Package 'PropensitySub'

July 29, 2021

Title Treatment Effect Estimate in Strata with Missing Data

Version 0.2.0

Maintainer Heng Wang <wangh107@gene.com>

Description Estimate treatment effect in strata when subjects have missing strata labels, via inverse probability weighting or propensity score matching.

Depends R (>= 3.5.0), survival

Imports Matching, rlang, plyr, nnet, ggplot2, survminer, dplyr, gridExtra, gtable, grid, pROC, scales

Suggests knitr, rmarkdown

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

VignetteBuilder knitr

NeedsCompilation no

Author Ning Leng [aut],
Heng Wang [aut, cre],
Shengchun Kong [aut],
Dominik Heinzmann [aut]

Repository CRAN

Date/Publication 2021-07-29 08:50:11 UTC

R topics documented:

`biomarker` *Biomarker data*

Description

simulated biomarker data

Usage

```
biomarker
```

Format

An object of class `data.frame` with 252 rows and 11 columns.

Source

internal

`bootstrap_propen` *Calculate bootstrap CI for treatment effect estimate*

Description

Calculate bootstrap CI for treatment effect estimate

Usage

```
bootstrap_propen(
  data.in,
  indicator.var = "indicator",
  formula,
  indicator.next = NULL,
  seed = 100,
  class.of.int,
  estimate.res,
  n.boot = 1000,
  method = "ipw",
  wild.boot = FALSE,
  tte = "AVAL",
```

```
  event = "event",
  trt = "trt",
  response = NULL,
  caliper = NULL,
  pairs = NULL,
  hr.ratio.ref = NULL,
  ref.denom = TRUE,
  model = "plain",
  max.num.run = 5000,
  non.converge.check = FALSE,
  multinom.maxit = 100,
  non.converge.check.thresh = 1
)
```

Arguments

data.in	(data.frame) input data
indicator.var	(string) column name of the strata indicator variable which must be numeric. Assume arm1 has strata labeling and arm2 does not have strata labeling. pts without strata labeling should be indicated as -1 (e.g. pts in the arm1, or pts in arm2 but with missing label). within arm1 (the arm with strata labeling), subclasss should be indicated as 0,1,2...
formula	(formula) to input to the logistic or multinomial logistic model (in the form of strata~features)
indicator.next	(string) column name of the column which indicates status at a different measurement. It should be coded in the same way as in indicator.var (e.g. -1, 0, 1). Patients who have both missing current status and missing next status should be excluded in the modeling.
seed	seed
class.of.int	(list) classes (stratum) of interest. Request to be in list format. It could be subset of classes in arm1; it could also define combined classes. For example: class.of.int = list("class1"=0, "class2"=1, "class3"=2, "class2or3"="c(1,2)"). for "class2or3", Prob(class 2 or 3) will be calculated as Prob(class2) + Prob(class3)
estimate.res	result object from ipw_strata() or ps_match_strata()
n.boot	number of bootstraps to run; note only runs without warning/error msg will be considered when calculating bootstrap CI; so it is possible more that n.boot runs are performed (but capped by max.num.run)
method	"ipw" or "ps". If "ipw", ipw_strata() will be used. If "ps", ps_match_strata() will be used.
wild.boot	whether wild bootstrap should be used. If so, weights will be generated using rexp(1)
tte	(string) column name of the time to event variable
event	(string) column name of the event variable (1: event, 0: censor)
trt	(string) column name of the treatment variable. The variable is expected to be in factor format and the first level will be considered as reference (control) arm when calculating summary statistics.

`response` (`string`) column name of the response variable. 1 indicates responder and 0 indicates non responder. if response is not NULL, tte and event will be ignored and the function will assume binary outcome.

`caliper` A scalar or vector denoting the caliper(s) which should be used when matching. A caliper is the distance which is acceptable for any match. Observations which are outside of the caliper are dropped. If a scalar caliper is provided, this caliper is used for all covariates in `X`. If a vector of calipers is provided, a caliper value should be provided for each covariate in `X`. The caliper is interpreted to be in standardized units. For example, `caliper=.25` means that all matches not equal to or within .25 standard deviations of each covariate in `X` are dropped. Note that dropping observations generally changes the quantity being estimated.

`pairs` pairs of interest when calculating ratio of HR (delta of delta for OR). this should be a matrix whose rows are names of strata, 1st column indicates the stratum to be used as numerator (HR or ORR diff); 2nd column indicates denominator. If pairs is NULL, ratio of HR (difference of OR difference) will not be calculated.

`hr.ratio.ref` no longer to be used, please use pairs instead

`ref.denom` no longer to be used, please use pairs instead

`model` (`string`) one of (`plain`, `dwc`, `wri`).

"plain" when 2 levels are specified in indicator variable, a binomial glm will be fitted; when more than 2 levels are specified, a multinomial glm will be fitted;

"dwc" Doubly weighted control: Two separated models will be fitted: one is binomial glm of 2 vs. (1, 0), the other one is bionomial glm of 1 vs 0. The probability of being each class is then calculated by aggregating these two models. Note this is similar to the plain method but with different (less rigid) covariance assumptions.

"wri" Weight regression imputation: the current status is going to be learned from the next status. Indicator of the next status should be specified using indicator.next. Currently "wri" only support the case where there are only two non-missing strata. In indicator variable, the two nonmissing strata should be coded as 0 and 1, the missing group should be coded as 2.

`max.num.run` max number of bootstraps to run (include both valid and not valid runs)

`non.converge.check`
whether to output number of time each level of each categorical variable for each stratum specified in indicator having N<non.converge.check.thresh when non-convergence occurs

`multinom.maxit` see parameter `maxit` in nnet::multinom, default is 100

`non.converge.check.thresh`
see above

Value

return a `list` containing the following components:

- boot.out.est a `matrix` with rows as estimates such as Coefficient and Variance in strata and columns as summary statistics such as Mean and Median of the estimates.

- est.ci.mat a `matrix` with rows as strata and columns as Estimate and Bootstrap CIs.
- eff.comp.ci.mat a `matrix` with rows as strata comparisons and columns as Estimate and Bootstrp CIs.
- conv.est a `logical` vector to indicate whether model in each bootstrap converges.
- error.est `numeric` to indicate the total number of models in bootstrap which gives errors.
- boot.results a `matrix` with rows as each bootstrap and columns as model results such as Coefficient in strata.
- glm.warn.est a `logical` vector to indicate whether glm model gives warning in each bootstrap.
- num.valid.boots `numeric` to indicate the total number of valid bootstraps.
- num.total.boots `numeric` for the total number of bootstrap runs.
- warning.msg a `list` to capture warning messages from models.
- error.msg a `list` to capture error messages from models.
- non.converge.dat a `matrix` with rows as each level of each categorical variable for each stratum specified in indicator having N less than `non.converge.check.thresh` and columns as treatment groups

Note

only estimates from runs without error or warning will be considered when calculating bootstrap CI. If none of the bootstrap runs is error/warning free, CI of est.ci.mat will be NA

Examples

```
library(dplyr)
clinical_1 <- clinical %>% mutate(
  indicator = case_when(
    STRATUM == "strata_1" ~ 0,
    STRATUM == "strata_2" ~ 1,
    is.na(STRATUM) & ARM == "experimental" ~ 1,
    TRUE ~ -1
  ),
  ARM  = factor(ARM, levels = c("control","experimental")),
  BNLR = case_when(
    is.na(BNLR) ~ median(BNLR, na.rm = TRUE),
    TRUE ~ BNLR
  )
)
# ipw: default model
ipw_res <- ipw_strata(
  data.in = clinical_1, formula = indicator ~ BECOG + SEX + BNLR,
  indicator.var = "indicator", tte = "OS_MONTH", event = "OS_EVENT", trt = "ARM",
  class.of.int = list("strata_1" = 1, "strata_2" = 0)
 )
boot_ipw <- bootstrap_propen(
  data.in = clinical_1, formula = indicator ~ BECOG + SEX + BNLR,
  indicator.var = "indicator", tte = "OS_MONTH", event = "OS_EVENT", trt = "ARM",
  class.of.int = list("strata_1" = 1, "strata_2" = 0),
  estimate.res = ipw_res, method = "ipw", n.boot = 5
```

```
)
boot_ipw$est.ci.mat
boot_ipw$boot.out.est
# ps: DWC model
clinical_2 <- clinical %>% mutate(
  indicator = case_when(
    STRATUM == "strata_1" ~ 0,
    STRATUM == "strata_2" ~ 1,
    is.na(STRATUM) & ARM == "experimental" ~ 2,
    TRUE ~ -1
  ),
  ARM  = factor(ARM, levels = c("control","experimental")),
  BNLR = case_when(
    is.na(BNLR) ~ median(BNLR, na.rm = TRUE),
    TRUE ~ BNLR
  )
)
ps_res <- ps_match_strata(
  data.in = clinical_2, formula = indicator ~ BECOG + SEX + BNLR, model = "dwc",
  indicator.var = "indicator", tte = "OS_MONTH", event = "OS_EVENT", trt = "ARM",
  class.of.int = list("strata_1" = 0, "strata_2" = 1, "missing" = 2)
 )
boot_ps <- bootstrap_propen(
  data.in = clinical_2, formula = indicator ~ BECOG + SEX + BNLR, model = "dwc",
  indicator.var = "indicator", tte = "OS_MONTH", event = "OS_EVENT", trt = "ARM",
  class.of.int = list("strata_1" = 0, "strata_2" = 1, "missing" = 2),
  estimate.res = ps_res, method = "ps", n.boot = 5
)
boot_ps$est.ci.mat
boot_ps$boot.out.est
```

`calc_std_diff` *Calculate standardized difference*

Description

Calculate standardized difference

Usage

```
calc_std_diff(vars, data0, weight0, data1, weight1)
```

Arguments

`vars`	variables of interest. standardized difference of each variable listed here will be calculated.
`data0`	A `data.frame` which include `vars` as columns from reference arm. All data are expected to be numerical. If a column is not numerical, it will be turned to numerical by `model.matrix`.

weight0	weights for each record in reference arm.
data1	A data.frame which include vars as columns from comparison arm. All data are expected to be numerical. If a column is not numerical, it will be turned to numerical by model.matrix.
weight1	weights for each record in comparison arm.

Value

return a numeric vector for standardized difference of each variable

Note

Calculation from Austin and Stuart (2015)

Examples

```
library(dplyr)
data0 <- clinical %>% filter(ARM == "experimental")
data1 <- clinical %>% filter(ARM == "control")
calc_std_diff(
  vars = c("BECOG", "SEX"),
  data0 = data0,
  weight0 = rep(1, nrow(data0)),
  data1 = data1,
  weight1 = rep(1, nrow(data1))
 )
```

---

clinical *Clinical data*

Description

simulated data from clinical studies

Usage

```
clinical
```

Format

An object of class data.frame with 652 rows and 12 columns.

Source

internal

`expected_feature_diff` *Expected number of not optimally balanced features as defined by a threshold*

Description

Calculate expected number of features showing balance difference greater than a threshold

Usage

```
expected_feature_diff(n.feature, n.arm1, n.arm2, threshold)
```

Arguments

`n.feature`	(`numeric`) total number of features
`n.arm1`	(`numeric`) number of patients in comparison arm.
`n.arm2`	(`numeric`) number of patients in control arm
`threshold`	(`numeric`) positive number(s) for threshold to compare to.

Value

return a `numeric` vector for expected number of unbalanced features

Note

The output number indicates when running a randomized trial with n.arm1 and n.arm2 samples in two arms and n.feature features are of interest, the expected number of features showing balance difference greater than threshold. p = Prob(|Y|>threshold) is calculated from t distribution. With n.feature features in total, expected number of features with abs value > threshold can be calculated from Binomial(n.feature, p)

Examples

```
expected_feature_diff(n.feature = 10, n.arm1 = 240, n.arm2 = 300, threshold = 0.2)
expected_feature_diff(n.feature = 10, n.arm1 = 240, n.arm2 = 300, threshold = c(0.1, 0.25))
```

forest_bygroup *Forest plot: colored by groups*

Description

Forest plot: colored by groups

Usage

```
forest_bygroup(
  data,
  summarystat,
  upperci,
  lowerci,
  population.name,
  group.name = population.name,
  color.group.name = population.name,
  stat.label = "Hazard Ratio",
  text.column = NULL,
  text.column.addition = NULL,
  color = NULL,
  stat.type.hr = TRUE,
  log.scale = FALSE,
  extra.stat = NULL,
  extra.stat.label = NULL,
  endpoint.name = "OS",
  study.name = "",
  draw = TRUE,
  shape.column = NULL,
  shape.vec = NULL
)
```

Arguments

data	data frame. plotting order will be following the order in the data frame.
summarystat	column name that indicates the summary statistics column (e.g. HR, ORR)
upperci, lowerci	column names that indicate the lower and upper CI of the summary statistics
population.name	column name of the column which indicates population names of the summary statistics
group.name	column name of the column which indicates which group each population is in. group names will be shown in the left panel of the forest plot
color.group.name	column name of the column which indicates how to color different groups

stat.label	Y axis label text
text.column	column name of the column which provides texts to be display on the right side of the forest plot. If it is NULL, the text will be generated by pasting summary stat column, the lowerci column and the upperci column
text.column.addition	additional columns to put (will be placed on the right of the figures), could be a vector of multiple column names
color	customized colors to different groups.
stat.type.hr	whether the summary statistics is HR. If so, the forest plot will be centered at 1
log.scale	whether show summary statistics in log scale
extra.stat	column name of the column which indicates the extra statistics to be drawn on the forest plot. The extra statistics will be drawn by "+". Could be NULL
extra.stat.label	label of the extra.stat (to be shown on y axis)
endpoint.name, study.name	text to be shown in title
draw	whether to draw
shape.column	column to pass to adjust shape. Use NULL for none
shape.vec	a vector to sepecify shapes, e.g. c(15, 16).

Value

return forest plot of class grob.

Examples

```
library(dplyr)
clinical_1 <- clinical %>% mutate(
  indicator = case_when(
    STRATUM == "strata_1" ~ 0,
    STRATUM == "strata_2" ~ 1,
    is.na(STRATUM) & ARM == "experimental" ~ 1,
    TRUE ~ -1
  ),
  ARM  = factor(ARM, levels = c("control","experimental")),
  BNLR = case_when(
    is.na(BNLR) ~ median(BNLR, na.rm = TRUE),
    TRUE ~ BNLR
  )
)
ipw_res <- ipw_strata(
  data.in = clinical_1, formula = indicator ~ BECOG + SEX + BNLR,
  indicator.var = "indicator", tte = "OS_MONTH", event = "OS_EVENT", trt = "ARM",
  class.of.int = list("strata_1" = 1, "strata_2" = 0)
 )
boot_ipw <- bootstrap_propen(
  data.in = clinical_1, formula = indicator ~ BECOG + SEX + BNLR,
```

```
  indicator.var = "indicator", tte = "OS_MONTH", event = "OS_EVENT", trt = "ARM",
  class.of.int = list("strata_1" = 1, "strata_2" = 0),
  estimate.res = ipw_res, method = "ipw", n.boot = 5
)
ps_res <- ps_match_strata(
  data.in = clinical_1, formula = indicator ~ BECOG + SEX + BNLR,
  indicator.var = "indicator", tte = "OS_MONTH", event = "OS_EVENT", trt = "ARM",
  class.of.int = list("strata_1" = 1, "strata_2" = 0)
 )
boot_ps <- bootstrap_propen(
  data.in = clinical_1, formula = indicator ~ BECOG + SEX + BNLR,
  indicator.var = "indicator", tte = "OS_MONTH", event = "OS_EVENT", trt = "ARM",
  class.of.int = list("strata_1" = 0, "strata_2" = 1),
  estimate.res = ps_res, method = "ps", n.boot = 5
)
boot.out.ipw <- boot_ipw$boot.out.est
boot.out.ps <- boot_ps$boot.out.est
ipw.ci.mat <- boot_ipw$est.ci.mat
ps.ci.mat <- boot_ps$est.ci.mat

data.fp <- data.frame(
  HR = round(exp(c(ipw.ci.mat[, 1], ps.ci.mat[, 1])), 2),
  LOWER = round(exp(c(ipw.ci.mat[, 2], ps.ci.mat[, 2])), 2),
  UPPER = round(exp(c(ipw.ci.mat[, 3], ps.ci.mat[, 3])), 2),
  ADA_Group = rep(rownames(ipw.ci.mat), 2),
  n = paste("n =", rep(table(clinical_1$indicator)[c("0", "1")], 2)),
  Methods_ADA = paste(
    rep(c("IPW", "PS"), each = 2), rep(rownames(ipw.ci.mat), 2)
  ),
  Methods = rep(c("IPW", "PS"), each = 2),
  bootstrapHR = c(
    boot.out.ipw[grep("HR", rownames(boot.out.ipw)), "Median"],
    boot.out.ps[grep("HR", rownames(boot.out.ps)), "Median"]
  )
)
forest_bygroup(
  data = data.fp, summarystat = "HR", upperci = "UPPER", lowerci = "LOWER",
  population.name = "Methods_ADA", group.name = "Methods",
  color.group.name = "ADA_Group", text.column.addition = "n",
  stat.label = "Hazard Ratio", text.column = NULL,
  stat.type.hr = TRUE, log.scale = FALSE, extra.stat = "bootstrapHR",
  extra.stat.label = "bootstrap median",
  endpoint.name = "OS", study.name = "Example Study", draw = TRUE
 )
```

`ipw_strata` *Inverse Probability weighting of strata (two or more strata, survival or binary endpoint)*

Description

This function performs inverse probability weighting of two or more strata.
It could be used when arm1 has 2 or more strata, while stratum information is unknown in arm2.
The function will fit a logistic regression (when 2 classes) or multinomial logistic regression (when > 2 classes) based on stratum labels in arm1 (model: label ~ features), then predict stratum labels for pts in arm2 based on the fitted model (as well as pts in arm1 who have missing labels, if there is any). The predicted probability of being stratum X will be used as weights when estimating treatment difference of two arms (Hazard ratio for survival endpoint; response rate difference for binary endpoint)

Usage

```
ipw_strata(
  data.in,
  formula,
  indicator.var = "indicator",
  class.of.int = NULL,
  tte = "AVAL",
  event = "event",
  trt = "trt",
  response = NULL,
  model = "plain",
  indicator.next = NULL,
  weights = NULL,
  multinom.maxit = 100,
  return.data = TRUE
)
```

Arguments

`data.in`	(`data.frame`) input data
`formula`	(`formula`) to input to the logistic or multinomial logistic model (in the form of strata~features)
`indicator.var`	(`string`) column name of the strata indicator variable which must be numeric. Assume arm1 has strata labeling and arm2 does not have strata labeling. pts without strata labeling should be indicated as -1 (e.g. pts in the arm1, or pts in arm2 but with missing label). within arm1 (the arm with strata labeling), subclasss should be indicated as 0,1,2...
`class.of.int`	(`list`) classes (stratum) of interest. Request to be in list format. It could be subset of classes in arm1; it could also define combined classes. For example: class.of.int = list("class1"=0, "class2"=1, "class3"=2, "class2or3"="c(1,2)"). for "class2or3", Prob(class 2 or 3) will be calculated as Prob(class2) + Prob(class3)
`tte`	(`string`) column name of the time to event variable
`event`	(`string`) column name of the event variable (1: event, 0: censor)
`trt`	(`string`) column name of the treatment variable. The variable is expected to be in `factor` format and the first level will be considered as reference (control) arm when calculating summary statistics.

`response` (`string`) column name of the response variable. 1 indicates responder and 0 indicates non responder. if response is not NULL, tte and event will be ignored and the function will assume binary outcome.

`model` (`string`) one of (`plain`, `dwc`, `wri`).

"plain" when 2 levels are specified in indicator variable, a binomial glm will be fitted; when more than 2 levels are specified, a multinomial glm will be fitted;

"dwc" Doubly weighted control: Two separated models will be fitted: one is binomial glm of 2 vs. (1, 0), the other one is bionomial glm of 1 vs 0. The probability of being each class is then calculated by aggregating these two models. Note this is similar to the plain method but with different (less rigid) covariance assumptions.

"wri" Weight regression imputation: the current status is going to be learned from the next status. Indicator of the next status should be specified using indicator.next. Currently "wri" only support the case where there are only two non-missing strata. In indicator variable, the two nonmissing strata should be coded as 0 and 1, the missing group should be coded as 2.

`indicator.next` (`string`) column name of the column which indicates status at a different measurement. It should be coded in the same way as in indicator.var (e.g. -1, 0, 1). Patients who have both missing current status and missing next status should be excluded in the modeling.

`weights` (`numeric`) weights of each subject. If not `NULL`, the estimated probabilities will be reweightsed to ensure sum(probability) of a subject = the subject's weights. If weights is not `NULL`, quasibinomial model will be used.

`multinom.maxit` see parameter `maxit` in nnet::multinom, default is 100

`return.data` (`logical`) whether to return data with estimated probabilities.

Value

return a `list` containing the following components:

- stat a `matrix` with rows as strata and columns as Estimate and CIs.
- converged `logical` to indicate whether model converges.
- any_warning_glm `logical` to indicate whether there's warning from glm model.
- warning.msg a `list` to capture any warning message from the modeling process.
- models a `list` to capture the glm model results.
- roc.list a `list` to capture information about Area under the curve from glm model.
- data a `data.frame` which is the original input data plus predicted probabilities.

Note

Three elements in the output list - the data element is a data frame that contains input data and estimated probabilities. The stat element contains estimated treatment difference between 2 arms, in each of the strata of interest. The converge element indicates whether the model converged (taking from $converged from stats::glm
and $convergency from nnet::multinom). if return.data is `FALSE`, data won't be returned.

Examples

```
 # example 1: Impute NA as one stratum in experimental arm; default model
 library(dplyr)
 clinical_1 <- clinical %>% mutate(
  indicator = case_when(
    STRATUM == "strata_1" ~ 0,
    STRATUM == "strata_2" ~ 1,
    is.na(STRATUM) & ARM == "experimental" ~ 1,
    TRUE ~ -1
  ),
  ARM  = factor(ARM, levels = c("control","experimental")),
  BNLR = case_when(
    is.na(BNLR) ~ median(BNLR, na.rm = TRUE),
    TRUE ~ BNLR
  )
)
ipw_res1 <- ipw_strata(
  data.in = clinical_1, formula = indicator ~ BECOG + SEX + BNLR,
  indicator.var = "indicator", tte = "OS_MONTH", event = "OS_EVENT", trt = "ARM",
  class.of.int = list("strata_1" = 1, "strata_2" = 0)
 )
 ## Weighted HRs
 ipw_res1$stat

 # example 2: "Weight regression imputation" model
clinical_2 <- clinical %>% mutate(
  indicator = case_when(
    STRATUM == "strata_1" ~ 0,
    STRATUM == "strata_2" ~ 1,
    is.na(STRATUM) & ARM == "experimental" ~ 2,
    TRUE ~ -1
  ),
  indicator_next = case_when(
    STRATUM_NEXT == "strata_1" ~ 0,
    STRATUM_NEXT == "strata_2" ~ 1,
    is.na(STRATUM_NEXT) & ARM == "experimental" ~ 2,
    TRUE ~ -1
  ),
  ARM  = factor(ARM, levels = c("control","experimental")),
  BNLR = case_when(
    is.na(BNLR) ~ median(BNLR, na.rm = TRUE),
    TRUE ~ BNLR
  )
)

ipw_res2 <- ipw_strata(
  data.in = clinical_2, formula = indicator ~ BECOG + SEX + BNLR, model = "wri",
  indicator.var = "indicator", indicator.next = "indicator_next",
   tte = "OS_MONTH", event = "OS_EVENT", trt = "ARM",
  class.of.int = list("strata_1" = 1, "strata_2" = 0)
 )
```

```
## Weighted HRs
ipw_res2$stat
```

---

km_plot_weight *Weighted KM plot*

Description

Weighted KM plot

Usage

```
km_plot_weight(
  data.in,
  indicator.var = "indicator",
  class.of.int = NULL,
  prob.names = NULL,
  filename = NULL,
  tte = "AVAL",
  event = "event",
  trt = "trt",
  time.unit = "month",
  prefix.title = "In strata:"
)
```

Arguments

data.in	input data, patients in rows and variables in columns. This could be an output from ipw_strata() or ps_match_strata().
indicator.var	(string) column name of the strata indicator variable which must be numeric. Assume arm1 has strata labeling and arm2 does not have strata labeling. pts without strata labeling should be indicated as -1 (e.g. pts in the arm1, or pts in arm2 but with missing label). within arm1 (the arm with strata labeling), subclasss should be indicated as 0,1,2...
class.of.int	(list) classes (stratum) of interest. Request to be in list format. It could be subset of classes in arm1; it could also define combined classes. For example: class.of.int = list("class1"=0, "class2"=1, "class3"=2, "class2or3"="c(1,2)"). for "class2or3", Prob(class 2 or 3) will be calculated as Prob(class2) + Prob(class3)
prob.names	column names for the probability scores to be used as weights. The order of probnames should match the order of class.of.int. if probnames is NULL, the function will assume that the probnames are pred0, pred1, prod2, prod1or2 in the example in class.of.int.
filename	if it is not NULL, a png file will be generated
tte	(string) column name of the time to event variable
event	(string) column name of the event variable (1: event, 0: censor)

trt	(string) column name of the treatment variable. The variable is expected to be in factor format and the first level will be considered as reference (control) arm when calculating summary statistics.
time.unit	time unit to be marked in x axis
prefix.title	prefix for title

Value

return a list of plots with class of ggsurvplot

Examples

```
library(dplyr)
clinical_1 <- clinical %>% mutate(
  indicator = case_when(
    STRATUM == "strata_1" ~ 0,
    STRATUM == "strata_2" ~ 1,
    is.na(STRATUM) & ARM == "experimental" ~ 1,
    TRUE ~ -1
  ),
  ARM  = factor(ARM, levels = c("control","experimental")),
  BNLR = case_when(
    is.na(BNLR) ~ median(BNLR, na.rm = TRUE),
    TRUE ~ BNLR
  )
)
ipw_res1 <- ipw_strata(
  data.in = clinical_1, formula = indicator ~ BECOG + SEX + BNLR,
  indicator.var = "indicator", tte = "OS_MONTH", event = "OS_EVENT", trt = "ARM",
  class.of.int = list("strata_1" = 1, "strata_2" = 0)
 )
 km_plot_weight(ipw_res1$data,
   indicator.var = "indicator", tte = "OS_MONTH", event = "OS_EVENT",
   trt = "ARM", class.of.int = list("strata_2" = 0))
ps_res1 <- ps_match_strata(
  data.in = clinical_1, formula = indicator ~ BECOG + SEX + BNLR,
  indicator.var = "indicator", tte = "OS_MONTH", event = "OS_EVENT", trt = "ARM",
  class.of.int = list("strata_1" = 1, "strata_2" = 0)
 )
 km_plot_weight(ps_res1$data,
   indicator.var = "indicator", tte = "OS_MONTH", event = "OS_EVENT",
   trt = "ARM", class.of.int = list("strata_1" = 1, "strata_2" = 0))
```

---

ps_match_strata *Propensity Score Matching of strata (two or more classes, survival or binary endpoint)*

Description

This function perfroms propensity score matching of two or more strata.
It could be used when arm1 has 2 or more strata, while strata information is unknown in arm2.
The function will fit a logistic regression (when 2 classes) or multinomial logistic regression (when > 2 classes) based on strata labels in arm1 (model: label~features), then predict strata labels in both arm1 and arm2 based on the fitted model. The predicted probability of being stratum X will be used for propensity score matching. The matching results will then be used to estimate treatment difference of two arms (Hazard ratio for survival endpoint; response rate difference for binary endpoint). When ties are allowed, weights from the ties will be used to calculate the HR or response rate difference.

Usage

```
ps_match_strata(
  data.in,
  formula,
  indicator.var = "indicator",
  ties = TRUE,
  class.of.int = NULL,
  tte = "AVAL",
  event = "event",
  trt = "trt",
  response = NULL,
  caliper = NULL,
  model = "plain",
  weights = NULL,
  multinom.maxit = 100,
  return.data = TRUE
)
```

Arguments

`data.in`	(`data.frame`) input data
`formula`	(`formula`) to input to the logistic or multinomial logistic model (in the form of strata~features)
`indicator.var`	(`string`) column name of the strata indicator variable which must be numeric. Assume arm1 has strata labeling and arm2 does not have strata labeling. pts without strata labeling should be indicated as -1 (e.g. pts in the arm1, or pts in arm2 but with missing label). within arm1 (the arm with strata labeling), subclasss should be indicated as 0,1,2...
`ties`	(`logical`) `TRUE` allows for ties.

ties is `TRUE`, all samples in the tie will be included. When calculating summary statistics, samples in ties will be assigned a smaller weight. for example, if two samples ties, these two samples will both be included in the summary statistics calculation with weight 0.5.

ties is FALSE, one random sample will be draw from the tied samples when calculating summary statistics. In this case, it is recommended to run ps_match_strata multiple times with different seeds and take the average or median summary statistics from multiple runs. Note when ties is FALSE, codes were tested less thoroughly and extra caution may be needed.

`class.of.int` (list) classes (stratum) of interest. Request to be in list format. It could be subset of classes in arm1; it could also define combined classes. For example: class.of.int = list("class1"=0, "class2"=1, "class3"=2, "class2or3"="c(1,2)"). for "class2or3", Prob(class 2 or 3) will be calculated as Prob(class2) + Prob(class3)

`tte` (string) column name of the time to event variable

`event` (string) column name of the event variable (1: event, 0: censor)

`trt` (string) column name of the treatment variable. The variable is expected to be in factor format and the first level will be considered as reference (control) arm when calculating summary statistics.

`response` (string) column name of the response variable. 1 indicates responder and 0 indicates non responder. if response is not NULL, tte and event will be ignored and the function will assume binary outcome.

`caliper` A scalar or vector denoting the caliper(s) which should be used when matching. A caliper is the distance which is acceptable for any match. Observations which are outside of the caliper are dropped. If a scalar caliper is provided, this caliper is used for all covariates in X. If a vector of calipers is provided, a caliper value should be provided for each covariate in X. The caliper is interpreted to be in standardized units. For example, caliper=.25 means that all matches not equal to or within .25 standard deviations of each covariate in X are dropped. Note that dropping observations generally changes the quantity being estimated.

`model` (string) one of (plain, dwc, wri).

"plain" when 2 levels are specified in indicator variable, a binomial glm will be fitted; when more than 2 levels are specified, a multinomial glm will be fitted;

"dwc" Doubly weighted control: Two separated models will be fitted: one is binomial glm of 2 vs. (1, 0), the other one is bionomial glm of 1 vs 0. The probability of being each class is then calculated by aggregating these two models. Note this is similar to the plain method but with different (less rigid) covariance assumptions.

"wri" Weight regression imputation: the current status is going to be learned from the next status. Indicator of the next status should be specified using indicator.next. Currently "wri" only support the case where there are only two non-missing strata. In indicator variable, the two nonmissing strata should be coded as 0 and 1, the missing group should be coded as 2.

`weights` (numeric) weights of each subject. If not NULL, the estimated probabilities will be reweightsed to ensure sum(probability) of a subject = the subject's weights. If weights is not NULL, quasibinomial model will be used.

`multinom.maxit` see parameter maxit in nnet::multinom, default is 100

`return.data` (logical) whether to return data with estimated probabilities.

Value

return a `list` containing the following components:

- stat a `matrix` with rows as strata and columns as Estimate and CIs.
- converged `logical` to indicate whether model converges.
- any_warning_glm `logical` to indicate whether there's warning from glm model.
- warning.msg a `list` to capture any warning message from the modeling process.
- models a `list` to capture the glm model results.
- roc.list a `list` to capture information about Area under the curve from glm model.
- data a `data.frame` which is the original input data plus predicted probabilities.

Note

Different from the original version, iter is no longer a parameter if tie = `FALSE` is specified, user need to run for loops of sapply outside of this function to get results from multiple seeds. Three elements in the output list - the data element is a data frame that contains input data and estimated probabilities. The stat element contains estimated treatment difference between 2 arms, in each of the strata of interest. The converge element indicates whether the model converged (taking from $converged from glm and $convergency from multinom) if return.data is `FALSE`, data won't be returned. model = "wri" is not supported in ps_match_strata

Examples

```
library(dplyr)
 # example 1:  Impute NA as one stratum in experimental arm; default model
clinical_1 <- clinical %>% mutate(
  indicator = case_when(
    STRATUM == "strata_1" ~ 0,
    STRATUM == "strata_2" ~ 1,
    is.na(STRATUM) & ARM == "experimental" ~ 1,
    TRUE ~ -1
  ),
  ARM  = factor(ARM, levels = c("control","experimental")),
  BNLR = case_when(
    is.na(BNLR) ~ median(BNLR, na.rm = TRUE),
    TRUE ~ BNLR
  )
)
ps_res1 <- ps_match_strata(
  data.in = clinical_1, formula = indicator ~ BECOG + SEX + BNLR,
  indicator.var = "indicator", tte = "OS_MONTH", event = "OS_EVENT", trt = "ARM",
  class.of.int = list("strata_1" = 1, "strata_2" = 0)
 )
 ## Weighted HRs
 ps_res1$stat

 # example 2: "doubly weighted control" model
clinical_2 <- clinical %>% mutate(
```

```
  indicator = case_when(
    STRATUM == "strata_1" ~ 0,
    STRATUM == "strata_2" ~ 1,
    is.na(STRATUM) & ARM == "experimental" ~ 2,
    TRUE ~ -1
  ),
  ARM  = factor(ARM, levels = c("control","experimental")),
  BNLR = case_when(
    is.na(BNLR) ~ median(BNLR, na.rm = TRUE),
    TRUE ~ BNLR
  )
)

ps_res2 <- ps_match_strata(
  data.in = clinical_2, formula = indicator ~ BECOG + SEX + BNLR, model = "dwc",
  indicator.var = "indicator", tte = "OS_MONTH", event = "OS_EVENT", trt = "ARM",
  class.of.int = list("strata_1" = 0, "strata_2" = 1, "missing" = 2)
 )
 ps_res2$stat
 ps_res2$converged
```

`std_diff` *Compare weighted and unweighted (naive analysis) standardized difference*

Description

Compare weighted and unweighted (naive analysis) standardized difference

Usage

```
std_diff(
  data.in,
  data.in.unadj = NULL,
  trt,
  vars,
  indicator.var = "indicator",
  class.of.int = NULL,
  prob.names = NULL,
  return.levels = FALSE,
  subj.aggr = TRUE,
  usubjid.var = "USUBJID"
)
```

Arguments

data.in	input data, patients in rows and variables in columns. This could be an output from ipw_strata() or ps_match_strata().
data.in.unadj	data set to use for the unadjusted analysis. For example, if PSM is used, the adjusted analysis should be done on the matched population but the unadjusted analysis should be done on the original population
trt	(string) column name of the treatment variable. The variable is expected to be in factor format and the first level will be considered as reference (control) arm when calculating summary statistics.
vars	variables of interest. standardized difference of each variable listed here will be calculated.
indicator.var	(string) column name of the strata indicator variable which must be numeric. Assume arm1 has strata labeling and arm2 does not have strata labeling. pts without strata labeling should be indicated as -1 (e.g. pts in the arm1, or pts in arm2 but with missing label). within arm1 (the arm with strata labeling), subclasss should be indicated as 0,1,2...
class.of.int	(list) classes (stratum) of interest. Request to be in list format. It could be subset of classes in arm1; it could also define combined classes. For example: class.of.int = list("class1"=0, "class2"=1, "class3"=2, "class2or3"="c(1,2)"). for "class2or3", Prob(class 2 or 3) will be calculated as Prob(class2) + Prob(class3)
prob.names	column names for the probability scores to be used as weights. The order of probnames should match the order of class.of.int. if probnames is NULL, the function will assume that the probnames are pred0, pred1, prod2, prod1or2 in the example in class.of.int.
return.levels	whether to return levels of each factor within each class.
subj.aggr	whether aggregate multiple entries from the same patients to one record
usubjid.var	column name indiacts subjuect id

Value

return a list, each list element is a data.frame containing absolute standardized difference for each variable.

Note

Calculation from Austin and Stuart (2015)

Examples

```
library(dplyr)
clinical_1 <- clinical %>% mutate(
  indicator = case_when(
    STRATUM == "strata_1" ~ 0,
    STRATUM == "strata_2" ~ 1,
    is.na(STRATUM) & ARM == "experimental" ~ 1,
    TRUE ~ -1
```

```
  ),
  ARM  = factor(ARM, levels = c("control","experimental")),
  BNLR = case_when(
    is.na(BNLR) ~ median(BNLR, na.rm = TRUE),
    TRUE ~ BNLR
  )
)
ipw_res1 <- ipw_strata(
  data.in = clinical_1, formula = indicator ~ BECOG + SEX + BNLR,
  indicator.var = "indicator", tte = "OS_MONTH", event = "OS_EVENT", trt = "ARM",
  class.of.int = list("strata_1" = 1, "strata_2" = 0)
 )
 std_diff(
  data.in = ipw_res1$data, vars = c("BECOG", "SEX", "BNLR"),
  indicator.var = "indicator", trt = "ARM",
  class.of.int = list("strata_1" = 1, "strata_2" = 0),
  usubjid.var = "SUBJID"
)
```

`std_diff_plot` *Compare weighted and unweighted (naive analysis) standardized difference in plot*

Description

Compare weighted and unweighted (naive analysis) standardized difference in plot

Usage

```
std_diff_plot(
  diff.list,
  legend.pos = "right",
  prefix.title = "In strata:",
  xlim.low = 0,
  xlim.high = 1
)
```

Arguments

`diff.list`	data list returned by function std_diff.
`legend.pos`	legend position: "left", "top", "right", "bottom".
`prefix.title`	prefix for title
`xlim.low`	(`numeric`) lower bound of xlim
`xlim.high`	(`numeric`) upper bound of xlim

Examples

```
## Not run:
library(dplyr)
clinical_1 <- clinical %>% mutate(
  indicator = case_when(
    STRATUM == "strata_1" ~ 0,
    STRATUM == "strata_2" ~ 1,
    is.na(STRATUM) & ARM == "experimental" ~ 1,
    TRUE ~ -1
  ),
  ARM  = factor(ARM, levels = c("control","experimental")),
  BNLR = case_when(
    is.na(BNLR) ~ median(BNLR, na.rm = TRUE),
    TRUE ~ BNLR
  )
)
ipw_res1 <- ipw_strata(
  data.in = clinical_1, formula = indicator ~ BECOG + SEX + BNLR,
  indicator.var = "indicator", tte = "OS_MONTH", event = "OS_EVENT", trt = "ARM",
  class.of.int = list("strata_1" = 1, "strata_2" = 0)
 )
ipw_diff <- std_diff(
  data.in = ipw_res1$data, vars = c("BECOG", "SEX", "BNLR"),
  indicator.var = "indicator", trt = "ARM",
  class.of.int = list("strata_1" = 1, "strata_2" = 0),
  usubjid.var = "SUBJID"
)
std_diff_plot(ipw_diff)

## End(Not run)
```

Index