Package: winnify (via r-universe)

November 14, 2026

Type Package

Title Visual Summaries and Wrappers for Estimating the Win Ratio and Related Measures

Version 1.0.0

Maintainer Hannah Johns <dr.hannah.johns@gmail.com>

Description Facilitates analysis of Generalised Pairwise Comparisons with Prioritised Outcomes (e.g. Win Ratio, Win Odds). Includes a wrapper function that enables rank-based Desirability of Outcome Ranking analyses with any existing ordinal analysis package (e.g. probabilistic index models provided by R package `pim') plus automated input processing and a formula-based interface for estimating win statistics (using R package `WINS') without needing to restructure or rename the data. In both cases, supports improved interpretation of prioritised outcomes analyses using the approach proposed by Johns et al. (2026) <doi:10.1177/17474930261475853> where the treatment effect is estimated separately for each outcome plus cumulatively as they are added to the outcome prioritisation. The contribution of each outcome to the overall effect size is then visualised with a supplied plotting function.

License GPL (>= 2.0)

Encoding UTF-8

LazyData true

Depends R (>= 3.5)

Imports ggplot2, RColorBrewer, patchwork, rlang, WINS, methods, sloop

Suggests genodds, pim, MASS, testthat (>= 3.0.0), WinRatio, survival

Config/roxygen2/version 8.1.0

Config/testthat/edition 3

NeedsCompilation no

Author Hannah Johns [aut, cre] (ORCID: <https://orcid.org/0000-0003-2135-0504>), Nathalie Launder [aut] (ORCID: <https://orcid.org/0000-0002-7898-0083>), Leonid Churilov [aut] (ORCID: <https://orcid.org/0000-0002-9807-6606>)

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

Date/Publication 2026-11-13 23:10:02 UTC

RemoteUrl https://github.com/cran/winnify

RemoteRef HEAD

RemoteSha d964444352fcc80bbaefc757f1e391e238126377

Contents

as_winnify 2
neuro_improvement 6
ranks 6
remapcap 9
tte_covariates 10
tte_longform 11
tte_outcomes 11
w_arm 12
w_id 13
w_iptw 13
w_stratum 14
w_time 14
winnify 15
wins 19

Index **23**

`as_winnify` *Convert other objects to a Winnify object*

Description

Generic function for converting objects to Winnify objects

Convenience function for visualising Win Ratio as estimated by `winratio` from the WinRatio package.

Usage

```
as_winnify(x, ...)

## S3 method for class 'data.frame'
as_winnify(
  x,
  effect_size = NULL,
  neutral_point = NULL,
  tie_handling = NULL,
  outcome_names = "outcome_names",
  win_cumulative = "win_cumulative",
```

```
  tie_cumulative = "tie_cumulative",
  loss_cumulative = "loss_cumulative",
  estimate_cumulative = "estimate_cumulative",
  lower_cumulative = "lower_cumulative",
  upper_cumulative = "upper_cumulative",
  pval_cumulative = "pval_cumulative",
  estimate_individual = "estimate_individual",
  lower_individual = "lower_individual",
  upper_individual = "upper_individual",
  pval_individual = "pval_individual",
  ignore_missing = FALSE,
  ...
)

## S3 method for class 'WinRatio'
as_winnify(x, ...)
```

Arguments

x	An object of class WinRatio containing the output from winratio()
...	further arguments passed to other methods
effect_size	A string indicating the the column of data containing the effect size name, or a string specifying this directly. See Details.
neutral_point	A string indicating the the column of data containing the neutral value, or a number specifying this directly. See Details.
tie_handling	A string indicating the the column of data containing how ties are handled, or a string specifying this directly. See Details.
outcome_names	A string indicating the column of data containing the names of the outcomes.
win_cumulative	A string indicating the column of data containing the total wins as each outcome is added.
tie_cumulative	A string indicating the column of data containing the total ties as each outcome is added.
loss_cumulative	A string indicating the column of data containing the total losses as each outcome is added.
estimate_cumulative	A string indicating the column of data containing the cumulative effect size as each outcome is added.
lower_cumulative	A string indicating the column of data containing the lower boundary of the confidence interval for the cumulative effect as each outcome is added.
upper_cumulative	A string indicating the column of data containing the upper boundary of the confidence interval for the cumulative effect size as each outcome is added.
pval_cumulative	A string indicating the column of data containing the p value for estimated cumulative effect size as each outcome is added.

`estimate_individual`
: A string indicating the column of `data` containing the effect size for each outcome when considered independently.

`lower_individual`
: A string indicating the column of `data` containing the lower boundary of the confidence interval for the effect size for each outcome when considered independently.

`upper_individual`
: A string indicating the column of `data` containing the upper boundary of the confidence interval for the effect size for each outcome when considered independently.

`pval_individual`
: A string indicating the column of `data` containing the p value for estimated cumulative effect size for each outcome when considered independently.

`ignore_missing`
: A logical indicating if behaviour of columns of data are not found. If `TRUE`, columns with no match in `data` are replaced with `NA` and a message is returned indicating this.

Details

as_winnify methods exist for the following objects:

- data.frame objects from base R. See `as_winnify.data.frame` for details.
- WinRatio objects returned by the `winratio` function. See `as_winnify.WinRatio` for details.

`data` is assumed to have rows ordered from most important to least important.

`tie_handling` is used to specify the default option for the `pairs_tie` argument in plot.Winnify. See this function's help file for more detail.

Arguments `effect_name`, `neutral_point` and `tie_handling` will first look for a column in `data` that matches the value supplied. If no matching column is found, the input is treated as a constant and is added to the output. By default, `effect_name` looks for a column "effect_size", and if this column is not found then it will use "Effect size" as a constant label for the output. By default, `neutral_point` and `tie_handling` are omitted.

If `NULL` is supplied a string arguments, then that column is skipped and NA values are returned where relevant with no message. This is more explicit than setting `ignore_missing` to be `TRUE` and is recommended if columns are missing in `data`.

Value

A `Winnify` object.

A `Winnify` object. See winnify for details.

A `Winnify` object. See winnify for details.

Examples

```
data <- data.frame(endpoint=c("Death", "Length of hospital stay", "Time off work", "Wellbeing"),
                   wins = c(15,25,33,48),
                   ties =c(80,60,40,15),
                   losses =c(5,15,27,37),
                   win_ratio=c(3,1,0.67,1.5),
                   ci_lower=c(2.6,0.6,0.27,1.1),
                   ci_upper=c(3.4,1.4,1.07,1.9),
                   win_ratio_cum=c(3,1.67,1.22,1.30),
                   ci_lower_cum=c(2.6,1.32,0.92,1.10),
                   ci_upper_cum=c(3.4,2.02,1.52,1.50)
)


result <- as_winnify(data,
                     effect_size = "Win Ratio",
                     neutral_point = 1,
                     tie_handling = "drop",
                     outcome_names = "endpoint",
                     win_cumulative = "wins",
                     tie_cumulative = "ties",
                     loss_cumulative = "losses",
                     estimate_cumulative = "win_ratio_cum",
                     lower_cumulative="ci_lower_cum",
                     upper_cumulative ="ci_upper_cum",
                     estimate_individual = "win_ratio",
                     lower_individual="ci_lower",
                     upper_individual ="ci_upper",
                     pval_cumulative = NULL,
                     pval_individual = NULL
)
plot(result)


# winratio function requires additional inputs than what's available in this dataset.

# Add an id variable to dataset
remapcap$id <- seq_len(nrow(remapcap))

# Calculate time at home as time post discharge
remapcap$homeTime <- (remapcap$deathTime-remapcap$dischargeTime)

# Collapse treatment groups so only one treatment group
remapcap$group <- ifelse(remapcap$group == "uc", "Control", "Treatment")

# Run winratio function on dataset
wr <- WinRatio::winratio(id = "id", trt = "group", active = "Treatment",
                outcomes = list("Death" = c("deathStatus", "s", "deathTime"),
                "Time at home" = c("homeTime", "c", "<")),
                fu = "deathTime", data = remapcap)

# Plot win ratio results
```

```
result <- as_winnify(wr)
plot(result)
```

`neuro_improvement` *Synthetic data showing modified Rankin Scale and neurological improvement*

Description

Synthetic data showing modified Rankin Scale and neurological improvement

Usage

```
neuro_improvement
```

Format

A data frame with 2000 rows and 3 variables:

treat Type of treatment received

mRS modified Rankin Score at 90 days (0= no disability, 6 = death)

early_neuro_improvement Change in neurological function at 24 hours

---

`ranks` *Wrapper function for rank-based statistical methods*

Description

A wrapper function for any rank-based statistical method, allowing it to be used with winnify().

Usage

```
ranks(
  formula,
  data,
  ranks_statistic,
  exponentiate = FALSE,
  outcome_as_factor = TRUE,
  ...
)
```

Arguments

formula	An object of class formula describing the analyses to run. Format for model specification are given in Details
data	A data.frame to analyse
ranks_statistic	A function to run. See Details.
exponentiate	A logical indicating if the results of ranks_statistic should be exponentiated
outcome_as_factor	A logical indicating if the outcome to be fed to ranks_statistic should be cast to an ordered factor. If FALSE, a numeric value is passed.
...	Any arguments to be passed to ranks_statistic()

Details

This wrapper combines items on the left hand side of formula into a single set of ranks and then runs ranks_statistic. When combined with winnify() it allows arbitrary analysis functions to be used for Generalised Pairwise Comparisons with Prioritised Outcomes.

The ranks_statistic function must satisfy the following criteria:

- The first argument of the function must be a formula
- There must be a named argument called data for providing the data
- It must have coef() and confint() methods available for its output, which return the effect size and confidence intervals respectively

The left hand side of formula should list outcomes in order of importance, with the most important outcome coming first. Note that these are ranked from smallest value to highest value, and a higher value is assumed to be preferred.

The right hand side of formula provides the grouping variable, marked with the function w_arm(x). It is expected that this variable is a factor containing exactly two values. If this term is not a factor, it will be coerced into one before ranks_statistic is run.

All terms of formula are fed into ranks_statistic. Note that the estimates for wins, losses and ties are unadjusted and ignore all terms other than the one marked as the grouping variable.

Note that if ranks_statistic requires profiling to be redone to estimate confidence intervals (via e.g. a vcov call), this function will crash. A notable example of this is polr. Any optional arguments to prevent re-profiling should be passed to ranks_statistic using ...

Value

A one-row data frame containing the following:

effect_size The name of the ranks_statistic function

estimate The estimate from ranks_statistic function

lower The lower bound of the condifence interval estimate from ranks_statistic function

upper The upper bound of the confidence interval estimate from ranks_statistic function

pval NA

win The unadjusted number of wins

loss The unadjusted number of losses

tie The unadjusted number of ties

Examples

```
data <- neuro_improvement
formula <- mRS+early_neuro_improvement~w_arm(treat)

## Use of ranks() by itself

# pim estimates the Win Odds for ordinal data
# and is structured to not require additional
# work to use it with ranks().



library(pim)
ranks(formula,data,ranks_statistic = pim)

## Use of ranks() in a winnify() call

# estim=estimator.glm is used here because it's fast
x <- winnify(formula, ranks, data, ranks_statistic = pim, exponentiate=TRUE)
plot(x,effect_neutral_point = 1,pairs_tie = "split")


# Running other analysis methods, e.g.
# Proportional odds model

this_data <- winnify::remapcap
this_data$days_alive <- this_data$deathTime
this_data$days_out_of_icu <- 21-this_data$dischargeTime

# We only want two groups - organise factor levels and drop other.
this_data$group <- factor(this_data$group,levels = c("uc","shock"))
this_data <- this_data[which(!is.na(this_data$group)),]

library(MASS)
ranks(days_alive + days_out_of_icu ~ w_arm(group),
      data = this_data,
      ranks_statistic = polr, Hess=TRUE,
      exponentiate = TRUE)
## Example of use of a function that needs a wrapper

# genodds estimates the Win Odds or Win Ratio for ordinal data
# but doesn't accept a formula input. This can be fixed by creating
# a wrapper function for it

library(genodds)

my_statistic <- function(formula,data,...){
```

```
        lhs <- stats::model.frame(formula[1:2],data=data)
        rhs <- stats::model.frame(formula[c(1,3)],data=data)

        if(ncol(lhs)!=1) stop("Left hand side of formula should have one term")
        if(ncol(rhs)!=1) stop("Left hand side of formula should have one term")

        fit <- genodds(response = lhs[,1],
                       group = rhs[,1],
                        ...)

        # genodds returns an object of class Genodds.
        # if it didn't, we would need to give it a class
        # that we can write write S3 methods for.
        # class(fit) <- c("MyRankStat",class(fit))
        fit
}

# There's no coef() or confint() functions for
# class Genodds. We need to create some S3 methods to
# extract this info.
coef.Genodds <- function(x){x$pooled_lnodds}
confint.Genodds <- function(x){matrix(x$pooled_lnconf.int,nrow=1)}

wo_result <- winnify(formula, ranks, data, ranks_statistic = my_statistic,
                     exponentiate=TRUE, ties="split")
plot(wo_result,effect_neutral_point = 1,pairs_tie = "split")

wr_result <- winnify(formula, ranks, data, ranks_statistic = my_statistic,
                     exponentiate=TRUE, ties="drop")
plot(wr_result,effect_neutral_point = 1,pairs_tie = "drop")
```

`remapcap` *A reconstructed dataset from the REMAP-CAP clinical trial by Angus et al. (2020)*

Description

A reconstructed dataset from the REMAP-CAP clinical trial by Angus et al. (2020)

Usage

```
remapcap
```

Format

A data frame with 379 rows and 6 variables:

group Type of treatment received

deathTime Time to death in days

dischargeTime Time to discharge in days

deathStatus If patient died within 21 days of follow-up (1=Yes; 0=No)

dischargeStatus If patient was discharged within 21 days of follow-up (1=Yes; 0=No)

score Number of organ support-free days (if patient died, assigned value of -1)

References

Angus, Derek C., et al. "Effect of hydrocortisone on mortality and organ support in patients with severe COVID-19: the REMAP-CAP COVID-19 corticosteroid domain randomized clinical trial." Jama 324.13 (2020): 1317-1329.

---

tte_covariates *Simulated time-varying covariates*

Description

This is a reformatted version of the simulated datasets provided by Z_t_trt and Z_t_con in the WINS package. It provides an example of data in time-varying covariates in a separate dataset to outcome variables, mimicking a common data structure in clinical trials where outcomes and data collected at a single time point are found in one dataset and repeating measures are found in a second dataset. It is intended to be used in combination with tte_outcomes

The ID variable has been reworked so that each ID corresponds to a different patient (i.e. there are not two patients with an ID of "1")

Usage

```
tte_covariates
```

Format

A data frame with 400 obeservations on the following 9 variables:

id The patient ID

time The time of observation

Z1 The first covariate

Z2 The second covariate

tte_longform *Simulated time to event data in long form*

Description

This is a reformatted version of the simulated dataset provided by data_tte, Z_t_con and Z_t_trt in the WINS package, with a stratification variable added. It provides an example of data in time-series format. The ID variable has been reworked so that each ID corresponds to a different patient (i.e. there are not two patients with an ID of "1")

Usage

```
tte_longform
```

Format

A data frame with 2420 obeservations on the following 9 variables:

id The patient ID

arm The treatment groups

time The time of observation. Used for both time-varying covariates and time to event outcomes

Z1 The first covariate

Z2 The second covariate

event_1 Event status for the first outcome. 0=censored at this time, 1=event at this time, NA = neither

event_2 Event status for the second outcome. 0=censored at this time, 1=event at this time, NA = neither

event_3 Event status for the third outcome. 0=censored at this time, 1=event at this time, NA = neither

strata The stratum for each patient

tte_outcomes *Simulated time to event data*

Description

This is a simulated dataset provided by data_tte in the WINS package, with a stratification variable added. It provides an example of outcome data in time-series format.

The ID variable has been reworked so that each ID corresponds to a different patient (i.e. there are not two patients with an ID of "1")

Usage

```
tte_outcomes
```

Format

A data frame with 400 obeservations on the following 9 variables:

id The patient ID

arm The treatment groups

Delta_1 The time event or censoring for the first outcome.

Delta_2 The time event or censoring for the second outcome.

Delta_3 The time event or censoring for the third outcome.

Y_1 Event status for the first outcome. 0=censored at this time, 1=event at this time, NA = neither

Y_2 Event status for the second outcome. 0=censored at this time, 1=event at this time, NA = neither

Y_3 Event status for the third outcome. 0=censored at this time, 1=event at this time, NA = neither

strata The stratum for each patient

---

w_arm *Identify group variable*

Description

This is a special function used by winnify and supplied wrapper functions. It identifies which variable corresponds to the group

Usage

```
w_arm(x)
```

Arguments

x	A term in a formula

Details

When used outside of a formula, this is an identity function.

Value

Whatever was supplied in x

w_id *Identify patient ID variable*

Description

This is a special function used by winnify and supplied wrapper functions. It identifies which variable corresponds to the patient ID

Usage

```
w_id(x)
```

Arguments

x	A term in a formula

Details

When used outside of a formula, this is an identity function.

Value

Whatever was supplied in x

w_iptw *Identify IPTW variable*

Description

This is a special function used by winnify and supplied wrapper functions. It identifies which variable corresponds to inverse probability of treatment weight

Usage

```
w_iptw(x)
```

Arguments

x	A term in a formula

Details

When used outside of a formula, this is a identity function.

Value

Whatever was supplied in x

w_stratum *Identify stratum variable*

Description

This is a special function used by winnify and supplied wrapper functions. It identifies which variable corresponds to the strata

Usage

```
w_stratum(x)
```

Arguments

x	A term in a formula

Details

When used outside of a formula, this is a identity function.

Value

Whatever was supplied in x

w_time *Identify time variable*

Description

This is a special function used by winnify and supplied wrapper functions. It identifies which variable corresponds to time

Usage

```
w_time(x)
```

Arguments

x	A term in a formula

Details

When used outside of a formula, this is a identity function.

Value

Whatever was supplied in x

winnify *Better Prioritised Outcomes Analyses*

Description

winnify is a wrapper function that facilitates improved reporting of Generalised Pairwise Comparisons with Prioritised Outcomes e.g. Win Ratio (Pocock et al. 2012), Win Odds (Dong et al. 2020) and Net Benefit (Buyse et al. 2010).

It takes a formula that specifies one or more outcome variables and then iteratively executes an analysis function to estimate a statistic based on the first outcome, then the first and second outcome, and so on. It also estimates the statistic on each outcome independently of all other outcomes to provide additional context.

In doing so, it provides convenient estimates of the statistic as successively less important outcomes are added to the analysis. This enables improved visualisation of results across outcomes and more transparent reporting (Johns et al. 2026).

It also facilitates the use of rank-based statistical functions for Generalised Pairwise Comparisons with Prioritised Outcomes when preferences among pairwise comparisons are fully transitive.

Usage

```
winnify(formula, statistic = winnify::wins, data, ...)

## S3 method for class 'Winnify'
plot(
  x,
  effect_size = NULL,
  label_effect = NULL,
  label_control = NULL,
  label_intervention = NULL,
  pairs_tie = NULL,
  pairs_tie.color = "#222222",
  pairs_tie.linetype = "dashed",
  pairs.fill = NULL,
  pairs.fill.reverse = NULL,
  pairs_component.width = 1,
  pairs_overall.width = 0.6,
  effect_scale_log = FALSE,
  effect_neutral_point = NULL,
  effect_neutral_line.color = "#222222",
  effect_background_loss.fill = NULL,
  effect_background_win.fill = NULL,
  effect_background.alpha = 0.15,
  effect_individual_color = "grey40",
  effect_cumulative.color = "black",
  effect_cumulative.dodge_width = 0.1,
  effect_cumulative_polygon.height = 0.36,
```

```
  effect_cumulative.linetype = "dashed",
  ...
)
```

Arguments

formula	An object of class formula describing the analyses to run. Details of model specification are given in Details
statistic	A function for performing analysis based on a formula and data. Details are given in Details.
data	A data.frame to analyse
...	additional arguments
x	a Winnify object returned by winnify() or as_winnify()
effect_size	a character string or integer specifying which effect size to plot. Only relevant when x reports multiple effect sizes.
label_effect	a character string for labelling the effect size. Inferred from x and effect_size if possible.
label_control	a character string indicating the name of the control group
label_intervention	a character string indicating the name of the intervention/comparator group
pairs_tie	a character string indicating where the line splitting ties in the Overall row of the pairs plot should be drawn. If NA, no line is drawn. See Details. Inferred from x if possible.
pairs_tie.color	a character string indicating the color of the line splitting ties
pairs_tie.linetype	a character string indicating the linetype of the line splitting ties
pairs.fill	a character string or character vector indicating the color scheme to use for the pairs plot. See Details.
pairs.fill.reverse	a logical indicating if the color scheme provided in pairs.fill should be reversed. See Details.
pairs_component.width	a number indicating the width of the component bars in the pairs plot.
pairs_overall.width	a number indicating the width of the overall bar in the pairs plot.
effect_scale_log	a logical indicating if the effect size should be shown on a log scale
effect_neutral_point	a numeric value indicating the neutral point for the effect plot. Inferred from x if possible.
effect_neutral_line.color	a character string indicating the color of the neutral line in the effect plot.

`effect_background_loss.fill`
: a character string indicating the color of the background corresponding to favoring control in the effect plot. By default, inferred from `pairs.fill`

`effect_background_win.fill`
: a character string indicating the color of the background corresponding to favoring intervention in the effect plot. By default, inferred from `pairs.fill`

`effect_background.alpha`
: a numeric value indicating the opacity (alpha) of the background in the effect plot

`effect_individual_color`
: a character string indicating the color of the individual treatment effects in the effect plot

`effect_cumulative.color`
: a character string indicating the color of the cumulative treatment effects in the effect plot

`effect_cumulative.dodge_width`
: a number indicating the dodging width for the cumulative effect in the effect plot

`effect_cumulative_polygon.height`
: a number indicating the height of the final diamond showing the overall effect

`effect_cumulative.linetype`
: = a character string indicating the linetype for the line connecting cumulative effects. `NA` removes this line.

Details

The left hand side of the formula should list all outcomes to consider, in order of decreasing importance from the left. The right hand side of the formula lists all independent variables. The following special functions are used to indicate fields of interest:

- `w_arm(x)` - marks `x` as being the grouping variable. Is mandatory in the formula.
- `w_id(x)` - marks `x` as being the ID variable
- `w_stratum(x)` - marks `x` as being the stratum variable
- `w_time(x)` - marks `x` as being the time variable for time-varying covariates.
- `w_iptw(x)` - marks `x` as being a weight variable for use with iptw weighting

`w_arm(x)` is needed for both the supplied wins and ranks functions, while `w_id(x)`, `w_stratum(x)` and `w_time(x)` are only accepted by wins.

`pairs_tie` controls how splitting of ties in the final overall value are visualised. There are three options for this:

- `"drop"` - ties are split to respect the final ratio of wins to losses. This is consistent with the Win Ratio
- `"split"` - ties are split evenly between wins and losses. This is consistent with the Win Odds
- `NA` - no line is drawn

`pairs.fill` can either be a character vector of length 5 or the name of a ColorBrewer palette (see <https://colorbrewer2.org>).

If a character vector is supplied, it should contain the following named values:

- `"loss_inherited"` - the fill color for losses inherited from the previous level
- `"loss"` - the fill color for losses new to the current level
- `"neutral"` - the fill color for neutral/tied pairs
- `"win"` - the fill color for wins new to the current level
- `"win_inherited"` - the fill color for wins inherited from the previous level

If the vector is unnamed, names will be assumed based on these positions.

If a ColorBrewer pallette is provided, a palette of length 5 is used. This pallette is reordered such that the sequence of colors is 2,1,3,5,4. This allocates the extremes of the colour pallette to new wins and losses. `pairs.fill.reverse` can be used to reverse the overall color sequence for a ColorBrewer pallette and is otherwise ignored.

Value

An object of class Winnify

References

Johns, Hannah, et al. "Practical guidance for Win Statistics and Tournament Methods for multi-faceted outcomes in stroke research: Review and recommendations." International Journal of Stroke (2026) DOI: https://doi.org/10.1177/17474930261475853

Examples

```
library(genodds)

my_statistic <- function(formula,data,...){
        lhs <- stats::model.frame(formula[1:2],data=data)
        rhs <- stats::model.frame(formula[c(1,3)],data=data)

        if(ncol(lhs)!=1) stop("Left hand side of formula should have one term")
        if(ncol(rhs)!=1) stop("Left hand side of formula should have one term")

        fit <- genodds(response = lhs[,1],
                       group = rhs[,1],
                        ...)

        # genodds returns an object of class Genodds.
        # if it didn't, we would need to give it a class
        # we can write write S3 methods for.
        # class(fit) <- c("MyRankStat",class(fit))
        fit
}

# There's no coef() or confint() functions for
# class Genodds. We need to create some S3 methods to
```

```
# extract this info.
coef.Genodds <- function(x){x$pooled_lnodds}
confint.Genodds <- function(x){matrix(x$pooled_lnconf.int,nrow=1)}

result <- winnify(mRS+early_neuro_improvement~w_arm(treat),
                  statistic = ranks,
                  ranks_statistic = my_statistic,
                  data = neuro_improvement)

plot(result,effect_neutral_point = 1,effect_scale_log = FALSE)
```

`wins` *Wrapper for the win.stat() function from the WINS package*

Description

A wrapper function for win.stat() function, allowing it to be used with winnify(). It also provides several quality of life improvements to the function's interface, including specifying variables using a formula instead of requiring the data to have specifically named columns.

Usage

```
wins(
  formula,
  data,
  data_covariates = NULL,
  effect.size = c("Win Ratio", "Win Odds", "Net Benefit"),
  alpha = 0.05,
  tau = 0,
  np.direction = "larger",
  horizon = Inf,
  win.strategy = NULL,
  pvalue = c("two-sided", "one-sided"),
 stratum.weight = c("unstratified", "MH-type", "wt.stratum1", "wt.stratum2", "equal"),
  method = c("unadjusted", "ipcw_tau", "ipcw", "covipcw", "iptw", "iptw_ipcw",
    "iptw_covipcw"),
  ...
)
```

Arguments

`formula`	An object of class formula describing the analyses to run. Format for model specification are given in Details
`data`	A data.frame to analyse
`data_covariates`	An optional data.frame containing covariates for inverse probability of censoring weighting

effect.size	A character vector indicating which effect size measures to return. Allows partial matches with abbreviations, e.g. "wr" will match to "Win Ratio"
alpha	A numeric argument giving the significance level. See win.stat
tau	A numeric vector argument giving the threshold for minimal clinically important difference. win.stat
np.direction	A chacaracer vector indicating if a larger or smaller value is better. See the np_direction argument in win.stat
horizon	A numeric argument giving Time horizon of interest. See win.stat
win.strategy	A custom function for determining wins, losses and ties. See win.stat
pvalue	A numeric value giving the significance level. See win.stat
stratum.weight	A character value giving the method for weighting each stratum. See win.stat
method	A character value giving the method for weighting individual participants. See win.stat
...	Arguments passed from user defined functions for win.strategy. See win.stat

Details

Note that this wrapper uses several different defaults to the win.stat function which either match common practice in other functions, or infer intent based on user input. These differences are:

- The default p-value is "two-sided", not "one-sided"
- If unspecified, stratum.weight is inferred based on the presence of stratum variables:
  - If no stratum variable is specified in formula, the default stratum.weight is "unstratified".
  - If a stratum variable is specified in formula, the default is "MH-type".
- If unspecified, the default value for method is inferred:
  - If iptw.weight is not provided:
    - If formula doesn't contain any covariates, the default is "unadjusted"
    - If formula contains covariates, the default is "covipcw"
  - If iptw.weight is provided:
    - If formula doesn't contain any covariates, the default is "iptw"
    - If formula contains covariates, the default is "iptw_covipcw"
- Comparator vs Reference group is determined based on factor levels, with the first level used as the reference group and the second as the comparator.
- Rows of input data can be ordered arbitrarily (they are sorted by arm, id and time before being passed to win.stat())

The wrapper does not provide arguments for specifying outcome priority or endpoint type. These values are inferred from user input.

The left hand side of formula provides each endpoint as successive terms, with the most important endpoint on the left and the least important on the right. Values for np.direction and tau correspond to each of these terms in this order.

The right hand side of formula provides grouping variable, ID variable, strata, time and any covariates used for inverse-probability of censoring weighting. These are marked using the following functions:

- `w_arm(x)` - marks `x` as being the grouping variable. Is mandatory in the formula.
- `w_id(x)` - marks `x` as being the ID variable
- `w_stratum(x)` - marks `x` as being the stratum variable
- `w_time(x)` - marks `x` as being the time variable for time-varying covariates.
- `w_iptw(x)` - marks `x` as being the IPTW weight variable

All unmarked terms are assumed to be covariates used for estimating inverse-probability of censoring weighting.

Value

A three-row data frame containing the following:

effect_size The name of the effect size

estimate The estimate from `win.stat` function

lower The lower bound of the condifence interval estimate from `win.stat` function

upper The upper bound of the confidence interval estimate from `win.stat` function

pval The p-value from the `win.stat` function

win The number of wins

loss The number of losses

tie The number of ties

neutral_point The effect size estimate considered to be a neutral effect

tie_handling If ties are evenly split between being wins and losses ("split") or ignored ("dropped")

Additionally, the following attributes are returned

arguments Input values for the function

arguments_win.stat Input values for the `win.stat()` function call within this wrapper function

Examples

```
library(WINS)
library(survival)

# Using time-to-event data from the WINS package.
# winnify() expects a single data frame.

df <- data_tte
df$strata <- rbinom(n=nrow(df),prob = 0.5,size = 1)
df1 <- Z_t_trt
df2 <- Z_t_con
df$id <- paste0(df$arm,df$id)
df1$id <- paste0("A",df1$id)
df2$id <- paste0("B",df2$id)
df$time <- 0
data <- merge(
        rbind(
```

```
                df1,
                df2
        ),
        df,
        all.x = TRUE
)

## Use of wins() by itself

formula <- Surv(Y_1,Delta_1)+Surv(Y_2,Delta_2)+Surv(Y_3,Delta_3)~
           w_arm(arm) + w_stratum(strata) + w_id(id) + w_time(time) + Z1 + Z2

wins(formula,data)

## Use of wins() in a winnify() call

    x <- winnify(formula,wins,data, tau=c(1,0.5,0))
    plot(x, effect_size = "wr")
    plot(x, effect_size = "wo")
    plot(x, effect_size = "nb")
```

Index

∗ **datasets**
    neuro_improvement, 6
    remapcap, 9
    tte_covariates, 10
    tte_longform, 11
    tte_outcomes, 11

as_winnify, 2, *16*
as_winnify.data.frame, *4*
as_winnify.WinRatio, *4*

data_tte, *11*

neuro_improvement, 6

plot.Winnify, *4*
plot.Winnify *(*winnify*)*, 15
polr, *7*

ranks, 6, *17*
remapcap, 9

tte_covariates, 10
tte_longform, 11
tte_outcomes, *10*, 11

w_arm, 12
w_id, 13
w_iptw, 13
w_stratum, 14
w_time, 14
win.stat, *20*
Winnify, *2*
winnify, *4*, *7*, 15, *16*
WinRatio, *2*
winratio, *2*, *4*
wins, *17*, 19

Z_t_con, *10, 11*
Z_t_trt, *10, 11*