Project no. 310806

Providing a new generation of methodologies and tools for cost-effective risk-based animal health surveillance systems for the benefit of livestock producers, decision makers and consumers

KBBE

Collaborative Project

FP7-KBBE-2012-6

www.fp7-risksur.eu

Deliverable 6.24

R package of functions for risk-based surveillance

WP 6 – Decision making tools for implementing risk-based surveillance

Authors: Angus Cameron (AusVet); Franz Conraths (FLI); Andreas Fröhlich (FLI); Birgit Schauer (FLI); Katja Schulz (FLI); Evan Sergeant (AusVet); Jana Sonnenburg (FLI); Christoph Staubach (FLI)

Lead participant: FLI

Delivery date: January 2015

Reviewers: Jenny Frössling (SVA), Thomas Rosendal (SVA)

Dissemination level: Public

Nature: Report

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement N°310806.

Contents

Summary

Work package 6 focusses on the translation of frameworks, methodologies and algorithms developed under work packages 1 to 5 into a practical and accessible web-based decision support tool enabling decision makers, industry and the broader scientific community to take advantage of the research outcomes. The objective of Task 6.1, which is covered in this deliverable report, was to implement algorithms for the design and analysis of surveillance activities, in particular involving risk-based sampling.

The package *RSurveillance* was published within the open-source software environment R on 8 December 2014 (http://cran.r-project.org/web/packages/surveillance/index.html). This package includes 53 functions for the design and analysis of surveillance systems. The functions cover the topics demonstrating disease freedom (*n* = 40), including risk-based approaches (*n* = 12), prevalence estimation (*n* = 7) and functions for combined and parallel testing (*n* = 6). The publication of these 53 functions allows broader application of these algorithms and methods and validation by consortium members and the wider R community. These tools will be linked to the design framework developed under RISKSUR, where they will be described in more detail to further assist the future users in the correct choice and application of these algorithms and methods.

1 Introduction

Algorithms and methods for the design and analysis of surveillance systems (in particular those involving risk-based sampling) are often highly complex, and this complexity is one of the barriers to broader implementation. Experience has shown that even when scientists implement complex algorithms (for instance using a spreadsheet model), there is a high risk of calculation error and incorrect or inconsistent application of a method.

Therefore, Task 6.1 of RISKSUR was designed to implement algorithms for the design and analysis of surveillance activities, in particular involving risk-based sampling, as a package in the open-source software environment R (http://www.r-project.org/). Since this software is freely available and recognized as a standard for the rapid implementation and distribution of new analytical techniques, publication of tools for surveillance design and analysis in this environment is expected to facilitate broader application of these algorithms and methods.

The Description of Work proposed the following topics to be covered:

1. Freedom from infection
2. Early detection of infection or disease
3. Prevalence estimation
4. Case detection
5. Optimisation of surveillance with multiple objectives
6. Methods for the validation of risk factors used in designing risk-based surveillance systems

The aim of this deliverable is to describe the package published in R.

2 Methods

Relevant algorithms used for planning and analysis of surveillance activities were identified and implemented as functions in R. Most of the algorithms are based on published methods and have been implemented and tested previously in either the EpiTools suite of epidemiology and disease surveillance utilities (http://epitools.ausvet.com.au/content.php?page=home) or in FreeCalc (part of SurveyToolbox: available at http://epitools.ausvet.com.au/content. ?page=SurveyToolbox). A range of simple functions for sample size calculation and population sensitivity were initially developed to provide basic functionality. More complex functions were then developed, building on the basic functionality to provide for more advanced applications, including risk-based methods.

Development of the package included the testing of each function and provision of example applications as well as basic user help for each function. Full documentation of the functions is provided in the Reference Manual accompanying the R package or on the GitHub repository at https://github.com/evansergeant/RSurveillance. The final package was submitted and published on the CRAN repository.

3 Description of the published R package

3.1 Functions

The R package *RSurveillance* v0.1.0 was published with a GPL-2/GPL-3 licence on the R Cran website (http://cran.r-project.org/) on 8 December 2014. The GPL-2/GPL-3 licence is the most widely used free software licence, which allows end users to use, study, share (copy) and modify the software. The package includes 53 functions to support the design and analysis of disease surveillance activities. These functions were originally developed for animal health surveillance activities but can be equally applied to aquatic animal, wildlife, plant and human health surveillance activities.

RSurveillance functions are organised into three broad areas of surveillance, namely

- Demonstrating freedom from disease (*n* = 40), including
  - Representative freedom surveys (*n* = 12);
  - Freedom methods for imperfect specificity and finite populations (FreeCalc) (*n* = 9);
  - Risk-based freedom surveys (*n* = 12);
  - Probability of freedom estimation (*n* = 7);
- Prevalence estimation (n = 7);
- Functions for combined and parallel testing (*n* = 6).

Within these areas, functions can be further grouped according to purpose (depending on surveillance area/purpose), such as sample size calculation, population sensitivity estimation, miscellaneous and background functions (see Table 1 in Annex 1 - List of R functions encapsulated in the R package). Details of these functions are summarised in the reference manual in R and additional information and examples are available in R using the help() and example() functions.

3.2 Documentation

Two documents are submitted along with this deliverable report for further information:

- Reference manual which serves as user manual in R[1] (Annex 2 - Package 'RSurveillance') and
- Terminology document listing important definitions and formulas that serve as basis for the functions of the package RSurveillance[2] (Annex 3 - Important Formulae for Surveillance).

3.3 Relevant references

3.3.1 Epitools methodology

- Brown et al. (2001)
- Brunk et al. (1968)
- Cameron and Baldock (1998)
- Cameron (1999)
- Greiner and Gardner (2000)
- Humphry et al. (2004)
- Jordan and McEwen (1998)
- MacDiarmid (1988)
- Martin et al. (1992)
- Martin et al. (2007)
- Reiczigel et al. (2010)
- Rogan and Gladen (1979)
- Thrusfield (2007)

[1] http://cran.r-project.org/web/packages/RSurveillance/index.html
[2] http://epitools.ausvet.com.au/docs/Important-formulae-for-surveillance.pdf

3.3.2 Epitools applications

The following references are examples where EpiTools functions have been applied:

- Andreassen et al. (2012)
- Curran (2012)
- European Food Safety Authority (2009)
- Kittelberger et al. (2011)

4 Discussion

The aim of this deliverable was to describe the published R package. The package *RSurveillance* was made publicly available on 8 December 2014. Hence, the following sub-tasks have been fulfilled: a) implementation of algorithms as an R package, b) documentation, and d) publication. The package has been published prior to validation by consortium partners independent of the code development team (sub-task c). This was considered feasible as these functions have been previously implemented and tested as part of the EpiTools, HerdPlus and FreeCalc epidemiology and surveillance utilities (http://www.ausvet.com.au/). However, further validation, also in comparison to other published and non-published tools will be done as part of this project. The package is published at Github.com under GPL-2/GPL-3 which will allow future development and validation by both consortium partners as well as the scientific community at large.

The package *RSurveillance* includes 53 functions, which are described in more detail in the reference manual in R[3]. A more detailed description, examples and specific user-advice regarding applications will be provided as part of the web-based frameworks developed under RISKSUR. A definition of terminology used to describe these tools can be found on the EpiTools website (see footnote 2).

The functions relate mainly to demonstrating freedom from disease (n = 40), including risk-based approaches (n = 12), prevalence estimation (n = 7) and functions for combined and parallel testing (n = 6). No specific functions are included that refer to early detection, case detection, validation of risk factors and multi-objective surveillance. Specific functions for these objectives were not available for inclusion at the time this package was released and waiting for their completion would significantly delay publication of the work that has already been done. Additional functions can easily be added to an updated version of the existing package or as a new package(s) depending on preference at the time.

For publication, a GNU General Public Licence (GNU GPL or GPL) has been used. This is the most widely used free software licence, which allows end users to use, study, share (copy) and modify the software. Hence, further developments can take place following publication and validation within the R community. Therefore, this licence fulfils the requirement that the functions are made available to users for free and that functions are published in an environment that allows taking account of future development. However, this licence also implies that no warranty is provided regarding the contents. Modifications need to be saved as new versions so that potentially introduced errors are not erroneously assigned to authors of previous versions. This is the general policy of the R Cran environment.

In conclusion, the publication of the functions included in the R package *RSurveillance* allows broader application of these algorithms and methods and validation by consortium members and the wider R community. These tools will be linked to the design framework developed under RISKSUR, where they will be described in more detail to further assist the user in correct choice and application of these algorithms and methods.

[3] http://cran.r-project.org/web/packages/RSurveillance/index.html

References

Andreassen, A., S. Jore, P. Cuber, S. Dudman, T. Tengs, K. Isaksen, H. O. Hygen, H. Viljugrein, G. Anestad, P. Ottesen and K. Vainio (2012). "Prevalence of tick borne encephalitis virus in tick nymphs in relation to climatic factors on the southern coast of Norway." Parasites & Vectors **5**.

Brown, L. D., T. T. Cai, A. DasGupta, A. Agresti, B. A. Coull, G. Casella, C. Corcoran, C. Mehta, M. Ghosh, T. J. Santner, L. D. Brown, T. T. Cai and A. DasGupta (2001). "Interval estimation for a binomial proportion - Comment - Rejoinder." Statistical Science **16**(2): 101-133.

Brunk, H. D., J. E. Holstein and F. Williams (1968). "A comparison of binomial approximations to the hypergeometric dsitribution." American Statistician **22**(1): 24-&.

Cameron, A. R. (1999). Survey Toolbox for livestock diseases - A practical manual and software package for active surveillance of livestock diseases in developing countries. Canberra, Australia, Australian Centre for International Agricultural Research.

Cameron, A. R. and F. C. Baldock (1998). "A new probability formula for surveys to substantiate freedom from disease." Preventive Veterinary Medicine **34**(1): 1-17.

Curran, J. M. (2012). The surveillance and risk assessment of wild birds in northern Australia for highly pathogenic avian influenza H5N1 virus, Murdoch University.

European Food Safety Authority (2009). "Porcine brucellosis (Brucella suis). Scientific Opinion of the Panel on Animal Health and Welfare." The EFSA Journal **1144**: 1-112.

Greiner, M. and I. A. Gardner (2000). "Application of diagnostic tests in veterinary epidemiologic studies." Preventive Veterinary Medicine **45**(1-2): 43-59.

Humphry, R. W., A. Cameron and G. J. Gunn (2004). "A practical approach to calculate sample size for herd prevalence surveys." Preventive Veterinary Medicine **65**(3-4): 173-188.

Jordan, D. and S. A. McEwen (1998). "Herd-level test performance based on uncertain estimates of individual test performance, individual true prevalence and herd true prevalence." Preventive Veterinary Medicine **36**(3): 187-209.

Kittelberger, R., A. M. J. McFadden, M. J. Hannah, J. Jenner, R. Bueno, J. Wait, P. D. Kirkland, G. Delbridge, H. G. Heine, P. W. Selleck, T. W. Pearce, C. J. Pigott and J. S. O'Keefe (2011). "Comparative evaluation of four competitive/blocking ELISAs for the detection of influenza A antibodies in horses." Veterinary Microbiology **148**(2-4): 377-383.

MacDiarmid, S. C. (1988). "Future options for brucellosis surveillance in New Zealand beef herds." New Zealand Veterinary Journal **36**(1): 39-42.

Martin, P. A. J., A. R. Cameron and M. Greiner (2007). "Demonstrating freedom from disease using multiple complex data sources 1: a new methodology based on scenario trees." Preventive veterinary medicine **79**(2-4): 71-97.

Martin, S. W., M. Shoukri and M. A. Thorburn (1992). "Evaluating the health status of herds based on tests applied to individuals." Preventive Veterinary Medicine **14**(1-2): 33-43.

Reiczigel, J., J. Foldi and L. Ozsvari (2010). "Exact confidence limits for prevalence of a disease with an imperfect diagnostic test." Epidemiology and Infection **138**(11): 1674-1678.

Rogan, W. J. and B. Gladen (1979). "Estimating prevalence from the results of a screening test." American Journal of Epidemiology **107**(1): 71-76.

Thrusfield, M. (2007). Veterinary epidemiology, 3rd edition (May 29, 2007). London, Wiley-Blackwell.

Annexes

Annex 1 - List of R functions encapsulated in the R package

Table 1. List of R functions encapsulated in the R package *RSurveillance* by group (source: https://github.com/evansergeant/RSurveillance).

Group	No.	Function	Details
I Representative freedom surveys			
I.1 Population sensitivity estimation			
	1	sep.binom	Binomial population sensitivity
	2	sep.hypergeo	Hypergeometric population sensitivity
	3	sep.exact	Population sensitivity for census (all units tested)
	4	spp	Population specificity
	5	sep	Population sensitivity
	6	sep.var.se	Population sensitivity for varying unit sensitivity
	7	sep.sys	Two-stage population sensitivity
I.2 Sample size estimation			
	8	n.binom	Binomial sample size
	9	n.hypergeo	Hypergeometric sample size
	10	n.freedom	Freedom sample size
	11	n.2stage	Two-stage freedom sample size
I.3 Miscellaneous functions			
	12	pstar.calc	Design prevalence back-calculation
II Freedom methods for imperfect specificity and finite populations (FreeCalc)			
II.1 Population sensitivity estimation			
	13	sep.freecalc	Population sensitivity estimation
	14	sep.hp	Hypergeometric (HerdPlus) population sensitivity for imperfect test
	15	sep.binom.imperfect	Binomial population sensitivity for imperfect test
II.2 Population specificity estimation			
	16	sph.binom	Binomial population specificity for imperfect test
	17	sph.hp	Hypergeometric population specificity calculation
II.3 Sample size estimation			
	18	n.freecalc	Freecalc sample size for a finite population and specified cut-point number of positives
	19	n.hp	Hypergeometric (HerdPlus) sample size for finite population and specified cut-point number of positives
	20	n.c.freecalc	Freecalc optimum sample size and cut-point number of positives
	21	n.c.hp	Hypergeometric (HerdPlus) optimum sample size and cut-point number of positives
III Risk-based freedom surveys			
III.1 Population sensitivity estimation			
	22	sep.rb.bin	Binomial risk-based sensitivity estimation
	23	sep.rb.hypergeo	Hypergeometric risk-based population sensitivity
	24	sep.rb.bin.varse	Binomial risk-based population sensitivity for varying unit sensitivity
	25	sep.rb.hypergeo.varse	Hypergeometric risk-based population sensitivity for varying unit sensitivity

Group	No.	Function	Details
	26	sep.rb2.bin	Binomial risk-based population sensitivity for two risk factors
	27	sep.rb2.hypergeo	Hypergeometric risk-based population sensitivity for two risk factors
	28	sse.rb.2stage	Two-stage risk-based system sensitivity
	29	sse.combined	System sensitivity by combining multiple surveillance components
III.2 Sample size estimation			
	30	n.rb	Risk-based sample size
	31	n.rb.varse	Risk-based sample size for varying unit sensitivity
III.3 Miscellaneous functions			
	32	adj.risk	Adjusted risk
	33	epi.calc	Effective probability of infection
IV Probability of freedom estimation			
IV.1 Probability of freedom estimation			
	34	pfree.1	Probability of freedom for single time period
	35	pfree.calc	Probability of freedom over time
	36	pfree.equ	Equilibrium probability of freedom
IV.2 Miscellaneous functions			
	37	n.pfree	Sample size to achieve desired (posterior) probability of freedom
	38	sep.pfree	Population sensitivity to achieve desired (posterior) probability of freedom
	39	sep.prior	Population sensitivity to achieve desired prior probability of freedom
IV.3 Background functions			
	40	disc.prior	Discounted prior probability of freedom
V Prevalence estimation			
V.1 Apparent prevalence and confidence interval estimation			
	41	ap	Apparent prevalence
	42	binom.agresti	Agresti-Coull confidence limits
	43	binom.jeffreys	Jeffreys confidence limits
	44	binom.cp	Clopper Pearson exact confidence limits
	45	n.ap	Sample size for apparent prevalence
V.2 True prevalence and confidence interval estimation			
	46	n.tp	Sample size for true prevalence
	47	sd.tp	Standard deviation of true prevalence estimate
VI Combining tests			
	48	se.series	Sensitivity of tests in series
	49	se.parallel	Sensitivity of tests in parallel
	50	sp.series	Specificity of tests in series
	51	sp.parallel	Specificity of tests in parallel
VII Pooled testing			
	52	sep.pooled	Pooled population sensitivity
	53	n.pooled	Sample size for pooled testing for freedom

Annex 2 - Package ‘RSurveillance’

Reference manual which serves as user manual in R; Pages 11-56

Annex 3 - Important Formulae for Surveillance

Terminology document listing important definitions and formulas that serve as basis for the functions of the package RSurveillance; Pages 57-64

Package ‘RSurveillance’

December 8, 2014

Type Package

Title Design and Analysis of Disease Surveillance Activities

Version 0.1.0

Date 2014-11-15

Author Evan Sergeant

Maintainer Evan Sergeant <evan@ausvet.com.au>

Description This package provides a range of functions for the design and analysis of disease surveillance activities. These functions were originally developed for animal health surveillance activities but can be equally applied to aquatic animal, wildlife, plant and human health surveillance activities. Utilities are included for sample size calculation and analysis of representative surveys for disease freedom, risk-based studies for disease freedom and for prevalence estimation.

License GPL-2 | GPL-3

LazyLoad yes

Imports epitools, epiR

NeedsCompilation no

Repository CRAN

Date/Publication 2014-12-08 07:06:21

R topics documented:

`adj.risk` *Adjusted risk*

Description

Calculates adjusted risk for given relative risk and population proportions. This is an intermediate calculation in the calculation of effective probability of infection for risk-based surveillance activities

Usage

```
adj.risk(rr, ppr)
```

Arguments

`rr`	relative risk values (vector of values corresponding to the number of risk strata)
`ppr`	population proportions corresponding to rr values (vector of equal length to rr)

Value

vector of adjusted risk values (in order corresponding to rr)

Examples

```
# examples for adj.risk
adj.risk(c(5, 1), c(0.1, 0.9))
adj.risk(c(5, 3, 1), c(0.1, 0.1, 0.8))
```

`ap` *Apparent prevalence*

Description

Estimates apparent prevalence and confidence limits for given sample size and result, assuming representative sampling

Usage

```
ap(x, n, type = "wilson", conf = 0.95)
```

Arguments

`x`	number of positives in sample
`n`	sample size, note: either x or n can be a vector, but at least one must be scalar
`type`	method for estimating CI, one of c("normal", "exact", "wilson", "jeffreys", "agresti-coull", "all"), default = "wilson"
`conf`	level of confidence required, default = 0.95 (scalar)

Value

either 1) if type = "all", a list with 5 elements, each element a matrix with 6 columns, x, n, proportion, lower confidence limit, upper confidence limit, confidence level and CI method; or 2) a matrix of results for the chosen method

Examples

```
# examples for ap function
n<- 200
x<- 25
conf<- 0.95
ap(x, n)
ap(seq(10, 100, 10), 200, type = "agresti")
ap(seq(10, 100, 10), 200, type = "all")
```

`binom.agresti` *Agresti-Coull confidence limits*

Description

Calculates Agresti-Coull confidence limits for a simple proportion (apparent prevalence)

Usage

```
binom.agresti(x, n, conf = 0.95)
```

Arguments

`x`	number of positives in sample
`n`	sample size, note: either x or n can be a vector, but at least one must be scalar
`conf`	level of confidence required, default 0.95 (scalar)

Value

a dataframe with 6 columns, x, n, proportion, lower confidence limit, upper confidence limit, confidence level and CI method

Examples

```
# test binom.agresti
binom.agresti(25, 200)
binom.agresti(seq(10, 100, 10), 200)
binom.agresti(50, seq(100, 1000, 100))
```

---

binom.cp *Clopper-Pearson exact confidence limits*

Description

Calculates Clopper-Pearson exact binomial confidence limits for a simple proportion (apparent prevalence)

Usage

```
binom.cp(x, n, conf = 0.95)
```

Arguments

`x`	number of positives in sample
`n`	sample size, note: either x or n can be a vector, but at least one must be scalar
`conf`	level of confidence required, default = 0.95 (scalar)

Value

a dataframe with 6 columns, x, n, proportion, lower confidence limit, upper confidence limit, confidence level and CI method

Examples

```
# test binom.cp
binom.cp(25, 200)
binom.cp(seq(10, 100, 10), 200)
binom.cp(50, seq(100, 1000, 100))
```

`binom.jeffreys` *Jeffreys confidence limits*

Description

Calculates Jeffreys confidence limits for a simple proportion (apparent prevalence)

Usage

```
binom.jeffreys(x, n, conf = 0.95)
```

Arguments

`x`	number of positives in sample
`n`	sample size, note: either x or n can be a vector, but at least one must be scalar
`conf`	level of confidence required, default = 0.95 (scalar)

Value

a dataframe with 6 columns, x, n, proportion, lower confidence limit, upper confidence limit, confidence level and CI method

Examples

```
# test binom.jeffreys
binom.jeffreys(25, 200)
binom.jeffreys(seq(10, 100, 10), 200)
binom.jeffreys(50, seq(100, 1000, 100))
```

`disc.prior` *Discounted prior probability of freedom*

Description

Calculates the discounted prior probability of disease freedom, after adjusting for the probability of disease exceeding the design prevalence during the time period of the surveillance data being analysed

Usage

```
disc.prior(prior, p.intro)
```

Arguments

`prior`	prior probability of freedom before surveillance
`p.intro`	probability of introduction (or of prevalence exceeding the design prevalence) during the time period (scalar or vector equal length to prior)

Value

vector of discounted prior probabilities of freedom

Examples

```
# examples for disc.prior
disc.prior(0.5, 0.01)
disc.prior(0.95, c(0.001, 0.005, 0.01, 0.02, 0.05))
disc.prior(c(0.5, 0.6, 0.7, 0.8, 0.9, 0.95), 0.01)
```

---

epi.calc *Effective probability of infection (EPI)*

Description

Calculates effective probability of infection (adjusted design prevalence) for each risk group for risk-based surveillance activities

Usage

```
epi.calc(pstar, rr, ppr)
```

Arguments

pstar	design prevalence (scalar)
rr	relative risk values (vector of values corresponding to the number of risk strata)
ppr	population proportions corresponding to rr values (vector of equal length to rr)

Value

list of 2 elements, a vector of EPI values and a vector of corresponding adjusted risks (in corresponding order to rr)

Examples

```
# examples for epi.calc
epi.calc(0.1, c(5, 1), c(0.1, 0.9))
epi.calc(0.02, c(5, 3, 1), c(0.1, 0.1, 0.8))
```

n.2stage *2-stage freedom sample size*

Description

Calculates sample sizes for a 2-stage representative survey (sampling of clusters and units within clusters) for disease freedom or detection, assuming imperfect test sensitivity, perfect test specificity and representative sampling

Usage

```
n.2stage(H = NA, N = NA, sep.sys = 0.95, sep.c, pstar.c, pstar.u,
  se = 1)
```

Arguments

`H`	population size = number of clusters or NA if not known, default = NA
`N`	population sizes for clusters, default = NA, scalar or vector of population sizes for clusters
`sep.sys`	desired population sensitivity (scalar)
`sep.c`	desired cluster-level sensitivity (scalar)
`pstar.c`	specified cluster-level design prevalence as proportion or integer (scalar)
`pstar.u`	specified population-level design prevalence as proportion or integer (scalar)
`se`	unit sensitivity (scalar)

Value

a list of 2 elements, the number of clusters to sample and a vector of sample sizes per cluster

Examples

```
# examples of n.2stage - checked
n.2stage(NA, NA, 0.95, 0.5, 0.01, 0.1, 0.95)
n.2stage(500, NA, 0.95, 0.5, 10, 0.1, 0.95)
n.2stage(1000, c(50, 100, 200, 500, 1000, 5000, NA), 0.95, 0.5, 0.01, 0.05, 0.8)
n.2stage(1000, c(50, 100, 200, 500, 1000, 5000, NA), 0.95, 0.5, 0.01, 1, 0.8)
n.2stage(1000, c(50, 100, 200, 500, 1000, 5000, NA), 0.9, 0.95, 1, 0.1, 0.8)
```

`n.ap` *Sample size for apparent prevalence*

Description

Calculates sample size for estimating apparent prevalence (simple proportion)

Usage

```
n.ap(p, precision, conf = 0.95)
```

Arguments

`p`	expected proportion, scalar or vector of values
`precision`	absolute precision, +/- proportion equivalent to half the width of the desired confidence interval, scalar or vector of values, note: at least one of p and precision must be a scalar
`conf`	level of confidence required, default = 0.95 (scalar)

Value

a vector of sample sizes

Examples

```
# examples of n.ap
n.ap(0.5, 0.1)
n.ap(0.5, 0.1, conf=0.99)
n.ap(seq(0.1, 0.5, by = 0.1), 0.05)
n.ap(0.2, c(0.01, 0.02, 0.05, 0.1))
```

`n.binom` *Binomial sample size*

Description

Calculates sample size for demonstrating freedom or detecting disease using binomial approach and assuming imperfect test sensitivity, perfect test specificity and representative sampling

Usage

```
n.binom(sep, pstar, se = 1)
```

Arguments

`sep`	desired population sensitivity (scalar or vector)
`pstar`	specified design prevalence (scalar or vector of same length as sep)
`se`	unit sensitivity, default = 1 (scalar or vector of same length as sep)

Value

vector of sample sizes

Examples

```
# examples for n.binom - checked
n.binom(sep=0.95, pstar=c(0.01, 0.02, 0.05, 0.1, 0.2))
n.binom(c(0.5, 0.8, 0.9, 0.95), 0.01)
```

`n.c.freecalc` *Freecalc optimum sample size and cut-point number of positives*

Description

Calculates optimum sample size and cut-point number of positives to achieve specified population sensitivity, for given population size and other parameters, using freecalc algorithm, all paramaters must be scalars

Usage

```
n.c.freecalc(N, sep = 0.95, c = 1, se, sp = 1, pstar, minSpH = 0.95)
```

Arguments

`N`	population size
`sep`	target population sensitivity
`c`	The maximum allowed cut-point number of positives to classify a cluster as positive, default=1, if positives < c result is negative, >= c is positive
`se`	test unit sensitivity
`sp`	test unit specificity, default=1
`pstar`	design prevalence as a proportion or integer (number of infected units)
`minSpH`	minimium desired population specificity

Value

a list of 3 elements, a dataframe with 1 row and six columns for the recommended sample size and corresponding values for population sensitivity (SeP), population specificity (SpP), N, c and pstar, a vector of SeP values and a vector of SpP values, for n = 1:N

Examples

```
# examples for n.c.hp
n.c.freecalc(120,0.95,c=5,se=0.9,sp=0.99,pstar=0.1, minSpH=0.9)[[1]]
n.c.freecalc(65,0.95,c=5,se=0.95,sp=0.99,pstar=0.05, minSpH=0.9)
```

`n.c.hp` *Hypergeometric (HerdPlus) optimum sample size and cut-point number of positives*

Description

Calculates optimum sample size and cut-point positives to achieve specified population sensitivity, for given population size and other parameters, all paramaters must be scalars

Usage

```
n.c.hp(N, sep = 0.95, c = 1, se, sp = 1, pstar, minSpH = 0.95)
```

Arguments

`N`	population size
`sep`	target population sensitivity
`c`	The maximum allowed cut-point number of positives to classify a cluster as positive, default=1, if positives < c result is negative, >= c is positive
`se`	test unit sensitivity
`sp`	test unit specificity, default=1
`pstar`	design prevalence as a proportion or integer (number of infected units)
`minSpH`	minimium desired population specificity

Value

a list of 3 elements, a dataframe with 1 row and six columns for the recommended sample size and corresponding values for population sensitivity (SeP), population specificity (SpP), N, c and pstar, a vector of SeP values and a vector of SpP values, for n = 1:N

Examples

```
# examples for n.c.hp
n.c.hp(65,0.95,c=5,se=0.95,sp=0.99,pstar=0.05, minSpH=0.9)[[1]]
tmp<- n.c.hp(120,0.95,c=5,se=0.9,sp=0.99,pstar=0.1, minSpH=0.9)
```

n.freecalc *Freecalc sample size for a finite population and specified cut-point number of positives*

Description

Calculates sample size required for a specified population sensitivity, for a given population size, cut-point number of positives and other parameters, using Freecalc algorithm. All paramaters must be scalars

Usage

```
n.freecalc(N, sep = 0.95, c = 1, se, sp = 1, pstar, minSpH = 0.95)
```

Arguments

`N`	population size
`sep`	target population sensitivity
`c`	The cut-point number of positives to classify a cluster as positive, default=1, if positives < c result is negative, >= c is positive
`se`	test unit sensitivity
`sp`	test unit specificity, default=1
`pstar`	design prevalence as a proportion or integer (number of infected units)
`minSpH`	minimium desired population specificity

Value

a list of 2 elements, a dataframe with 1 row and six columns for the recommended sample size and corresponding values for population sensitivity (SeP), population specificity (SpP), N, c and pstar and a dataframe of n rows with SeP and SpP values for each value of n up to the recommended value

Examples

```
# examples for n.freecalc
n.freecalc(65,0.95,c=1,se=0.95,sp=0.99,pstar=0.05, minSpH=0.9)[[1]]
n.freecalc(65,0.95,c=2,se=0.95,sp=0.99,pstar=0.05, minSpH=0.9)[[1]]
n.freecalc(65,0.95,c=3,se=0.95,sp=0.99,pstar=0.05, minSpH=0.9)
```

n.freedom *Freedom sample size*

Description

Calculates sample size for demonstrating freedom or detecting disease using the appropriate method, depending on whether or not N provided (hypergeometric if N provided, binomial otherwise), assuming imperfect test sensitivity, perfect test specificity and representative sampling

Usage

```
n.freedom(N = NA, sep = 0.95, pstar, se = 1)
```

Arguments

`N`	population size, default = NA (unknown) (scalar or vector of same length as sep)
`sep`	desired population sensitivity (scalar or vector)
`pstar`	specified design prevalence as proportion or integer (scalar or vector of same length as sep)
`se`	unit sensitivity (scalar or vector of same length as sep)

Value

vector of sample sizes, NA if N is specified and n>N

Examples

```
# examples for n.freedom - checked
n.freedom(NA, sep=0.95, pstar=0.01, se=1)
n.freedom(500, sep=0.95, pstar=0.01, se=1)
n.freedom(N=c(100, 500, 1000, 5000, 10000, 100000, NA), sep=0.95, pstar=0.01, se=1)
n.freedom(500, sep=0.95, pstar=0.01, se=c(0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 1))
```

n.hp *Hypergeometric (HerdPlus) sample size for finite population and specified cut-point number of positives*

Description

Calculates sample size to achieve specified population sensitivity with population specificity >= specified minimum value, for given population size, cut-point number of positives and other parameters, all paramaters must be scalars

Usage

```
n.hp(N, sep = 0.95, c = 1, se, sp = 1, pstar, minSpH = 0.95)
```

Arguments

`N`	population size
`sep`	target population sensitivity
`c`	The cut-point number of positives to classify a cluster as positive, default=1, if positives < c result is negative, >= c is positive
`se`	test unit sensitivity
`sp`	test unit specificity, default=1
`pstar`	design prevalence as a proportion or integer (number of infected units)
`minSpH`	minimium desired population specificity

Value

A list of 2 elements, a dataframe with 1 row and six columns for the recommended sample size and corresponding values for population sensitivity (SeP), population specificity (SpP), N, c and pstar and a dataframe of n rows with SeP and SpP values for each value of n up to the recommended value. Returns sample size for maximum achievable sep if it is not possible to achieve target sep AND SpP>= minSpH.

Examples

```
# examples for n.hp
n.hp(65,0.95,c=1,se=0.95,sp=0.99,pstar=0.05, minSpH=0.9)[[1]]
n.hp(65,0.95,c=2,se=0.95,sp=0.99,pstar=0.05, minSpH=0.9)
```

`n.hypergeo` *Hypergeometric sample size*

Description

Calculates sample size for demonstrating freedom or detecting disease using hypergeometric approximation and assuming imperfect test sensitivity, perfect test specificity and representative sampling

Usage

```
n.hypergeo(sep, N, d, se = 1)
```

Arguments

`sep`	desired population sensitivity (scalar or vector)
`N`	population size (scalar or vector of same length as sep)
`d`	expected number of infected units in population, = design prevalence*N rounded to next integer (scalar or vector of same length as sep)
`se`	unit sensitivity, default = 1 (scalar or vector of same length as sep)

Value

vector of sample sizes, NA if n>N

Examples

```
# examples for n.hypergeo - checked
n.hypergeo(0.95, N=100, d=1, se = 0.95)
n.hypergeo(sep=0.95, N=c(100, 200, 500, 1000, 10000), d=ceiling(0.01*c(100, 200, 500, 1000, 10000)))
n.hypergeo(c(0.5, 0.8, 0.9, 0.95), N=100, d=5)
n.hypergeo(0.95, N=80, d=c(1, 2, 5, 10))
n.hypergeo(0.95, N=80, d=c(1, 2, 5, 10), se = 0.8)
```

`n.pfree` *Sample size to achieve desired (posterior) probability of freedom*

Description

Calculates the sample size required to achieve a given value for probability of disease freedom

Usage

```
n.pfree(pfree, prior, p.intro, pstar, se, N = NA)
```

Arguments

`pfree`	desired probability of freedom (scalar or vector)
`prior`	prior probability of freedom before surveillance (scalar or vector of same length as pfree)
`p.intro`	probability of introduction for time period (scalar or vector of same length as pfree)
`pstar`	design prevalence (scalar or vector of same length as pfree)
`se`	unit sensitivity (scalar or vector of same length as pfree)
`N`	population size (scalar or vector of same length as pfree)

Value

vector of sample sizes

Examples

```
# examples for n.pfree
n.pfree(0.95, 0.5, 0.01, 0.05, 0.9)
n.pfree(0.95, 0.5, 0.01, 0.05, 0.9, N=300)
n.pfree(pfree = c(0.9, 0.95, 0.98, 0.99), prior = 0.7, 0.01, 0.01, 0.8, 1000)
n.pfree(0.95, 0.7, 0.01, 0.1, 0.96)
```

n.pooled *Sample size for pooled testing for freedom*

Description

Calculates sample size to achieve desired population-level sensitivity, assuming pooled sampling and allowing for imperfect sensitivity and specificity of the pooled test

Usage

```
n.pooled(sep, k, pstar, pse, psp = 1)
```

Arguments

sep	desired population sensitivity (scalar or vector)
k	pool size (constant across pools) (scalar or vector of same length as sep)
pstar	design prevalence (scalar or vector of same length as sep)
pse	pool-level sensitivity (scalar or vector of same length as sep)
psp	pool-level specificity (scalar or vector of same length as sep)

Value

vector of sample sizes

Examples

```
# examples for n.pooled
n.pooled(0.95, 5, 0.01, 1, 1)
n.pooled(0.95, 10, 0.1, 0.9, 1)
n.pooled(0.95, c(2, 5, 10, 20), 0.1, c(0.99, 0.98, 0.97, 0.95), 1)
```

n.rb *Risk-based sample size*

Description

Calculates sample size for risk-based sampling for a single risk factor and using binomial method

Usage

```
n.rb(pstar, rr, ppr, spr, se, sep)
```

Arguments

pstar	design prevalence (scalar)
rr	relative risk values (vector, length equal to the number of risk strata)
ppr	population proportions corresponding to rr values (vector of equal length to rr)
spr	planned surveillance proportion for each risk group (vector equal length to rr, ppr)
se	unit sensitivity (fixed or vector same length as rr, ppr, n)
sep	required population sensitivity (scalar)

Value

list of 2 elements, a vector of sample sizes for each risk group a scalar of total sample size, a vector of EPI values and a vector of adjusted risks

Examples

```
# examples for n.rb
n.rb(0.1, c(5, 3, 1), c(0.1, 0.10, 0.80), c(0.5, 0.3, 0.2), 0.9, 0.95)
n.rb(0.01, c(5, 1), c(0.1, 0.9), c(0.8, 0.2), c(0.9, 0.95), 0.95)
```

---

n.rb.varse *Risk-based sample size for varying unit sensitivity*

Description

Calculates sample size for risk-based sampling for a single risk factor and varying unit sensitivity, using binomial method

Usage

```
n.rb.varse(pstar, rr, ppr, spr, se, spr.rg, sep)
```

Arguments

pstar	design prevalence (scalar)
rr	relative risk values (vector, length equal to the number of risk strata)
ppr	population proportions for each risk group, vector of same length as rr
spr	planned surveillance proportions for each risk group, vector of same length as rr
se	unit sensitivities (vector of group values)
spr.rg	proportions of samples for each sensitivity value in each risk group (matrix with rows = risk groups, columns = sensitivity values), row sums must equal 1
sep	required population sensitivity (scalar)

Value

list of 3 elements, a matrix of sample sizes for each risk and sensitivity group, a vector of EPI values and a vector of mean sensitivity for each risk group

Examples

```
# examples for n.rb.varse
m<- rbind(c(0.8, 0.2), c(0.5, 0.5), c(0.7, 0.3))
n.rb.varse(0.01, c(5, 3, 1), c(0.1, 0.1, 0.8), c(0.4, 0.4, 0.2), c(0.92, 0.8), m, 0.95)

m<- rbind(c(0.8, 0.2), c(0.6, 0.4))
n.rb.varse(0.05, c(3, 1), c(0.2, 0.8), c(0.7, 0.3), c(0.95, 0.8), m, 0.95)

m<- rbind(c(1), c(1))
n.rb.varse(0.05, c(3, 1), c(0.2, 0.8), c(0.7, 0.3), c(0.95), m, 0.99)
```

`n.tp` *Sample size for true prevalence*

Description

Calculates sample size for estimating true prevalence using normal approximation

Usage

```
n.tp(p, se, sp, precision, conf = 0.95)
```

Arguments

`p`	estimated true prevalence (scalar or vector)
`se`	test sensitivity (scalar or vector)
`sp`	test specificity (scalar or vector)
`precision`	absolute precision, +/- proportion equal to half the width of the desired confidence interval (scalar or vector)
`conf`	desired level of confidence for CI, default = 0.95 (scalar or vector)

Value

a vector of sample sizes

Examples

```
# examples for n.tp
n.tp(0.1, 0.9, 0.99, 0.05)
n.tp(0.1, 0.9, 0.99, 0.05, conf = 0.99)
n.tp(c(0.05, 0.1, 0.2, 0.3, 0.4, 0.5), 0.9, 0.99, 0.05)
n.tp(0.5, 0.9, 0.99, c(0.01, 0.02, 0.05, 0.1, 0.2))
```

`pfree.1` *Probability of freedom for single time period*

Description

Calculates the posterior probability (confidence) of disease freedom (negative predictive value) for a single time period

Usage

```
pfree.1(sep, p.intro, prior = 0.5)
```

Arguments

`sep`	population sensitivity for time period (scalar or vector)
`p.intro`	probability of introduction for time period (scalar or vector of same length as sep)
`prior`	prior probability of freedom before surveillance (scalar or vector of same length as sep)

Value

`data.frame` with columns for sep, p.intro, discounted prior, pfree, pfree.equ and prior.equ

Examples

```
# examples for pfree.1
pfree.1(0.8, 0.01, 0.5)
pfree.1(0.6, c(0.001, 0.005, 0.01, 0.02, 0.05), 0.5)
pfree.1(runif(10, 0.4, 0.6), 0.01, 0.5)
pfree.1(runif(10, 0.4, 0.6), runif(10, 0.005, 0.015), 0.5)
```

`pfree.calc` *Probability of freedom over time*

Description

Calculates the probability (confidence) of disease freedom for given prior, sep and p.intro over 1 or more time periods

Usage

```
pfree.calc(sep, p.intro, prior = 0.5)
```

Arguments

`sep`	population sensitivity for each time period (vector)
`p.intro`	probability of introduction for each time period (scalar or vector of same length as sep)
`prior`	prior probability of freedom before surveillance (scalar)

Value

`data.frame` with columns for sep, p.intro, discounted prior, probability of freedom, equilibrium probability of freedom and equilibrium prior

Examples

```
# examples for pfree.calc
pfree.calc(0.8, 0.01, 0.5)
pfree.calc(rep(0.6,24), 0.01, 0.5)
pfree.calc(runif(10, 0.4, 0.6), 0.01, 0.5)
pfree.calc(runif(10, 0.4, 0.6), runif(10, 0.005, 0.015), 0.5)
```

`pfree.equ` *Equilibrium probability of freedom*

Description

Calculates equilibrium probability of disease freedom and equilibrium prior probability of freedom, after discounting for probability of introduction

Usage

```
pfree.equ(sep, p.intro)
```

Arguments

`sep`	population sensitivity for time period (scalar or vector)
`p.intro`	probability of introduction for time period (scalar or vector of same length as sep)

Value

a list of 2 vectors, equilibrium posterior probability of freedom and equilibrium prior (discounted) probability of freedom

Examples

```
# examples of pfree.equ
pfree.equ(runif(10, 0.4, 0.6), 0.01)
pfree.equ(0.8, 0.05)
pfree.equ(rep(0.9, 6), c(0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05))
```

`pstar.calc` *Design prevalence back-calculation*

Description

Calculates design prevalence required for given sample size and desired population-level sensitivity, assuming imperfect test sensitivity, perfect test specificity and representative sampling

Usage

```
pstar.calc(N = NA, n, sep, se)
```

Arguments

`N`	populaton size if known (scalar or vector of same length as n)
`n`	sample size (scalar or vector)
`sep`	desired population sensitivity (scalar or vector of same length as n)
`se`	unit sensitivity (scalar or vector of same length as n)

Value

vector of design prevalence values

Examples

```
# examples of pstar.calc- checked
pstar.calc(NA, 280, 0.95, 0.98)
pstar.calc(500, 250, sep=0.95, se=1)
pstar.calc(N=c(100, 500, 1000, 5000, 10000, 100000, NA), n=30, sep=0.95, se=1)
pstar.calc(500, n=30, sep=0.95, se=c(0.5, 0.6, 0.7, 0.8, 0.9, 0.99, 1))
```

`sd.tp` *Standard deviation of true prevalence estimate*

Description

Calculates the standard deviation of true prevalence estimate assuming se and sp known exactly, used to calculate normal approximation CI for estimate

Usage

```
sd.tp(x, n, se, sp)
```

Arguments

`x`	number of positive results in sample (scalar or vector)
`n`	sample size (scalar or vector)
`se`	test sensitivity (scalar or vector)
`sp`	test specificity (scalar or vector)

Value

vector of standard deviation values for true prevalence estimates

Examples

```
# example of sd.tp
sd.tp(1:10, 20, 0.9, 0.99)
```

`se.parallel` *Sensitivity of tests in parallel*

Description

Calculates the combined sensitivity for multiple tests interpreted in parallel (assuming independence)

Usage

```
se.parallel(se)
```

Arguments

`se`	vector of unit sensitivity values

Value

scalar of combined sensitivity, assuming independence

Examples

```
# examples for se.parallel
se.parallel(c(0.99, 0.95, 0.8))
```

se.series — *Sensitivity of tests in series*

Description

Calculates the combined sensitivity for multiple tests interpreted in series (assuming independence)

Usage

```
se.series(se)
```

Arguments

`se`	vector of unit sensitivity values

Value

scalar of combined sensitivity, assuming independence

Examples

```
# examples for se.series
se.series(c(0.99, 0.95, 0.8))
```

sep — *Population sensitivity*

Description

Calculates population sensitivity using appropriate method, depending on whether or not N provided (hypergeometric if N provided, binomial otherwise), assuming perfect test specificity and representative sampling

Usage

```
sep(N = NA, n, pstar, se = 1)
```

Arguments

`N`	population size, NA or vector of same length as n
`n`	sample size (number tested), scalar or vector
`pstar`	design prevalence as a proportion or integer, scalar or vector of same length as n
`se`	unit sensitivity, scalar or vector of same length as n

Value

a vector of population-level sensitivities

Examples

```
# examples for sep - checked
sep(n=300, pstar=0.01, se=1)
sep(NA, 300, 0.01, 1)
sep(10000, 150, 0.02, 1)
sep(n=1:100, pstar = 0.05, se=0.95)
N<- seq(30, 100, by = 5)
se<- 0.95
pstar<- 0.1
n<- rep(30, length(N))
sep(N, n, pstar, se = se)
sep(rep(100, 10), seq(10, 100, by = 10), pstar = 1, se=0.99)
N<- c(55, 134, NA, 44, 256)
n<- c(15, 30, 28, 15, 33)
sep(N, n, 0.1, 0.95)
```

---

sep.binom *Binomial Population sensitivity*

Description

Calculates population sensitivity for detecting disease, assuming imperfect test sensitivity and specificity and representative sampling, using binomial distribution (assumes large or unknown population size and that cut-point number of reactors for a positive result = 1)

Usage

```
sep.binom(n, pstar, se = 1, sp = 1)
```

Arguments

`n`	sample size = number of units tested (integer), scalar or vector
`pstar`	design prevalence as a proportion (scalar or vector of same length as n)
`se`	unit sensitivity of test (proportion), default = 1 (scalar or vector of same length as n)
`sp`	unit specificity of test (proportion), default = 1 (scalar or vector of same length as n)

Value

vector of population-level sensitivities

Examples

```
# examples for sep.binom - checked
sep.binom(n=300, pstar = 0.02, se = 0.92)
tested<- seq(10,100, by=10)
prev<- 0.05
sens<- 0.9
sep.binom(tested, prev, sens)
```

`sep.binom.imperfect` *Binomial population sensitivity for imperfect test*

Description

Calculates population sensitivity for a large or unknown population and allowing for imperfect test sensitivity and specificity, using Binomial distribution an allowing for a variable cut-point number of positives to classify as positive

Usage

```
sep.binom.imperfect(n, c = 1, se, sp = 1, pstar)
```

Arguments

`n`	sample size (scalar or vector)
`c`	The cut-point number of positives to classify a cluster as positive, default=1, if positives < c result is negative, >= c is positive (scalar or vector of same length as n)
`se`	test unit sensitivity (scalar or vector of same length as n)
`sp`	test unit specificity, default=1 (scalar or vector of same length as n)
`pstar`	design prevalence as a proportion (scalar or vector of same length as n)

Value

a vector of population-level sensitivities

Examples

```
# examples for sep.imperfect.binom
sep.binom.imperfect(1:10*5, 2, 0.95, 0.98, 0.1)
sep.binom.imperfect(50, 1:5, 0.95, 0.98, 0.1)
sep.binom.imperfect(30, 2, 0.9, 0.98, 0.1)
sep.binom.imperfect(30, 1, 0.9, 0.98, 0.1)
```

sep.exact *Population sensitivity for census (all units tested)*

Description

Calculates population sensitivity for detecting disease assuming imperfect test sensitivity, perfect test specificity and a census of all units in the population

Usage

```
sep.exact(d = 1, se = 1)
```

Arguments

d	expected number of infected units in population (=design prevalence*N rounded to next integer), scalar or vector of same length as se
se	unit sensitivity of test (proportion), scalar or vector

Value

vector of population-level sensitivities

Examples

```
# examples for sep.exact - checked
sep.exact(d=1, se = 0.92)
inf<- 1:5
sens<- 0.8
sep.exact(d=inf, se=sens)
sep.exact(se=0.8, d = ceiling(0.01*c(10, 50, 100, 250, 500)))
```

sep.freecalc *FreeCalc population sensitivity for imperfect test*

Description

Calculates population sensitivity for a finite population and allowing for imperfect test sensitivity and specificity, using Freecalc method

Usage

```
sep.freecalc(N, n, c = 1, se, sp = 1, pstar)
```

Arguments

`N`	population size (scalar)
`n`	sample size (scalar)
`c`	The cut-point number of positives to classify a cluster as positive, default=1, if positives < c result is negative, >= c is positive (scalar)
`se`	test unit sensitivity (scalar)
`sp`	test unit specificity, default=1 (scalar)
`pstar`	design prevalence as a proportion - assumed or target prevalence for detection of disease in the population (scalar)

Value

population-level sensitivity

Examples

```
# examples of sep.freecalc
sep.freecalc(150, 30, 2, 0.9, 0.98, 0.1)
sep.freecalc(150, 30, 1, 0.9, 0.98, 0.1)
```

`sep.hp` *Hypergeometric (HerdPlus) population sensitivity for imperfect test*

Description

Calculates population sensitivity for a finite population and allowing for imperfect test sensitivity and specificity, using Hypergeometric distribution

Usage

```
sep.hp(N, n, c = 1, se, sp = 1, pstar)
```

Arguments

`N`	population size (scalar or vector of same length as n)
`n`	sample size (scalar or vector)
`c`	The cut-point number of positives to classify a cluster as positive, default=1, if positives < c result is negative, >= c is positive (scalar)
`se`	test unit sensitivity (scalar)
`sp`	test unit specificity, default=1 (scalar)
`pstar`	design prevalence as a proportion (scalar)

Value

a vector of population-level sensitivities

Examples

```
# examples of sep.hp
sep.hp(150, 1:5*10, 2, 0.9, 0.98, 0.1)
sep.hp(150, 30, 2, 0.9, 0.98, 15)
sep.hp(150, 30, 1, 0.9, 0.98, 15)
sep.hp(150, 30, 1, 0.9, 0.98, 0.1)
```

---

sep.hypergeo *Hypergeometric Population sensitivity*

Description

Calculates population sensitivity for detecting disease, assuming imperfect test sensitivity, perfect test specificity and representative sampling, using hypergeometric approximation (assumes known population size)

Usage

```
sep.hypergeo(N, n, d, se = 1)
```

Arguments

N	population size, scalar or vector of same length as n
n	sample size (number tested), scalar or vector
d	expected number of infected units in population (=design prevalence*N rounded to next integer)
se	unit sensitivity of test (proportion), scalar or vector of same length as n

Value

a vector of population-level sensitivities

Examples

```
# examples for sep.hypergeo - checked
sep.hypergeo(N=100, n=50, d=1, se = 0.92)
inf<- 1:5
sens<- 0.8
sep.hypergeo(N=100, n=50, d=inf, se=sens)
N<- c(10, 50, 100, 250, 500)
sep.hypergeo(se=0.8, N=N, n=c(5, 25, 50, 125, 250), d = ceiling(0.01*N))
```

`sep.pfree` *Population sensitivity to achieve desired (posterior) probability of freedom*

Description

Calculates the population sensitivity required to achieve a given value for probability of disease freedom

Usage

```
sep.pfree(prior, pfree)
```

Arguments

`prior`	prior probability of freedom before surveillance (scalar or vector)
`pfree`	desired probability of freedom (scalar or vector)

Value

a vector of population-level sensitivities

Examples

```
# examples of sep.pfree
sep.pfree(0.5, 0.95)
sep.pfree(c(0.5, 0.6, 0.7, 0.8, 0.9, 0.95), 0.99)
sep.pfree(0.5, c(0.8, 0.9, 0.95, 0.99))
```

`sep.pooled` *Pooled population sensitivity*

Description

Calculates population sensitivity (sep) and population specificity (spp) assuming pooled sampling and allowing for imperfect sensitivity and specificity of the pooled test

Usage

```
sep.pooled(r, k, pstar, pse, psp = 1)
```

Arguments

`r`	number of pools sampled (scalar or vector)
`k`	pool size (scalar or vector of same length as r)
`pstar`	design prevalence (scalar or vector of same length as r)
`pse`	pool-level sensitivity (scalar or vector of same length as r)
`psp`	pool-level specificity (scalar or vector of same length as r)

Value

list of 2 elements, vector of sep values and vector of spp values

Examples

```
# examples for sep.pooled
sep.pooled(60, 5, 0.01, 1, 1)
sep.pooled(4, 10, 0.1, 0.9, 1)
sep.pooled(1:10*5, 5, 0.02, 0.9, 0.99)
sep.pooled(10, 5, 0.05, c(0.8, 0.9, 0.95, 0.99), 1)
```

`sep.prior` *Population sensitivity to achieve desired prior probability of freedom*

Description

Calculates the population sensitivity required to achieve a given value for the prior (discounted) probability of disease freedom

Usage

```
sep.prior(prior, p.intro)
```

Arguments

`prior`	prior probability of freedom before surveillance (scalar or vector)
`p.intro`	probability of introduction for time period (scalar or vector equal length to sep)

Value

a vector of population-level sensitivities

Examples

```
# examples of sep.prior
sep.prior(0.95, 0.01)
sep.prior(c(0.9, 0.95, 0.98, 0.99), 0.01)
sep.prior(0.95, c(0.001, 0.005, 0.01, 0.02, 0.05))
```

`sep.rb.bin`	*Binomial risk-based population sensitivity*

Description

Calculates risk-based population sensitivity with a single risk factor, using binomial method (assumes a large population), allows for unit sensitivity to vary among risk strata

Usage

```
sep.rb.bin(pstar, rr, ppr, n, se)
```

Arguments

`pstar`	design prevalence (scalar)
`rr`	relative risk values (vector of values corresponding to the number of risk strata)
`ppr`	population proportions corresponding to rr values (vector of equal length to rr)
`n`	sample size per risk category (vector same length as rr and ppr)
`se`	unit sensitivity, can vary among risk strata (fixed value or vector same length as rr, ppr, n)

Value

list of 3 elements, a scalar of population-level sensitivity a vector of EPI values and a vector of corresponding adjusted risks

Examples

```
# examples for sep.rb.bin
sep.rb.bin(0.1, c(5, 3, 1), c(0.1, 0.1, 0.8), c(5, 5, 5), 0.9)
sep.rb.bin(0.1, c(5, 1), c(0.1, 0.9), c(10, 5), c(0.95, 0.9))
sep.rb.bin(0.1, c(5, 1), c(0.1, 0.9), c(10, 5), c(0.9, 0.9))
sep.rb.bin(0.01, c(5, 1), c(0.1, 0.9), c(90, 50), c(0.9, 0.9))
```

`sep.rb.bin.varse`	*Binomial risk-based population sensitivity for varying unit sensitivity*

Description

Calculates population sensitivity for a single risk factor and varying unit sensitivity using binomial method (assumes large population)

Usage

```
sep.rb.bin.varse(pstar, rr, ppr, df)
```

Arguments

pstar	design prevalence (scalar)
rr	relative risk values (vector of values corresponding to the number of risk strata)
ppr	population proportions corresponding to rr values (vector of equal length to rr)
df	dataframe of values for each combination of risk stratum and sensitivity level, col 1 = risk group index, col 2 = unit Se, col 3 = n (sample size for that risk group and unit sensitivity)

Value

list of 3 elements, a scalar of population-level sensitivity a vector of EPI values and a vector of corresponding adjusted risks

Examples

```
# examples for sep.rb.bin.varse
rg<- c(1, 1, 2, 2)
se<- c(0.92, 0.85, 0.92, 0.85)
n<- c(80, 30, 20, 30)
df<- data.frame(rg, se, n)
sep.rb.bin.varse(0.01, c(5, 1), c(0.1, 0.9), df)

rg<- c(1, 1, 2, 2)
se<- c(0.95, 0.8, 0.95, 0.8)
n<- c(20, 10, 10, 5)
df<- data.frame(rg, se, n)
sep.rb.bin.varse(0.05, c(3, 1), c(0.2, 0.8), df)

rg<- c(rep(1, 30), rep(2, 15))
se<- c(rep(0.95, 20), rep(0.8, 10), rep(0.95, 10), rep(0.8, 5))
n<- rep(1, 45)
df<- data.frame(rg, se, n)
sep.rb.bin.varse(0.02, c(3, 1), c(0.2, 0.8), df)

rg<- c(1, 2, 3, 1, 2, 3)
se<- c(0.95, 0.95, 0.95, 0.8, 0.8, 0.8)
n<- c(20, 10, 10, 30, 5, 5)
df<- data.frame(rg, se, n)
sep.rb.bin.varse(0.01, c(5, 3, 1), c(0.1, 0.3, 0.6), df)
```

---

sep.rb.hypergeo *Hypergeometric risk-based population sensitivity*

Description

Calculates risk-based population sensitivity with a single risk factor, using the hypergeometric method (assuming a finite and known population size), allows for unit sensitivity to vary among risk strata

Usage

```
sep.rb.hypergeo(pstar, rr, N, n, se)
```

Arguments

pstar	design prevalence (scalar)
rr	relative risk values (vector of values corresponding to the number of risk strata)
N	Population size per risk category (vector same length as rr and ppr)
n	sample size per risk category (vector same length as rr and ppr)
se	unit sensitivity, can vary among risk strata (fixed value or a vector the same length as rr, ppr, n)

Value

list of 3 elements, a scalar of population-level sensitivity a vector of EPI values and a vector of corresponding adjusted risks

Examples

```
# examples for sep.rb.bin
sep.rb.hypergeo(0.1, c(5, 3, 1), c(10, 10, 80), c(5, 5, 5), 0.9)
sep.rb.hypergeo(0.1, c(5, 1), c(15, 140), c(10, 5), c(0.95, 0.9))
sep.rb.hypergeo(0.1, c(5, 1), c(23, 180), c(10, 5), c(0.9, 0.9))
sep.rb.hypergeo(0.01, c(5, 1), c(100, 900), c(90, 50), c(0.9, 0.9))
```

---

sep.rb.hypergeo.varse *Hypergeometric risk-based population sensitivity for varying unit sensitivity*

Description

Calculates population sensitivity for a single risk factor and varying unit sensitivity using hypergeometric approximation method (assumes known population size)

Usage

```
sep.rb.hypergeo.varse(pstar, rr, N, df)
```

Arguments

pstar	design prevalence (scalar)
rr	relative risk values (vector of values corresponding to the number of risk strata)
N	vector of population size for each risk group, corresponding to rr values (vector of equal length to rr)
df	dataframe of values for each combination of risk stratum and sensitivity level, col 1 = risk group index, col 2 = unit Se, col 3 = n (sample size for risk group and unit sensitivity)

Value

list of 5 elements, a scalar of population-level sensitivity a vector of EPI values, a vector of corresponding Adjusted risks a vector of sample sizes (n) per risk group and a vector of mean unit sensitivities per risk group

Examples

```
# examples for sep.rb.hypergeo.varse
rg<- c(1, 1, 2, 2)
se<- c(0.92, 0.85, 0.92, 0.85)
n<- c(80, 30, 20, 30)
df<- data.frame(rg, se, n)
sep.rb.hypergeo.varse(0.01, c(5, 1), c(200, 1800), df)

rg<- c(1, 1, 2, 2)
se<- c(0.95, 0.8, 0.95, 0.8)
n<- c(20, 10, 10, 5)
df<- data.frame(rg, se, n)
sep.rb.hypergeo.varse(0.05, c(3, 1), c(100, 400), df)

rg<- c(rep(1, 30), rep(2, 15))
se<- c(rep(0.95, 20), rep(0.8, 10), rep(0.95, 10), rep(0.8, 5))
n<- rep(1, 45)
df<- data.frame(rg, se, n)
sep.rb.hypergeo.varse(0.02, c(3, 1), c(100, 400), df)

rg<- c(1, 2, 3, 1, 2, 3)
se<- c(0.95, 0.95, 0.95, 0.8, 0.8, 0.8)
n<- c(20, 10, 10, 30, 5, 5)
df<- data.frame(rg, se, n)
sep.rb.hypergeo.varse(0.01, c(5, 3, 1), c(100, 300, 600), df)
```

sep.rb2.binom *Binomial risk-based population sensitivity for 2 risk factors*

Description

Calculates risk-based population sensitivity for two risk factors, using binomial method (assumes a large population)

Usage

```
sep.rb2.binom(pstar, rr1, ppr1, rr2, ppr2, n, se)
```

Arguments

pstar	design prevalence (scalar)
rr1	relative risks for first level risk factor (vector of values corresponding to the number of risk strata)

`ppr1`	population proportions for first level risk factor (vector of same length as rr1)
`rr2`	relative risks for second level risk factor, matrix, rows = levels of rr1, cols = levels of rr2
`ppr2`	population proportions for second level risk factor, matrix, rows = levels of rr1, cols = levels of rr2
`n`	matrix of number tested for each risk group (rows = levels of rr1, cols = levels of rr2)
`se`	test unit sensitivity (scalar)

Value

list of 4 elements, a scalar of population-level sensitivity a matrix of EPI values, a vector of corresponding Adjusted risks for the first risk factor and a matrix of adjusted risks for the second risk factor

Examples

```
# examples for sep.rb2.binom
pstar<- 0.01
rr1<- c(3, 1)
ppr1<- c(0.2, 0.8)
rr2<- rbind(c(4,1), c(4,1))
ppr2<- rbind(c(0.1, 0.9), c(0.3, 0.7))
se<- 0.8
n<- rbind(c(50, 20), c(20, 10))
sep.rb2.binom(pstar, rr1, ppr1, rr2, ppr2, n, se)
```

`sep.rb2.hypergeo` *Hypergeometric risk-based population sensitivity for 2 risk factors*

Description

Calculates risk-based population sensitivity for two risk factors, using hypergeometric approximation method (assumes a known population size)

Usage

```
sep.rb2.hypergeo(pstar, rr1, rr2, N, n, se)
```

Arguments

`pstar`	design prevalence (scalar)
`rr1`	relative risks for first level risk factor (vector of values corresponding to the number of risk strata)
`rr2`	relative risks for second level risk factor, matrix, rows = levels of rr1, cols = levels of rr2

`N`	matrix of population size for each risk group (rows = levels of rr1, cols = levels of rr2)
`n`	matrix of number tested (sample size) for each risk group (rows = levels of rr1, cols = levels of rr2)
`se`	test unit sensitivity (scalar)

Value

list of 6 elements, a scalar of population-level sensitivity a matrix of EPI values, a vector of corresponding Adjusted risks for the first risk factor and a matrix of adjusted risks for the second risk factor, a vector of population proportions for the first risk factor and a matrix of population proportions for the second risk factor

Examples

```
# examples for sep.rb2.hypergeo
pstar<- 0.01
rr1<- c(3, 1)
rr2<- rbind(c(4,1), c(4,1))
N<- rbind(c(100, 500), c(300, 1000))
n<- rbind(c(50, 20), c(20, 10))
se<- 0.8
sep.rb2.hypergeo(pstar, rr1, rr2, N, n, se)
```

`sep.sys` *2-stage population sensitivity*

Description

Calculates population-level (system) sensitivity for representative 2-stage sampling (sampling of clusters and units within clusters), assuming imperfect test sensitivity and perfect test specificity

Usage

```
sep.sys(H = NA, N = NA, n, pstar.c, pstar.u, se = 1)
```

Arguments

`H`	population size = number of clusters in the population, default = NA
`N`	population size within clusters, scalar or a vector of same length as n, default = NA
`n`	sample size (vector of number tested per cluster)
`pstar.c`	cluster (herd) level design prevalence, scalar, either proportion or integer
`pstar.u`	unit (animal) level design prevalence, scalar, either proportion or integer
`se`	unit sensitivity of test (proportion), scalar, default = 1

Value

list of 6 elements, 1) population level sensitivity, 2) vector of cluster-level sensitivities, 3) N, 4) n, 5) vector of design prevalences and 6) unit sensitivity

Note

if pstar.c is not a proportion N must be provided (and N>=n)

Examples

```
# examples for sep.sys - checked
H<- 500
N<- rep(1000, 150)
N[5]<- NA
n<- rep(30, 150)
pstar.u<- 0.1
pstar.c<- 0.01
se<- 0.98
sep.sys(H, N, n, pstar.c, pstar.u, se)
sep.sys(NA, N, n, 0.02, 0.05, 0.95)
N<- round(runif(105)*900+100)
n<- round(runif(105)*30+10)
sse<- sep.sys(1000, N, n, 0.02, 0.05, 0.9)
data.frame(N, n, sse[[2]])
```

---

sep.var.se *Population sensitivity for varying unit sensitivity*

Description

Calculates population-level sensitivity where unit sensitivity varies and using the appropriate method, depending on whether or not N provided (hypergeometric if N provided, binomial otherwise), assuming perfect test specificity and representative sampling

Usage

```
sep.var.se(N = NA, se, pstar)
```

Arguments

N	population size (number of units or clusters), N must be >= length(se)) or NA if unknown
se	vector of unit sensitivity values (proportion) for each unit sampled
pstar	specified design prevalence (scalar)

Value

a scalar of population-level sensitivity

Examples

```
# examples of sep.var.se - checked
sens<- c(rep(0.9, 50), rep(0.95, 100))
sep.var.se(NA, sens, 0.01)
sep.var.se(se=sens, pstar=0.01)
sep.var.se(N=500, sens, 0.01)
sep.var.se(NA, runif(150, 0.95, 0.99), 0.02)
sep.var.se(500, runif(150, 0.95, 0.99), 0.02)
```

`sp.parallel` *Specificity of tests in parallel*

Description

Calculates the combined specificity for multiple tests interpreted in parallel (assuming independence)

Usage

```
sp.parallel(sp)
```

Arguments

`sp`	vector of unit specificity values

Value

scalar of combined specificity, assuming independence

Examples

```
# examples for sp.parallel
sp.parallel(c(0.99, 0.95, 0.8))
```

`sp.series` *Specficity of tests in series*

Description

Calculates the combined specificity for multiple tests interpreted in series (assuming independence)

Usage

```
sp.series(sp)
```

Arguments

`sp`	vector of unit specificity values

Value

scalar of combined specificity, assuming independence

Examples

```
# examples for sp.series
sp.series(c(0.99, 0.95, 0.8))
```

`sph.binom` *Binomial population specificity for imperfect test*

Description

Calculates population specificity for a large or unknown population, using the Binomial distribution and adjusting for cut-point number of positives

Usage

```
sph.binom(n, c = 1, sp)
```

Arguments

`n`	sample size (scalar or vector)
`c`	The cut-point number of positives to classify a cluster as positive, default=1, if positives < c result is negative, >= c is positive (scalar or vector of same length as n)
`sp`	test unit specificity (scalar or vector of same length as n)

Value

a vector of population-level specificities

Examples

```
# examples for sph.imperfect.sp
sph.binom(30, 2, 0.98)
sph.binom(30, 1, 0.98)
sph.binom(1:5*10, 2, 0.98)
sph.binom(100, 1:5, 0.98)
sph.binom(100, 3, 95:100/100)
sph.binom(c(5, 10, 15, 20, 30, 50, 100, 200), 2, 0.98)
```

`sph.hp` *Hypergeometric population specificity calculation*

Description

Calculates population specificity for a finite population and imperfect test, using Hypergeometric distribution

Usage

```
sph.hp(N, n, c = 1, sp)
```

Arguments

`N`	population size (scalar or vector of same length as n)
`n`	sample size (scalar or vector)
`c`	The cut-point number of positives to classify a cluster as positive, default=1, if positives < c result is negative, >= c is positive (scalar or vector of same length as n)
`sp`	test unit specificity (scalar or vector of same length as n)

Value

a vector of population-level specificities

Examples

```
# examples of sph.hp
sph.hp(150, 30, 2, 0.98)
sph.hp(150, 30, 1, 0.98)
sph.hp(150, 1:5*10, 2, 0.98)
sph.hp(500, 30, 2, 95:100/100)
```

`spp` *Population specificity*

Description

Calculates population specificity assuming representative sampling

Usage

```
spp(n, sp)
```

Arguments

`n`	sample size (number tested), integer, scalar or vector
`sp`	unit specificity of test (proportion), scalar or vector of same length as n

Value

a vector of population-level specificities

Examples

```
# examples for spp - checked
spp(10, 0.9)
spp(c(10, 20, 50, 100), 0.99)
spp(100, c(0.999, 0.99, 0.98, 0.95, 0.9))
```

`sse.combined` *System sensitivity by combining multiple surveillance components*

Description

Calculates overall system sensitivity for multiple components, accounting for lack of independence (overlap) between components

Usage

```
sse.combined(C = NA, pstar.c, rr, ppr, sep)
```

Arguments

`C`	population sizes (number of clusters) for each risk group, NA or vector of same length as rr
`pstar.c`	cluster level design prevalence (scalar)
`rr`	cluster level relative risks (vector, length equal to the number of risk strata)
`ppr`	cluster level population proportions (optional), not required if C is specified (NA or vector of same length as rr)
`sep`	sep values for clusters in each component and corresponding risk group. A list with multiple elements, each element is a dataframe of sep values from a separate component, first column= clusterid, 2nd =cluster-level risk group index, 3rd col = sep

Value

list of 2 elements, a matrix (or vector if C not specified) of population-level (surveillance system) sensitivities (binomial and hypergeometric and adjusted vs unadjusted) and a matrix of adjusted and unadjusted component sensitivities for each component

Examples

```
# example for sse.combined (checked in excel combined components.xlsx)
C<- c(300, 1200)
pstar<- 0.01
rr<- c(3,1)
ppr<- c(0.2, 0.8)
comp1<- data.frame(id=1:100, rg=c(rep(1,50), rep(2,50)), cse=rep(0.5,100))
comp2<- data.frame(id=seq(2, 120, by=2), rg=c(rep(1,25), rep(2,35)), cse=runif(60, 0.5, 0.8))
comp3<- data.frame(id=seq(5, 120, by=5), rg=c(rep(1,10), rep(2,14)), cse=runif(24, 0.7, 1))
sep<- list(comp1, comp2, comp3)
sse.combined(C, pstar, rr, sep = sep)
sse.combined(C=NA, pstar, rr, ppr, sep = sep)
```

---

sse.rb.2stage *Two-stage risk-based system sensitivity*

Description

Calculates system sensitivity for 2 stage risk-based sampling, llowing for a single risk factor at each stage and using either binomial or hypergeometric approxiation

Usage

```
sse.rb.2stage(C = NA, pstar.c, pstar.u, rr.c, ppr.c, rr.u, ppr.u, N = NA, n,
  rg, se)
```

Arguments

`C`	Population size (number of clusters), NA = unknown (default)
`pstar.c`	cluster level design prevalence (scalar)
`pstar.u`	unit level design prevalence (scalar)
`rr.c`	cluster level relative risks (vector with length corresponding to the number of risk strata), use rr.c = c(1,1) if risk factor does not apply
`ppr.c`	cluster level population proportions for risk categories (vector), NA if no cluster level risk factor
`rr.u`	unit level relative risks (vector with length corresponding to the number of risk strata), use rr.u = c(1,1) if risk factor does not apply
`ppr.u`	unit level population proportions for each risk group (optional) matrix, 1 row for each cluster, columns = unit level risk groups, not required if N is provided
`N`	population size per risk group for each cluster, NA or matrix of N for each risk group for each cluster, N=NA means cluster sizes not provided
`n`	sample size per risk group for each cluster sampled, matrix, 1 row for each cluster, columns = unit level risk groups
`rg`	vector of cluster level risk group (index) for each cluster
`se`	unit sensitivity for each cluster, scalar or vector of values for each cluster, equal in length to n

Value

list of 2 elements, a scalar of population-level (surveillance system) sensitivity and a vector of cluster-level sensitivities

Examples

```
# examples for sse.rb.2stage
pstar.c<- 0.02
pstar.u<- 0.1
rr.c<- c(5, 1)
ppr.c<- c(0.1, 0.9)
rr.u<- c(3, 1)
se<- 0.9
n<- cbind(rep(10, 50), rep(5, 50))
rg<- c(rep(1, 30), rep(2, 20))
ppr.u<- cbind(rep(0.2, 50), rep(0.8, 50))
N<- cbind(rep(30, 50), rep(120, 50))
C<- 500
sse.rb.2stage(C=NA, pstar.c, pstar.u, rr.c, ppr.c, rr.u, ppr.u, N=NA, n, rg, se)
sse.rb.2stage(C, pstar.c, pstar.u, rr.c, ppr.c, rr.u, ppr.u, N=NA, n, rg, se)
sse.rb.2stage(C=NA, pstar.c, pstar.u, rr.c, ppr.c, rr.u, ppr.u, N, n, rg, se)
sse.rb.2stage(C, pstar.c, pstar.u, rr.c, ppr.c, rr.u, ppr.u, N, n, rg, se)
```

---

tp *True prevalence*

Description

Estimates true prevalence and confidence limits for given sample size and result, according to specified method

Usage

```
tp(x, n, se, sp, type = "blaker", conf = 0.95)
```

Arguments

x	number of positive units (scalar)
n	sample size (no. units sampled) (scalar)
se	test sensitivity (scalar)
sp	test specificity (scalar)
type	method for estimating CI, one of c("normal", "c-p", "sterne", "blaker", "wilson", "all")
conf	desired level of confidence for CI, default = 0.95 (scalar)

Value

list with 2 elements, a matrix of apparent prevalence and lower and upper confidence limits and a matrix of true prevalence and lower and upper confidence limits using the chosen method(s)

Examples

```
# examples for tp
x<- 20
n<- 120
se<- 0.9
sp<- 0.99
conf<- 0.95
tp(x, n, se, sp, "all")
tp(x, n, se, sp, "c-p")
tp(x, n, 0.95, 0.9, "c-p")
```

`tp.normal` *Normal approximation confidence limits for true prevalence*

Description

Estimates true prevalence and confidence limits for estimates based on normal approximation

Usage

```
tp.normal(x, n, se, sp, conf = 0.95)
```

Arguments

`x`	number of positive results in sample (scalar or vector)
`n`	sample size (scalar or vector)
`se`	test unit sensitivity (scalar or vector)
`sp`	test unit specificity (scalar or vector)
`conf`	desired level of confidence for CI, default = 0.95 (scalar or vector)

Value

list with 2 elements, a matrix of apparent prevalence and wilson lower and upper confidence limits and a matrix of true prevalence and normal approximation lower and upper confidence limits

Examples

```
# examples for tp.normal
tp.normal(25, 120, 0.9, 0.99)
tp.normal(seq(5, 25, by=5), 120, 0.9, 0.99)
```

Index

Important formulae for surveillance

Evan Sergeant
evan@ausvet.com.au
AusVet Animal Health Services
July 2012
http://epitools.ausvet.com.au/docs/Important_formulae_for_surveillance.pdf

Representative surveillance for disease Freedom

Terminology:

In this document some specific terminology relating to unit, cluster and population values have been used to try and simplify the formulae presented. These terms are explained here:

*A **unit** is either an individual (animal, plant, etc) as part of a cluster, or a cluster as part of a larger population.*

*A **cluster** is a grouping level of individual units (animals, plants, fish, etc) at a higher level such as a herd, flock, tank, pen, farm, etc. Clusters usually are only considered at one level, but can occur at multiple levels (for example pens within farms within districts).*

***Unit sensitivity** is the sensitivity at the unit level for a particular analysis. For cluster-level analyses, unit sensitivity is the sensitivity of the test (or combination of tests) used, whereas for population-level analyses unit sensitivity is the cluster-level sensitivity for clusters sampled.*

***Population sensitivity** is a sensitivity calculated at some population or grouping level. Depending on the context, the population can be either a cluster of multiple individuals or a larger population comprising multiple clusters.*

***Component sensitivity** is a population-level sensitivity, usually at a country or regional level, calculated for one part (component) of a surveillance system which comprises multiple separate components or activities.*

***System sensitivity** is a population-level sensitivity, usually at a country or regional level, calculated from one or more components*

Population sensitivity and Sample size

Binomial	Population sensitivity: $SeP = 1 - (1 - SeU \times P^*_U)^n$ $= 1 - \prod (1 - SeU_i \times P^*_U)$ where SeU varies among units Sample size: $n = \log(1 - SeP)/\log(1 - SeU \times P^*_U)$	Assumes: • sampling with replacement or sample small (<10%) relative to population • Specificity = 100%

Hypergeometric approximation	Population sensitivity: $SeP = 1 - (1 - SeU \times n/N)^d$ $SeP = 1 - (1 - SeU_{avg} \times n/N)^d$ where SeU varies among units Sample size: $n = (N/SeU)*(1 - (1 - SeP)^{1/(P^* \times N)})$	Assumes: • Sampling without replacement or where sample size is large relative to population • Specificity = 100%
Exact	Population sensitivity: $SeP = 1 - (1 - SeU)^d$ $SeP = 1 - (1 - SeU_{avg})^d$ where SeU varies among units	Assumes: • Sampling of the entire population • Specificity = 100%

Negative Predictive Value (or confidence of population freedom: PFree)

$PFree = (1 - PrInf)/(1 - PrInf \times SeP)$ or
$= PriorPFree/(1 - SeP \times (1 - PriorPFree))$

assuming specificity = 100%.

Revising confidence of freedom in successive time periods

$PFree_t = 1 - [1 - PFree_{t-1} + PIntro_t - ((1 - PFree_{t-1}) \times PIntro_t)]$

Equilibrium PFree

Maximum or minimum stable value for PFree for given combinations of SeP and PIntro

$PFree_{equ} = (1 - (PIntro / SeP)) / (1 - PIntro)$

Maximum or minimum value for PriorPFree (after discounting) for given combinations of SeP and PIntro

$PriorPFree_{equ} = 1 - (PIntro / SeP)$

Design prevalence to achieve specified population sensitivity

Where cluster size is unknown (binomial):

$P^*_U = (1 - \exp((\log(1 - SeP))/n))/SeU$

Where cluster size is known (hypergeometric approximation):

$P^*_U = \log(1 - SeP)/\log(1 - SeU \times n/N)/N$

Population sensitivity required to achieve desired PFree

SeP = (1 - PriorPFree/PFree)/(1 - PriorPFree)

where PFree is the target value and PriorPFree is the current prior value.

Population sensitivity required to stay above specified threshold PFree

SeP = PIntro/(1 – Target PFree)

Combining test sensitivities in series

(For example in a diagnostic process with multiple steps)

$SeU_{combined} = \prod Se_i$

Combining component sensitivities in parallel, assuming independence

Calculates system sensitivity from multiple components, assuming independence (no overlap between units sampled) between components, for example different compartments or different clusters represented in the surveillance system.

$SeP = 1 - \prod (1 - CSe_i)$

Updating cluster sensitivities between components where there is overlap

This assumes no independence between components, for example where the same clusters (herds or flocks, etc) are represented in multiple surveillance system components. The probability of infection for each cluster is adjusted between components and resulting component sensitivities are then combined as for assuming independence. For this example binomial calculations are used, but hypergeometric or exact could also be used if appropriate:

Method 1: Adjusting effective probability of infection between components:

1. Calculate SeC for each cluster [$SeC = 1 - (1 - SeU \times P^*)^n$] for each component.
2. Calculate posterior confidence of freedom and hence posterior probability of infection for each cluster for the first component (component order is a matter of convenience):

   $PFree_c = (1 - P^*)/(1 - P^*_c \times SeC)$ where P^*_c is the cluster-level design prevalence

   $PostPInf_c = (1 - PFree_c)$
3. Calculate probability that each cluster has a negative test result and hence component sensitivity (CSe) for first component:

   $P(Neg) = 1 - P^*_c \times SeC$

   $CSe = 1 - \prod (P(Neg))$
4. Calculate P(Neg) for each cluster and CSe for the second component after substituting $PostPInf_h$ instead of P* in formula:

   $P(Neg) = 1 - PostPInf_h \times SeC_2$

$CSe = 1 - \prod (P(Neg))$

5. Repeat for as many components as necessary
6. Clusters start with P* at the first component in which they appear and then get updated as necessary
7. When all component sensitivities have been calculated, calculate overall system sensitivity (probability that one or more components will yield a positive result if the population is infected at the design prevalence), using independence formula.

   $SSe = 1 - \prod (1 - CSe_i)$

Method 2: Aggregating data between components:

An alternative (often simpler) approach is to aggregate the data for each cluster to calculate single SeC values and then combine these values to calculate overall system sensitivity:

$SeC = 1 - \prod((1 - P^* \times SeU_i)\text{^}n_i)$

For where SeU_i and n_i are test sensitivity and sample size for each of the i components in the surveillance system.

Key:

Abbreviation/symbol	Meaning
n, N	Sample size and corresponding population size
d	Number of diseased elements in population
t	Time period
P^*_U	Unit level design prevalence (individual or cluster)
Se	Test sensitivity
SeU	Unit level sensitivity (test sensitivity when calculating cluster/herd-level sensitivity or cluster/herd-level sensitivity when calculating population or component sensitivity)
SeP	Population sensitivity (can be cluster level or overall population level)
SeC	Cluster sensitivity
SeC_i	Cluster sensitivity for the i-th cluster
SeU_{avg}	Average unit sensitivity across all units (individuals or clusters) sampled
CSe_i	Component sensitivity for the i-th surveillance system component
SSe	System sensitivity
PFree	Confidence of population freedom (= negative predictive value)
PriorPFree	Confidence of population freedom before undertaking current surveillance
PrInf	Prior probability of being infected = 1 – prior confidence of freedom
PostPInf	Posterior probability of being infected = 1 – posterior confidence of freedom (NPV)

Risk-based freedom surveillance

Adjusted risk and effective probability of infection

$AR_L = 1/(RR \times PPr_H + PPr_L)$
$AR_H = RR \times AR_L$

or for multiple risk levels:

$AR_i = RR_i / \sum(RR \times PPr)$

$EPI = P^* \times AR$ (for respective risk categories)
$EPI > 1$ is invalid – design prevalence and/or relative risk should be revised to ensure $EPI < 1$.

Population sensitivity for simple, 1-stage, no risk factors, one factor affecting sensitivity

Assuming large population relative to sample size (binomial) and only two unit sensitivity values:

$$SeP = 1- (1 - P^* \times SeU_H)^{n(h)} \times (1 - P^* \times SeU_L)^{n(l)}$$

n(h) and n(l) are sample sizes for high and low sensitivity groups, respectively; or

assuming small population:

$$SeP = 1 - (1 -SeU_{avg} \times n/N)^d$$

Sample size for simple, 1-stage, one risk factor (2 levels), constant sensitivity

$USe = EPI_H \times SeU_H \times SPr_H + EPI_L \times SeU_L \times SPr_L$

$n = \log(1 - SeP)/\log(1 - USe)$

SeU_H and SeU_L are the mean values for SeU for high and low risk groups respectively.

Population sensitivity for simple, 1-stage, one risk factor, one factor affecting sensitivity

$$SeP = 1- (1 - EPI_H \times SeU_H)^{n(hh)} \times (1 - EPI_H \times SeU_L)^{n(hl)} \times (1 - EPI_L \times SeU_H)^{n(lh)} \times (1 - EPI_L \times SeU_L)^{n(ll)}$$

n(hh), n(hl),n(lh) and n(ll) are sample sizes for high risk & high sensitivity, high risk & low sensitivity, low risk & high sensitivity and low risk & low sensitivity groups, respectively.

Sample size for simple, 1-stage, one risk factor, one factor affecting sensitivity

$LRSe = SPr_{LH} \times SeU_{H} + (1 - SPr_{LH}) \times SeU_{L}$

$HRSe = SPr_{HH} \times SeU_{H} + (1 - SPr_{HH}) \times SeU_{L}$

$USe = EPI_{H} \times HRSe \times SPr_{H} + EPI_{L} \times LRSe \times SPr_{L}$

$n = \log(1 - SeP)/\log(1 - USe)$

LRSe, HRSe are weighted average sensitivity in low and high risk samples respectively.

USe is the probability of a single randomly selected animal from the sample being positive, given the population is infected at the design prevalence.

SPr_{H}, SPr_{L}, SPr_{LH} , SPr_{HH} are proposed sample proportions from the high-risk sub-population, low-risk sub-population, high sensitivity group in high-risk sub-population and high sensitivity group in low-risk sub-population respectively.

Key:

See also key for representative freedom surveys

Abbreviation/symbol	**Meaning**
RR	Relative risk
AR	Adjusted risk
PPr_{H}, PPr_{L}	Population proportions in high and low risk groups, respectively
SPr_{H}, SPr_{L}	The proportion of the surveillance sample from the respective risk group
EPI, EPI_{H}, EPI_{L}	Effective probability of infection and EPI in high and low risk groups. Probabilities of infection after adjusting design prevalence for group relative risks
SeU_{H}, SeU_{L}	Sensitivity in high and low risk groups, respectively. May be test (animal) sensitivity or herd-sensitivity, depending on level at which being calculated.
USe	The probability of a single randomly selected animal from the surveillance sample being positive, given the population is infected at the design prevalence.

Prevalence estimation

Apparent or seroprevalence

(assumes perfect test sensitivity and specificity)

Estimated prevalence: $P = x/n$

Asymptotic (normal approximation) confidence intervals:

$$CI = P \pm Z\sqrt{((P(1 - P)/n)}$$

Alternative (binomial, Wilson binomial) CI methods usually better, particularly as P approaches 0 or 100%.

Sample size: $n = (Z^2 \times P(1 - P))/e^2$

Assumes a large population. Where expected sample size is large (10%) relative to populations size use following adjustment:

$$n_{adj} = (N \times n)/(N + n)$$

Estimated true prevalence

(allows adjustment for imperfect sensitivity and specificity)

$$TP = (AP + SP - 1)/(Se + Sp - 1)$$

Note: Method fails when $Se + Sp = 1$ due to division by 0.
TP may be negative if $AP + Sp < 1$ (Sp estimate is lower than suggested by the results).

Asymptotic (normal approximation) confidence intervals assuming known sensitivity and specificity :

$$CI = TP \pm Z\sqrt{[AP(1 - AP)/(n \times (Se + Sp - 1)^2)]}$$

Assumes Se and Sp known exactly (no uncertainty).
Lower CI may be <0 if TP is close to 0.

Sample size:

$$n = (Z/e)^2 \times (Se \times TP + (1 - Sp) \times (1 - TP)) \times (1 - Se \times TP - (1 - Sp)\times(1 - TP))/(Se + Sp - 1)^2$$

Asymptotic (normal approximation) confidence intervals assuming uncertain sensitivity and specificity :

$$CI = TP \pm Z\sqrt{[AP \times (1\text{-}AP)/(n \times (Se + Sp - 1)^2) + (Se \times (1\text{-}Se) \times TP^2)/(M \times (Se + Sp - 1)^2) + (Sp \times (1\text{-}Sp)*(1\text{-}TP)^2)/(R*(Se + Sp - 1)^2)]}$$

Key:

Abbreviation/symbol	**Meaning**
n, N	Sample size and corresponding population size (animal level)
P	Observed or expected prevalence (proportion)
x	Number of units with the characteristic of interest
Z	Z distribution value corresponding to desired confidence level Z = 1.96 for 95%, 2.58 for 99% and 1.64 for 90%
e	Desired precision of estimate (± relative to estimate). Confidence interval width = 2e
n_{adj}	Sample size adjusted for small population
TP	True prevalence estimate
AP	Apparent prevalence estimate
Se, Sp	Sensitivity and specificity of the test used
CI	Confidence interval
M	Sample size for estimating test sensitivity
R	Sample size for estimating test specificity