Package 'FFD'

July 21, 2025

Version 1.0-9

Date 2022-11-08

Author Ian Kopacka

Maintainer Ian Kopacka <ian.kopacka@ages.at>

Title Freedom from Disease

Depends R (>= 2.10), methods, graphics, tcltk, R2HTML

Suggests tkrplot

SystemRequirements BWidget

Description

Functions, S4 classes/methods and a graphical user interface (GUI) to design surveys to substantiate freedom from disease using a modified hypergeometric function (see Cameron and Baldock, 1997, <doi:10.1016/s0167-5877(97)00081-0>). Herd sensitivities are computed according to sampling strategies ``individual sampling'' or ``limited sampling'' (see M. Ziller, T. Selhorst, J. Teuffert, M. Kramer and H. Schlueter, 2002, <doi:10.1016/S0167-5877(01)00245-8>). Methods to compute the a-posteriori alpha-error are implemented. Risk-based targeted sampling is supported.

License GPL (>= 2)

URL http://ffd.r-forge.r-project.org, https://www.ages.at/

NeedsCompilation no

Repository CRAN

Date/Publication 2022-11-08 10:10:06 UTC

Contents

`computeAlpha` *FUNCTION to compute the herd-based alpha-errors (= 1 - herd sensitivity).*

Description

For a vector of herd sizes the herd-based alpha-errors (= 1-herd sensitivity) are computed for either limited or individual sampling; see Ziller et al.

Usage

```
computeAlpha(nAnimalVec, method, sampleSizeLtd, herdSensitivity,
    intraHerdPrevalence, diagSensitivity, diagSpecificity = 1)
```

Arguments

`nAnimalVec`	Integer vector. Stock sizes of the herds.
`method`	Character string. `"individual"` for individual sampling or `"limited"` for limited sampling.
`sampleSizeLtd`	Integer. Required only if `method == "limited"`. Sample size for limited sampling, i.e., for each herd `sampleSizeLtd` animals are tested, or of the herd contains less than `sampleSizeLtd` animals the entire herd is tested.
`herdSensitivity`	Numeric between 0 and 1. Required only if `method == "individual"`. Desired (minimal) hed sensitivity for individual sampling. The number of animals to test per herd is determined according to that value.
`intraHerdPrevalence`	Numeric between 0 and 1. Intra-herd prevalence. The number of diseased animals per herd is computed as `max(1,round(intraHerdPrevalence*nAnimalVec))` (it is assumed that at least one animal is diseased).

`diagSensitivity`
: Numeric between 0 and 1. Sensitivity (= probability of a testpositive result, given the tested individual is diseased) of the diagnostic test.

`diagSpecificity`
: Numeric between 0 and 1. Specificity (= probability of a testnegative result, given the tested individual is not diseased) of the diagnostic test. The default value is 1, i.e., perfect specificity, and is recommended.

Value

Returns a vector containing the herd-based alpha-errors, where each entry in the vector corresponds to an entry in the input argument `nAnimalVec`.

Author(s)

Ian Kopacka <ian.kopacka@ages.at>

References

M. Ziller, T. Selhorst, J. Teuffert, M. Kramer and H. Schlueter, "Analysis of sampling strategies to substantiate freedom from disease in large areas", Prev. Vet. Med. 52 (2002), pp. 333-343.

See Also

Is used in the method `sample` for classes `IndSampling` and `LtdSampling`.

Examples

```
data(sheepData)
## Compute the herd sensitivities usinh limited sampling:
alphaVec <- computeAlpha(nAnimalVec = sheepData$nSheep,
    method = "limited", sampleSizeLtd = 7,
    intraHerdPrevalence = 0.2, diagSensitivity = 0.9)
```

---

computeAlphaLimitedSampling

FUNCTION to compute the average alpha-error (= error of first kind) for limited sampling.

Description

For sampling strategy "limited sampling" (see Ziller et al., 2002) the function computes the herd-level alpha-errors (= 1-herd sensitivity) for each stock size, as well as the average herd-level alpha-error.

Usage

```
computeAlphaLimitedSampling(stockSizeVector, sampleSizeLtd,
    intraHerdPrevalence, diagSensitivity,
    diagSpecificity = 1, groupVec = NULL)
```

Arguments

`stockSizeVector`	Integer vector. Stock sizes of the herds.
`sampleSizeLtd`	Integer. Sample size for limited sampling, i.e., for each herd `sampleSizeLtd` animals are tested, or of the herd contains less than `sampleSizeLtd` animals the entire herd is tested.
`intraHerdPrevalence`	Numeric between 0 and 1. Intra-herd prevalence. The number of diseased animals per herd is computed as `max(1,round(intraHerdPrevalence*stockSizeVector))` (it is assumed that at least one animal is diseased).
`diagSensitivity`	Numeric between 0 and 1. Sensitivity (= probability of a testpositive result, given the tested individual is diseased) of the diagnostic test.
`diagSpecificity`	Numeric between 0 and 1. Specificity (= probability of a testnegative result, given the tested individual is not diseased) of the diagnostic test.
`groupVec`	Character vector. Optional parameter. If specified it must have the same length as `stockSizeVector`. Defines the gouping of the data. Mean Alpha is then returned for each group.

Value

List of 3 elements:

`alphaDataFrame`	Data frame. Variables `size` = vector of the unique herd sizes in the population, `alpha` = vector of herd-level alpha errors attained by limited sampling scheme for the different herd sizes.
`meanAlpha`	Numeric between 0 and 1. Mean alpha-error attained by strategy "limited sampling" for given sample size and herd size distribution.
`meanAlphaRiskGroups`	If `groupVec` is specified, the mean alpha error is returned for each group, concatenated into a vector. Otherwise this list element is NULL

Author(s)

Ian Kopacka <ian.kopacka@ages.at>

References

M. Ziller, T. Selhorst, J. Teuffert, M. Kramer and H. Schlueter, "Analysis of sampling strategies to substantiate freedom from disease in large areas", Prev. Vet. Med. 52 (2002), pp. 333-343.

See Also

computePValue

Examples

```
data(sheepData)
alphaList <- computeAlphaLimitedSampling(stockSizeVector =
    sheepData$nSheep, sampleSizeLtd = 7,
    intraHerdPrevalence = 0.2, diagSensitivity = 0.9,
    diagSpecificity = 1)
```

---

computeAposterioriError

FUNCTION to compute the a-posteriori error.

Description

For a sampling scheme designed to substantiate freedom from disease the function computes the a-posteriori alpha-error, i.e., the actual alpha-error based on the drawn sample.

Usage

```
computeAposterioriError(alphaErrorVector, nPopulation,
    nDiseased, method = "default")
```

Arguments

alphaErrorVector	Numeric vector. Alpha-error (between 0 and 1) of each herd in the sample.
nPopulation	Integer. Population size, i.e., total numer of herds in the population.
nDiseased	Integer. Number of diseased herds in the population according to the design prevalence.
method	Character string. "exact" for exact error, "approx" for approximation (recommended for nDiseased > 7).

Details

The exact evaluation of the alpha-error is computationally complex, due to combinatirical issues. In order to increase effectivity parts of the code were implemented in C. Still, for nDiseased > 7 the computation may take very long and it is generally not recommended to use the exact calculation. Rather the approximation should be used for nDiseased > 7.

Value

The return value is the a-posteriori alpha-error based on the sample at hand (numeric scalar).

Author(s)

Ian Kopacka <ian.kopacka@ages.at>

Examples

```
## Freedom from disease using limited sampling with sampleSizeLtd = 7.
## Data: sheep holdings in state "Steiermark".
#######################################################################
data(sheepData)
popVec <- sheepData$nSheep[sheepData$state == 6]
N1 <- length(popVec)
sampleSizeLtd <- 7
intraHerdPrev <- 0.13
designPrev <- 0.002
nDiseased <- round(designPrev*N1)
## Draw the sample:
n1 <- 1560
samplePop <- sample(x = popVec, size = n1, replace = FALSE, prob = NULL)
## Compute alpha-errors for the sample:
alphaList <- computeAlphaLimitedSampling(stockSizeVector = samplePop,
        sampleSizeLtd = sampleSizeLtd,
        intraHerdPrevalence = intraHerdPrev,
        diagSensitivity = 0.9, diagSpecificity = 1)
alphaDataFrame <- merge(x = data.frame(size = samplePop),
    y = alphaList$alphaDataFrame, by = "size", all.x = TRUE, all.y = FALSE)
## Compute the a-posteriori alpha-error:
alphaAPostApprox <- computeAposterioriError(alphaErrorVector =
    alphaDataFrame$alpha, nPopulation = N1, nDiseased = nDiseased,
    method = "approx")
alphaAPostExact <- computeAposterioriError(alphaErrorVector =
    alphaDataFrame$alpha, nPopulation = N1, nDiseased = nDiseased,
    method = "exact")
```

---

computeAposterioriErrorRiskGroups

FUNCTION to compute the a-posteriori error considering a population stratified into risk geroups.

Description

For a sampling scheme designed to substantiate freedom from disease the function computes the a-posteriori alpha-error, i.e., the actual alpha-error based on the drawn sample, when the population is stratified into risk groups for which the relative infection risk is known.

Usage

```
computeAposterioriErrorRiskGroups(alphaErrorVector,
    groupVec, groupLevels, nPopulationVec, nRelRiskVec,
    nDiseased, method = "default")
```

Arguments

alphaErrorVector	Numeric vector. Alpha-error (between 0 and 1) of each herd in the sample.
groupVec	Character vector. Risk group to which each of the herds in the sample belongs. Must have the same length (and order) as alphaErrorVector.
groupLevels	Character vector. (Unique) levels/names of the risk group. Defines the order of the values in nPopulationVec and nRelRiskVec.
nPopulationVec	Integer vector. Population sizes of the risk groups. Must have the same length (and order) as groupLevels.
nRelRiskVec	Numeric vector. (Relative) infection risks of the risk groups. Must have the same length (and order) as groupLevels.
nDiseased	Integer. Number of diseased herds in the population according to the design prevalence.
method	Character string. "exact" for exact error, "approx" for approximation (recommended for nDiseased > 7).

Details

The exact evaluation of the alpha-error is computationally complex, due to combinatirical issues. In order to increase effectivity parts of the code were implemented in C. Still, for nDiseased > 3 the computation may take very long and it is generally not recommended to use the exact calculation. Rather the approximation should be used for nDiseased > 3.

Value

The return value is the a-posteriori alpha-error based on the sample at hand (numeric scalar).

Author(s)

Ian Kopacka <ian.kopacka@ages.at>

Examples

```
data(sheepData)
sheepData$size <- ifelse(sheepData$nSheep < 30, "small", "large")
riskValueData <- data.frame(riskGroup = c("small", "large"),
    riskValues = c(1,2))
mySurvey <- surveyData(nAnimalVec = sheepData$nSheep,
riskGroupVec = sheepData$size,
riskValueData = riskValueData,
populationData = sheepData, designPrevalence = 0.002,
alpha = 0.05, intraHerdPrevalence = 0.13,
diagSensitivity = 0.9, costHerd = 30, costAnimal = 7.1)
## Limited sampling with risk groups:
myLtdSamplingRG <- ltdSampling(survey.Data = mySurvey, sampleSizeLtd = 20,
nSampleFixVec = NULL, probVec = c(1,2))
## Draw sample manually:
set.seed(20110708)
```

```
x <- myLtdSamplingRG
indexSampleRG <- sapply(seq(along = x@surveyData@riskValueData$riskGroup),
function(ii){
riskGroup <- as.character(x@surveyData@riskValueData$riskGroup[ii])
nSampleRG <- x@nHerdsPerRiskGroup[riskGroup]
indexRG <- which(x@surveyData@riskGroupVec == riskGroup)
indexOut <- sample(x = indexRG, size = nSampleRG, replace = FALSE)
})
indexSample <- sort(Reduce(function(x,y) c(x,y), indexSampleRG))
## Compute the a-posteriori alpha error:
alphaErrorVector <- computeAlpha(nAnimalVec = x@surveyData@nAnimalVec[indexSample],
     method = "limited", sampleSizeLtd = x@sampleSizeLtd,
     intraHerdPrevalence = x@surveyData@intraHerdPrevalence,
     diagSensitivity = x@surveyData@diagSensitivity, diagSpecificity = 1)
## Determine the number of herds in each risk group:
riskValueDf <- mySurvey@riskValueData[,1:2]
names(riskValueDf) <- c("riskGroup", "riskValues")
riskValueDf$riskGroup <- as.character(riskValueDf$riskGroup)
riskValueDf$id <- seq(along = riskValueDf[,1])
riskGroupTab <- table(mySurvey@riskGroupVec)
riskGroupDf <- data.frame(riskGroup = as.character(names(riskGroupTab)),
nPopulation = as.vector(riskGroupTab))
riskValueDf <- merge(x = riskValueDf, y = riskGroupDf, by = "riskGroup",
sort = FALSE)
riskValueDf <- riskValueDf[order(riskValueDf$id, decreasing = FALSE),]
aPostAlphaRG <- computeAposterioriErrorRiskGroups(alphaErrorVector = alphaErrorVector,
     groupVec = x@surveyData@riskGroupVec[indexSample],
 groupLevels = riskValueDf$riskGroup,
 nPopulationVec = riskValueDf$nPopulation,
 nRelRiskVec = riskValueDf$riskValues,
   nDiseased = max(round(length(x@surveyData@nAnimalVec)*x@surveyData@designPrevalence),1),
     method = "approx")
```

`computeOptimalSampleSize` *FUNCTION to compute the optimal sample size.*

Description

Computes the optimal sample size for a survey to substantiate freedom from disease. The optimal sample size is the smallest sample size that produces an alpha-error less than or equal to a prediscribed value `alpha`. The population is considered as diseased if at least one individual has a positive test result. The sample size is computed using a bisection method. The function either computes the sample size for a fixed population (`lookupTable == FALSE`) or a lookup table with sampling sizes depending on the population size for individual sampling (`lookupTable == TRUE`); see Ziller et al., 2002.

Usage

```
computeOptimalSampleSize(nPopulation, prevalence, alpha = 0.05,
    sensitivity = 1, specificity = 1, lookupTable = FALSE)
```

Arguments

nPopulation	Integer. Population size if lookupTable == FALSE or the largest considered herd size for the lookup table if lookupTable == TRUE .
prevalence	Numeric between 0 and 1. Design prvalence. The number of diseased is then computed as max(1,nPopulation*prevalence).
alpha	Numeric between 0 and 1. Alpha-Error (=error of the first kind, significance level) of the underlying significance test. Default value = 0.05.
sensitivity	Numeric between 0 and 1. Sensitivity of the diagnostic (for one-stage sampling) or herd test (for two stage sampling). Default value = 1.
specificity	Numeric between 0 and 1. Specificity of the diagnostic (for one-stage sampling) or herd test (for two stage sampling). Default value = 1.
lookupTable	Logical. TRUE if a lookup table of sample sizes for individual sampling (see, Ziller et al., 2002) should be produced. FALSE if the sample size is desired for a fixed population size (default).

Value

The return value is either an integer, the minimal sample size that produces the desired alpha-error if lookupTable == FALSE or a matrix with columns N_lower, N_upper, sampleSize containing the sample sizes for the different herd sizes N.

E.g., N_lower = 17, N_upper = 21, sampleSize = 11 means that for holdings with 17-21 animals 11 animals need to be tested in order to achieve the desired accuracy (=herd sensitivity).

Author(s)

Ian Kopacka <ian.kopacka@ages.at>

References

A.R. Cameron and F.C. Baldock, "A new probablility formula to substantiate freedom from disease", Prev. Vet. Med. 34 (1998), pp. 1-17.

M. Ziller, T. Selhorst, J. Teuffert, M. Kramer and H. Schlueter, "Analysis of sampling strategies to substantiate freedom from disease in large areas", Prev. Vet. Med. 52 (2002), pp. 333-343.

See Also

computePValue

Examples

```
## Compute the number of herds to be tested for limited sampling
## with sampleSizeLtd = 7:
#################################################################
data(sheepData)
## Compute the average herd sensitivity:
alphaList <- computeAlphaLimitedSampling(stockSizeVector =
    sheepData$nSheep, sampleSizeLtd = 7,
```

```
    intraHerdPrevalence = 0.13, diagSensitivity = 0.9,
    diagSpecificity = 1)
sensHerd <- 1 - alphaList$meanAlpha
## Number of herds to be tested:
nHerds <- computeOptimalSampleSize(nPopulation = dim(sheepData)[1],
    prevalence = 0.002, alpha = 0.05, sensitivity = sensHerd,
    specificity = 1)

## Compute the number of animals to be tested for individual
## sampling:
#####################################################################
sampleSizeIndividual <- computeOptimalSampleSize(nPopulation = 300,
    prevalence = 0.2, alpha = 0.05, sensitivity = 0.97,
    specificity = 1, lookupTable = TRUE)
```

`computeOptimalSampleSizeRiskGroups` *FUNCTION to compute the optimal sample size for populations stratified by risk factors.*

Description

Computes the optimal sample size (for each risk group) for a survey to substantiate freedom from disease for a population stratified into risk groups. The optimal sample size is the smallest sample size that produces an alpha-error less than or equal to a prediscribed value for alpha. The population is considered as diseased if at least one individual has a positive test result. The sample size is computed using a bisection method. The sample size can be fixed for a subset of the risk groups via the input parameter 'nSampleFixVec' (vector containing sample sizes for the risk groups with fixed values and NA for the risk groups for which the sample size is to be computed). For those risk groups for which the sample size is to be computed a vector specifying the proportional distribution among the risk groups ('nSamplePropVec') needs to be specified.

Example: We have 3 risk groups. For the 2nd risk group we want 20 farms to be sampled. For the other risk groups we specify that the sample size for risk group 1 should be double the sample size of risk group 3. We then set: nSampleFixVec <- c(NA, 20, NA) nSamplePropVec <- c(2,1)

Usage

```
computeOptimalSampleSizeRiskGroups(nPopulationVec,
    nRelRiskVec, nSampleFixVec = NULL, nSamplePropVec = NULL,
    prevalence, alpha = 0.05, sensitivity = 1,
    specificity = 1)
```

Arguments

`nPopulationVec`	Integer vector. Population sizes of the risk groups.
`nRelRiskVec`	Numeric vector. (Relative) infection risks of the risk groups.

nSampleFixVec	Numeric vector containing NAs (optional argument). For risk groups for which the sample size is fixed specify the sample size. For the risk groups for which the sample size should be computed set NA (order of the risk groups must be the same order as in nPopulationVec and nRelRiskVec) .
nSamplePropVec	Numeric vector. For those risk groups for which the sample size should be computed a proportional distribution of the overall sample size must be specified. The vector must have the same length as the number of NA entries in nSampleFixVec or if nSampleFixVec is not specified, nSamplePropVec must have the same length as nPopulationVec.
prevalence	Numeric between 0 and 1. Design prvalence. The number of diseased is then computed as max(1,nPopulation*prevalence).
alpha	Numeric between 0 and 1. Alpha-Error (=error of the first kind, significance level) of the underlying significance test. Default value = 0.05.
sensitivity	Numeric between 0 and 1. Sensitivity of the diagnostic (for one-stage sampling) or herd test (for two stage sampling). Default value = 1.
specificity	Numeric between 0 and 1. Specificity of the diagnostic (for one-stage sampling) or herd test (for two stage sampling). Default value = 1.

Value

The return value is an integer vector containing the optimal sample size for every risk group specified in the input variables nPopulationVec and nRelRiskVec.

Author(s)

Ian Kopacka <ian.kopacka@ages.at>

References

A.R. Cameron and F.C. Baldock, "A new probablility formula to substantiate freedom from disease", Prev. Vet. Med. 34 (1998), pp. 1-17.

P.A.J.Martin, A.R. Cameron, M. Greiner, "Demonstrating freedom from disease using multiple complex data sources. : A new methodology based on scenario trees", Prev. Vet. Med. 79 (2007), pp. 71 - 97.

See Also

computePValueRiskGroups

Examples

```
nPopulationVec <- c(500,700)
nRelRiskVec <- c(1,3)
prevalence <- 0.01
alpha <- 0.05
herdSensitivity <- 0.7
specificity <- 1
```

```
## Optimal sample size with risk groups:
nRisk <- computeOptimalSampleSizeRiskGroups(nPopulationVec =
    nPopulationVec, nRelRiskVec = nRelRiskVec,
nSamplePropVec = c(1,4), prevalence = prevalence,
alpha = alpha, sensitivity = herdSensitivity,
specificity = specificity)
## Optimal sample size without risk groups:
nNoRisk <- computeOptimalSampleSize(sum(nPopulationVec),
    prevalence, alpha, herdSensitivity, specificity, FALSE)
```

`computePValue` *FUNCTION to compute the probability of finding no testpositives in a sample of a certain size.*

Description

For a population of size `nPopulation` with a given design prevalence the function computes the probability of finding no testpositives in a sample of size `nSample` if an imperfect test is used (given sensitivity and specificity). This probability corresponds to the alpha-error (=error of the first kind) of the overall test with null hypothesis: prevalence = design prevalence. A modified hypergeometric formula is used; see Cameron, Baldock, 1998.

Usage

```
computePValue(nPopulation, nSample, nDiseased,
    sensitivity, specificity = 1)
```

Arguments

`nPopulation`	Integer. Population size.
`nSample`	Integer. Size of sample.
`nDiseased`	Integer. Number of diseased elements in the population according to the design prevalence.
`sensitivity`	Numeric between 0 and 1. Sensitivity (= probability of a testpositive result, given the tested individual is diseased) of the test (e.g., diagnostic test or herd test).
`specificity`	Numeric between 0 and 1. Specificity (= probability of a testnegative result, given the tested individual is not diseased) of the test (e.g., diagnostic test or herd test). The default value is 1.

Value

The return value is a numeric between 0 and 1. It is the probability of finding no testpositives (not diseased!) in the sample.

Author(s)

Ian Kopacka <ian.kopacka@ages.at>

References

A.R. Cameron and F.C. Baldock, "A new probablility formula to substantiate freedom from disease", Prev. Vet. Med. 34 (1998), pp. 1-17.

See Also

computeOptimalSampleSize, computeAlphaLimitedSampling

Examples

```
alphaError <- computePValue(nPopulation = 3000,
    nSample = 1387, nDiseased = 6, sensitivity = 0.85, specificity = 1)
```

---

computePValueRiskGroups

FUNCTION to compute the probability of finding no testpositives in a sample of a certain size for a population stratified into risk groups.

Description

For a population that is stratified into risk groups the function computes the probability of finding no testpositives in a sample of given size using an imperfect diagnostic test. For each of the risk groups the population size `nPopulationVec`, the sample size `nSampleVec` and the relative infection risk `nRelRiskVec` must be specified. The discussed probability corresponds to the alpha-error (=error of the first kind) of the overall test with null hypothesis: prevalence = design prevalence.

Usage

```
computePValueRiskGroups(nPopulationVec, nSampleVec,
    nRelRiskVec, nDiseased, sensitivity,
    specificity = 1)
```

Arguments

`nPopulationVec`	Integer vector. Population sizes of the risk groups.
`nSampleVec`	Integer vector. Sample sizes of the risk groups.
`nRelRiskVec`	Numeric vector. (Relative) infection risks of the risk groups.
`nDiseased`	Integer. Number of diseased elements in the population according to the design prevalence.
`sensitivity`	Numeric between 0 and 1. Sensitivity (= probability of a testpositive result, given the tested individual is diseased) of the test (e.g., diagnostic test or herd test).
`specificity`	Numeric between 0 and 1. Specificity (= probability of a testnegative result, given the tested individual is not diseased) of the test (e.g., diagnostic test or herd test). The default value is 1.

Value

The return value is a numeric between 0 and 1. It is the probability of finding no testpositives (not diseased!) in the sample.

Author(s)

Ian Kopacka <ian.kopacka@ages.at>

References

A.R. Cameron and F.C. Baldock, "A new probablility formula to substantiate freedom from disease", Prev. Vet. Med. 34 (1998), pp. 1-17.

P.A.J.Martin, A.R. Cameron, M. Greiner, "Demonstrating freedom from disease using multiple complex data sources. : A new methodology based on scenario trees", Prev. Vet. Med. 79 (2007), pp. 71 - 97.

See Also

Calls computePValue

Examples

```
nPopulationVec <- c(500,700)
nSampleVec <- c(300,200)
nRelRiskVec <- c(1,1)
nDiseased <- round(sum(nPopulationVec)*0.01)
sensitivity <- 0.9
specificity <- 1
alphaError <- computePValue(sum(nPopulationVec), sum(nSampleVec),
nDiseased, sensitivity, specificity)
```

FFD_GUI *Freedom From Disease sampling plan calculator*

Description

This function opens the Graphical User Interface for the FFD sampling plan calculator.

The GUI offers the functionality to

- compute first and second-stage sample sizes for two-stage sampling schemes to substantiate freedom from disease,
- analyze two-stage sampling schemes with respect to overall cost and sensitivity,
- determine cost-optimal strategies and
- draw samples from a population.

The GUI is structured into three tabs, *Data Input, Parameters, Calculations*:

Data Input: The farm data is specified. A list of farms is required with one row per farm and one column containing the herd size, i.e. the number of animals on the farm. The data must be provided in the CSV file format, however currently only the central European format with comma = ",", seperator = ";" is supported. The location of the csv-file is specified in the field *Data file*, the name of the column containing the herd size is set in the field *Herd sizes column* via a dropdown menu.

Parameters: Next the survey parameters: design prevalence, Type I error level, intraherd prevalence, and test sensitivity must be set and a sampling strategy (limited sampling or individual sampling, see, e.g. Ziller et al.) must be chosen. Furthermore the cost of each tested herd (excluding the cost per tested animal) and the cost of each tested animal can be specified for cost analyses.

Calculations: Once a sampling strategy is chosen the tab is activated, where cost optimal sampling plans can be determined (*Sample size Diagnostics*), cost and sensitivity of a fixed sampling schemes can be computed (*Compute sample size -> Calculate*) and an actual sample can be drawn from the farm file (*Compute sample size -> Sample*).

Usage

```
FFD_GUI()
```

Details

The *FFD GUI* provides the following menu options:

File

- **Load** loads population data and survey parameter settings from a ffd-RData file.
- **Save** saves population data and survey parameter settings to a ffd-RData file.
- **Load examples** loads built-in examples.
- **Reset** clears all entered data.

Help

- **R Documentation** opens the help file for the FFD GUI (*cf* `?FFD_GUI`).
- **FFD Manual (PDF)** opens the pdf manual
- **About FFD_GUI** displays version and developers info.

Author(s)

Ian Kopacka

References

- Cameron A.R. and Baldock F.C. (1998). *A new probability formula for surveys to substantiate freedom from disease*, Prev. Vet. Med. 34 (1), pp. 1-17

- Cameron A.R. and Baldock F.C. (1998). *Two-stage sampling in surveys to substantiate freedom from disease*, Prev. Vet. Med. 34 (1), pp. 19-30
- Ziller M., Selhorst T. Teuffert J., Kramer M. and Schlueter H. (2002). *Analysis of sampling strategies to substantiate freedom from disease in large areas*, Prev. Vet. Med. 52 (3-4), pp. 333-343
- http://ffd.r-forge.r-project.org/

`indSampling` *Constructor for class 'IndSampling'.*

Description

Creates an object of the class 'IndSampling'. For given survey parameters (passed to the function as an object of the class `SurveyData`) `indSampling()` computes the the number of herds to test, the expected total number of animals to test, the expected total cost of a survey using limited sampling with a given herd sensitivity `herdSensitivity`, as well as a lookup table for the number of animals to test per herd, depending on the herd size.

Usage

```
indSampling(survey.Data, herdSensitivity, nSampleFixVec = NULL,
probVec = NULL)
```

Arguments

`survey.Data`	Object of class `SurveyData`. Created by using the function `surveyData`.
`herdSensitivity`	Numeric between 0 and 1. Desired herd sensitivity.
`nSampleFixVec`	Numeric vector containing some NAs (optional argument). For risk groups for which the sample size is fixed specify the sample size. For the risk groups for which the sample size should be computed set NA (order of the risk groups must be the same order as in `survey.Data@riskValueData`).
`probVec`	Numeric vector. For those risk groups for which the sample size should be computed sample probabilities must be specified. The vector must have the same length as the number of NA entries in `nSampleFixVec` or if `nSampleFixVec` is not specified, probVec must have the same length as the number of rows in `survey.Data@riskValueData`.

Value

The function returns an object of the class `IndSampling`.

Author(s)

Ian Kopacka <ian.kopacka@ages.at>

References

A.R. Cameron and F.C. Baldock, "A new probablility formula to substantiate freedom from disease", Prev. Vet. Med. 34 (1998), pp. 1-17.

A.R. Cameron and F.C. Baldock, "Two-stage sampling surveys to substantiate freedom from disease", Prev. Vet. Med. 34 (1998), pp. 19-30.

M. Ziller, T. Selhorst, J. Teuffert, M. Kramer and H. Schlueter, "Analysis of sampling strategies to substantiate freedom from disease in large areas", Prev. Vet. Med. 52 (2002), pp. 333-343.

See Also

See IndSampling and SurveyData for additional details.

Examples

```
data(sheepData)
sheepData$size <- ifelse(sheepData$nSheep < 30, "small", "large")
riskValueData <- data.frame(riskGroup = c("small", "large"),
    riskValues = c(1,2))
mySurvey <- surveyData(nAnimalVec = sheepData$nSheep,
riskGroupVec = sheepData$size,
riskValueData = riskValueData,
populationData = sheepData, designPrevalence = 0.002,
alpha = 0.05, intraHerdPrevalence = 0.13,
diagSensitivity = 0.9, costHerd = 30, costAnimal = 7.1)
## Individual sampling without risk groups:
myIndSampling <- indSampling(survey.Data = mySurvey,
    herdSensitivity = 0.7)
## Individual sampling with risk groups:
myIndSamplingRG <- indSampling(survey.Data = mySurvey,
    herdSensitivity = 0.7, nSampleFixVec = NULL, probVec = c(1,4))
```

IndSampling-class *Class "IndSampling"*

Description

Contains the parameters and the data necessary for a survey to substantiate freedom from disease using "individual sampling". Additionally to the survey parameters: design prevalence (=prevalence of the disease under the null hypothesis), overall significance level (=1-confidence), intraherd prevalence, sensitivity of the diagnostic test, cost per tested animal and cost per tested herd, the object contains the herd sensitivity the number of herds to be tested, the mean overall number of animals to be tested, the expected costs, as well as a lookup table for the number of animals to test depending on the herd size.

Objects from the Class

Objects can be created by calls of the form `new("IndSampling", ...)`.

Slots

`surveyData`: Object of class `"SurveyData"`. Contains all the necessary data and specifications for the survey.

`herdSensitivity`: Object of class `"numeric"` with values between 0 and 1. Desired herd sensitivity.

`nHerds`: Object of class `"numeric"`. Number of herds to be tested according to the herd sensitivity `herdSensitivity`.

`nHerdsPerRiskGroup`: Object of class `"numeric"`. Number of herds to be tested per risk group (if population is stratified by risk groups).

`nSampleFixVec`: Object of class `"numeric"`. Numeric vector containing some NAs (optional argument). For risk groups for which the sample size is fixed it specifies the sample size. For the risk groups for which the sample size was computed it was set to NA (order of the risk groups is the same as in `survey.Data@riskValueData`).

`probVec`: Object of class `"numeric"`. Contains the sample probabilities for those risk groups for which the sample size was computed (=NA entries in `nSampleFixVec`).

`nAnimalsMean`: Object of class `"numeric"`. Expected total number of animals to be tested in the survey.

`expectedCost`: Object of class `"numeric"`. Expected costs of the survey.

`lookupTable`: Object of class `"matrix"` with columns `N_lower`, `N_upper` and `sampleSize` containing the number of animals to test for each herd size.

Methods

HTML `signature(x = "IndSampling")`: Creates an html file containing the summary data and the diagnostic plots. Title, file name, output directory, css-file, etc. can additionally be specified using the parameters, `filename`, `outdir`, `CSSFile`, `Title`, as well as all the other parameters of the R2HTML-function `HTMLInitFile`.

sample `signature(x = "IndSampling", size = c("fixed", "dynamic"))`: Sample herds using individual sampling. Additionally to the argument `x` of type `IndSampling` the method takes an argument `size`, which is a character string. For `size == "fixed"` the fixed number of herds given in `x@nHerds` is sampled using simple random sampling. For `size == "dynamic"` dynamic sampling is used, i.e., based on real-time computation of the a-posteriori alpha- error the sample is updated until the a-posteriori alpha-error falls below the predefined significance level x@alpha. The return value is a list with two items: `indexSample` is a vector of indices of the sample corresponding to `x@surveyData@nAnimalVec` and `aPostAlpha` containing the a-posteriori alpha-error of the sample.

show `signature(object = "IndSampling")`: Display structure of the class and content of the slots.

summary `signature(object = "IndSampling")`: Display structure of the class and a summary of the content of the slots.

Note

No notes yet.

Author(s)

Ian Kopacka <ian.kopacka@ages.at>

See Also

The slot surveyData contains an object of the class SurveyData which is created using surveyData. Objects of the class IndSampling are create using the constructor indSampling.

Examples

```
## Show the structure of the class:
showClass("IndSampling")
## Create an object:
data(sheepData)
mySurvey <- surveyData(nAnimalVec = sheepData$nSheep,
    populationData = sheepData, designPrevalence = 0.002,
    alpha = 0.05, intraHerdPrevalence = 0.13,
    diagSensitivity = 0.9, costHerd = 30, costAnimal = 7.1)
myIndSampling <- indSampling(survey.Data = mySurvey, herdSensitivity = 0.7)
## Display results:
summary(myIndSampling)
## Write results to an html-file:
## Not run:
target <- HTMLInitFile(getwd(), filename = "IndSampling")
HTML(myIndSampling)
HTMLEndFile()
## End(Not run)
```

---

indSamplingSummary *Constructor for class 'IndSamplingSummary'.*

Description

Creates an object of the class 'IndSamplingSummary'. For given survey parameters (passed to the function as an object of the class SurveyData) indSamplingSummary() computes the number of herds to test, the expected total number of animals to test and the expected total cost of a survey using individual sampling with a given sequence of herd sensitivities. This sequence ranges from 0.1 to the sensitivity of the diagnostic test specified in survey.Data. The step size for the herd sensitivities can be specified by the user via the argument stepSize. If no step size is specified a step size of 0.02 is used.

Usage

```
indSamplingSummary(survey.Data, stepSize = 0.02,
    nSampleFixVec = NULL, probVec = NULL)
```

Arguments

survey.Data	Object of class SurveyData. Created by using the function surveyData.
stepSize	Numeric. A series of parameters is computed for a sequence of herd sensitivities. The argument stepSize specifies the step size used in the discretization of the herd sensitivities (default = 0.02).
nSampleFixVec	Numeric vector containing some NAs (optional argument). For risk groups for which the sample size is fixed specify the sample size. For the risk groups for which the sample size should be computed set NA (order of the risk groups must be the same order as in survey.Data@riskValueData).
probVec	Numeric vector. For those risk groups for which the sample size should be computed sample probabilities must be specified. The vector must have the same length as the number of NA entries in nSampleFixVec or if nSampleFixVec is not specified, probVec must have the same length as the number of rows in survey.Data@riskValueData.

Value

The function returns an object of the class IndSamplingSummary.

Author(s)

Ian Kopacka <ian.kopacka@ages.at>

References

A.R. Cameron and F.C. Baldock, "A new probablility formula to substantiate freedom from disease", Prev. Vet. Med. 34 (1998), pp. 1-17.

A.R. Cameron and F.C. Baldock, "Two-stage sampling surveys to substantiate freedom from disease", Prev. Vet. Med. 34 (1998), pp. 19-30.

M. Ziller, T. Selhorst, J. Teuffert, M. Kramer and H. Schlueter, "Analysis of sampling strategies to substantiate freedom from disease in large areas", Prev. Vet. Med. 52 (2002), pp. 333-343.

See Also

See IndSamplingSummary and SurveyData for additional details.

Examples

```
data(sheepData)
sheepData$size <- ifelse(sheepData$nSheep < 30, "small", "large")
riskValueData <- data.frame(riskGroup = c("small", "large"),
    riskValues = c(1,2))
mySurvey <- surveyData(nAnimalVec = sheepData$nSheep,
riskGroupVec = sheepData$size,
riskValueData = riskValueData,
populationData = sheepData, designPrevalence = 0.002,
alpha = 0.05, intraHerdPrevalence = 0.13,
diagSensitivity = 0.9, costHerd = 30, costAnimal = 7.1)
```

```
## Individual sampling without risk groups:
myIndSamplingSummary <- indSamplingSummary(survey.Data = mySurvey,
    stepSize = 0.06)
## Individual sampling with risk groups:
myIndSamplingSummaryRG <- indSamplingSummary(survey.Data = mySurvey,
    stepSize = 0.06, nSampleFixVec = NULL, probVec = c(1,4))
```

IndSamplingSummary-class *Class "IndSamplingSummary"*

Description

Contains the parameters and the data necessery for a survey to substantiate freedom from disease using "individual sampling". Additionally to the survey parameters: design prevalence (=prevalence of the disease under the null hypothesis), overall significance level (=1-confidence), intra-herd prevalence, sensitivity of the diagnostic test, cost per tested animal and cost per tested herd, the object contains the the number of herds to be tested, the mean overall number of animals to be tested and the expected costs for a range of possible herd sensitivities.

Objects from the Class

Objects can be created by calls of the form `new("IndSamplingSummary", ...)`.

Slots

`surveyData`**:** Object of class `"SurveyData"`. Containing the necessary survey parameters.

`herdSensVec`**:** Object of class `"numeric"` with values between 0 and 1. Mean herd sensitivity in the population (vector).

`nHerdsVec`**:** Object of class `"numeric"`. Number of herds to be tested according to the herd sensitivity `herdSensVec` (vector).

`nHerdsPerRiskGroupMx`**:** Object of class `"matrix"`. Number of herds to be tested per risk group [columns] and sample limit [rows] (if population is stratified by risk groups).

`nSampleFixVec`**:** Object of class `"numeric"`. Numeric vector containing some NAs (optional argument). For risk groups for which the sample size is fixed it specifies the sample size. For the risk groups for which the sample size was computed it was set to NA (order of the risk groups is the same as in `survey.Data@riskValueData`).

`probVec`**:** Object of class `"numeric"`. Contains the sample probabilities for those risk groups for which the sample size was computed (=NA entries in `nSampleFixVec`).

`nAnimalsMeanVec`**:** Object of class `"numeric"`. Expected total number of animals to be tested in the survey (vector).

`expectedCostVec`**:** Object of class `"numeric"`. Expected costs of the survey (vector).

Methods

HTML `signature(x = "IndSamplingSummary")`: Creates an html file containing the summary data and the diagnostic plots. Title, file name, output directory, css-file, etc. can additionally be specified using the parameters, `filename`, `outdir`, `CSSFile`, `Title`, as well as all the other parameters of the R2HTML-function `HTMLInitFile`.

plot `signature(x = "IndSamplingSummary")`: Create plots of 1) the mean number of animals to be tested per herd, 2) the number of herds to be tested, 3) the expected total number of animals to be tested, 4) the expected total cost of the survey plotted against the vector of herd sensitivities.

show `signature(object = "IndSamplingSummary")`: Display structure of the class and content of the slots.

summary `signature(object = "IndSamplingSummary")`: Display structure of the class and a summary of the content of the slots.

Note

No notes yet.

Author(s)

Ian Kopacka <ian.kopacka@ages.at>

See Also

The slot `surveyData` contains an object of the class `SurveyData`.

Examples

```
## Show the structure of the class:
showClass("IndSamplingSummary")
## Create an object:
data(sheepData)
mySurvey <- surveyData(nAnimalVec = sheepData$nSheep,
    populationData = sheepData, designPrevalence = 0.002,
    alpha = 0.05, intraHerdPrevalence = 0.13,
    diagSensitivity = 0.9, costHerd = 15, costAnimal = 20)
myIndSamplingSummary <- indSamplingSummary(survey.Data = mySurvey,
    stepSize = 0.06)
## Display results:
summary(myIndSamplingSummary)
plot(myIndSamplingSummary)
## Write results to an html-file:
## Not run:
target <- HTMLInitFile(getwd(), filename = "ItdSampling")
HTML(myIndSamplingSummary)
HTMLEndFile()
## End(Not run)
```

`lls` *List Objects*

Description

Function works like ls() but it returns the names of the objects in the workspace, together with class, dimension and size in the form of a data frame.

Usage

```
lls(name, pos = -1, envir = as.environment(pos), all.names = FALSE,
    pattern, classFilter, sort = "size")
```

Arguments

`name`	which environment to use in listing the available objects. Defaults to the current environment. Although called name for back compatibility, in fact this argument can specify the environment in any form; see the details section.
`pos`	an alternative argument to name for specifying the environment as a position in the search list. Mostly there for back compatibility.
`envir`	an alternative argument to name for specifying the environment. Mostly there for back compatibility.
`all.names`	a logical value. If TRUE, all object names are returned. If FALSE, names which begin with a . are omitted.
`pattern`	an optional regular expression. Only names matching pattern are returned. glob2rx can be used to convert wildcard patterns to regular expressions.
`classFilter`	optional character string. Only objects of the specified class are returned.
`sort`	optional character string either "size" (default) or "name". Objects are either sorted by size or alphabetically.

Details

The name argument can specify the environment from which object names are taken in one of several forms: as an integer (the position in the search list); as the character string name of an element in the search list; or as an explicit environment (including using sys.frame to access the currently active function calls). By default, the environment of the call to lls or objects is used. The pos and envir arguments are an alternative way to specify an environment, but are primarily there for back compatibility.

Value

Returns a data frame.

Author(s)

Original author unknown, modified by Ian Kopacka

Examples

```
lls()
```

`ltdSampling` *Constructor for class 'LtdSampling'.*

Description

Creates an object of the class 'LtdSampling'. For given survey parameters (passed to the function as an object of the class `SurveyData`) `ltdSampling()` computes the mean herd sensitivity, the number of herds to test, the expected total number of animals to test and the expected total cost of a survey using limited sampling with a given animal-level sample size `sampleSizeLtd`. If values for `nSampleFixVec` and/or `probVec` are specified, sampling if performed with stratification of the population by risk groups.

Usage

```
ltdSampling(survey.Data, sampleSizeLtd, nSampleFixVec = NULL,
probVec = NULL)
```

Arguments

`survey.Data`	Object of class `SurveyData`. Created by using the function `surveyData`.
`sampleSizeLtd`	Positive integer. Pre-fixed number of animals to be tested per holding, irrespective of the herd size (if the herd contains fewer animals then the entire herd needs to be tested).
`nSampleFixVec`	Numeric vector containing some NAs (optional argument). For risk groups for which the sample size is fixed specify the sample size. For the risk groups for which the sample size should be computed set NA (order of the risk groups must be the same order as in `survey.Data@riskValueData`).
`probVec`	Numeric vector. For those risk groups for which the sample size should be computed sample probabilities must be specified. The vector must have the same length as the number of NA entries in `nSampleFixVec` or if `nSampleFixVec` is not specified, probVec must have the same length as the number of rows in `survey.Data@riskValueData`.

Value

The function returns an object of the class `LtdSampling`.

Author(s)

Ian Kopacka <ian.kopacka@ages.at>

References

A.R. Cameron and F.C. Baldock, "A new probablility formula to substantiate freedom from disease", Prev. Vet. Med. 34 (1998), pp. 1-17.

A.R. Cameron and F.C. Baldock, "Two-stage sampling surveys to substantiate freedom from disease", Prev. Vet. Med. 34 (1998), pp. 19-30.

M. Ziller, T. Selhorst, J. Teuffert, M. Kramer and H. Schlueter, "Analysis of sampling strategies to substantiate freedom from disease in large areas", Prev. Vet. Med. 52 (2002), pp. 333-343.

See Also

See `LtdSampling` and `SurveyData` for additional details.

Examples

```
data(sheepData)
sheepData$size <- ifelse(sheepData$nSheep < 30, "small", "large")
riskValueData <- data.frame(riskGroup = c("small", "large"),
    riskValues = c(1,2))
mySurvey <- surveyData(nAnimalVec = sheepData$nSheep,
riskGroupVec = sheepData$size,
riskValueData = riskValueData,
populationData = sheepData, designPrevalence = 0.002,
alpha = 0.05, intraHerdPrevalence = 0.13,
diagSensitivity = 0.9, costHerd = 30, costAnimal = 7.1)
## Limited sampling without risk groups:
myLtdSampling <- ltdSampling(survey.Data = mySurvey, sampleSizeLtd = 7)
## Limited sampling with risk groups:
myLtdSamplingRG <- ltdSampling(survey.Data = mySurvey, sampleSizeLtd = 7,
nSampleFixVec = NULL, probVec = c(1,4))
```

LtdSampling-class *Class "LtdSampling"*

Description

Contains the parameters and the data necessary for a survey to substantiate freedom from disease using "limited sampling". Additionally to the survey parameters: design prevalence (=prevalence of the disease under the null hypothesis), overall significance level (=1-confidence), intra-herd prevalence, sensitivity of the diagnostic test, cost per tested animal and cost per tested herd, the object contains the mean herd sensitivity, the number of herds to be tested, the mean overall number of animals to be tested and the expected costs.

Objects from the Class

Objects can be created by calls of the form `new("LtdSampling", ...)`.

Slots

`surveyData`: Object of class `"SurveyData"`. Contains all the necessary data and specifications for the survey.

`sampleSizeLtd`: Object of class `"numeric"`. Pre-fixed number of animals to be tested per holding, irrespective of the herd size. If a herd contains fewer animals the entire herd is tested.

`meanHerdSensitivity`: Object of class `"numeric"` with values between 0 and 1. Mean herd sensitivity in the population.

`meanHerdSensPerRG`: Object of class `"numeric"` with values between 0 and 1. Mean herd sensitivity of each risk group (if population is stratified by risk groups).

`nHerds`: Object of class `"numeric"`. Number of herds to be tested according to the herd sensitivity `meanHerdSensitivity`.

`nHerdsPerRiskGroup`: Object of class `"numeric"`. Number of herds to be tested per risk group (if population is stratified by risk groups).

`nSampleFixVec`: Object of class `"numeric"`. Numeric vector containing some NAs (optional argument). For risk groups for which the sample size is fixed it specifies the sample size. For the risk groups for which the sample size was computed it was set to NA (order of the risk groups is the same as in `survey.Data@riskValueData`).

`probVec`: Object of class `"numeric"`. Contains the sample probabilities for those risk groups for which the sample size was computed (=NA entries in `nSampleFixVec`).

`nAnimalsMean`: Object of class `"numeric"`. Expected total number of animals to be tested in the survey.

`expectedCost`: Object of class `"numeric"`. Expected costs of the survey.

Methods

HTML `signature(x = "LtdSampling")`: Creates an html file containing the summary data and the diagnostic plots. Title, file name, output directory, css-file, etc. can additionally be specified using the parameters, `filename`, `outdir`, `CSSFile`, `Title`, as well as all the other parameters of the R2HTML-function `HTMLInitFile`.

sample `signature(x = "LtdSampling", size = c("fixed", "dynamic"))`: Sample herds using limited sampling. Additionally to the argument `x` of type `LtdSampling` the method takes an argument `size`, which is a character string. For `size == "fixed"` the fixed number of herds given in `x@nHerds` is sampled using simple random sampling. For `size == "dynamic"` dynamic sampling is used, i.e., based on real-time computation of the a-posteriori alpha- error the sample is updated until the a-posteriori alpha-error falls below the predefined significance level x@alpha. The return value is a list with two items: `indexSample` is a vector of indices of the sample corresponding to `x@surveyData@nAnimalVec` and `aPostAlpha` containing the a-posteriori alpha-error of the sample.

show `signature(object = "LtdSampling")`: Display structure of the class and content of the slots.

summary `signature(object = "LtdSampling")`: Display structure of the class and a summary of the content of the slots.

Note

No notes yet.

Author(s)

Ian Kopacka <ian.kopacka@ages.at>

See Also

The slot surveyData contains an object of the class SurveyData which is created using surveyData. Objects of the class LtdSampling are create using the constructor ltdSampling.

Examples

```
## Show the structure of the class:
showClass("LtdSampling")
## Create an object:
data(sheepData)
mySurvey <- surveyData(nAnimalVec = sheepData$nSheep,
    populationData = sheepData, designPrevalence = 0.002,
    alpha = 0.05, intraHerdPrevalence = 0.13,
    diagSensitivity = 0.9, costHerd = 30, costAnimal = 7.1)
myLtdSampling <- ltdSampling(survey.Data = mySurvey, sampleSizeLtd = 7)
## Display results:
summary(myLtdSampling)
## Write results to an html-file:
## Not run:
target <- HTMLInitFile(getwd(), filename = "LtdSampling")
HTML(myLtdSampling)
HTMLEndFile()
## End(Not run)
```

---

ltdSamplingSummary *Constructor for class 'LtdSamplingSummary'.*

Description

Creates an object of the class 'LtdSamplingSummary'. For given survey parameters (passed to the function as an object of the class SurveyData) ltdSamplingSummary() computes the mean herd sensitivity, the number of herds to test, the expected total number of animals to test and the expected total cost of a survey using limited sampling with a given sequence of animal-level sample sizes. This sequence ranges from 1 to a given upper bound sampleSizeLtdMax. If no upper bound is specified the maximal herd size is used.

Usage

```
ltdSamplingSummary(survey.Data, sampleSizeLtdMax, nSampleFixVec = NULL,
probVec = NULL)
```

Arguments

survey.Data	Object of class SurveyData. Created by using the function surveyData.
sampleSizeLtdMax	Positive integer. A series of parameters is computed for a sequence of sample limits. These sample limits range from 1 to a given upper bound, defined by sampleSizeLtdMax. If no upper bound is specified then 20 or - if less - the maximal herd size is used.
nSampleFixVec	Numeric vector containing some NAs (optional argument). For risk groups for which the sample size is fixed specify the sample size. For the risk groups for which the sample size should be computed set NA (order of the risk groups must be the same order as in survey.Data@riskValueData).
probVec	Numeric vector. For those risk groups for which the sample size should be computed sample probabilities must be specified. The vector must have the same length as the number of NA entries in nSampleFixVec or if nSampleFixVec is not specified, probVec must have the same length as the number of rows in survey.Data@riskValueData.

Value

The function returns an object of the class LtdSamplingSummary.

Author(s)

Ian Kopacka <ian.kopacka@ages.at>

References

A.R. Cameron and F.C. Baldock, "A new probablility formula to substantiate freedom from disease", Prev. Vet. Med. 34 (1998), pp. 1-17.

A.R. Cameron and F.C. Baldock, "Two-stage sampling surveys to substantiate freedom from disease", Prev. Vet. Med. 34 (1998), pp. 19-30.

M. Ziller, T. Selhorst, J. Teuffert, M. Kramer and H. Schlueter, "Analysis of sampling strategies to substantiate freedom from disease in large areas", Prev. Vet. Med. 52 (2002), pp. 333-343.

See Also

See LtdSamplingSummary and SurveyData for additional details.

Examples

```
data(sheepData)
sheepData$size <- ifelse(sheepData$nSheep < 30, "small", "large")
riskValueData <- data.frame(riskGroup = c("small", "large"),
    riskValues = c(1,2))
mySurvey <- surveyData(nAnimalVec = sheepData$nSheep,
riskGroupVec = sheepData$size,
riskValueData = riskValueData,
populationData = sheepData, designPrevalence = 0.002,
```

```
alpha = 0.05, intraHerdPrevalence = 0.13,
diagSensitivity = 0.9, costHerd = 30, costAnimal = 7.1)
## Limited sampling without risk groups:
myLtdSamplingSummary <- ltdSamplingSummary(survey.Data = mySurvey,
    sampleSizeLtdMax = 10)
## Limited sampling with risk groups:
myLtdSamplingRG <- ltdSamplingSummary(survey.Data = mySurvey,
    sampleSizeLtdMax = 10, nSampleFixVec = NULL, probVec = c(1,4))
```

`LtdSamplingSummary-class`
Class "LtdSamplingSummary"

Description

Contains the parameters and the data necessery for a survey to substantiate freedom from disease using "limited sampling". Additionally to the survey parameters: design prevalence (=prevalence of the disease under the null hypothesis), overall significance level (=1-confidence), intra-herd prevalence, sensitivity of the diagnostic test, cost per tested animal and cost per tested herd, the object contains the mean herd sensitivity, the number of herds to be tested, the mean overall number of animals to be tested and the expected costs for a range of possible sample limits (= fixed number of animals to test per herd).

Objects from the Class

Objects can be created by calls of the form `new("LtdSamplingSummary", ...)`.

Slots

`surveyData`**:** Object of class `"SurveyData"`. Containing the necessary survey parameters.

`sampleSizeLtdVec`**:** Object of class `"numeric"`. Pre-fixed number of animals to be tested per holding, irrespective of the herd size. If a herd contains fewer animals the entire herd is tested (vector).

`meanHerdSensVec`**:** Object of class `"numeric"` with values between 0 and 1. Mean herd sensitivity in the population (vector).

`meanHerdSensPerRGMx`**:** Object of class `"matrix"` with values between 0 and 1. Mean herd sensitivity of each risk group [columns] and sample limit [rows] (if population is stratified by risk groups).

`nHerdsVec`**:** Object of class `"numeric"`. Number of herds to be tested according to the herd sensitivity `meanHerdSensitivity` (vector).

`nHerdsPerRiskGroupMx`**:** Object of class `"matrix"`. Number of herds to be tested per risk group [columns] and sample limit [rows] (if population is stratified by risk groups).

`nSampleFixVec`**:** Object of class `"numeric"`. Numeric vector containing some NAs (optional argument). For risk groups for which the sample size is fixed it specifies the sample size. For the risk groups for which the sample size was computed it was set to NA (order of the risk groups is the same as in `survey.Data@riskValueData`).

`probVec`: Object of class `"numeric"`. Contains the sample probabilities for those risk groups for which the sample size was computed (=NA entries in `nSampleFixVec`).

`nAnimalsMeanVec`: Object of class `"numeric"`. Expected total number of animals to be tested in the survey (vector).

`expectedCostVec`: Object of class `"numeric"`. Expected costs of the survey (vector).

Methods

HTML `signature(x = "LtdSamplingSummary")`: Creates an html file containing the summary data and the diagnostic plots. Title, file name, output directory, css-file, etc. can additionally be specified using the parameters, `filename`, `outdir`, `CSSFile`, `Title`, as well as all the other parameters of the R2HTML-function `HTMLInitFile`.

plot `signature(x = "LtdSamplingSummary")`: Create plots of 1) the mean herd sensitivity, 2) the number of herds to be tested, 3) the expected total number of animals to be tested, 4) the expected total cost of the survey plotted against the vector of sample limits.

show `signature(object = "LtdSamplingSummary")`: Display structure of the class and content of the slots.

summary `signature(object = "LtdSamplingSummary")`: Display structure of the class and a summary of the content of the slots.

Note

No notes yet.

Author(s)

Ian Kopacka <ian.kopacka@ages.at>

See Also

The slot `surveyData` contains an object of the class `SurveyData`.

Examples

```
## Show the structure of the class:
showClass("LtdSamplingSummary")
## Create an object:
data(sheepData)
mySurvey <- surveyData(nAnimalVec = sheepData$nSheep,
    populationData = sheepData, designPrevalence = 0.002,
    alpha = 0.05, intraHerdPrevalence = 0.13,
    diagSensitivity = 0.9, costHerd = 30, costAnimal = 7.1)
myLtdSamplingSummary <- ltdSamplingSummary(survey.Data = mySurvey,
    sampleSizeLtdMax = 7)
## Display results:
summary(myLtdSamplingSummary)
plot(myLtdSamplingSummary)
## Write results to an html-file:
## Not run:
```

```
target <- HTMLInitFile(getwd(), filename = "LtdSampling")
HTML(myLtdSamplingSummary)
HTMLEndFile()
## End(Not run)
```

`sheepData` *Simulated test data frame of Austrian sheep holdings.*

Description

This test data frame contains a ficticious list of Austrian sheep holdings with information on the state and the number of animals.

Usage

```
data(sheepData)
```

Format

A data frame with 15287 observations on the following 3 variables.

`herdId` Numeric vector. Unique identification number of the herd.

`state` Numeric vector. The 9 Austrian states are coded using the numbers 1 to 9.

`nSheep` Numeric vector. Number of animals for each holding.

Source

Simulated test data.

Examples

```
data(sheepData)
```

`surveyData` *Constructor for class 'SurveyData'.*

Description

Constructor for objects of the class 'SurveyData'.

Usage

```
surveyData(nAnimalVec = numeric(), riskGroupVec = character(),
riskValueData = data.frame(), populationData = data.frame(),
    designPrevalence = numeric(), alpha = numeric(),
    intraHerdPrevalence = numeric(), diagSensitivity = numeric(),
    costHerd = numeric(), costAnimal = numeric())
```

Arguments

nAnimalVec	Positive numeric vector containing the number of animals per holding.
riskGroupVec	Character vector. Vector containing the the name of a risk group to which the farm belongs. Optional argument. If provided, it must have the same length as nAnimalVec.
riskValueData	Data frame. Data frame where the first column contains the labels in riskGroupVec and the second column contains the numeric values for the relative infection risk.
populationData	Data frame. Columns of the data frame must have the same length as the vector in nAnimalVec. The data frame can contain additional data such as herd id, name and address of the owner etc.
designPrevalence	Numeric with values between 0 and 1. Prevalence of the disease under the null hypothesis.
alpha	Numeric with values between 0 and 1. Type one error for the statistical test (=significance level).
intraHerdPrevalence	Numeric with values between 0 and 1. Intra-herd prevalence, i.e., the assumed prevalence of the disease within an infected herd.
diagSensitivity	Numeric with values between 0 and 1. Sensitivity of the diagnostic test used.
costHerd	Numeric. Cost per tested herd excluding costs for sampling of animals (e.g., travel costs of the vet).
costAnimal	Numeric. Cost per tested animal, e.g., drawing of samples + analysis in the lab.

Value

The function returns an object of the class SurveyData.

Author(s)

Ian Kopacka <ian.kopacka@ages.at>

See Also

See SurveyData for additional details on the class.

Examples

```
data(sheepData)
mySurvey <- surveyData(nAnimalVec = sheepData$nSheep,
    populationData = sheepData, designPrevalence = 0.002,
    alpha = 0.05, intraHerdPrevalence = 0.13,
    diagSensitivity = 0.9, costHerd = 30, costAnimal = 7.1)
```

SurveyData-class *Class "SurveyData"*

Description

Contains the parameters and the data necessery for a survey to substantiate freedom from disease using "individual sampling" or "limited sampling". The parameters are: design prevalence (=prevalence of the disease under the null hypothesis), overall significance level (=1-confidence), intra-herd prevalence, sensitivity of the diagnostic test, as well as cost per tested animal and cost per tested herd.

Objects from the Class

Objects can be created by calls of the form `new("SurveyData", ...)`.

Slots

`nAnimalVec`**:** Object of class `"numeric"` (nonnegative). Vector containing the number of animals per holding.

`riskGroupVec`**:** Object of class `"character"`. Vector containing the the name of a risk group to which the farm belongs. Optional argument. If provided, it must have the same length as `nAnimalVec`.

`riskValueData`**:** Object of class `"data.frame"`. Data frame where the first column contains the labels in `riskGroupVec` and the second column contains the numeric values for the relative infection risk.

`populationData`**:** Object of class `"data.frame"`. Columns of the data frame must have the same length as the vector in slot `nAnimalVec`. The data frame can contain additional data such as herd id, name and address of the owner etc.

`designPrevalence`**:** Object of class `"numeric"` with values between 0 and 1. Prevalence of the disease under the null hypothesis.

`alpha`**:** Object of class `"numeric"` with values between 0 and 1. Type one error for the statistical test (significance level).

`intraHerdPrevalence`**:** Object of class `"numeric"` with values between 0 and 1. Intra-herd prevalence, i.e., the assumed prevalence of the disease within an infected herd.

`diagSensitivity`**:** Object of class `"numeric"` with values between 0 and 1. Sensitivity of the diagnostic test used.

`costHerd`**:** Object of class `"numeric"`. Cost per tested herd excluding costs for sampling of animals (e.g., travel costs of the vet.)

`costAnimal`**:** Object of class `"numeric"`. Cost per tested animal, e.g., drawing of samples + analysis in the lab.

Methods

show `signature(object = "SurveyData")`: Display structure of the class and content of the slots.

summary `signature(object = "SurveyData")`: Display structure of the class and a summary of the content of the slots.

Note

No notes added yet.

Author(s)

Ian Kopacka <ian.kopacka@ages.at>

See Also

Objects of the class 'SurveyData' can be created using the constructor `surveyData` and are used as types for slots in the class `LtdSampling`.

Examples

```
## Show the structure of the class:
showClass("SurveyData")
## Create an object:
data(sheepData)
mySurvey <- surveyData(nAnimalVec = sheepData$nSheep,
    populationData = sheepData, designPrevalence = 0.002,
    alpha = 0.05, intraHerdPrevalence = 0.13,
    diagSensitivity = 0.9, costHerd = 30, costAnimal = 7.1)
## Display results:
summary(mySurvey)
```

Index