Simulating credible sets (sup. meeting 3)

Anna Hutchinson

16/10/2018

---

1) Update the simref function to simulate reference haplotypes

```
library(bindata)

# A function to simulate reference haplotypes, based on more realistic LD data

ref <- function(nsnps=100,nhaps=1000,LD1=0.08,LD2=0.02,maf=0.1){ # LD1 is value
                                                                 # eitherside of lead diagonal
  R<-diag(1,nsnps)                                               # LD2 is next diagonal out
  R[cbind(1:(nsnps-1),2:nsnps)] <- LD1
  R[cbind(2:nsnps,1:(nsnps-1))] <- LD1
  R[cbind(1:(nsnps-2),3:nsnps)] <- LD2
  R[cbind(3:nsnps,1:(nsnps-2))] <- LD2
  x=rmvbin(nhaps, rep(maf,nsnps), bincorr=R)
  x=x+1
  x=as.data.frame(x)
  snps <- colnames(x) <- paste0("s",1:nsnps)
  x$Probability <- 1/nrow(x)
  x
}
```

---

2) Update the simdata and wrapper functions to use these new reference haplotypes

```
# amend simdata function

simdata_x <- function(x,OR=1.5,nrep=100,N0=1000,N1=1000,thr=0.5) {
  snps <- colnames(x)[-ncol(x)]
  CV=sample(snps,1)

  FP <- make_GenoProbList(snps=snps,W=CV,freq=x)
  zsim <- simulated_z_score(N0=N0, # number of controls
                            N1=N1, # number of cases
                            snps=snps,
                            W=CV,
                            gamma.W=log(OR),
                            freq=x,# 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(x[,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
}

# amend wrapper function

wrapper_x <- function(thr=0.9,...) {
  data <- simdata_x(x) # 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$contained))
  })
  cs <- do.call("rbind",cs)
  cs.ord <- subset(cs,cs$order==TRUE)
  cs.noord <- subset(cs,cs$order==FALSE)

  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))
}
```

---

3) Update wrapper2 so that it incorporates the new reference haplotypes and varies theshold values

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

wrapper2_x <- function(...) {
  n <- sample(1:5,1)*1000 # vary sample size
  or <- sample(c(1,1.05,1.1,1.2,1.3),1) # vary or
  thr <- sample(c(0.5,0.6,0.7,0.8,0.9),1) # vary threshold
  data <- simdata_x(x,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$OR <- or
  cs$thr <- thr
  cs
}
# note, still need x <- ref() command prior to use
```

Wrapper2_x() algorithm

1) Randomly sample values for threshold, OR and N.
2) Generate 100 'systems' (posterior probability systems) using these values.
3) For each system, generate one credible set using ordered posterior probabilities and one credible set using non-ordered posterior probabilities, obtaining 200 credible sets from 100 systems.
4) For each credible set calculate the sum of posterior probabilities of elements in the credible set (size), number of variables in the credible set (nvar), whether the causal variant is contained in the credible set (covered) and the entropy. Note that the entropy is the same for each system, resulting in only 100 values.
5) Replicate to repeat the process using different combinations of threshold, OR and N values.

Example simulation

The following code simulates 20,000 credible sets from 100 different combinations of threshold, OR and N values. For each of these combinations, 100 systems are considered and two credible sets derived from each of these (one using ordered pps and one using non-ordered pps).

```
x <- ref()
example_sim <- replicate(100,wrapper2_x(), simplify=FALSE)
example_sim <- data.table::rbindlist(example_sim)
dim(example_sim)
```

```
## [1] 20000     8
```

```
# split data into ordered and non-ordered sets
example_sim_ord <- example_sim[order=="TRUE"]
example_sim_noord <- example_sim[order=="FALSE"]
head(example_sim_ord)
```

```
##    order thr      size nvar covered  entropy    N OR
## 1:  TRUE 0.8 0.8049165   62    TRUE 4.824014 3000  1
## 2:  TRUE 0.8 0.8023256   44   FALSE 5.272261 3000  1
```

```
## 3:  TRUE 0.8 0.8024073   58    TRUE 4.882744 3000  1
## 4:  TRUE 0.8 0.8038594   54   FALSE 5.064989 3000  1
## 5:  TRUE 0.8 0.8008082   65    TRUE 4.842719 3000  1
## 6:  TRUE 0.8 0.8035134   63   FALSE 4.841289 3000  1
```

```
tail(example_sim_ord)
```

```
##    order thr      size nvar covered  entropy    N OR
## 1:  TRUE 0.5 0.5945104    1   FALSE 5.663294 2000  1
## 2:  TRUE 0.5 0.5093206   14   FALSE 4.999599 2000  1
## 3:  TRUE 0.5 0.5146684    7   FALSE 5.173867 2000  1
## 4:  TRUE 0.5 0.5007760   12   FALSE 5.035659 2000  1
## 5:  TRUE 0.5 0.5004056   17   FALSE 4.918003 2000  1
## 6:  TRUE 0.5 0.5082255   12   FALSE 5.063214 2000  1
```

```
head(example_sim_noord)
```

```
##    order thr      size nvar covered  entropy    N OR
## 1: FALSE 0.8 0.8018651   79    TRUE 4.824014 3000  1
## 2: FALSE 0.8 0.8025838   66   FALSE 5.272261 3000  1
## 3: FALSE 0.8 0.8034538   84    TRUE 4.882744 3000  1
## 4: FALSE 0.8 0.9015286   84    TRUE 5.064989 3000  1
## 5: FALSE 0.8 0.8080307   77    TRUE 4.842719 3000  1
## 6: FALSE 0.8 0.8021864   82    TRUE 4.841289 3000  1
```

```
tail(example_sim_noord)
```

```
##    order thr      size nvar covered  entropy    N OR
## 1: FALSE 0.5 0.6633741   17   FALSE 5.663294 2000  1
## 2: FALSE 0.5 0.5088322   65    TRUE 4.999599 2000  1
## 3: FALSE 0.5 0.5006280   69    TRUE 5.173867 2000  1
## 4: FALSE 0.5 0.5060451   51    TRUE 5.035659 2000  1
## 5: FALSE 0.5 0.5028097   49    TRUE 4.918003 2000  1
## 6: FALSE 0.5 0.5050030   38   FALSE 5.063214 2000  1
```