

PooledInfRate

Brad J. Biggerstaff

2022-03-15

Introduction

Applications of *pooled* or *group testing* include two general aims: estimation of the population individual-level probability of positivity; and cost-efficient schemes of identification of positive individuals. The *PooledInfRate* package is concerned only with estimation applications. Functionality includes point and interval estimation of a proportion from binary (*positive* / *negative*) samples that have been pooled or grouped before assessment of positivity, which is determined only at the pool level. Estimation applications date back at least to Chiang and Reeves (1962), who developed asymptotic likelihood-based methods for pools of equal size in application to estimation of virus infection in mosquito collections. These authors also extended their analyses to two different pool sizes under a simplifying assumption to make the mathematical development tractable, but they were constrained from further extension by computational tools available at the time. The package name “PooledInfRate” is based on the use of “infection rate” in entomological applications and follows the author’s development of the Microsoft Excel™ (Microsoft Corporation (n.d.)) add-in of the same name (<https://www.cdc.gov/westnile/resourcepages/mosqSurvSoft.html>).

The computational functionality of routines in this package for point estimation are the standard maximum likelihood estimate (MLE); the bias corrected estimate based on the methods of Gart (see Gart (1991)), as implemented by Hepworth (2005); the bias-reduced estimate based on Firth’s correction (see Firth (1993) and Hepworth and Biggerstaff (2017)); and the traditional *minimum infection rate* (MIR) estimate used in entomology. Confidence intervals (CI) included are the score, skewness-corrected score, and bias-and-skewness-corrected score intervals following Gart (Gart (1991)) (see Hepworth 2005); the interval obtained by inverting the likelihood ratio test; the Wald interval; and an interval based on the MIR (see Hepworth (2005) and Biggerstaff (2008) for evaluations of these intervals). Also included are routines to compute point and CI estimates for the differences of proportions estimated from pooled samples, as detailed in Biggerstaff (2008).

To complement applications involving entomological surveillance for pathogens in disease vectors such as mosquitoes, the package provides routines to compute the *Vector Index* (Fauver et al. (2015)), a measure incorporating both population size or density and of vector infection prevalences (“infection rate”). Its computation is outlined below.

The methods mentioned so far assume that perfect tests or assays are used in testing the pools for “positivity.” Extending this, functionality is included for point estimation when using an imperfect test with known sensitivity and specificity. Following Hepworth and Biggerstaff (2021), methods based both on Firth’s correction (Firth (1993)) and on the standard MLE are included. There is currently a lack of theoretical work in this area for CI estimation in the case of imperfect tests, so none are presently included; future theoretical and corresponding package development in this area is anticipated.

Data structure

A variety of data structures may be used with estimation functions in *PooledInfRate*. As with R functionality in general, either individual vector objects or data.frame objects supply the data to the functions. The data must have at a minimum a recording of the number of positive pools, x , and the pool sizes, m , corresponding to those pools. A variable giving the number of pools, n , corresponding to those records may also be specified, and when none is given, the values for x are assumed to be 0/1 and each n is assumed to be 1 by default. To

carry out the same analyses for multiple groups at once, a grouping variable, either as a factor or a numeric vector that will be treated as a factor may be used. Illustrations for acceptable data formats are

```
x <- c(1,0,0,0)
m <- c(50,25,10,5)
```

and

```
ex1.dat <- data.frame(Pos = c(1,0,0,0), PoolSize = c(50,25,10,5))
```

```
ex1.dat
```

```
#>   Pos PoolSize
#> 1   1      50
#> 2   0      25
#> 3   0      10
#> 4   0       5
```

and

```
ex2.dat <- data.frame(Pos = c(2,1,0,1),
                      PoolSize = c(50,25,10,5),
                      NumPools = c(10,10,5,2),
                      Location = c("A","A","B","B"))
```

```
ex2.dat
```

```
#>   Pos PoolSize NumPools Location
#> 1   2      50      10      A
#> 2   1      25      10      A
#> 3   0      10       5      B
#> 4   1       5       2      B
```

and if there are different groups, say Species,

```
mosq.dat <- data.frame(Zone = c(1,1,2,2, 1,1,1,2,2, 1,1,2),
                      Species = rep(c("Culex pipiens","Culex tarsalis", "Culex
→ quinquefasciatus"),
                                   c(4,5,3)),
                      Pos = c(1,0,0,1, 1,0,1,0,0, 0,1,0),
                      PoolSize = c(100,50,25,5, 200,100,50,25,10, 25,10,5),
                      Nights = c(5,5,3,3,5,5,5,3,3,5,5,3))
```

```
mosq.dat
```

```
#>   Zone Species Pos PoolSize Nights
#> 1   1   Culex pipiens  1    100     5
#> 2   1   Culex pipiens  0     50     5
#> 3   2   Culex pipiens  0     25     3
#> 4   2   Culex pipiens  1      5     3
#> 5   1   Culex tarsalis  1    200     5
#> 6   1   Culex tarsalis  0    100     5
#> 7   1   Culex tarsalis  1     50     5
#> 8   2   Culex tarsalis  0     25     3
#> 9   2   Culex tarsalis  0     10     3
#> 10  1 Culex quinquefasciatus  0     25     5
#> 11  1 Culex quinquefasciatus  1     10     5
#> 12  2 Culex quinquefasciatus  0      5     3
```

Other data may naturally be included in data frames, and we will include these data sets and needed extensions of these data sets in certain examples below.

One-sample estimation (perfect test)

Function interfaces The functions `pooledBin` (“pooled binary”) provides the interface to both point and CI estimates, and there are both default and formula interfaces provided. The function `pIR` (“pooled infection rate”) is a copy of `pooledBin` with a slightly different, and shorter name merely for convenience; in all the examples below, `pooledBin` and `pIR` can be used interchangeably. For single samples, the default methods are straightforward, with printed output being the point estimate and computed CI:

```
library(PooledInfRate)

pooledBin(x,m)
#>      P  Lower  Upper
#> 0.0126 0.0007 0.0783

with(ex2.dat, pooledBin(Pos, PoolSize, NumPools))
#>      P  Lower  Upper
#> 0.0052 0.0017 0.0126

with(mosq.dat, pIR(Pos, PoolSize))
#>      P  Lower  Upper
#> 0.0120 0.0048 0.0289
```

When there is a grouping variable, as with the `mosq.dat` data set, an optional `group=` may be used to obtain individual results for each group:

```
with(mosq.dat, pooledBin(Pos, PoolSize, group = Species))
#>      Group   PointEst      Lower      Upper
#> 1      Culex pipiens 0.013370328 0.002778929 0.05287548
#> 2      Culex tarsalis 0.006729853 0.001286999 0.02790341
#> 3 Culex quinquefasciatus 0.022851731 0.001595994 0.10915702
```

The formula interface in R provides a convenient, generally consistent interface to modeling function expressions, and this interface has been adapted to ease use of the functions in `PooledInfRate`. As an example,

```
pooledBin(Pos ~ PoolSize, mosq.dat) # the second function argument is data=
#>      P  Lower  Upper
#> 0.0120 0.0048 0.0289
```

Specification of a grouping variable is made using a vertical bar (`|`) in the formula:

```
pooledBin(Pos ~ PoolSize | Species, mosq.dat) # the second function argument is data=
#>      Species   PointEst      Lower      Upper
#> 1      Culex pipiens 0.013370328 0.002778929 0.05287548
#> 2      Culex tarsalis 0.006729853 0.001286999 0.02790341
#> 3 Culex quinquefasciatus 0.022851731 0.001595994 0.10915702
```

This example data set `mosq.dat` does not have a “number of pools” variable, so the above evaluations do not need this specified, since by default this is a vector of 1s. Use of formula interface to specify such a variable is required for the `ex2.dat` data set, however, and this is accommodated using “special” functions `m()` and `n()` (echoing the mathematical exposition in Hepworth and Biggerstaff (2017)). As an illustration, the following incorporates this variable:

```
pooledBin(Pos ~ m(PoolSize) + n(NumPools), ex2.dat)
#>      P  Lower  Upper
#> 0.0052 0.0017 0.0126
```

Because the pool size variable (`m()`) is *required* for any pooled estimation (and because it is easier when there is less typing), the `m()` specification is optional, so that

```
pooledBin(Pos ~ PoolSize + n(NumPools), ex2.dat)
#>      P  Lower  Upper
#> 0.0052 0.0017 0.0126
```

gives the same result. Further, since the `m()` and `n()` identifiers communicate to the fitting functions the roles of the variables, the order does not matter in the formula, so that the following all provide the same result:

```
pooledBin(Pos ~ m(PoolSize) + n(NumPools), ex2.dat)
#>      P  Lower  Upper
#> 0.0052 0.0017 0.0126
pooledBin(Pos ~ PoolSize + n(NumPools), ex2.dat)
#>      P  Lower  Upper
#> 0.0052 0.0017 0.0126
pooledBin(Pos ~ n(NumPools) + m(PoolSize), ex2.dat)
#>      P  Lower  Upper
#> 0.0052 0.0017 0.0126
pooledBin(Pos ~ n(NumPools) + PoolSize , ex2.dat)
#>      P  Lower  Upper
#> 0.0052 0.0017 0.0126
```

Specification of the grouping variable is as above:

```
pooledBin(Pos ~ PoolSize + n(NumPools) | Location, ex2.dat)
#>   Location   PointEst      Lower      Upper
#> 1         A 0.004242141 0.0011295580 0.01162209
#> 2         B 0.016119609 0.0009872462 0.07615332
```

and use the (significant) 'digits' argument of `print()` to make for easier reading

```
print(pooledBin(Pos ~ PoolSize + n(NumPools) | Location, ex2.dat), digits = 3)
#>   Location   PointEst      Lower      Upper
#> 1         A 0.004242141 0.0011295580 0.01162209
#> 2         B 0.016119609 0.0009872462 0.07615332
```

Summary and plot methods To print out more detailed information on the estimation results, summary methods are provided. The resulting output contains details including the estimates themselves, estimation methods used, the total numbers of positive pools, the total number of individuals, and the total number of pools.

```
pir.combined.out <- pIR(Pos ~ PoolSize + n(NumPools), ex2.dat)
summary(pir.combined.out)
#> Estimation of Binomial Proportion for Pooled Data
#>
#>      P  Lower  Upper
#> 0.0052 0.0017 0.0126
#>
#>
#> Call: pIR(x = Pos ~ PoolSize + n(NumPools), data = ex2.dat)
#>
#> Point estimator: Firth's Correction
#> CI method: Skew-Corrected Score (Gart)
#>
#> Number of individuals: 810
#> Number of pools: 27
#> Number of positive pools: 4
```

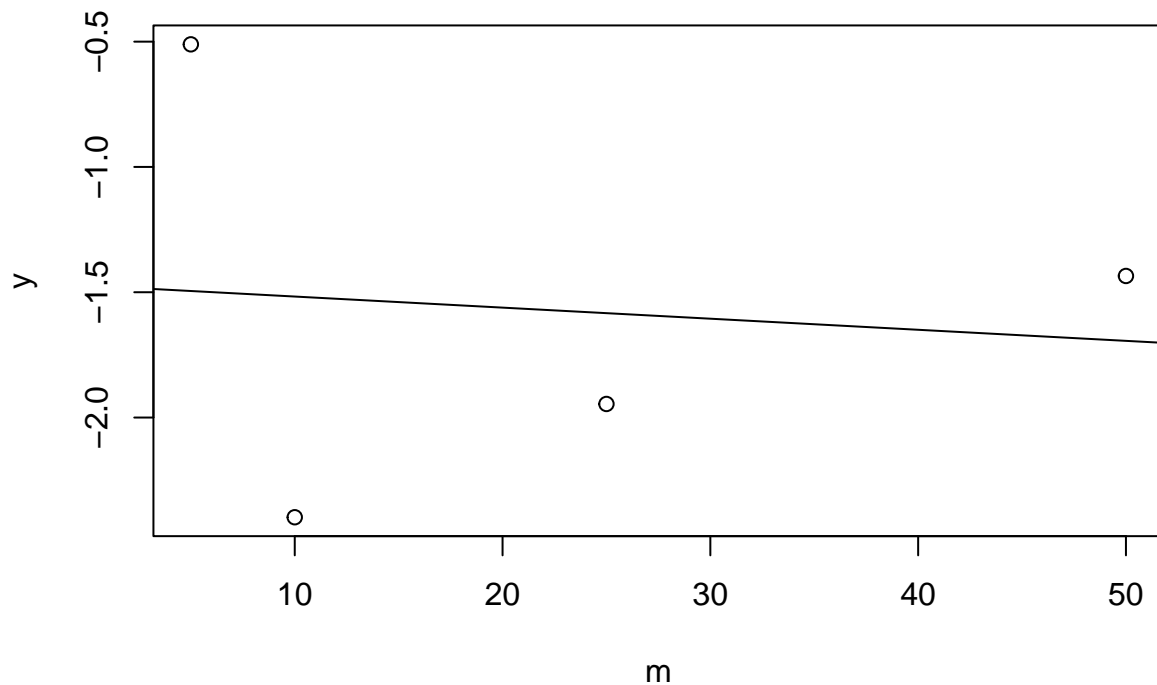
```
pir.location.out <- pIR(Pos ~ PoolSize + n(NumPools) | Location, ex2.dat)
```

```
summary(pir.location.out)
#>
#> Estimation of Binomial Proportion for Pooled Data
#>
#> Call: pIR(x = Pos ~ PoolSize + n(NumPools) | Location, data = ex2.dat)
#>
#> Point estimator      : Firth's Correction
#> CI method           : Skew-Corrected Score (Gart)
#> Confidence coefficient : 95%
#>
#>   Location   PointEst      Lower      Upper    N NumPools NumPosPools Scale
#> 1         A 0.004242141 0.0011295580 0.01162209 750         20          3      1
#> 2         B 0.016119609 0.0009872462 0.07615332  60          7          1      1

pir.mosq.out <- pIR(Pos ~ PoolSize | Species, mosq.dat)
print(summary(pir.mosq.out), digits=3)
#>
#> Estimation of Binomial Proportion for Pooled Data
#>
#> Call: pIR(x = Pos ~ PoolSize | Species, data = mosq.dat)
#>
#> Point estimator      : Firth's Correction
#> CI method           : Skew-Corrected Score (Gart)
#> Confidence coefficient : 95%
#>
#>           Species   PointEst      Lower      Upper    N NumPools NumPosPools
↪ Scale
#> 1         Culex pipiens 0.013370328 0.002778929 0.05287548 180          4          2
↪ 1
#> 2         Culex tarsalis 0.006729853 0.001286999 0.02790341 385          5          2
↪ 1
#> 3 Culex quinquefasciatus 0.022851731 0.001595994 0.10915702  40          3          1
↪ 1
```

A plot method of the diagnostic tool described by Chen and Swallow (1990) to evaluate the suitability of the binomial model is available.

```
plot(pir.combined.out)
```



```
#> R-squared for diagnostic line fit = 0.0122
```

and this also works when there are groups (though adjustment of the layout parameter may be required).

Missing data Records with missing data on any of the variables used in the function calls are removed for estimation. For example,

```
x.na <- c(1,0,0,NA,1)
m.na <- c(10,50,25,10,10)
```

```
pooledBin(x.na, m.na)
#>      P  Lower  Upper
#> 0.0198 0.0046 0.0605
```

```
summary(pooledBin(x.na,m.na))
#> Estimation of Binomial Proportion for Pooled Data
#>
#>      P  Lower  Upper
#> 0.0198 0.0046 0.0605
#>
#>
#> Call: pooledBin(x = x.na, m = m.na)
#>
#> Point estimator: Firth's Correction
#> CI method: Skew-Corrected Score (Gart)
#>
#> Number of individuals: 95
```

```
#> Number of pools: 4
#> Number of positive pools: 2
```

and this is also true using the formula method interface.

Options Estimation methods

As described above, there are several point and CI methods available, and the choices are specified with the `pt.method` and `ci.method` parameters.

```
pIR(Pos ~ PoolSize, mosq.dat, pt.method = "firth", ci.method = "skew-score") # the
  ↪ defaults
#>      P Lower Upper
#> 0.0120 0.0048 0.0289
pIR(Pos ~ PoolSize, mosq.dat, pt.method = "gart", ci.method = "score")
#>      P Lower Upper
#> 0.0118 0.0050 0.0274
pIR(Pos ~ PoolSize, mosq.dat, pt.method = "mle", ci.method = "wald")
#>      P Lower Upper
#> 0.0135 0.0003 0.0266
# specification of "mir" for either sets the other to "mir"
pIR(Pos ~ PoolSize, mosq.dat, pt.method = "mir", ci.method = "mir")
#>      P Lower Upper
#> 0.0083 0.0011 0.0155
```

Also, note that when all pool sizes are 1, then there is no pooling, so the standard Wilson score interval is returned, and `pt.method` is set to “mle” and `ci.method` is set to “score” in this case.

For users of the Microsoft Excel™ PooledInfRate add-in: *the default point estimation method for this R package is Firth’s method, which is different than the one in Excel™, which is Gart’s method.* This is because Firth’s method was shown in Hepworth and Biggerstaff (2017) to perform generally better.

Scale

Estimated values for the prevalence are often very small in pooled testing applications, so a scale parameter is provided to facilitate reading output (`scale = 1000` is often used in entomological applications). Results are printed using the specified scale, both with the standard print and summary methods.

```
pIR(Pos ~ PoolSize, mosq.dat) # default scale = 1
#>      P Lower Upper
#> 0.0120 0.0048 0.0289
pIR(Pos ~ PoolSize, mosq.dat, scale=1000)
#>      P Lower Upper Scale
#> 12.0484 4.7876 28.9463 1000
```

Confidence level and algorithm precision

To set the desired confidence level for CIs, the parameter `alpha`, with default `alpha = 0.05`, is used, with the confidence level equal to $100(1-\alpha)\%$.

The estimation methods used require iterative numerical computation, and the parameter `tol` (for “tolerance”) is used to indicate how precisely estimation is required; the default is `.Machine$double.eps^0.5`.

Differences of proportions (two-sample)

The “equal” pair of functions `pooledBinDiff` / `pIRDiff` compute confidence intervals for differences of proportions, as detailed in Biggerstaff (2008). The interface for the default call is an expansion of the one-sample call: `pooledBin(x1,m1,x2,m2)`. For the formula interface, these functions are exactly the same as for the one-sample case, only a group variable is in this case *required*. Estimates of differences and CIs for all

pairwise differences of the levels of the grouping variable are computed, and a summary method provides detail, including individual group estimates. Examples follow.

```
x1 <- c(1,0,0,0)
m1 <- c(100,50,25,10)
x2 <- c(1,1,0,0)
m2 <- c(50,40,30,20)
pooledBinDiff(x1,m1,x2,m2)
#> Comparison Diff Lower Upper
#> 1 - 2 -0.0124 -0.0714 0.0235

n1 <- c(10,20,30,40)
n2 <- rep(1,4)
pooledBinDiff(x1,m1,x2,m2,n1,n2)
#> Comparison Diff Lower Upper
#> 1 - 2 -0.0182 -0.0773 -0.003

pooledBinDiff(Pos ~ PoolSize + n(NumPools) | Location, ex2.dat)
#> Comparison Diff Lower Upper
#> A - B -0.0119 -0.0719 0.007

mdiff.out <- pIRDiff(Pos ~ PoolSize | Species, mosq.dat, scale=1000) # scale for easier
  ↪ interpretation

# Print fewer digits for easier reading
print(summary(mdiff.out), digits = 3)
#>
#> Estimation of Difference of Binomial Proportions for Pooled Data
#>
#> Call: pIRDiff.formula(x = Pos ~ PoolSize | Species, data = mosq.dat, scale = 1000)
#>
#> Point estimator: Firth's Correction
#> CI method: Skew-Corrected Score (Gart)
#>
#>
#> Comparison Diff Lower Upper Scale
#> 1 Culex pipiens - Culex tarsalis 6.640 -16.885 45.699 1000
#> 2 Culex pipiens - Culex quinquefasciatus -9.481 -94.134 37.599 1000
#> 3 Culex tarsalis - Culex quinquefasciatus -16.122 -101.228 16.103 1000
#>
#> Group summaries:
#>
#> Species P Lower Upper Scale Individuals Pools Positive
#> ↪ Pools
#> 1 Culex pipiens 13.370328 2.778929 52.87548 1000 180 4
#> ↪ 2
#> 2 Culex tarsalis 6.729853 1.286999 27.90341 1000 385 5
#> ↪ 2
#> 3 Culex quinquefasciatus 22.851731 1.595994 109.15702 1000 40 3
#> ↪ 1
```

Missing data As with the one-sample functions, records with missing data on any of the variables used in the function calls are removed for estimation.

Options Options scale, alpha, and tol are the same as for the one-sample functions.

Vector Index

In mosquito-borne disease surveillance, the *Vector Index* (*VI*) is a measure used in evaluating risk of infection in a human population, often to aid decisions on community interventions, such as area-wide vector mosquito abatement (see, e.g., Fauver et al. (2015)). Primarily used in mosquito-borne disease surveillance, the *VI* is the sum of the products of a measure of population “size” and the infection rate over vector species. Because this definition requires the specification of “vector species” over which to perform these computations, an additional variable is needed beyond those specified in the `pooledBin` / `pIR` functions above. This is accommodated in the default and formula interfaces as follows.

Field collections of mosquitoes used for such surveillance can result in a wide range of data configurations, and the individuals that contribute to the population size measures may not be the same as those used in estimating the infection rate (prevalence).

All and only individuals in pools contribute to population size measure

The most straightforward data situation is when all individuals in the collection are used in both components, the population measure and the infection rate estimate. Data in this case may be formatted as above, say with the example `mosq.dat` data set and the example just shown.

For the default interface, a vector parameter is used, so using the `mosq.dat` example data set above,

```
vi.out <- with(mosq.dat, vectorIndex(Pos,PoolSize,vector=Species)) # or use VI()
vi.out
#>    VI
#> 1.425
```

and there is a summary method available to provide detail:

```
summary(vi.out)
#> Vector Index = 1.425
#>
#> Call:  vectorIndex(x = Pos, m = PoolSize, vector = Species)
#>
#> Prevalence estimate method : firth
#> Use non-tested individuals in abundance estimate : FALSE
#> Use tested pools' pool sizes in abundance estimate : TRUE
#>
#> Detail by vector:
#>
#>
#>           Species      Avg N      P (Avg N) * P
#> Culex pipiens      Culex pipiens 45.00000 0.013370328  0.6016647
#> Culex tarsalis      Culex tarsalis 77.00000 0.006729853  0.5181987
#> Culex quinquefasciatus Culex quinquefasciatus 13.33333 0.022851731  0.3046898
```

Using the formula interface, the vector species variable is specified after a forward slash symbol (/), written after the main part of the formula:

```
vectorIndex(Pos ~ PoolSize / Species, mosq.dat)
#>    VI
#> 1.425
```

Computing the *VI* by groups is available using the same interface as groups for the one- and two-sample functions. Note, however, that using the formula interface, the group variable **must** come before the slash (/) indicating the vector species variable. Examples using the `mosq.dat` data set are

```
with(mosq.dat, vectorIndex(Pos, PoolSize, vector=Species, group=Zone))
#>   Zone      VI
#> 1     1 1.8732123
#> 2     2 0.3963581
```

```

VI(Pos ~ PoolSize | Zone / Species, mosq.dat)
#>   Zone      VI
#> 1     1 1.8732123
#> 2     2 0.3963581

```

The above calculations assume that collection effort is constant (i.e., represents the same time in the environment for collection) across traps. Mosquito trapping effort, often expressed “per trap night” to reflect the duration between retrievals of specimens, may differ by trap, however, and when pools are trap-specific or contain individuals caught in traps using the same effort, a trapping effort variable provided to specify this. For the default interface, a trap.time variable is specified, while for the formula interface the variable is included *after* the Vector variable, separated by a colon (:). Examples using the mosq.dat data set are

```

with(mosq.dat, vectorIndex(Pos, PoolSize, vector=Species, trap.time = Nights,
  ↪ group=Zone))
#>   Zone      VI
#> 1     1 0.3746425
#> 2     2 0.1321194

```

```

VI(Pos ~ PoolSize | Zone / Species:Nights, mosq.dat)
#>   Zone      VI
#> 1     1 0.3746425
#> 2     2 0.1321194

```

An external data set (only) is used to provide the population size measure

It may be that field collections used for computing the measure of population size differ, or that individual mosquitoes are pooled from different traps possibly representing different collection efforts. If the counts of mosquitoes caught are aggregated separately by collection effort, and otherwise some or all of them are pooled for testing and infection rate estimation, these pieces of data may be combined in computing the VI by formatting the data using missing values (NA) for the response data to be used in computing the population size measure. This is possible in both the default and formula interfaces by correct specification of the n.use.traps and n.use.na parameter options, which indicate what subsets of the data are to be used to compute the population measure (reflected in the “n.use.” in the parameter names).

As an example data set, the document *West Nile Virus in the United States: Guidelines for Surveillance, Prevention, and Control* of the US Centers for Disease Control and Prevention, Division of Vector-Borne Diseases, contains this example data set in Appendix 2 (written here in aggregated form):

```

pools.dat <- data.frame(Species=c("Cx. tarsalis", "Cx. pipiens"),
  Pos=c(1,1),
  PoolSize=c(50,50),
  NumPools=c(6,5))

```

```

pools.dat
#>   Species Pos PoolSize NumPools
#> 1 Cx. tarsalis 1      50        6
#> 2 Cx. pipiens 1      50        5

```

In addition to the pool testing data, the total number of *Cx. tarsalis* caught for 6 traps was 442, and the total number of *Cx. pipiens* caught for 6 traps was 233. To include this information in the pools.dat data set, (1) enter missing values (NA) for the “number positive” variable (x); (2) treat the “pool size” variable (m) as the collection count; and augment “number of pools” variable (n) as 1.

```

traps.dat <- data.frame(Species = c("Cx. tarsalis", "Cx. pipiens"),
  Pos = c(NA, NA),
  PoolSize=c(442, 233),
  NumPools=c(1,1)) # note NumPools should be 1 for each entry
vi.dat <- rbind(pools.dat, traps.dat)

```

```

vi.dat
#>      Species Pos PoolSize NumPools
#> 1 Cx. tarsalis  1      50        6
#> 2 Cx. pipiens  1      50        5
#> 3 Cx. tarsalis NA     442        1
#> 4 Cx. pipiens  NA     233        1

VI(Pos ~ PoolSize + n(NumPools) / Species, vi.dat, n.use.traps = FALSE, n.use.na = TRUE)
#> VI
#> 2.412
summary(VI(Pos ~ PoolSize + n(NumPools) / Species, vi.dat, n.use.traps = FALSE, n.use.na
  ↪ = TRUE))
#> Vector Index = 2.412
#>
#> Call: VI(x = Pos ~ PoolSize + n(NumPools)/Species, data = vi.dat, n.use.traps =
  ↪ FALSE, n.use.na = TRUE)
#>
#> Prevalence estimate method : firth
#> Use non-tested individuals in abundance estimate : TRUE
#> Use tested pools' pool sizes in abundance estimate : FALSE
#>
#> Detail by vector:
#>
#>      Species Avg N      P (Avg N) * P
#> Cx. tarsalis Cx. tarsalis  442 0.003341092  1.4767625
#> Cx. pipiens  Cx. pipiens  233 0.004013436  0.9351305

```

Finally, to account for trapping effort, recall that these vector mosquito counts are from 6 traps (assumed to be for 1 night each). Augment the vi.dat data set as

```

# put 1s in for the pool data for later; use values for trap nights in application
vi.dat$TrapNights <- c(1,1,6,6)
vi.dat

```

```

#>      Species Pos PoolSize NumPools TrapNights
#> 1 Cx. tarsalis  1      50        6          1
#> 2 Cx. pipiens  1      50        5          1
#> 3 Cx. tarsalis NA     442        1          6
#> 4 Cx. pipiens  NA     233        1          6

VI(Pos ~ PoolSize + n(NumPools) / Species:TrapNights, vi.dat,
  n.use.traps = FALSE, n.use.na = TRUE)
#> VI
#> 0.402
summary(VI(Pos ~ PoolSize + n(NumPools) / Species:TrapNights, vi.dat,
  n.use.traps = FALSE, n.use.na = TRUE))
#> Vector Index = 0.402
#>
#> Call: VI(x = Pos ~ PoolSize + n(NumPools)/Species:TrapNights, data = vi.dat,
  ↪ n.use.traps = FALSE, n.use.na = TRUE)
#>
#> Prevalence estimate method : firth
#> Use non-tested individuals in abundance estimate : TRUE
#> Use tested pools' pool sizes in abundance estimate : FALSE
#>
#> Detail by vector:

```

```
#>
#>               Species      Avg N          P (Avg N) * P
#> Cx. tarsalis Cx. tarsalis 73.66667 0.003341092    0.2461271
#> Cx. pipiens  Cx. pipiens 38.83333 0.004013436    0.1558551
```

The Avg N values reported in the summary result match the CDC result. To match the VI result exactly, note that the CDC computation used the MIR as the estimate of the infection rate, so compute this using

```
VI(Pos ~ PoolSize + n(NumPools) / Species:TrapNights, vi.dat, n.use.traps = FALSE,
  ↪ n.use.na = TRUE, pt.method="mir")
#>      VI
#> 0.4009
summary(VI(Pos ~ PoolSize + n(NumPools) / Species:TrapNights, vi.dat, n.use.traps =
  ↪ FALSE, n.use.na = TRUE, pt.method="mir"))
#> Vector Index = 0.4009
#>
#> Call:  VI(x = Pos ~ PoolSize + n(NumPools)/Species:TrapNights, data = vi.dat,
  ↪ n.use.traps = FALSE, n.use.na = TRUE,
#>
#> Prevalence estimate method : mir
#> Use non-tested individuals in abundance estimate : TRUE
#> Use tested pools' pool sizes in abundance estimate : FALSE
#>
#> Detail by vector:
#>
#>               Species      Avg N          P (Avg N) * P
#> Cx. tarsalis Cx. tarsalis 73.66667 0.003333333    0.2455556
#> Cx. pipiens  Cx. pipiens 38.83333 0.004000000    0.1553333
```

An external data set together with the pooled data sizes (counts) are used to provide the population size measure

Finally, if the individuals in the pools are not also counted in the “not-tested” data set (since doing so would over-count the collected number of individuals), these counts can be included in the computation of the population size measure, making sure that the collection effort variable (trap.time; TrapNights in the example) is correct for all the individuals in the pools. To include these in the computation, simply set the n.use.traps variable to TRUE:

```
# not trying to match the CDC guidelines report here, since in the example more
  ↪ individuals are
# included in the population size (mosquito density) measure
VI(Pos ~ PoolSize + n(NumPools) / Species:TrapNights, vi.dat,
  n.use.traps = TRUE, n.use.na = TRUE)
#>      VI
#> 0.6311
summary(VI(Pos ~ PoolSize + n(NumPools) / Species:TrapNights, vi.dat,
  n.use.traps = TRUE, n.use.na = TRUE))
#> Vector Index = 0.6311
#>
#> Call:  VI(x = Pos ~ PoolSize + n(NumPools)/Species:TrapNights, data = vi.dat,
  ↪ n.use.traps = TRUE, n.use.na = TRUE)
#>
#> Prevalence estimate method : firth
#> Use non-tested individuals in abundance estimate : TRUE
#> Use tested pools' pool sizes in abundance estimate : TRUE
#>
```

```
#> Detail by vector:
#>
#>           Species Avg N          P (Avg N) * P
#> Cx. tarsalis Cx. tarsalis 106 0.003341092 0.3541557
#> Cx. pipiens  Cx. pipiens  69 0.004013436 0.2769271
```

Options

Other options to the `vectorIndex` / `VI` functions are the same as those for `pooledBin`, so that the user may specify the method for point estimation.

Confidence interval for *VI*

The *VI* is a measure used for evaluation of infection risk of some human pathogen, but in principle it is not estimating a population quantity, though it is expected to be proportional to the number of infected individual vectors. Because of this, at this time functionality is not included to compute CIs (which are inferential beasts, after all) to accompany *VI* computations. Inclusion of CI computations to provide some measure of “uncertainty” or “variability” of the *VI* computed is being evaluated, and this would be straightforward conditional on the collection counts. A more complete measure of such variability would include uncertainty in the counts, too, and this is not presently available (but is being evaluated).

Imperfect test

The methods used for point estimation of population individual-level prevalence using `pooledBin` assume that the test used to assess the “positivity” of a pool are perfect. Hepworth and Biggerstaff (2021) extend the methods of Hepworth and Biggerstaff (2017) to include imperfect tests, quantified through known values for test *sensitivity* and *specificity*. This functionality is available via the function `ipooledBin`; there is not at present a corresponding `ipIR`, as further development will see this extension incorporated directly into `pooledBin`.

The principle difference between `pooledBin` and `ipooledBin` is the specification of the parameters `sens` and `spec`, which both default to 1, a default perfect test. Using the `mosq.dat` data set from above:

```
ipooledBin(Pos ~ PoolSize | Species, mosq.dat, sens=0.9, spec=0.95)
#>           Species PointEst
#> 1      Culex pipiens 0.009979804
#> 2      Culex tarsalis 0.006631455
#> 3 Culex quinquefasciatus 0.019706390
summary(ipooledBin(Pos ~ PoolSize | Species, mosq.dat, sens=0.9, spec=0.95))
#>
#> Estimation of Binomial Proportion for Pooled Data using an Imperfect Test
#> Call: ipooledBin(x = Pos ~ PoolSize | Species, data = mosq.dat, sens = 0.9, spec =
  ↪ 0.95)
#>
#>
#> Call: ipooledBin(x = Pos ~ PoolSize | Species, data = mosq.dat, sens = 0.9, spec =
  ↪ 0.95)
#>
#> Sensitivity : 0.9
#> Specificity : 0.95
#> Point estimator : firth
#>           Species PointEst  N NumPools NumPosPools Scale
#> 1      Culex pipiens 0.009979804 180      4      2      1
#> 2      Culex tarsalis 0.006631455 385      5      2      1
#> 3 Culex quinquefasciatus 0.019706390 40      3      1      1
```

Estimates based on the uncorrected MLE for imperfect tests are also available by setting the `pt.method` parameter:

```

ipooledBin(Pos ~ PoolSize | Species, mosq.dat, sens=0.9, spec=0.95, pt.method="mle")
#>           Species      PointEst
#> 1      Culex pipiens 0.015051887
#> 2      Culex tarsalis 0.009239927
#> 3 Culex quinquefasciatus 0.028026274
summary(ipooledBin(Pos ~ PoolSize | Species, mosq.dat, sens=0.9, spec=0.95,
  ↪ pt.method="mle"))
#>
#> Estimation of Binomial Proportion for Pooled Data using an Imperfect Test
#> Call: ipooledBin(x = Pos ~ PoolSize | Species, data = mosq.dat, pt.method = "mle",
  ↪ sens = 0.9, spec = 0.95)
#>
#>
#> Call: ipooledBin(x = Pos ~ PoolSize | Species, data = mosq.dat, pt.method = "mle",
  ↪ sens = 0.9, spec = 0.95)
#>
#> Sensitivity : 0.9
#> Specificity : 0.95
#> Point estimator : mle
#>
#>           Species      PointEst      N NumPools NumPosPools Scale
#> 1      Culex pipiens 0.015051887 180         4          2      1
#> 2      Culex tarsalis 0.009239927 385         5          2      1
#> 3 Culex quinquefasciatus 0.028026274 40         3          1      1

```

Compared to `pooledBin`, the additional parameter `p.start` gives the user the options of specifying a starting value for the numerical algorithm (Newton-Raphson) used in estimation; there is a default value set for this when `p.start = NULL`, as is the default.

Confidence intervals are not included as an option for `ipooledBin`, because at present there is not a theoretically recommended method. (This is why this functionality has not yet been incorporated into the base `pooledBin` function.)

Finally, a word of caution: as noted in Hepworth and Biggerstaff (2021), convergence of the computational algorithm in the presence of an imperfect test is not assured (either computationally or theoretically) for every value of sensitivity and specificity for a given data set, as some data are simply incompatible with some test performance specifications. Should the user encounter convergence issues using specified values for `sens` and `spec`, the recommendation is to evaluate estimates for a variety of these parameter specifications, perhaps beginning by assuming a perfect test, to see how estimates are impacted by such specifications.

-
- Biggerstaff, BJ. 2008. "Confidence Intervals for the Difference of Two Proportions Estimated from Pooled Samples." *Journal of Agricultural, Biological, and Environmental Statistics* 13 (4): 478–96.
- Chen, CL, and WH Swallow. 1990. "Using Group Testing to Estimate a Proportion, and to Test the Binomial Model." *Biometrics* 46 (4): 1035–46.
- Chiang, CL, and WC Reeves. 1962. "Statistical Estimation of Virus Infection Rates in Mosquito Vector Populations." *American Journal of Hygiene* 75 (3): 377–91.
- Fauver, JR, L Pecher, JA Schurich, BG Bolling, M Calhoun, ND Grubaugh, KL Burkhalter, et al. 2015. "Temporal and Spatial Variability of Entomological Risk Indices for West Nile Virus Infection in Northern Colorado: 2006–2013." *Journal of Medical Entomology* 53 (2): 425–34. <https://doi.org/10.1093/jme/tjv234>.
- Firth, D. 1993. "Bias Reduction of Maximum Likelihood Estimates." *Biometrika* 80: 27–38.
- Gart, JJ. 1991. "An Application of Score Methodology: Confidence Intervals and Tests of Fit for One-Hit Curves." In *Handbook of Statistics*, 8:395–406. Elsevier Science Publishers.
- Hepworth, G. 2005. "Confidence Intervals for Proportions Estimated by Group Testing with Groups of Unequal Size." *Journal of Agricultural Biological and Environmental Statistics* 10 (4): 478–97.
- Hepworth, G, and BJ Biggerstaff. 2017. "Bias Correction in Estimating Proportions by Pooled Testing."

Journal of Agricultural, Biological and Environmental Statistics 22 (4): 602–14. <https://doi.org/10.1007/s13253-017-0297-2>.

———. 2021. “Bias Correction in Estimating Proportions by Imperfect Pooled Testing.” *Journal of Agricultural, Biological and Environmental Statistics* 26 (1): 90–104. <https://doi.org/10.1007/s13253-020-00411-5>.

Microsoft Corporation. n.d. *Microsoft Excel*. <https://office.microsoft.com/excel>.