Package 'Wats'

May 7, 2026

Title Wrap Around Time Series Graphics

Description Wrap-around Time Series (WATS) plots for interrupted time series designs with seasonal patterns.
Longitudinal trajectories are shown in both Cartesian and polar coordinates.
In many scenarios, a WATS plot more clearly shows the existence and effect size of of an intervention.
This package accompanies
``Graphical Data Analysis on the Circle: Wrap-
Around Time Series Plots for (Interrupted) Time Series Designs"
by Rodgers, Beasley, & Schuelke (2014)
<doi:10.1080/00273171.2014.946589>;
see 'citation(``Wats")' for details.

Version 1.0.1

URL https://ouhscbbmc.github.io/Wats/,
https://github.com/OuhscBbmc/Wats

BugReports https://github.com/OuhscBbmc/Wats/issues

Depends R (>= 4.2.0)

Imports colorspace, dplyr, ggplot2, grid, lubridate, RColorBrewer, rlang, testit, tibble, zoo

Suggests boot, covr, devtools, knitr, scales, testthat

License MIT + file LICENSE

LazyData TRUE

VignetteBuilder knitr

Language en-US

Encoding UTF-8

RoxygenNote 7.2.3

Config/testthat/edition 3

NeedsCompilation no

Author Will Beasley [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0002-5613-5006>),
Joe Rodgers [aut],
Matthew Schuelke [ctb],
Ronnie Coleman [ctb],
Mark Joseph Lachowicz [ctb],
Oklahoma Shared Clinical and Translational Resource (OSCTR) [fnd]

Maintainer Will Beasley <wibeasley@hotmail.com>

Repository CRAN

Date/Publication 2023-03-10 22:50:05 UTC

Contents

Wats-package 2
annotate_data 3
augment_cycle_data 5
cartesian_periodic 5
cartesian_rolling 8
county_month_birth_rate 10
polarize_cartesian 12
polar_periodic 14

Index **19**

`Wats-package` *Wrap Around Time Series graphics*

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 `remotes::install_github("OuhscBbmc/Wats")`. (make sure remotes 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 an issue, or email Will.

Funding Continued development and maintenance of this package supported by the Oklahoma Shared Clinical and Translational Resources (OSCTR, U54GM104938) with an Institutional Development Award (IDeA) from NIGMS. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

Author(s)

William Howard Beasley –Associate 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, Office of the Vice President for Research Saint Louis University

References

Rodgers, J.L., Beasley, W.H., and Schuelke, M. (2014). Graphical Data Analysis on the Circle: Wrap-around Time Series Plots for (Interrupted) Time Series Designs. *Multivariate Behavioral Research*.

Rodgers, J.L., St. John, C. A. & Coleman R. (2005). Did Fertility Go Up after the Oklahoma City Bombing? An Analysis of Births in Metropolitan Counties in Oklahoma, 1990-1999. *Demography, 42*, 675-692.

`annotate_data` *Finds midpoints and bands for the within and between cycles.*

Description

Finds midpoints and bands for the within and between cycles. This the second of two functions that needs to be called to produce WATS Plots. `AugmentZZZ` is the first.

Usage

```
annotate_data(
  ds_linear,
  dv_name,
  center_function,
  spread_function,
  cycle_tally_name = "cycle_tally",
  stage_id_name = "stage_id",
  stage_progress_name = "stage_progress",
  proportion_through_cycle_name = "proportion_through_cycle",
  proportion_id_name = "proportion_id",
  terminal_point_in_cycle_name = "terminal_point_in_cycle"
)
```

Arguments

`ds_linear`	The data.frame to containing the detailed data.
`dv_name`	The name of the dependent/criterion variable.

`center_function`
: A function to calculate the center of a subsample.

`spread_function`
: A function to calculate the bands of a subsample.

`cycle_tally_name`
: The variable name indicating how many cycles have been completed.

`stage_id_name`
: The variable name indicating the stage. In a typical interrupted time series, these values are "1" before the interruption and "2" after.

`stage_progress_name`
: The variable name indicating the stage in a decimal form. This is mostly for internal uses.

`proportion_through_cycle_name`
: 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`.

`proportion_id_name`
: The variable name indicating the ordinal position through a cycle.

`terminal_point_in_cycle_name`
: The variable name indicating the last point within a given cycle.

Value

Returns a `tibble::tibble()` with additional variables. TODO: say what the variables are.

Examples

```
system.time({
library(Wats)
ds_linear <-
  Wats::county_month_birth_rate_2005_version |>
  dplyr::filter(county_name == "oklahoma") |>
  augment_year_data_with_month_resolution(date_name = "date")

h_spread <- \(scores) { quantile(x = scores, probs = c(.25, .75)) }

portfolio <- annotate_data(
  ds_linear       = ds_linear,
  dv_name         = "birth_rate",
  center_function = median,
  spread_function = h_spread
)
portfolio$ds_stage_cycle
portfolio$ds_linear
portfolio$ds_periodic
})
```

augment_cycle_data *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. `annotate_data()` is the second.

Usage

```
augment_year_data_with_month_resolution(ds_linear, date_name)
augment_year_data_with_second_resolution(ds_linear, date_name)
```

Arguments

`ds_linear`	The data.frame to containing the detailed data.
`date_name`	The variable name in `ds_linear` containing the date or datetime value.

Value

Returns a tibble::tibble with additional variables: `cycle_tally`, `proportion_through_cycle`, `proportion_id`, and `terminal_point_in_cycle`.

Examples

```
library(Wats)
ds_linear <-
  Wats::county_month_birth_rate_2005_version |>
  dplyr::filter(county_name == "oklahoma") |>
  augment_year_data_with_month_resolution(date_name = "date")

head(ds_linear)
```

cartesian_periodic *Linear Plot with Periodic Elements*

Description

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

Usage

```
cartesian_periodic(
  ds_linear,
  ds_periodic,
  x_name,
  y_name,
  stage_id_name,
  periodic_lower_name = "position_lower",
  periodic_upper_name = "position_upper",
  palette_dark = NULL,
  palette_light = NULL,
  change_points = NULL,
  change_point_labels = NULL,
  draw_periodic_band = TRUE,
  jagged_point_size = 2,
  jagged_line_size = 0.5,
  band_alpha_dark = 0.4,
  band_alpha_light = 0.15,
  change_line_alpha = 0.5,
  change_line_size = 3,
  title = NULL,
  x_title = NULL,
  y_title = NULL
)
```

Arguments

`ds_linear`	The data.frame to containing the simple linear data. There should be one record per observation.
`ds_periodic`	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 `ds_linear`.
`x_name`	The variable name containing the date.
`y_name`	The variable name containing the dependent/criterion variable.
`stage_id_name`	The variable name indicating which stage the record belongs to. For example, before the first interruption, the `stage_id` is "1", and is "2" afterwards.
`periodic_lower_name`	The variable name showing the lower bound of a stage's periodic estimate.
`periodic_upper_name`	The variable name showing the upper bound of a stage's periodic estimate.
`palette_dark`	A vector of colors used for the dark/heavy graphical elements. The vector should have one color for each `stage_id` value. If no vector is specified, a default will be chosen, based on the number of stages.
`palette_light`	A vector of colors used for the light graphical elements. The vector should have one color for each `stage_id` value. If no vector is specified, a default will be chosen, based on the number of stages.

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

`change_point_labels`
The text plotted above each interruption.

`draw_periodic_band`
A boolean value indicating if the bands should be plotted (whose values are take from the `periodic_lower_name` and `periodic_upper_name`).

`jagged_point_size`
The size of the observed data points.

`jagged_line_size`
The size of the line connecting the observed data points.

`band_alpha_dark`
The amount of transparency of the band appropriate for a stage's *x* values.

`band_alpha_light`
The amount of transparency of the band comparison stages for a given *x* value.

`change_line_alpha`
The amount of transparency marking each interruption.

`change_line_size`
The width of a line marking an interruption.

`title` The string describing the plot.

`x_title` The string describing the *x*-axis.

`y_title` The string describing the *y*-axis.

Value

Returns a ggplot2 graphing object

Examples

```
library(Wats) # Load the package
change_month <- base::as.Date("1996-02-15")
ds_linear <-
  Wats::county_month_birth_rate_2005_version |>
  dplyr::filter(county_name == "oklahoma") |>
  augment_year_data_with_month_resolution(date_name = "date")

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

portfolio <- annotate_data(
  ds_linear,
  dv_name = "birth_rate",
  center_function = median,
  spread_function = h_spread
)

cartesian_periodic(
  portfolio$ds_linear,
  portfolio$ds_periodic,
```

```
  x_name               = "date",
  y_name               = "birth_rate",
  stage_id_name        = "stage_id",
  change_points        = change_month,
  change_point_labels  = "Bombing Effect"
)
```

`cartesian_rolling` *Linear Plot with Rolling Summaries*

Description

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

Usage

```
cartesian_rolling(
  ds_linear,
  x_name,
  y_name,
  stage_id_name,
  rolling_lower_name = "rolling_lower",
  rolling_center_name = "rolling_center",
  rolling_upper_name = "rolling_upper",
  palette_dark = NULL,
  palette_light = NULL,
  color_sparse = grDevices::adjustcolor("tan1", 0.5),
  change_points = NULL,
  change_point_labels = NULL,
  draw_jagged_line = TRUE,
  draw_rolling_line = TRUE,
  draw_rolling_band = TRUE,
  draw_sparse_line_and_points = TRUE,
  jagged_point_size = 2,
  jagged_line_size = 0.5,
  rolling_line_size = 1,
  sparse_point_size = 4,
  sparse_line_size = 0.5,
  band_alpha = 0.4,
  change_line_alpha = 0.5,
  change_line_size = 3,
  title = NULL,
  x_title = NULL,
  y_title = NULL
)
```

Arguments

`ds_linear`	The data.frame to containing the data.
`x_name`	The variable name containing the date.
`y_name`	The variable name containing the dependent/criterion variable.
`stage_id_name`	The variable name indicating which stage the record belongs to. For example, before the first interruption, the `stage_id` is "1", and is "2" afterwards.
`rolling_lower_name`	The variable name showing the lower bound of the rolling estimate.
`rolling_center_name`	The variable name showing the rolling estimate.
`rolling_upper_name`	The variable name showing the upper bound of the rolling estimate.
`palette_dark`	A vector of colors used for the dark/heavy graphical elements. The vector should have one color for each `stage_id` value. If no vector is specified, a default will be chosen, based on the number of stages.
`palette_light`	A vector of colors used for the light graphical elements. The vector should have one color for each `stage_id` value. If no vector is specified, a default will be chosen, based on the number of stages.
`color_sparse`	The color of the 'slowest' trend line, which plots only one value per cycle.
`change_points`	A vector of values indicate the interruptions between stages. It typically works best as a Date or a POSIXct class.
`change_point_labels`	The text plotted above each interruption.
`draw_jagged_line`	A boolean value indicating if a line should be plotted that connects the observed data points.
`draw_rolling_line`	A boolean value indicating if a line should be plotted that connects the rolling estimates specified by `rolling_center_name`.
`draw_rolling_band`	A boolean value indicating if a band should be plotted that envelopes the rolling estimates (whose values are take from the `rolling_lower_name` and `rolling_upper_name`.
`draw_sparse_line_and_points`	A boolean value indicating if the sparse line and points should be plotted.
`jagged_point_size`	The size of the observed data points.
`jagged_line_size`	The size of the line connecting the observed data points.
`rolling_line_size`	The size of the line connecting the rolling estimates.
`sparse_point_size`	The size of the sparse estimates.
`sparse_line_size`	The size of the line connecting the sparse estimates.

band_alpha The amount of transparency of the rolling estimate band.

change_line_alpha
The amount of transparency marking each interruption.

change_line_size
The width of a line marking an interruption.

title The string describing the plot.

x_title The string describing the *x*-axis.

y_title The string describing the *y*-axis.

Value

Returns a ggplot2 graphing object

Examples

```
library(Wats) # Load the package
change_month <- base::as.Date("1996-02-15")
ds_linear <-
  Wats::county_month_birth_rate_2005_version |>
  dplyr::filter(county_name == "oklahoma") |>
  augment_year_data_with_month_resolution(date_name = "date")

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

portfolio <- annotate_data(
  ds_linear,
  dv_name         = "birth_rate",
  center_function = median,
  spread_function = h_spread
)

cartesian_rolling(
  portfolio$ds_linear,
  x_name               = "date",
  y_name               = "birth_rate",
  stage_id_name        = "stage_id",
  change_points        = change_month,
  change_point_labels  = "Bombing Effect"
)
```

---

county_month_birth_rate

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`

county_name 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`

fecund_population The number of females in the county, ages of 15 to 44. `numeric`

birth_count 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`

days_in_month The number of days in the specific month. `integer`

days_in_year The number of days in the specific years `integer`

stage_id The "Stage" of the month. The pre-bombing records are "1" (accounting for 9 months of gestation); the post-bombing months are "2". `integer`

birth_rate 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 `fecund_population`. Values for February-December are interpolated using `approx()`.

The datasets were manipulated to produce this data frame by the two R files isolate-census-pops-for-gfr.R and calculate-gfr.R.

Author(s)

Will Beasley

References

- Rodgers, J. L., St. John, C. A. & Coleman R. (2005). Did Fertility Go Up after the Oklahoma City Bombing? An Analysis of Births in Metropolitan Counties in Oklahoma, 1990-1999. *Demography, 42*, 675-692.
- Intercensal estimates for 199x
- Intercensal estimates for 200x
- Documentation: US Census Intercensal Estimates for 199x and 200x.

Examples

```
library(ggplot2)

# 2005 Version (see description above)
ds2005 <- county_month_birth_rate_2005_version
ggplot(ds2005, aes(x = date, y = birth_rate, color = factor(fips))) +
  geom_line() +
  labs(title="County Fertility - Longitudinal")

ggplot(ds2005, aes(x = birth_rate, color = factor(fips))) +
  geom_density() +
  labs(title="Distributions of County Fertility")


# 2014 Version (see description above)
ds2014 <- county_month_birth_rate_2014_version
ggplot(ds2014, aes(x = date, y = birth_rate, color = factor(fips))) +
  geom_line() +
  labs(title="County Fertility - Longitudinal")

ggplot(ds2014, aes(x = birth_rate, color = factor(fips))) +
  geom_density() +
  labs(title="Distributions of County Fertility")
```

`polarize_cartesian` *Manipulate Cartesian data to use in the WATS polar plot*

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.

Usage

```
polarize_cartesian(
  ds_linear,
  ds_stage_cycle,
  y_name,
  stage_id_name,
  cycle_tally_name = "cycle_tally",
  proportion_through_cycle_name = "proportion_through_cycle",
  periodic_lower_name = "position_lower",
  periodic_center_name = "position_center",
  periodic_upper_name = "position_upper",
  plotted_point_count_per_cycle = 120,
  graph_floor = min(base::pretty(ds_linear[[y_name]]))
)
```

Arguments

`ds_linear`	The data.frame to containing the simple linear data. There should be one record per observation.
`ds_stage_cycle`	The data.frame to containing the reoccurring/periodic bands. There should be one record per observation per stage. If there are three stages, this tibble::tibble should have three times as many rows as `ds_linear`.
`y_name`	The variable name containing the dependent/criterion variable.
`stage_id_name`	The variable name indicating which stage the record belongs to. For example, before the first interruption, the `stage_id` is "1", and is "2" afterwards.
`cycle_tally_name`	The variable name indicating how many *complete* cycles have occurred at that observation.
`proportion_through_cycle_name`	The variable name showing how far through a cycle the observation (or summarized observations) occurred.
`periodic_lower_name`	The variable name showing the lower bound of a stage's periodic estimate.
`periodic_center_name`	The variable name showing the center estimate of a stage's periodic estimate.
`periodic_upper_name`	The variable name showing the upper bound of a stage's periodic estimate.
`plotted_point_count_per_cycle`	The number of points that are plotted per cycle. If the polar graph has 'sharp corners', then increase this value.
`graph_floor`	The value of the criterion/dependent variable at the center of the polar plot.

Value

Returns a tibble::tibble.

Examples

```
library(Wats)
ds_linear <-
  Wats::county_month_birth_rate_2005_version |>
  dplyr::filter(county_name == "oklahoma") |>
  augment_year_data_with_month_resolution(date_name = "date")

h_spread <- function(scores) { quantile(x = scores, probs = c(.25, .75)) }
portfolio <- annotate_data(
  ds_linear        = ds_linear,
  dv_name          = "birth_rate",
  center_function  = median,
  spread_function  = h_spread
)
rm(ds_linear)

polarized <- polarize_cartesian(
  ds_linear      = portfolio$ds_linear,
  ds_stage_cycle = portfolio$ds_stage_cycle,
  y_name         = "birth_rate",
  stage_id_name  = "stage_id"
)

library(ggplot2)
polarized$ds_stage_cycle_polar |>
  ggplot(aes(color = factor(stage_id))) +
  geom_path(aes(x = polar_lower_x , y = polar_lower_y), linetype = 2) +
  geom_path(aes(x = polar_center_x, y = polar_center_y), linewidth = 2) +
  geom_path(aes(x = polar_upper_x , y = polar_upper_y), linetype = 2) +
  geom_path(aes(x = observed_x    , y = observed_y), data = polarized$ds_observed_polar) +
  coord_fixed(ratio = 1) +
  guides(color = NULL)
```

---

polar_periodic *Polar Plot with Periodic Elements*

Description

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

Usage

```
polar_periodic(
  ds_linear,
  ds_stage_cycle_polar,
  x_name,
  y_name,
  stage_id_name,
  periodic_lower_name = "position_lower",
```

```
  periodic_upper_name = "position_upper",
  palette_dark = NULL,
  palette_light = NULL,
  change_points = NULL,
  change_point_labels = NULL,
  draw_observed_line = TRUE,
  draw_periodic_band = TRUE,
  draw_stage_labels = FALSE,
  draw_radius_labels = FALSE,
  jagged_point_size = 2,
  jagged_line_size = 1,
  band_alpha_dark = 0.4,
  band_alpha_light = 0.15,
  color_labels = "gray50",
  color_gridlines = "gray80",
  label_color = "orange3",
  change_line_alpha = 0.5,
  change_line_size = 3,
  tick_locations = base::pretty(x = ds_linear[[y_name]]),
  graph_floor = min(tick_locations),
  graph_ceiling = max(tick_locations),
  cardinal_labels = NULL,
  origin_label = paste0("The origin represents ", graph_floor,
    ";\nthe perimeter represents ", graph_ceiling, "."),
  plot_margins = c(3.5, 2, 0.5, 2)
)
```

Arguments

ds_linear	The data.frame to containing the simple linear data. There should be one record per observation.
ds_stage_cycle_polar	The data.frame to 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 ds_linear.
x_name	The variable name containing the date.
y_name	The variable name containing the dependent/criterion variable.
stage_id_name	The variable name indicating which stage the record belongs to. For example, before the first interruption, the stage_id is "1", and is "2" afterwards.
periodic_lower_name	The variable name showing the lower bound of a stage's periodic estimate.
periodic_upper_name	The variable name showing the upper bound of a stage's periodic estimate.
palette_dark	A vector of colors used for the dark/heavy graphical elements. The vector should have one color for each stage_id value. If no vector is specified, a default will be chosen, based on the number of stages.

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

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

`change_point_labels`
The text plotted above each interruption.

`draw_observed_line`
A boolean value indicating if the longitudinal observed line should be plotted (whose values are take from `ds_linear`).

`draw_periodic_band`
A boolean value indicating if the bands should be plotted (whose values are take from the `periodic_lower_name` and `periodic_upper_name` fields).

`draw_stage_labels`
A boolean value indicating if the stage labels should be plotted (whose values are take from `ds_linear`).

`draw_radius_labels`
A boolean value indicating if the gridline/radius labels should be plotted (whose values are take from `tick_locations`).

`jagged_point_size`
The size of the observed data points.

`jagged_line_size`
The size of the line connecting the observed data points.

`band_alpha_dark`
The amount of transparency of the band appropriate for a stage's x values.

`band_alpha_light`
The amount of transparency of the band comparison stages for a given x value.

`color_labels` The color for `cardinal_labels` and `origin_label`.

`color_gridlines`
The color for the gridlines.

`label_color` The color of the text labels imposed on the line.

`change_line_alpha`
The amount of transparency marking each interruption.

`change_line_size`
The width of a line marking an interruption.

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

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

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

`cardinal_labels`
The four labels placed where "North", "East", "South", and "West" typically are.

`origin_label` Explains what the criterion variable's value is at the origin. Use `NULL` if no explanation is desired.

`plot_margins` 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 (*i.e.*, a `grid::grob()`.)

Examples

```
requireNamespace("grid")
library(Wats)
ds_linear <-
  Wats::county_month_birth_rate_2005_version |>
  dplyr::filter(county_name == "oklahoma") |>
  augment_year_data_with_month_resolution(date_name = "date")

h_spread <- function(scores) { quantile(x = scores, probs = c(.25, .75)) }
portfolio <- annotate_data(
  ds_linear       = ds_linear,
  dv_name         = "birth_rate",
  center_function = median,
  spread_function = h_spread
)
rm(ds_linear)

polarized <- polarize_cartesian(
  portfolio$ds_linear,
  portfolio$ds_stage_cycle,
  y_name        = "birth_rate",
  stage_id_name = "stage_id"
)

grid::grid.newpage()
polar_periodic(
  ds_linear             = polarized$ds_observed_polar,
  ds_stage_cycle_polar  = polarized$ds_stage_cycle_polar,
  y_name                = "radius",
  stage_id_name         = "stage_id",
  cardinal_labels       = c("Jan1", "Apr1", "July1", "Oct1")
)

grid::grid.newpage()
polar_periodic(
  ds_linear              = polarized$ds_observed_polar,
  ds_stage_cycle_polar   = polarized$ds_stage_cycle_polar,
  y_name                 = "radius",
  stage_id_name          = "stage_id",
  draw_periodic_band     = FALSE
)

grid::grid.newpage()
polar_periodic(
  ds_linear            = polarized$ds_observed_polar,
  ds_stage_cycle_polar = polarized$ds_stage_cycle_polar,
  y_name               = "radius",
  stage_id_name        = "stage_id",
```

```
  draw_observed_line   = FALSE,
  cardinal_labels      = c("Jan1", "Apr1", "July1", "Oct1")
)
```

Index

∗ **Cartesian**
    cartesian_periodic, 5
    cartesian_rolling, 8
∗ **datasets**
    county_month_birth_rate, 10
∗ **package**
    Wats-package, 2
∗ **polar**
    polar_periodic, 14
    polarize_cartesian, 12

annotate_data, 3
annotate_data(), *5*
approx(), *11*
augment_cycle_data, 5
augment_year_data_with_month_resolution
        *(*augment_cycle_data*)*, 5
augment_year_data_with_second_resolution
        *(*augment_cycle_data*)*, 5

cartesian_periodic, 5
cartesian_rolling, 8
county_month_birth_rate, 10
county_month_birth_rate_2005_version
        *(*county_month_birth_rate*)*, 10
county_month_birth_rate_2014_version
        *(*county_month_birth_rate*)*, 10

data.frame, *3*, *5*, *6*, *9*, *13*, *15*

grid::grob(), *17*

polar_periodic, 14
polarize_cartesian, 12

tibble::tibble, *5*, *13*
tibble::tibble(), *4*

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