Package 'averisk'

October 12, 2022

Type Package

Title Calculation of Average Population Attributable Fractions and Confidence Intervals

Version 1.0.3

Imports MASS (>= 7.3.0)

Date 2015-10-22

Maintainer John Ferguson <john.ferguson@nuigalway.ie>

Description Average population attributable fractions are calculated for a set of risk factors (either binary or ordinal valued) for both prospective and case-control designs. Confidence intervals are found by Monte Carlo simulation. The method can be applied to either prospective or case control designs, provided an estimate of disease prevalence is provided. In addition to an exact calculation of AF, an approximate calculation, based on randomly sampling permutations has been implemented to ensure the calculation is computationally tractable when the number of risk factors is large.

RoxygenNote 5.0.1

NeedsCompilation no

Author John Ferguson [aut, cre]

Repository CRAN

Date/Publication 2017-03-21 06:50:31 UTC

R topics documented:

averisk *averisk: Calculation of Average Population Attributable Fractions and Confidence Intervals.*

Description

Average population attributable fractions are calculated for a set of risk factors (either binary or ordinal valued) for both prospective and case-control designs. Confidence intervals are found by Monte Carlo simulation. The method can be applied to either prospective or case control designs, provided an estimate of disease prevalence is provided. In addition to an exact calculation of AF, an approximate calculation, based on randomly sampling permutations has been implemented to ensure the calculation is computationally tractable when the number of risk factors is large.

averisk functions

getAF

getAF *Calculate average attributable fractions*

Description

This function calculates average attributable fractions and confidence intervals for discrete riskfactors

Usage

```
getAF(f, the.data, ref_cat = NULL, refy = 0, cat_confounders = NULL,
  cont_confounders = NULL, prev = NA, allperm = TRUE,
  nsample_perm = 1000, approx_error = NA, ci = FALSE, conf_level = 0.99,
  nsample_var = 100, correction_factor = TRUE, quantile_int = FALSE,
  sep_est = TRUE, w = NULL)
```

Arguments

f	A formula object specifying a logistic regression model for the risk factors. See Details for more information.
the.data	A dataframe containing the disease indicator, riskfactors and possible confounders.
ref_cat	A character vector indicating the reference values for each risk factor. Defaults to NULL. In the case that this argument is not specified, the default R assignment for the reference level is used for risk factors coded as character and factor variables (see ?factor) whereas the minimum value is assigned as the reference level for risk factors coded as numeric vectors. Attributable fractions calculate the proportional change in disease prevalence that might be expected if the entire population had the reference value for each risk factor.

refy Response value that specifies controls. Defaults to 0.

cat_confounders
A character vector indicating categorical variables that need to be adjusted for (excluding risk factors). Defaults to NULL.

cont_confounders
A character vector indicating categorical variables that need to be adjusted for (excluding risk factors). Defaults to NULL.

prev A proportion specifying the percentage of the population that have the disease. Defaults to NA. NA is appropriate for prospective and survey designs. Prevalence should be specified for case control designs.

allperm TRUE gives an exact calculation of the sample average fraction. FALSE gives an approximate calculation. Defaults to TRUE.

nsample_perm How many permutations are used when calculating approximate average attributable fractions? (only necessary to specify when allperm=FALSE). If approx_error is specified, then nsample_perm is the number of sampled permutations for the construction of the standard error (if ci=TRUE), but is overrided in the construction of the point estimate.

approx_error Specifying this option will calculate average fractions using the number of permutations necessary to approximate the true sample average fraction point estimate to within approx_error. Defaults to NA.

ci Is a confidence interval required? Defaults to FALSE.

conf_level The confidence level specified as a proportion. i.e. conf_level=0.95 would imply 95 percent confidence. Only necessary to specify when ci=TRUE

nsample_var The number of monte carlo iterates of the average fraction that are used when calculating the confidence interval. Only necessary to specify when ci=TRUE

correction_factor
Whether an extra correction term is subtracted from the estimated standard error due to the monte carlo simulated AFs and the point estimate for the AF being based on different numbers of permutations. Defaults to TRUE. (Only necessary to specify when ci=TRUE and allperm=FALSE)

quantile_int confidence interval is calculated from empirical quantiles of the monte carlo simulated AFs. Defaults to FALSE. Only necessary to specify when ci=TRUE. See Details for more information.

sep_est The point estimate of the AF in a separate calculation to the confidence interval. Defaults to TRUE (Only necessary to specify when ci=TRUE)

w Optional weighting vector for survey data Defaults to NULL

Details

The model formula f is specified using traditional R notation. For instance, in the situation where the binary response is 'y' and there are 3 risk factors: x1, x2 and x3, f would be specified as y~x1+x2+x3. Interactions are not permitted in the model. Confounders (either categorical or continuous) are added as separate arguments in character vectors. When a confidence interval is requested, a symmetric interval around the point estimate AF is given by default. In the case that nsample_var is large, a possibly assymetric interval may instead be requested using quantile_int=TRUE. In this

case, the interval is generated from percentiles of the Monte Carlo simulates of the AF. Since estimating percentiles of a distribution is more difficult than estimating the overall variance, `nsample_var` should be increased if quantile based confidence intervals are desired, and doing so will significantly increase run-time. Quantile based confidence intervals maybe superior to symmetric intervals when the distribution of the simulated AFs is skewed. However, in our experience, the distribution of Monte Carlo simulates is usually relatively symmetric.

Value

If `ci=TRUE`, a 3 x (K+1) matrix, where K is the number of risk factors. The first row represents the point estimate for the AF, and the second and third rows lower and upper confidnece bounds. If `ci=FALSE`, a (K+1)-dimensional vector with the calculated point estimates for the AF is returned.

Author(s)

John Ferguson (john.ferguson@nuigalway.ie)

References

Eide, Geir Egil and Olaf Gefeller. Sequential and Average Attributable Fractions as Aids in the Selection of Preventative Strategies. J. Clinical Epidemiology, Vol. 48, No. 5, pp. 645-655, 1995.

Ferguson John, Alvarez Alberto, Newell John, Hinde John and O'Donnell Martin. Estimating average attributable fractions with confidence intervals for cohort and case control studies. Statistical Methods in Medical Research, 2016.

Examples

```
# the following example is from Eide and Gefeller, 1995
# simulate data

ex_probs <- c(.06732,.02976,.01570,.01787,.01445,.01008,.06986,.06553,.03,.05766,
          .09680,.04194,.02741,.02194,.02474,.01031,.12410,.09537,.08408,.09509) # P(E|D)
disease_probs <- c(.036,.0621,.0236,.0411,.0507,.0864,.1066,.1745,.1867,.2891,.0514,
              .0875,.0339,.0584,.0718,.1206,.1474,.2345,.2497,.3708) # P(D|E)
pe <- ex_probs/disease_probs ## marginal P(E)
pe <- pe/sum(pe)
nond_exposure_probs <- (1-disease_probs)*pe  # P(E|not D)
nond_exposure_probs <- nond_exposure_probs/sum(nond_exposure_probs)
ex_probs <- ex_probs/sum(ex_probs)
the.mat <- cbind(c(rep(0,10),rep(1,10)),rep(rep(1:5,each=2),2),rep(c(0,1),10))
ncase <- 500
ncontrol <- 500
casemat <- the.mat[sample(1:20,size=ncase,replace=TRUE,prob=ex_probs),]
case_rows <- cbind(rep(1,ncase),casemat)
controlmat <- the.mat[sample(1:20,size=ncase,replace=TRUE,prob=nond_exposure_probs),]
control_rows <- cbind(rep(0,ncontrol),controlmat)
the.d <- rbind(case_rows,control_rows)
colnames(the.d) <- c("y","urban.rural","smoking.category","occupational.exposure")

# Just get the estimate (no confidence interval)
getAF(y~urban.rural+smoking.category+occupational.exposure,the.d,prev=0.09)
```

```
## find the average fraction and associated monte-carlo calculated 99% confidence
##  No need for approximation here.  Assume population prevalence is 9 percent.


getAF(y~urban.rural+smoking.category+occupational.exposure,the.d,prev=0.09,ci=TRUE,conf_level=0.99)


## genetic simulation using more risk factors (disease prevalence = 0.01)  might be slow.

## Not run:
thevec <- dbinom(0:40, size= 40, prob=0.2, log = FALSE)
bin_fun <- function(beta_0){
 sum(thevec*exp(beta_0+.1*(0:40))/(1+exp(beta_0+.1*(0:40))))-0.01
}
beta_0 <- uniroot(bin_fun,lower=-8,upper=5)$root
total_risk <- (0.01-exp(beta_0)/(1+exp(beta_0)))/0.01
risk_per_snp <- total_risk/20
case_probabilities <- (thevec*exp(beta_0 + (0:40)*0.1)/(1+exp(beta_0 + (0:40)*0.1)))/0.01
control_probabilities <- thevec*1/(1+exp(beta_0 + (0:40)*0.1))/0.99
simdata_genetic <- function(ncase,ncontrol){
numbersnps_case <- sample(0:40,ncase,prob=case_probabilities,replace=TRUE)
 numbersnps_control <- sample(0:40,ncase,prob=control_probabilities,replace=TRUE)
 case_rows <- cbind(rep(1,ncase),matrix(0,nrow=ncase,ncol=20))
 control_rows <- cbind(rep(0,ncase),matrix(0,nrow=ncontrol,ncol=20))
 for(i in 1:ncase){
   if(numbersnps_case[i]>0){
     positions <- sample(1:40,numbersnps_case[i])
     positions <- ceiling(positions/2)
  for(j in 1:length(positions)) case_rows[i,positions[j]+1] <- case_rows[i,positions[j]+1]+1
   }
 }
 for(i in 1:ncontrol){
   if(numbersnps_control[i]>0){
     positions <- sample(1:40,numbersnps_control[i])
     positions <- ceiling(positions/2)
     for(j in 1:length(positions)){
         control_rows[i,positions[j]+1]<- control_rows[i,positions[j]+1]+1
     }
   }
}
 return(rbind(case_rows,control_rows))
}
the.d <- simdata_genetic(ncase=250, ncontrol=250)
colnames(the.d) <- c("y",paste("SNP",1:20,sep=""))

## Here we just calculate the approximate average fraction
## from 50 permutations and no confidence interval.
## If CI desired add the argument ci=TRUE and nsample_var to the function.
## 50 permuations is chosen for speed. In reality, 1000 maybe needed

thesnp <- paste0("SNP", 1:20,sep="")
(fmla <- as.formula(paste("y ~ ", paste(thesnp, collapse= "+"))))
```

```
getAF(fmla, the.d,prev=0.01, allperm=FALSE,nsample_perm=50,ci=FALSE)

## Instead of specifying the number of permutations,
## you can specify an estimated approximation error.
## The approximation error will be within this bound with 95% confidence
## approximation error of 0.01 specified for reasons of speed.
## In reality, you may want to use a smaller value for approx_error.

getAF(fmla, the.d,prev=0.01, allperm=FALSE,approx_error=0.01,ci=FALSE)

## End(Not run)
```

Index