Package: ipw (via r-universe)

November 3, 2026

Version 1.3.2.9000

Date 2026-10-31

Title Estimate Inverse Probability Weights

Depends R (>= 3.2.0)

Imports MASS, nnet, survival, geepack, graphics, stats

Suggests knitr, rmarkdown, mgcv, nlme, survey, boot, testthat (>= 3.0.0)

VignetteBuilder knitr

Description Functions to estimate the probability to receive the observed treatment, based on individual characteristics. The inverse of these probabilities can be used as weights when estimating causal effects from observational data via marginal structural models. Both point treatment situations and longitudinal studies can be analysed. The same functions can be used to correct for informative censoring.

Encoding UTF-8

License MIT + file LICENSE

NeedsCompilation no

LazyData true

Config/testthat/edition 3

URL https://tranhung93.github.io/ipw/, https://github.com/TranHung93/ipw

BugReports https://github.com/TranHung93/ipw/issues

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libicu-dev

Repository https://tranhung93.r-universe.dev

Date/Publication 2026-11-03 18:13:12 UTC

RemoteUrl https://github.com/tranhung93/ipw

RemoteRef HEAD

RemoteSha bdecdc3cc1d6985b2cf4717e9bae710d8581fffc

Contents

basdat . . . 2
haartdat . . . 3
healthdat . . . 4
ipwboot . . . 5
ipwplot . . . 8
ipwpoint . . . 9
ipwprob . . . 12
ipwtm . . . 14
timedat . . . 20
tstartfun . . . 21

Index **22**

`basdat` *HIV: TB and Survival (Baseline Data)*

Description

Simulated dataset. Baseline data of 386 HIV positive individuals, including time of first active tuberculosis, time of death, individual end time. Time varying CD4 measurements of these patients are included in dataset `timedat`.

Usage

```
data(basdat)
```

Format

A data frame with 386 observations on the following 4 variables:

id Patient ID.

Ttb Time of first active tuberculosis, measured in days since HIV seroconversion.

Tdeath Time of death, measured in days since HIV seroconversion.

Tend Individual end time (either death or censoring), measured in days since HIV seroconversion.

Details

These simulated data are used together with data in `timedat` in a detailed causal modelling example using inverse probability weighting (IPW). See `ipwtm` for the example. Data were simulated using the algorithm described in Van der Wal e.a. (2009).#' [Image of a multistate model for HIV, TB, and death]

Author(s)

Willem M. van der Wal <willem@vanderwalresearch.com>, Ronald B. Geskus <rgeskus@oucru.org>

References

Van der Wal W.M. & Geskus R.B. (2011). ipw: An R Package for Inverse Probability Weighting. *Journal of Statistical Software*, **43**(13), 1-23. doi:10.18637/jss.v043.i13.

Van der Wal W.M., Prins M., Lumbreras B. & Geskus R.B. (2009). A simple G-computation algorithm to quantify the causal effect of a secondary illness on the progression of a chronic disease. *Statistics in Medicine*, **28**(18), 2325-2337.

See Also

haartdat, ipwplot, ipwpoint, ipwtm, timedat, tstartfun

Examples

```
# Detailed examples can be found in the ipwtm documentation:
# ?ipwtm
```

---

haartdat *HAART and Survival in HIV Patients*

Description

Survival data measured in 1200 HIV positive patients. Start of follow-up is HIV seroconversion. Each row corresponds to a 100 day interval of follow-up time, using the counting process notation. Patients can initiate HAART therapy. CD4 count is a confounder for the effect of HAART on mortality.

Usage

```
data(haartdat)
```

Format

A data frame with 1200 patients and multiple observations per patient (counting process notation) on the following 8 variables:

patient Patient ID.

tstart Starting time for each interval of follow-up, measured in days since HIV seroconversion.

fuptime End time for each interval of follow-up, measured in days since HIV seroconversion.

haartind Indicator for the initiation of HAART therapy at the end of the interval (0 = HAART not initiated / 1 = HAART initiated).

event Indicator for death at the end of the interval (0 = alive / 1 = died).

sex Sex (0 = male / 1 = female).

age Age at the start of follow-up (years).

cd4.sqrt Square root of CD4 count, measured at `fuptime`, before `haartind`.

endtime The final observed time point for the individual.

dropout Indicator for dropout/censoring at the end of the interval (0 = no, 1 = yes).#'

Details

These data were simulated to demonstrate Inverse Probability Weighting (IPW). To allow for models predicting the initiation of HAART at `fuptime = 0`, the starting time for the first interval of each patient is set to -100.

Author(s)

Willem M. van der Wal <willem@vanderwalresearch.com>, Ronald B. Geskus <rgeskus@oucru.org>

References

Van der Wal W.M. & Geskus R.B. (2011). ipw: An R Package for Inverse Probability Weighting. *Journal of Statistical Software*, **43**(13), 1-23. doi:10.18637/jss.v043.i13.

See Also

basdat, ipwplot, ipwpoint, ipwtm, timedat, tstartfun

Examples

```
# For a full example of how to use this data with ipwtm, see:
# ?ipwtm
```

---

healthdat *IQ, Income and Health*

Description

A simulated dataset containing measurements of IQ, monthly income, and health scores for 1000 individuals.

Usage

```
data(healthdat)
```

Format

A data frame with 1000 rows, with each row corresponding to a separate individual. The following 4 variables are included:

id Individual ID.

iq IQ score.

income Gross monthly income (EUR).

health Health score (0-100).

Details

In these simulated data, IQ acts as a confounder for the causal effect of income on health. This dataset is primarily used to illustrate point-treatment inverse probability weighting.

Author(s)

Willem M. van der Wal <willem@vanderwalresearch.com>, Ronald B. Geskus <rgeskus@oucru.org>

References

Van der Wal W.M. & Geskus R.B. (2011). ipw: An R Package for Inverse Probability Weighting. *Journal of Statistical Software*, **43**(13), 1-23. doi:10.18637/jss.v043.i13

See Also

basdat, haartdat, ipwplot, ipwpoint, ipwtm, timedat, tstartfun

Examples

```
# For an example of how to use this data with ipwpoint, see:
# ?ipwpoint
```

---

ipwboot *Bootstrap Standard Errors for Inverse Probability Weighted Analyses*

Description

Bootstraps a whole inverse probability weighted analysis: in each bootstrap sample the weights are re-estimated and the marginal structural model is refitted, so the standard error accounts for the uncertainty in the estimated weights. For data in long format (several rows per subject), whole subjects are resampled and each sampled copy gets a new identifier.

Usage

```
ipwboot(data, statistic, id = NULL, R = 200, strata = NULL, conf = 0.95, ...)

## S3 method for class 'ipwboot'
print(x, digits = 4, ...)

## S3 method for class 'ipwboot'
confint(object, parm, level = object$conf, ...)
```

Arguments

data	Data frame with the original data.
statistic	Function that takes a data frame (a bootstrap sample of data) as its first argument and returns a numeric vector of estimates, for example the coefficients of a weighted outcome model. It should compute the weights itself, e.g. by calling ipwpoint or ipwtm.
id	Optional name of the subject identifier column (unquoted or as a string). If given, subjects are resampled with all their rows, and in each bootstrap sample the column is replaced by new identifiers 1, 2, ..., one per sampled copy, so that a subject drawn twice counts as two subjects. If NULL, rows are resampled.
R	Number of bootstrap samples. About 200 are usually enough for a standard error; use more (e.g. 1000) for percentile confidence intervals.
strata	Optional name of a column (unquoted or as a string) defining strata; resampling is then done within each stratum, keeping the number of subjects (or rows) per stratum fixed. With id, the column must be constant within each subject.
conf	Confidence level of the percentile intervals.
...	Further arguments passed to statistic; for the methods, ignored.
x, object	An object of class "ipwboot".
digits	Number of significant digits to print.
parm	Estimates (names or positions) for which to give intervals; all by default.
level	Confidence level.

Details

Austin (2016) found in simulations that, for weighted Cox models, naive standard errors were too small and robust (sandwich) standard errors too large, while the bootstrap in which the weights are re-estimated in each sample gave correct standard errors and confidence intervals. Austin studied a point treatment with baseline confounders; for longitudinal data the same idea applies with subjects as the resampling unit.

The package's simulation study (article "Simulation study" on the package website) shows when the bootstrap helps. For a point treatment with good overlap, the bootstrap standard error was accurate and robust standard errors were about 12% too large. For a weighted Cox model with good overlap, naive, robust and bootstrap standard errors were all close to correct. When some weights were extreme, through poor overlap or through products of time-varying weights, all standard errors were too small, the bootstrap ones included, and 95% confidence intervals covered the truth in only 57 to 80% of samples. The bootstrap does not repair unstable weights: check the weights first with summary() and ipwplot, and consider truncation, overlap weights or a different exposure model.

Bootstrap samples in which statistic fails, or returns a result of a different length, are left out of the summaries; their number is reported. Use set.seed() before calling ipwboot to make the results reproducible.

Value

A list of class "ipwboot" containing:

t0	Estimates from the original data.
t	Matrix of bootstrap estimates, one row per bootstrap sample (NA for failed samples).
se	Bootstrap standard errors.
ci	Percentile confidence intervals.
R, conf	Number of bootstrap samples and confidence level.
n.fail	Number of failed bootstrap samples.
errors	Distinct error messages of failed samples.
id, strata	Names of the identifier and strata columns, if used.
call	The original function call.

Author(s)

Hung Thai Tran <hungtt@oucru.org>

References

Austin, P.C. (2016). Variance estimation when using inverse probability of treatment weighting (IPTW) with survival analysis. *Statistics in Medicine*, **35**, 5642-5655. doi:10.1002/sim.7084.

See Also

ipwpoint, ipwtm.

Examples

```
# Point treatment: true causal effect of a on y is 10
set.seed(16)
n <- 500
simdat <- data.frame(l = rnorm(n, 10, 5))
simdat$a <- rbinom(n, 1, plogis(simdat$l - 10))
simdat$y <- 10 * simdat$a + 0.5 * simdat$l + rnorm(n, -10, 5)

msm.effect <- function(d) {
  w <- ipwpoint(exposure = a, family = "binomial", link = "logit",
                numerator = ~ 1, denominator = ~ l, data = d)$ipw.weights
  coef(lm(y ~ a, data = d, weights = w))["a"]
}
set.seed(1)
res <- ipwboot(simdat, msm.effect, R = 50)  # use about 200 in practice
res
confint(res)


# Time-varying treatment: resample patients, re-estimate both weights
data(haartdat)
haart.effect <- function(d) {
  w1 <- ipwtm(exposure = haartind, family = "survival", numerator = ~ sex + age,
              denominator = ~ sex + age + cd4.sqrt, id = patient,
```

```
            tstart = tstart, timevar = fuptime, type = "first", data = d)
  w2 <- ipwtm(exposure = dropout, family = "survival", numerator = ~ sex + age,
            denominator = ~ sex + age + cd4.sqrt, id = patient,
            tstart = tstart, timevar = fuptime, type = "cens", data = d)
  fit <- survival::coxph(survival::Surv(tstart, fuptime, event) ~ haartind,
                         data = d, weights = w1$ipw.weights * w2$ipw.weights)
  coef(fit)
}
set.seed(2)
ipwboot(haartdat, haart.effect, id = patient, R = 20)  # use about 200 in practice
```

`ipwplot` *Plot Inverse Probability Weights*

Description

Plots the distribution of inverse probability weights. Displays a density plot for point treatment weights or boxplots across follow-up intervals for time-varying weights.

Usage

```
ipwplot(
  weights,
  timevar = NULL,
  binwidth = NULL,
  logscale = TRUE,
  xlab = NULL,
  ylab = NULL,
  main = "",
  ref = TRUE,
  ...
)
```

Arguments

`weights`	Numerical vector of inverse probability weights.
`timevar`	Optional numerical vector of follow-up time.
`binwidth`	Interval width for grouping follow-up time when `timevar` is supplied.
`logscale`	Logical; if `TRUE`, weights are plotted on a log scale. Default is `TRUE`.
`xlab`	Horizontal axis label.
`ylab`	Vertical axis label.
`main`	Main plot title.
`ref`	Logical; if `TRUE`, adds a reference line at 1 (or log(1) = 0).
`...`	Additional graphical parameters passed to `boxplot` or `plot`.

Author(s)

Willem M. van der Wal <willem@vanderwalresearch.com>, Ronald B. Geskus <rgeskus@oucru.org>

Examples

```
# Point treatment weights density plot
set.seed(123)
w_point <- runif(500, 0.5, 4)
ipwplot(weights = w_point, logscale = FALSE, main = "Density Plot of Point Treatment Weights")

# Time-varying weights boxplot across follow-up strata
w_tv <- runif(1100, 0.2, 5)
time_tv <- rep(seq(0, 500, by = 50), each = 100)
ipwplot(weights = w_tv, timevar = time_tv, binwidth = 50, main = "Time-Varying Weights Boxplot")
```

---

ipwpoint *Estimate Inverse Probability Weights (Point Treatment)*

Description

Estimates inverse probability weights to fit marginal structural models in a point treatment situation. Exposure can be binomial, multinomial, ordinal, a count (Poisson or negative binomial) or continuous (gaussian, gamma or log-normal). Both stabilized and unstabilized weights can be estimated, and for a binomial exposure also weights for the effect in the treated (ATT) and overlap weights (ATO). Formulas may contain smooth terms such as `s(x)`, which are then fitted as generalized additive models with **mgcv**.

Usage

```
ipwpoint(
  exposure,
  family,
  link = NULL,
  numerator = NULL,
  denominator,
  data,
  trunc = NULL,
  estimand = c("ATE", "ATT", "ATO"),
  ...
)
```

Arguments

`exposure`	Name of the exposure column in `data` (unquoted or as a string). Binomial exposure variables should be coded as 0/1.

family	Model family for exposure: "binomial", "multinomial", "ordinal", "gaussian", "poisson", "negbin" (negative binomial, fitted with MASS::glm.nb), "gamma" (gamma GLM with log link) or "lognormal" (normal model for the log of the exposure). Count exposures must be non-negative integers; gamma and log-normal exposures must be positive.
link	Link function for binomial ("logit", "probit", "cauchit", "log", "cloglog") or ordinal ("logit", "probit", "cauchit", "cloglog"). Ignored for multinomial and gaussian.
numerator	Formula specifying the right-hand side of the numerator model. If NULL, unstabilized weights (numerator = 1) are calculated. Required for the continuous families ("gaussian", "gamma", "lognormal"); must be NULL when estimand is "ATT" or "ATO".
denominator	Formula specifying the right-hand side of the denominator model. Smooth terms (s(), te(), ti(), t2()) are allowed for the binomial, gaussian and poisson families; the model is then fitted with mgcv::gam.
data	Data frame containing exposure and model covariates.
trunc	Optional truncation percentile between 0 and 0.5 (e.g., 0.01 for 1st/99th percentiles).
estimand	Target estimand for a binomial exposure: "ATE" (default; inverse probability weights), "ATT" (treated get weight 1, untreated $p/(1-p)$) or "ATO" (overlap weights: treated $1-p$, untreated p), where p is the fitted probability of exposure from the denominator model.
...	Additional arguments passed to model fitting functions (glm, multinom, polr, glm.nb, gam).

Details

Standard errors: ipwpoint only estimates the weights. Standard errors for the marginal structural model have to be obtained when that model is fitted. Robust (sandwich) standard errors, for example from svyglm() in the survey package as in Van der Wal and Geskus (2011), treat the weights as known and tend to be conservative (too large). Austin (2016) compared naive, robust and bootstrap standard errors in simulations and recommends the bootstrap, in which the weights are re-estimated in each bootstrap sample. This takes the uncertainty in the estimated weights into account; see the bootstrap example below, or use ipwboot, which does this for you. About 200 bootstrap samples are usually enough to estimate a standard error. In the package's simulation study (article "Simulation study" on the package website), the bootstrap standard error was accurate for a point treatment with good overlap, while robust standard errors were about 12% too large. When some weights are extreme (poor overlap), no standard error was reliable, the bootstrap included. Check the weights first with summary() and ipwplot, and consider truncation (trunc) or overlap weights (estimand = "ATO").

Value

A list of class "ipwpoint" containing:

ipw.weights	Numeric vector of inverse probability weights.
weights.trunc	Numeric vector of truncated weights (if trunc is specified).

call	The original function call.
num.mod	Numerator model object (if numerator is specified).
den.mod	Denominator model object.

Author(s)

Willem M. van der Wal <willem@vanderwalresearch.com>, Ronald B. Geskus <rgeskus@oucru.org>, Hung Thai Tran <hungtt@oucru.org>

References

Cole, S.R. & Hernán, M.A. (2008). Constructing inverse probability weights for marginal structural models. *American Journal of Epidemiology*, **168**(6), 656-664.

Robins, J.M., Hernán, M.A. & Brumback, B.A. (2000). Marginal structural models and causal inference in epidemiology. *Epidemiology*, **11**, 550-560.

Van der Wal W.M. & Geskus R.B. (2011). ipw: An R Package for Inverse Probability Weighting. *Journal of Statistical Software*, **43**(13), 1-23. doi:10.18637/jss.v043.i13.

Austin, P.C. (2016). Variance estimation when using inverse probability of treatment weighting (IPTW) with survival analysis. *Statistics in Medicine*, **35**, 5642-5655. doi:10.1002/sim.7084.

See Also

basdat, haartdat, ipwplot, ipwpoint, ipwtm, timedat, tstartfun.

Examples

```
# Simulate data with continuous confounder and outcome, binomial exposure.
# Marginal causal effect of exposure on outcome: 10.
set.seed(16)
n <- 1000
simdat <- data.frame(l = rnorm(n, 10, 5))
a.lin <- simdat$l - 10
pa <- exp(a.lin)/(1 + exp(a.lin))
simdat$a <- rbinom(n, 1, prob = pa)
simdat$y <- 10*simdat$a + 0.5*simdat$l + rnorm(n, -10, 5)
simdat[1:5,]

# Estimate ipw weights
temp <- ipwpoint(
  exposure = a,
  family = "binomial",
  link = "logit",
  numerator = ~ 1,
  denominator = ~ l,
  data = simdat)
summary(temp$ipw.weights)

# Plot inverse probability weights
ipwplot(weights = temp$ipw.weights, logscale = FALSE,
        main = "Stabilized weights", xlim = c(0, 8))
```

```
# Examine numerator and denominator models
summary(temp$num.mod)
summary(temp$den.mod)

# Paste inverse probability weights
simdat$sw <- temp$ipw.weights

# Marginal structural model for the causal effect of a on y
# corrected for confounding by l using inverse probability weighting
if (requireNamespace("survey", quietly = TRUE)) {
  library(survey)
  msm <- svyglm(y ~ a,
                design = svydesign(~1, weights = ~temp$ipw.weights,
                data = simdat))
  summary(msm)
}

# Bootstrap standard error and percentile confidence interval for the MSM.
# The robust standard error from svyglm() treats the weights as known and
# tends to be conservative. Austin (2016) recommends the bootstrap, with the
# weights re-estimated in each bootstrap sample: each bootstrap sample below
# re-fits ipwpoint() and the MSM. (A weighted lm() gives the same point
# estimate as svyglm() and is faster.)
if (requireNamespace("boot", quietly = TRUE)) {
  boot.fun <- function(data, index) {
    d <- data[index, ]
    w <- ipwpoint(
      exposure = a,
      family = "binomial",
      link = "logit",
      numerator = ~ 1,
      denominator = ~ l,
      data = d)
    d$sw <- w$ipw.weights
    coef(lm(y ~ a, data = d, weights = sw))["a"]
  }
  set.seed(1)
  bootres <- boot::boot(data = simdat, statistic = boot.fun, R = 200)
  print(bootres)
  # bootstrap standard error
  print(sd(bootres$t[, 1]))
  # percentile 95 percent confidence interval
  print(boot::boot.ci(bootres, type = "perc"))
}
```

---

ipwprob *Inverse Probability Weights From User-Supplied Probabilities*

Description

Computes inverse probability weights from probabilities (or densities) of the observed exposure that you estimated yourself, for example with a machine-learning model or SuperLearner. Point treatment weights are `num / den`; with `id` and `timevar`, the ratios are multiplied cumulatively over time within each subject, as in `ipwtm`.

Usage

```
ipwprob(den, num = NULL, id = NULL, timevar = NULL, trunc = NULL)
```

Arguments

`den`	Numeric vector: for each row, the probability (or density) of the observed exposure given the covariates of the denominator model.
`num`	Optional numeric vector of the same length: the probability (or density) of the observed exposure given the covariates of the numerator model. If `NULL`, unstabilized weights (numerator = 1) are computed.
`id`	Optional vector of subject identifiers, same length as `den`. When given, weights are cumulated over time within each subject.
`timevar`	Numeric vector of follow-up times, same length as `den`; required when `id` is given. It sets the order in which the ratios are multiplied within a subject.
`trunc`	Optional truncation percentile between 0 and 0.5 (e.g., 0.01 for 1st/99th percentiles).

Details

For time-varying weights of `type = "first"` or `"cens"` (see `ipwtm`), set the probabilities to 1 for the rows after the first switch, so that those rows do not change the cumulative weight.

Value

A list of class `"ipwprob"` containing:

`ipw.weights`	Numeric vector of inverse probability weights, in the order of the input.
`weights.trunc`	Numeric vector of truncated weights (if `trunc` is specified).
`call`	The original function call.

Author(s)

Hung Thai Tran <hungtt@oucru.org>

See Also

`ipwpoint`, `ipwtm`.

Examples

```
# Point treatment: probabilities from any model, here a logistic regression
set.seed(1)
d <- data.frame(l = rnorm(200))
d$a <- rbinom(200, 1, plogis(d$l))
p <- fitted(glm(a ~ l, family = binomial, data = d))
w <- ipwprob(den = ifelse(d$a == 1, p, 1 - p),
             num = ifelse(d$a == 1, mean(d$a), 1 - mean(d$a)))
summary(w$ipw.weights)

# Time-varying: the ratios are multiplied within each subject over time
long <- data.frame(id = rep(1:3, each = 2), time = rep(1:2, 3),
                   p = c(0.5, 0.8, 0.4, 0.9, 0.6, 0.7))
ipwprob(den = long$p, id = long$id, timevar = long$time)$ipw.weights
```

`ipwtm` *Estimate Inverse Probability Weights (Time Varying)*

Description

Estimates inverse probability weights to fit marginal structural models with time-varying exposures and time-varying confounders across follow-up intervals. Supports binomial, survival, multinomial, ordinal, and gaussian families.

Usage

```
ipwtm(
  exposure,
  family = c("binomial", "survival", "multinomial", "ordinal", "gaussian"),
  link = NULL,
  numerator = NULL,
  denominator,
  id,
  tstart,
  timevar,
  type = c("first", "cens", "all"),
  data,
  corstr = "ar1",
  trunc = NULL,
  ...
)
```

Arguments

`exposure`	Name of the exposure column in `data` (unquoted or as a string).
`family`	Model family: `"binomial"`, `"survival"`, `"multinomial"`, `"ordinal"`, or `"gaussian"`.
`link`	Link function for binomial or ordinal models.

numerator	Formula for numerator model. If NULL, unstabilized weights are computed.
denominator	Formula for denominator model. For family = "binomial", smooth terms (s(), te(), ti(), t2()) are allowed; the model is then fitted with mgcv::gam.
id	Name of the subject identifier column in data.
tstart	Name of the interval start time column (required for family = "survival").
timevar	Name of the follow-up (interval end) time column.
type	Exposure switch type: "first", "cens", or "all". "all" is available for all families. With family = "survival" and type = "all", exposure is an event indicator (1 when an exposure event, such as starting treatment, happens at the end of the interval) that may occur more than once per subject: an Andersen-Gill Cox model is fitted on all rows and every row contributes to the weights (Graffeo et al. 2018). For treatment that is started and stopped, compute weights separately for starting (on the rows off treatment) and for stopping (on the rows on treatment).
data	Data frame containing all required variables.
corstr	Correlation structure for GEE when family = "gaussian" (default: "ar1").
trunc	Optional truncation percentile between 0 and 0.5.
...	Additional arguments passed to model fitting engines.

Details

Standard errors: ipwtm only estimates the weights. Standard errors for the marginal structural model have to be obtained when that model is fitted. Robust standard errors, for example from coxph() with cluster(id) as in Van der Wal and Geskus (2011), treat the weights as known and tend to be conservative (too large). Austin (2016) compared naive, robust and bootstrap standard errors in simulations and recommends the bootstrap, in which the weights are re-estimated in each bootstrap sample. Austin studied a point treatment with baseline confounders; for longitudinal data in long format the bootstrap must resample subjects, not rows: keep all rows of a sampled subject, give each sampled copy a new id, re-run ipwtm (for treatment and, if used, censoring weights) on the bootstrap sample and refit the marginal structural model. See the cluster bootstrap in Example 1 below; ipwboot does this resampling for you (argument id). About 200 bootstrap samples are usually enough to estimate a standard error. Time-varying weights are products over time, so a few subjects can get very large weights. In the package's simulation study (500 subjects, 4 visits), bootstrap and robust standard errors were then both too small, with 95% confidence intervals covering the truth in only 72 to 80% of samples. Check the weights with summary() and ipwplot, consider truncation (trunc), and interpret standard errors with care when the weights are extreme.

Value

A list of class "ipwtm" containing:

ipw.weights	Numeric vector of inverse probability weights in original row order.
weights.trunc	Numeric vector of truncated weights (if trunc is specified).
call	Original function call.
selvar	Selection indicator vector (1 = active observation, 0 = post-switch).
num.mod	Numerator model object.
den.mod	Denominator model object.

References

Graffeo N., Latouche A., Geskus R.B. & Chevret S. (2018). Modeling time-varying exposure using inverse probability of treatment weights. *Biometrical Journal*, **60**(2), 323-332. doi:10.1002/bimj.201600223.

Van der Wal W.M. & Geskus R.B. (2011). ipw: An R Package for Inverse Probability Weighting. *Journal of Statistical Software*, **43**(13), 1-23. doi:10.18637/jss.v043.i13.

Austin, P.C. (2016). Variance estimation when using inverse probability of treatment weighting (IPTW) with survival analysis. *Statistics in Medicine*, **35**, 5642-5655. doi:10.1002/sim.7084.

Examples

```
#############################################################################
#EXAMPLE 1

#Load longitudinal data from HIV positive individuals.
data(haartdat)

#CD4 is confounder for the effect of initiation of HAART therapy on mortality.
#Estimate inverse probability weights to correct for confounding.
#Exposure allocation model is Cox proportional hazards model.
temp <- ipwtm(
  exposure = haartind,
  family = "survival",
  numerator = ~ sex + age,
  denominator = ~ sex + age + cd4.sqrt,
  id = patient,
  tstart = tstart,
  timevar = fuptime,
  type = "first",
  data = haartdat)

#plot inverse probability weights
graphics.off()
ipwplot(weights = temp$ipw.weights, timevar = haartdat$fuptime,
      binwidth = 100, ylim = c(-1.5, 1.5), main = "Stabilized inverse probability weights")

#CD4 count has an effect both on dropout and mortality, which causes informative censoring.
#Use inverse probability of censoring weighting to correct for effect of CD4 on dropout.
#Use Cox proportional hazards model for dropout.
temp2 <- ipwtm(
  exposure = dropout,
  family = "survival",
  numerator = ~ sex + age,
  denominator = ~ sex + age + cd4.sqrt,
  id = patient,
  tstart = tstart,
  timevar = fuptime,
  type = "cens",
  data = haartdat)

#plot inverse probability of censoring weights
```

```
graphics.off()
ipwplot(weights = temp2$ipw.weights, timevar = haartdat$fuptime,
        binwidth = 100, ylim = c(-1.5, 1.5),
        main = "Stabilized inverse probability of censoring weights")

#MSM for the causal effect of initiation of HAART on mortality.
#Corrected both for confounding and informative censoring.
#With robust standard error obtained using cluster().
require(survival)
summary(coxph(Surv(tstart, fuptime, event) ~ haartind + cluster(patient),
              data = haartdat, weights = temp$ipw.weights*temp2$ipw.weights))

#uncorrected model
summary(coxph(Surv(tstart, fuptime, event) ~ haartind, data = haartdat))

#Bootstrap standard error for the MSM above (cluster bootstrap by patient).
#cluster() gives a robust standard error that treats the weights as known and
#tends to be conservative. Austin (2016) recommends the bootstrap, with the
#weights re-estimated in each bootstrap sample.
#With longitudinal data, resample patients, not rows: keep all rows of a
#sampled patient, give each sampled copy a new id, re-run both ipwtm() calls
#and refit the MSM.
#B = 20 keeps the example short; use about 200 in practice.

msm.fun <- function(d) {
  tw <- ipwtm(exposure = haartind, family = "survival",
              numerator = ~ sex + age, denominator = ~ sex + age + cd4.sqrt,
              id = patient, tstart = tstart, timevar = fuptime,
              type = "first", data = d)
  cw <- ipwtm(exposure = dropout, family = "survival",
              numerator = ~ sex + age, denominator = ~ sex + age + cd4.sqrt,
              id = patient, tstart = tstart, timevar = fuptime,
              type = "cens", data = d)
  d$w <- tw$ipw.weights * cw$ipw.weights
  coef(coxph(Surv(tstart, fuptime, event) ~ haartind, data = d,
             weights = w))["haartind"]
}
rows.pat <- split(seq_len(nrow(haartdat)), haartdat$patient)
set.seed(123)
B <- 20
est.boot <- numeric(B)
for (b in seq_len(B)) {
  pick <- sample(length(rows.pat), replace = TRUE)
  d.boot <- haartdat[unlist(rows.pat[pick]), ]
  d.boot$patient <- rep(seq_along(pick), lengths(rows.pat[pick]))
  est.boot[b] <- msm.fun(d.boot)
}
#log hazard ratio with bootstrap standard error
c(estimate = unname(msm.fun(haartdat)), boot.se = sd(est.boot))
#percentile interval (crude with B = 20)
quantile(est.boot, c(0.025, 0.975))
```

```
#########################################################################
#EXAMPLE 2

data(basdat)
data(timedat)

#Aim: to model the causal effect of active tuberculosis (TB) on mortality.
#Longitudinal CD4 is a confounder as well as intermediate for the effect of TB.

#process original measurements
#check for ties (not allowed)
table(duplicated(timedat[,c("id", "fuptime")]))
#take square root of CD4 because of skewness
timedat$cd4.sqrt <- sqrt(timedat$cd4count)
#add TB time to dataframe
timedat <- merge(timedat, basdat[,c("id", "Ttb")], by = "id", all.x = TRUE)
#compute TB status
timedat$tb.lag <- ifelse(with(timedat, !is.na(Ttb) & fuptime > Ttb), 1, 0)
#longitudinal CD4-model
require(nlme)
cd4.lme <- lme(cd4.sqrt ~ fuptime + tb.lag, random = ~ fuptime | id,
               data = timedat)

#build new dataset:
#rows corresponding to TB-status switches, and individual end times
times <- sort(unique(c(basdat$Ttb, basdat$Tend)))
startstop <- data.frame(
  id = rep(basdat$id, each = length(times)),
  fuptime = rep(times, nrow(basdat)))
#add baseline data to dataframe
startstop <- merge(startstop, basdat, by = "id", all.x = TRUE)
#limit individual follow-up using Tend
startstop <- startstop[with(startstop, fuptime <= Tend),]
startstop$tstart <- tstartfun(id, fuptime, startstop) #compute tstart (?tstartfun)
#indicate TB status
startstop$tb <- ifelse(with(startstop, !is.na(Ttb) & fuptime >= Ttb), 1, 0)
#indicate TB status at previous time point
startstop$tb.lag <- ifelse(with(startstop, !is.na(Ttb) & fuptime > Ttb), 1, 0)
#indicate death
startstop$event <- ifelse(with(startstop, !is.na(Tdeath) & fuptime >= Tdeath),
                          1, 0)
#impute CD4, based on TB status at previous time point.
startstop$cd4.sqrt <- predict(cd4.lme,
                              newdata = data.frame(id = startstop$id,
                                                   fuptime = startstop$fuptime,
                                                   tb.lag = startstop$tb.lag))
#compute inverse probability weights
temp <- ipwtm(
  exposure = tb,
  family = "survival",
  numerator = ~ 1,
  denominator = ~ cd4.sqrt,
  id = id,
```

```
  tstart = tstart,
  timevar = fuptime,
  type = "first",
  data = startstop)
summary(temp$ipw.weights)
ipwplot(weights = temp$ipw.weights, timevar = startstop$fuptime, binwidth = 100)

#models
#IPW-fitted MSM, using cluster() to obtain robust standard error estimate
require(survival)
summary(coxph(Surv(tstart, fuptime, event) ~ tb + cluster(id),
              data = startstop, weights = temp$ipw.weights))
#unadjusted
summary(coxph(Surv(tstart, fuptime, event) ~ tb, data = startstop))
#adjusted using conditioning: part of the effect of TB is adjusted away
summary(coxph(Surv(tstart, fuptime, event) ~ tb + cd4.sqrt, data = startstop))

## Not run:
#compute bootstrap CI for TB parameter (takes a few hours)
#taking into account the uncertainty introduced by modelling longitudinal CD4
#taking into account the uncertainty introduced by estimating the inverse probability weights
#robust with regard to weights unequal to 1
#  require(boot)
#  boot.fun <- function(data, index, data.tm){
#     data.samp <- data[index,]
#     data.samp$id.samp <- 1:nrow(data.samp)
#     data.tm.samp <- do.call("rbind", lapply(data.samp$id.samp, function(id.samp) {
#       cbind(data.tm[data.tm$id == data.samp$id[data.samp$id.samp == id.samp],],
#         id.samp = id.samp)
#       }
#     ))
#    cd4.lme <- lme(cd4.sqrt ~ fuptime + tb.lag, random = ~ fuptime | id.samp, data = data.tm.samp)
#     times <- sort(unique(c(data.samp$Ttb, data.samp$Tend)))
#    startstop.samp <- data.frame(id.samp = rep(data.samp$id.samp, each = length(times)),
#                                 fuptime = rep(times, nrow(data.samp)))
#     startstop.samp <- merge(startstop.samp, data.samp, by = "id.samp", all.x = TRUE)
#     startstop.samp <- startstop.samp[with(startstop.samp, fuptime <= Tend),]
#     startstop.samp$tstart <- tstartfun(id.samp, fuptime, startstop.samp)
#    startstop.samp$tb <- ifelse(with(startstop.samp, !is.na(Ttb) & fuptime >= Ttb), 1, 0)
#   startstop.samp$tb.lag <- ifelse(with(startstop.samp, !is.na(Ttb) & fuptime > Ttb), 1, 0)
#   startstop.samp$event <- ifelse(with(startstop.samp, !is.na(Tdeath) & fuptime >= Tdeath), 1, 0)
#     startstop.samp$cd4.sqrt <- predict(cd4.lme, newdata = data.frame(id.samp =
#     startstop.samp$id.samp, fuptime = startstop.samp$fuptime, tb.lag = startstop.samp$tb.lag))
#
#     return(coef(coxph(Surv(tstart, fuptime, event) ~ tb, data = startstop.samp,
#        weights = ipwtm(
#             exposure = tb,
#             family = "survival",
#             numerator = ~ 1,
#             denominator = ~ cd4.sqrt,
#             id = id.samp,
#             tstart = tstart,
#             timevar = fuptime,
```

```
#             type = "first",
#             data = startstop.samp)$ipw.weights))[1])
#     }
#  bootres <- boot(data = basdat, statistic = boot.fun, R = 999, data.tm = timedat)
#  bootres
#  boot.ci(bootres, type = "basic")
#
## End(Not run)
```

`timedat` *HIV: TB and Survival (Longitudinal Measurements)*

Description

A simulated dataset containing time-varying CD4 measurements for 386 HIV-positive individuals. Corresponding baseline data, including timing of tuberculosis and death, are available in `basdat`.

Usage

```
data(timedat)
```

Format

A data frame with 6291 observations on the following 3 variables:

id Patient ID.

fuptime Follow-up time (days since HIV seroconversion).

cd4count CD4 count measured at `fuptime`.

Details

These simulated data are used together with `basdat` in a detailed causal modeling example using inverse probability weighting (IPW). See `ipwtm` for the full example. Data were simulated using the algorithm described in Van der Wal et al. (2009).

Author(s)

Willem M. van der Wal <willem@vanderwalresearch.com>, Ronald B. Geskus <rgeskus@oucru.org>

References

Cole, S.R. & Hernán, M.A. (2008). Constructing inverse probability weights for marginal structural models. *American Journal of Epidemiology*, **168**(6), 656-664.

Robins, J.M., Hernán, M.A. & Brumback, B.A. (2000). Marginal structural models and causal inference in epidemiology. *Epidemiology*, **11**, 550-560.

Van der Wal W.M. & Geskus R.B. (2011). ipw: An R Package for Inverse Probability Weighting. *Journal of Statistical Software*, **43**(13), 1-23. doi:10.18637/jss.v043.i13

Van der Wal W.M., Prins M., Lumbreras B. & Geskus R.B. (2009). A simple G-computation algorithm to quantify the causal effect of a secondary illness on the progression of a chronic disease. *Statistics in Medicine*, **28**(18), 2325-2337.

See Also

basdat, haartdat, ipwplot, ipwpoint, ipwtm, tstartfun

Examples

```
# For an example of how to use these longitudinal measurements with basdat, see:
# ?ipwtm
```

---

tstartfun *Compute Starting Time For Counting Process Notation*

Description

Computes starting time for follow-up intervals in longitudinal data using counting process notation. Within each subject, starting time equals -1 for the initial record and the previous record's follow-up time for subsequent records.

Usage

```
tstartfun(id, timevar, data)
```

Arguments

`id`	Patient or unit identifier (unquoted symbol, string, or vector).
`timevar`	Follow-up time variable (unquoted symbol, string, or vector).
`data`	Data frame containing `id` and `timevar`.

Value

Numeric vector of starting times in the same row order as `data`.

Author(s)

Willem M. van der Wal <willem@vanderwalresearch.com>, Ronald B. Geskus <rgeskus@oucru.org>

Examples

```
mydata1 <- data.frame(
  patient = c(1, 1, 1, 1, 1, 1, 2, 2, 2, 2),
  time.days = c(14, 34, 41, 56, 72, 98, 0, 11, 28, 35)
)
mydata1$tstart <- tstartfun(patient, time.days, mydata1)
mydata1
```

Index

∗ **datasets**
basdat, 2
haartdat, 3
healthdat, 4
timedat, 20
∗ **htest**
ipwtm, 14
∗ **models**
ipwtm, 14

basdat, 2, *4, 5*, *11*, *20, 21*

confint.ipwboot *(*ipwboot*)*, 5

haartdat, 3, *3*, *5*, *11*, *21*
healthdat, 4

ipwboot, 5, *10*, *15*
ipwplot, *3–6*, 8, *10, 11*, *15*, *21*
ipwpoint, *3–7*, 9, *11*, *13*, *21*
ipwprob, 12
ipwtm, *2–7*, *11*, *13*, 14, *20, 21*

print.ipwboot *(*ipwboot*)*, 5

timedat, *2–5*, *11*, 20
tstartfun, *3–5*, *11*, 21, *21*