Package ‘crso’

October 12, 2022

Title Cancer Rule Set Optimization (‘crso’)
Version 0.1.1
Author Michael Klein <michael.klein@yale.edu>
Maintainer Michael Klein <michael.klein@yale.edu>
Description An algorithm for identifying candidate driver combinations in cancer. CRSO is based on a theoretical model of cancer in which a cancer rule is defined to be a collection of two or more events (i.e., alterations) that are minimally sufficient to cause cancer. A cancer rule set is a set of cancer rules that collectively are assumed to account for all of ways to cause cancer in the population. In CRSO every event is designated explicitly as a passenger or driver within each patient. Each event is associated with a patient-specific, event-specific passenger penalty, reflecting how unlikely the event would have happened by chance, i.e., as a passenger. CRSO evaluates each rule set by assigning all samples to a rule in the rule set, or to the null rule, and then calculating the total statistical penalty from all unassigned event. CRSO uses a three phase procedure find the best rule set of fixed size K for a range of Ks. A core rule set is then identified from among the best rule sets of size K as the rule set that best balances rule set size and statistical penalty.

Users should consult the ‘crso’ vignette for an example walk through of a full CRSO run. The full description, of the CRSO algorithm is presented in:


Please cite this article if you use ‘crso’.

Depends R (>= 3.5.0), foreach
Imports stats, utils
License GPL-2
Encoding UTF-8
LazyData true
RoxygenNote 6.1.1
Suggests knitr, rmarkdown
VignetteBuilder knitr
**buildRuleLibrary**

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2019-07-07 17:00:03 UTC

---

**R topics documented:**

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>buildRuleLibrary</td>
<td>2</td>
</tr>
<tr>
<td>evaluateListOfIMs</td>
<td>3</td>
</tr>
<tr>
<td>getBestRsList</td>
<td>4</td>
</tr>
<tr>
<td>getCoreK</td>
<td>4</td>
</tr>
<tr>
<td>getCoreRS</td>
<td>5</td>
</tr>
<tr>
<td>getGCDs</td>
<td>6</td>
</tr>
<tr>
<td>getGCEs</td>
<td>6</td>
</tr>
<tr>
<td>getGCRs</td>
<td>7</td>
</tr>
<tr>
<td>getPoolSizes</td>
<td>7</td>
</tr>
<tr>
<td>getRulesAsStrings</td>
<td>8</td>
</tr>
<tr>
<td>makeFilteredImList</td>
<td>9</td>
</tr>
<tr>
<td>makePhaseOneOrderedRM</td>
<td>9</td>
</tr>
<tr>
<td>makePhaseThreeImList</td>
<td>10</td>
</tr>
<tr>
<td>makePhaseTwoImList</td>
<td>11</td>
</tr>
<tr>
<td>makeSubCoreList</td>
<td>12</td>
</tr>
<tr>
<td>skcm.list</td>
<td>13</td>
</tr>
</tbody>
</table>

---

**Index**

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>skcm.list</td>
<td>13</td>
</tr>
</tbody>
</table>

---

**Description**

Make full rule library of all rules that satisfy minimum coverage threshold.

**Usage**

`buildRuleLibrary(D, rule.thresh, min.epr)`

**Arguments**

- **D**: Binary matrix of N events and M samples
- **rule.thresh**: Minimum fraction of rules covered. Default is .03
- **min.epr**: Minimum events per rule. Default is 2.
evaluateListOfIMs

Evaluate list of rule set matrices

Usage

evaluateListOfIMs(D, Q, rm, im.list)

Arguments

D binary matrix of events by samples
Q penalty matrix of events by samples
rm matrix of rules ordered by phase one
im.list list of rule set matrices

Value

list of Js for each rule set matrix

Examples

library(crso)
data(skm)
list2env(skm.list,envir=globalenv())
rm.full <- buildRuleLibrary(D,rule.thresh = 0.05) # build rule library
dim(rm.full) # Should be matrix with dimension 60 x 71

Q <- log10(P)
rm.full <- buildRuleLibrary(D,rule.thresh = 0.05) # Rule library matrix, dimension: 60 x 71
p2.im.list <- makePhaseTwoImList(D,Q,rm.full,k.max = 3,pool.sizes=c(60,20,20),max.stored=100,
shouldPrint = TRUE)
p2.performance.list <- evaluateListOfIMs(D,Q,rm.full,p2.im.list)
getBestRsList  
Get list of best rule sets of size K for all K

Description
Get list of best rule sets of size K for all K

Usage
getBestRsList(rm, tpl, til)

Arguments
rm  binary rule matrix
tpl  list of top performances
til  list of top rule set index matrices

Examples
library(crso)
data(skcm)
list2env(skcm.list, envir = globalenv())
Q <- log10(P)
rm.full <- buildRuleLibrary(D, rule.thresh = 0.05) # Rule library matrix, dimension: 60 x 71
til.p2 <- makePhaseTwoImList(D, Q, rm.full, k.max = 3, pool.sizes = c(60, 20, 20),
                           max.stored = 100, shouldPrint = FALSE)
tpl.p2 <- evaluateListOfIMs(D, Q, rm.full, til.p2)
best.rs.list <- getBestRsList(rm = rm.full, tpl = tpl.p2, til = til.p2)

gerCoreK  
Determine core K from phase 3 tpl and til

Description
Determine core K from phase 3 tpl and til

Usage
getCoreK(D, rm, tpl, til, cov.thresh, perf.thresh)

Arguments
D  input matrix D
rm  binary rule matrix
tpl  list of top performances
til  list of top rule set index matrices
cov.thresh  core coverage threshold, default is 95
perf.thresh  core performance threshold, default is 90
getCoreRS

Examples

```r
library(crso)
data(skcm)
list2env(skcm.list,envir=globalenv())
Q <- log10(P)
rm.full <- buildRuleLibrary(D,rule.thresh = 0.05) # Rule library matrix, dimension: 60 x 71
til.p2 <- makePhaseTwoImlist(D,Q,rm.full,k.max = 3,pool.sizes=c(60,20,20),
max.stored=100,shouldPrint = FALSE)
tpl.p2 <- evaluateListOfIMs(D,Q,rm.full,til.p2)
core.K <- getCoreK(D,rm.full,tpl.p2,til.p2)
# core.K should be 3 almost always for this example, can run a few time to confirm
```

---

getCoreRS

Get core rules from phase 3 tpl and til

Description

Get core rules from phase 3 tpl and til

Usage

```r
getCoreRS(D, rm, tpl, til, cov.thresh, perf.thresh)
```

Arguments

- **D**: input matrix D
- **rm**: binary rule matrix
- **tpl**: list of top performances
- **til**: list of top rule set index matrices
- **cov.thresh**: core coverage threshold, defaults is 95
- **perf.thresh**: core performance threshold, default is 90

Examples

```r
library(crso)
data(skcm)
list2env(skcm.list,envir=globalenv())
Q <- log10(P)
rm.full <- buildRuleLibrary(D,rule.thresh = 0.05) # Rule library matrix, dimension: 60 x 71
til.p2 <- makePhaseTwoImlist(D,Q,rm.full,k.max = 3,pool.sizes=c(60,20,20),
max.stored=100,shouldPrint = FALSE)
tpl.p2 <- evaluateListOfIMs(D,Q,rm.full,til.p2)
core.rs <- getCoreRS(D,rm.full,tpl.p2,til.p2) # core.rs should be r1, r2, r3
```
getGCDs

Get Generalized Core Duos

Description
Get Generalized Core Duos

Usage
getGCDs(list.subset.cores)

Arguments
list.subset.cores
list of subset cores

Examples
list.subset.cores <- list(c("A.B.C","D.E","A.D"),
c("A.C","B.C.D","D.E"),
c("A.B.C","D.E"),c("A.B.C","D.E","B.C.D"))
gETCHs(list.subset.cores) # Confidence column should be 100, 100, 100, 75, 50, 25, 25

getGCEs

Get Generalized Core Events

Description
Get Generalized Core Events

Usage
getGCEs(list.subset.cores)

Arguments
list.subset.cores
list of subset cores

Examples
list.subset.cores <- list(c("A.B.C","D.E","A.D"),
c("A.C","B.C.D","D.E"),c("A.B.C","D.E"),c("A.B.C","D.E","B.C.D"))
gETCHes(list.subset.cores) # Confidence column should be 100, 100, 100, 100, 100
getGCRs

Get Generalized Core Rules

Description
Get Generalized Core Rules

Usage
getGCRs(list.subset.cores)

Arguments
list.subset.cores
list of subset cores

Examples
list.subset.cores <- list(c("A.B.C","D.E","A.D"),c("A.C","B.C.D","D.E"),
c("A.B.C","D.E"),c("A.B.C","D.E","B.C.D"))
getGCRs(list.subset.cores) # Confidence column should be 100, 75, 50, 25, 25

gPOOLsizes

Get pool sizes for phase 2

Description
Get pool sizes for phase 2

Usage
gPOOLsizes(rm.ordered, k.max, max.nrs.ee, max.compute)

Arguments
rm.ordered binary rule matrix ordered from phase 1
k.max maximum rule set size
max.nrs.ee max number of rule sets per k
max.compute maximum raw rule sets considered per k
getRulesAsStrings

Represent binary rule matrix as strings

Description

Represent binary rule matrix as strings

Usage

getRulesAsStrings(rm)

Arguments

rm binary rule matrix

Value

vector or rules represented as strings

Examples

library(crso)
data(skm)
list2env(skm.list,envir=globalenv())
rm.full <- buildRuleLibrary(D,rule.thresh = 0.05) \# Rule library matrix, dimension: 60 x 71
rn.ordered <- rm.full \# Skip phase one in this example
getPoolSizes(rm.ordered,k.max = 7,max.nrs.ee = 10000)
# [1] 60 60 40 23 18 16 15

getRulesAsStrings(rm.full)
# output should be: "BRAF-M.CDKN2A-MD" "CDKN2A-MD.NRAS-M"
# "BRAF-M.PTEN-MD" "ADAM18-M.BRAF-M" "ADAM18-M.CDKN2A-MD"
### makeFilteredImList

**Description**

Make filtered im list from phase 3 im list

**Usage**

```r
makeFilteredImList(D, Q, rm, til, filter.thresh)
```

**Arguments**

- **D**: binary matrix of events by samples
- **Q**: penalty matrix of events by samples
- **rm**: matrix of rules ordered by phase one
- **til**: im list from phase 3
- **filter.thresh**: minimum percentage of samples assigned to each rule in rs

**Value**

filtered top im list

**Examples**

```r
data(skcm)
list2env(skcm.list, envir=globalenv())
Q <- log10(P)
rm.full <- buildRuleLibrary(D, rule.thresh = 0.05) # Rule library matrix, dimension: 60 x 71
til.p2 <- makePhaseTwoImList(D, Q, rm.full, k.max = 3,
                               pool.sizes=c(60,20,20),max.stored=100,shouldPrint = FALSE)
filtered.im.list <- makeFilteredImList(D, Q, rm.full, til.p2, filter.thresh = 0.05)
```

### makePhaseOneOrderedRM

**Description**

Order rules according to phase one importance ranking

**Usage**

```r
makePhaseOneOrderedRM(D, rm.start, spr, Q, trn, n.splits, shouldPrint)
```
makePhaseThreeImList

Arguments

D Binary matrix of N events and M samples
rm.start Starting binary rule matrix (i.e., rule library)
spr Random rule sets per rule in each phase one iteration. Default is 40.
Q Penalty matrix, negative log of passenger probability matrix.
trn Target rule number for stopping iterating. Default is 16.
n.splits number of splits for parallelization. Default is all available cpus.
shouldPrint Print progress updates? Default is TRUE

Value

binary rule matrix ordered by phase one importance ranking

Examples

data(skcm)
list2env(skcm.list, envir = globalenv())
Q <- log10(P)
rm.full <- buildRuleLibrary(D, rule.thresh = 0.06) # Rule library matrix, dimension: 36s x 71
rm.ordered <- makePhaseOneOrderedRM(D, rm.full, spr = 1, Q, trn = 34, shouldPrint = TRUE)
# note, for real applications, spr should be at least 40.

Description

Make phase 3 im list from phase 2 im list

Usage

makePhaseThreeImList(D, Q, rm.ordered, til.ee, pool.sizes, max.stored,
max.nrs.borrow, shouldPrint)

Arguments

D binary matrix of events by samples
Q penalty matrix of events by samples
rm.ordered matrix of rules ordered by phase one
til.ee list of rule set matrices (im list) from phase two
pool.sizes pool sizes for phase two
max.stored max number of rule sets saved
max.nrs.borrow max number of new rule sets per k, default is 10^5
shouldPrint Print progress updates? Default is TRUE
**makePhaseTwoImList**

**Value**

phase 3 top im list

**Examples**

```r
library(crso)
data(skm)
list2env(skm.list,envir=globalenv())
Q <- log10(P)
rm.full <- buildRuleLibrary(D,rule.thresh = 0.05) # Rule library matrix, dimension: 60 x 71
til.p2 <- makePhaseTwoImList(D,Q,rm.full,k.max = 3,pool.sizes=c(60,10,10),
max.stored=100,shouldPrint = FALSE)
til.p3 <- makePhaseThreeImList(D,Q,rm.ordered = rm.full,til.ee = til.p2, pool.sizes=c(60,20,20),
max.stored=100,max.nrs.borrow=100,shouldPrint = TRUE)
```

**Value**

largest n such that n choose k < max.num.rs
Examples

library(crso)
data(skcm)
list2env(skcm.list,envir=globalenv())
Q <- log10(P)
rm.full <- buildRuleLibrary(D,rule.thresh = 0.05) # Rule library matrix, dimension: 60 x 71
til.p2 <- makePhaseTwoImList(D,Q,rm.full,k.max = 3, pool.sizes=c(60,20,20),max.stored=100,shouldPrint = TRUE)

makeSubCoreList

Get list of core rules from random subsets of samples

Description
Get list of core rules from random subsets of samples

Usage
makeSubCoreList(D, Q, rm, til, num.subsets, num.evaluated, shouldPrint)

Arguments

D  input matrix D
Q  input matrix Q
rm  binary rule matrix
til  list of top rule set index matrices
num.subsets  number of subset iterations, default is 100
num.evaluated  number of top rs considered per k per iteration, default is 1000
shouldPrint  Print progress updates? Default is TRUE

Examples

library(crso)
data(skcm)
list2env(skcm.list,envir=globalenv())
Q <- log10(P)
rm.full <- buildRuleLibrary(D,rule.thresh = 0.05) # Rule library matrix, dimension: 60 x 71
til.p2 <- makePhaseTwoImList(D,Q,rm.full,k.max = 3, pool.sizes=c(60,20,20),max.stored=100,shouldPrint = FALSE)
subcore.list <- makeSubCoreList(D,Q,rm.full,til.p2,num.subsets=3,num.evaluated=50)
**skcm.list**

---

**Example data set derived from TCGA skin cutaneous melanoma (SKCM) data.**

---

**Description**

A dataset containing the processed inputs used in the melanoma analysis within the CRSO publication.

**Usage**

`skcm.list`

**Format**

A list with 3 items

- **D** Binary alteration matrix. Rows are candidate driver events, columns are samples.
- **P** Passenger probability matrix corresponding to D.
- **cnv.dictionary** Data frame containing copy number genes. ...

**Source**

Dataset derived from data generated by the TCGA Research Network: [https://www.cancer.gov/tcga](https://www.cancer.gov/tcga)
Index

* datasets
  skcm.list, 13

buildRuleLibrary, 2

evaluateListOfIMs, 3

getBestRsList, 4
getCoreK, 4
getCoreRS, 5
getGCDs, 6
getGCEs, 6
getGCRs, 7
getPoolSizes, 7
getRulesAsStrings, 8

makeFilteredImList, 9
makePhaseOneOrderedRM, 9
makePhaseThreeImList, 10
makePhaseTwoImList, 11
makeSubCoreList, 12

skcm.list, 13