Package: PoolTestR (via r-universe)

October 18, 2026

Title Prevalence and Regression for Pool-Tested (Group-Tested) Data

Version 0.2.1

Description An easy-to-use tool for working with presence/absence tests on 'pooled' or 'grouped' samples. The primary application is for estimating prevalence of a marker in a population based on the results of tests on pooled specimens. This sampling method is often employed in surveillance of rare conditions in humans or animals (e.g. molecular xenomonitoring). The package was initially conceived as an R-based alternative to the molecular xenomonitoring software, 'PoolScreen' <https://sites.uab.edu/statgenetics/software/>. However, it goes further, allowing for estimates of prevalence to be adjusted for hierarchical sampling frames, and perform flexible mixed-effect regression analyses (McLure et al. Environmental Modelling and Software. <DOI:10.1016/j.envsoft.2021.105158>). The package is currently in early stages, however more features are planned or in the works: e.g. adjustments for imperfect test specificity/sensitivity, functions for helping with optimal experimental design, and functions for spatial modelling.

License GPL (>= 3)

URL https://github.com/AngusMcLure/PoolTestR

BugReports https://github.com/AngusMcLure/PoolTestR/issues

Depends R (>= 3.4.0)

Imports brms, dplyr, lme4 (>= 1.1-35.1), methods, progress, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rlang, rstan (>= 2.26.0), rstantools (>= 2.3.1.1), stats, stringr, tibble

Suggests covr, testthat (>= 3.0.0)

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.26.0), StanHeaders (>= 2.26.0)

Biarch true

Encoding UTF-8

Language en-AU

LazyData true

RoxygenNote 7.3.2

SystemRequirements GNU make

Config/testthat/edition 3

Config/pak/sysreqs cmake make libicu-dev

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

Date/Publication 2025-11-25 08:25:17 UTC

RemoteUrl https://github.com/angusmclure/pooltestr

RemoteRef HEAD

RemoteSha 8de4f3e5dcf34a6bb2e5d6ffb76eea72a3478db2

Contents

CheckInputData 2
ExampleData 4
getPrevalence 6
HierPoolPrev 9
PoolLink 12
PoolPrev 14
PoolReg 17
PoolRegBayes 19
PrepareClusterData 21
SimpleExampleData 24
TruePrev 24

Index **26**

`CheckInputData` *Checking input data for errors and inconsistencies*

Description

Test the input data is adequately formatted for use with `PoolPrev()` and `HierPoolPrev()`.

Usage

```
CheckInputData(data, result, poolSize, ...)
```

Arguments

data	A data.frame with one row for each pooled sampled and columns for the size of the pool (i.e., the number of specimens / isolates / insects pooled to make that particular pool), the result of the test of the pool. It may also contain additional columns with additional information (e.g. location where pool was taken) which can optionally be used for stratifying the data into smaller groups and calculating prevalence by group (e.g. calculating prevalence for each location).
result	The name of column with the result of each test on each pooled sample. The result must be stored with 1 indicating a positive test result and 0 indicating a negative test result.
poolSize	The name of the column with number of specimens/isolates/insects in each pool.
...	Optional name(s) of columns with variables to stratify the data by.

Value

Returns data invisibly, using invisible(x)

This function is used to check the input data for formatting problems including:

- Incorrect class of the input data
- incorrect class of result and poolSize columns
- Missing columns
- Missing values in rows
- Invalid values in rows

If any problems are detected, an error or warning will be raised describing the issue.

See Also

PrepareClusterData, SimpleExampleData, PoolPrev, HierPoolPrev

Examples

```
# Check whether the SimpleExampleData is formatted
# appropriately for estimating prevalence in
# PoolTestR
SimpleExample_output <-
  CheckInputData(
    data = SimpleExampleData,
    result = "Result", poolSize = "NumInPool"
  )
# No errors/warnings were raised
identical(SimpleExample_output, SimpleExampleData)
# The hierarchical scheme is formatted properly so
# the output is identical to the input


## Not run:
  # Error raised when input data is not class data.frame
```

```
CheckInputData(
  data = 1,
  result = "Result", poolSize = "NumInPool"
)
# Error raised when result/poolSize column names are incorrect
CheckInputData(
  data = SimpleExampleData,
  result = "WrongResultName", poolSize = "WrongNumInPoolName"
)
# Error raised when optional stratifying variable column names are incorrect
CheckInputData(
  data = SimpleExampleData,
  result = "Result", poolSize = "NumInPool",
  "WrongRegionName", "WrongYearName"
)
# Error raised when Result/poolSize columns are not numeric/integer
CheckInputData(
  data = SimpleExampleData %>%
    mutate(Result = as.character(.data$Result),
           .keep = "all"),
  result = "Result", poolSize = "NumInPool"
)
CheckInputData(
  data = SimpleExampleData %>%
    mutate(NumInPool = as.character(.data$NumInPool),
           .keep = "all"),
  result = "Result", poolSize = "NumInPool"
)
# Error raised when Result column values are not numeric 0 and 1
CheckInputData(
  data = SimpleExampleData %>%
    mutate(Result = 2,
           .keep = "all"),
  result = "Result", poolSize = "NumInPool"
)
# Error raised when poolSize column values are not positive
CheckInputData(
  data = SimpleExampleData %>%
    mutate(NumInPool = (-1*.data$NumInPool),
           .keep = "all"),
  result = "Result", poolSize = "NumInPool"
)

## End(Not run)
```

ExampleData *A synthetic dataset for pooled testing*

Description

A synthetic dataset mimicking a realistic hierarchical sampling frame. Simulated samples are taken from across three regions (A, B, and C) in which the vectors have a low (0.5%), medium (2%), and high (4%) prevalence of the marker of interest. Ten villages are chosen within each region, and traps are placed at ten sites within each village. Every site is sampled once a year over three years (0, 1, and 2). Prevalence is not uniform within each region or over time. At baseline (year 0), prevalence varies between villages within each region around the mean for the region, and prevalence varies between sites within each village around the mean for the village. Consequently though the prevalence is different for each site, two sites within the same village are likely to have a more similar prevalence than two sites in different villages, or two sites in different regions. On average the prevalence is declining over time (odds ratio of 0.8 per year), however, the growth rate varies between villages. Consequently two sites in different villages with similar prevalence at baseline may have different prevalence by the third year, and prevalence may go up in some villages. Each year the traps at each site catch a negative binomial number (mean 200, dispersion 5) of vectors. The catch size at each site and year is independent. Each year, the catches at each site are pooled into groups of 25 with an additional pool for any remainder (e.g. a catch of 107 vectors will be pooled into 4 pools of 25 and one pool of 7). Test results on each pool are simulated assuming the test has perfect sensitivity and specificity.

Usage

```
ExampleData
```

Format

A data frame with 6 variables:

NumInPool Number of specimens in pool. Range = 1:25

Region ID of the region the pool was taken from. "A", "B", or "C"

Village ID of village that pool was taken from. Includes name of region e.g. "B-3" is village 3 from region B

Site ID of site that pool was taken from. Includes name of region and village e.g. "B-3-7" is site 7 from village 3 from region B

Result Result of test on pool; 0 = negative, 1 = positive

Year Year of sampling. Years are 0, 1, or 2

Details

The 'true' model can be summarised in formula notation as:

Result ~ Region + Year + (1+Year|Village) + (1|Site)

where the coefficient for Year is log(0.8), the standard deviation for intercept random effects for village and site are both 0.5, the standard deviation for the year random effect for village is 0.2 and the random effects are all uncorrelated/independent.

`getPrevalence` *Predicting Prevalence from a Mixed or Fixed Effect Logistic Regression with Presence/Absence Tests on Pooled Samples*

Description

This function works somewhat like a `predict` or `fitted` generic function returning the model predicted prevalence for a given set of data; however, as the quantity of interest (prevalence) is neither on the response or link scale we do not use either of these generic functions. Further, when the model accounts for the hierarchical structure of the sampling frame (e.g. Region/Village/Site), it is common to want to know the predicted values at each level of sampling (e.g. Prevalence at each region, village or site) so these are calculated automatically. Also to calculate population-level prevalence from a mixed model, random/group effects need to marginalised out to avoid biased estimates. This is performed automatically.

Usage

```
getPrevalence(model, ...)

## S3 method for class 'glm'
getPrevalence(model, newdata = NULL, level = 0.95, ...)

## S3 method for class 'glmerMod'
getPrevalence(
  model,
  newdata = NULL,
  re.form = NULL,
  all.negative.pools = "zero",
  ...
)

## S3 method for class 'brmsfit'
getPrevalence(
  model,
  newdata = NULL,
  re.form = NULL,
  robust = TRUE,
  level = 0.95,
  all.negative.pools = "zero",
  summary = TRUE,
  ...
)
```

Arguments

`model`	An object returned by [PoolReg()] or [PoolRegBayes()]
`...`	Arguments passed to methods for each class

`newdata` The data for which prevalence needs to be estimated/predicted. If not provided, defaults to using the data used to train the model (i.e. returns the fitted values of the prevalence)

`level` Defines the confidence level to be used for the confidence and credible intervals. Defaults to 0.95 (i.e. 95% intervals).

`re.form` A description of which random effects to include in the prediction. If omitted, an attempt is made to infer from model and data structure.

`all.negative.pools` The kind of point estimate and interval to use when all pools are negative. Typically ignored unless newdata is NULL. If `'zero'` (default), uses 0 as the point estimate and lower bound for the interval and `level` posterior quantile the upper bound of the interval. If `'consistent'`, result is the same as for the case where at least one pool is positive.

`robust` Logical. Option when model class is `brmsfit`. If `TRUE` (default) the point estimate of prevalence is the posterior median. If `FALSE`, the the posterior mean is used instead.

`summary` If `TRUE` (default), output is point estimate and interval for prevalence. If `FALSE`, outputs returns prevalence for each draw from the posterior of the supplied model

Details

If `re.form` is omitted (probably the most common use case) `getPrevalence` will test to see if there are any random effect terms in the model formula extracted from the `model` object. If not, it just returns the estimates based on population effects. If there are random effects, it tests to see if the random effect variables form a nested hierarchical structure in the data provided. If so, in addition to the estimates based on population effects only, it will estimate at different levels of the nested hierarchical structure in order of increasing granularity. For manual control you can set to NA for population effects only, or a one-sided formula specifying the form of the random effects to include in estimates, or a list of such objects. Any random effects omitted will be marginalised out. For automatically detected nested hierarchical structures this means that higher level estimates marginalise over lower-level random effect; in particular, population level estimates will marginalise over all random effects.

Value

A `list` with at least one field `PopulationEffects` and an additional field for every random/group effect variable. The field `PopulationEffects` contains a `data.frame` with the prevalence estimated based only the fixed/population effects. When the intercept is the only fixed/population effect, this is just the population mean (possibly adjusted for random/group effects). When there are group effects terms, `getPrevalence` attempts to order these with respect to 'granularity' and extract the prevalence estimates for these random effects; e.g. if the random/group effects included are there to account for a hierarchical sampling frame with levels 'Village' and 'Site' with a formula like `Result ~ Cov1 + Cov2 + (1|Village) + (1|Site)`, then getPrevalence will be a list of three data frames: estimates for every combination of covariates, estimates for every combination of covariates and village, and estimates for every combination of covariates, village, and site.

See Also

PrepareClusterData, PoolReg, PoolRegBayes

Examples

```
# Perform logistic-type regression modelling for a synthetic dataset consisting
# of pools (sizes 1, 5, or 10) taken from 4 different regions and 3 different
# years. Within each region specimens are collected at 4 different villages,
# and within each village specimens are collected at 8 different sites.


### Models in a frequentist framework
#ignoring hierarchical sampling frame within each region
Mod <- PoolReg(Result ~ Region + Year,
               data = SimpleExampleData,
               poolSize = NumInPool)
summary(Mod)

#accounting hierarchical sampling frame within each region
HierMod <- PoolReg(Result ~ Region + Year + (1|Village) + (1|Site),
                   data = SimpleExampleData,
                   poolSize = NumInPool)
summary(HierMod)


### Models in a Bayesian framework with default (non-informative) priors
#ignoring hierarchical sampling frame within each region

  BayesMod <- PoolRegBayes(Result ~ Region + Year,
                           data = SimpleExampleData,
                           poolSize = NumInPool)
  summary(BayesMod)

  #we could also account for hierarchical sampling frame within each region but
  #note that this is more complex and slower)

  # BayesHierMod <- PoolRegBayes(Result ~ Region + Year + (1|Village) + (1|Site),
  #                              data = SimpleExampleData,
  #                              poolSize = NumInPool)


### Calculate adjusted estimates of prevalence
# We use the same function for all four models, but the outputs are slightly different

#For models without hierarchical sampling structure there is an estimate of
#prevalence for every combination of population (fixed) effects: e.g. Region and
#Year
getPrevalence(Mod) #Frequentist model

  getPrevalence(BayesMod) #Bayesian model
```

```
#For models without hierarchical sampling structure, there is a prevalence
#estimate for each combination of region and year and then at each level of the
#hierarchical sampling frame (i.e. for each village in each region and each site
#in each village)
getPrevalence(HierMod)

# You can also use getPrevalence to predict prevalence for other values of the
# covariates (e.g. predict prevalence in year 4 based on linear trend on the
# logit scale)

#Making a data frame containing data make predictions on
DataFuture <- unique(data.frame(Region = SimpleExampleData$Region,
                                Village = SimpleExampleData$Village,
                                Site = SimpleExampleData$Site,
                                Year = 4))

getPrevalence(Mod, newdata = DataFuture)
getPrevalence(HierMod, newdata = DataFuture)
```

`HierPoolPrev` *Estimation of prevalence based on presence/absence tests on pooled samples in a hierarchical sampling frame. Uses an intercept-only random effects model to model prevalence at population level. See PoolReg and PoolRegBayes for full mixed-effect modelling*

Description

Estimation of prevalence based on presence/absence tests on pooled samples in a hierarchical sampling frame. Uses an intercept-only random effects model to model prevalence at population level. See PoolReg and PoolRegBayes for full mixed-effect modelling

Usage

```
HierPoolPrev(
  data,
  result,
  poolSize,
  hierarchy,
  ...,
  prior = NULL,
  robust = TRUE,
  level = 0.95,
  verbose = FALSE,
  cores = NULL,
  iter = 2000,
  warmup = iter/2,
  chains = 4,
  control = list(adapt_delta = 0.9),
  all.negative.pools = "zero"
)
```

Arguments

`data`	A `data.frame` with one row for each pooled sampled and columns for the size of the pool (i.e. the number of specimens / isolates / insects pooled to make that particular pool), the result of the test of the pool. It may also contain additional columns with additional information (e.g. location where pool was taken) which can optionally be used for splitting the data into smaller groups and calculating prevalence by group (e.g. calculating prevalence for each location)
`result`	The name of column with the result of each test on each pooled sample. The result must be stored with 1 indicating a positive test result and 0 indicating a negative test result.
`poolSize`	The name of the column with number of specimens/isolates/insects in each pool
`hierarchy`	The name of column(s) indicating the group membership, ordered from largest to smallest. In a nested sampling design with multiple levels of grouping, the lower-level groups must have names/numbers that differentiate them from all other groups at the same level. E.g. If sampling was performed at 200 sites across 10 villages (20 site per village), then there should be 200 unique names for the sites. If, for instance, the sites are instead numbered 1 to 20 within each village, the village identifier (e.g. A, B, C...) should be combined with the site number to create unique identifiers for each site (e.g. A-1, A-2... for sites in village A and B-1, B-2... for the sites in village B etc.)
`...`	Optional name(s) of columns with variables to stratify the data by. If omitted the complete dataset is used to estimate a single prevalence. If included prevalence is estimated separately for each group defined by these columns
`prior`	List of parameters specifying the parameters for the the priors on the population intercept and standard deviations of group-effect terms. See details.
`robust`	Logical. If `TRUE` (default), the point estimate of prevalence is the posterior median. If `FALSE`, the posterior mean is used instead.
`level`	The confidence level to be used for the confidence and credible intervals. Defaults to 0.95 (i.e. 95% intervals)
`verbose`	Logical indicating whether to print progress to screen. Defaults to false (no printing to screen)
`cores`	The number of CPU cores to be used. By default one core is used
`iter`, `warmup`, `chains`	MCMC options for passing onto the sampling routine. See stan for details.
`control`	A named list of parameters to control the sampler's behaviour. Defaults to default values as defined in stan, except for `adapt_delta` which is set to the more conservative value of 0.9. See stan for details.
`all.negative.pools`	The kind of point estimate and interval to use when all pools are negative (Bayesian estimates only). If `'zero'` (default), uses 0 as the point estimate and lower bound for the interval and `level` posterior quantile the upper bound of the interval. If `'consistent'`, result is the same as for the case where at least one pool is positive.

Details

When using the default value of the `prior` argument (NULL), the model uses the following prior: `list(intercept = list(nu = 3, mu = 0, sigma = 4.0), group_sd = list(nu = 3, mu = 0, sigma = 2.5), individual_sd = FALSE)` This models the prior of the linear scale intercept as t-distributed with parameters in 'intercept' and the standard deviation of the group-level effects as truncated (non-negative) t-distribution. 'individual_sd = FALSE' means that this prior is for the root-sum-square of group-effect standard deviations for models with multiple grouping levels. The default implies a prior on population prevalence that is approximately distributed as beta(0.5,0.5). To set custom priors, use the same nested list format. Any omitted parameters will be replaced with the default values and additional parameters ignored silently. For example, to change the parameters to be equal to the defaults for intercept-only random-effect model in PoolRegBayes you can use: `list(individual_sd = TRUE)`, which puts a prior on each the standard deviations of each of group-level effects separately, but doesn't change the priors used.

Value

An object of class `HierPoolPrevOutput`, which inherits from class `tbl`. The output includes the following columns:

- `PrevBayes` – the (Bayesian) posterior expectation
- `CrILow` and `CrIHigh` – lower and upper bounds for credible intervals
- `NumberOfPools` – number of pools
- `NumberPositive` – the number of positive pools
- `ICC` – the estimated intra-cluster correlation coefficient. Matrix column containing one column for each variable included in the `hierarchy`.
- `ICC_CrILow` and `ICC_CrIHigh` – lower and upper bounds for credible intervals of the estimated ICC. Matrix columns containing one column for each variable included in the `hierarchy`.

The `hierarchy` input must have the name of column(s) indicating the group membership ordered from largest to smallest. For example, the hierarchy for the `SimpleExampleData` dataset included in PoolTestR is `hierarchy = c("Region", "Village", "Site")` or `hierarchy = c("Village", "Site")` (depending on the analysis structure).

If grouping variables are provided in `...` there will be an additional column for each grouping variable. When there are no grouping variables (supplied in `...`) then the output has only one row with the prevalence estimates for the whole dataset. When grouping variables are supplied, then there is a separate row for each group.

We provide two functions to check input data prior to prevalence estimation. To test the data for formatting errors, use the function `CheckInputData()`. To test the nesting of the hierarchy/clustering scheme, use the function `PrepareClusterData()`.

The custom print method summarises the output data frame by representing output variables with credible intervals (i.e., `PrevBayes`, `ICC`) as a single column in the form `"X (CrILow - CrIHigh)"` where `X` is the variable, `CrILow` is the lower credible interval and `CrIHigh` is the upper credible interval. In the print method, prevalence `PrevBayes` is represented as a percentage (i.e., per 100 units).

See Also

`CheckInputData`, `PrepareClusterData`, `PoolPrev`, `getPrevalence`

Examples

```
# Calculate prevalence for a synthetic dataset consisting of pools (sizes 1, 5,
# or 10) taken from 3 different years. Specimens are collected at 16 different
# villages, and within each village specimens are collected at 8 different
# sites.


  #Prevalence for each year:
  #ignoring hierarchical sampling frame be:
  PoolPrev(SimpleExampleData, Result, NumInPool, Year)
  #accounting hierarchical sampling frame within each region
  HierPoolPrev(SimpleExampleData, Result, NumInPool, c("Village","Site"), Year)
```

`PoolLink` *Link Function for Logistic Regression with Presence/Absence Tests on Pooled Samples*

Description

A custom link function for the `binomial` family to be used with `glm`

Usage

```
PoolLink(PoolSize = 1)
```

Arguments

`PoolSize`	The number of specimens/isolates/insects in each pool. When used with glm, the length must either be 1 if all the pools are the same size, but the same length as the data otherwise

Value

An object of class `link-glm`

Examples

```
# Perform logistic-type regression modelling for a synthetic dataset consisting
# of pools (sizes 1, 5, or 10) taken from 4 different regions and 3 different
# years. Within each region specimens are collected at 4 different villages,
# and within each village specimens are collected at 8 different sites.


### Models in a frequentist framework
#ignoring hierarchical sampling frame within each region
Mod <- PoolReg(Result ~ Region + Year,
```

```
               data = SimpleExampleData,
               poolSize = NumInPool)
summary(Mod)

#accounting hierarchical sampling frame within each region
HierMod <- PoolReg(Result ~ Region + Year + (1|Village) + (1|Site),
                   data = SimpleExampleData,
                   poolSize = NumInPool)
summary(HierMod)


### Models in a Bayesian framework with default (non-informative) priors
#ignoring hierarchical sampling frame within each region

  BayesMod <- PoolRegBayes(Result ~ Region + Year,
                           data = SimpleExampleData,
                           poolSize = NumInPool)
  summary(BayesMod)

  #we could also account for hierarchical sampling frame within each region but
  #note that this is more complex and slower)

  # BayesHierMod <- PoolRegBayes(Result ~ Region + Year + (1|Village) + (1|Site),
  #                              data = SimpleExampleData,
  #                              poolSize = NumInPool)


### Calculate adjusted estimates of prevalence
# We use the same function for all four models, but the outputs are slightly different

#For models without hierarchical sampling structure there is an estimate of
#prevalence for every combination of population (fixed) effects: e.g. Region and
#Year
getPrevalence(Mod) #Frequentist model

  getPrevalence(BayesMod) #Bayesian model


#For models without hierarchical sampling structure, there is a prevalence
#estimate for each combination of region and year and then at each level of the
#hierarchical sampling frame (i.e. for each village in each region and each site
#in each village)
getPrevalence(HierMod)

# You can also use getPrevalence to predict prevalence for other values of the
# covariates (e.g. predict prevalence in year 4 based on linear trend on the
# logit scale)

#Making a data frame containing data make predictions on
DataFuture <- unique(data.frame(Region = SimpleExampleData$Region,
                                Village = SimpleExampleData$Village,
                                Site = SimpleExampleData$Site,
                                Year = 4))
```

```
getPrevalence(Mod, newdata = DataFuture)
getPrevalence(HierMod, newdata = DataFuture)
```

PoolPrev *Estimation of prevalence based on presence/absence tests on pooled samples*

Description

Estimation of prevalence based on presence/absence tests on pooled samples

Usage

```
PoolPrev(
  data,
  result,
  poolSize,
  ...,
  bayesian = TRUE,
  prior = NULL,
  robust = TRUE,
  level = 0.95,
  all.negative.pools = "zero",
  reproduce.poolscreen = FALSE,
  verbose = FALSE,
  cores = NULL,
  iter = 2000,
  warmup = iter/2,
  chains = 4,
  control = list(adapt_delta = 0.98)
)
```

Arguments

data	A `data.frame` with one row for each pooled sampled and columns for the size of the pool (i.e. the number of specimens / isolates / insects pooled to make that particular pool), the result of the test of the pool. It may also contain additional columns with additional information (e.g. location where pool was taken) which can optionally be used for stratifying the data into smaller groups and calculating prevalence by group (e.g. calculating prevalence for each location)
result	The name of column with the result of each test on each pooled sample. The result must be stored with 1 indicating a positive test result and 0 indicating a negative test result.
poolSize	The name of the column with number of specimens/isolates/insects in each pool

`...` Optional name(s) of columns with variables to stratify the data by. If omitted the complete dataset is used to estimate a single prevalence. If included, prevalence is estimated separately for each group defined by these columns

`bayesian` Logical indicating whether Bayesian calculations should be calculated. If TRUE (the default) calculates frequentist and Bayesian estimates of prevalence, otherwise only calculates frequentist estimates (MLE and likelihood ratio confidence intervals).

`prior` Prior for prevalence, ignored if `bayesian == FALSE`. If NULL (the default) the prior for the prevalence is the uninformative Jeffrey's prior. The only alternative prior is a possibly zero-inflated beta distribution. Zero inflation allows for some prior (and posterior) probability that the marker of interest is totally absent from the population. The parameters for this are specified with a list with three numeric non-negative entries named alpha, beta, and absent. For instance, a uniform prior with no probability of true absence can be specified as `prior = list(alpha = 1, beta = 1, absent = 0`.

`robust` Logical. If `TRUE` (default), the point estimate of prevalence is the posterior median. If `FALSE`, the posterior mean is used instead. Applies to Bayesian estimates only and therefore ignored if `bayesian = FALSE`.

`level` Defines the confidence level to be used for the confidence and credible intervals. Defaults to 0.95 (i.e. 95% intervals)

`all.negative.pools` The kind of point estimate and interval to use when all pools are negative (Bayesian estimates only). If `'zero'` (default), uses 0 as the point estimate and lower bound for the interval and `level` posterior quantile the upper bound of the interval. If `'consistent'`, result is the same as for the case where at least one pool is positive. Applies to Bayesian estimates only and therefore ignored if `bayesian == FALSE`.

`reproduce.poolscreen` (defaults to FALSE). If TRUE this changes the way that likelihood ratio confidence intervals are computed to be somewhat wider and more closely match those returned by Poolscreen. We recommend using the default (FALSE). However setting to TRUE can help to make comparisons between PoolPrev and Poolscreen.

`verbose` Logical indicating whether to print progress to screen. Defaults to false (no printing to screen). Ignored if `bayesian == FALSE`.

`cores` The number of CPU cores to be used. By default one core is used. Ignored if `bayesian == FALSE`.

`iter`, `warmup`, `chains` MCMC options for passing onto the sampling routine. See stan for details. Ignored if `bayesian == FALSE`.

`control` A named list of parameters to control the sampler's behaviour. Defaults to default values as defined in stan, except for `adapt_delta` which is set to the more conservative value of 0.98. See stan for details. Ignored if `bayesian == FALSE`.

Value

An object of class `PoolPrevOutput`, which inherits from class `tbl`. The output includes the following columns:

- PrevMLE – (the Maximum Likelihood Estimate of prevalence)
- CILow and CIHigh - lower and upper confidence intervals using the likelihood ratio method
- PrevBayes – the (Bayesian) posterior expectation. Omitted if bayesian == FALSE.
- CrILow and CrIHigh – lower and upper bounds for credible intervals. Omitted if bayesian == FALSE.
- ProbAbsent – the posterior probability that prevalence is exactly 0 (i.e. disease marker is absent). NA if using default Jeffrey's prior or if prior$absent == 0. Omitted if bayesian == FALSE.
- NumberOfPools – number of pools
- NumberPositive – the number of positive pools

If grouping variables are provided in ... there will be an additional column for each grouping variable. When there are no grouping variables, the output has only one row with the prevalence estimates for the whole dataset. When grouping variables are supplied in ..., then there is a separate row for each group.

We provide a function to check input data prior to prevalence estimation. To test the data for formatting errors, use the function CheckInputData().

The custom print method summarises the output data frame by representing the prevalence and credible intervals as a single column in the form "Prev (CLow - CHigh)" where Prev is the prevalence, CLow is the lower confidence/credible interval and CHigh is the upper confidence/credible interval. In the print method, prevalence is represented as a percentage (i.e., per 100 units)

See Also

CheckInputData HierPoolPrev, getPrevalence

Examples

```
#Try out on a synthetic dataset consisting of pools (sizes 1, 5, or 10) taken
#from 4 different regions and 3 different years. Within each region specimens
#are collected at 4 different villages, and within each village specimens are
#collected at 8 different sites.

# Start by calculate frequentist estimates only (much faster)

#Prevalence across the whole (synthetic) dataset
PoolPrev(SimpleExampleData, Result, NumInPool, bayesian = FALSE)
#Prevalence in each Region
PoolPrev(SimpleExampleData, Result, NumInPool, Region, bayesian = FALSE)
#Prevalence for each year
PoolPrev(SimpleExampleData, Result, NumInPool, Year, bayesian = FALSE)
#Prevalence for each combination of region and year
PoolPrev(SimpleExampleData, Result, NumInPool, Region, Year, bayesian = FALSE)


#Prevalence across the whole (synthetic) dataset, also including Bayesian Estimates - slower
PoolPrev(SimpleExampleData, Result, NumInPool)
```

PoolReg *Frequentist Mixed or Fixed Effect Logistic Regression with Presence/Absence Tests on Pooled Samples*

Description

It can be useful to do mixed effects logistic regression on the presence/absence results from pooled samples, however one must adjust for the size of each pool to correctly identify trends and associations. This can done by using a custom link function [PoolTestR::PoolLink()], defined in this package, in conjunction with using `glm` from the `stats` package (fixed effect models) or `glmer` from the `lme4` package (mixed effect models).

Usage

```
PoolReg(formula, data, poolSize, link = "logit", ...)
```

Arguments

`formula`	A `formula` of the kind used to define models in `lme4`, which are generalisation of the formulae used in `lm` or `glm` that allow for random/group effects. The left-hand side of the formula should be the name of column in `data` with the result of the test on the pooled samples. The result must be encoded with 1 indicating a positive test result and 0 indicating a negative test result.
`data`	A `data.frame` with one row for each pooled sampled and columns for the size of the pool (i.e. the number of specimens / isolates / insects pooled to make that particular pool), the result of the test of the pool and any number of columns to be used as the dependent variables in the logistic regression
`poolSize`	The name of the column with number of specimens/isolates/insects in each pool
`link`	link function. There are two options ''logit'' (logistic regression, the default) and ''cloglog'' (complementary log log regression).
`...`	Arguments to be passed on to `stats::glm` or `lme4::glmer` e.g. weights

Value

An object of class `glmerMod` (or `glm` if there are no random/group effects)

See Also

`getPrevalence`, `PoolRegBayes`

Examples

```
# Perform logistic-type regression modelling for a synthetic dataset consisting
# of pools (sizes 1, 5, or 10) taken from 4 different regions and 3 different
# years. Within each region specimens are collected at 4 different villages,
# and within each village specimens are collected at 8 different sites.
```

```
### Models in a frequentist framework
#ignoring hierarchical sampling frame within each region
Mod <- PoolReg(Result ~ Region + Year,
               data = SimpleExampleData,
               poolSize = NumInPool)
summary(Mod)

#accounting hierarchical sampling frame within each region
HierMod <- PoolReg(Result ~ Region + Year + (1|Village) + (1|Site),
                   data = SimpleExampleData,
                   poolSize = NumInPool)
summary(HierMod)


### Models in a Bayesian framework with default (non-informative) priors
#ignoring hierarchical sampling frame within each region

  BayesMod <- PoolRegBayes(Result ~ Region + Year,
                           data = SimpleExampleData,
                           poolSize = NumInPool)
  summary(BayesMod)

  #we could also account for hierarchical sampling frame within each region but
  #note that this is more complex and slower)

  # BayesHierMod <- PoolRegBayes(Result ~ Region + Year + (1|Village) + (1|Site),
  #                              data = SimpleExampleData,
  #                              poolSize = NumInPool)


### Calculate adjusted estimates of prevalence
# We use the same function for all four models, but the outputs are slightly different

#For models without hierarchical sampling structure there is an estimate of
#prevalence for every combination of population (fixed) effects: e.g. Region and
#Year
getPrevalence(Mod) #Frequentist model

  getPrevalence(BayesMod) #Bayesian model


#For models without hierarchical sampling structure, there is a prevalence
#estimate for each combination of region and year and then at each level of the
#hierarchical sampling frame (i.e. for each village in each region and each site
#in each village)
getPrevalence(HierMod)

# You can also use getPrevalence to predict prevalence for other values of the
# covariates (e.g. predict prevalence in year 4 based on linear trend on the
# logit scale)

#Making a data frame containing data make predictions on
```

```
DataFuture <- unique(data.frame(Region = SimpleExampleData$Region,
                                Village = SimpleExampleData$Village,
                                Site = SimpleExampleData$Site,
                                Year = 4))

getPrevalence(Mod, newdata = DataFuture)
getPrevalence(HierMod, newdata = DataFuture)
```

`PoolRegBayes` *Bayesian Mixed or Fixed Effect Logistic Regression with Presence/Absence Tests on Pooled Samples*

Description

It can be useful to do mixed effects logistic regression on the presence/absence results from pooled samples, however one must adjust for the size of each pool to correctly identify trends and associations.

Usage

```
PoolRegBayes(
  formula,
  data,
  poolSize,
  link = "logit",
  prior = NULL,
  cores = NULL,
  ...
)
```

Arguments

`formula`	A `formula` of the kind used to define models in `brms`, which are generalisation of the formulae used in `lm`,`glm` or `lme4`. The left-hand side of the formula should be the name of column in `data` with the result of the test on the pooled samples. The result must be stored with 1 indicating a positive test result and 0 indicating a negative test result.
`data`	A `data.frame` with one row for each pooled sampled and columns for the size of the pool (i.e. the number of specimens / isolates / insects pooled to make that particular pool), the result of the test of the pool and any number of columns to be used as the dependent variables in the logistic regression.
`poolSize`	The name of the column with number of specimens / isolates / insects in each pool.
`link`	Link function. There are three options 'logit' (i.e logistic regression, the default), 'cloglog' (complementary log-log), and 'loglogit'. The final option blends a log link function and the logit function so that parameters are (log) prevalence/rate ratios as long as predicted prevalence is <0.8 (for details see Clark and Barr, Stat Methods Med Res (2018) <DOI:10.1177/0962280217698174>)

prior	The priors to be used for the regression parameters. Defaults to a non-informative (normal(0,100)) prior on linear coefficients and a zero-truncated student-t prior on the group effect standard deviations. Custom priors must be brmsprior objects produced by brms::set_prior
cores	The number of CPU cores to be used. By default one core is used
...	Additional arguments to be passed to brms::brms.

Value

An object of class brms with the regression outputs.

References

Clark RG, Barr M: A blended link approach to relative risk regression. Statistical Methods in Medical Research 2018, 27(11):3325-3339. <DOI:10.1177/0962280217698174>

Angus McLure, Ben O'Neill, Helen Mayfield, Colleen Lau, Brady McPherson (2021). PoolTestR: An R package for estimating prevalence and regression modelling for molecular xenomonitoring and other applications with pooled samples. Environmental Modelling & Software, 145:105158. <DOI:10.1016/j.envsoft.2021.105158>

See Also

PoolReg, getPrevalence

Examples

```
# Perform logistic-type regression modelling for a synthetic dataset consisting
# of pools (sizes 1, 5, or 10) taken from 4 different regions and 3 different
# years. Within each region specimens are collected at 4 different villages,
# and within each village specimens are collected at 8 different sites.


### Models in a frequentist framework
#ignoring hierarchical sampling frame within each region
Mod <- PoolReg(Result ~ Region + Year,
               data = SimpleExampleData,
               poolSize = NumInPool)
summary(Mod)

#accounting hierarchical sampling frame within each region
HierMod <- PoolReg(Result ~ Region + Year + (1|Village) + (1|Site),
                   data = SimpleExampleData,
                   poolSize = NumInPool)
summary(HierMod)


### Models in a Bayesian framework with default (non-informative) priors
#ignoring hierarchical sampling frame within each region

  BayesMod <- PoolRegBayes(Result ~ Region + Year,
```

```
                          data = SimpleExampleData,
                          poolSize = NumInPool)
  summary(BayesMod)

  #we could also account for hierarchical sampling frame within each region but
  #note that this is more complex and slower)

  # BayesHierMod <- PoolRegBayes(Result ~ Region + Year + (1|Village) + (1|Site),
  #                              data = SimpleExampleData,
  #                              poolSize = NumInPool)


### Calculate adjusted estimates of prevalence
# We use the same function for all four models, but the outputs are slightly different

#For models without hierarchical sampling structure there is an estimate of
#prevalence for every combination of population (fixed) effects: e.g. Region and
#Year
getPrevalence(Mod) #Frequentist model

  getPrevalence(BayesMod) #Bayesian model


#For models without hierarchical sampling structure, there is a prevalence
#estimate for each combination of region and year and then at each level of the
#hierarchical sampling frame (i.e. for each village in each region and each site
#in each village)
getPrevalence(HierMod)

# You can also use getPrevalence to predict prevalence for other values of the
# covariates (e.g. predict prevalence in year 4 based on linear trend on the
# logit scale)

#Making a data frame containing data make predictions on
DataFuture <- unique(data.frame(Region = SimpleExampleData$Region,
                                Village = SimpleExampleData$Village,
                                Site = SimpleExampleData$Site,
                                Year = 4))

getPrevalence(Mod, newdata = DataFuture)
getPrevalence(HierMod, newdata = DataFuture)
```

`PrepareClusterData` *Prepare hierarchical/clustered survey data for analysis*

Description

Helper function to prepare data for analysis with `HierPoolPrev()`.

Usage

```
PrepareClusterData(data, result, poolSize, hierarchy = NULL, ...)
```

Arguments

data	A data.frame with one row for each pooled sampled and columns for the size of the pool (i.e. the number of specimens / isolates / insects pooled to make that particular pool), the result of the test of the pool. It may also contain additional columns with additional information (e.g. location where pool was taken) which can optionally be used for stratifying the data into smaller groups and calculating prevalence by group (e.g. calculating prevalence for each location)
result	The name of column with the result of each test on each pooled sample. The result must be stored with 1 indicating a positive test result and 0 indicating a negative test result.
poolSize	The name of the column with number of specimens/isolates/insects in each pool
hierarchy	The name of column(s) indicating the group membership, ordered from largest to smallest.
...	Optional name(s) of columns with variables to stratify the data by.

Details

When including information about hierarchical sampling frame (e.g. where units were sampled from sites and sites were selected from a broader region), it is critical that each cluster can be uniquely identified. It's not enough for the combination of columns specifying the hierarchy to be unique. Each different location needs to have a unique label. It's not enough for the combination of columns specifying the hierarchy to be unique. Each different location needs to have a unique label.

This function is provided to assist users by detecting certain types of nesting and clustering issues within data. As we do not make assumptions about the hierarchical/clustering scheme, we cannot guarantee that this function will detect all errors in hierarchical/clustering schemes. This function checks for the most common type of error, which is multiple different locations (e.g., collection sites) having the same value within a hierarchical/clustering column.

Value

An object of class data.frame, identical to the input data. If there were issues with nesting inside the hierarchy, the output will have a single additional column PoolTestR_ID, containing unique identifier for each location in the survey (created by concatenating the hierarchy column values within each row).

This function can be used to check levels of the hierarchical/clustering scheme. For example, the SimpleExampleData has the scheme Region > Village > Site. The full geographical hierarchy/clustering scheme can be tested using hierarchy = c("Region", "Village", "Site"). This function can also be used to check only the levels of the hierarchical/clustering scheme that will be included in prevalence estimates, e.g., if planning to stratify SimpleExampleData by Region, the hierarchy will be hierarchy = c("Village", "Site").

Functions in PoolTools do not make assumptions about the number of levels present or the names of hierarchical columns. They can be applied in any cases where a hierarchical sampling frame is involved.

See Also

CheckInputData, HierPoolPrev, getPrevalence, SimpleExampleData

Examples

```
# Check whether the SimpleExampleData is nested
# appropriately for estimating prevalence in
# HierPoolPrev()
SimpleExample_output <-
  PrepareClusterData(
    data = SimpleExampleData,
    result = "Result", poolSize = "NumInPool",
    hierarchy = c("Region", "Village", "Site")
  )
# No errors/warnings were raised
identical(SimpleExample_output, SimpleExampleData)
# The hierarchical scheme is formatted properly so
# the output is identical to the input


## Not run:
  # Checking another example data set for clustering issues
  # Create a test data frame that has incorrectly nested
  # Village and Site variables
  check_data <- data.frame(
    Region = rep(c("A", "B"), each = 4),
    Village = rep(rep(c("W", "X"), each = 2), 2),
    Site = c(1:4, 4:1),
    Year = rep(0, 8),
    NumInPool = rep(10, 8),
    Result = c(rep(0, 8))
  )
  # Test whether the data.frame is formatted appropriately
  # for HierPoolPrev()
  check_output <- PrepareClusterData(
    data = check_data,
    result = "Result", poolSize = "NumInPool",
    hierarchy = c("Region", "Village", "Site")
  )
  # New column has been added with unique identifier for
  # each location
  check_output$PoolTestR_ID

## End(Not run)
```

SimpleExampleData *A synthetic dataset for pooled testing*

Description

The simple synthetic dataset consisting of pools (sizes 1, 5, or 10) taken from 4 different regions and 3 different years. Within each region specimens are collected at 4 different villages, and within each village specimens are collected at 8 different sites.

Usage

```
SimpleExampleData
```

Format

A data frame with 1152 rows and 6 variables:

NumInPool Number of specimens in pool. Takes values 1, 5, or 10.

Region ID of the region the pool was taken from. "A", "B", "C", or "D"

Village ID of village pool was taken from. Includes name of region e.g. "B-3" is village 3 from region B

Site ID of site pool was taken from. Includes name of region and village e.g. "B-3-7" is site 7 from village 3 from region B

Result Result of test on pool; 0 = negative, 1 = positive

Year Year of sampling. Years are 0, 1, or 2

TruePrev *A synthetic dataset for pooled testing*

Description

This `data.frame` contains the 'true' values of prevalence for each, site, village, region and year used to generate the synthetic dataset `ExampleData`

Usage

```
TruePrev
```

Format

A data frame with 900 rows and 7 variables:

Region ID of the region the pool was taken from. "A", "B", or "C"

Village ID of village pool was taken from. Includes name of region e.g. "B-3" is village 3 from region B

Site ID of sampling site pool was taken from. Includes name of region and village e.g. "B-3-7" is site 7 from village 3 from region B

Year Year of sampling. Years are 0, 1, or 2

PrevalenceRegion 'True' average prevalence in the region (in that year)

PrevalenceVillage 'True' average prevalence in the village (in that year)

PrevalenceSite 'True' prevalence at that site (in that year)

Details

The 'true' model can be summarised in formula notation as:

Result ~ Region + Year + (1|Village) + (0 + Year|Village) + (1|Site)

where the coefficient for Year is log(0.8), the standard deviation for intercept random effects for village and site are both 0.5, the standard deviation for the year random effect for village is 0.2 and the random effects are all uncorrelated/independent.

Index

∗ **datasets**
ExampleData, 4
SimpleExampleData, 24
TruePrev, 24

binomial, *12*
brms::set_prior, *20*

CheckInputData, 2, *11*, *16*, *23*

ExampleData, 4

getPrevalence, 6, *11*, *16, 17*, *20*, *23*
glm, *12*

HierPoolPrev, *3*, 9, *16*, *23*

PoolLink, 12
PoolPrev, *3*, *11*, 14
PoolReg, *8*, 17, *20*
PoolRegBayes, *8*, *17*, 19
PrepareClusterData, *3*, *8*, *11*, 21

SimpleExampleData, *3*, *23*, 24
stan, *10*, *15*

TruePrev, 24