Credible Sets: Functions and Examples

Anna Hutchinson

10/10/2018

```
set.seed(20)
```

Install required packages

```
library(devtools)
library(coloc)
library(simGWAS)
library(ggplot2)
```

Functions

1. simref

If phased haplotypes are not avaliable, simref can be used to simulate some reference haplotypes. Lag is set to 5 due to the simple structure it provides for the variance/covariance matrix. This can be altered to allow for more or less linkage disequilibrium in simulations. The output is a $nhaps * nsnps$ matrix of 0s and 1s with an extra columm, 'Probability', representing the probability of each haplotype.

```
# simref generates the reference haplotype
simref <- function(nsnps=100,nhaps=1000,lag=5) {
  maf <- runif(nsnps+lag,0.05,0.5) # common SNPs
  laghaps <- do.call("cbind", lapply(maf, function(f) rbinom(nhaps,1,f)))
  haps <- laghaps[,1:nsnps]
  for(j in 1:lag)
    haps <- haps + laghaps[,(1:nsnps)+j]
  haps <- round(haps/(lag+1))

  snps <- colnames(haps) <- paste0("s",1:nsnps)
  freq <- as.data.frame(haps+1)
  freq$Probability <- 1/nrow(freq)
  freq
}
```

2. simdata

This function uses the output from simref and incorporates functions avaliable in the `simGWAS` and `coloc` packages to ultimately obtain the posterior probabilities of each SNP being the causal variant. The output is a list of nrep data frames. A flat prior of $p1 = 1e-04$ is used.

```
# simdata generates the posterior probabilities for each SNP being causal
simdata <- function(freq,OR=1.5,nrep=10,N0=1000,N1=1000) {
  snps <- colnames(freq)[-ncol(freq)] # snp names
  CV=sample(snps,1) # choose a random CV
```

```
  # calculate genotype probabilties at each SNP relative to the genotype at the CV
  FP <- make_GenoProbList(snps=snps,W=CV,freq=freq)
  zsim <- simulated_z_score(N0=N0, # number of controls
                            N1=N1, # number of cases
                            snps=snps,
                            W=CV, # causal variants, subset of snps
                            gamma.W=log(OR), # log odds ratios
                            freq=freq,# reference haplotypes
                            GenoProbList=FP,
                            nrep=nrep)
  p <- 2*pnorm(-abs(zsim)) # generate p values from z values

  # generate posterior probabilities using finemap.abf function
  MAF <- colMeans(freq[,snps]-1) # minor allele frequencies
  PP <- matrix(0,nrow(p),ncol(p)) # will hold the posterior probs
  results <- lapply(1:nrep, function(i) {
    tmp <- subset(finemap.abf(dataset=list(pvalues=p[i,], N=N0+N1, MAF=MAF,
                  s=N1/(N0+N1), type="cc"), p1=1e-04),snp!="null")
    tmp$SNP.PP <- tmp$SNP.PP/sum(tmp$SNP.PP) # replace post probs with the
    # ratio of evidence for each variant being causal vrs all the others
    colnames(tmp) <- sub("\\.$","",colnames(tmp))
    colnames(tmp) <- sub("SNP.PP","PP",colnames(tmp)) # clean up column names
    tmp$CV <- sub("SNP.","",tmp$snp)==sub("s","",CV)
    tmp[,c("snp","pvalues","MAF","PP","CV")] # include all the rows and the named columns
  })
  results
}
```

Example 1

This example illustrates the use of the `expected_z_score` and the `simulated_z_score` functions in the `simGWAS` package and is taken from the simGWAS vignette. Firstly, the expected z scores are calculated for each SNP, where

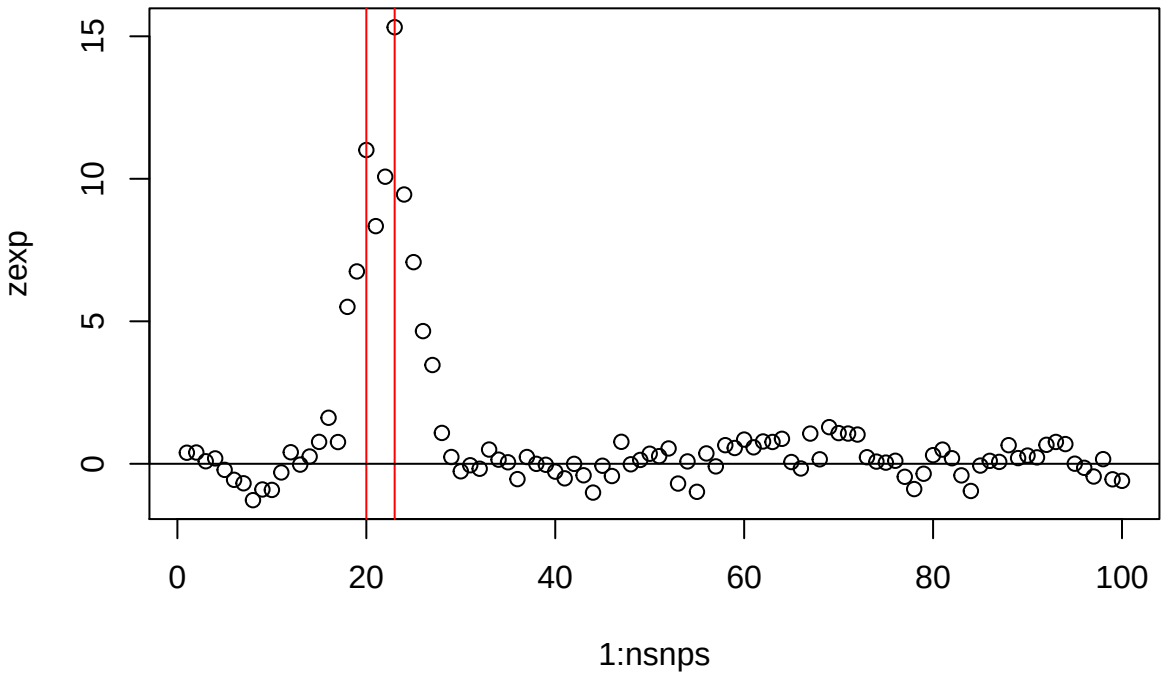
$$z_E = E(Z).$$

```
nsnps <- 100
nhaps <- 1000
lag <- 5 # genotypes are correlated between neighbouring variants
maf <- runif(nsnps+lag,0.05,0.5) # common SNPs
laghaps <- do.call("cbind", lapply(maf, function(f) rbinom(nhaps,1,f)))
haps <- laghaps[,1:nsnps]
for(j in 1:lag)
    haps <- haps + laghaps[,(1:nsnps)+j]
haps <- round(haps/matrix(apply(haps,2,max),nhaps,nsnps,byrow=TRUE))
snps <- colnames(haps) <- paste0("s",1:nsnps)
freq <- as.data.frame(haps+1)
freq$Probability <- 1/nrow(freq)
sum(freq$Probability)
```

```
## [1] 1
```

```
CV=sample(snps[which(colMeans(haps)>0.1)],2) # choose two random SNPs as CVs
g1 <- c(1.4,1.2) # assign OR to these SNPs
```

```
FP <- make_GenoProbList(snps=snps,W=CV,freq=freq)
zexp <- expected_z_score(N0=10000, # number of controls
              N1=10000, # number of cases
              snps=snps, # column names in freq of SNPs for which Z scores should be generated
              W=CV, # causal variants, subset of snps
              gamma.W=log(g1), # odds ratios
              freq=freq, # reference haplotypes
              GenoProbList=FP) # FP above
plot(1:nsnps,zexp); abline(v=which(snps %in% CV),col="red"); abline(h=0)
```



The simulated z scores are then calculated for each SNP where,

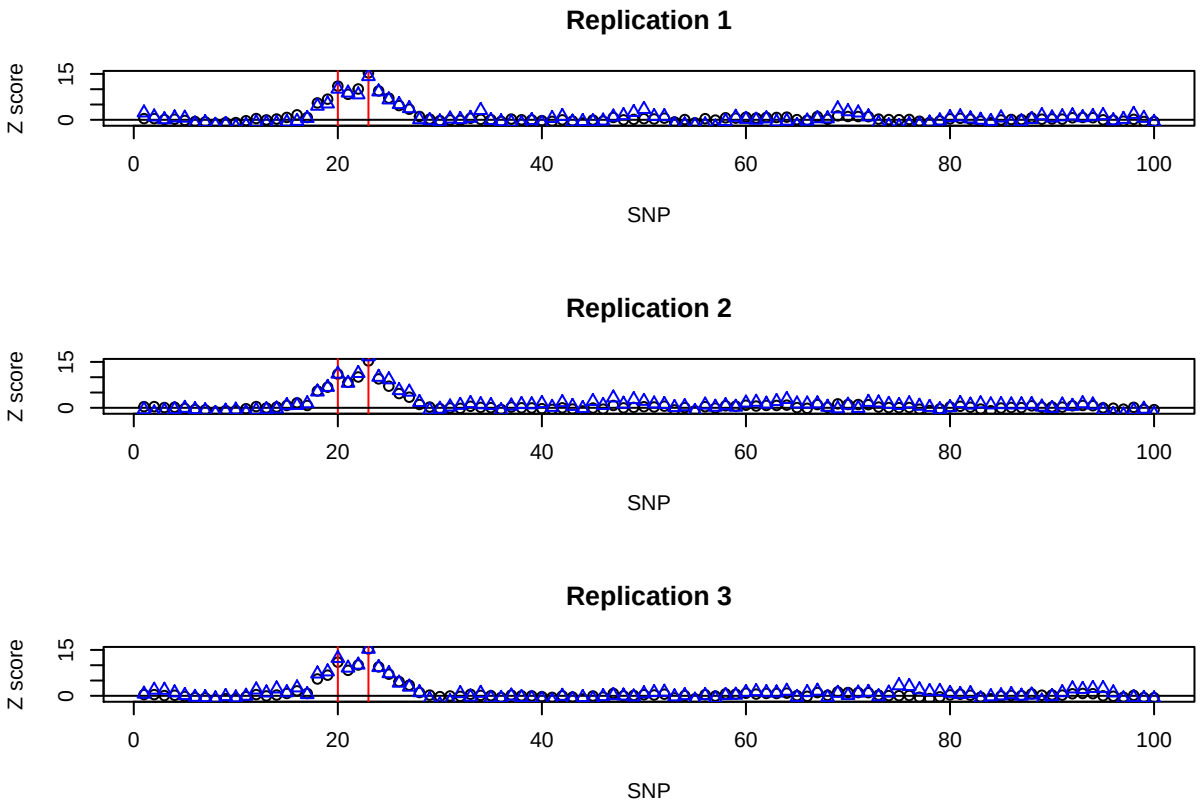
$$z^* \sim N(z_E, \Sigma).$$

Note that the genotype probabilities do not need to be given as input to the function.

```
zsim <- simulated_z_score(N0=10000, # number of controls
              N1=10000, # number of cases
              snps=snps, # column names in freq of SNPs for which Z scores should be generated
              W=CV, # causal variants, subset of snps
              gamma.W=log(g1), # log odds ratios
              freq=freq, # reference haplotypes
            nrep=3)
```

The following plots show the simulated z scores plotted over the expected z scores.

```
par(mfcol=c(3,1))
for(i in 1:3) {
  plot(1:nsnps,zexp,xlab="SNP",ylab="Z score"); abline(v=which(snps %in% CV),col="red"); abline(h=0)
  title(main=paste("Replication",i))
  points(1:nsnps,zsim[i,],col="blue",pch=2)
}
```



Example 2

This example illustrates the use of the `simdata` function. The output for the first 6 SNPs of the first repetition (100 repetitions in total) is also shown. This gives the p-values, the MAF (mean of each column in 'freq') and the posterior probabilities for each SNP. Note that the posterior probability for SNP i is given by,

$$pp_i = \frac{pp_i}{\sum_k pp}.$$

```
freq <- simref()
example_simdata <- simdata(freq, nrep=100)
head(example_simdata[[1]])
```

```
##      snp   pvalues   MAF          PP    CV
## s1 SNP.1 0.4934488 0.005 0.009714097 FALSE
## s2 SNP.2 0.1867151 0.009 0.011045895 FALSE
## s3 SNP.3 0.9058197 0.008 0.008928453 FALSE
## s4 SNP.4 0.7395401 0.009 0.008911872 FALSE
## s5 SNP.5 0.7851162 0.031 0.007036466 FALSE
## s6 SNP.6 0.8268046 0.078 0.005289268 FALSE
```

3. credset

This function uses the posterior probabilities to generate credible sets. The 'do.order' input allows users to specify whether the posterior probabilities are ordered prior to forming credible sets. By default this is set to 'TRUE' to match current credible set practices.

```
credset <- function(pp, causal, thr=0.9,do.order=TRUE) {
  o <- if(do.order) {
    order(pp,decreasing=TRUE)
  } else {
    sample(seq_along(pp))
  }
  cumpp <- cumsum(pp[o])
  wh <- which(cumpp > thr)[1]
  credset <- o[1:wh]
  return(list(credset=credset,
              thr=thr,
              size=sum( pp[credset] ),
              contained=causal %in% credset))
}
```

Example 3

This example shows how the credset function can be used to generate credible sets using both ordered and non-ordered posterior probabilities.

```
# Credible set obtained from ordered posterior probabilities
example_ord_credset <- credset(example_simdata[[1]]$PP, which(example_simdata[[1]]$CV))
example_ord_credset
```

```
## $credset
##  [1]  45  11 100  71  99  72  73  10  44  80  13  95  23  79  29  98  75
## [18]  96  77  70  76   2  36  69  81  43  34   1  46  47  31  50  74  56
## [35]   3  51   4  85  12   8  52  32  37  30  48  97  63  62  33  26  14
## [52]  84   5  92  35  53  78  93  15  87  86  18  49  64  83  90  88  27
## [69]  82  28  89  66  67  16  61  91  54  65  38  94
##
## $thr
## [1] 0.9
##
## $size
## [1] 0.9031255
##
## $contained
## [1] TRUE
```

```
# Credible set obtained from non-ordered posterior probabilities
example_noord_credset <- credset(example_simdata[[1]]$PP, which(example_simdata[[1]]$CV),
                                 do.order=FALSE)
example_noord_credset
```

```
## $credset
##  [1]  29  17  83  18  11  85  80  88  51  96   7  48  45  79  38  32  76
## [18]  22  25  82  62  93  86  42  77  91   2  34  67  99  37  78  46  84
## [35]  47  44  54  60 100  94  87  95   5  26  59  23   6  55  49  52  30
## [52]  41  27  53   9  81  92  68  73  15  98  72  36  71  16  33  12   8
## [69]  35  75  70  39  63  21  20   1  50  90  10  58  65   4  13  31  56
## [86]  14
##
## $thr
## [1] 0.9
```

```
##
## $size
## [1] 0.9038037
##
## $contained
## [1] TRUE
```

4. wrapper

The wrapper function incorporates the above functions to generate 2 credible sets (one from ordering and one from not ordering posterior probabilities) from a user specific OR and threshold. It uses the haplotype matrix obtained from the `simref` function.

```
wrapper <- function(thr=0.9,...) {
  data <- simdata(freq,...) # data is a list of data.frames

  # work out the credsets
  cs <- lapply(data, function(d) {
    tmp.ord <- credset(d$PP, which(d$CV), thr=thr)
    tmp.noord <- credset(d$PP, which(d$CV), do.order=FALSE,thr=thr)
    data.frame(order=c(TRUE,FALSE),
               thr=tmp.ord$thr,
               size=c(tmp.ord$size,tmp.noord$size),
               nvar=c(length(tmp.ord$credset),
                      length(tmp.noord$credset)),
               covered=c(tmp.ord$contained,
                         tmp.noord$contained))
  })
  cs <- do.call("rbind",cs)
  cs.ord <- subset(cs,cs$order==TRUE)
  cs.noord <- subset(cs,cs$order==FALSE)

  ## what was the coverage?
  c(size.ord=mean(cs.ord$size), cov.ord=mean(cs.ord$covered),
    size.noord=mean(cs.noord$size), cov.noord=mean(cs.noord$covered))
}
```

Example 4

This example uses the replicate and wrapper functions to perform 100 simulations on data with an odds ratio of 1.1 and a threshold of 0.5.

```
# only need one reference haplotype
freq <- simref()

results_0.5 <- replicate(100,wrapper(OR=1.1,thr=0.5))
results_0.5[,1:8] # shows the size and coverage (for ordered and non-ordered pps) for the first 8 repli
```

```
##                 [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## size.ord   0.5036894 0.5044828 0.5058999 0.5036019 0.5074345 0.5052644
## cov.ord    0.4000000 0.2000000 0.7000000 0.4000000 0.4000000 0.3000000
## size.noord 0.5047984 0.5048886 0.5090178 0.5169484 0.5067333 0.5043617
## cov.noord  0.4000000 0.4000000 0.4000000 0.3000000 0.5000000 0.8000000
##                 [,7]      [,8]
```

```
## size.ord   0.5056726 0.5044836
## cov.ord    0.6000000 0.3000000
## size.noord 0.5254775 0.5405220
## cov.noord  0.7000000 0.8000000
```

```
# confidence interval
se_0.5 <- apply(results_0.5,1,function(x) sd(x)/sqrt(length(x)))
mn_0.5 <- rowMeans(results_0.5)
cbind(average=mn_0.5, lci=mn_0.5 - 1.96*se_0.5, uci=mn_0.5 + 1.96*se_0.5)
```

```
##              average       lci       uci
## size.ord   0.5058017 0.5051074 0.5064960
## cov.ord    0.3920000 0.3560337 0.4279663
## size.noord 0.5161712 0.5133277 0.5190147
## cov.noord  0.5220000 0.4893238 0.5546762
```

Example 5

This example uses the replicate and wrapper functions to perform 100 simulations on data with an odds ratio of 1.1 and a threshold of 0.9.

```
results_0.9 <- replicate(100,wrapper(OR=1.1,thr=0.9))

# confidence interval
se_0.9 <- apply(results_0.9,1,function(x) sd(x)/sqrt(length(x)))
mn_0.9 <- rowMeans(results_0.9)
cbind(average=mn_0.9, lci=mn_0.9 - 1.96*se_0.9, uci=mn_0.9 + 1.96*se_0.9)
```

```
##              average       lci       uci
## size.ord   0.9029975 0.9028932 0.9031018
## cov.ord    0.8440000 0.8099582 0.8780418
## size.noord 0.9085798 0.9078133 0.9093464
## cov.noord  0.8910000 0.8707007 0.9112993
```

5. wrapper2

This function generates credible sets simultaneously for different ORs and sample sizes to allow analysis of how disorder affects the number of variants in a credible set. Disorder has been set as

$$disorder = -mean(log(pp)).$$

The output is a data.frame with 20 rows, since nrep is set to 10 in the `simdata` function and each simulation is ran for ordered and non-ordered posterior probabilities.

```
entropy <- function(p) -mean(log(p))

wrapper2 <- function(thr=0.5,...) {
  n <- sample(1:5,1)*1000
  or <- sample(c(1,1.05,1.1,1.2,1.3),1)
  data <- simdata(freq,N0=n,N1=n,OR=or) #,...) # data is a list of data.frames

  cs <- lapply(data, function(d) {
    tmp.ord <- credset(d$PP, which(d$CV), thr=thr)
    tmp.noord <- credset(d$PP, which(d$CV), do.order=FALSE,thr=thr)
    data.frame(order=c(TRUE,FALSE),
```

```
                thr=tmp.ord$thr,
                size=c(tmp.ord$size,tmp.noord$size),
                nvar=c(length(tmp.ord$credset),
                       length(tmp.noord$credset)),
                covered=c(tmp.ord$contained,
                          tmp.noord$contained))
  })
  cs <- do.call("rbind",cs)
  ent <- sapply(data, function(x) entropy(x$PP))
  cs$entropy <- rep(ent,each=2)
  cs$N <- n
  cs$OR <- or
  cs
}
```

Example 6

The following example uses wrapper2 to simulate 2000 credible sets (nrep=10 in the `simdata` function, so for each repetition in the `replicate` input, we simulate credible sets from 10 ordered and 10 non-ordered posterior probabilities). The input in the `replicate` function determines how many combinations of different OR and N values will be analysed.

```
covent <- replicate(100,wrapper2(), simplify=FALSE)
head(covent[[1]])
```

```
##   order thr      size nvar covered  entropy    N OR
## 1  TRUE 0.5 0.5046975   27   FALSE 4.787451 4000  1
## 2 FALSE 0.5 0.5113161   51   FALSE 4.787451 4000  1
## 3  TRUE 0.5 0.5005817   22   FALSE 4.834281 4000  1
## 4 FALSE 0.5 0.5067836   55   FALSE 4.834281 4000  1
## 5  TRUE 0.5 0.5042397   27   FALSE 4.769001 4000  1
## 6 FALSE 0.5 0.5059016   48    TRUE 4.769001 4000  1
```

```
covent <- do.call("rbind", covent) # put them together into a single data frame

# What is the coverage?
covent$covered <- as.numeric(covent$covered)
coverage <- mean(covent$covered)
coverage
```

```
## [1] 0.516
```

Over the 2000 simulations, 51.6% of credible sets contained the causal variant.

Example 7

Increasing the threshold to 0.9 increases the coverage to 89.85%.

```
covent_0.9 <- replicate(100,wrapper2(thr=0.9), simplify=FALSE)
head(covent_0.9[[1]])
```

```
##   order thr      size nvar covered  entropy    N OR
## 1  TRUE 0.9 0.9048761   78    TRUE 4.901073 1000  1
## 2 FALSE 0.9 0.9033825   89    TRUE 4.901073 1000  1
## 3  TRUE 0.9 0.9028159   82    TRUE 4.775833 1000  1
## 4 FALSE 0.9 0.9361678   92    TRUE 4.775833 1000  1
```

```
## 5  TRUE 0.9 0.9023523   83    TRUE 4.711884 1000  1
## 6 FALSE 0.9 0.9031046   92    TRUE 4.711884 1000  1
```

```
covent_0.9 <- do.call("rbind", covent_0.9) # put them together into a single data frame

# What is the coverage?
covent_0.9$covered <- as.numeric(covent_0.9$covered)
coverage_0.9 <- mean(covent_0.9$covered)
coverage_0.9
```

```
## [1] 0.8985
```

Analysis of coverage

Coverage is defined as the proportion of simulations for which the causal variant is in the credible set. It is frequentist in nature as it describes long run behaviour. Coverage is closely related to the threshold of the credible set, which is a value set by the user describing the value of which the cumulative posterior probability of the credible set must exceed. Undercoverage/overcoverage refer to the case where more/less SNPs need to be added to the credible set, in order to achieve appropriate coverage.

In most cases, it is beneficial to sort posterior probabilities in descending order prior to credible set analysis. However, ordering does not always improve coverage. For example, if all SNP posterior probabilities are equal, then ordering does not provide the extra information which it is thought to, and undercoverage occurs.

Hope has shown in her thesis that undercoverage occurs for ordered sets at low OR values (1-1.1) and that overcoverage occurs for ordered sets at high OR values (1.2-1.5). She has also shown that undercoverage occurs for ordered sets with low sample sizes (N=1000) and that overcoverage occurs for ordered sets with high samples sizes (N=5000, 10000).

The `wrapper2` function has been used to simulate credible sets with varying sample sizes and odds ratios to further analyse their relationship with coverage for ordered and non-ordered credible sets.

```
library(data.table)
cs_cov1000 <- replicate(1000,wrapper2(), simplify=FALSE) # obtain 1000*20=20000 credible sets
cs_cov1000 <- do.call("rbind", cs_cov1000)

DT <- data.table(cs_cov1000)

DT_ord <- DT[order=="TRUE"]
DT_noord <- DT[order=="FALSE"]

DT_ord_coverage <- DT_ord[,mean(covered),by=c("N","OR")]
DT_noord_coverage <- DT_noord[,mean(covered),by=c("N","OR")]
```

The plots clearly show that undercoverage occurs for ordered sets with low ORs (even at high sample sizes). It also seems that for low OR values in non-ordered sets, coverage is fairly consistant regardless of sample size. However, in ordered sets, coverage increases in a more predictable manner for each value of OR and sample size. We hope to exploit this relationship of sample size and OR on coverage for ordered credible sets to correct for over- and under- coverage.

Analysis of the relationship of coverage and entropy dependent on sample size and OR

$$disorder = -mean(log(pp))$$