Package 'spsurvey'

August 29, 2026

Title Spatial Sampling Design and Analysis

Version 5.7.0

Maintainer Michael Dumelle <Dumelle.Michael@epa.gov>

Description

A design-based approach to statistical inference, with a focus on spatial data. Spatially balanced samples are selected using the Generalized Random Tessellation Stratified (GRTS) algorithm. The GRTS algorithm can be applied to finite resources (point geometries) and infinite resources (linear / linestring and areal / polygon geometries) and flexibly accommodates a diverse set of sampling design features, including stratification, unequal inclusion probabilities, proportional (to size) inclusion probabilities, legacy (historical) sites, a minimum distance between sites, and two options for replacement sites (reverse hierarchical order and nearest neighbor). Data are analyzed using a wide range of analysis functions that perform categorical variable analysis, continuous variable analysis, attributable risk analysis, risk difference analysis, relative risk analysis, change analysis, and trend analysis. spsurvey can also be used to summarize objects, visualize objects, select samples that are not spatially balanced, select panel samples, measure the amount of spatial balance in a sample, adjust design weights, and more. For additional details, see Dumelle et al. (2023) <doi:10.18637/jss.v105.i03>.

Depends R (>= 3.5.0), sf, survey (>= 4.1-1)

Imports boot, crossdes, deldir, graphics, grDevices, lme4, MASS, sampling, stats, units

Suggests knitr, testthat, rmarkdown

License GPL (>= 3)

URL https://usepa.github.io/spsurvey/, https://github.com/USEPA/spsurvey

BugReports https://github.com/USEPA/spsurvey/issues

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

NeedsCompilation no

Author Michael Dumelle [aut, cre] (ORCID:
<https://orcid.org/0000-0002-3393-5529>),
Tom Kincaid [aut],
Anthony (Tony) R. Olsen [aut],
Marc Weber [aut],
Don Stevens [ctb],
Denis White [ctb],
Amanda M. Nahlik [ctb],
Sarah Lehmann [ctb]

Repository CRAN

Date/Publication 2026-08-29 12:30:02 UTC

Contents

spsurvey-package . . . 3
adjwgt . . . 4
adjwgtNR . . . 5
ash1_wgt . . . 6
attrisk_analysis . . . 7
cat_analysis . . . 14
cdf_plot . . . 19
change_analysis . . . 22
cont_analysis . . . 32
cont_cdfplot . . . 39
cont_cdftest . . . 41
cov_panel_dsgn . . . 47
diffrisk_analysis . . . 49
errorprnt . . . 55
grts . . . 56
Illinois_River . . . 63
Illinois_River_Legacy . . . 63
irs . . . 64
Lake_Ontario . . . 70
localmean_cov . . . 71
localmean_var . . . 72
localmean_weight . . . 72
NE_Lakes . . . 73
NE_Lakes_df . . . 73
NE_Lakes_Legacy . . . 74
NLA_PNW . . . 75
NRSA_EPA7 . . . 75
pd_summary . . . 76
plot . . . 77
plot.sp_CDF . . . 80
power_dsgn . . . 83
ppd_plot . . . 85
relrisk_analysis . . . 88

revisit_bibd 95
revisit_dsgn 97
revisit_rand 100
sp_balance 101
sp_frame 103
sp_plot 104
sp_rbind 106
sp_summary 107
stopprnt 109
summary 109
trend_analysis 111
warnprnt 119

Index **120**

`spsurvey-package` *spsurvey: Spatial Sampling Design and Analysis*

Description

spsurvey implements a design-based approach to statistical inference, with a focus on spatial data. Spatially balanced samples are selected using the Generalized Random Tessellation Stratified (GRTS) algorithm. The GRTS algorithm can be applied to finite resources (point geometries) and infinite resources (linear / linestring and areal / polygon geometries) and flexibly accommodates a diverse set of sampling design features, including stratification, unequal inclusion probabilities, proportional (to size) inclusion probabilities, legacy (historical) sites, a minimum distance between sites, and two options for replacement sites (reverse hierarchical order and nearest neighbor). Data are analyzed using a wide range of analysis functions that perform categorical variable analysis, continuous variable analysis, attributable risk analysis, risk difference analysis, relative risk analysis, change analysis, and trend analysis. spsurvey can also be used to summarize objects, visualize objects, select samples that are not spatially balanced, select panel samples, measure the amount of spatial balance in a sample, adjust design weights, and more. This R package has been reviewed in accordance with U.S. Environmental Protection Agency policy and approved for publication. Mention of trade names or commercial products does not constitute endorsement or recommendation for use.

Author(s)

Maintainer: Michael Dumelle <Dumelle.Michael@epa.gov> (ORCID)

Authors:

- Tom Kincaid
- Anthony (Tony) R. Olsen
- Marc Weber

Other contributors:

- Don Stevens [contributor]

- Denis White [contributor]
- Amanda M. Nahlik [contributor]
- Sarah Lehmann [contributor]

See Also

Useful links:

- https://usepa.github.io/spsurvey/
- https://github.com/USEPA/spsurvey
- Report bugs at https://github.com/USEPA/spsurvey/issues

`adjwgt` *Adjust survey design weights by categories*

Description

Adjust initial survey design weights so that the final weights sum to a desired frame size. Adjusted weights proportionally scale the initial weights to sum to the desired frame size. Separate adjustments are applied to each category specified in `wgtcat`.

Usage

```
adjwgt(wgt, wgtcat = NULL, framesize, sites = NULL)
```

Arguments

`wgt`	Vector of initial weights for each site. These equal the reciprocal of the site's inclusion probability.
`wgtcat`	Vector containing each site's weight adjustment category name. The default is `NULL`, which assumes every site is in the same category.
`framesize`	Vector containing the known size of the frame for each category name in `wgtcat`. If `wgtcat` is provided, the names in `framesize` must match the names in `wgtcat`. If `wgtcat` is not provided, an unnamed scalar is given to `framesize`.
`sites`	Vector indicating site use; `TRUE` indicates the site should be included in the weight adjustment and `FALSE` indicates the site should not be included in the weight adjustment. The default is `NULL`, which assumes every site should be included.

Value

Vector of adjusted weights, where the adjusted weight is set to `0` for sites whose value in the sites argument was set to `FALSE`.

Author(s)

Tony Olsen <olsen.tony@epa.gov>

Examples

```
wgt <- runif(50)
wgtcat <- rep(c("A", "B"), c(30, 20))
framesize <- c(A = 15, B = 10)
sites <- rep(rep(c(TRUE, FALSE), c(9, 1)), 5)
adjwgt(wgt, wgtcat, framesize, sites)
```

`adjwgtNR` *Adjust survey design weights for non-response by categories*

Description

Adjust weights for target sample units that do not respond and are missing at random within categories. The missing at random assumption implies that their sample weight may be assigned to specific categories of units that have responded (i.e., have been sampled). This is a class-based method for non-response adjustment.

Usage

```
adjwgtNR(wgt, MARClass = NULL, EvalStatus, TNRClass, TRClass)
```

Arguments

`wgt`	vector of weights for each sample unit that will be adjusted for non-response. Weights must be weights for the design as implemented. All weights must be greater than zero.
`MARClass`	vector that identifies for each sample unit the class (i.e., category, level) that will be used in non-response weight adjustment for sample units that are known to be target. Within each missing at random (MAR) class, the missing sample units that are not sampled are assumed to be missing at random. If MARClass is not specified, all sample units are assumed to be from the same MAR class.
`EvalStatus`	vector of the evaluation status for each sample unit. Values must include the values given in TNRclass and TRClass. May include other values not required for the non-response adjustment.
`TNRClass`	subset of values in EvalStatus that identify sample units whose target status is known and that do not respond (i.e., are not sampled).
`TRClass`	Subset of values in EvalStatus that identify sample units whose target status is known and that respond (i.e., are target and sampled).

Value

Vector of sample unit weights that are adjusted for non-response and that is the same length of input weights. Weights for sample units that did not response but were known to be eligible are set to zero. Weights for all other sample units are also set to zero.

Author(s)

Tony Olsen <olsen.tony@epa.gov>

Examples

```
set.seed(5)
wgt <- runif(40)
MARClass <- rep(c("A", "B"), rep(20, 2))
EvalStatus <- sample(c("Not_Target", "Target_Sampled", "Target_Not_Sampled"), 40, replace = TRUE)
TNRClass <- "Target_Not_Sampled"
TRClass <- "Target_Sampled"
adjwgtNR(wgt, MARClass, EvalStatus, TNRClass, TRClass)
# function that has an error check
```

---

ash1_wgt *Compute the average shifted histogram (ASH) for one-dimensional weighted data*

Description

Calculate the average shifted histogram estimate of a density based on one-dimensional data from a survey design with weights.

Usage

```
ash1_wgt(
  x,
  wgt = rep(1, length(x)),
  m = 5,
  nbin = 50,
  ab = NULL,
  support = "Continuous"
)
```

Arguments

x	Vector used to estimate the density. NA values are allowed.
wgt	Vector of weights for each observation from a probability sample. The default assigns equal weights (equal probability).
m	Number of empty bins to add to the ends when the range is not completely specified. The default is 5.
nbin	Number of bins for density estimation. The default is 50.
ab	Optional range for support associated with the density. Both values may be equal to NA. If equal to NA, then corresponding limit will be based on nicerange(). The default is NULL.
support	Type of support. If equal to "Continuous", then data are from a continuous distribution. If equal to "Ordinal", then data are from a discrete distribution defined for integers only. The default is "Continuous".

Value

List containing the ASH density estimate. List consists of

`tcen` x-coordinate for center of bin

`f` y-coordinate for density estimate height

Author(s)

Tony Olsen <Olsen.tony@epa.gov>

References

Scott, D. W. (1985). "Averaged shifted histograms: effective nonparametric density estimators in several dimensions." *The Annals of Statistics* 13(3): 1024-1040.

Examples

```
x <- rnorm(100, 10, sqrt(10))
wgt <- runif(100, 10, 100)
rslt <- ash1_wgt(x, wgt)
plot(rslt)
```

---

attrisk_analysis *Attributable risk analysis*

Description

This function organizes input and output for the analysis of attributable risk (for categorical variables). The analysis data, `dframe`, can be either a data frame or a simple features (`sf`) object. If an `sf` object is used, coordinates are extracted from the geometry column in the object, arguments `xcoord` and `ycoord` are assigned values `"xcoord"` and `"ycoord"`, respectively, and the geometry column is dropped from the object.

Usage

```
attrisk_analysis(
  dframe,
  vars_response,
  vars_stressor,
  response_levels = NULL,
  stressor_levels = NULL,
  subpops = NULL,
  siteID = NULL,
  weight = "weight",
  xcoord = NULL,
  ycoord = NULL,
  stratumID = NULL,
```

```
  clusterID = NULL,
  weight1 = NULL,
  xcoord1 = NULL,
  ycoord1 = NULL,
  sizeweight = FALSE,
  sweight = NULL,
  sweight1 = NULL,
  fpc = NULL,
  popsize = NULL,
  vartype = "local",
  conf = 95,
  All_Sites = FALSE,
  subset_local = TRUE
)
```

Arguments

dframe	Data to be analyzed (analysis data). A data frame or sf object containing survey design variables, response variables, stressor variables, and subpopulation (domain) variables.
vars_response	Vector composed of character values that identify the names of response variables in dframe. Each response variable must have two category values (levels), where one level is associated with poor condition and the other level is associated with good condition.
vars_stressor	Vector composed of character values that identify the names of stressor variables in dframe. Each stressor variable must have two category values (levels), where one level is associated with poor condition and the other level is associated with good condition.
response_levels	List providing the category values (levels) for each element in the vars_response argument. Each element in the list must contain two values, where the first value identifies poor condition, and the second value identifies good condition. This argument must be named and must be the same length as argument vars_response. Names for this argument must match the values in the vars_response argument. If this argument equals NULL, then a named list is created that contains the values "Poor" and "Good" for the first and second levels, respectively, of each element in the vars_response argument and that uses values in the vars_response argument as names for the list. If response_levels is provided without names, then the names of response_levels are set to vars_response. The default value is NULL.
stressor_levels	List providing the category values (levels) for each element in the vars_stressor argument. Each element in the list must contain two values, where the first value identifies poor condition, and the second value identifies good condition. This argument must be named and must be the same length as argument vars_stressor. Names for this argument must match the values in the vars_stressor argument. If this argument equals NULL, then a named list is

created that contains the values "Poor" and "Good" for the first and second levels, respectively, of each element in the vars_stressor argument and that uses values in the vars_stressor argument as names for the list. If stressor_levels is provided without names, then the names of stressor_levels are set to vars_stressor. The default value is NULL.

subpops — Vector composed of character values that identify the names of subpopulation (domain) variables in dframe. If a value is not provided, the value "All_Sites" is assigned to the subpops argument and a factor variable named "All_Sites" that takes the value "All Sites" is added to dframe. The default value is NULL.

siteID — Character value providing the name of the site ID variable in dframe. For a two-stage sample, the site ID variable identifies stage two site IDs. The default value is NULL, which assumes that each row in dframe represents a unique site.

weight — Character value providing the name of the design weight variable in dframe. For a two-stage sample, the weight variable identifies stage two weights. The default value is "weight".

xcoord — Character value providing name of the x-coordinate variable in dframe. For a two-stage sample, the x-coordinate variable identifies stage two x-coordinates. Note that x-coordinates are required for calculation of the local mean variance estimator. If dframe is an sf object, this argument is not required (as the geometry column in dframe is used to find the x-coordinate). The default value is NULL.

ycoord — Character value providing name of the y-coordinate variable in dframe. For a two-stage sample, the y-coordinate variable identifies stage two y-coordinates. Note that y-coordinates are required for calculation of the local mean variance estimator. If dframe is an sf object, this argument is not required (as the geometry column in dframe is used to find the t-coordinate). The default value is NULL.

stratumID — Character value providing the name of the stratum ID variable in dframe. The default value is NULL.

clusterID — Character value providing the name of the cluster (stage one) ID variable in dframe. Note that cluster IDs are required for a two-stage sample. The default value is NULL.

weight1 — Character value providing the name of the stage one weight variable in dframe. The default value is NULL.

xcoord1 — Character value providing the name of the stage one x-coordinate variable in dframe. Note that x coordinates are required for calculation of the local mean variance estimator. The default value is NULL.

ycoord1 — Character value providing the name of the stage one y-coordinate variable in dframe. Note that y-coordinates are required for calculation of the local mean variance estimator. The default value is NULL.

sizeweight — Logical value that indicates whether size weights should be used during estimation, where TRUE uses size weights and FALSE does not use size weights. To employ size weights for a single-stage sample, a value must be supplied for argument weight. To employ size weights for a two-stage sample, values must be supplied for arguments weight and weight1. The default value is FALSE.

sweight — Character value providing the name of the size weight variable in dframe. For a two-stage sample, the size weight variable identifies stage two size weights. The default value is NULL.

sweight1 — Character value providing the name of the stage one size weight variable in dframe. The default value is NULL.

fpc — Object that specifies values required for calculation of the finite population correction factor used during variance estimation. The object must match the survey design in terms of stratification and whether the design is single-stage or two-stage. For an unstratified design, the object is a vector. The vector is composed of a single numeric value for a single-stage design. For a two-stage unstratified design, the object is a named vector containing one more than the number of clusters in the sample, where the first item in the vector specifies the number of clusters in the population and each subsequent item specifies the number of stage two units for the cluster. The name for the first item in the vector is arbitrary. Subsequent names in the vector identify clusters and must match the cluster IDs. For a stratified design, the object is a named list of vectors, where names must match the strata IDs. For each stratum, the format of the vector is identical to the format described for unstratified single-stage and two-stage designs. Note that the finite population correction factor is not used with the local mean variance estimator.

Example fpc for a single-stage unstratified survey design:

```
fpc <- 15000
```

Example fpc for a single-stage stratified survey design:

```
fpc <- list(
    Stratum_1 = 9000,
    Stratum_2 = 6000)
```

Example fpc for a two-stage unstratified survey design:

```
fpc <- c(
    Ncluster = 150,
    Cluster_1 = 150,
    Cluster_2 = 75,
    Cluster_3 = 75,
    Cluster_4 = 125,
    Cluster_5 = 75)
```

Example fpc for a two-stage stratified survey design:

```
fpc <- list(
    Stratum_1 = c(
      Ncluster_1 = 100,
      Cluster_1 = 125,
      Cluster_2 = 100,
      Cluster_3 = 100,
      Cluster_4 = 125,
      Cluster_5 = 50),
    Stratum_2 = c(
      Ncluster_2 = 50,
      Cluster_1 = 75,
```

```
      Cluster_2 = 150,
      Cluster_3 = 75,
      Cluster_4 = 75,
      Cluster_5 = 125))
```

popsize

Object that provides values for the population argument of the calibrate or postStratify functions in the survey package. If a value is provided for popsize, then either the calibrate or postStratify function is used to modify the survey design object that is required by functions in the survey package. Whether to use the calibrate or postStratify function is dictated by the format of popsize, which is discussed below. Post-stratification adjusts the sampling and replicate weights so that the joint distribution of a set of post-stratifying variables matches the known population joint distribution. Calibration, generalized raking, or GREG estimators generalize post-stratification and raking by calibrating a sample to the marginal totals of variables in a linear regression model. For the calibrate function, the object is a named list, where the names identify factor variables in dframe. Each element of the list is a named vector containing the population total for each level of the associated factor variable. For the postStratify function, the object is either a data frame, table, or xtabs object that provides the population total for all combinations of selected factor variables in the dframe data frame. If a data frame is used for popsize, the variable containing population totals must be the last variable in the data frame. If a table is used for popsize, the table must have named dimnames where the names identify factor variables in the dframe data frame. If the popsize argument is equal to NULL, then neither calibration nor post-stratification is performed. The default value is NULL.

Example popsize for calibration:

```
popsize <- list(
    Ecoregion = c(
      East = 750,
      Central = 500,
      West = 250),
    Type = c(
      Streams = 1150,
      Rivers = 350))
```

Example popsize for post-stratification using a data frame:

```
popsize <- data.frame(
    Ecoregion = rep(c("East", "Central", "West"),
      rep(2, 3)),
    Type = rep(c("Streams", "Rivers"), 3),
    Total = c(575, 175, 400, 100, 175, 75))
```

Example popsize for post-stratification using a table:

```
popsize <- with(MySurveyFrame,
    table(Ecoregion, Type))
```

Example popsize for post-stratification using an xtabs object:

```
popsize <- xtabs(~Ecoregion + Type,
```

```
data = MySurveyFrame)
```

`vartype` Character value providing the choice of the variance estimator, where `"Local"` indicates the local mean estimator and `"SRS"` indicates the simple random sampling estimator. The default value is `"local"`.

`conf` Numeric value providing the Gaussian-based confidence level. The default value is `95`.

`All_Sites` A logical variable used when `subpops` is not `NULL`. If `All_Sites` is `TRUE`, then alongside the subpopulation output, output for all sites (ignoring subpopulations) is returned for each variable in `vars`. If `All_Sites` is `FALSE`, then alongside the subpopulation output, output for all sites (ignoring subpopulations) is not returned for each variable in `vars`. The default is `FALSE`.

`subset_local` When 'subset_local = TRUE' (the default), subpopulations are subset to include only subpopulation members prior to variance estimation. When 'subset_local = FALSE', subpopulations are not subset prior to variance estimation. 'subset_local = FALSE' follows standard design-based subpopulation (i.e., domain) estimation theory but is computationally more complex, requiring the neighborhood weights matrix of the full data (which scales cubically with the sample size). 'subset_local' only applies when 'vartype = "local"'.

Value

The analysis results. A data frame of population estimates for all combinations of subpopulations, categories within each subpopulation, response variables, and categories within each response variable. Estimates are provided for proportion and size of the population plus standard error, margin of error, and confidence interval estimates. The data frame contains the following variables:

Type subpopulation (domain) name

Subpopulation subpopulation name within a domain

Response response variable

Stressor stressor variable

nResp sample size

Estimate attributable risk estimate

StdError_log attributable risk standard error (on the log scale)

MarginofError_log attributable risk margin of error (on the log scale)

LCBxxPct xx% (default 95%) lower confidence bound

UCBxxPct xx% (default 95%) upper confidence bound

WeightTotal sum of design weights

Count_RespPoor_StressPoor number of observations in the poor response and poor stressor group

Count_RespPoor_StressGood number of observations in the poor response and good stressor group

Count_RespGood_StressPoor number of observations in the good response and poor stressor group

Count_RespGood_StressGood number of observations in the good response and good stressor group

Prop_RespPoor_StressPoor weighted proportion of observations in the poor response and poor stressor group

Prop_RespPoor_StressGood weighted proportion of observations in the poor response and good stressor group

Prop_RespGood_StressPoor weighted proportion of observations in the good response and poor stressor group

Prop_RespGood_StressGood weighted proportion of observations in the good response and good stressor group

Details

Attributable risk measures the proportional reduction in the extent of poor condition of a response variable that presumably would result from eliminating a stressor variable, where the response and stressor variables are classified as either good (i.e., reference condition) or poor (i.e., different from reference condition). Attributable risk is defined as one minus the ratio of two probabilities. The numerator of the ratio is the conditional probability that the response variable is in poor condition given that the stressor variable is in good condition. The denominator of the ratio is the probability that the response variable is in poor condition. Attributable risk values close to zero indicate that removing the stressor variable will have little or no impact on the probability that the response variable is in poor condition. Attributable risk values close to one indicate that removing the stressor variable will result in extensive reduction of the probability that the response variable is in poor condition.

Author(s)

Tom Kincaid <Kincaid.Tom@epa.gov>

References

Sickle, J. V., & Paulsen, S. G. (2008). Assessing the attributable risks, relative risks, and regional extents of aquatic stressors. *Journal of the North American Benthological Society*, 27(4), 920-931.

See Also

`relrisk_analysis` for relative risk analysis

`diffrisk_analysis` for risk difference analysis

Examples

```
dframe <- data.frame(
  siteID = paste0("Site", 1:100),
  wgt = runif(100, 10, 100),
  xcoord = runif(100),
  ycoord = runif(100),
  stratum = rep(c("Stratum1", "Stratum2"), 50),
  RespVar1 = sample(c("Poor", "Good"), 100, replace = TRUE),
  RespVar2 = sample(c("Poor", "Good"), 100, replace = TRUE),
  StressVar = sample(c("Poor", "Good"), 100, replace = TRUE),
  All_Sites = rep("All Sites", 100),
```

```
  Resource_Class = rep(c("Agr", "Forest"), c(55, 45))
)
myresponse <- c("RespVar1", "RespVar2")
mystressor <- c("StressVar")
mysubpops <- c("All_Sites", "Resource_Class")
attrisk_analysis(dframe,
  vars_response = myresponse,
  vars_stressor = mystressor, subpops = mysubpops, siteID = "siteID",
  weight = "wgt", xcoord = "xcoord", ycoord = "ycoord",
  stratumID = "stratum"
)
```

`cat_analysis` *Categorical variable analysis*

Description

This function organizes input and output for the analysis of categorical variables. The analysis data, `dframe`, can be either a data frame or a simple features (`sf`) object. If an `sf` object is used, coordinates are extracted from the geometry column in the object, arguments `xcoord` and `ycoord` are assigned values `"xcoord"` and `"ycoord"`, respectively, and the geometry column is dropped from the object.

Usage

```
cat_analysis(
  dframe,
  vars,
  subpops = NULL,
  siteID = NULL,
  weight = "weight",
  xcoord = NULL,
  ycoord = NULL,
  stratumID = NULL,
  clusterID = NULL,
  weight1 = NULL,
  xcoord1 = NULL,
  ycoord1 = NULL,
  sizeweight = FALSE,
  sweight = NULL,
  sweight1 = NULL,
  fpc = NULL,
  popsize = NULL,
  vartype = "local",
  jointprob = "overton",
  conf = 95,
  All_Sites = FALSE,
  subset_local = TRUE
)
```

Arguments

dframe	Data to be analyzed (analysis data). A data frame or sf object containing survey design variables, response variables, and subpopulation (domain) variables.
vars	Vector composed of character values that identify the names of response variables in dframe.
subpops	Vector composed of character values that identify the names of subpopulation (domain) variables in dframe. If a value is not provided, the value "All_Sites" is assigned to the subpops argument and a factor variable named "All_Sites" that takes the value "All Sites" is added to the dframe data frame. The default value is NULL.
siteID	Character value providing name of the site ID variable in the dframe data frame. For a two-stage sample, the site ID variable identifies stage two site IDs. The default value is NULL, which assumes that each row in dframe represents a unique site.
weight	Character value providing name of the design weight variable in dframe. For a two-stage sample, the weight variable identifies stage two weights. The default value is "weight".
xcoord	Character value providing name of the x-coordinate variable in the dframe data frame. For a two-stage sample, the x-coordinate variable identifies stage two x-coordinates. Note that x-coordinates are required for calculation of the local mean variance estimator. If dframe is an sf object, this argument is not required (as the geometry column in dframe is used to find the x-coordinate). The default value is NULL.
ycoord	Character value providing name of the y-coordinate variable in the dframe data frame. For a two-stage sample, the y-coordinate variable identifies stage two y-coordinates. Note that y-coordinates are required for calculation of the local mean variance estimator. If dframe is an sf object, this argument is not required (as the geometry column in dframe is used to find the y-coordinate). The default value is NULL.
stratumID	Character value providing name of the stratum ID variable in the dframe data frame. The default value is NULL.
clusterID	Character value providing the name of the cluster (stage one) ID variable in dframe. Note that cluster IDs are required for a two-stage sample. The default value is NULL.
weight1	Character value providing name of the stage one weight variable in dframe. The default value is NULL.
xcoord1	Character value providing the name of the stage one x-coordinate variable in dframe. Note that x coordinates are required for calculation of the local mean variance estimator. The default value is NULL.
ycoord1	Character value providing the name of the stage one y-coordinate variable in dframe. Note that y-coordinates are required for calculation of the local mean variance estimator. The default value is NULL.
sizeweight	Logical value that indicates whether size weights should be used during estimation, where TRUE uses size weights and FALSE does not use size weights. To

employ size weights for a single-stage sample, a value must be supplied for argument weight. To employ size weights for a two-stage sample, values must be supplied for arguments `weight` and `weight1`. The default value is `FALSE`.

`sweight` Character value providing the name of the size weight variable in `dframe`. For a two-stage sample, the size weight variable identifies stage two size weights. The default value is `NULL`.

`sweight1` Character value providing name of the stage one size weight variable in `dframe`. The default value is `NULL`.

`fpc` Object that specifies values required for calculation of the finite population correction factor used during variance estimation. The object must match the survey design in terms of stratification and whether the design is single-stage or two-stage. For an unstratified design, the object is a vector. The vector is composed of a single numeric value for a single-stage design. For a two-stage unstratified design, the object is a named vector containing one more than the number of clusters in the sample, where the first item in the vector specifies the number of clusters in the population and each subsequent item specifies the number of stage two units for the cluster. The name for the first item in the vector is arbitrary. Subsequent names in the vector identify clusters and must match the cluster IDs. For a stratified design, the object is a named list of vectors, where names must match the strata IDs. For each stratum, the format of the vector is identical to the format described for unstratified single-stage and two-stage designs. Note that the finite population correction factor is not used with the local mean variance estimator.

Example fpc for a single-stage unstratified survey design:

```
fpc <- 15000
```

Example fpc for a single-stage stratified survey design:

```
fpc <- list(
    Stratum_1 = 9000,
    Stratum_2 = 6000)
```

Example fpc for a two-stage unstratified survey design:

```
fpc <- c(
    Ncluster = 150,
    Cluster_1 = 150,
    Cluster_2 = 75,
    Cluster_3 = 75,
    Cluster_4 = 125,
    Cluster_5 = 75)
```

Example fpc for a two-stage stratified survey design:

```
fpc <- list(
    Stratum_1 = c(
      Ncluster_1 = 100,
      Cluster_1 = 125,
      Cluster_2 = 100,
      Cluster_3 = 100,
      Cluster_4 = 125,
```

```
    Cluster_5 = 50),
  Stratum_2 = c(
    Ncluster_2 = 50,
    Cluster_1 = 75,
    Cluster_2 = 150,
    Cluster_3 = 75,
    Cluster_4 = 75,
    Cluster_5 = 125))
```

popsize

Object that provides values for the population argument of the `calibrate` or `postStratify` functions in the survey package. If a value is provided for popsize, then either the `calibrate` or `postStratify` function is used to modify the survey design object that is required by functions in the survey package. Whether to use the `calibrate` or `postStratify` function is dictated by the format of popsize, which is discussed below. Post-stratification adjusts the sampling and replicate weights so that the joint distribution of a set of post-stratifying variables matches the known population joint distribution. Calibration, generalized raking, or GREG estimators generalize post-stratification and raking by calibrating a sample to the marginal totals of variables in a linear regression model. For the `calibrate` function, the object is a named list, where the names identify factor variables in `dframe`. Each element of the list is a named vector containing the population total for each level of the associated factor variable. For the `postStratify` function, the object is either a data frame, table, or xtabs object that provides the population total for all combinations of selected factor variables in the `dframe` data frame. If a data frame is used for `popsize`, the variable containing population totals must be the last variable in the data frame. If a table is used for `popsize`, the table must have named `dimnames` where the names identify factor variables in the `dframe` data frame. If the popsize argument is equal to `NULL`, then neither calibration nor post-stratification is performed. The default value is `NULL`.

Example popsize for calibration:

```
popsize <- list(
    Ecoregion = c(
      East = 750,
      Central = 500,
      West = 250),
    Type = c(
      Streams = 1150,
      Rivers = 350))
```

Example popsize for post-stratification using a data frame:

```
popsize <- data.frame(
    Ecoregion = rep(c("East", "Central", "West"),
      rep(2, 3)),
    Type = rep(c("Streams", "Rivers"), 3),
    Total = c(575, 175, 400, 100, 175, 75))
```

Example popsize for post-stratification using a table:

```
popsize <- with(MySurveyFrame,
    table(Ecoregion, Type))
```

Example popsize for post-stratification using an xtabs object:

```
popsize <- xtabs(~Ecoregion + Type,
    data = MySurveyFrame)
```

`vartype` Character value providing the choice of the variance estimator, where `"Local"` indicates the local mean estimator, `"SRS"` indicates the simple random sampling estimator, `"HT"` indicates the Horvitz-Thompson estimator, and `"YG"` indicates the Yates-Grundy estimator.

`jointprob` Character value providing the choice of joint inclusion probability approximation for use with Horvitz-Thompson and Yates-Grundy variance estimators, where `"overton"` indicates the Overton approximation, `"hr"` indicates the Hartley-Rao approximation, and `"brewer"` equals the Brewer approximation. The default value is `"overton"`.

`conf` Numeric value providing the Gaussian-based confidence level. The default value is `95`.

`All_Sites` A logical variable used when `subpops` is not `NULL`. If `All_Sites` is `TRUE`, then alongside the subpopulation output, output for all sites (ignoring subpopulations) is returned for each variable in `vars`. If `All_Sites` is `FALSE`, then alongside the subpopulation output, output for all sites (ignoring subpopulations) is not returned for each variable in `vars`. The default is `FALSE`.

`subset_local` When ‘subset_local = TRUE‘ (the default), subpopulations are subset to include only subpopulation members prior to variance estimation. When ‘subset_local = FALSE‘, subpopulations are not subset prior to variance estimation. ‘subset_local = FALSE‘ follows standard design-based subpopulation (i.e., domain) estimation theory, especially relevant for variance estimates of totals, but is computationally more complex, requiring the neighborhood weights matrix of the full data (which scales cubically with the sample size). ‘subset_local‘ only applies when ‘vartype = "local"‘.

Value

The analysis results. A data frame of population estimates for all combinations of subpopulations, categories within each subpopulation, response variables, and categories within each response variable. Estimates are provided for proportion and total of the population plus standard error, margin of error, and confidence interval estimates. The data frame contains the following variables:

Type subpopulation (domain) name

Subpopulation subpopulation name within a domain

Indicator response variable

Category category of response variable

nResp sample size

Estimate.P proportion estimate (in %)

StdError.P standard error of proportion estimate

MarginofError.P margin of error of proportion estimate

LCBxxPct.P xx% (default 95%) lower confidence bound of proportion estimate

UCBxxPct.P xx% (default 95%) upper confidence bound of proportion estimate

Estimate.U total estimate

StdError.U standard error of total estimate

MarginofError.U margin of error of total estimate

LCBxxPct.U xx% (default 95%) lower confidence bound of total estimate

UCBxxPct.U xx% (default 95%) upper confidence bound of total estimate

Author(s)

Tom Kincaid <Kincaid.Tom@epa.gov>

See Also

cont_analysis for continuous variable analysis

Examples

```
dframe <- data.frame(
  siteID = paste0("Site", 1:100),
  wgt = runif(100, 10, 100),
  xcoord = runif(100),
  ycoord = runif(100),
  stratum = rep(c("Stratum1", "Stratum2"), 50),
  CatVar = rep(c("north", "south", "east", "west"), 25),
  All_Sites = rep("All Sites", 100),
  Resource_Class = rep(c("Good", "Poor"), c(55, 45))
)
myvars <- c("CatVar")
mysubpops <- c("All_Sites", "Resource_Class")
mypopsize <- data.frame(
  Resource_Class = c("Good", "Poor"),
  Total = c(4000, 1500)
)
cat_analysis(dframe,
  vars = myvars, subpops = mysubpops, siteID = "siteID",
  weight = "wgt", xcoord = "xcoord", ycoord = "ycoord",
  stratumID = "stratum", popsize = mypopsize
)
```

---

cdf_plot *Plot a cumulative distribution function (CDF)*

Description

This function creates a CDF plot. Input data for the plots is provided by a data frame with the same structure as the "CDF" output from cont_analysis. Confidence limits for the CDF also are plotted.

Usage

```
cdf_plot(
  cdfest,
  var = NULL,
  subpop = NULL,
  subpop_level = NULL,
  units_cdf = "Percent",
  type_cdf = "Continuous",
  log = "",
  xlab = NULL,
  ylab = NULL,
  ylab_r = NULL,
  main = NULL,
  legloc = NULL,
  confcut = 0,
  conflev = 95,
  cex.main = 1.2,
  cex.legend = 1,
  ...
)
```

Arguments

cdfest	Data frame with the same structure as the "CDF" output from cont_analysis.
var	If cdfest has multiple variables in the "Indicator" column, then var is the single variable to be plotted. The default is NULL, which assumes that only one variable is in the "Indicator" column of cdfest.
subpop	If cdfest has multiple variables in the "Type" column, then subpop is the single variable to be plotted. The default is NULL, which assumes that only one variable is in the "Type" column of cdfest.
subpop_level	If cdfest has multiple levels of subpop in the "Subpopulation" column, then subpop_level is the single level to be plotted. The default is NULL, which assumes that only one level is in the "Subpopulation" column of cdfest.
units_cdf	Indicator for the label utilized for the left side y-axis and the values used for the left side y-axis tick marks, where "Percent" means the label and values are in terms of percent of the population, and "Units" means the label and values are in terms of units (count, length, or area) of the population. The default is "Percent".
type_cdf	Character string consisting of the value "Continuous" or "Ordinal" that controls the type of CDF plot. The default is "Continuous".
log	Character string consisting of the value "" or "x" that controls whether the x axis uses the original scale ("") or the base 10 logarithmic scale ("x"). The default is "".
xlab	Character string providing the x-axis label. If this argument equals NULL, then the indicator name is used as the label. The default is NULL.

ylab	Character string providing the left side y-axis label. If argument units_cdf equals "Units", a value should be provided for this argument. Otherwise, the label will be "Percent". The default is "Percent".
ylab_r	Character string providing the label for the right side y-axis (and, hence, determining the values used for the right side y-axis tick marks), where NULL means a right side y-axis is not created. If this argument equals "Same", the right side y-axis will have the same label and tick mark values as the left side y-axis. If this argument equals a character string other than "Same", the right side y-axis label will be the value provided for argument ylab_r, and the right side y-axis tick mark values will be determined by the choice not utilized for argument units_cdf, which means that the default value of argument units_cdf (i.e., "Percent") will result in the right side y-axis tick mark values being expressed in terms of units of the population (i.e., count, length, or area). The default is NULL.
main	Character string providing the plot title. The default is NULL.
legloc	Indicator for location of the plot legend, where "BR" means bottom right, "BL" means bottom left, "TR" means top right, "TL" means top left, and NULL means no legend. The default is NULL.
confcut	Numeric value that controls plotting confidence limits at the CDF extremes. Confidence limits for CDF values (percent scale) less than confcut or greater than 100 minus confcut are not plotted. A value of zero means confidence limits are plotted for the complete range of the CDF. The default is 0.
conflev	Numeric value of the confidence level used for confidence limits. The default is 95.
cex.main	Expansion factor for the plot title. The default is 1.2.
cex.legend	Expansion factor for the legend title. The default is 1.
...	Additional arguments passed to the `plot.default` function (aside from those already used and `ylim`).

Value

A plot of a variable's CDF estimates associated confidence limits.

Author(s)

Tom Kincaid <Kincaid.Tom@epa.gov>

See Also

`cont_cdfplot` for creating a PDF file containing CDF plots

`cont_cdftest` for CDF hypothesis testing

Examples

```
## Not run:
dframe <- data.frame(
  siteID = paste0("Site", 1:100),
```

```
  wgt = runif(100, 10, 100),
  xcoord = runif(100),
  ycoord = runif(100),
  stratum = rep(c("Stratum1", "Stratum2"), 50),
  ContVar = rnorm(100, 10, 1),
  All_Sites = rep("All Sites", 100),
  Resource_Class = rep(c("Good", "Poor"), c(55, 45))
)
myvars <- c("ContVar")
mysubpops <- c("All_Sites", "Resource_Class")
mypopsize <- data.frame(
  Resource_Class = c("Good", "Poor"),
  Total = c(4000, 1500)
)
myanalysis <- cont_analysis(dframe,
  vars = myvars, subpops = mysubpops,
  siteID = "siteID", weight = "wgt", xcoord = "xcoord", ycoord = "ycoord",
  stratumID = "stratum", popsize = mypopsize
)
keep <- with(myanalysis$CDF, Type == "Resource_Class" &
  Subpopulation == "Good")
par(mfrow = c(2, 1))
cdf_plot(myanalysis$CDF[keep, ],
  xlab = "ContVar",
  ylab = "Percent of Stream Length", ylab_r = "Stream Length (km)",
  main = "Estimates for Resource Class: Good"
)
cdf_plot(myanalysis$CDF[keep, ],
  xlab = "ContVar",
  ylab = "Percent of Stream Length", ylab_r = "Same",
  main = "Estimates for Resource Class: Good"
)

## End(Not run)
```

`change_analysis` *Change analysis*

Description

This function organizes input and output for the estimation of change between two samples (for categorical and continuous variables). The analysis data, `dframe`, can be either a data frame or a simple features (`sf`) object. If an `sf` object is used, coordinates are extracted from the geometry column in the object, arguments `xcoord` and `ycoord` are assigned values `"xcoord"` and `"ycoord"`, respectively, and the geometry column is dropped from the object.

Usage

```
change_analysis(
  dframe,
```

```
  vars_cat = NULL,
  vars_cont = NULL,
  test = "mean",
  subpops = NULL,
  surveyID = "surveyID",
  survey_names = NULL,
  siteID = "siteID",
  weight = "weight",
  revisitwgt = FALSE,
  xcoord = NULL,
  ycoord = NULL,
  stratumID = NULL,
  clusterID = NULL,
  weight1 = NULL,
  xcoord1 = NULL,
  ycoord1 = NULL,
  sizeweight = FALSE,
  sweight = NULL,
  sweight1 = NULL,
  fpc = NULL,
  popsize = NULL,
  vartype = "local",
  jointprob = "overton",
  conf = 95,
  All_Sites = FALSE,
  subset_local = TRUE
)
```

Arguments

dframe	Data to be analyzed (analysis data). A data frame or sf object containing survey design variables, response variables, and subpopulation (domain) variables.
vars_cat	Vector composed of character values that identify the names of categorical response variables in dframe. The default is NULL.
vars_cont	Vector composed of character values that identify the names of continuous response variables in dframe. The default is NULL.
test	Character string or character vector providing the location measure(s) to use for change estimation for continuous variables. The choices are "mean", "total", "median", or some combination of the three options (e.g., c("mean", "total")). The default is "mean".
subpops	Vector composed of character values that identify the names of subpopulation (domain) variables in dframe. If a value is not provided, the value "All_Sites" is assigned to the subpops argument and a factor variable named "All_Sites" that takes the value "All Sites" is added to dframe. The default value is NULL.
surveyID	Character value providing name of the survey ID variable in dframe. The default value is "surveyID".

survey_names — Character vector of length two that provides the survey names contained in the surveyID variable in the dframe data frame. The two values in the vector identify the first survey and second survey, respectively. If a value is not provided, unique values of the surveyID variable are assigned to the survey_names argument. The default is NULL.

siteID — Character value providing name of the site ID variable in dframe. For a two-stage sample, the site ID variable identifies stage two site IDs. The default value is "siteID". If a unique site is visited in both surveys, the corresponding siteID should be the same for both entries.

weight — Character value providing name of the design weight variable in dframe. For a two-stage sample, the weight variable identifies stage two weights. The default value is "weight".

revisitwgt — Logical value that indicates whether each repeat visit site has the same design weight in the two surveys, where TRUE = the weight for each repeat visit site is the same and FALSE = the weight for each repeat visit site is not the same. When this argument is FALSE, all of the repeat visit sites are assigned equal weights when calculating the covariance component of the change estimate standard error. The default is FALSE.

xcoord — Character value providing name of the x-coordinate variable in dframe. For a two-stage sample, the x-coordinate variable identifies stage two x-coordinates. Note that x-coordinates are required for calculation of the local mean variance estimator. If dframe is an sf object, this argument is not required (as the geometry column in dframe is used to find the x-coordinate). The default value is NULL.

ycoord — Character value providing name of the y-coordinate variable in dframe. For a two-stage sample, the y-coordinate variable identifies stage two y-coordinates. Note that y-coordinates are required for calculation of the local mean variance estimator. If dframe is an sf object, this argument is not required (as the geometry column in dframe is used to find the y-coordinate). The default value is NULL.

stratumID — Character value providing name of the stratum ID variable in dframe. The default value is NULL.

clusterID — Character value providing the name of the cluster (stage one) ID variable in dframe. Note that cluster IDs are required for a two-stage sample. The default value is NULL.

weight1 — Character value providing name of the stage one weight variable in dframe. The default value is NULL.

xcoord1 — Character value providing the name of the stage one x-coordinate variable in dframe. Note that x coordinates are required for calculation of the local mean variance estimator. The default value is NULL.

ycoord1 — Character value providing the name of the stage one y-coordinate variable in dframe. Note that y-coordinates are required for calculation of the local mean variance estimator. The default value is NULL.

sizeweight — Logical value that indicates whether size weights should be used during estimation, where TRUE uses size weights and FALSE does not use size weights. To

employ size weights for a single-stage sample, a value must be supplied for argument weight. To employ size weights for a two-stage sample, values must be supplied for arguments `weight` and `weight1`. The default value is `FALSE`.

`sweight` Character value providing the name of the size weight variable in `dframe`. For a two-stage sample, the size weight variable identifies stage two size weights. The default value is `NULL`.

`sweight1` Character value providing name of the stage one size weight variable in `dframe`. The default value is `NULL`.

`fpc` Object that specifies values required for calculation of the finite population correction factor used during variance estimation. The object must match the survey design in terms of stratification and whether the design is single-stage or two-stage. For an unstratified design, the object is a vector. The vector is composed of a single numeric value for a single-stage design. For a two-stage unstratified design, the object is a named vector containing one more than the number of clusters in the sample, where the first item in the vector specifies the number of clusters in the population and each subsequent item specifies the number of stage two units for the cluster. The name for the first item in the vector is arbitrary. Subsequent names in the vector identify clusters and must match the cluster IDs. For a stratified design, the object is a named list of vectors, where names must match the strata IDs. For each stratum, the format of the vector is identical to the format described for unstratified single-stage and two-stage designs. Note that the finite population correction factor is not used with the local mean variance estimator.

Example fpc for a single-stage unstratified survey design:

```
fpc <- 15000
```

Example fpc for a single-stage stratified survey design:

```
fpc <- list(
    Stratum_1 = 9000,
    Stratum_2 = 6000)
```

Example fpc for a two-stage unstratified survey design:

```
fpc <- c(
    Ncluster = 150,
    Cluster_1 = 150,
    Cluster_2 = 75,
    Cluster_3 = 75,
    Cluster_4 = 125,
    Cluster_5 = 75)
```

Example fpc for a two-stage stratified survey design:

```
fpc <- list(
    Stratum_1 = c(
      Ncluster_1 = 100,
      Cluster_1 = 125,
      Cluster_2 = 100,
      Cluster_3 = 100,
      Cluster_4 = 125,
```

```
      Cluster_5 = 50),
    Stratum_2 = c(
      Ncluster_2 = 50,
      Cluster_1 = 75,
      Cluster_2 = 150,
      Cluster_3 = 75,
      Cluster_4 = 75,
      Cluster_5 = 125))
```

popsize

Object that provides values for the population argument of the `calibrate` or `postStratify` functions in the survey package. If a value is provided for popsize, then either the `calibrate` or `postStratify` function is used to modify the survey design object that is required by functions in the survey package. Whether to use the `calibrate` or `postStratify` function is dictated by the format of popsize, which is discussed below. Post-stratification adjusts the sampling and replicate weights so that the joint distribution of a set of post-stratifying variables matches the known population joint distribution. Calibration, generalized raking, or GREG estimators generalize post-stratification and raking by calibrating a sample to the marginal totals of variables in a linear regression model. For the `calibrate` function, the object is a named list, where the names identify factor variables in `dframe`. Each element of the list is a named vector containing the population total for each level of the associated factor variable. For the `postStratify` function, the object is either a data frame, table, or xtabs object that provides the population total for all combinations of selected factor variables in the `dframe` data frame. If a data frame is used for `popsize`, the variable containing population totals must be the last variable in the data frame. If a table is used for `popsize`, the table must have named `dimnames` where the names identify factor variables in the `dframe` data frame. If the popsize argument is equal to `NULL`, then neither calibration nor post-stratification is performed. The default value is `NULL`.

Example popsize for calibration:

```
popsize <- list(
    Ecoregion = c(
      East = 750,
      Central = 500,
      West = 250),
    Type = c(
      Streams = 1150,
      Rivers = 350))
```

Example popsize for post-stratification using a data frame:

```
popsize <- data.frame(
    Ecoregion = rep(c("East", "Central", "West"),
      rep(2, 3)),
    Type = rep(c("Streams", "Rivers"), 3),
    Total = c(575, 175, 400, 100, 175, 75))
```

Example popsize for post-stratification using a table:

```
popsize <- with(MySurveyFrame,
    table(Ecoregion, Type))
```

Example popsize for post-stratification using an xtabs object:

```
popsize <- xtabs(~Ecoregion + Type,
    data = MySurveyFrame)
```

`vartype` Character value providing the choice of the variance estimator, where `"Local"` indicates the local mean estimator and `"SRS"` indicates the simple random sampling estimator. The default value is `"local"`.

`jointprob` Character value providing the choice of joint inclusion probability approximation for use with Horvitz-Thompson and Yates-Grundy variance estimators, where `"overton"` indicates the Overton approximation, `"hr"` indicates the Hartley-Rao approximation, and `"brewer"` equals the Brewer approximation. The default value is `"overton"`.

`conf` Numeric value providing the Gaussian-based confidence level. The default value is 95.

`All_Sites` A logical variable used when `subpops` is not `NULL`. If `All_Sites` is `TRUE`, then alongside the subpopulation output, output for all sites (ignoring subpopulations) is returned for each variable in `vars`. If `All_Sites` is `FALSE`, then alongside the subpopulation output, output for all sites (ignoring subpopulations) is not returned for each variable in `vars`. The default is `FALSE`.

`subset_local` When 'subset_local = TRUE' (the default), subpopulations are subset to include only subpopulation members prior to variance estimation. When 'subset_local = FALSE', subpopulations are not subset prior to variance estimation. 'subset_local = FALSE' follows standard design-based subpopulation (i.e., domain) estimation theory but is computationally more complex, requiring the neighborhood weights matrix of the full data (which scales cubically with the sample size). 'subset_local' only applies when 'vartype = "local"'.

Value

List of change estimates composed of four items: (1) `catsum` contains change estimates for categorical variables, (2) `contsum_mean` contains estimates for continuous variables using the mean, (3) `contsum_total` contains estimates for continuous variables using the total, and (4) `contsum_median` contains estimates for continuous variables using the median. The items in the list will contain `NULL` for estimates that were not calculated. Each data frame includes estimates for all combinations of population Types, subpopulations within types, response variables, and categories within each response variable (for categorical variables and continuous variables using the median). Change estimates are provided plus standard error estimates and confidence interval estimates.

The `catsum` data frame contains the following variables:

Survey_1 first survey name

Survey_2 second survey name

Type subpopulation (domain) name

Subpopulation subpopulation name within a domain

Indicator response variable

Category category of response variable

DiffEst.P proportion difference estimate (in %; second survey - first survey)

StdError.P standard error of proportion difference estimate

MarginofError.P margin of error of proportion difference estimate

LCBxxPct.P xx% (default 95%) lower confidence bound of proportion difference estimate

UCBxxPct.P xx% (default 95%) upper confidence bound of proportion difference estimate

Estimate.U total difference estimate (second survey - first survey)

StdError.U standard error of total difference estimate

MarginofError.U margin of error of total difference estimate

LCBxxPct.U xx% (default 95%) lower confidence bound of total difference estimate

UCBxxPct.U xx% (default 95%) upper confidence bound of total difference estimate

nResp_1 sample size in the first survey

Estimate.P_1 proportion estimate (in %) from the first survey

StdError.P_1 standard error of proportion estimate from the first survey

MarginofError.P_1 margin of error of proportion estimate from the first survey

LCBxxPct.P_1 xx% (default 95%) lower confidence bound of proportion estimate from the first survey

UCBxxPct.P_1 xx% (default 95%) upper confidence bound of proportion estimate from the first survey

nResp_2 sample size in the second survey

Estimate.U_1 total estimate from the first survey

StdError.U_1 standard error of total estimate from the first survey

MarginofError.U_1 margin of error of total estimate from the first survey

LCBxxPct.U_1 xx% (default 95%) lower confidence bound of total estimate from the first survey

UCBxxPct.U_1 xx% (default 95%) upper confidence bound of total estimate from the first survey

Estimate.P_2 proportion estimate (in %) from the second survey

StdError.P_2 standard error of proportion estimate from the second survey

MarginofError.P_2 margin of error of proportion estimate from the second survey

LCBxxPct.P_2 xx% (default 95%) lower confidence bound of proportion estimate from the second survey

UCBxxPct.P_2 xx% (default 95%) upper confidence bound of proportion estimate from the second survey

Estimate.U_2 total estimate from the second survey

StdError.U_2 standard error of total estimate from the second survey

MarginofError.U_2 margin of error of total estimate from the second survey

LCBxxPct.U_2 xx% (default 95%) lower confidence bound of total estimate from the second survey

UCBxxPct.U_2 xx% (default 95%) upper confidence bound of total estimate from the second survey

The contsum_mean data frame contains the following variables:

Survey_1 first survey name

Survey_2 second survey name

Type subpopulation (domain) name

Subpopulation subpopulation name within a domain

Indicator response variable

Statistic value of percentile

nResp sample size at or below Value

DiffEst mean difference estimate

StdError standard error of mean difference estimate

MarginofError margin of error of mean difference estimate

LCBxxPct xx% (default 95%) lower confidence bound of mean difference estimate

UCBxxPct xx% (default 95%) upper confidence bound of mean difference estimate

nResp_1 sample size in the first survey

Estimate_1 mean estimate from the first survey

StdError_1 standard error of mean estimate from the first survey

MarginofError_1 margin of error of mean estimate from the first survey

LCBxxPct_1 xx% (default 95%) lower confidence bound of mean estimate from the first survey

UCBxxPct_1 xx% (default 95%) upper confidence bound of mean estimate from the first survey

nResp_2 sample size in the second survey

Estimate_2 mean estimate from the second survey

StdError_2 standard error of mean estimate from the second survey

MarginofError_2 margin of error of mean estimate from the second survey

LCBxxPct_2 xx% (default 95%) lower confidence bound of mean estimate from the second survey

UCBxxPct_2 xx% (default 95%) upper confidence bound of mean estimate from the second survey

The contsum_total data frame contains the following variables:

Survey_1 first survey name

Survey_2 second survey name

Type subpopulation (domain) name

Subpopulation subpopulation name within a domain

Indicator response variable

Statistic value of percentile

nResp sample size at or below Value

DiffEst total difference estimate

StdError standard error of total difference estimate

MarginofError margin of error of total difference estimate

LCBxxPct xx% (default 95%) lower confidence bound of total difference estimate

UCBxxPct xx% (default 95%) upper confidence bound of total difference estimate

nResp_1 sample size in the first survey

Estimate_1 total estimate from the first survey

StdError_1 standard error of total estimate from the first survey

MarginofError_1 margin of error of total estimate from the first survey

LCBxxPct_1 xx% (default 95%) lower confidence bound of total estimate from the first survey

UCBxxPct_1 xx% (default 95%) upper confidence bound of total estimate from the first survey

nResp_2 sample size in the second survey

Estimate_2 total estimate from the second survey

StdError_2 standard error of total estimate from the second survey

MarginofError_2 margin of error of total estimate from the second survey

LCBxxPct_2 xx% (default 95%) lower confidence bound of total estimate from the second survey

UCBxxPct_2 xx% (default 95%) upper confidence bound of total estimate from the second survey

The `contsum_median` data frame contains the following variables:

Survey_1 first survey name

Survey_2 second survey name

Type subpopulation (domain) name

Subpopulation subpopulation name within a domain

Indicator response variable

Category category of response variable

DiffEst.P proportion above or below median difference estimate (in %; second survey - first survey)

StdError.P standard error of proportion above or below median difference estimate

MarginofError.P margin of error of proportion above or below median difference estimate

LCBxxPct.P xx% (default 95%) lower confidence bound of proportion above or below median difference estimate

UCBxxPct.P xx% (default 95%) upper confidence bound of proportion above or below median difference estimate

Estimate.U total above or below median difference estimate (second survey - first survey)

StdError.U standard error of total above or below median difference estimate

MarginofError.U margin of error of total above or below median difference estimate

LCBxxPct.U xx% (default 95%) lower confidence bound of total above or below median difference estimate

UCBxxPct.U xx% (default 95%) upper confidence bound of total above or below median difference estimate

nResp_1 sample size in the first survey

Estimate.P_1 proportion above or below median estimate (in %) from the first survey

StdError.P_1 standard error of proportion above or below median estimate from the first survey

MarginofError.P_1 margin of error of proportion above or below median estimate from the first survey

LCBxxPct.P_1 xx% (default 95%) lower confidence bound of proportion above or below median estimate from the first survey

UCBxxPct.P_1 xx% (default 95%) upper confidence bound of proportion above or below median estimate from the first survey

nResp_2 sample size in the second survey

Estimate.U_1 total above or below median estimate from the first survey

StdError.U_1 standard error of total above or below median estimate from the first survey

MarginofError.U_1 margin of error of total above or below median estimate from the first survey

LCBxxPct.U_1 xx% (default 95%) lower confidence bound of total above or below median estimate from the first survey

UCBxxPct.U_1 xx% (default 95%) upper confidence bound of total above or below median estimate from the first survey

Estimate.P_2 proportion above or below median estimate (in %) from the second survey

StdError.P_2 standard error of proportion above or below median estimate from the second survey

MarginofError.P_2 margin of error of proportion above or below median estimate from the second survey

LCBxxPct.P_2 xx% (default 95%) lower confidence bound of proportion above or below median estimate from the second survey

UCBxxPct.P_2 xx% (default 95%) upper confidence bound of proportion above or below median estimate from the second survey

Estimate.U_2 total above or below median estimate from the second survey

StdError.U_2 standard error of total above or below median estimate from the second survey

MarginofError.U_2 margin of error of total above or below median estimate from the second survey

LCBxxPct.U_2 xx% (default 95%) lower confidence bound of total above or below median estimate from the second survey

UCBxxPct.U_2 xx% (default 95%) upper confidence bound of total above or below median estimate from the second survey

Author(s)

Tom Kincaid <Kincaid.Tom@epa.gov>

See Also

`trend_analysis` for trend analysis

Examples

```
# Categorical variable example for three resource classes
dframe <- data.frame(
  surveyID = rep(c("Survey 1", "Survey 2"), c(100, 100)),
  siteID = paste0("Site", 1:200),
  wgt = runif(200, 10, 100),
  xcoord = runif(200),
  ycoord = runif(200),
  stratum = rep(rep(c("Stratum 1", "Stratum 2"), c(2, 2)), 50),
  CatVar = rep(c("North", "South"), 100),
  All_Sites = rep("All Sites", 200),
  Resource_Class = sample(c("Good", "Fair", "Poor"), 200, replace = TRUE)
)
myvars <- c("CatVar")
mysubpops <- c("All_Sites", "Resource_Class")
change_analysis(dframe,
  vars_cat = myvars, subpops = mysubpops,
  surveyID = "surveyID", siteID = "siteID", weight = "wgt",
  xcoord = "xcoord", ycoord = "ycoord", stratumID = "stratum"
)
```

---

cont_analysis *Continuous variable analysis*

Description

This function organizes input and output for the analysis of continuous variables. The analysis data, `dframe`, can be either a data frame or a simple features (`sf`) object. If an `sf` object is used, coordinates are extracted from the geometry column in the object, arguments `xcoord` and `ycoord` are assigned values `"xcoord"` and `"ycoord"`, respectively, and the geometry column is dropped from the object.

Usage

```
cont_analysis(
  dframe,
  vars,
  subpops = NULL,
  siteID = NULL,
  weight = "weight",
  xcoord = NULL,
  ycoord = NULL,
  stratumID = NULL,
  clusterID = NULL,
  weight1 = NULL,
  xcoord1 = NULL,
  ycoord1 = NULL,
  sizeweight = FALSE,
```

```
  sweight = NULL,
  sweight1 = NULL,
  fpc = NULL,
  popsize = NULL,
  vartype = "local",
  jointprob = "overton",
  conf = 95,
  pctval = c(5, 10, 25, 50, 75, 90, 95),
  statistics = c("CDF", "Pct", "Mean", "Total"),
  All_Sites = FALSE,
  subset_local = TRUE
)
```

Arguments

dframe	Data to be analyzed (analysis data). A data frame or sf object containing survey design variables, response variables, and subpopulation (domain) variables.
vars	Vector composed of character values that identify the names of response variables in dframe.
subpops	Vector composed of character values that identify the names of subpopulation (domain) variables in dframe. If a value is not provided, the value "All_Sites" is assigned to the subpops argument and a factor variable named "All_Sites" that takes the value "All Sites" is added to the dframe data frame. The default value is NULL.
siteID	Character value providing name of the site ID variable in the dframe data frame. For a two-stage sample, the site ID variable identifies stage two site IDs. The default value is NULL, which assumes that each row in dframe represents a unique site.
weight	Character value providing name of the design weight variable in dframe. For a two-stage sample, the weight variable identifies stage two weights. The default value is "weight".
xcoord	Character value providing name of the x-coordinate variable in the dframe data frame. For a two-stage sample, the x-coordinate variable identifies stage two x-coordinates. Note that x-coordinates are required for calculation of the local mean variance estimator. If dframe is an sf object, this argument is not required (as the geometry column in dframe is used to find the x-coordinate). The default value is NULL.
ycoord	Character value providing name of the y-coordinate variable in the dframe data frame. For a two-stage sample, the y-coordinate variable identifies stage two y-coordinates. Note that y-coordinates are required for calculation of the local mean variance estimator. If dframe is an sf object, this argument is not required (as the geometry column in dframe is used to find the y-coordinate). The default value is NULL.
stratumID	Character value providing name of the stratum ID variable in the dframe data frame. The default value is NULL.

clusterID — Character value providing the name of the cluster (stage one) ID variable in dframe. Note that cluster IDs are required for a two-stage sample. The default value is NULL.

weight1 — Character value providing name of the stage one weight variable in dframe. The default value is NULL.

xcoord1 — Character value providing the name of the stage one x-coordinate variable in dframe. Note that x coordinates are required for calculation of the local mean variance estimator. The default value is NULL.

ycoord1 — Character value providing the name of the stage one y-coordinate variable in dframe. Note that y-coordinates are required for calculation of the local mean variance estimator. The default value is NULL.

sizeweight — Logical value that indicates whether size weights should be used during estimation, where TRUE uses size weights and FALSE does not use size weights. To employ size weights for a single-stage sample, a value must be supplied for argument weight. To employ size weights for a two-stage sample, values must be supplied for arguments weight and weight1. The default value is FALSE.

sweight — Character value providing the name of the size weight variable in dframe. For a two-stage sample, the size weight variable identifies stage two size weights. The default value is NULL.

sweight1 — Character value providing name of the stage one size weight variable in dframe. The default value is NULL.

fpc — Object that specifies values required for calculation of the finite population correction factor used during variance estimation. The object must match the survey design in terms of stratification and whether the design is single-stage or two-stage. For an unstratified design, the object is a vector. The vector is composed of a single numeric value for a single-stage design. For a two-stage unstratified design, the object is a named vector containing one more than the number of clusters in the sample, where the first item in the vector specifies the number of clusters in the population and each subsequent item specifies the number of stage two units for the cluster. The name for the first item in the vector is arbitrary. Subsequent names in the vector identify clusters and must match the cluster IDs. For a stratified design, the object is a named list of vectors, where names must match the strata IDs. For each stratum, the format of the vector is identical to the format described for unstratified single-stage and two-stage designs. Note that the finite population correction factor is not used with the local mean variance estimator.

Example fpc for a single-stage unstratified survey design:

```
fpc <- 15000
```

Example fpc for a single-stage stratified survey design:

```
fpc <- list(
    Stratum_1 = 9000,
    Stratum_2 = 6000)
```

Example fpc for a two-stage unstratified survey design:

```
fpc <- c(
    Ncluster = 150,
```

```
    Cluster_1 = 150,
    Cluster_2 = 75,
    Cluster_3 = 75,
    Cluster_4 = 125,
    Cluster_5 = 75)
```

Example fpc for a two-stage stratified survey design:

```
fpc <- list(
    Stratum_1 = c(
      Ncluster_1 = 100,
      Cluster_1 = 125,
      Cluster_2 = 100,
      Cluster_3 = 100,
      Cluster_4 = 125,
      Cluster_5 = 50),
    Stratum_2 = c(
      Ncluster_2 = 50,
      Cluster_1 = 75,
      Cluster_2 = 150,
      Cluster_3 = 75,
      Cluster_4 = 75,
      Cluster_5 = 125))
```

popsize

Object that provides values for the population argument of the `calibrate` or `postStratify` functions in the survey package. If a value is provided for popsize, then either the `calibrate` or `postStratify` function is used to modify the survey design object that is required by functions in the survey package. Whether to use the `calibrate` or `postStratify` function is dictated by the format of popsize, which is discussed below. Post-stratification adjusts the sampling and replicate weights so that the joint distribution of a set of post-stratifying variables matches the known population joint distribution. Calibration, generalized raking, or GREG estimators generalize post-stratification and raking by calibrating a sample to the marginal totals of variables in a linear regression model. For the `calibrate` function, the object is a named list, where the names identify factor variables in `dframe`. Each element of the list is a named vector containing the population total for each level of the associated factor variable. For the `postStratify` function, the object is either a data frame, table, or xtabs object that provides the population total for all combinations of selected factor variables in the `dframe` data frame. If a data frame is used for `popsize`, the variable containing population totals must be the last variable in the data frame. If a table is used for `popsize`, the table must have named `dimnames` where the names identify factor variables in the `dframe` data frame. If the popsize argument is equal to `NULL`, then neither calibration nor post-stratification is performed. The default value is `NULL`.

Example popsize for calibration:

```
popsize <- list(
    Ecoregion = c(
      East = 750,
```

```
      Central = 500,
      West = 250),
    Type = c(
      Streams = 1150,
      Rivers = 350))
```

Example popsize for post-stratification using a data frame:

```
popsize <- data.frame(
    Ecoregion = rep(c("East", "Central", "West"),
      rep(2, 3)),
    Type = rep(c("Streams", "Rivers"), 3),
    Total = c(575, 175, 400, 100, 175, 75))
```

Example popsize for post-stratification using a table:

```
popsize <- with(MySurveyFrame,
    table(Ecoregion, Type))
```

Example popsize for post-stratification using an xtabs object:

```
popsize <- xtabs(~Ecoregion + Type,
    data = MySurveyFrame)
```

`vartype` Character value providing the choice of the variance estimator, where `"Local"` indicates the local mean estimator, `"SRS"` indicates the simple random sampling estimator, `"HT"` indicates the Horvitz-Thompson estimator, and `"YG"` indicates the Yates-Grundy estimator.

`jointprob` Character value providing the choice of joint inclusion probability approximation for use with Horvitz-Thompson and Yates-Grundy variance estimators, where `"overton"` indicates the Overton approximation, `"hr"` indicates the Hartley-Rao approximation, and `"brewer"` equals the Brewer approximation. The default value is `"overton"`.

`conf` Numeric value providing the Gaussian-based confidence level. The default value is 95.

`pctval` Vector of the set of values at which percentiles are estimated. The default set is: `c(5, 10, 25, 50, 75, 90, 95)`.

`statistics` Character vector specifying desired estimates, where `"CDF"` specifies CDF estimates, `"Pct"` specifies percentile estimates, `"Mean"` specifies mean estimates, and "Total" specifies total estimates. Any combination of the four choices may be provided by the user. The default value is `c("CDF", "Pct", "Mean", "Total")`.

`All_Sites` A logical variable used when `subpops` is not `NULL`. If `All_Sites` is `TRUE`, then alongside the subpopulation output, output for all sites (ignoring subpopulations) is returned for each variable in `vars`. If `All_Sites` is `FALSE`, then alongside the subpopulation output, output for all sites (ignoring subpopulations) is not returned for each variable in `vars`. The default is `FALSE`.

`subset_local` When 'subset_local = TRUE' (the default), subpopulations are subset to include only subpopulation members prior to variance estimation. When 'subset_local = FALSE', subpopulations are not subset prior to variance estimation. 'subset_local = FALSE' follows standard design-based subpopulation (i.e., domain) estimation theory, especially relevant for variance estimates of totals, but is computationally more complex, requiring the neighborhood weights matrix of the

full data (which scales cubically with the sample size). ‘subset_local‘ only applies when ‘vartype = "local"‘.

Value

The analysis results. A list composed of one, two, three, or four data frames that contain population estimates for all combinations of subpopulations, categories within each subpopulation, and response variables, where the number of data frames is determined by argument `statistics`. The possible data frames in the output list are:

`CDF` : a data frame containing CDF estimates

`Pct` : data frame containing percentile estimates

`Mean` : a data frame containing mean estimates

`Total` : a data frame containing total estimates

The `CDF` data frame contains the following variables:

Type subpopulation (domain) name

Subpopulation subpopulation name within a domain

Indicator response variable

Value value of response variable

nResp sample size at or below `Value`

Estimate.P CDF proportion estimate (in %)

StdError.P standard error of CDF proportion estimate

MarginofError.P margin of error of CDF proportion estimate

LCBxxPct.P xx% (default 95%) lower confidence bound of CDF proportion estimate

UCBxxPct.P xx% (default 95%) upper confidence bound of CDF proportion estimate

Estimate.U CDF total estimate

StdError.U standard error of CDF total estimate

MarginofError.U margin of error of CDF total estimate

LCBxxPct.U xx% (default 95%) lower confidence bound of CDF total estimate

UCBxxPct.U xx% (default 95%) upper confidence bound of CDF total estimate

The `Pct` data frame contains the following variables:

Type subpopulation (domain) name

Subpopulation subpopulation name within a domain

Indicator response variable

Statistic value of percentile

nResp sample size at or below `Value`

Estimate percentile estimate

StdError standard error of percentile estimate

MarginofError margin of error of percentile estimate

LCBxxPct xx% (default 95%) lower confidence bound of percentile estimate

UCBxxPct xx% (default 95%) upper confidence bound of percentile estimate

The `Mean` data frame contains the following variables:

Type subpopulation (domain) name

Subpopulation subpopulation name within a domain

Indicator response variable

nResp sample size at or below `Value`

Estimate mean estimate

StdError standard error of mean estimate

MarginofError margin of error of mean estimate

LCBxxPct xx% (default 95%) lower confidence bound of mean estimate

UCBxxPct xx% (default 95%) upper confidence bound of mean estimate

The `Total` data frame contains the following variables:

Type subpopulation (domain) name

Subpopulation subpopulation name within a domain

Indicator response variable

nResp sample size at or below `Value`

Estimate total estimate

StdError standard error of total estimate

MarginofError margin of error of total estimate

LCBxxPct xx% (default 95%) lower confidence bound of total estimate

UCBxxPct xx% (default 95%) upper confidence bound of total estimate

Author(s)

Tom Kincaid <Kincaid.Tom@epa.gov>

See Also

`cat_analysis` for categorical variable analysis

Examples

```
dframe <- data.frame(
  siteID = paste0("Site", 1:100),
  wgt = runif(100, 10, 100),
  xcoord = runif(100),
  ycoord = runif(100),
  stratum = rep(c("Stratum1", "Stratum2"), 50),
  ContVar = rnorm(100, 10, 1),
  All_Sites = rep("All Sites", 100),
  Resource_Class = rep(c("Good", "Poor"), c(55, 45))
```

```
)
myvars <- c("ContVar")
mysubpops <- c("All_Sites", "Resource_Class")
mypopsize <- data.frame(
  Resource_Class = c("Good", "Poor"),
  Total = c(4000, 1500)
)
cont_analysis(dframe,
  vars = myvars, subpops = mysubpops, siteID = "siteID",
  weight = "wgt", xcoord = "xcoord", ycoord = "ycoord",
  stratumID = "stratum", popsize = mypopsize, statistics = "Mean"
)
```

`cont_cdfplot` *Create a PDF file containing cumulative distribution functions (CDF) plots*

Description

This function creates a PDF file containing CDF plots. Input data for the plots is provided by a data frame with the same structure as the "CDF" output from `cont_analysis`. Plots are produced for every combination of `Type` of population, `Subpopulation` within `Type`, and `Indicator` (every combination of subpopulations, subpopulation levels, and variables).

Usage

```
cont_cdfplot(
  pdffile = "cdf2x2.pdf",
  cdfest,
  units_cdf = "Percent",
  ind_type = rep("Continuous", nind),
  log = rep("", nind),
  xlab = NULL,
  ylab = NULL,
  ylab_r = NULL,
  legloc = NULL,
  cdf_page = 4,
  width = 10,
  height = 8,
  confcut = 0,
  cex.main = 1.2,
  cex.legend = 1,
  ...
)
```

Arguments

`pdffile`	Name of the PDF file. The default is "cdf2x2.pdf".

cdfest	Data frame with the same structure as the "CDF" output from cont_analysis.
units_cdf	Indicator for the label utilized for the left side y-axis and the values used for the left side y-axis tick marks, where "Percent" means the label and values are in terms of percent of the population, and "Units" means the label and values are in terms of units (count, length, or area) of the population. The default is "Percent".
ind_type	Character vector consisting of the values "Continuous" or "Ordinal" that controls the type of CDF plot for each indicator. The default is "Continuous" for every indicator.
log	Character vector consisting of the values "" or "x" that controls whether the x axis uses the original scale ("") or the base 10 logarithmic scale ("x") for each indicator. The default is "" for every indicator.
xlab	Character vector consisting of the x-axis label for each indicator. If this argument equals NULL, then indicator names are used as the labels. The default is NULL.
ylab	Character string providing the left side y-axis label. If argument units_cdf equals "Units", a value should be provided for this argument. Otherwise, the label will be "Percent". The default is "Percent".
ylab_r	Character string providing the label for the right side y-axis (and, hence, determining the values used for the right side y-axis tick marks), where NULL means a right side y-axis is not created. If this argument equals "Same", the right side y-axis will have the same label and tick mark values as the left side y-axis. If this argument equals a character string other than "Same", the right side y-axis label will be the value provided for argument ylab_r, and the right side y-axis tick mark values will be determined by the choice not utilized for argument units_cdf, which means that the default value of argument units_cdf (i.e., "Percent") will result in the right side y-axis tick mark values being expressed in terms of units of the population (i.e., count, length, or area). The default is NULL.
legloc	Indicator for location of the plot legend, where "BR" means bottom right, "BL" means bottom left, "TR" means top right, "TL" means top left, and NULL means no legend. The default is NULL.
cdf_page	Number of CDF plots on each page, which must be chosen from the values: 1, 2, 4, or 6. The default is 4.
width	Width of the graphic region in inches. The default is 10.
height	Height of the graphic region in inches. The default is 8.
confcut	Numeric value that controls plotting confidence limits at the CDF extremes. Confidence limits for CDF values (percent scale) less than confcut or greater than 100 minus confcut are not plotted. A value of zero means confidence limits are plotted for the complete range of the CDF. The default is 0.
cex.main	Expansion factor for the plot title. The default is 1.2.
cex.legend	Expansion factor for the legend title. The default is 1.
...	Additional arguments passed to the cdf_plot function.

Value

A PDF file containing the CDF plots.

Author(s)

Tom Kincaid <Kincaid.Tom@epa.gov>

See Also

cdf_plot for plotting a cumulative distribution function (CDF)

cont_cdftest for CDF hypothesis testing

Examples

```
## Not run:
dframe <- data.frame(
  siteID = paste0("Site", 1:100),
  wgt = runif(100, 10, 100),
  xcoord = runif(100),
  ycoord = runif(100),
  stratum = rep(c("Stratum1", "Stratum2"), 50),
  ContVar = rnorm(100, 10, 1),
  All_Sites = rep("All Sites", 100),
  Resource_Class = rep(c("Good", "Poor"), c(55, 45))
)
myvars <- c("ContVar")
mysubpops <- c("All_Sites", "Resource_Class")
mypopsize <- data.frame(
  Resource_Class = c("Good", "Poor"),
  Total = c(4000, 1500)
)
myanalysis <- cont_analysis(dframe,
  vars = myvars, subpops = mysubpops,
  siteID = "siteID", weight = "wgt", xcoord = "xcoord", ycoord = "ycoord",
  stratumID = "stratum", popsize = mypopsize
)
cont_cdfplot("myanalysis.pdf", myanalysis$CDF, ylab_r = "Stream Length (km)")

## End(Not run)
```

---

cont_cdftest *Cumulative distribution function (CDF) inference for a probability survey*

Description

This function organizes input and output for conducting inference regarding cumulative distribution functions (CDFs) generated by a probability survey. For every response variable and every subpopulation (domain) variable, differences between CDFs are tested for every pair of subpopulations within the domain. Data input to the function can be either a single survey or multiple surveys (two or more). If the data contain multiple surveys, then the domain variables will reference those surveys and (potentially) subpopulations within those surveys. The inferential procedures divide the CDFs into a discrete set of intervals (classes) and then utilize procedures that have been developed for analysis of categorical data from probability surveys. Choices for inference are the Wald, adjusted Wald, Rao-Scott first order corrected (mean eigenvalue corrected), and Rao-Scott second order corrected (Satterthwaite corrected) test statistics. The default test statistic is the adjusted Wald statistic. The input data argument can be either a data frame or a simple features (sf) object. If an sf object is used, coordinates are extracted from the geometry column in the object, arguments xcoord and ycoord are assigned values "xcoord" and "ycoord", respectively, and the geometry column is dropped from the object.

Usage

```
cont_cdftest(
  dframe,
  vars,
  subpops = NULL,
  surveyID = NULL,
  siteID = "siteID",
  weight = "weight",
  xcoord = NULL,
  ycoord = NULL,
  stratumID = NULL,
  clusterID = NULL,
  weight1 = NULL,
  xcoord1 = NULL,
  ycoord1 = NULL,
  sizeweight = FALSE,
  sweight = NULL,
  sweight1 = NULL,
  fpc = NULL,
  popsize = NULL,
  vartype = "local",
  jointprob = "overton",
  testname = "adjWald",
  nclass = 3,
  subset_local = TRUE
)
```

Arguments

dframe	Data frame containing survey design variables, response variables, and subpopulation (domain) variables.

vars — Vector composed of character values that identify the names of response variables in the dframe data frame.

subpops — Vector composed of character values that identify the names of subpopulation (domain) variables in the dframe data frame. If a value is not provided, the value "All_Sites" is assigned to the subpops argument and a factor variable named "All_Sites" that takes the value "All Sites" is added to the dframe data frame. The default value is NULL.

surveyID — Character value providing name of the survey ID variable in the dframe data frame. If this argument equals NULL, then the dframe data frame contains data for a single survey. The default value is NULL.

siteID — Character value providing name of the site ID variable in the dframe data frame. For a two-stage sample, the site ID variable identifies stage two site IDs. The default value is "siteID".

weight — Character value providing name of the survey design weight variable in the dframe data frame. For a two-stage sample, the weight variable identifies stage two weights. The default value is "weight".

xcoord — Character value providing name of the x-coordinate variable in the dframe data frame. For a two-stage sample, the x-coordinate variable identifies stage two x-coordinates. Note that x-coordinates are required for calculation of the local mean variance estimator. The default value is NULL.

ycoord — Character value providing name of the y-coordinate variable in the dframe data frame. For a two-stage sample, the y-coordinate variable identifies stage two y-coordinates. Note that y-coordinates are required for calculation of the local mean variance estimator. The default value is NULL.

stratumID — Character value providing name of the stratum ID variable in the dframe data frame. The default value is NULL.

clusterID — Character value providing the name of the cluster (stage one) ID variable in the dframe data frame. Note that cluster IDs are required for a two-stage sample. The default value is NULL.

weight1 — Character value providing name of the stage one weight variable in the dframe data frame. The default value is NULL.

xcoord1 — Character value providing the name of the stage one x-coordinate variable in the dframe data frame. Note that x coordinates are required for calculation of the local mean variance estimator. The default value is NULL.

ycoord1 — Character value providing the name of the stage one y-coordinate variable in the dframe data frame. Note that y-coordinates are required for calculation of the local mean variance estimator. The default value is NULL.

sizeweight — Logical value that indicates whether size weights should be used during estimation, where TRUE uses size weights and FALSE does not use size weights. To employ size weights for a single-stage sample, a value must be supplied for argument weight. To employ size weights for a two-stage sample, values must be supplied for arguments weight and weight1. The default value is FALSE.

sweight — Character value providing the name of the size weight variable in the dframe data frame. For a two-stage sample, the size weight variable identifies stage two size weights. The default value is NULL.

sweight1 Character value providing name of the stage one size weight variable in the dframe data frame. The default value is NULL.

fpc Object that specifies values required for calculation of the finite population correction factor used during variance estimation. The object must match the survey design in terms of stratification and whether the design is single-stage or two-stage. For an unstratified design, the object is a vector. The vector is composed of a single numeric value for a single-stage design. For a two-stage unstratified design, the object is a named vector containing one more than the number of clusters in the sample, where the first item in the vector specifies the number of clusters in the population and each subsequent item specifies the number of stage two units for the cluster. The name for the first item in the vector is arbitrary. Subsequent names in the vector identify clusters and must match the cluster IDs. For a stratified design, the object is a named list of vectors, where names must match the strata IDs. For each stratum, the format of the vector is identical to the format described for unstratified single-stage and two-stage designs. Note that the finite population correction factor is not used with the local mean variance estimator.

Example fpc for a single-stage unstratified survey design:

```
fpc <- 15000
```

Example fpc for a single-stage stratified survey design:

```
fpc <- list(
    Stratum_1 = 9000,
    Stratum_2 = 6000)
```

Example fpc for a two-stage unstratified survey design:

```
fpc <- c(
    Ncluster = 150,
    Cluster_1 = 150,
    Cluster_2 = 75,
    Cluster_3 = 75,
    Cluster_4 = 125,
    Cluster_5 = 75)
```

Example fpc for a two-stage stratified survey design:

```
fpc <- list(
    Stratum_1 = c(
      Ncluster_1 = 100,
      Cluster_1 = 125,
      Cluster_2 = 100,
      Cluster_3 = 100,
      Cluster_4 = 125,
      Cluster_5 = 50),
    Stratum_2 = c(
      Ncluster_2 = 50,
      Cluster_1 = 75,
      Cluster_2 = 150,
      Cluster_3 = 75,
      Cluster_4 = 75,
```

```
      Cluster_5 = 125))
```

popsize — Object that provides values for the population argument of the calibrate or postStratify functions in the survey package. If a value is provided for popsize, then either the calibrate or postStratify function is used to modify the survey design object that is required by functions in the survey package. Whether to use the calibrate or postStratify function is dictated by the format of popsize, which is discussed below. Post-stratification adjusts the sampling and replicate weights so that the joint distribution of a set of post-stratifying variables matches the known population joint distribution. Calibration, generalized raking, or GREG estimators generalize post-stratification and raking by calibrating a sample to the marginal totals of variables in a linear regression model. For the calibrate function, the object is a named list, where the names identify factor variables in the dframe data frame. Each element of the list is a named vector containing the population total for each level of the associated factor variable. For the postStratify function, the object is either a data frame, table, or xtabs object that provides the population total for all combinations of selected factor variables in the dframe data frame. If a data frame is used for popsize, the variable containing population totals must be the last variable in the data frame. If a table is used for popsize, the table must have named dimnames where the names identify factor variables in the dframe data frame. If the popsize argument is equal to NULL, then neither calibration nor post-stratification is performed. The default value is NULL.

Example popsize for calibration:

```
popsize <- list(
    Ecoregion = c(
      East = 750,
      Central = 500,
      West = 250),
    Type = c(
      Streams = 1150,
      Rivers = 350))
```

Example popsize for post-stratification using a data frame:

```
popsize <- data.frame(
    Ecoregion = rep(c("East", "Central", "West"),
      rep(2, 3)),
    Type = rep(c("Streams", "Rivers"), 3),
    Total = c(575, 175, 400, 100, 175, 75))
```

Example popsize for post-stratification using a table:

```
popsize <- with(MySurveyFrame,
    table(Ecoregion, Type))
```

Example popsize for post-stratification using an xtabs object:

```
popsize <- xtabs(~Ecoregion + Type,
    data = MySurveyFrame)
```

vartype — Character value providing the choice of the variance estimator, where "Local" indicates the local mean estimator, "SRS" indicates the simple random sampling

estimator, "HT" indicates the Horvitz-Thompson estimator, and "YG" indicates the Yates-Grundy estimator.

jointprob — Character value providing the choice of joint inclusion probability approximation for use with Horvitz-Thompson and Yates-Grundy variance estimators, where "overton" indicates the Overton approximation, "hr" indicates the Hartley-Rao approximation, and "brewer" equals the Brewer approximation. The default value is "overton".

testname — Name of the test statistic to be reported in the output data frame. Choices for the name are: "Wald", "adjWald", "RaoScott_First", and "RaoScott_Second", which correspond to the Wald statistic, adjusted Wald statistic, Rao-Scott first-order corrected statistic, and Rao-Scott second-order corrected statistic, respectively. The default is "adjWald".

nclass — Number of classes into which the CDFs will be divided (binned), which must equal at least 2. The default is 3.

subset_local — Logical value that controls how the local mean variance estimator (the default vartype = "local") handles the two subpopulations being compared. See cat_analysis for the full explanation. subset_local is not yet supported for two-stage (clustered) samples. The default value is TRUE.

Value

Data frame of CDF test results for all pairs of subpopulations within each population type for every response variable. The data frame includes the test statistic specified by argument testname plus its degrees of freedom and p-value.

Author(s)

Tom Kincaid <Kincaid.Tom@epa.gov>

See Also

cdf_plot for visualizing CDF plots

cont_cdfplot for making CDF plots output to pdfs

Examples

```
n <- 200
mysiteID <- paste("Site", 1:n, sep = "")
dframe <- data.frame(
  siteID = mysiteID,
  wgt = runif(n, 10, 100),
  xcoord = runif(n),
  ycoord = runif(n),
  stratum = rep(c("Stratum1", "Stratum2"), n / 2),
  Resource_Class = sample(c("Agr", "Forest", "Urban"), n, replace = TRUE)
)
ContVar <- numeric(n)
tst <- dframe$Resource_Class == "Agr"
ContVar[tst] <- rnorm(sum(tst), 10, 1)
```

```
tst <- dframe$Resource_Class == "Forest"
ContVar[tst] <- rnorm(sum(tst), 10.1, 1)
tst <- dframe$Resource_Class == "Urban"
ContVar[tst] <- rnorm(sum(tst), 10.5, 1)
dframe$ContVar <- ContVar
myvars <- c("ContVar")
mysubpops <- c("Resource_Class")
mypopsize <- data.frame(
  Resource_Class = rep(c("Agr", "Forest", "Urban"), rep(2, 3)),
  stratum = rep(c("Stratum1", "Stratum2"), 3),
  Total = c(2500, 1500, 1000, 500, 600, 450)
)
cont_cdftest(dframe,
  vars = myvars, subpops = mysubpops, siteID = "siteID",
  weight = "wgt", xcoord = "xcoord", ycoord = "ycoord",
  stratumID = "stratum", popsize = mypopsize
)
```

`cov_panel_dsgn` *Create a covariance matrix for a panel design*

Description

Covariance structure accounts for the panel design and the four variance components: unit variation, period variation, unit by period interaction variation and index (or residual) variation. The model incorporates unit, period, unit by period, and index variance components. It also includes a provision for unit correlation and period autocorrelation.

Usage

```
cov_panel_dsgn(
  paneldsgn = matrix(50, 1, 10),
  nrepeats = 1,
  unit_var = NULL,
  period_var = NULL,
  unitperiod_var = NULL,
  index_var = NULL,
  unit_rho = 1,
  period_rho = 0
)
```

Arguments

`paneldsgn`	A matrix (dimensions: number of panels (rows) by number of periods (columns)) containing the number of units visited for each combination of panel and period. Default is matrix(50, 1, 10) which is a single panel of 50 units visited 10 times, typical time is a period.

nrepeats	Either NULL or a list of matrices the same length as paneldsgn specifying the number of revisits made to units in a panel in the same period for each design. Specifying NULL indicates that number of revisits to units is the same for all panels and for all periods and for all panel designs. The default is NULL, a single visit. Names must match list names in paneldsgn.
unit_var	The variance component estimate for unit. The default is NULL.
period_var	The variance component estimate for period The default is NULL.
unitperiod_var	The variance component estimate for unit by period interaction. The default is NULL.
index_var	The variance component estimate for index error. The default is NULL.
unit_rho	Unit correlation across periods. The default is 1.
period_rho	Period autocorrelation. The default is 0.

Details

Covariance structure accounts for the panel design and the four variance components: unit variation, period variation, unit by period interaction variation and index (or residual) variation. Uses the model structure defined by Urquhart 2012.

If nrepeats is NULL, then no units sampled more than once in a specific panel, period combination) and then unit by period and index variances are added together or user may have only estimated unit, period and unit by period variance components so that index component is zero. It calculates the covariance matrix for the simple linear regression. The standard error for a linear trend coefficient is the square root of the variance.

Value

A list containing the covariance matrix (cov) for the panel design, the input panel design (paneldsgn), the input nrepeats design (nrepeats.dsgn) and the function call.

Author(s)

Tony Olsen <Olsen.Tony@epa.gov>

References

Urquhart, N. S., W. S. Overton, et al. (1993) Comparing sampling designs for monitoring ecological status and trends: impact of temporal patterns. In: *Statistics for the Environment.* V. Barnett and K. F. Turkman. John Wiley & Sons, New York, pp. 71-86.

Urquhart, N. S. and T. M. Kincaid (1999). Designs for detecting trends from repeated surveys of ecological resources. *Journal of Agricultural, Biological, and Environmental Statistics*, **4(4)**, 404-414.

Urquhart, N. S. (2012). The role of monitoring design in detecting trend in long-term ecological monitoring studies. In: *Design and Analysis of Long-term Ecological Monitoring Studies.* R. A. Gitzen, J. J. Millspaugh, A. B. Cooper, and D. S. Licht (eds.). Cambridge University Press, New York, pp. 151-173.

See Also

power_dsgn for power calculations of multiple panel designs

`diffrisk_analysis` *Risk difference analysis*

Description

This function organizes input and output for risk difference analysis (of categorical variables). The analysis data, `dframe`, can be either a data frame or a simple features (`sf`) object. If an `sf` object is used, coordinates are extracted from the geometry column in the object, arguments `xcoord` and `ycoord` are assigned values `"xcoord"` and `"ycoord"`, respectively, and the geometry column is dropped from the object.

Usage

```
diffrisk_analysis(
  dframe,
  vars_response,
  vars_stressor,
  response_levels = NULL,
  stressor_levels = NULL,
  subpops = NULL,
  siteID = NULL,
  weight = "weight",
  xcoord = NULL,
  ycoord = NULL,
  stratumID = NULL,
  clusterID = NULL,
  weight1 = NULL,
  xcoord1 = NULL,
  ycoord1 = NULL,
  sizeweight = FALSE,
  sweight = NULL,
  sweight1 = NULL,
  fpc = NULL,
  popsize = NULL,
  vartype = "local",
  conf = 95,
  All_Sites = FALSE,
  subset_local = TRUE
)
```

Arguments

`dframe`	Data to be analyzed (analysis data). A data frame or `sf` object containing survey design variables, response variables, stressor variables, and subpopulation (domain) variables.

vars_response Vector composed of character values that identify the names of response variables in dframe. Each response variable must have two category values (levels), where one level is associated with poor condition and the other level is associated with good condition.

vars_stressor Vector composed of character values that identify the names of stressor variables in dframe. Each stressor variable must have two category values (levels), where one level is associated with poor condition and the other level is associated with good condition.

response_levels

List providing the category values (levels) for each element in the vars_response argument. Each element in the list must contain two values, where the first value identifies poor condition, and the second value identifies good condition. This argument must be named and must be the same length as argument vars_response. Names for this argument must match the values in the vars_response argument. If this argument equals NULL, then a named list is created that contains the values "Poor" and "Good" for the first and second levels, respectively, of each element in the vars_response argument and that uses values in the vars_response argument as names for the list. If response_levels is provided without names, then the names of response_levels are set to vars_response. The default value is NULL.

stressor_levels

List providing the category values (levels) for each element in the vars_stressor argument. Each element in the list must contain two values, where the first value identifies poor condition, and the second value identifies good condition. This argument must be named and must be the same length as argument vars_stressor. Names for this argument must match the values in the vars_stressor argument. If this argument equals NULL, then a named list is created that contains the values "Poor" and "Good" for the first and second levels, respectively, of each element in the vars_stressor argument and that uses values in the vars_stressor argument as names for the list. If stressor_levels is provided without names, then the names of stressor_levels are set to vars_stressor. The default value is NULL.

subpops Vector composed of character values that identify the names of subpopulation (domain) variables in dframe. If a value is not provided, the value "All_Sites" is assigned to the subpops argument and a factor variable named "All_Sites" that takes the value "All Sites" is added to dframe. The default value is NULL.

siteID Character value providing the name of the site ID variable in dframe. For a two-stage sample, the site ID variable identifies stage two site IDs. The default value is NULL, which assumes that each row in dframe represents a unique site.

weight Character value providing the name of the design weight variable in dframe. For a two-stage sample, the weight variable identifies stage two weights. The default value is "weight".

xcoord Character value providing name of the x-coordinate variable in dframe. For a two-stage sample, the x-coordinate variable identifies stage two x-coordinates. Note that x-coordinates are required for calculation of the local mean variance estimator. If dframe is an sf object, this argument is not required (as the ge-

ometry column in dframe is used to find the x-coordinate). The default value is NULL.

ycoord — Character value providing name of the y-coordinate variable in dframe. For a two-stage sample, the y-coordinate variable identifies stage two y-coordinates. Note that y-coordinates are required for calculation of the local mean variance estimator. If dframe is an sf object, this argument is not required (as the geometry column in dframe is used to find the t-coordinate). The default value is NULL.

stratumID — Character value providing the name of the stratum ID variable in dframe. The default value is NULL.

clusterID — Character value providing the name of the cluster (stage one) ID variable in dframe. Note that cluster IDs are required for a two-stage sample. The default value is NULL.

weight1 — Character value providing the name of the stage one weight variable in dframe. The default value is NULL.

xcoord1 — Character value providing the name of the stage one x-coordinate variable in dframe. Note that x coordinates are required for calculation of the local mean variance estimator. The default value is NULL.

ycoord1 — Character value providing the name of the stage one y-coordinate variable in dframe. Note that y-coordinates are required for calculation of the local mean variance estimator. The default value is NULL.

sizeweight — Logical value that indicates whether size weights should be used during estimation, where TRUE uses size weights and FALSE does not use size weights. To employ size weights for a single-stage sample, a value must be supplied for argument weight. To employ size weights for a two-stage sample, values must be supplied for arguments weight and weight1. The default value is FALSE.

sweight — Character value providing the name of the size weight variable in dframe. For a two-stage sample, the size weight variable identifies stage two size weights. The default value is NULL.

sweight1 — Character value providing the name of the stage one size weight variable in dframe. The default value is NULL.

fpc — Object that specifies values required for calculation of the finite population correction factor used during variance estimation. The object must match the survey design in terms of stratification and whether the design is single-stage or two-stage. For an unstratified design, the object is a vector. The vector is composed of a single numeric value for a single-stage design. For a two-stage unstratified design, the object is a named vector containing one more than the number of clusters in the sample, where the first item in the vector specifies the number of clusters in the population and each subsequent item specifies the number of stage two units for the cluster. The name for the first item in the vector is arbitrary. Subsequent names in the vector identify clusters and must match the cluster IDs. For a stratified design, the object is a named list of vectors, where names must match the strata IDs. For each stratum, the format of the vector is identical to the format described for unstratified single-stage and two-stage designs. Note that the finite population correction factor is not used with the local mean variance estimator.

Example fpc for a single-stage unstratified survey design:

```
fpc <- 15000
```

Example fpc for a single-stage stratified survey design:

```
fpc <- list(
    Stratum_1 = 9000,
    Stratum_2 = 6000)
```

Example fpc for a two-stage unstratified survey design:

```
fpc <- c(
    Ncluster = 150,
    Cluster_1 = 150,
    Cluster_2 = 75,
    Cluster_3 = 75,
    Cluster_4 = 125,
    Cluster_5 = 75)
```

Example fpc for a two-stage stratified survey design:

```
fpc <- list(
    Stratum_1 = c(
      Ncluster_1 = 100,
      Cluster_1 = 125,
      Cluster_2 = 100,
      Cluster_3 = 100,
      Cluster_4 = 125,
      Cluster_5 = 50),
    Stratum_2 = c(
      Ncluster_2 = 50,
      Cluster_1 = 75,
      Cluster_2 = 150,
      Cluster_3 = 75,
      Cluster_4 = 75,
      Cluster_5 = 125))
```

`popsize` Object that provides values for the population argument of the `calibrate` or `postStratify` functions in the survey package. If a value is provided for popsize, then either the `calibrate` or `postStratify` function is used to modify the survey design object that is required by functions in the survey package. Whether to use the `calibrate` or `postStratify` function is dictated by the format of popsize, which is discussed below. Post-stratification adjusts the sampling and replicate weights so that the joint distribution of a set of post-stratifying variables matches the known population joint distribution. Calibration, generalized raking, or GREG estimators generalize post-stratification and raking by calibrating a sample to the marginal totals of variables in a linear regression model. For the `calibrate` function, the object is a named list, where the names identify factor variables in `dframe`. Each element of the list is a named vector containing the population total for each level of the associated factor variable. For the `postStratify` function, the object is either a data frame, table, or xtabs object that provides the population total for all combi-

nations of selected factor variables in the dframe data frame. If a data frame is used for popsize, the variable containing population totals must be the last variable in the data frame. If a table is used for popsize, the table must have named dimnames where the names identify factor variables in the dframe data frame. If the popsize argument is equal to NULL, then neither calibration nor post-stratification is performed. The default value is NULL.

Example popsize for calibration:

```
popsize <- list(
    Ecoregion = c(
      East = 750,
      Central = 500,
      West = 250),
    Type = c(
      Streams = 1150,
      Rivers = 350))
```

Example popsize for post-stratification using a data frame:

```
popsize <- data.frame(
    Ecoregion = rep(c("East", "Central", "West"),
      rep(2, 3)),
    Type = rep(c("Streams", "Rivers"), 3),
    Total = c(575, 175, 400, 100, 175, 75))
```

Example popsize for post-stratification using a table:

```
popsize <- with(MySurveyFrame,
    table(Ecoregion, Type))
```

Example popsize for post-stratification using an xtabs object:

```
popsize <- xtabs(~Ecoregion + Type,
    data = MySurveyFrame)
```

vartype — Character value providing the choice of the variance estimator, where "Local" indicates the local mean estimator and "SRS" indicates the simple random sampling estimator. The default value is "local".

conf — Numeric value providing the Gaussian-based confidence level. The default value is 95.

All_Sites — A logical variable used when subpops is not NULL. If All_Sites is TRUE, then alongside the subpopulation output, output for all sites (ignoring subpopulations) is returned for each variable in vars. If All_Sites is FALSE, then alongside the subpopulation output, output for all sites (ignoring subpopulations) is not returned for each variable in vars. The default is FALSE.

subset_local — When 'subset_local = TRUE' (the default), subpopulations are subset to include only subpopulation members prior to variance estimation. When 'subset_local = FALSE', subpopulations are not subset prior to variance estimation. 'subset_local = FALSE' follows standard design-based subpopulation (i.e., domain) estimation theory but is computationally more complex, requiring the neighborhood weights matrix of the full data (which scales cubically with the sample size). 'subset_local' only applies when 'vartype = "local"'.

Value

The analysis results. A data frame of population estimates for all combinations of subpopulations, categories within each subpopulation, response variables, and categories within each response variable. Estimates are provided for proportion and size of the population plus standard error, margin of error, and confidence interval estimates. The data frame contains the following variables:

Type subpopulation (domain) name

Subpopulation subpopulation name within a domain

Response response variable

Stressor stressor variable

nResp sample size

Estimate risk difference estimate

Estimate_StressPoor risk estimate for poor condition stressor

Estimate_StressGood risk estimate for good condition stressor

StdError risk difference standard error

MarginofError risk difference margin of error

LCBxxPct xx% (default 95%) lower confidence bound

UCBxxPct xx% (default 95%) upper confidence bound

WeightTotal sum of design weights

Count_RespPoor_StressPoor number of observations in the poor response and poor stressor group

Count_RespPoor_StressGood number of observations in the poor response and good stressor group

Count_RespGood_StressPoor number of observations in the good response and poor stressor group

Count_RespGood_StressGood number of observations in the good response and good stressor group

Prop_RespPoor_StressPoor weighted proportion of observations in the poor response and poor stressor group

Prop_RespPoor_StressGood weighted proportion of observations in the poor response and good stressor group

Prop_RespGood_StressPoor weighted proportion of observations in the good response and poor stressor group

Prop_RespGood_StressGood weighted proportion of observations in the good response and good stressor group

Details

Risk difference measures the absolute strength of association between conditional probabilities defined for a response variable and a stressor variable, where the response and stressor variables are classified as either good (i.e., reference condition) or poor (i.e., different from reference condition). Risk difference is defined as the difference between two conditional probabilities: the probability that the response variable is in poor condition given that the stressor variable is in poor condition

and the probability that the response variable is in poor condition given that the stressor variable is in good condition. Risk difference values close to zero indicate that the stressor variable has little or no impact on the probability that the response variable is in poor condition. Risk difference values much greater than zero indicate that the stressor variable has a significant impact on the probability that the response variable is in poor condition.

Author(s)

Tom Kincaid <Kincaid.Tom@epa.gov>

See Also

attrisk_analysis for attributable risk analysis

relrisk_analysis for relative risk analysis

Examples

```
dframe <- data.frame(
  siteID = paste0("Site", 1:100),
  wgt = runif(100, 10, 100),
  xcoord = runif(100),
  ycoord = runif(100),
  stratum = rep(c("Stratum1", "Stratum2"), 50),
  RespVar1 = sample(c("Poor", "Good"), 100, replace = TRUE),
  RespVar2 = sample(c("Poor", "Good"), 100, replace = TRUE),
  StressVar = sample(c("Poor", "Good"), 100, replace = TRUE),
  All_Sites = rep("All Sites", 100),
  Resource_Class = rep(c("Agr", "Forest"), c(55, 45))
)
myresponse <- c("RespVar1", "RespVar2")
mystressor <- c("StressVar")
mysubpops <- c("All_Sites", "Resource_Class")
diffrisk_analysis(dframe,
  vars_response = myresponse,
  vars_stressor = mystressor, subpops = mysubpops, siteID = "siteID",
  weight = "wgt", xcoord = "xcoord", ycoord = "ycoord",
  stratumID = "stratum"
)
```

---

errorprnt *Print errors from analysis functions*

Description

This function prints the error messages vector in the analysis functions.

Usage

```
errorprnt(error_vec = get("error_vec", envir = .GlobalEnv))
```

Arguments

error_vec	Data frame that contains error messages. The default is "error_vec", which is the name given to the error messages vector created by functions in the spsurvey package.

Value

Printed errors.

Author(s)

Tom Kincaid <Kincaid.Tom@epa.gov>

---

grts *Select a generalized random tessellation stratified (GRTS) sample*

Description

Select a spatially balanced sample from a point (finite), linear / linestring (infinite), or areal / polygon (infinite) sampling frame using the Generalized Random Tessellation Stratified (GRTS) algorithm. The GRTS algorithm accommodates unstratified and stratified sampling designs and allows for equal inclusion probabilities, unequal inclusion probabilities according to a categorical variable, and inclusion probabilities proportional to a positive auxiliary variable. Several additional sampling options are included, such as including legacy (historical) sites, requiring a minimum distance between sites, and selecting replacement sites. For technical details, see Stevens and Olsen (2004).

Usage

```
grts(
  sframe,
  n_base,
  stratum_var = NULL,
  seltype = NULL,
  caty_var = NULL,
  caty_n = NULL,
  aux_var = NULL,
  legacy_var = NULL,
  legacy_sites = NULL,
  legacy_stratum_var = NULL,
  legacy_caty_var = NULL,
  legacy_aux_var = NULL,
  mindis = NULL,
  maxtry = 10,
  n_over = NULL,
  n_near = NULL,
  wgt_units = NULL,
  pt_density = NULL,
```

```
  DesignID = "Site",
  SiteBegin = 1,
  sep = "-",
  projcrs_check = TRUE
)
```

Arguments

sframe	A sampling frame as an sf object. The coordinate system for sframe must projected (not geographic). If m or z values are in sframe's geometry, they are silently dropped (i.e., only x-coordinates and y-coordinates are preserved).
n_base	The base sample size required. If the sampling design is unstratified, this is a single numeric value. If the sampling design is stratified, this is a named vector or list whose names represent each stratum and whose values represent each stratum's sample size. These names must match the values of the stratification variable represented by stratum_var. Legacy sites are considered part of the base sample, so the value for n_base should be equal to the number of legacy sites plus the number of desired non-legacy sites.
stratum_var	A character string containing the name of the column from sframe that identifies stratum membership for each element in sframe. If stratum equals NULL, the sampling design is unstratified and all elements in sframe are eligible to be selected in the sample. The default is NULL.
seltype	A character string or vector indicating the inclusion probability type, which must be one of following: "equal" for equal inclusion probabilities; "unequal" for unequal inclusion probabilities according to a categorical variable specified by caty_var; and "proportional" for inclusion probabilities proportional to a positive auxiliary variable specified by aux_var. If the sampling design is unstratified, seltype is a single character vector. If the sampling design is stratified, seltype is a named vector whose names represent each stratum and whose values represent each stratum's inclusion probability type. seltype's default value tries to match the intended inclusion probability type: If caty_var and aux_var are not specified, seltype is "equal"; if caty_var is specified, seltype is "unequal"; and if aux_var is specified, seltype is "proportional".
caty_var	A character string containing the name of the column from sframe that represents the unequal probability variable.
caty_n	A character vector indicating the expected sample size for each level of caty_var, the unequal probability variable. If the sampling design is unstratified, caty_n is a named vector whose names represent each level of caty_var and whose values represent each level's expected sample size. The sum of caty_n must equal n_base. If the sampling design is stratified and the expected sample sizes are the same among strata, caty_n is a named vector whose names represent represent each level of caty_var and whose values represent each level's expected sample size – these expected sample sizes are applied to all strata. The sum of caty_n must equal each stratum's value in n_base. If the sampling design is stratified and the expected sample sizes differ among strata, caty_n is a list where each element is named as a stratum in n_base. Each stratum's list

element is a named vector whose names represent each level of caty_var and whose values represent each level's expected sample size (within the stratum). The sum of the values in each stratum's list element must equal that stratum's value in n_base.

aux_var A character string containing the name of the column from sframe that represents the proportional (to size) inclusion probability variable (auxiliary variable). This auxiliary variable must be positive, and the resulting inclusion probabilities are proportional to the values of the auxiliary variable. Larger values of the auxiliary variable result in higher inclusion probabilities.

legacy_var This argument can be used instead of legacy_sites when sframe is a POINT or MULTIPOINT geometry (i.e. a finite sampling frame), When legacy_var is used, it is a character string containing the name of the column from sframe that represents whether each site is a legacy site. For legacy sites, the values of the legacy_var must contain character strings that act as a legacy site identifier. For non-legacy sites, the values of the legacy_var column must be NA. Using this approach, legacy_stratum_var, legacy_caty_var, and legacy_aux_var are not required and should not be used (because legacy_var represents a column in sframe). spsurvey assumes that the legacy sites were selected from a previous sampling design that incorporated randomness into site selection and that the legacy sites are elements of the current sampling frame.

legacy_sites An sf object with a POINT or MULTIPOINT geometry representing the legacy sites. spsurvey assumes that the legacy sites were selected from a previous sampling design that incorporated randomness into site selection and that the legacy sites are elements of the current sampling frame. If sframe has a POINT or MULTIPOINT geometry, the observations in legacy_sites should not also be in sframe (i.e., duplicates are not removed). Thus, sframe and legacy_sites together compose the current sampling frame. If m or z values are in legacy_sites' geometry, they are silently dropped (i.e., only x-coordinates and y-coordinates are preserved).

legacy_stratum_var

A character string containing the name of the column from legacy_sites that identifies stratum membership for each element of legacy_sites. This argument is required when the sampling design is stratified and its levels must be contained in the levels of the stratum_var variable. The default value of legacy_stratum_var is stratum_var, so legacy_stratum_var need only be specified explicitly when the name of the stratification variable in legacy_sites differs from stratum_var.

legacy_caty_var

A character string containing the name of the column from legacy_sites that identifies the unequal probability variable for each element of legacy_sites. This argument is required when the sampling design uses unequal selection probabilities and its categories must be contained in the levels of the caty_var variable. The default value of legacy_caty_var is caty_var, so legacy_caty_var need only be specified explicitly when the name of the unequal probability variable in legacy_sites differs from caty_var.

legacy_aux_var A character string containing the name of the column from legacy_sites that identifies the proportional probability variable for each element of legacy_sites.

This argument is required when the sampling design uses proportional selection probabilities and the values of the legacy_aux_var variable must be positive. The default value of legacy_aux_var is aux_var, so legacy_aux_var need only be specified explicitly when the name of the proportional probability variable in legacy_sites differs from aux_var.

mindis — A numeric value indicating the desired minimum distance between sampled sites. If the sampling design is stratified and mindis is an numeric value, the minimum distance is applied to all strata. If the sampling design is stratified and different minimum distances are desired among strata, then mindis is a list whose names match the names of n_base and whose and values are the minimum distance for the corresponding stratum. If a minimum distance is not desired for a particular stratum, then the corresponding value in mindis should be 0 or NULL (which is equivalent to 0). The units of mindis must represent the units in sframe. A warning is returned if the minimum distance could not be reached after maxtry attempts. If legacy sites are used, the minimum distance requirement (and subsequent warning if maxtry attempts are reached) is enforced for all base sites that are not legacy sites (i.e., the minimum distance is enforced for these sites by comparing distances against all base sites (legacy and non-legacy)).

maxtry — The number of maximum attempts to apply the minimum distance algorithm to obtain the desired minimum distance between sites. Each iteration takes roughly as long as the standard GRTS algorithm. Successive iterations will always contain at least as many sites satisfying the minimum distance requirement as the previous iteration. The algorithm stops when the minimum distance requirement is met or there are maxtry iterations. The default number of maximum iterations is 10.

n_over — The number of reverse hierarchically ordered (rho) replacement sites. If the sampling design is unstratified, then n_over is an integer specifying the number of rho replacement sites desired. If the sampling design is stratified, then n_over is a vector (or list) whose names match the names of n_base and whose values indicate the number of rho replacement sites for each stratum. If replacement sites are not desired for a particular stratum, then the corresponding value in n_over should be 0 or NULL (which is equivalent to 0). If the sampling design is stratified but the number of n_over sites is the same in each stratum, n_over can be a vector which is used for each stratum. If n_over is an unnamed, length-one vector, it's value is recycled and used for each stratum. Note that if the sampling design has unequal selection probabilities (seltype = "unequal"), then n_over sites are given the same proportion of caty_n values as n_base.

n_near — The number of nearest neighbor (nn) replacement sites. If the sampling design is unstratified, n_near is integer from 1 to 10 specifying the number of nn replacement sites to be selected for each base site. If the sampling design is stratified but the same number of nn replacement sites is desired for each stratum, n_near is integer from 1 to 10 specifying the number of nn replacement sites to be selected for each base site. If the sampling design is unstratified and a different number of nn replacement sites is desired for each stratum, n_near is a vector (or list) whose names represent strata and whose values is integer from 1 to 10 specifying the number of nn replacement sites to be selected for each base site in the stratum. If replacement sites are not desired for a particular stratum, then

the corresponding value in n_over should be 0 or NULL (which is equivalent to 0). For infinite sampling frames, the distance between a site and its nn depends on pt_density. The larger pt_density, the closer the nn neighbors.

wgt_units — The units used to compute the design weights. These units must be standard units as defined by the set_units() function in the units package. The default units match the units of the sf object.

pt_density — A positive integer controlling the density of the GRTS approximation for infinite sampling frames. The GRTS approximation for infinite sample frames vastly improves computational efficiency by generating many finite points and selecting a sample from the points. pt_density represents the density of finite points per unit to use in the approximation. More specifically, for each stratum, the number of points used in the approximation equals pt_density * (n_base + n_over). A larger value of pt_density means a closer approximation to the infinite sampling frame but less computational efficiency. The default value of pt_density is 10. Note that when used with caty_n, the unequal inclusion probabilities generated from this approach are also approximations.

DesignID — A character string indicating the naming structure for each site's identifier selected in the sample, which is matched with SiteBegin and included as a variable in the sf object in the function's output. Default is "Site".

SiteBegin — A character string indicating the first number to use to match with DesignID while creating each site's identifier selected in the sample. Successive sites are given successive integers. The default starting number is 1 and the number of digits is equal to number of digits in nbase + nover. For example, if nbase is 50 and nover is 0, then the default site identifiers are Site-01 to Site-50

sep — A character string that acts as a separator between DesignID and SiteBegin. The default is "-".

projcrs_check — A check for whether the coordinates are projected. If TRUE, an error is returned if coordinates are not projected (i.e., they are geographic or NA). If FALSE, the check is not performed, which means that the crs in sframe (and legacy_sites if provided) can be projected, geographic, or NA.

Details

n_base is the number of sites used to calculate the design weights, which is typically the number of sites used in an analysis. When a panel sampling design is implemented, n_base is typically the number of sites in all panels that will be sampled in the same temporal period – n_base is not the total number of sites in all panels. The sum of n_base and n_over is equal to the total number of sites to be visited for all panels plus any replacement sites that may be required.

Value

The sampling design sites and additional information about the sampling design. More specifically, it is, a list with five elements:

- sites_legacy An sf object containing legacy sites. This is NULL if legacy sites were not included in the sample.
- sites_base An sf object containing the base sites. This is NULL if n_base equals the number of legacy sites.

- sites_over An sf object containing the reverse hierarchically ordered replacement sites. This is NULL if no reverse hierarchically ordered replacement sites were included in the sample.
- sites_near An sf object containing the nearest neighbor replacement sites. This is NULL if no nearest neighbor replacement sites were included in the sample.
- design A list documenting the specifications of this sampling design. This can be checked to verify your sampling design ran as intended.
  - call The original function call.
  - stratum_var The name of the stratification variable in sframe. This equals NULL if no stratification is used.
  - stratum The unique strata. This equals "None" if the sampling design is unstratified.
  - n_base The base sample size per stratum.
  - seltype The selection type per stratum.
  - caty_var The name of the unequal probability variable in sframe. This equals NULL if no unequal probability variable is used.
  - caty_n The expected sample sizes for each level of the unequal probability grouping variable per stratum. This equals NULL when seltype is not "unequal".
  - aux_var The name of the proportional probability (auxiliary) variable in sframe. This equals NULL if no proportional probability variable is used.
  - legacy A logical variable indicating whether legacy sites were included in the sample.
  - legacy_stratum_var The name of the stratification variable in legacy_sites. Omitted if legacy sites are not used. This equals NULL if legacy sites were used but no stratification variable is used.
  - legacy_caty_var The name of the unequal probability variable in legacy_sites. Omitted if legacy sites are not used. This equals NULL if legacy sites were used but no unequal probability variable is used.
  - legacy_aux_var The name of the proportional probability (auxiliary) variable in legacy_sites. Omitted if legacy sites are not used. This equals NULL if legacy sites were used but no proportional probability variable is used.
  - mindis The minimum distance requirement desired. This is NULL when no minimum distance requirement was applied.
  - n_over The reverse hierarchically ordered replacement site sample sizes per stratum. If seltype is unequal, this represents the expected sample sizes. This is NULL when no reverse hierarchically ordered replacement sites were selected.
  - n_near The number of nearest neighbor replacement sites desired. This is NULL when no nearest neighbor replacement sites were selected.

When non-NULL, the sites_legacy, sites_base, sites_over, and sites_near objects contain the original columns in sframe and include a few additional columns. These additional columns are

- siteID A site identifier (as named using the DesignID and SiteBegin arguments to grts()).
- siteuse Whether the site is a legacy site (Legacy), base site (Base), reverse hierarchically ordered replacement site (Over), or nearest neighbor replacement site (Near).
- replsite The replacement site ordering. replsite is None if the site is not a replacement site, Next if it is the next reverse hierarchically ordered replacement site to use, or Near_, where the word following _ indicates the ordering of sites closest to the originally sampled site.

- lon_WGS84 Longitude coordinates using the WGS84 coordinate system (EPSG:4326). Only given if coordinates are projected.
- lat_WGS84 Latitude coordinates using the WGS84 coordinate system (EPSG:4326). Only given if coordinates are projected.
- X Longitude coordinates using the provided coordinate system. Only given if coordinates are not projected (i.e., they are geographic or NA).
- Y Latitude coordinates using the provided coordinate system. Only given if coordinates are not projected (i.e., they are geographic or NA).
- stratum A stratum indicator. stratum is None if the sampling design was unstratified. If the sampling design was stratified, stratum indicates the stratum.
- wgt The design weight.
- ip The site's original inclusion probability (the reciprocal) of (wgt).
- caty An unequal probability grouping indicator. caty is None if the sampling design did not use unequal inclusion probabilities. If the sampling design did use unequal inclusion probabilities, caty indicates the unequal probability level.
- aux The auxiliary proportional probability variable. This column is only returned if seltype was proportional in the original sampling design.

If any columns in sframe contain these names, those columns from sframe will be automatically prefixed with sframe_ in the sites object. When output is printed, a summary of site counts by the levels in stratum_var and caty_var is shown.

Author(s)

Tony Olsen <olsen.tony@epa.gov>

References

Stevens Jr., Don L. and Olsen, Anthony R. (2004). Spatially balanced sampling of natural resources. *Journal of the American Statistical Association*, 99(465), 262-278.

See Also

irs to select a sample that is not spatially balanced

Examples

```
## Not run:
samp <- grts(NE_Lakes, n_base = 100)
print(samp)
strata_n <- c(low = 25, high = 30)
samp_strat <- grts(NE_Lakes, n_base = strata_n, stratum_var = "ELEV_CAT")
print(samp_strat)
samp_over <- grts(NE_Lakes, n_base = 30, n_over = 5)
print(samp_over)

## End(Not run)
```

Illinois_River *Illinois River data*

Description

An (sf) MULTILINESTRING object of 244 segments of the Illinois River in Arkansas and Oklahoma.

Usage

```
Illinois_River
```

Format

244 rows and 2 variables:

STATE_NAME State name.

geometry MULTILINESTRING geometry using the NAD83 / Conus Albers coordinate reference system (EPSG: 5070).

Illinois_River_Legacy *Illinois River legacy data*

Description

An (sf) POINT object of legacy sites for the Illinois River data.

Usage

```
Illinois_River_Legacy
```

Format

5 rows and 2 variables:

STATE_NAME State name.

geometry POINT geometry using the NAD83 / Conus Albers coordinate reference system (EPSG: 5070).

irs *Select an independent random sample (IRS)*

Description

Select a sample that is not spatially balanced from a point (finite), linear / linestring (infinite), or areal / polygon (infinite) sampling frame using the Independent Random Sampling (IRS) algorithm. The IRS algorithm accommodates unstratified and stratified sampling designs and allows for equal inclusion probabilities, unequal inclusion probabilities according to a categorical variable, and inclusion probabilities proportional to a positive auxiliary variable. Several additional sampling options are included, such as including legacy (historical) sites, requiring a minimum distance between sites, and selecting replacement sites.

Usage

```
irs(
  sframe,
  n_base,
  stratum_var = NULL,
  seltype = NULL,
  caty_var = NULL,
  caty_n = NULL,
  aux_var = NULL,
  legacy_var = NULL,
  legacy_sites = NULL,
  legacy_stratum_var = NULL,
  legacy_caty_var = NULL,
  legacy_aux_var = NULL,
  mindis = NULL,
  maxtry = 10,
  n_over = NULL,
  n_near = NULL,
  wgt_units = NULL,
  pt_density = NULL,
  DesignID = "Site",
  SiteBegin = 1,
  sep = "-",
  projcrs_check = TRUE
)
```

Arguments

sframe	A sampling frame as an sf object. The coordinate system for sframe must projected (not geographic). If m or z values are in sframe's geometry, they are silently dropped (i.e., only x-coordinates and y-coordinates are preserved).
n_base	The base sample size required. If the sampling design is unstratified, this is a single numeric value. If the sampling design is stratified, this is a named vector

or list whose names represent each stratum and whose values represent each stratum's sample size. These names must match the values of the stratification variable represented by `stratum_var`. Legacy sites are considered part of the base sample, so the value for `n_base` should be equal to the number of legacy sites plus the number of desired non-legacy sites.

`stratum_var` A character string containing the name of the column from `sframe` that identifies stratum membership for each element in `sframe`. If stratum equals `NULL`, the sampling design is unstratified and all elements in `sframe` are eligible to be selected in the sample. The default is `NULL`.

`seltype` A character string or vector indicating the inclusion probability type, which must be one of following: `"equal"` for equal inclusion probabilities; `"unequal"` for unequal inclusion probabilities according to a categorical variable specified by `caty_var`; and `"proportional"` for inclusion probabilities proportional to a positive auxiliary variable specified by `aux_var`. If the sampling design is unstratified, `seltype` is a single character vector. If the sampling design is stratified, `seltype` is a named vector whose names represent each stratum and whose values represent each stratum's inclusion probability type. `seltype`'s default value tries to match the intended inclusion probability type: If `caty_var` and `aux_var` are not specified, `seltype` is `"equal"`; if `caty_var` is specified, `seltype` is `"unequal"`; and if `aux_var` is specified, `seltype` is `"proportional"`.

`caty_var` A character string containing the name of the column from `sframe` that represents the unequal probability variable.

`caty_n` A character vector indicating the expected sample size for each level of `caty_var`, the unequal probability variable. If the sampling design is unstratified, `caty_n` is a named vector whose names represent each level of `caty_var` and whose values represent each level's expected sample size. The sum of `caty_n` must equal `n_base`. If the sampling design is stratified and the expected sample sizes are the same among strata, `caty_n` is a named vector whose names represent represent each level of `caty_var` and whose values represent each level's expected sample size – these expected sample sizes are applied to all strata. The sum of `caty_n` must equal each stratum's value in `n_base`. If the sampling design is stratified and the expected sample sizes differ among strata, `caty_n` is a list where each element is named as a stratum in `n_base`. Each stratum's list element is a named vector whose names represent each level of `caty_var` and whose values represent each level's expected sample size (within the stratum). The sum of the values in each stratum's list element must equal that stratum's value in `n_base`.

`aux_var` A character string containing the name of the column from `sframe` that represents the proportional (to size) inclusion probability variable (auxiliary variable). This auxiliary variable must be positive, and the resulting inclusion probabilities are proportional to the values of the auxiliary variable. Larger values of the auxiliary variable result in higher inclusion probabilities.

`legacy_var` This argument can be used instead of `legacy_sites` when `sframe` is a `POINT` or `MULTIPOINT` geometry (i.e. a finite sampling frame), When `legacy_var` is used, it is a character string containing the name of the column from `sframe` that represents whether each site is a legacy site. For legacy sites, the values of

the legacy_var must contain character strings that act as a legacy site identifier. For non-legacy sites, the values of the legacy_var column must be NA. Using this approach, legacy_stratum_var, legacy_caty_var, and legacy_aux_var are not required and should not be used (because legacy_var represents a column in sframe). spsurvey assumes that the legacy sites were selected from a previous sampling design that incorporated randomness into site selection and that the legacy sites are elements of the current sampling frame.

legacy_sites An sf object with a POINT or MULTIPOINT geometry representing the legacy sites. spsurvey assumes that the legacy sites were selected from a previous sampling design that incorporated randomness into site selection and that the legacy sites are elements of the current sampling frame. If sframe has a POINT or MULTIPOINT geometry, the observations in legacy_sites should not also be in sframe (i.e., duplicates are not removed). Thus, sframe and legacy_sites together compose the current sampling frame. If m or z values are in legacy_sites' geometry, they are silently dropped (i.e., only x-coordinates and y-coordinates are preserved).

legacy_stratum_var
A character string containing the name of the column from legacy_sites that identifies stratum membership for each element of legacy_sites. This argument is required when the sampling design is stratified and its levels must be contained in the levels of the stratum_var variable. The default value of legacy_stratum_var is stratum_var, so legacy_stratum_var need only be specified explicitly when the name of the stratification variable in legacy_sites differs from stratum_var.

legacy_caty_var
A character string containing the name of the column from legacy_sites that identifies the unequal probability variable for each element of legacy_sites. This argument is required when the sampling design uses unequal selection probabilities and its categories must be contained in the levels of the caty_var variable. The default value of legacy_caty_var is caty_var, so legacy_caty_var need only be specified explicitly when the name of the unequal probability variable in legacy_sites differs from caty_var.

legacy_aux_var A character string containing the name of the column from legacy_sites that identifies the proportional probability variable for each element of legacy_sites. This argument is required when the sampling design uses proportional selection probabilities and the values of the legacy_aux_var variable must be positive. The default value of legacy_aux_var is aux_var, so legacy_aux_var need only be specified explicitly when the name of the proportional probability variable in legacy_sites differs from aux_var.

mindis A numeric value indicating the desired minimum distance between sampled sites. If the sampling design is stratified and mindis is an numeric value, the minimum distance is applied to all strata. If the sampling design is stratified and different minimum distances are desired among strata, then mindis is a list whose names match the names of n_base and whose and values are the minimum distance for the corresponding stratum. If a minimum distance is not desired for a particular stratum, then the corresponding value in mindis should be 0 or NULL (which is equivalent to 0). The units of mindis must represent the units in sframe. A warning is returned if the minimum distance could not

be reached after maxtry attempts. If legacy sites are used, the minimum distance requirement (and subsequent warning if maxtry attempts are reached) is enforced for all base sites that are not legacy sites (i.e., the minimum distance is enforced for these sites by comparing distances against all base sites (legacy and non-legacy)).

maxtry The number of maximum attempts to apply the minimum distance algorithm to obtain the desired minimum distance between sites. Each iteration takes roughly as long as the standard GRTS algorithm. Successive iterations will always contain at least as many sites satisfying the minimum distance requirement as the previous iteration. The algorithm stops when the minimum distance requirement is met or there are maxtry iterations. The default number of maximum iterations is 10.

n_over The number of reverse hierarchically ordered (rho) replacement sites. If the sampling design is unstratified, then n_over is an integer specifying the number of rho replacement sites desired. If the sampling design is stratified, then n_over is a vector (or list) whose names match the names of n_base and whose values indicate the number of rho replacement sites for each stratum. If replacement sites are not desired for a particular stratum, then the corresponding value in n_over should be 0 or NULL (which is equivalent to 0). If the sampling design is stratified but the number of n_over sites is the same in each stratum, n_over can be a vector which is used for each stratum. If n_over is an unnamed, length-one vector, it's value is recycled and used for each stratum. Note that if the sampling design has unequal selection probabilities (seltype = "unequal"), then n_over sites are given the same proportion of caty_n values as n_base.

n_near The number of nearest neighbor (nn) replacement sites. If the sampling design is unstratified, n_near is integer from 1 to 10 specifying the number of nn replacement sites to be selected for each base site. If the sampling design is stratified but the same number of nn replacement sites is desired for each stratum, n_near is integer from 1 to 10 specifying the number of nn replacement sites to be selected for each base site. If the sampling design is unstratified and a different number of nn replacement sites is desired for each stratum, n_near is a vector (or list) whose names represent strata and whose values is integer from 1 to 10 specifying the number of nn replacement sites to be selected for each base site in the stratum. If replacement sites are not desired for a particular stratum, then the corresponding value in n_over should be 0 or NULL (which is equivalent to 0). For infinite sampling frames, the distance between a site and its nn depends on pt_density. The larger pt_density, the closer the nn neighbors.

wgt_units The units used to compute the design weights. These units must be standard units as defined by the set_units() function in the units package. The default units match the units of the sf object.

pt_density A positive integer controlling the density of the GRTS approximation for infinite sampling frames. The GRTS approximation for infinite sample frames vastly improves computational efficiency by generating many finite points and selecting a sample from the points. pt_density represents the density of finite points per unit to use in the approximation. More specifically, for each stratum, the number of points used in the approximation equals pt_density * (n_base + n_over). A larger value of pt_density means a closer approximation to the

infinite sampling frame but less computational efficiency. The default value of `pt_density` is `10`. Note that when used with `caty_n`, the unequal inclusion probabilities generated from this approach are also approximations.

`DesignID` A character string indicating the naming structure for each site's identifier selected in the sample, which is matched with `SiteBegin` and included as a variable in the sf object in the function's output. Default is "Site".

`SiteBegin` A character string indicating the first number to use to match with `DesignID` while creating each site's identifier selected in the sample. Successive sites are given successive integers. The default starting number is `1` and the number of digits is equal to number of digits in `nbase + nover`. For example, if `nbase` is 50 and `nover` is 0, then the default site identifiers are `Site-01` to `Site-50`

`sep` A character string that acts as a separator between `DesignID` and `SiteBegin`. The default is `"-"`.

`projcrs_check` A check for whether the coordinates are projected. If `TRUE`, an error is returned if coordinates are not projected (i.e., they are geographic or NA). If `FALSE`, the check is not performed, which means that the crs in `sframe` (and `legacy_sites` if provided) can be projected, geographic, or NA.

Details

`n_base` is the number of sites used to calculate the design weights, which is typically the number of sites used in an analysis. When a panel sampling design is implemented, `n_base` is typically the number of sites in all panels that will be sampled in the same temporal period – `n_base` is not the total number of sites in all panels. The sum of `n_base` and `n_over` is equal to the total number of sites to be visited for all panels plus any replacement sites that may be required.

Value

The sampling design sites and additional information about the sampling design. More specifically, it is, a list with five elements:

- `sites_legacy` An sf object containing legacy sites. This is `NULL` if legacy sites were not included in the sample.
- `sites_base` An sf object containing the base sites. This is `NULL` if `n_base` equals the number of legacy sites.
- `sites_over` An sf object containing the reverse hierarchically ordered replacement sites. This is `NULL` if no reverse hierarchically ordered replacement sites were included in the sample.
- `sites_near` An sf object containing the nearest neighbor replacement sites. This is `NULL` if no nearest neighbor replacement sites were included in the sample.
- `design` A list documenting the specifications of this sampling design. This can be checked to verify your sampling design ran as intended.
  - `call` The original function call.
  - `stratum_var` The name of the stratification variable in `sframe`. This equals `NULL` if no stratification is used.
  - `stratum` The unique strata. This equals `"None"` if the sampling design is unstratified.
  - `n_base` The base sample size per stratum.

  - seltype The selection type per stratum.
  - caty_var The name of the unequal probability variable in sframe. This equals NULL if no unequal probability variable is used.
  - caty_n The expected sample sizes for each level of the unequal probability grouping variable per stratum. This equals NULL when seltype is not "unequal".
  - aux_var The name of the proportional probability (auxiliary) variable in sframe. This equals NULL if no proportional probability variable is used.
  - legacy A logical variable indicating whether legacy sites were included in the sample.
  - legacy_stratum_var The name of the stratification variable in legacy_sites. Omitted if legacy sites are not used. This equals NULL if legacy sites were used but no stratification variable is used.
  - legacy_caty_var The name of the unequal probability variable in legacy_sites. Omitted if legacy sites are not used. This equals NULL if legacy sites were used but no unequal probability variable is used.
  - legacy_aux_var The name of the proportional probability (auxiliary) variable in legacy_sites. Omitted if legacy sites are not used. This equals NULL if legacy sites were used but no proportional probability variable is used.
  - mindis The minimum distance requirement desired. This is NULL when no minimum distance requirement was applied.
  - n_over The reverse hierarchically ordered replacement site sample sizes per stratum. If seltype is unequal, this represents the expected sample sizes. This is NULL when no reverse hierarchically ordered replacement sites were selected.
  - n_near The number of nearest neighbor replacement sites desired. This is NULL when no nearest neighbor replacement sites were selected.

When non-NULL, the sites_legacy, sites_base, sites_over, and sites_near objects contain the original columns in sframe and include a few additional columns. These additional columns are

- siteID A site identifier (as named using the DesignID and SiteBegin arguments to grts()).
- siteuse Whether the site is a legacy site (Legacy), base site (Base), reverse hierarchically ordered replacement site (Over), or nearest neighbor replacement site (Near).
- replsite The replacement site ordering. replsite is None if the site is not a replacement site, Next if it is the next reverse hierarchically ordered replacement site to use, or Near_, where the word following _ indicates the ordering of sites closest to the originally sampled site.
- lon_WGS84 Longitude coordinates using the WGS84 coordinate system (EPSG:4326). Only given if coordinates are projected.
- lat_WGS84 Latitude coordinates using the WGS84 coordinate system (EPSG:4326). Only given if coordinates are projected.
- X Longitude coordinates using the provided coordinate system. Only given if coordinates are not projected (i.e., they are geographic or NA).
- Y Latitude coordinates using the provided coordinate system. Only given if coordinates are not projected (i.e., they are geographic or NA).
- stratum A stratum indicator. stratum is None if the sampling design was unstratified. If the sampling design was stratified, stratum indicates the stratum.

- wgt The design weight.
- ip The site's original inclusion probability (the reciprocal) of (wgt).
- caty An unequal probability grouping indicator. caty is None if the sampling design did not use unequal inclusion probabilities. If the sampling design did use unequal inclusion probabilities, caty indicates the unequal probability level.
- aux The auxiliary proportional probability variable. This column is only returned if seltype was proportional in the original sampling design.

If any columns in sframe contain these names, those columns from sframe will be automatically prefixed with sframe_ in the sites object. When output is printed, a summary of site counts by the levels in stratum_var and caty_var is shown.

Author(s)

Tony Olsen <olsen.tony@epa.gov>

See Also

grts to select a sample that is spatially balanced

Examples

```
## Not run:
samp <- irs(NE_Lakes, n_base = 100)
print(samp)
strata_n <- c(low = 25, high = 30)
samp_strat <- irs(NE_Lakes, n_base = strata_n, stratum_var = "ELEV_CAT")
print(samp_strat)
samp_over <- irs(NE_Lakes, n_base = 30, n_over = 5)
print(samp_over)

## End(Not run)
```

Lake_Ontario *Lake Ontario data*

Description

An sf MULTIPOLYGON object of 187 polygons consisting of shore segments in Lake Ontario.

Usage

```
Lake_Ontario
```

Format

187 rows and 5 variables:

`COUNTRY` Country.

`RSRC_CLASS` Bay class.

`PSTL_CODE` Postal code.

`AREA_SQKM` Area in square kilometers

`geometry` MULTIPOLYGON geometry using the NAD83 / Conus Albers coordinate reference system (EPSG: 5070).

`localmean_cov` *Internal Function: Variance-Covariance Matrix Based on Local Mean Estimator*

Description

This function calculates the variance-covariance matrix using the local mean estimator.

Usage

```
localmean_cov(zmat, weight_1st)
```

Arguments

`zmat`	Matrix of weighted response values or weighted residual values for the sample points.
`weight_1st`	List from the local mean weight function containing two elements: a matrix named `ij` composed of the index values of neighboring points and a vector named `gwt` composed of weights.

Value

The local mean estimator of the variance-covariance matrix.

Author(s)

Tom Kincaid <Kincaid.Tom@epa.gov>

localmean_var *Internal Function: Local Mean Variance Estimator*

Description

This function calculates the local mean variance estimator.

Usage

```
localmean_var(z, weight_1st)
```

Arguments

z	Vector of weighted response values or weighted residual values for the sample points.
weight_1st	List from the local mean weight function containing two elements: a matrix named ij composed of the index values of neighboring points and a vector named gwt composed of weights.

Value

The local mean estimator of the variance.

Author(s)

Tom Kincaid <Kincaid.Tom@epa.gov>

localmean_weight *Internal Function: Local Mean Variance Neighbors and Weights*

Description

This function calculates the index values of neighboring points and associated weights required by the local mean variance estimator.

Usage

```
localmean_weight(x, y, prb, nbh = 4)
```

Arguments

x	Vector of x-coordinates for location of the sample points.
y	Vector of y-coordinates for location of the sample points.
prb	Vector of inclusion probabilities for the sample points.
nbh	Number of neighboring points to use in the calculations.

Value

If ginv fails to return valid output, a NULL object. Otherwise, a list containing two elements: a matrix named `ij` composed of the index values of neighboring points and a vector named `gwt` composed of weights.

Author(s)

Tom Kincaid <Kincaid.Tom@epa.gov>

NE_Lakes *New England Lakes data*

Description

An `sf` POINT object of 195 lakes in the Northeastern United States.

Usage

```
NE_Lakes
```

Format

195 rows and 5 variables:

`AREA` Lake area in hectares.

`AREA_CAT` Lake area categories based on a hectare cutoff.

`ELEV` Elevation in meters.

`ELEV_CAT` Elevation categories based on a meter cutoff.

`geometry` POINT geometry using the NAD83 / Conus Albers coordinate reference system (EPSG: 5070).

NE_Lakes_df *New England Lakes data (as a data frame)*

Description

An data frame of 195 lakes in the Northeastern United States.

Usage

```
NE_Lakes_df
```

Format

195 rows and 6 variables:

`AREA` Lake area in hectares.

`AREA_CAT` Lake area categories based on a hectare cutoff.

`ELEV` Elevation in meters.

`ELEV_CAT` Elevation categories based on a meter cutoff.

`XCOORD` x-coordinate using the WGS 84 coordinate reference system (EPSG: 4326)

`YCOORD` y-coordinate using WGS 84 coordinate reference system (EPSG: 4326)

NE_Lakes_Legacy *New England Lakes legacy data*

Description

An `sf` POINT object of 5 legacy sites for the NE Lakes data

Usage

```
NE_Lakes_Legacy
```

Format

5 rows and 5 variables:

`AREA` Lake area in hectares.

`AREA_CAT` Lake area categories based on a hectare cutoff.

`ELEV` Elevation in meters.

`ELEV_CAT` Elevation categories based on a meter cutoff.

`geometry` POINT geometry using the NAD83 / Conus Albers coordinate reference system (EPSG: 5070).

NLA_PNW *NLA PNW data*

Description

An sf POINT object of 96 lakes in the Pacific Northwest Region of the United States during the year 2017, from a subset of the Environmental Protection Agency's "National Lakes Assessment."

Usage

```
NLA_PNW
```

Format

96 rows and 9 variables:

SITE_ID A unique lake identifier.

WEIGHT The sampling design weight.

URBAN Urban category.

STATE State name.

BMMI Benthic MMI value.

BMMI_COND Benthic MMI condition categories.

PHOS_COND Phosphorus condition categories.

NITR_COND Nitrogen condition categories.

geometry POINT geometry using the NAD83 / Conus Albers coordinate reference system (EPSG: 5070).

NRSA_EPA7 *NRSA EPA7 data*

Description

An sf POINT object of 353 stream segments in the Central United States during the years 2008 and 2013, from a subset of the Environmental Protection Agency's "National Rivers and Streams Assessment."

Usage

```
NRSA_EPA7
```

Format

353 rows and 10 variables:

`SITE_ID` A unique site identifier.

`YEAR` Year of design cycle.

`WEIGHT` Sampling design weights.

`ECOREGION` Ecoregion.

`STATE` State name.

`BMMI` Benthic MMI value.

`BMMI_COND` Benthic MMI categories.

`PHOS_COND` Phosphorus condition categories.

`NITR_COND` Nitrogen condition categories.

`geometry` POINT geometry using the NAD83 / Conus Albers coordinate reference system (EPSG: 5070).

`pd_summary` *Summary characteristics of a panel revisit design*

Description

Panel revisit design characteristics are summarized: number of panels, number of time periods, total number of sample events for the revisit design, total number of sample events for each panel, total number of sample events for each time period and cumulative number of unique units sampled by time periods.

Usage

```
pd_summary(object, visitdsgn = NULL, ...)
```

Arguments

`object`	Two-dimensional array from `panel_design` and dimnames specifying revisit panel design. Typically, array is output from `revisit_dsgn`, `revisit_bibd` or `revisit_rand` functions.
`visitdsgn`	Two-dimensional array with same dimensions as `paneldsgn` specifying the number of times a sample unit is sampled at each time period. Default is `visitdsgn=NULL`, where default assumes that a sample unit will be sampled only once at each time period.
`...`	Additional arguments (S3 consistency)

Details

The revisit panel design and the visit design (if present) are summarized. Summaries can be useful to know the effort required to complete the survey design. See the values returned for the summaries that are produced.

Value

List of six elements.

`n_panel` number of panels in revisit design

`n_period` number of time periods in revisit design

`n_total` total number of sample events across all panels and all time periods, accounting for `visitdsgn`, that will be sampled in the revisit design

`n_periodunit` vector of the number of time periods a unit will be sampled in each panel

`n_unitpnl` vector of the number of sample units, accounting for `visitdsgn`, that will be sampled in each panel

`n_unitperiod` vector of the number of sample units, accounting for `visitdsgn`, that will be sampled during each time period

`ncum_unit` vector of the cumulative number of unique units that will be sampled in time periods up to and including the current time period.

Author(s)

Tony Olsen <Olsen.Tony@epa.gov>

Examples

```
# Serially alternating panel revisit design summary
sa_dsgn <- revisit_dsgn(20, panels = list(SA60N = list(
  n = 60, pnl_dsgn = c(1, 4),
  pnl_n = NA, start_option = "None"
)), begin = 1)
pd_summary(sa_dsgn)
# Add visit design where first panel is sampled twice at every time period
sa_visit <- sa_dsgn
sa_visit[sa_visit > 0] <- 1
sa_visit[1, sa_visit[1, ] > 0] <- 2
pd_summary(sa_dsgn, sa_visit)
```

---

plot *Plot sampling frames, design sites, and analysis data.*

Description

This function plots sampling frames, design sites, and analysis data. If the left-hand side of the formula is empty, plots are of the distributions of the right-hand side variables. If the left-hand side of the variable contains a variable, plots are of the left-hand size variable for each level of each right-hand side variable. This function is largely built on `plot.sf()`, and all spsurvey plotting methods can supply additional arguments to `plot.sf()`. For more information on plotting in `sf`, run `?sf::plot.sf()`. Equivalent to `sp_plot()`; both are currently maintained for backwards compatibility.

Usage

```
## S3 method for class 'sp_frame'
plot(
  x,
  formula = ~1,
  xcoord,
  ycoord,
  crs,
  var_args = NULL,
  varlevel_args = NULL,
  geom = FALSE,
  onlyshow = NULL,
  fix_bbox = TRUE,
  ...
)

## S3 method for class 'sp_design'
plot(
  x,
  sframe = NULL,
  formula = ~siteuse,
  siteuse = NULL,
  var_args = NULL,
  varlevel_args = NULL,
  geom = FALSE,
  onlyshow = NULL,
  fix_bbox = TRUE,
  ...
)
```

Arguments

x	An object to plot. When plotting sampling frames an `sf` object given the appropriate class using `sp_frame`. When plotting design sites, an object created by `grts()` or `irs()` (which has class `sp_design`). When plotting analysis data, a data frame or an `sf` object given the appropriate class using `sp_frame`.
formula	A formula. One-sided formulas are used to summarize the distribution of numeric or categorical variables. For one-sided formulas, variable names are placed to the right of ~ (a right-hand side variable). Two sided formulas are used to summarize the distribution of a left-hand side variable for each level of each right-hand side categorical variable in the formula. Note that only for two-sided formulas are numeric right-hand side variables coerced to a categorical variables. If an intercept is included as a right-hand side variable (whether the formula is one-sided or two-sided), the total will also be summarized. When plotting sampling frames or analysis data, the default formula is `~ 1`. When plotting design sites, `siteuse` should be used in the formula, and the default formula is `~ siteuse`.

xcoord	Name of the x-coordinate (east-west) in object (only required if object is not an sf object).
ycoord	Name of y (north-south)-coordinate in object (only required if object is not an sf object).
crs	Projection code for xcoord and ycoord (only required if object is not an sf object).
var_args	A named list. The name of each list element corresponds to a right-hand side variable in formula. Values in the list are composed of graphical arguments that are to be passed to **every** level of the variable. To see all graphical arguments available, run ?plot.sf.
varlevel_args	A named list. The name of each list element corresponds to a right-hand side variable in formula. The first element in this list should be "levels" and contain all levels of the particular right-hand side variable. Subsequent names correspond to graphical arguments that are to be passed to the specified levels (in order) of the right-hand side variable. Values for each graphical argument must be specified for each level of the right-hand side variable, but applicable sf defaults will be matched by inputting the value NA. To see all graphical arguments available, run ?plot.sf
geom	Should separate geometries for each level of the right-hand side formula variables be plotted? Defaults to FALSE.
onlyshow	A string indicating the single level of the single right-hand side variable for which a summary is requested. This argument is only used when a single right-hand side variable is provided.
fix_bbox	Should the geometry bounding box be fixed across plots? If a length-four vector with names "xmin", "ymin", "xmax", and "ymax" and values indicating bounding box edges, the bounding box will be fixed as fix_bbox across plots. If TRUE, the bounding box will be fixed across plots as the bounding box of object. If FALSE, the bounding box will vary across plots according to the unique geometry for each plot. Defaults to TRUE.
...	Additional arguments to pass to plot.sf().
sframe	The sampling frame (an sf object) to plot alongside design sites. This argument is only used when object corresponds to the design sites.
siteuse	A character vector of site types to include when plotting design sites. It can only take on values "sframe" (sampling frame), "Legacy" (for legacy sites), "Base" (for base sites), "Over" (for n_over replacement sites), and "Near" (for n_near replacement sites). The order of sites represents the layering in the plot (e.g. siteuse = c("Base", "Legacy") will plot legacy sites on top of base sites. Defaults to all non-NULL elements in x and y with plot order "sframe", "Legacy", "Base", "Over", "Near".

Author(s)

Michael Dumelle <Dumelle.Michael@epa.gov>

Examples

```
## Not run:
data("NE_Lakes")
NE_Lakes <- sp_frame(NE_Lakes)
plot(NE_Lakes, formula = ~ELEV_CAT)
sample <- grts(NE_Lakes, 30)
plot(sample, NE_Lakes)

## End(Not run)
```

---

plot.sp_CDF *Plot a cumulative distribution function (CDF)*

Description

This function creates a CDF plot. Input data for the plots is provided by a data frame from the "CDF" output given by `cont_analysis`. Confidence limits for the CDF also are plotted. Equivalent to `cdf_plot()`; both are currently maintained for backwards compatibility.

Usage

```
## S3 method for class 'sp_CDF'
plot(
  x,
  var = NULL,
  subpop = NULL,
  subpop_level = NULL,
  units_cdf = "Percent",
  type_cdf = "Continuous",
  log = "",
  xlab = NULL,
  ylab = NULL,
  ylab_r = NULL,
  main = NULL,
  legloc = NULL,
  confcut = 0,
  conflev = 95,
  cex.main = 1.2,
  cex.legend = 1,
  ...
)
```

Arguments

`x`	Data frame from the "CDF" output given by `cont_analysis`.
`var`	If `cdfest` has multiple variables in the "Indicator" column, then `var` is the single variable to be plotted. The default is `NULL`, which assumes that only one variable is in the "Indicator" column of `cdfest`.

subpop	If cdfest has multiple variables in the "Type" column, then subpop is the single variable to be plotted. The default is NULL, which assumes that only one variable is in the "Type" column of cdfest.
subpop_level	If cdfest has multiple levels of subpop in the "Subpopulation" column, then subpop_level is the single level to be plotted. The default is NULL, which assumes that only one level is in the "Subpopulation" column of cdfest.
units_cdf	Indicator for the label utilized for the left side y-axis and the values used for the left side y-axis tick marks, where "Percent" means the label and values are in terms of percent of the population, and "Units" means the label and values are in terms of units (count, length, or area) of the population. The default is "Percent".
type_cdf	Character string consisting of the value "Continuous" or "Ordinal" that controls the type of CDF plot. The default is "Continuous".
log	Character string consisting of the value "" or "x" that controls whether the x axis uses the original scale ("") or the base 10 logarithmic scale ("x"). The default is "".
xlab	Character string providing the x-axis label. If this argument equals NULL, then the indicator name is used as the label. The default is NULL.
ylab	Character string providing the left side y-axis label. If argument units_cdf equals "Units", a value should be provided for this argument. Otherwise, the label will be "Percent". The default is "Percent".
ylab_r	Character string providing the label for the right side y-axis (and, hence, determining the values used for the right side y-axis tick marks), where NULL means a right side y-axis is not created. If this argument equals "Same", the right side y-axis will have the same label and tick mark values as the left side y-axis. If this argument equals a character string other than "Same", the right side y-axis label will be the value provided for argument ylab_r, and the right side y-axis tick mark values will be determined by the choice not utilized for argument units_cdf, which means that the default value of argument units_cdf (i.e., "Percent") will result in the right side y-axis tick mark values being expressed in terms of units of the population (i.e., count, length, or area). The default is NULL.
main	Character string providing the plot title. The default is NULL.
legloc	Indicator for location of the plot legend, where "BR" means bottom right, "BL" means bottom left, "TR" means top right, "TL" means top left, and NULL means no legend. The default is NULL.
confcut	Numeric value that controls plotting confidence limits at the CDF extremes. Confidence limits for CDF values (percent scale) less than confcut or greater than 100 minus confcut are not plotted. A value of zero means confidence limits are plotted for the complete range of the CDF. The default is 0.
conflev	Numeric value of the confidence level used for confidence limits. The default is 95.
cex.main	Expansion factor for the plot title. The default is 1.2.
cex.legend	Expansion factor for the legend title. The default is 1.
...	Additional arguments passed to the plot.default function (aside from those already used and ylim).

Value

A plot of a variable's CDF estimates associated confidence limits.

Author(s)

Tom Kincaid <Kincaid.Tom@epa.gov>

See Also

cont_cdfplot for creating a PDF file containing CDF plots

cont_cdftest for CDF hypothesis testing

Examples

```
## Not run:
dframe <- data.frame(
  siteID = paste0("Site", 1:100),
  wgt = runif(100, 10, 100),
  xcoord = runif(100),
  ycoord = runif(100),
  stratum = rep(c("Stratum1", "Stratum2"), 50),
  ContVar = rnorm(100, 10, 1),
  All_Sites = rep("All Sites", 100),
  Resource_Class = rep(c("Good", "Poor"), c(55, 45))
)
myvars <- c("ContVar")
mysubpops <- c("All_Sites", "Resource_Class")
mypopsize <- data.frame(
  Resource_Class = c("Good", "Poor"),
  Total = c(4000, 1500)
)
myanalysis <- cont_analysis(dframe,
  vars = myvars, subpops = mysubpops,
  siteID = "siteID", weight = "wgt", xcoord = "xcoord", ycoord = "ycoord",
  stratumID = "stratum", popsize = mypopsize
)
keep <- with(myanalysis$CDF, Type == "Resource_Class" &
  Subpopulation == "Good")
par(mfrow = c(2, 1))
plot(myanalysis$CDF[keep, ],
  xlab = "ContVar",
  ylab = "Percent of Stream Length", ylab_r = "Stream Length (km)",
  main = "Estimates for Resource Class: Good"
)
plot(myanalysis$CDF[keep, ],
  xlab = "ContVar",
  ylab = "Percent of Stream Length", ylab_r = "Same",
  main = "Estimates for Resource Class: Good"
)

## End(Not run)
```

power_dsgn *Power calculation for multiple panel designs*

Description

Calculates the power for trend detection for one or more variables, for one or more panel designs, for one or more linear trends, and for one or more significance levels. The panel designs create a covariance model where the model includes variance components for units, periods, the interaction of units and periods, and the residual (or index) variance.

Usage

```
power_dsgn(
  ind_names,
  ind_values,
  unit_var,
  period_var,
  unitperiod_var,
  index_var,
  unit_rho = 1,
  period_rho = 0,
  paneldsgn,
  nrepeats = NULL,
  trend_type = "mean",
  ind_pct = NULL,
  ind_tail = NULL,
  trend = 2,
  alpha = 0.05
)
```

Arguments

ind_names	Vector of indicator names
ind_values	Vector of indicator mean values
unit_var	Vector of variance component estimates for unit variability for the indicators
period_var	Vector of variance component estimates for period variability for the indicators
unitperiod_var	Vector of variance component estimates for unit by period interaction variability for the indicators
index_var	Vector of variance component estimates for index (residual) error for the indicators
unit_rho	Correlation across units. Default is 1.
period_rho	Correlation across periods. Default is 0.

paneldsgn	A list of panel designs each as a matrix. Each element of the list is a matrix with dimnames (dimensions: number of panels (rows) by number of periods (columns)) containing the number of units visited for each combination of panel and period. Dimnames for columns must be able to be coerced into an integer (e.g., 2016). All designs must span the same number of periods. Typically, the panel designs are the output of the function revisit_dsgn.
nrepeats	Either NULL or a list of matrices the same length as paneldsgn specifying the number of revisits made to units in a panel in the same period for each design. Specifying NULL indicates that number of revisits to units is the same for all panels and for all periods and for all panel designs. The default is NULL, a single visit. Names must match list names in paneldsgn.
trend_type	Trend type is either "mean" where trend is applied as percent trend in the indicator mean or "percent" where the trend is applied as percent trend in the proportion (percent) of the distribution that is below or above a fixed value. Default is trend_type="mean"
ind_pct	When trend_type is equal to "percent", a vector of the values of the indicator fixed value that defines the percent. Default is NULL
ind_tail	When trend_type is equal to "percent", a character vector with values of either "lower" or "upper" for each indicator. "lower" states that the percent is associated with the lower tail of the distribution and "upper" states that the percent is associated with the upper tail of the distribution. Default is NULL.
trend	Single value or vector of assumed percent change from initial value in the indicator for each period. Assumes the trend is expressed as percent per period. Note that the trend may be either positive or negative. The default is 2.
alpha	Single value or vector of significance level for linear trend test, alpha, Type I error, level. The default is 0.05.

Details

Calculates the power for detecting a change in the mean for different panel design structures. The model incorporates unit, period, unit by period, and index variance components as well as correlation across units and across periods. See references for methods.

Value

A list with components trend_type, ind_pct, ind_tail, trend values across periods, periods (all periods included in one or more panel designs), significance levels, a five-dimensional array of power calculations (dimensions: panel, design names, periods, indicator names, trend names, alpha_names), an array of indicator mean values for each trend and the function call.

Author(s)

Tony Olsen <Olsen.Tony@epa.gov>

References

Urquhart, N. S., W. S. Overton, et al. (1993) Comparing sampling designs for monitoring ecological status and trends: impact of temporal patterns. In: *Statistics for the Environment.* V. Barnett and K. F. Turkman. John Wiley & Sons, New York, pp. 71-86.

Urquhart, N. S. and T. M. Kincaid (1999). Designs for detecting trends from repeated surveys of ecological resources. *Journal of Agricultural, Biological, and Environmental Statistics*, **4(4)**, 404-414.

Urquhart, N. S. (2012). The role of monitoring design in detecting trend in long-term ecological monitoring studies. In: *Design and Analysis of Long-term Ecological Monitoring Studies.* R. A. Gitzen, J. J. Millspaugh, A. B. Cooper, and D. S. Licht (eds.). Cambridge University Press, New York, pp. 151-173.

See Also

ppd_plot to plot power curves for panel designs

Examples

```
# Power for rotating panel with sample size 60
power_dsgn("Variable_Name",
  ind_values = 43, unit_var = 280, period_var = 4,
  unitperiod_var = 40, index_var = 90, unit_rho = 1, period_rho = 0,
  paneldsgn = list(NoR60 = revisit_dsgn(20,
    panels = list(NoR60 = list(
      n = 60, pnl_dsgn = c(1, NA),
      pnl_n = NA, start_option = "None"
    )), begin = 1
  )),
  nrepeats = NULL, trend_type = "mean", trend = 1.0, alpha = 0.05
)
```

---

ppd_plot *Plot power curves for panel designs*

Description

Plot power curves and relative power curves for trend detection for set of panel designs, time periods, indicators, significance levels and trend. Trend may be based on percent change per period in mean or percent change in proportion of cumulative distribution function above or below a fixed cut point. Types of plots are combinations of standard/relative, mean/percent, period/change and design/indicator. Input must be be of class powerpaneldesign and is normally the output of function power_dsgn.

Usage

```
ppd_plot(
  object,
  plot_type = "standard",
  trend_type = "mean",
  xaxis_type = "period",
  comp_type = "design",
  dsgns = NULL,
  indicator = NULL,
  trend = NULL,
  period = NULL,
  alpha = NULL,
  ...
)
```

Arguments

object	List object of class powerpaneldesign. Object provides power calculated for a set of panel designs, set of indicators, set of trend values, and set of alpha values. Expect input as list as output from function power_dsgn.
plot_type	Default is "standard" which plots standard power curve. If equal to "relative", then plot power of one panel design compared to one or more other panel designs.
trend_type	Character value for trend in mean ("mean") or or percent change in proportion ("percent") of cumulative distribution function above or below a fixed cut point. Default is "mean".
xaxis_type	Character value equal to "period" or "change" which designates the type of x-axis for power plot where power is plotted on y-axis. For xaxis_type = "period", x-axis is periods in dsgnpower. If xaxis_type = "change", then x-axis is percent per period with secondary x-axes for total percent per period and associated change in mean. Default is "period". Note that xaxis_type controls how the input for "period" and "trend" parameters is used.
comp_type	Character value equal to "design" or "indicator" which designates the type of power curve comparison that will occur on a single plot. If comp_type = "design", then on a single plot of power curves all panel designs specified in "dsgns" are plotted for a single indicator, single trend value and single alpha. If comp_type = "indicator", then on a single plot of power curves all indicators specified in "indicator" are plotted for a single panel design, single trend value and single alpha. Default is "design".
dsgns	Vector of names of panel designs that are to be plotted. Names must be all, or a subset of, names of designs in dsgnpower. Default is NULL which results in only the first panel design in dsgnpower being used.
indicator	Vector of indicator names contained in dsgnpower that are to be plotted. Indicator names must be all, or a subset of, indicator names in dsgnpower. Default is NULL which results in only the first indicator in dsgnpower being used.

trend	NULL. A single value or vector of values contained in dsgnpower that will be plotted. Values must be all, or a subset of, trend values in dsgnpower. If xaxis_type is equal to "period", then NULL results in maximum trend value being used and a single value or vector of values results in a separate plot for each value specified. If xaxis_type is equal to "change", then NULL results in all trend values in dsgnpower being plotted on x-axis and a vector of values results in all trend values in dsgnpower from minimum value to maximum value specified being plotted on x-axis.
period	NULL, a single value or vector of values contained in dsgnpower that will be plotted. Values must be all, or a subset of, period values in dsgnpower. If xaxis_type is equal to "period", then NULL results in all time periods in dsgnpower being plotted on x-axis and a vector of values results in all period values in dsgnpower from minimum value to maximum value specified being plotted on x-axis. If xaxis_type is equal to "change", then NULL results in all time periods in dsgnpower being plotted in separate plots and a vector of values results in time periods specified being plotted in separate plots.
alpha	A single value or vector of significance levels (as proportion, e.g. 0.05) contained in dsgnpower to used for power plots. Specifying more than a single value results in multiple plots. Default is NULL which results in the minimum significance level in dsgnpower being used.
...	Additional arguments (S3 consistency)

Details

By default the plot function produces a standard power curve at end of each time period on the x-axis with y-axis as power. When more than one panel design is in dsgnpower, the first panel design is used. When more than one indicator is in dsgnpower, the first indicator is used. When more than one trend value is in dsgnpower, the maximum trend value is used. When more than one significance level, alpha, is in dsgnpower, the minimum significance level is used.

Control of the type of plot produced is governed by plot_type, trend_type, xaxis_type and comp_type. The number of plots produced is governed by the number of panel designs (dsgn) specified, the number of indicators (indicator) specified, the number of time periods (period) specifies, the number of trend values (trend) specified and the number of significance levels (alpha) specified.

When the comparison type ("comp_type") is equal to "design", all power curves specified by dsgn are plotted on the same plot. When comp_type is equal to "indicator", all power curves specified by "indicator" are plotted on the same plot. Typically, no more than 4-5 power curves should be plotted on same plot.

Value

One or more power curve plots are created and plotted. User must specify output graphical device if more than one plot is created. See Devices for graphical output options.

Author(s)

Tony Olsen <Olsen.Tony@epa.gov>

Examples

```
## Not run:
# Construct a rotating panel design with sample size of 60
R60N <- revisit_dsgn(20, panels = list(R60N = list(
  n = 60, pnl_dsgn = c(1, NA),
  pnl_n = NA, start_option = "None"
)), begin = 1)

# Construct a fixed panel design with sample size of 60
F60 <- revisit_dsgn(20, panels = list(F60 = list(
  n = 60, pnl_dsgn = c(1, 0),
  pnl_n = NA, start_option = "None"
)), begin = 1)

# Power for rotating panel with sample size 60
Power_tst <- power_dsgn("Variable_Name",
  ind_values = 43, unit_var = 280,
  period_var = 4, unitperiod_var = 40, index_var = 90,
  unit_rho = 1, period_rho = 0, paneldsgn = list(
    R60N = R60N, F60 = F60
  ), nrepeats = NULL,
  trend_type = "mean", trend = c(1.0, 2.0), alpha = 0.05
)
ppd_plot(Power_tst)
ppd_plot(Power_tst, dsgns = c("F60", "R60N"))
ppd_plot(Power_tst, dsgns = c("F60", "R60N"), trend = 1.0)
ppd_plot(Power_tst,
  plot_type = "relative", comp_type = "design",
  trend_type = "mean", trend = c(1, 2), dsgns = c("R60N", "F60"),
  indicator = "Variable_Name"
)

## End(Not run)
```

`relrisk_analysis` *Relative risk analysis*

Description

This function organizes input and output for relative risk analysis (of categorical variables). The analysis data, `dframe`, can be either a data frame or a simple features (`sf`) object. If an `sf` object is used, coordinates are extracted from the geometry column in the object, arguments `xcoord` and `ycoord` are assigned values `"xcoord"` and `"ycoord"`, respectively, and the geometry column is dropped from the object.

Usage

```
relrisk_analysis(
  dframe,
```

```
  vars_response,
  vars_stressor,
  response_levels = NULL,
  stressor_levels = NULL,
  subpops = NULL,
  siteID = NULL,
  weight = "weight",
  xcoord = NULL,
  ycoord = NULL,
  stratumID = NULL,
  clusterID = NULL,
  weight1 = NULL,
  xcoord1 = NULL,
  ycoord1 = NULL,
  sizeweight = FALSE,
  sweight = NULL,
  sweight1 = NULL,
  fpc = NULL,
  popsize = NULL,
  vartype = "local",
  conf = 95,
  All_Sites = FALSE,
  subset_local = TRUE
)
```

Arguments

dframe	Data to be analyzed (analysis data). A data frame or sf object containing survey design variables, response variables, stressor variables, and subpopulation (domain) variables.
vars_response	Vector composed of character values that identify the names of response variables in dframe. Each response variable must have two category values (levels), where one level is associated with poor condition and the other level is associated with good condition.
vars_stressor	Vector composed of character values that identify the names of stressor variables in dframe. Each stressor variable must have two category values (levels), where one level is associated with poor condition and the other level is associated with good condition.
response_levels	List providing the category values (levels) for each element in the vars_response argument. Each element in the list must contain two values, where the first value identifies poor condition, and the second value identifies good condition. This argument must be named and must be the same length as argument vars_response. Names for this argument must match the values in the vars_response argument. If this argument equals NULL, then a named list is created that contains the values "Poor" and "Good" for the first and second levels, respectively, of each element in the vars_response argument and that uses values in the vars_response argument as names for the list. If response_levels

is provided without names, then the names of response_levels are set to vars_response. The default value is NULL.

stressor_levels

List providing the category values (levels) for each element in the vars_stressor argument. Each element in the list must contain two values, where the first value identifies poor condition, and the second value identifies good condition. This argument must be named and must be the same length as argument vars_stressor. Names for this argument must match the values in the vars_stressor argument. If this argument equals NULL, then a named list is created that contains the values "Poor" and "Good" for the first and second levels, respectively, of each element in the vars_stressor argument and that uses values in the vars_stressor argument as names for the list. If stressor_levels is provided without names, then the names of stressor_levels are set to vars_stressor. The default value is NULL.

subpops — Vector composed of character values that identify the names of subpopulation (domain) variables in dframe. If a value is not provided, the value "All_Sites" is assigned to the subpops argument and a factor variable named "All_Sites" that takes the value "All Sites" is added to dframe. The default value is NULL.

siteID — Character value providing the name of the site ID variable in dframe. For a two-stage sample, the site ID variable identifies stage two site IDs. The default value is NULL, which assumes that each row in dframe represents a unique site.

weight — Character value providing the name of the design weight variable in dframe. For a two-stage sample, the weight variable identifies stage two weights. The default value is "weight".

xcoord — Character value providing name of the x-coordinate variable in dframe. For a two-stage sample, the x-coordinate variable identifies stage two x-coordinates. Note that x-coordinates are required for calculation of the local mean variance estimator. If dframe is an sf object, this argument is not required (as the geometry column in dframe is used to find the x-coordinate). The default value is NULL.

ycoord — Character value providing name of the y-coordinate variable in dframe. For a two-stage sample, the y-coordinate variable identifies stage two y-coordinates. Note that y-coordinates are required for calculation of the local mean variance estimator. If dframe is an sf object, this argument is not required (as the geometry column in dframe is used to find the t-coordinate). The default value is NULL.

stratumID — Character value providing the name of the stratum ID variable in dframe. The default value is NULL.

clusterID — Character value providing the name of the cluster (stage one) ID variable in dframe. Note that cluster IDs are required for a two-stage sample. The default value is NULL.

weight1 — Character value providing the name of the stage one weight variable in dframe. The default value is NULL.

xcoord1 — Character value providing the name of the stage one x-coordinate variable in dframe. Note that x coordinates are required for calculation of the local mean variance estimator. The default value is NULL.

ycoord1 — Character value providing the name of the stage one y-coordinate variable in dframe. Note that y-coordinates are required for calculation of the local mean variance estimator. The default value is NULL.

sizeweight — Logical value that indicates whether size weights should be used during estimation, where TRUE uses size weights and FALSE does not use size weights. To employ size weights for a single-stage sample, a value must be supplied for argument weight. To employ size weights for a two-stage sample, values must be supplied for arguments weight and weight1. The default value is FALSE.

sweight — Character value providing the name of the size weight variable in dframe. For a two-stage sample, the size weight variable identifies stage two size weights. The default value is NULL.

sweight1 — Character value providing the name of the stage one size weight variable in dframe. The default value is NULL.

fpc — Object that specifies values required for calculation of the finite population correction factor used during variance estimation. The object must match the survey design in terms of stratification and whether the design is single-stage or two-stage. For an unstratified design, the object is a vector. The vector is composed of a single numeric value for a single-stage design. For a two-stage unstratified design, the object is a named vector containing one more than the number of clusters in the sample, where the first item in the vector specifies the number of clusters in the population and each subsequent item specifies the number of stage two units for the cluster. The name for the first item in the vector is arbitrary. Subsequent names in the vector identify clusters and must match the cluster IDs. For a stratified design, the object is a named list of vectors, where names must match the strata IDs. For each stratum, the format of the vector is identical to the format described for unstratified single-stage and two-stage designs. Note that the finite population correction factor is not used with the local mean variance estimator.

Example fpc for a single-stage unstratified survey design:

```
fpc <- 15000
```

Example fpc for a single-stage stratified survey design:

```
fpc <- list(
    Stratum_1 = 9000,
    Stratum_2 = 6000)
```

Example fpc for a two-stage unstratified survey design:

```
fpc <- c(
    Ncluster = 150,
    Cluster_1 = 150,
    Cluster_2 = 75,
    Cluster_3 = 75,
    Cluster_4 = 125,
    Cluster_5 = 75)
```

Example fpc for a two-stage stratified survey design:

```
fpc <- list(
    Stratum_1 = c(
```

```
    Ncluster_1 = 100,
    Cluster_1 = 125,
    Cluster_2 = 100,
    Cluster_3 = 100,
    Cluster_4 = 125,
    Cluster_5 = 50),
  Stratum_2 = c(
    Ncluster_2 = 50,
    Cluster_1 = 75,
    Cluster_2 = 150,
    Cluster_3 = 75,
    Cluster_4 = 75,
    Cluster_5 = 125))
```

popsize

Object that provides values for the population argument of the calibrate or postStratify functions in the survey package. If a value is provided for popsize, then either the calibrate or postStratify function is used to modify the survey design object that is required by functions in the survey package. Whether to use the calibrate or postStratify function is dictated by the format of popsize, which is discussed below. Post-stratification adjusts the sampling and replicate weights so that the joint distribution of a set of post-stratifying variables matches the known population joint distribution. Calibration, generalized raking, or GREG estimators generalize post-stratification and raking by calibrating a sample to the marginal totals of variables in a linear regression model. For the calibrate function, the object is a named list, where the names identify factor variables in dframe. Each element of the list is a named vector containing the population total for each level of the associated factor variable. For the postStratify function, the object is either a data frame, table, or xtabs object that provides the population total for all combinations of selected factor variables in the dframe data frame. If a data frame is used for popsize, the variable containing population totals must be the last variable in the data frame. If a table is used for popsize, the table must have named dimnames where the names identify factor variables in the dframe data frame. If the popsize argument is equal to NULL, then neither calibration nor post-stratification is performed. The default value is NULL.

Example popsize for calibration:

```
popsize <- list(
  Ecoregion = c(
    East = 750,
    Central = 500,
    West = 250),
  Type = c(
    Streams = 1150,
    Rivers = 350))
```

Example popsize for post-stratification using a data frame:

```
popsize <- data.frame(
  Ecoregion = rep(c("East", "Central", "West"),
```

```
  rep(2, 3)),
Type = rep(c("Streams", "Rivers"), 3),
Total = c(575, 175, 400, 100, 175, 75))
```

Example popsize for post-stratification using a table:

```
popsize <- with(MySurveyFrame,
    table(Ecoregion, Type))
```

Example popsize for post-stratification using an xtabs object:

```
popsize <- xtabs(~Ecoregion + Type,
    data = MySurveyFrame)
```

`vartype` Character value providing the choice of the variance estimator, where `"Local"` indicates the local mean estimator and `"SRS"` indicates the simple random sampling estimator. The default value is `"local"`.

`conf` Numeric value providing the Gaussian-based confidence level. The default value is `95`.

`All_Sites` A logical variable used when `subpops` is not `NULL`. If `All_Sites` is `TRUE`, then alongside the subpopulation output, output for all sites (ignoring subpopulations) is returned for each variable in `vars`. If `All_Sites` is `FALSE`, then alongside the subpopulation output, output for all sites (ignoring subpopulations) is not returned for each variable in `vars`. The default is `FALSE`.

`subset_local` When 'subset_local = TRUE' (the default), subpopulations are subset to include only subpopulation members prior to variance estimation. When 'subset_local = FALSE', subpopulations are not subset prior to variance estimation. 'subset_local = FALSE' follows standard design-based subpopulation (i.e., domain) estimation theory but is computationally more complex, requiring the neighborhood weights matrix of the full data (which scales cubically with the sample size). 'subset_local' only applies when 'vartype = "local"'.

Value

The analysis results. A data frame of population estimates for all combinations of subpopulations, categories within each subpopulation, response variables, and categories within each response variable. Estimates are provided for proportion and size of the population plus standard error, margin of error, and confidence interval estimates. The data frame contains the following variables:

Type subpopulation (domain) name

Subpopulation subpopulation name within a domain

Response response variable

Stressor stressor variable

nResp sample size

Estimate relative risk estimate

Estimate_num relative risk numerator estimate

Estimate_denom relative risk denominator estimate

StdError relative risk standard error

MarginofError relative risk margin of error

LCBxxPct xx% (default 95%) lower confidence bound

UCBxxPct xx% (default 95%) upper confidence bound

WeightTotal sum of design weights

Count_RespPoor_StressPoor number of observations in the poor response and poor stressor group

Count_RespPoor_StressGood number of observations in the poor response and good stressor group

Count_RespGood_StressPoor number of observations in the good response and poor stressor group

Count_RespGood_StressGood number of observations in the good response and good stressor group

Prop_RespPoor_StressPoor weighted proportion of observations in the poor response and poor stressor group

Prop_RespPoor_StressGood weighted proportion of observations in the poor response and good stressor group

Prop_RespGood_StressPoor weighted proportion of observations in the good response and poor stressor group

Prop_RespGood_StressGood weighted proportion of observations in the good response and good stressor group

Details

Relative risk measures the relative strength of association between conditional probabilities defined for a response variable and a stressor variable, where the response and stressor variables are classified as either good (i.e., reference condition) or poor (i.e., different from reference condition). Relative risk is defined as the ratio of two conditional probabilities. The numerator of the ratio is the probability that the response variable is in poor condition given that the stressor variable is in poor condition. The denominator of the ratio is the probability that the response variable is in poor condition given that the stressor variable is in good condition. A relative risk value equal to one indicates that the response variable is independent of the stressor variable. Relative risk values greater than one measure the extent to which poor condition of the stressor variable is associated with poor condition of the response variable.

Author(s)

Tom Kincaid <Kincaid.Tom@epa.gov>

See Also

attrisk_analysis for attributable risk analysis

diffrisk_analysis for risk difference analysis

Examples

```
dframe <- data.frame(
  siteID = paste0("Site", 1:100),
  wgt = runif(100, 10, 100),
```

```
  xcoord = runif(100),
  ycoord = runif(100),
  stratum = rep(c("Stratum1", "Stratum2"), 50),
  RespVar1 = sample(c("Poor", "Good"), 100, replace = TRUE),
  RespVar2 = sample(c("Poor", "Good"), 100, replace = TRUE),
  StressVar = sample(c("Poor", "Good"), 100, replace = TRUE),
  All_Sites = rep("All Sites", 100),
  Resource_Class = rep(c("Agr", "Forest"), c(55, 45))
)
myresponse <- c("RespVar1", "RespVar2")
mystressor <- c("StressVar")
mysubpops <- c("All_Sites", "Resource_Class")
relrisk_analysis(dframe,
  vars_response = myresponse,
  vars_stressor = mystressor, subpops = mysubpops, siteID = "siteID",
  weight = "wgt", xcoord = "xcoord", ycoord = "ycoord",
  stratumID = "stratum"
)
```

`revisit_bibd` *Create a balanced incomplete block panel revisit design*

Description

Create a revisit design for panels in a survey that specifies the time periods for the units of each panel to be sampled based on searching for a D-optimal block design that is a member of the class of generalized Youden designs. The resulting design need not be a balanced incomplete block design. Based on algorithmic idea by Cook and Nachtsheim (1989) and implemented by Robert Wheeler.

Usage

```
revisit_bibd(
  n_period,
  n_pnl,
  n_visit,
  nsamp,
  panel_name = "BIB",
  begin = 1,
  skip = 1,
  iter = 30
)
```

Arguments

`n_period`	Number of time periods for the survey design. Typically, number of periods if sampling occurs once per period or number of months if sampling occurs once per month. (v, number of varieties/treatments in BIBD terms)

n_pnl	Number of panels (b, number of blocks in BIBD terms)
n_visit	Number of time periods to be visited in a panel (k, block size in BIBD terms)
nsamp	Number of samples in each panel.
panel_name	Prefix for name of each panel
begin	Numeric name of first sampling occasion, e.g. a specific period.
skip	Number of sampling occasions to skip between planned sampling periods, e.g., sampling will occur only every 5 periods if skip = 5.
iter	Maximum number of iterations in search for D-optimal Generalized Youden Design.

Details

The function uses find.BIB function from crossdes package to search for a D-optimal block design. crossdes uses package AlgDesign to search balanced incomplete block designs.

Value

A two-dimensional array of sample sizes to be sampled for each panel and each sampling occasion.

Author(s)

Tony Olsen <Olsen.Tony@epa.gov>

References

Cook R. D. and C. Nachtsheim. (1989). Computer-aided blocking of factorial and response-surface designs. *Technometrics* **31(3)**, 339-346.

See Also

revisit_dsgn to create a panel revisit design

revisit_rand to create a panel revisit design with random assignment to panels and time periods

pd_summary to summarize characteristics of a panel revisit design

Examples

```
# Balanced incomplete block design with 20 sample occasions, 20 panels,
# 3 visits to each unit, and 20 units in each panel.
revisit_bibd(n_period = 20, n_pnl = 20, n_visit = 3, nsamp = 20)
```

`revisit_dsgn` *Create a panel revisit design*

Description

Create a revisit design for panels in a survey that specifies the time periods that members of each panel will be sampled. Three basic panel design structures may be created: always revisit panel, serially alternating panels, or rotating panels.

Usage

```
revisit_dsgn(n_period, panels, begin = 1, skip = 1)
```

Arguments

`n_period`	Number of time periods for the panel design. For example, number of periods if sampling occurs once per period or number of months if sampling occurs once per month.
`panels`	List of lists where each list specifies a revisit panel structure. Each sublist consists of four components: `n` - sample size for each panel in the sublist, `pnl_dsgn` - a vector with an even number of elements specifying the panel revisit schedule in terms of the number of consecutive time periods sample units will be sampled, followed by number of consecutive time periods skipped, and then repeated as necessary. `pnl_n` - number of panels in the sublist, and `start_option` - option for starting the `revisit_dsgn` (`None`, `Partial_Begin`, or `Partial_End`) which must be the same a `pnl_dsgn`. Three basic panel structures are possible: a) if `pnl_dsgn` ends in `0`, then the sample units are visited on all subsequent time periods, b) if `pnl_dsgn` ends in `NA`, then panel follows a rotating panel structure, and c) if `pnl_dsgn` ends in any number > `0`, then panel follows a serially alternating panel structure. See details for further information.
`begin`	Numeric name of first sampling occasion, e.g. a specific period.
`skip`	Number of time periods to skip between planned sampling periods, e.g., sampling will occur only every 5 periods if `skip = 5`.

Details

The function creates revisit designs using the concepts in McDonald (2003) to specify the revisit pattern across time periods for each panel. The panel revisit schedule is specified by a vector. Odd positions in vector specify the number of consecutive time periods when panel units are sampled. Even positions in vector specify the number of consecutive time periods when panel units are not sampled.

If last even position is a `"0"`, then a single panel follows an always revisit panel structure. After satisfying the initial revisit schedule specified prior to the `"0"`, units in a panel are always visited for rest of the time periods. The simplest always revisit panel design is to revisit every sample unit on every time period, specified as `pnl_dsgn = c(1,0)` or using McDonald's notation [1-0].

If the last even position is NA, the panels follow a rotating panel structure. For example, pnl_dsgn = c(1, NA) designates that sample units in a panel will be visited once and then never again, [1-n] in McDonald's notation. pnl_dsgn =c(1, 4, 1, NA) designates that sample units in a panel will be visited once, then not sampled on next four time periods, then sampled again once at the next time period and then never sampled again, [1-4-1-n] in McDonald/s notation.

If the last even position is > 0, the panels follow a serially alternating panel structure. For example, pnl_dsgn = c(1, 4) designates that sample units in a panel will be visited once, then not sampled during the next four time periods, then sampled once and not sampled for next four time periods, and that cycle repeated until end of the number of time periods, [1-4] in McDonald's notation. pnl_dsgn = c(2, 3, 1, 4) designates that the cycle has sample units in a panel being visited during two consecutive time periods, not sampled for three consecutive time periods, sampled for one time period and then not sampled on next four time periods, and the cycle is repeated until end of the number of time periods, [2-3-1-4] in McDonald's notation.

The number of panels in a single panel design is specified by pnl_n. For an always revisit panel structure, a single panel is created and pnl_n is ignored. For a rotating panel structure, when pnl_n = NA, the number of panels is equal to n_period. Note that this should only be used when the rotating panel structure is the only panel design, i.e., no split panel design (see below for split panel details). If pnl_n = m is specified for a rotating panel design, then then number of panels will be m. For example, pnl_dsgn = c(1, 4, 1, NA) and and pnl_n = 5 means that only 5 panels will be constructed and the last time period to be sampled will be time period 10. In McDonald's notation the panel design structure is [(1-4-1-n)^5]. If the number of time periods, n_period, is 20 and no other panel design structure is specified, then the last 10 time periods will not be sampled. For serially alternating panels, when pnl_n = NA, the number of panels will be the sum of the elements in pan_dsgn (ignoring NA). If pnl_n is specified as m, then m panels will be created. For example, pnl_dsgn = c(1, 4, 1, 4) and pnl_n = 3, [(1-4-1-4)^3] in McDonald's notation, will create first three panels of the 510 serially alternating panels specified by pnl_dsgn.

A serially alternating or rotating panel revisit design may not result in the same number of units being sampled during each time period, particularly during the initial start up period. The default is to not specify a startup option ("None"). Start up option "Partial_Begin" initiates the revisit design at the last time period scheduled for sampling in the first panel. For example, a [2-3-1-4] design starts at time period 6 instead of time period 1 under the Partial_Begin option. For a serially alternating panel structure, start up option "Partial_End" initiates the revisit design at the time period that begins the second serially alternating pattern. For example, a [2-3-1-4] design starts at time period 11 instead of time period 1. For a rotating panel structure design, use of Partial_End makes the assumption that the number of panels equals the number of time periods and adds units to the last "m" panels for time periods 1 to "m" as if number of time periods was extended by "m" where "m" is one less than then the sum of the panel design. For example, a [1-4-1-4-1-n] design would result in m = 10. Note that some designs with pnl_n not equal to the number of sample occasions can produce unexpected panel designs. See examples.

Different types of panel structures can be combined, these are termed split panels by many authors, by specifying more than one list for the panels parameter. The total number of panels is the sum of the number of panels in each of the panel structures specified by the split panel design.

Value

A two-dimensional array of sample sizes to be sampled at each combination of panel and time period.

Author(s)

Tony Olsen <Olsen.Tony@epa.gov>

References

McDonald, T. (2003). Review of environmental monitoring methods: survey designs. *Environmental Monitoring and Assessment* **85**, 277-292.

See Also

revisit_bibd to create a balanced incomplete block panel revisit design

revisit_rand to create a revisit design with random assignment to panels and time periods

pd_summary to summarize characteristics of a panel revisit design

Examples

```
# One panel of  60 sample units sampled at every time period: [1-0]
revisit_dsgn(20, panels = list(
  Annual = list(
    n = 60, pnl_dsgn = c(1, 0), pnl.n = NA,
    start_option = "None"
  )
), begin = 1)

# Rotating panels of 60 units sampled once and never again: [1-n].  Number
# of panels equal n_period.
revisit_dsgn(20,
  panels = list(
    R60N = list(n = 60, pnl_dsgn = c(1, NA), pnl_n = NA, start_option = "None")
  ),
  begin = 1
)

# Serially alternating panel with three visits to sample unit then skip
# next two time periods: [3-2]
revisit_dsgn(20, panels = list(
  SA60PE = list(
    n = 20, pnl_dsgn = c(3, 2), pnl_n = NA,
    start_option = "Partial_End"
  )
), begin = 1)

# Split panel of sample units combining above two panel designs: [1-0, 1-n]
revisit_dsgn(n_period = 20, begin = 2017, panels = list(
  Annual = list(
    n = 60, pnl_dsgn = c(1, 0), pnl.n = NA,
    start_option = "None"
  ),
  R60N = list(n = 60, pnl_dsgn = c(1, NA), pnl_n = NA, start_option = "None")
))
```

`revisit_rand` *Create a revisit design with random assignment to panels and time periods*

Description

Create a revisit design for a survey that specifies the panels and time periods that will be sampled by random selection of panels and time periods. Three options for random assignments are `"period"` where the number of time periods to be sampled in a panel is fixed, `"panel"` where the number panels to be sampled in a time period is fixed, and `"none"` where the number of panel-period combinations is fixed.

Usage

```
revisit_rand(
  n_period,
  n_pnl,
  rand_control = "period",
  n_visit,
  nsamp,
  panel_name = "Random",
  begin = 1,
  skip = 1
)
```

Arguments

`n_period`	Number of time periods for the survey design. Typically, number of periods if sampling occurs once per period or number of months if sampling occurs once per month. (v, number of varieties (or treatments) in BIBD terms)
`n_pnl`	Number of panels
`rand_control`	Character value must be `"none"`, `"panel"`, or `"period"`. Specifies whether the number of sample events will be fixed for each panel (`"panel"`), for each sample occasion (`"occasion"`), or for total panel-period combinations (`"none"`). Default is `"panel"`.
`n_visit`	If `rand_control` is `"panel"`, this is the number of panels that will be sampled in each time period. If rand_control is `"period"`, this is the number of time periods to be sampled in each panel. If `rand_control` is `"none"`, this is the total number of panel-period combinations that will have units sampled in the revisit design.
`nsamp`	Number of samples in each panel.
`panel_name`	Prefix for name of each panel
`begin`	Numeric name of first sampling occasion, e.g. a specific period.
`skip`	Number of sampling occasions to skip between planned sampling periods, e.g., sampling will occur only every 5 periods if `skip = 5`.

Details

The revisit design for a survey is created by random selection of panels and time periods that will have sample events. The number of sample occasions that will be visited by a panel is random.

Value

A two-dimensional array of sample sizes to be sampled for each panel and each time period.

Author(s)

Tony Olsen <Olsen.Tony@epa.gov>

See Also

`revisit_bibd` create a balanced incomplete block panel revisit design

`revisit_dsgn` create a panel revisit design

`pd_summary` to summarize characteristics of a panel revisit design

Examples

```
revisit_rand(
  n_period = 20, n_pnl = 10, rand_control = "none", n_visit = 50,
  nsamp = 20
)
revisit_rand(
  n_period = 20, n_pnl = 10, rand_control = "panel", n_visit = 5,
  nsamp = 10
)
revisit_rand(
  n_period = 20, n_pnl = 10, rand_control = "period",
  n_visit = 5, nsamp = 10
)
```

---

sp_balance *Calculate spatial balance metrics*

Description

This function measures the spatial balance (with respect to the sampling frame) of design sites using Voronoi polygons (Dirichlet tessellations).

Usage

```
sp_balance(
  object,
  sframe,
  stratum_var = NULL,
  ip = NULL,
  metrics = "pielou",
  extents = FALSE
)
```

Arguments

object	An sf object containing some design sites.
sframe	The sampling frame as an sf object. The coordinate system for sframe must be one where distance for coordinates is meaningful.
stratum_var	The name of the stratum variable in object and sframe. If NULL (the default), no strata is assumed. If a single character vector is provided, it is assumed this is the name of the stratum variable in object and sframe. If a two-dimensional character vector is provided, one element must be named "object" and corresponds to the name of the stratum variable in object, while the other element must be named "sframe" and corresponds to the name of the stratum variable in sframe.
ip	Inclusion probabilities associated with each row of sframe. If these are not provided, an equal probability design is assumed (within strata).
metrics	A character vector of spatial balance metrics: pielou Pielou's Evenness Index (the default). This statistic can take on a value between zero and one. simpsons Simpsons Evenness Index. This statistic can take on a value between zero and logarithm of the sample size. rmse Root-Mean-Squared Error. This statistic can take on a value between zero and infinity. mse Mean-Squared Error. This statistic can take on a value between zero and infinity. mae Mean-Absolute Error. This statistic can take on a value between zero and infinity. medae Median-Absolute Error. This statistic can take on a value between zero and infinity. chisq Chi-Squared Loss. This statistic can take on a value between zero and infinity. All spatial balance metrics have a lower bound of zero, which indicates perfect spatial balance. As the metric value increases, the spatial balance decreases.
extents	Should the extent (total units) within each Voronoi polygon be returned? Defaults to FALSE.

Value

A data frame with columns providing the stratum (stratum), spatial balance metric (metric), and spatial balance (value).

Author(s)

Michael Dumelle <Dumelle.Michael@epa.gov>

Examples

```
## Not run:
sample <- grts(NE_Lakes, 30)
sp_balance(sample$sites_base, NE_Lakes)
strata_n <- c(low = 25, high = 30)
sample_strat <- grts(NE_Lakes, n_base = strata_n, stratum_var = "ELEV_CAT")
sp_balance(sample_strat$sites_base, NE_Lakes, stratum_var = "ELEV_CAT", metric = "rmse")

## End(Not run)
```

---

sp_frame *sp_frame objects*

Description

Turn sampling frames or analysis data into an sp_frame object or transform sp_frame objects back into their original object.

Usage

```
sp_frame(frame)

sp_unframe(sp_frame)
```

Arguments

frame	A sampling frame or analysis data
sp_frame	An sp_frame object.

Details

The sp_frame() function assigns frame class sp_frame to be used by summary() and plot(). sp_frame() objects can sometimes clash with other sf and tidyverse generics, so un_spframe() removes class sp_frame(), leaving the original classes of frame intact.

Value

An sp_frame object.

Examples

```
NE_Lakes <- sp_frame(NE_Lakes)
class(NE_Lakes)
NE_Lakes <- sp_unframe(NE_Lakes)
class(NE_Lakes)
```

`sp_plot` *Plot sampling frames, design sites, and analysis data.*

Description

This function plots sampling frames, design sites, and analysis data. If the left-hand side of the formula is empty, plots are of the distributions of the right-hand side variables. If the left-hand side of the variable contains a variable, plots are of the left-hand size variable for each level of each right-hand side variable. This function is largely built on `plot.sf()`, and all spsurvey plotting methods can supply additional arguments to `plot.sf()`. For more information on plotting in `sf`, run `?sf::plot.sf()`. Equivalent to `spsurvey::plot()`; both are currently maintained for backwards compatibility.

Usage

```
sp_plot(object, ...)

## Default S3 method:
sp_plot(
  object,
  formula = ~1,
  xcoord,
  ycoord,
  crs,
  var_args = NULL,
  varlevel_args = NULL,
  geom = FALSE,
  onlyshow = NULL,
  fix_bbox = TRUE,
  ...
)

## S3 method for class 'sp_design'
sp_plot(
  object,
  sframe = NULL,
  formula = ~siteuse,
  siteuse = NULL,
  var_args = NULL,
  varlevel_args = NULL,
  geom = FALSE,
```

```
  onlyshow = NULL,
  fix_bbox = TRUE,
  ...
)
```

Arguments

object	An object to plot. When plotting sampling frames or analysis data, a data frame or sf object. When plotting design sites, an object created by grts() or irs() (which has class sp_design).
...	Additional arguments to pass to plot.sf().
formula	A formula. One-sided formulas are used to summarize the distribution of numeric or categorical variables. For one-sided formulas, variable names are placed to the right of ~ (a right-hand side variable). Two sided formulas are used to summarize the distribution of a left-hand side variable for each level of each right-hand side categorical variable in the formula. Note that only for two-sided formulas are numeric right-hand side variables coerced to a categorical variables. If an intercept is included as a right-hand side variable (whether the formula is one-sided or two-sided), the total will also be summarized. When plotting sampling frames or analysis data, the default formula is ~ 1. When plotting design sites, siteuse should be used in the formula, and the default formula is ~ siteuse.
xcoord	Name of the x-coordinate (east-west) in object (only required if object is not an sf object).
ycoord	Name of y (north-south)-coordinate in object (only required if object is not an sf object).
crs	Projection code for xcoord and ycoord (only required if object is not an sf object).
var_args	A named list. The name of each list element corresponds to a right-hand side variable in formula. Values in the list are composed of graphical arguments that are to be passed to **every** level of the variable. To see all graphical arguments available, run ?plot.sf.
varlevel_args	A named list. The name of each list element corresponds to a right-hand side variable in formula. The first element in this list should be "levels" and contain all levels of the particular right-hand side variable. Subsequent names correspond to graphical arguments that are to be passed to the specified levels (in order) of the right-hand side variable. Values for each graphical argument must be specified for each level of the right-hand side variable, but applicable sf defaults will be matched by inputting the value NA. To see all graphical arguments available, run ?plot.sf
geom	Should separate geometries for each level of the right-hand side formula variables be plotted? Defaults to FALSE.
onlyshow	A string indicating the single level of the single right-hand side variable for which a summary is requested. This argument is only used when a single right-hand side variable is provided.

fix_bbox	Should the geometry bounding box be fixed across plots? If a length-four vector with names "xmin", "ymin", "xmax", and "ymax" and values indicating bounding box edges, the bounding box will be fixed as fix_bbox across plots. If TRUE, the bounding box will be fixed across plots as the bounding box of object. If FALSE, the bounding box will vary across plots according to the unique geometry for each plot. Defaults to TRUE.
sframe	The sampling frame (an sf object) to plot alongside design sites. This argument is only used when object corresponds to the design sites.
siteuse	A character vector of site types to include when plotting design sites. It can only take on values "sframe" (sampling frame), "Legacy" (for legacy sites), "Base" (for base sites), "Over" (for n_over replacement sites), and "Near" (for n_near replacement sites). The order of sites represents the layering in the plot (e.g. siteuse = c("Base", "Legacy") will plot legacy sites on top of base sites. Defaults to all non-NULL elements in x and y with plot order "sframe", "Legacy", "Base", "Over", "Near".

Author(s)

Michael Dumelle <Dumelle.Michael@epa.gov>

Examples

```
## Not run:
data("NE_Lakes")
sp_plot(NE_Lakes, formula = ~ELEV_CAT)
sample <- grts(NE_Lakes, 30)
sp_plot(sample, NE_Lakes)
data("NLA_PNW")
sp_plot(NLA_PNW, formula = ~BMMI)

## End(Not run)
```

---

sp_rbind *Combine rows from GRTS or IRS samples.*

Description

This function row binds the sites_legacy, sites_base, sites_over, and sites_near objects from a GRTS or IRS sample into a single sf object. This function is most useful when a single sf object that contains all design sites is desired (e.g. writing out a single shapefile using sf::write_sf()).

Usage

```
sp_rbind(object, siteuse = NULL)
```

Arguments

object	The design sites (output from grts() or irs()).
siteuse	A character vector of site types to return. Can contain "Legacy" (for legacy sites), "Base" (for base sites), "Over" (for n_over replacement sites), and "Near" (for n_near replacement sites). The default is NULL, which returns all non-NULL output from object$sites_legacy, object$sites_base, object$sites_over, and object$sites_near.

Value

A single sf object containing all requested design sites.

Author(s)

Michael Dumelle <Dumelle.Michael@epa.gov>

Examples

```
## Not run:
sample <- grts(NE_Lakes, 50, n_over = 10)
sample <- sp_rbind(sample)
write_sf(sample, "mypath/sample.shp")

## End(Not run)
```

---

sp_summary *Summarize sampling frames, design sites, and analysis data.*

Description

sp_summary() summarizes sampling frames, design sites, and analysis data. The right-hand of the formula specifies the variables (or factors) to summarize by. If the left-hand side of the formula is empty, the summary will be of the distributions of the right-hand side variables. If the left-hand side of the formula contains a variable, the summary will be of the left-hand size variable for each level of each right-hand side variable. Equivalent to spsurvey::summary(); both are currently maintained for backwards compatibility.

Usage

```
sp_summary(object, ...)

## Default S3 method:
sp_summary(object, formula = ~1, onlyshow = NULL, ...)

## S3 method for class 'sp_design'
sp_summary(object, formula = ~siteuse, siteuse = NULL, onlyshow = NULL, ...)
```

Arguments

object	An object to summarize. When summarizing sampling frames, an sf object. When summarizing design sites, an object created by grts() or irs() (which has class sp_design). When summarizing analysis data, a data frame or an sf object.
...	Additional arguments to pass to sp_summary(). If the left-hand side of the formula is empty, the appropriate generic arguments are passed to summary.data.frame. If the left-hand side of the formula is provided, the appropriate generic arguments are passed to summary.default.
formula	A formula. One-sided formulas are used to summarize the distribution of numeric or categorical variables. For one-sided formulas, variable names are placed to the right of ~ (a right-hand side variable). Two sided formulas are used to summarize the distribution of a left-hand side variable for each level of each right-hand side categorical variable in the formula. Note that only for two-sided formulas are numeric right-hand side variables coerced to a categorical variables. If an intercept is included as a right-hand side variable (whether the formula is one-sided or two-sided), the total will also be summarized. When summarizing sampling frames or analysis data, the default formula is ~ 1. When summarizing design sites, siteuse should be used in the formula, and the default formula is ~ siteuse.
onlyshow	A string indicating the single level of the single right-hand side variable for which a summary is requested. This argument is only used when a single right-hand side variable is provided.
siteuse	A character vector indicating the design sites for which summaries are requested in object. Defaults to computing summaries for each non-NULL sites_* list in object.

Value

If the left-hand side of the formula is empty, a named list containing summaries of the count distribution for each right-hand side variable is returned. If the left-hand side of the formula contains a variable, a named list containing five number summaries (numeric left-hand side) or tables (categorical or factor left hand side) is returned for each right-hand side variable.

Author(s)

Michael Dumelle <Dumelle.Michael@epa.gov>

Examples

```
## Not run:
data("NE_Lakes")
sp_summary(NE_Lakes, ELEV ~ 1)
sp_summary(NE_Lakes, ~ ELEV_CAT * AREA_CAT)
sample <- grts(NE_Lakes, 100)
sp_summary(sample, ~ ELEV_CAT * AREA_CAT)

## End(Not run)
```

stopprnt *Print grts() and irs() errors.*

Description

This function prints the error messages vector in the `grts` and `irs` functions.

Usage

```
stopprnt(stop_df = get("stop_df", envir = .GlobalEnv), m = 1:nrow(stop_df))
```

Arguments

`stop_df`	Data frame that contains stop messages. The default is `stop_df`, which is the name given to the stop data frame created by functions in the spsurvey package.
`m`	Vector of indices for stop messages that are to be printed. The default is a vector containing the integers from 1 through the number of rows in `stop_df`, which will print all stop messages in the data frame.

Value

Printed errors

Author(s)

Tony Olsen <Olsen.Tony@epa.gov>

summary *Summarize sampling frames, design sites, and analysis data.*

Description

`summary()` summarizes sampling frames, design sites, and analysis data. The right-hand of the formula specifies the variables (or factors) to summarize by. If the left-hand side of the formula is empty, the summary will be of the distributions of the right-hand side variables. If the left-hand side of the formula contains a variable, the summary will be of the left-hand size variable for each level of each right-hand side variable. Equivalent to `sp_summary()`; both are currently maintained for backwards compatibility.

Usage

```
## S3 method for class 'sp_frame'
summary(object, formula = ~1, onlyshow = NULL, ...)

## S3 method for class 'sp_design'
summary(object, formula = ~siteuse, siteuse = NULL, onlyshow = NULL, ...)
```

Arguments

object	An object to summarize. When summarizing sampling frames, an sf object given the appropriate class using sp_frame. When summarizing design sites, an object created by grts() or irs() (which has class sp_design). When summarizing analysis data, a data frame or an sf object given the appropriate class using sp_frame.
formula	A formula. One-sided formulas are used to summarize the distribution of numeric or categorical variables. For one-sided formulas, variable names are placed to the right of ~ (a right-hand side variable). Two sided formulas are used to summarize the distribution of a left-hand side variable for each level of each right-hand side categorical variable in the formula. Note that only for two-sided formulas are numeric right-hand side variables coerced to a categorical variables. If an intercept is included as a right-hand side variable (whether the formula is one-sided or two-sided), the total will also be summarized. When summarizing sampling frames or analysis data, the default formula is ~ 1. When summarizing design sites, siteuse should be used in the formula, and the default formula is ~ siteuse.
onlyshow	A string indicating the single level of the single right-hand side variable for which a summary is requested. This argument is only used when a single right-hand side variable is provided.
...	Additional arguments to pass to sp_summary(). If the left-hand side of the formula is empty, the appropriate generic arguments are passed to summary.data.frame. If the left-hand side of the formula is provided, the appropriate generic arguments are passed to summary.default.
siteuse	A character vector indicating the design sites for which summaries are requested in object. Defaults to computing summaries for each non-NULL sites_* list in object.

Value

If the left-hand side of the formula is empty, a named list containing summaries of the count distribution for each right-hand side variable is returned. If the left-hand side of the formula contains a variable, a named list containing five number summaries (numeric left-hand side) or tables (categorical or factor left hand side) is returned for each right-hand side variable.

Author(s)

Michael Dumelle <Dumelle.Michael@epa.gov>

Examples

```
## Not run:
data("NE_Lakes")
summary(NE_Lakes, ELEV ~ 1)
summary(NE_Lakes, ~ ELEV_CAT * AREA_CAT)
sample <- grts(NE_Lakes, 100)
summary(sample, ~ ELEV_CAT * AREA_CAT)
```

```
## End(Not run)
```

`trend_analysis` *Trend analysis*

Description

This function organizes input and output for estimation of trend across time for a series of samples (for categorical and continuous variables). Trend is estimated using the analytical procedure identified by the model arguments. For categorical variables, the choices for the `model_cat` argument are: (1) simple linear regression, (2) weighted linear regression, and (3) generalized linear mixed-effects model. For continuous variables, the choices for the `model_cont` argument are: (1) simple linear regression, (2) weighted linear regression, and (3) linear mixed-effects model. The analysis data, `dframe`, can be either a data frame or a simple features (`sf`) object. If an `sf` object is used, coordinates are extracted from the geometry column in the object, arguments `xcoord` and `ycoord` are assigned values `"xcoord"` and `"ycoord"`, respectively, and the geometry column is dropped from the object.

Usage

```
trend_analysis(
  dframe,
  vars_cat = NULL,
  vars_cont = NULL,
  subpops = NULL,
  model_cat = "SLR",
  cat_rhs = NULL,
  model_cont = "LMM",
  cont_rhs = NULL,
  siteID = "siteID",
  yearID = "year",
  weight = "weight",
  xcoord = NULL,
  ycoord = NULL,
  stratumID = NULL,
  clusterID = NULL,
  weight1 = NULL,
  xcoord1 = NULL,
  ycoord1 = NULL,
  sizeweight = FALSE,
  sweight = NULL,
  sweight1 = NULL,
  fpc = NULL,
  popsize = NULL,
  invprboot = TRUE,
  nboot = 1000,
  vartype = "local",
```

```
  jointprob = "overton",
  conf = 95,
  All_Sites = FALSE,
  subset_local = TRUE
)
```

Arguments

dframe	Data to be analyzed (analysis data). A data frame or sf object containing survey design variables, response variables, and subpopulation (domain) variables.
vars_cat	Vector composed of character values that identify the names of categorical response variables in dframe. If argument model_cat equals "GLMM", the categorical variables in the dframe data frame must be factors each of which has two levels, where the second level will be assumed to specify "success". The default value is NULL.
vars_cont	Vector composed of character values that identify the names of continuous response variables in dframe. The default value is NULL.
subpops	Vector composed of character values that identify the names of subpopulation (domain) variables in dframe. If a value is not provided, the value "All_Sites" is assigned to the subpops argument and a factor variable named "All_Sites" that takes the value "All Sites" is added to dframe. The default value is NULL.
model_cat	Character value identifying the analytical procedure used for trend estimation for categorical variables. The choices are: "SLR" (simple linear regression), "WLR" (weighted linear regression), and "GLMM" (generalized linear mixed-effects model). The default value is "SLR".
cat_rhs	Character value specifying the right hand side of the formula for a generalized linear mixed-effects model. If a value is not provided, the argument is assigned a value that specifies the Piepho and Ogutu (2002) model. The default value is NULL.
model_cont	Character value identifying the analytical procedure used for trend estimation for continuous variables. The choices are: "SLR" (simple linear regression), "WLR" (weighted linear regression), and "LMM" (linear mixed-effects model). The default value is "LMM".
cont_rhs	Character value specifying the right hand side of the formula for a linear mixed-effects model. If a value is not provided, the argument is assigned a value that specifies the Piepho and Ogutu (2002) model. The default value is NULL.
siteID	Character value providing name of the site ID variable in dframe. If repeat visit sites are present, the site ID value for each revisit site will be the same for each survey. For a two-stage sample, the site ID variable identifies stage two site IDs. The default value is "siteID".
yearID	Character value providing name of the time period variable in dframe, which must be numeric and will be forced to numeric if it is not. The default assumption is that the time period variable is years. The default value is "year".
weight	Character value providing name of the design weight variable in dframe. For a two-stage sample, the weight variable identifies stage two weights. The default value is "weight".

xcoord — Character value providing name of the x-coordinate variable in dframe. For a two-stage sample, the x-coordinate variable identifies stage two x-coordinates. Note that x-coordinates are required for calculation of the local mean variance estimator. If dframe is an sf object, this argument is not required (as the geometry column in dframe is used to find the x-coordinate). The default value is NULL.

ycoord — Character value providing name of the y-coordinate variable in dframe. For a two-stage sample, the y-coordinate variable identifies stage two y-coordinates. Note that y-coordinates are required for calculation of the local mean variance estimator. If dframe is an sf object, this argument is not required (as the geometry column in dframe is used to find the y-coordinate). The default value is NULL.

stratumID — Character value providing name of the stratum ID variable in dframe. The default value is NULL.

clusterID — Character value providing name of the cluster (stage one) ID variable in dframe. Note that cluster IDs are required for a two-stage sample. The default value is NULL.

weight1 — Character value providing name of the stage one weight variable in dframe. The default value is NULL.

xcoord1 — Character value providing name of the stage one x-coordinate variable in dframe. Note that x-coordinates are required for calculation of the local mean variance estimator. The default value is NULL.

ycoord1 — Character value providing name of the stage one y-coordinate variable in dframe. Note that y-coordinates are required for calculation of the local mean variance estimator. The default value is NULL.

sizeweight — Logical value that indicates whether size weights should be used during estimation, where TRUE = use size weights and FALSE = do not use size weights. To employ size weights for a single-stage sample, a value must be supplied for argument weight. To employ size weights for a two-stage sample, values must be supplied for arguments weight and weight1. The default value is FALSE.

sweight — Character value providing name of the size weight variable in dframe. For a two-stage sample, the size weight variable identifies stage two size weights. The default value is NULL.

sweight1 — Character value providing name of the stage one size weight variable in dframe. The default value is NULL.

fpc — Object that specifies values required for calculation of the finite population correction factor used during variance estimation. The object must match the survey design in terms of stratification and whether the design is single-stage or two-stage. For an unstratified design, the object is a vector. The vector is composed of a single numeric value for a single-stage design. For a two-stage unstratified design, the object is a named vector containing one more than the number of clusters in the sample, where the first item in the vector specifies the number of clusters in the population and each subsequent item specifies the number of stage two units for the cluster. The name for the first item in the vector is arbitrary. Subsequent names in the vector identify clusters and must match the cluster IDs. For a stratified design, the object is a named list of vectors, where

names must match the strata IDs. For each stratum, the format of the vector is identical to the format described for unstratified single-stage and two-stage designs. Note that the finite population correction factor is not used with the local mean variance estimator.

Example fpc for a single-stage unstratified survey design:

```
fpc <- 15000
```

Example fpc for a single-stage stratified survey design:

```
fpc <- list(
    Stratum_1 = 9000,
    Stratum_2 = 6000)
```

Example fpc for a two-stage unstratified survey design:

```
fpc <- c(
    Ncluster = 150,
    Cluster_1 = 150,
    Cluster_2 = 75,
    Cluster_3 = 75,
    Cluster_4 = 125,
    Cluster_5 = 75)
```

Example fpc for a two-stage stratified survey design:

```
fpc <- list(
    Stratum_1 = c(
      Ncluster_1 = 100,
      Cluster_1 = 125,
      Cluster_2 = 100,
      Cluster_3 = 100,
      Cluster_4 = 125,
      Cluster_5 = 50),
    Stratum_2 = c(
      Ncluster_2 = 50,
      Cluster_1 = 75,
      Cluster_2 = 150,
      Cluster_3 = 75,
      Cluster_4 = 75,
      Cluster_5 = 125))
```

`popsize` Object that provides values for the population argument of the `calibrate` or `postStratify` functions in the survey package. If a value is provided for popsize, then either the `calibrate` or `postStratify` function is used to modify the survey design object that is required by functions in the survey package. Whether to use the `calibrate` or `postStratify` function is dictated by the format of popsize, which is discussed below. Post-stratification adjusts the sampling and replicate weights so that the joint distribution of a set of post-stratifying variables matches the known population joint distribution. Calibration, generalized raking, or GREG estimators generalize post-stratification and raking by calibrating a sample to the marginal totals of variables in a linear regression model. For the `calibrate` function, the object is a named list, where

the names identify factor variables in dframe. Each element of the list is a named vector containing the population total for each level of the associated factor variable. For the postStratify function, the object is either a data frame, table, or xtabs object that provides the population total for all combinations of selected factor variables in the dframe data frame. If a data frame is used for popsize, the variable containing population totals must be the last variable in the data frame. If a table is used for popsize, the table must have named dimnames where the names identify factor variables in the dframe data frame. If the popsize argument is equal to NULL, then neither calibration nor post-stratification is performed. The default value is NULL.

Example popsize for calibration:

```
popsize <- list(
    Ecoregion = c(
      East = 750,
      Central = 500,
      West = 250),
    Type = c(
      Streams = 1150,
      Rivers = 350))
```

Example popsize for post-stratification using a data frame:

```
popsize <- data.frame(
    Ecoregion = rep(c("East", "Central", "West"),
      rep(2, 3)),
    Type = rep(c("Streams", "Rivers"), 3),
    Total = c(575, 175, 400, 100, 175, 75))
```

Example popsize for post-stratification using a table:

```
popsize <- with(MySurveyFrame,
    table(Ecoregion, Type))
```

Example popsize for post-stratification using an xtabs object:

```
popsize <- xtabs(~Ecoregion + Type,
    data = MySurveyFrame)
```

invprboot — Logical value that indicates whether the inverse probability bootstrap procedure is used to calculate trend parameter estimates. This bootstrap procedure is only available for the "LMM" option for continuous variables. Inverse probability references the design weights, which are the inverse of the sample inclusion probabilities. The default value is TRUE.

nboot — Numeric value for the number of bootstrap iterations. The default is 1000.

vartype — Character value providing choice of the variance estimator, where "Local" = the local mean estimator, "SRS" = the simple random sampling estimator, "HT" = the Horvitz-Thompson estimator, and "YG" = the Yates-Grundy estimator. The default value is "local".

jointprob — Character value providing choice of joint inclusion probability approximation for use with Horvitz-Thompson and Yates-Grundy variance estimators, where "overton" indicates the Overton approximation, "hr" indicates the Hartley_Rao

approximation, and "brewer" equals the Brewer approximation. The default value is "overton".

conf Numeric value for the Gaussian-based confidence level. The default is 95.

All_Sites A logical variable used when subpops is not NULL. If All_Sites is TRUE, then alongside the subpopulation output, output for all sites (ignoring subpopulations) is returned for each variable in vars. If All_Sites is FALSE, then alongside the subpopulation output, output for all sites (ignoring subpopulations) is not returned for each variable in vars. The default is FALSE.

subset_local Logical value that controls how the local mean variance estimator (the default vartype = "local") handles subpopulations for the per-year category/mean estimates that feed the "SLR"/"WLR" trend models. See cat_analysis for the full explanation. Only affects model_cat/model_cont = "WLR": for "SLR", the per-year variance estimate is computed but never used (an unweighted lm() trend line is fit through the point estimates alone), so subset_local has no effect on "SLR" results. Has no effect on "GLMM"/"LMM" models, which do not use these functions. 'subset_local' only applies when 'vartype = "local"'. The default value is TRUE.

Value

The analysis results. A list composed of two data frames containing trend estimates for all combinations of population Types, subpopulations within Types, and response variables. For categorical variables, trend estimates are calculated for each category of the variable. The two data frames in the output list are:

catsum data frame containing trend estimates for categorical variables

contsum data frame containing trend estimates for continuous variables

For the SLR and WLR model options, the data frame contains the following variables:

Type subpopulation (domain) name

Subpopulation subpopulation name within a domain

Indicator response variable

Trend_Estimate trend estimate

Trend_Std_Error trend standard error

Trend_LCBxxPct trend xx% (default 95%) lower confidence bound

Trend_UCBxxPct trend xx% (default 95%) upper confidence bound

Trend_p_Value trend p-value

Intercept_Estimate intercept estimate

Intercept_Std_Error intercept standard error

Intercept_LCBxxPct intercept xx% (default 95%) lower confidence bound

Intercept_UCBxxPct intercept xx% (default 95%) upper confidence bound

Intercept_p_Value intercept p-value

R_Squared R-squared value

Adj_R_Squared adjusted R-squared value

For the GLMM and LMM model options, contents of the data frames will vary depending on the model specified by arguments `cat_rhs` and `cont_rhs`. For the default PO model, the data frame contains the following variables:

Type subpopulation (domain) name

Subpopulation subpopulation name within a domain

Indicator response variable

Trend_Estimate trend estimate

Trend_Std_Error trend standard error

Trend_LCBxxPct trend xx% (default 95%) lower confidence bound

Trend_UCBxxPct trend xx% (default 95%) upper confidence bound

Trend_p_Value trend p-value

Intercept_Estimate intercept estimate

Intercept_Std_Error intercept standard error

Intercept_LCBxxPct intercept xx% (default 95%) lower confidence bound

Intercept_UCBxxPct intercept xx% (default 95%) upper confidence bound

Intercept_p_Value intercept p-value

Var_SiteInt variance of the site intercepts

Var_SiteTrend variance of the site trends

Corr_SiteIntSlope correlation of site intercepts and site trends

Var_Year year variance

Var_Residual residual variance

AIC generalized Akaike Information Criterion

Details

For the simple linear regression (SLR) model, a design-based estimate of the category proportion (categorical variables) or the mean (continuous variables) is calculated for each time period (year). Four choices of variance estimator are available for calculating variance of the design-based estimates: (1) the local mean estimator, (2) the simple random sampling estimator, (3) the Horvitz-Thompson estimator, and (4) the Yates-Grundy estimator. For the Horvitz-Thompson and Yates-Grundy estimators, there are three choices for calculating joint inclusion probabilities: (1) the Overton approximation, (2) the Hartley-Rao approximation, and (3) the Brewer approximation. The `lm` function in the stats package is used to fit a linear model using a `formula` argument that specifies the proportion or mean estimates as the response variable and years as the regressor variable. For fitting the SLR model, the `yearID` variable from the `dframe` argument is modified by subtracting the minimum value of years from all values of the variable. Parameter estimates are extracted from the object returned by the `lm` function. For the weighted linear regression (WLR) model, the process is the same as the SLR model except that the inverse of the variances of the proportion or mean estimates is used as the `weights` argument in the call to the `lm` function. For the LMM option, the `lmer` function in the lme4 package is used to fit a linear mixed-effects model for trend across years. For both the GLMM and LMM options, the default Piepho and Ogutu (PO) model

includes fixed effects for intercept and trend (slope) and random effects for intercept and trend for individual sites, where the siteID variable from the dframe argument identifies sites. Correlation between the random effects for site intercepts and site trends is included in the model. Finally, the PO model contains random effects for year variance and residual variance. For the GLMM and LMM options, arguments cat_rhs and cont_rhs, respectively, can be used to specify the right hand side of the model formula. Internally, a variable named Wyear is created that is useful for specifying the cat_rhs and cont_rhs arguments. The Wyear variable is created by subtracting the minimum value of the yearID variable from all values of the variable. If argument invprboot is FALSE, parameter estimates are extracted from the object returned by the lmer function. If argument invprboot is TRUE, the boot function in the boot package is used to generate bootstrap replicates using a function named bootfcn as the statistic argument passed to the boot function. For each bootstrap replicate, bootfcn calls the glmer or lmer function, as appropriate, using the specified model. design weights identified by the weight argument for the trend_analysis function are passed as the weights argument for the boot function, which specifies importance weights. Using the design weights as the weights argument ensures that bootstrap replicates are representative of the survey population. Parameter estimates are calculated using the object returned by the boot function.

Author(s)

Tom Kincaid <Kincaid.Tom@epa.gov>

See Also

change_analysis for change analysis

Examples

```
# Example using a categorical variable with three resource classes and a
# continuous variable
mydframe <- data.frame(
  siteID = rep(paste0("Site", 1:40), rep(5, 40)),
  yearID = rep(seq(2000, 2020, by = 5), 40),
  wgt = rep(runif(40, 10, 100), rep(5, 40)),
  xcoord = rep(runif(40), rep(5, 40)),
  ycoord = rep(runif(40), rep(5, 40)),
  All_Sites = rep("All Sites", 200),
  Region = sample(c("North", "South"), 200, replace = TRUE),
  Resource_Class = sample(c("Good", "Fair", "Poor"), 200, replace = TRUE),
  ContVar = rnorm(200, 10, 1)
)
myvars_cat <- c("Resource_Class")
myvars_cont <- c("ContVar")
mysubpops <- c("All_Sites", "Region")
trend_analysis(
  dframe = mydframe,
  vars_cat = myvars_cat,
  vars_cont = myvars_cont,
  subpops = mysubpops,
  model_cat = "WLR",
  model_cont = "SLR",
```

```
  siteID = "siteID",
  yearID = "yearID",
  weight = "wgt",
  xcoord = "xcoord",
  ycoord = "ycoord"
)
```

---

warnprnt *Print grts(), irs()), and analysis function warnings*

Description

This function prints the warnings messages from the `grts()`, `irs()`, and analysis functions.

Usage

```
warnprnt(warn_df = get("warn_df", envir = .GlobalEnv), m = 1:nrow(warn_df))
```

Arguments

`warn_df`	Data frame that contains warning messages. The default is `"warn_df"`, which is the name given to the warnings data frame created by functions in the spsurvey package.
`m`	Vector of indices for warning messages that are to be printed. The default is a vector containing the integers from 1 through the number of rows in `warn_df`, which will print all warning messages in the data frame.

Value

Printed warnings.

Author(s)

Tom Kincaid <Kincaid.Tom@epa.gov>

Index

∗ **adjustment**
adjwgtNR, 5
∗ **datasets**
Illinois_River, 63
Illinois_River_Legacy, 63
Lake_Ontario, 70
NE_Lakes, 73
NE_Lakes_df, 73
NE_Lakes_Legacy, 74
NLA_PNW, 75
NRSA_EPA7, 75
∗ **misc**
adjwgt, 4
∗ **non-response**
adjwgtNR, 5
∗ **plot**
cont_cdfplot, 39
∗ **survey**
adjwgt, 4
adjwgtNR, 5
attrisk_analysis, 7
cat_analysis, 14
cdf_plot, 19
change_analysis, 22
cont_analysis, 32
cont_cdfplot, 39
cont_cdftest, 41
cov_panel_dsgn, 47
diffrisk_analysis, 49
localmean_cov, 71
localmean_var, 72
pd_summary, 76
plot.sp_CDF, 80
power_dsgn, 83
ppd_plot, 85
relrisk_analysis, 88
revisit_bibd, 95
revisit_dsgn, 97
revisit_rand, 100
trend_analysis, 111
∗ **univar**
attrisk_analysis, 7
cat_analysis, 14
cont_analysis, 32
diffrisk_analysis, 49
relrisk_analysis, 88
∗ **weight**
adjwgtNR, 5

adjwgt, 4
adjwgtNR, 5
ash1_wgt, 6
attrisk_analysis, 7, *55*, *94*

cat_analysis, 14, *38*, *46*, *116*
cdf_plot, 19, *41*, *46*
change_analysis, 22, *118*
cont_analysis, *19*, 32
cont_cdfplot, *21*, 39, *46*, *82*
cont_cdftest, *21*, *41*, 41, *82*
cov_panel_dsgn, 47

diffrisk_analysis, *13*, 49, *94*

errorprnt, 55

grts, 56, *70*

Illinois_River, 63
Illinois_River_Legacy, 63
irs, *62*, 64

Lake_Ontario, 70
localmean_cov, 71
localmean_var, 72
localmean_weight, 72

NE_Lakes, 73
NE_Lakes_df, 73
NE_Lakes_Legacy, 74

NLA_PNW, 75
NRSA_EPA7, 75

pd_summary, 76, *96*, *99*, *101*
plot, 77
plot.sp_CDF, 80
power_dsgn, *49*, 83
ppd_plot, *85*, 85

relrisk_analysis, *13*, *55*, 88
revisit_bibd, 95, *99*, *101*
revisit_dsgn, *96*, 97, *101*
revisit_rand, *96*, *99*, 100

sp_balance, 101
sp_frame, 103
sp_plot, 104
sp_rbind, 106
sp_summary, 107
sp_unframe *(*sp_frame*)*, 103
spsurvey *(*spsurvey-package*)*, 3
spsurvey-package, 3
stopprnt, 109
summary, 109

trend_analysis, *31*, 111

warnprnt, 119