

Package ‘hiPOD’

July 2, 2014

Type Package

Title hierarchical Pooled Optimal Design

Version 1.0

Date 2012-04-27

Author Wei E. Liang

Maintainer Wei E. Liang <liang1@usc.edu>

Description Based on hierarchical modeling, this package provides a few practical functions to find and present the optimal designs for a pooled NGS design.

Depends rgl

License GPL-3

LazyLoad yes

Repository CRAN

Date/Publication 2012-06-01 14:26:57

NeedsCompilation no

R topics documented:

hiPOD-package	2
diffVariantError	4
FindOptPower	5
ncp.pool.pred	7
PlotOptPower	7
predictPoolNCP	11
probDetEqual3	12
ShowOptDesign	13
XmeanGivenCost	14

Index	16
--------------	-----------

 hiPOD-package

hierarchical Pooled Optimal Design

Description

Based on hierarchical modeling, this package provides a few practical functions to find and present the optimal designs for a pooled NGS study.

Details

```

Package:    hiPOD
Type:      Package
Version:    1.0
Date:      2012-04-27
License:    GPL
LazyLoad:  yes
  
```

Author(s)

Wei Liang Maintainer: Wei Liang <liang1@usc.edu>

See Also

[FindOptPower](#), [PlotOptPower](#), [ShowOptDesign](#)

Examples

```

#####
##### Example 1: A simple example, with very rough grid points (only 20X20 grid points)
#####

##### Find the optimal design
example.1 <- FindOptPower(cost=452915, sample.size=5000, MAF=0.03, OR=2, error=0.01, upper.P=200, Number.Grids=20)

##### assign a directory to store the contour plots
##### with your own choice
proj.Dir <- paste(getwd(), "/hiPOD_examples", sep="")
if(!file.exists(proj.Dir)) dir.create(proj.Dir)

##### Inferences on the optimal designs
PlotOptPower(example.1, save.contour=TRUE, contour.filename=paste(proj.Dir, "/example1_contour.bmp", sep=""))

# # snapshot3d(filename = paste(proj.Dir, "/example1_3d.bmp", sep=""))
ShowOptDesign(example.1, 5)
  
```

```
write.csv(ShowOptDesign(example.1), paste(proj.Dir, "/example1.csv", sep=""))

##### Example 2: A realist example comparing with individual sequencing with 200 individuals 4X each
##### Now varying the CostPerX with other parameters fixed
##### NOTE:: It might take a long time as 6 design questions are investigated

##### Find the optimal design
example.2 <- FindOptPower(cost=452915, sample.size=5000, MAF=0.03, OR=2, error=0.01)

##### assign a directory to store the contour plots
##### with your own choice
proj.Dir <- paste(getwd(), "/hiPOD_examples", sep="")
if(!file.exists(proj.Dir)) dir.create(proj.Dir)

##### Inferences on the optimal designs
PlotOptPower(example.2, save.contour=TRUE, contour.filename=paste(proj.Dir, "/example2_contour.bmp", sep=""))

# # snapshot3d(filename = paste(proj.Dir, "/example2_3d.bmp", sep=""))

ShowOptDesign(example.2)
write.csv(ShowOptDesign(example.2), paste(proj.Dir, "/example2.csv", sep=""))

##### Example 3: comparing with individual sequencing with 200 individuals 4X each
##### Now varying the CostPerX with other parameters fixed
##### NOTE:: It might take a long time as 6 design questions are investigated

##### specify the cost function
# CostPerExp.3 <- 10000
# CostPerPool.3 <- 1000
# CostPerX.3.range <- c(50,100,200,300,400,500)
#####

##### assign a directory to store the contour plots
##### with your own choice
# proj.Dir <- paste(getwd(), "/hiPOD_examples", sep="")
# if(!file.exists(proj.Dir)) dir.create(proj.Dir)

# top.choices.3 <- data.frame()
# for(CostPerX.3 in CostPerX.3.range)
# {
# cost.3 <- CostPerExp.3 + CostPerPool.3*200 + CostPerX.3*200*4

##### Find the optimal design
# example.3 <- FindOptPower(cost=cost.3, sample.size=5000, MAF=0.03, OR=2, error=0.01, costPerExp=CostPerExp.3,
```

```
# ##### Inferences on the optimal designs
# PlotOptPower(example.3, save.contour=TRUE, contour.filename=paste(proj.Dir, "/example3_contour_CostPerX=", Co

# temp <- ShowOptDesign(example.3, 10)
# temp <- cbind(CostPerX.3, temp)
# top.choices.3 <- rbind(top.choices.3, temp)
# }
# write.csv(top.choices.3, paste(proj.Dir, "/example3.csv", sep=""))
```

diffVariantError *differentiate a variant from sequencing error*

Description

This function finds a practical threshold to differentiate a variant call from sequencing error.

Usage

```
diffVariantError(Xmean, N.p, error, N.test = 1)
```

Arguments

Xmean	The average coverage on the pool
N.p	The pool size: number of individuals per pool
error	Sequencing error rate
N.test	Number of tests, usually the same as number of pools P

Details

It is a helper function to calculate the probability of detection.

Value

diffVariantError() returns a vector $c(v, p0, p1)$, where v is the threshold for a variant call, $p0$ is the false discovery rate and $p1$ is the lower bound of true discovery rate.

Author(s)

Wei E. Liang

Examples

```

##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.

## The function is currently defined as
function(Xmean, N.p, error, N.test=1)
{
theta <- (1-error)/(2*N.p) + error*(1-1/(2*N.p))

for(v in 1:Xmean)
{
p1 <- pbinom(v-1, Xmean, theta, lower.tail=FALSE)
p0 <- pbinom(v-1, Xmean, error, lower.tail=FALSE)

if(p1/p0 > 5 & p0<0.05/N.test) break;
}

c(v, p0, p1)
}

```

FindOptPower

search for the optimal pooled design

Description

Perform a grid search over potential design space, and find the predicted power and validity of the designs.

Usage

```
FindOptPower(cost, sample.size, MAF, OR, error, costPerExp = 18915, costPerPool = 970, costPerX = 300,
```

Arguments

cost	cost constraint of the study
sample.size	sample size constraint of the study
MAF	assumed MAF of the variant of interest
OR	assume effect size (odds ratio) of the variant of interest
error	assume sequencing error rate
costPerExp	cost per experiment
costPerPool	cost per pool
costPerX	cost per 1X coverage
lower.P	lower bound of number of pools in potential consideration
upper.P	upper bound of number of pools in potential consideration

lower.N.p	lower bound of number of pool size in potential consideration
upper.N.p	upper bound of number of pool size in potential consideration
lower.Xmean	lower bound of number of coverage per pool in potential consideration
upper.Xmean	upper bound of number of coverage per pool in potential consideration
sig.level	significance level of the statistic test, usually 0.05 for a single test
Number.Grids	number of grids in the search space, preset as 100

Details

Given the research question and the study constraints, this function calculates the power and validity of all the potential pooled designs.

Value

Returns a list:

```
cost
sample.size
constraint.set

scenario.set
designs          the potential designs, validity and power
```

Author(s)

Wei E. Liang

See Also

[PlotOptPower](#), [ShowOptDesign](#)

Examples

```
#####
##### Example 1: A simple example, with very rough grid points (only 20X20 grid points)
#####

##### Find the optimal design
example.1 <- FindOptPower(cost=452915, sample.size=5000, MAF=0.03, OR=2, error=0.01, upper.P=200, Number.Grids=200)

##### assign a directory to store the contour plots
##### with your own choice
proj.Dir <- paste(getwd(), "/hiPOD_examples", sep="")
if(!file.exists(proj.Dir)) dir.create(proj.Dir)

##### Inferences on the optimal designs
PlotOptPower(example.1, save.contour=FALSE, plot.3d=FALSE)
# # snapshot3d(filename = paste(proj.Dir, "/example1_3d.bmp", sep=""))
```

```
ShowOptDesign(example.1, 5)
```

ncp.pool.pred	<i>prediction model of the non-centrality parameter from simulation results</i>
---------------	---

Description

the prediction model, summarized from the simulation results

Usage

```
data(ncp.pool.pred)
```

Format

The format is: List of 2 \$: num [1:4] 0.02 0.05 0.1 1 \$:List of 4 ..\$: Named num [1:29] -1.067976 19.704888 1.217199 -2.612229 -0.000886 attr(*, "names")= chr [1:29] "(Intercept)" "MAF" "log(OR)" "error"\$: Named num [1:29] -1.11889 4.47646 1.57193 -2.83448 -0.00108 attr(*, "names")= chr [1:29] "(Intercept)" "MAF" "log(OR)" "error"\$: Named num [1:29] -1.152848 2.57742 1.816014 -1.521528 -0.000948 attr(*, "names")= chr [1:29] "(Intercept)" "MAF" "log(OR)" "error"\$: Named num [1:29] -1.17164 0.65682 1.89042 -0.81605 -0.00127 attr(*, "names")= chr [1:29] "(Intercept)" "MAF" "log(OR)" "error" ...

Examples

```
data(ncp.pool.pred)
## maybe str(ncp.pool.pred) ; plot(ncp.pool.pred) ...
```

PlotOptPower	<i>Plots a contour plot / 3d plot of the power of the potential designs.</i>
--------------	--

Description

Based on the predicted values from FindOptPower, plot the 3d or contour figures. Depends on the package rgl for 3d plots.

Usage

```
PlotOptPower(opt.design.results, save.contour = FALSE, contour.filename = NA, plot.3d = TRUE)
```

Arguments

opt.design.results

save.contour Whether or not save the contour plot

contour.filename
 The name (including the path) of the contour plot

plot.3d Whether or not plot 3d version.

Value

Contour plot of power 3d plot of power, where the design parameters are plotted as X, Y and Z, and power is presented by the color.

Author(s)

Wei E. Liang

See Also

[FindOptPower](#), [ShowOptDesign](#)

Examples

```
##### Example 1: A simple example, with very rough grid points (only 20X20 grid points)
##### Find the optimal design
example.1 <- FindOptPower(cost=452915, sample.size=5000, MAF=0.03, OR=2, error=0.01, upper.P=200, Number.Grids=200)

##### assign a directory to store the contour plots
##### with your own choice
proj.Dir <- paste(getwd(), "/hiPOD_examples", sep="")
if(!file.exists(proj.Dir)) dir.create(proj.Dir)

##### Inferences on the optimal designs
PlotOptPower(example.1, save.contour=FALSE, plot.3d=FALSE)

## with 3d plots
PlotOptPower(example.1, save.contour=FALSE, plot.3d=TRUE)

## The function is currently defined as
function(opt.design.results, save.contour=FALSE, contour.filename=NA, plot.3d=TRUE)
{
  cost <- opt.design.results[[1]]
```



```

sample.size <- opt.design.results[[2]]
constraint.set <- opt.design.results[[3]]
scenario.set <- opt.design.results[[4]]
newdata <- opt.design.results[[5]]

newdata.P <- sort(unique(newdata$P))
newdata.N.p <- sort(unique(newdata$N.p))

pred.power <- matrix(newdata$pred.power, nrow=length(newdata.P), ncol=length(newdata.N.p))

sample.good <- subset(newdata, subset=(newdata$is.valid.design & newdata$upper.sample.good), select=c(P,N.p))
cost.good <- subset(newdata, subset=(newdata$Xmean.good), select=c(P, N.p))

sample.N.p.temp <- sort(unique(sample.good[,2]))
cost.N.p.temp <- sort(unique(cost.good[,2]))

for(cur.N.p in sample.N.p.temp)
{
  sample.P.min.temp <- min(subset(sample.good, subset=(N.p==cur.N.p))$P)
  sample.P.max.temp <- max(subset(sample.good, subset=(N.p==cur.N.p))$P)
  if(cur.N.p == sample.N.p.temp[1])
  {
    sample.P.min <- sample.P.min.temp
    sample.P.max <- sample.P.max.temp
  }
  else
  {
    sample.P.min <- c(sample.P.min, sample.P.min.temp)
    sample.P.max <- c(sample.P.max, sample.P.max.temp)
  }
}

for(cur.N.p in cost.N.p.temp)
{
  cost.P.min.temp <- min(subset(cost.good, subset=(N.p==cur.N.p))$P)
  cost.P.max.temp <- max(subset(cost.good, subset=(N.p==cur.N.p))$P)
  if(cur.N.p == sample.N.p.temp[1])
  {
    cost.P.min <- cost.P.min.temp
    cost.P.max <- cost.P.max.temp
  }
  else
  {
    cost.P.min <- c(cost.P.min, cost.P.min.temp)
    cost.P.max <- c(cost.P.max, cost.P.max.temp) }
}

sample.N.p <- c(sample.N.p.temp, rev(sample.N.p.temp), sample.N.p.temp[1])
sample.P <- c(sample.P.min, rev(sample.P.max), sample.P.min[1])
cost.N.p <- c(cost.N.p.temp, rev(cost.N.p.temp), cost.N.p.temp[1])

```

```

cost.P <- c(cost.P.min, rev(cost.P.max), cost.P.min[1])

## plot all the constraints:
## P*N.p>=lower.sample ==> ln(N.p)>=ln(lower.sample)-ln(P)
## P*N.p<=sample.size ==> ln(N.p)<=ln(sample.size)-ln(P)
## Xmean, P, N.p satisfies the cost function

if(save.contour==TRUE)
{
  bmp(filename=contour.filename, width=720,height=720)
}

#####
# # # # # Color Prints!!!
#####
# # {
# # filled.contour(x=log(newdata.N.p), y=log(newdata.P), pred.power, plot.title = title(main = "Grid Search of Op
# # }

#####
# # # # # Grey Prints!!!
#####
filled.contour(x=log(newdata.N.p), y=log(newdata.P), pred.power, plot.title = title(main = "Grid Search of Optim

description <- bquote(paste("Scenario: VAF=", .(as.numeric(scenario.set["MAF"])), " OR=", .(round(as.numeric(scen

mtext(description, adj=0, padj=-0.5 )

if(save.contour==TRUE)
{
  dev.off()
}

if(plot.3d==TRUE)
{
  ln.Xmean <- matrix(log(newdata$Xmean), nrow=length(newdata.P), ncol=length(newdata.N.p))
  pred.power.3d <- newdata$pred.power
  ## Set-up the rgl device
  plot3d(log(newdata.N.p), log(newdata.P), ln.Xmean, type = "n", xlab="ln(Np)", ylab="ln(P)", zlab="ln(lambda)")

  ## Need a scale for pred.power to display as colours
  ## Here I choose 25 equally spaced colours from a palette
  cols <- terrain.colors(25)

  ## Break pred.power into 25 equal regions
  cuts <- cut(pred.power.3d, breaks = 25)

```

```

## Add in the surface, colouring by pred.power
surface3d(log(newdata.N.p), log(newdata.P), ln.Xmean, color = cols[cuts], shininess=80)

## refine the vector x, with length(x)>1
RefineVector <- function(x, num)
{
  x.origin <- x[-length(x)]
  x.diff <- diff(x)/num
  for(i in 1:(num-1))
  {
    x.temp <- x.origin+i*x.diff
    if(i==1) x.final <- rbind(x.origin, x.temp) else x.final <- rbind(x.final, x.temp)
  }
  c(as.vector(x.final), x[length(x)])
}
for(cur.z in 1:length(sample.N.p))
{
  index.temp <- with(newdata, which(P==sample.P[cur.z] & N.p==sample.N.p[cur.z]))
  sample.points.z.temp <- newdata[index.temp, "Xmean"]
  if(cur.z==1) sample.points.z <- sample.points.z.temp
  else sample.points.z <- c(sample.points.z, sample.points.z.temp)
}
lines3d(RefineVector(log(sample.N.p),10), RefineVector(log(sample.P),10), RefineVector(log(sample.points.z),10))
for(cur.z in 1:length(cost.N.p))
{
  index.temp <- with(newdata, which(P==cost.P[cur.z] & N.p==cost.N.p[cur.z]))
  cost.points.z.temp <- newdata[index.temp, "Xmean"]
  if(cur.z==1) cost.points.z <- cost.points.z.temp
  else cost.points.z <- c(cost.points.z, cost.points.z.temp)
}
lines3d(log(cost.N.p), log(cost.P), log(cost.points.z)+0.1, color=4, alpha=0.6, lwd=8)
bg3d(color="snow")
title3d(main="3D optimal power")
}

}

```

predictPoolNCP

The prediction model of non-centrality parameter

Description

It is based on our simulation results.

Usage

```
predictPoolNCP(MAF, OR, error, P, N.p, Xmean)
```

Arguments

MAF	minor allele frequency
OR	odds ratio
error	sequencing error
P	number of pools (case+control)
N.p	number of individuals per pool
Xmean	average coverage per pool

Author(s)

Wei E. Liang

Examples

```
## The function is currently defined as
function(MAF, OR, error, P, N.p, Xmean){
  count <- 1
  while(ncp.pool.pred[[1]][count] <= MAF) count<-count+1

  ncp.pool.pred.coef <- ncp.pool.pred[[2]][count]

  covariates <- c(1, MAF, log(OR), error, P, Xmean, N.p, (P^(1/3)), (Xmean^(1/3)), (N.p^(1/3)), (log(OR)^2), MAF*log(OR))

  ncp.pool.predicted <- (sum(ncp.pool.pred.coef * covariates))^3

  ncp.pool.predicted
}
```

probDetEqual3

Probability of Detection

Description

Find the probability of detection.

Usage

```
probDetEqual3(MAF, Xmean, T, N.p, error)
```

Arguments

MAF	
Xmean	
T	The threshold, usually calculated from diffVariantError()
N.p	
error	

Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.

## The function is currently defined as
function(MAF, Xmean, T, N.p, error)
{
  pDetEqual <- 0;
  for(n in 0:(2*N.p))
  {
    pConf <- dbinom(n,2*N.p,MAF);
    pDet.not <- 0;
    theta <- (1-error)*n/(2*N.p) + error*(1-n/(2*N.p))

    pDet <- pbinom(T-1, Xmean, theta, lower.tail=FALSE)

    pDetEqual <- pDetEqual + pConf*pDet;
  }
  pDetEqual
}
```

ShowOptDesign

Print the top choices of designs

Description

Show the top [num.designs] choices of valid designs.

Usage

```
ShowOptDesign(opt.design.results, num.designs = 10)
```

Arguments

opt.design.results

num.designs

Value

a dataframe including the top choices.

Author(s)

Wei E. Liang

See Also

[FindOptPower](#), [PlotOptPower](#)

Examples

```
##### Example 1: A simple example, with very rough grid points (only 20X20 grid points)
##### Find the optimal design
example.1 <- FindOptPower(cost=700000, sample.size=5000, MAF=0.03, OR=2, error=0.01, upper.P=200, Number.Grids=5000)

##### assign a directory to store the contour plots
##### with your own choice
proj.Dir <- paste(getwd(), "/hiPOD_examples", sep="")
if(!file.exists(proj.Dir)) dir.create(proj.Dir)

##### Inferences on the optimal designs
PlotOptPower(example.1, save.contour=FALSE, plot.3d=FALSE)

ShowOptDesign(example.1, 5)
ShowOptDesign(example.1, 10)

## The function is currently defined as
function(opt.design.results, num.designs=10)
{
  designs.good <- subset(opt.design.results[[5]], subset=(is.valid.design & upper.sample.good & Xmean.good))
  head(designs.good[order(-designs.good$pred.power), ], num.designs)
}
```

XmeanGivenCost

A function used to find best coverage given cost and other constraints

Description

A function used to find best coverage given cost and other constraints

Usage

```
XmeanGivenCost(costPar, cost, P, lower.Xmean, upper.Xmean)
```

Arguments

costPar
cost
P
lower.Xmean
upper.Xmean

Author(s)

Wei E. Liang

Examples

```
##---- Should be DIRECTLY executable !! ----  
##-- ==> Define data, use random,  
##--or do help(data=index) for the standard data sets.  
  
## The function is currently defined as  
function(costPar, cost, P, lower.Xmean, upper.Xmean)  
{  
  
  CostPerExp <- costPar[1]  
  CostPerPool <- costPar[2]  
  CostPerX <- costPar[3]  
  
  cost <- rep(cost, length(P))  
  
  Xmean.max <- floor((cost-CostPerExp-CostPerPool*P)/(CostPerX*P))  
  Xmean.good <- (Xmean.max >= rep(lower.Xmean, length(Xmean.max)))  
  
  Xmean.max <- ifelse(Xmean.max<lower.Xmean, lower.Xmean, Xmean.max)  
  Xmean.max <- ifelse(Xmean.max>upper.Xmean, upper.Xmean, Xmean.max)  
  
  list(Xmean.good, Xmean.max)  
}
```

Index

*Topic **\textasciitildekwd1**

diffVariantError, 4
FindOptPower, 5
PlotOptPower, 7
predictPoolNCP, 11
probDetEqual3, 12
ShowOptDesign, 13
XmeanGivenCost, 14

*Topic **\textasciitildekwd2**

diffVariantError, 4
FindOptPower, 5
PlotOptPower, 7
predictPoolNCP, 11
probDetEqual3, 12
ShowOptDesign, 13
XmeanGivenCost, 14

*Topic **datasets**

ncp.pool.pred, 7

*Topic **package**

hiPOD-package, 2

diffVariantError, 4

FindOptPower, 2, 5, 8, 14

hiPOD (hiPOD-package), 2

hiPOD-package, 2

ncp.pool.pred, 7

PlotOptPower, 2, 6, 7, 14

predictPoolNCP, 11

probDetEqual3, 12

ShowOptDesign, 2, 6, 8, 13

XmeanGivenCost, 14