLinear$CountyName=="oklahoma", ]
dsLinear <- AugmentYearDataWithMonthResolution(dsLinear=dsLinear, dateName="Date")
hSpread <- function( scores ) { return( quantile(x=scores, probs=c(.25, .75)) ) }
portfolio <- AnnotateData(
   dsLinear,
   dvName = "BirthRate",
   centerFunction = median,
   spreadFunction = hSpread
)

CartesianRolling(
   portfolio$dsLinear,
   xName = "Date",
   yName = "BirthRate",
   stageIDName = "StageID",
   changePoints = changeMonth,
   changePointLabels = "Bombing Effect"
)
Monthly Growth Fertility Rates (GFR) for 12 urban Oklahoma counties

**Description**

Monthly Growth Fertility Rates (GFR) for 12 urban counties in Oklahoma between January 1990 and December 1999. The GFR is defined as the number of births divided by the number of females (ages 15-44), multiplied by 1,000.

There are two datasets in this package that are almost identical. The 2014 version is better suited for substantive researchers in the areas of fertility and traumatic cultural events. The 2005 version recreates the 2005 article and, therefore is better suited for the graphical aims of the 2014 manuscript.

The difference is that the 2005 version uses constant estimate for a county population –specifically the US Census 1990 estimates. The 2014 version uses different estimates for each month –specifically the US intercensal annual estimates, with linear interpolation for February through December of each year.

**Format**

A data frame with 1,440 observations on the following 11 variables.

- **Fips** The county’s 5-digit value according to the Federal Information Processing Standards. integer
- **CountyName** The lower case name of the county. character
- **Year** The year of the record, ranging from 1990 to 1999. integer
- **Month** The month of the record, ranging from 1 to 12. integer
- **FecundPopulation** The number of females in the county, ages of 15 to 44. numeric
- **BirthCount** The number of births in a county for the given month. integer
- **Date** The year and month of the record, with a date of the 15th. Centering the date within the month makes the value a little more representative and the graphs a little easier. date
- **DaysInMonth** The number of days in the specific month. integer
- **DaysInYear** The number of days in the specific years integer
- **StageID** The ‘Stage’ of the month. The pre-bombing records are ‘1’ (accounting for 9 months of gestation); the post-bombing months are ‘2’. integer
- **BirthRate** The Growth Fertility Rate (GFR). numeric

**Details**

«Joe, can you please finish/edit this sentence?» The monthly birth counts were copied from county records by Ronnie Coleman during the summer of 2001 from state vital statistics records. It was collected for Rodgers, St. John, & Coleman (2005).

The US Census’ intercensal estimates are used for the January values of FecundPopulation. Values for February-December are interpolated using approx.

The datasets were manipulated to produce this data frame by the two R files IsolateCensusPops-ForGfr.R and CalculateGfr.R.
Author(s)
Will Beasley

References

Intercensal estimates for 199x.
Intercensal estimates for 200x.

Examples

```r
library(ggplot2)

# 2005 Version (see description above)
ds2005 <- CountyMonthBirthRate2005Version
ggplot(ds2005, aes(x=Date, y=BirthRate, color=factor(Fips))) +
geom_line() +
labs(title="County Fertility - Longitudinal")

# 2014 Version (see description above)
ds2014 <- CountyMonthBirthRate2014Version
ggplot(ds2014, aes(x=Date, y=BirthRate, color=factor(Fips))) +
geom_line() +
labs(title="County Fertility - Longitudinal")
```

Description
Three operations are performed. First, within each stage, the first row is repeated at the end, to close the loop. Second, multiple points are interpolated (still in a Cartesian coordinates) so that the polar graph doesn’t have sharp edges. These sharp edges would be artifacts of the conversion, and not reflect the observed data. Third, the Cartesian points are converted to polar coordinates.
PolarizeCartesian

Usage

PolarizeCartesian(dsLinear, dsStageCycle, yName, stageIDName, 
cycleTallyName = "CycleTally", 
proportionThroughCycleName = "ProportionThroughCycle", 
periodicLowerName = "PositionLower", 
periodicCenterName = "PositionCenter", 
periodicUpperName = "PositionUpper", plottedPointCountPerCycle = 120, 
graphFloor = min(base::pretty(x = dsLinear[, yName])))

Arguments

dsLinear The data.frame to containing the simple linear data. There should be one record per observation.
dsStageCycle The data.frame to containing the reoccurring/periodic bands. There should be one record per observation per stage. If there are three stages, this data.frame should have three times as many rows as dsLinear.
yName The variable name containing the dependent/criterion variable.
stageIDName The variable name indicating which stage the record belongs to. For example, before the first interruption, the StageID is 1, and is 2 afterwards.
cycleTallyName The variable name indicating how many complete cycles have occurred at that observation.
proportionThroughCycleName The variable name showing how far through a cycle the observation (or summarized observations) occurred.
periodicLowerName The variable name showing the lower bound of a stage’s periodic estimate.
periodicCenterName The variable name showing the center estimate of a stage’s periodic estimate.
periodicUpperName The variable name showing the upper bound of a stage’s periodic estimate.
plottedPointCountPerCycle The number of points that are plotted per cycle. If the polar graph has ‘sharp corners’, then increase this value.
graphFloor The value of the criterion/dependent variable at the center of the polar plot.

Value

Returns a data.frame.

Examples

library(Wats)
dsLinear <- CountyMonthBirthRate2005Version
dsLinear <- dsLinear[dsLinear$CountyName=="oklahoma", ]
dsLinear <- AugmentYearDataWithMonthResolution(dslinear=dsLinear, dateName="Date")

hSpread <- function(scores ) { return( quantile(x=scores, probs=c(.25, .75)) ) }
portfolio <- AnnotateData(
  dslinear = dslinear,
  dvName = "BirthRate",
  centerFunction = median,
  spreadFunction = hspread
)
rm(dslinear)
polarized <- PolarizeCartesian(
  dslinear = portfolio$dslinear,
  dsStageCycle = portfolio$dsStageCycle,
  yName = "BirthRate",
  stageIDName = "StageID"
)
library(ggplot2)
ggplot(polarized$dsStageCyclePolar, aes(color=factor(StageID))) +
  geom_path(aes(x=PolarLowerX, y=PolarLowerY), linetype=2) +
  geom_path(aes(x=PolarCenterX, y=PolarCenterY), size=2) +
  geom_path(aes(x=PolarUpperX, y=PolarUpperY), linetype=2) +
  geom_path(aes(x=ObservedX, y=ObservedY), data=polarized$dsObservedPolar) +
  coord_fixed(ratio=1) +
  guides(color=FALSE)

---

PolarPeriodic  

**Polar Plot with Periodic Elements**

**Description**

Shows the interrupted time series in Cartesian coordinates and its a periodic/cyclic components.

**Usage**

PolarPeriodic(dslinear, dsStageCyclePolar, xName, yName, stageIDName,
periodicLowerName = "PositionLower", periodicUpperName = "PositionUpper",
paletteDark = NULL, paletteLight = NULL, changePoints = NULL,
changePointLabels = NULL, drawObservedLine = TRUE,
drawPeriodicBand = TRUE, drawStageLabels = FALSE,
drawRadiusLabels = FALSE, jaggedPointSize = 2, jaggedLineSize = 1,
bandAlphaDark = 0.4, bandAlphaLight = 0.15, colorLabels = "gray50",
colorGridlines = "gray80", labelColor = "orange3",
changeLineAlpha = 0.5, changeLineSize = 3,
tickLocations = base::pretty(x = dslinear[, yName]),
graphFloor = min(tickLocations), graphCeiling = max(tickLocations),
cardinalLabels = NULL, originLabel = paste0("The origin represents ",
graphFloor, "; \the perimeter represents ", graphCeiling, "."),
plotMargins = c(3.5, 2, 0.5, 2))
Arguments

- **dsLinear**: The `data.frame` containing the simple linear data. There should be one record per observation.
- **dsStageCyclePolar**: The `data.frame` containing the bands for a single period. There should be one record per theta per stage. If there are three stages, this `data.frame` should have three times as many rows as `dsLinear`.
- **xName**: The variable name containing the date.
- **yName**: The variable name containing the dependent/criterion variable.
- **stageIDName**: The variable name indicating which stage the record belongs to. For example, before the first interruption, the `stageID` is 1, and is 2 afterwards.
- **periodicLowerName**: The variable name showing the lower bound of a stage’s periodic estimate.
- **periodicUpperName**: The variable name showing the upper bound of a stage’s periodic estimate.
- **paletteDark**: A vector of colors used for the dark/heavy graphical elements. The vector should have one color for each `stageID` value. If no vector is specified, a default will be chosen, based on the number of stages.
- **paletteLight**: A vector of colors used for the light graphical elements. The vector should have one color for each `stageID` value. If no vector is specified, a default will be chosen, based on the number of stages.
- **changePoints**: A vector of values indicating the interruptions between stages. It typically works best as a `Date` or a `POSIXct` class.
- **changePointLabels**: The text plotted above each interruption.
- **drawObservedLine**: A boolean value indicating if the longitudinal observed line should be plotted (whose values are taken from `dsLinear`).
- **drawPeriodicBand**: A boolean value indicating if the bands should be plotted (whose values are taken from the `periodicLowerName` and `periodicUpperName` fields).
- **drawStageLabels**: A boolean value indicating if the stage labels should be plotted (whose values are taken from `dsLinear`).
- **drawRadiusLabels**: A boolean value indicating if the gridline/radius labels should be plotted (whose values are taken from `tickLocations`).
- **jaggedPointSize**: The size of the observed data points.
- **jaggedLineSize**: The size of the line connecting the observed data points.
- **bandAlphaDark**: The amount of transparency of the band appropriate for a stage’s `x` values.
- **bandAlphaLight**: The amount of transparency of the band comparison stages for a given `x` value.
- **colorLabels**: The color for `cardinalLabels` and `originLabel`. 
colorGridlines  The color for the gridlines.

labelColor  The color of the text labels imposed on the line.

changeLineAlpha  The amount of transparency marking each interruption.

changeLineSize  The width of a line marking an interruption.

tickLocations  The desired locations for ticks showing the value of the criterion/dependent variable.

graphFloor  The value of the criterion/dependent variable at the center of the polar plot.

graphCeiling  The value of the criterion/dependent variable at the outside of the polar plot.

cardinalLabels  The four labels placed where ‘North’, ‘East’, ‘South’, and ‘West’ typically are.

originLabel  Explains what the criterion variable’s value is at the origin. Use NULL if no explanation is desired.

plotMargins  A vector of four numeric values, specifying the number of lines in the bottom, left, top and right margins.

Value

Returns a grid graphical object (ie, a grob.)

Examples

library(grid)
library(Wats)
dsLinear <- CountyMonthBirthRate2005Version
dLinear <- dsLinear[dsLinear$CountyName=="oklahoma", ]
dLinear <- AugmentYearDataWithMonthResolution(dLinear=dsLinear, dateName="Date")

hSpread <- function( scores ) { return( quantile(x=scores, probs=c(.25, .75)) ) }

portfolio <- AnnotateData(
  dsLinear = dsLinear,
  dvName = "BirthRate",
  centerFunction = median,
  spreadFunction = hSpread
)

rm(dsLinear)

polarized <- PolarizeCartesian(
  portfolio=dsLinear,
  portfolio$dsStageCycle,
  yName = "BirthRate",
  stageIDName = "StageID"
)

grid.newpage()

PolarPeriodic(
  dsLinear = polarized$dsObservedPolar,
  dsStageCyclePolar = polarized$dsStageCyclePolar,
  yName = "Radius",
  stageIDName = "StageID",
)
cardinalLabels = c("Jan1", "Apr1", "July1", "Oct1")
)

grid.newpage()
PolarPeriodic(
  dslinear = polarized$dsObservedPolar,
  dsStageCyclePolar = polarized$dsStageCyclePolar,
  yName = "Radius",
  stageIDName = "StageID",
  drawPeriodicBand = FALSE
)

grid.newpage()
PolarPeriodic(
  dslinear = polarized$dsObservedPolar,
  dsStageCyclePolar = polarized$dsStageCyclePolar,
  yName = "Radius",
  stageIDName = "StageID",
  drawObservedLine = FALSE,
  cardinalLabels = c("Jan1", "Apr1", "July1", "Oct1")
)
Index

*Topic Cartesian
   CartesianPeriodic, 5
   CartesianRolling, 6
*Topic datasets
   CountyMonthBirthRate, 9
*Topic package
   Wats-package, 2
*Topic polar
   PolarizeCartesian, 10
   PolarPeriodic, 12

AnnotateData, 3
AugmentCycleData, 4
AugmentYearDataWithMonthResolution
   (AugmentCycleData), 4
AugmentYearDataWithSecondResolution
   (AugmentCycleData), 4

CartesianPeriodic, 5
CartesianRolling, 6
CountyMonthBirthRate, 9
CountyMonthBirthRate2005Version
   (CountyMonthBirthRate), 9
CountyMonthBirthRate2014Version
   (CountyMonthBirthRate), 9

PolarizeCartesian, 10
PolarPeriodic, 12

Wats (Wats-package), 2
Wats-package, 2