An Introduction to *PROMISE*

Stan Pounds, Xueyuan Cao

June 24, 2014

1 Introduction

PROMISE, PROjection onto the Most Interesting Statistical Evidence, is a general procedure to identify genomic features that exhibit a specific biologically interesting pattern of association with multiple phenotypic endpoint variables. Biological knowledge of the endpoint variables is used to define a vector that represents the biologically most interesting values for a set of association statistics. If prior biological knowledge of the endpoint variables is not available, a projection to point 0 is performed. The PROMISE performs one hypothesis test for each genomic feature, and is flexible to accommodate various types of endpoints. In this update, we also incorporte a fast permutation based on negative binomial sampling strategy, in which further permuation for a gene will not be performed for a gene with fixed number of success achieved.

In this document, we describe how to perform PROMISE procedure using hypothetical example data sets provided with the package.

2 Requirements

The PROMISE package depends on *Biobase* and *GSEABase*. The understanding of *ExpressionSet* and *GeneSetCollection* is a prerequiste to perform the PROMISE procedure. Due to the internal handling of multiple endpoints, the consistency of *ExpressionSet* and *GeneSetCollection* is assumed. The detailed requirements are illustrated below.

Load the PROMISE package and the example data sets: sampExprSet, sampGeneSet, and phPatt into R.

```
> library(PROMISE)
> data(sampExprSet)
> data(sampGeneSet)
> data(phPatt)
```

The *ExpressionSet* should contain at least two components: *exprs* (array data) and *phenoData* (endpoint data). *exprs* is a data frame with column names representing the array identifiers (IDs) and row names representing the

probe (genomic feature) IDs. *phenoData* is an *AnnotatedDataFrame* with column names representing the endpoint variables and row names representing array. The array IDs of *phenoData* and *exprs* should be matched.

```
> arrayData<-exprs(sampExprSet)
> ptData<- pData(phenoData(sampExprSet))
> head(arrayData[, 1:4])

         array_1  array_2  array_3  array_4
probe_1 7.594327 6.710827 8.151585 5.885169
probe_2 4.719728 5.611497 5.819049 3.896193
probe_3 8.815163 5.451243 6.670862 7.295233
probe_4 5.187320 5.142358 6.105922 5.333343
probe_5 4.102526 2.655307 4.139362 1.801595
probe_6 5.415526 4.498429 4.393805 5.219712

> head(ptData)

        drugLevel residualDisease   obsTime obsCensor strat
array_1  2.486889               3 0.6117827         1     2
array_2  2.931522               3 1.2657796         1     2
array_3  2.569262               3 1.6318570         0     2
array_4  2.157207               2 0.3879765         0     1
array_5  1.649005               3 0.5763599         1     2
array_6  2.795997               3 0.5474852         0     1

> all(colnames(arrayData)==rownames(ptData))

[1] TRUE
```

The *GeneSetCollection* should be extractable in the following way. The probe IDs should be a subset of probe IDs of *exprs*.

```
> GS.data<-NULL
> for (i in 1:length(sampGeneSet)){
+     tt<-sampGeneSet[i][[1]]
+     this.name<-unlist(geneIds(tt))
+     this.set<-setName(tt)
+     this.data<- cbind.data.frame(featureID=as.character(this.name),
+              setID=rep(as.character(this.set), length(this.name)))
+     GS.data<-rbind.data.frame(GS.data, this.data)
+ }
> sum(!is.element(GS.data[,1], rownames(arrayData)))==0

[1] TRUE
```

The association pattern definition is critical. The prior biological knowledge is required to define the vector that represents the biologically most interesting

values for statistics. If prior biological knowledge of the endpoint variables cannot be assumed, an arbitrary *stat.coef* can be used as it will be ignored with *proj0=TRUE*. In this hypothetical example, we are interested in identifying genomic features that are positively associated with active drug level, negatively associated with minimum disease, and positively associated with survival. The three endpoints are represented in three rows as shown below:

```
> phPatt

              stat.coef      stat.func        endpt.vars
Association 1         1 spearman.rstat         drugLevel
Association 2        -1 spearman.rstat   residualDisease
Association 3         1     jung.rstat obsTime,obsCensor
```

3 PROMISE Analysis

As mentioned in section 2, the *ExpressionSet* and pattern definition are required by PROMISE procedure. *GeneSetCollection* is required if a gene set enrichment analysis (GSEA) is to be performed within the PROMISE analysis.

The code below performs a PROMISE analysis without GSEA. As mentioned above, *GeneSetCollection* is not needed. The gene set result is NULL.

```
> test1 <- PROMISE(exprSet=sampExprSet,
+                  geneSet=NULL,
+                  promise.pattern=phPatt,
+                  strat.var=NULL,
+                  proj0=FALSE,
+                  nbperm=FALSE,
+                  max.ntail=10,
+                  seed=13,
+                  nperms=100)
```

Gene level (genomic feature) result:

```
> gene.res<-test1$generes
> head(gene.res)

  probeid drugLevel.stat residualDisease.stat
1 probe_1     0.50745310           -0.4410378
2 probe_2    -0.05129198            0.1083797
3 probe_3     0.34392723           -0.3069893
4 probe_4    -0.21510307            0.1812378
5 probe_5    -0.43215514            0.3725875
6 probe_6     0.06213032           -0.1796821
  obsTime.obsCensor.stat PROMISE.stat drugLevel.perm.p
1           -0.328713893   0.20659234             0.00
2            0.050025633  -0.03654867             0.68
```

```
3           -0.277173083   0.12458114             0.02
4            0.003140076  -0.13106692             0.10
5            0.346205081  -0.15284585             0.01
6           -0.201508221   0.01343473             0.67
  residualDisease.perm.p obsTime.obsCensor.perm.p
1                   0.00                     0.00
2                   0.46                     0.76
3                   0.03                     0.01
4                   0.17                     0.99
5                   0.00                     0.00
6                   0.18                     0.17
  PROMISE.perm.p
1           0.00
2           0.65
3           0.06
4           0.07
5           0.03
6           0.81
```

Gene set level result:

```
> set.res<-test1$setres
> head(set.res)

NULL
```

The code below performs a PROMISE analysis with GSEA and using fast permutation. As mentioned above, *GeneSetCollection* is required. sampGeneSet, a *GeneSetCollection*, is passed as an argument to PROMISE.

```
> test2 <- PROMISE(exprSet=sampExprSet,
+                  geneSet=sampGeneSet,
+                  promise.pattern=phPatt,
+                  strat.var=NULL,
+                  proj0=FALSE,
+                  nbperm=TRUE,
+                  max.ntail=10,
+                  seed=13,
+                  nperms=100)
```

Gene level (genomic feature) result:

```
> gene.res2<-test2$generes
> head(gene.res2)

  probeid drugLevel.stat residualDisease.stat
1 probe_1     0.50745310           -0.4410378
2 probe_2    -0.05129198            0.1083797
```

```
3 probe_3     0.34392723           -0.3069893
4 probe_4    -0.21510307            0.1812378
5 probe_5    -0.43215514            0.3725875
6 probe_6     0.06213032           -0.1796821
  obsTime.obsCensor.stat PROMISE.stat drugLevel.perm.p
1           -0.328713893   0.20659234             0.00
2            0.050025633  -0.03654867             0.68
3           -0.277173083   0.12458114             0.02
4            0.003140076  -0.13106692             0.10
5            0.346205081  -0.15284585             0.01
6           -0.201508221   0.01343473             0.67
  residualDisease.perm.p obsTime.obsCensor.perm.p
1                   0.00                     0.00
2                   0.46                     0.76
3                   0.03                     0.01
4                   0.17                     0.99
5                   0.00                     0.00
6                   0.18                     0.17
  PROMISE.perm.p nperms
1           0.00    100
2           0.65    100
3           0.06    100
4           0.07    100
5           0.03    100
6           0.81    100
```

Gene set level result:

```
> set.res2<-test2$setres
> head(set.res2)

      setid drugLevel.stat residualDisease.stat
1 GeneSet 1     0.30998611           0.28204640
2 GeneSet 2     0.18012248           0.15382306
3 GeneSet 3     0.19852231           0.18092662
4 GeneSet 4     0.07289261           0.13631292
5 GeneSet 5     0.11745341           0.06539079
6 GeneSet 6     0.08167737           0.08478505
  obsTime.obsCensor.stat PROMISE.stat drugLevel.perm.p
1             0.20105155   0.13032699        0.0000000
2             0.13759326   0.09124962        0.0300000
3             0.20620859   0.06608593        0.0600000
4             0.12875576   0.04299261        0.6500000
5             0.09114311   0.06327372        0.4300000
6             0.09922607   0.03234743        0.8095238
  residualDisease.perm.p obsTime.obsCensor.perm.p
1              0.0000000                0.0100000
```

```
2              0.0300000                0.0900000
3              0.0500000                0.0100000
4              0.1600000                0.2300000
5              0.8900000                0.7200000
6              0.8095238                0.6190476
  PROMISE.perm.p nperms
1      0.0000000    100
2      0.0000000    100
3      0.3500000    100
4      0.8200000    100
5      0.3400000    100
6      0.9047619     21
```

The code below performs a PROMISE analysis with GSEA and using fast permutation without prior knowlege of the three endpoint variables (*proj0=TRUE*). As mentioned above, *GeneSetCollection* is required. sampGeneSet, a *GeneSetCollection*, is passed as an argument to PROMISE.

```
> test3 <- PROMISE(exprSet=sampExprSet,
+                  geneSet=sampGeneSet,
+                  promise.pattern=phPatt,
+                  strat.var=NULL,
+                  proj0=TRUE,
+                  nbperm=TRUE,
+                  max.ntail=10,
+                  seed=13,
+                  nperms=100)
```

Gene level (genomic feature) result:

```
> gene.res3<-test3$generes
> head(gene.res3)

  probeid drugLevel.stat residualDisease.stat
1 probe_1     0.50745310           -0.4410378
2 probe_2    -0.05129198            0.1083797
3 probe_3     0.34392723           -0.3069893
4 probe_4    -0.21510307            0.1812378
5 probe_5    -0.43215514            0.3725875
6 probe_6     0.06213032           -0.1796821
  obsTime.obsCensor.stat PROMISE.stat drugLevel.perm.p
1           -0.328713893   0.56007583             0.00
2            0.050025633   0.01687958             0.68
3           -0.277173083   0.28935328             0.02
4            0.003140076   0.07912632             0.10
5            0.346205081   0.44543747             0.01
6           -0.201508221   0.07675139             0.67
```

```
  residualDisease.perm.p obsTime.obsCensor.perm.p
1                   0.00                     0.00
2                   0.46                     0.76
3                   0.03                     0.01
4                   0.17                     0.99
5                   0.00                     0.00
6                   0.18                     0.17
  PROMISE.perm.p nperms
1           0.00    100
2           0.76    100
3           0.01    100
4           0.29    100
5           0.00    100
6           0.25    100
```

Gene set level result:

```
> set.res3<-test3$setres
> head(set.res3)

      setid drugLevel.stat residualDisease.stat
1 GeneSet 1     0.30998611           0.28204640
2 GeneSet 2     0.18012248           0.15382306
3 GeneSet 3     0.19852231           0.18092662
4 GeneSet 4     0.07289261           0.13631292
5 GeneSet 5     0.11745341           0.06539079
6 GeneSet 6     0.08167737           0.08478505
  obsTime.obsCensor.stat PROMISE.stat drugLevel.perm.p
1             0.20105155   0.27817450        0.0000000
2             0.13759326   0.15129808        0.0300000
3             0.20620859   0.16602416        0.0500000
4             0.12875576   0.06543595        0.8000000
5             0.09114311   0.04097703        0.4074074
6             0.09922607   0.03950889        0.8000000
  residualDisease.perm.p obsTime.obsCensor.perm.p
1              0.0000000                0.0100000
2              0.0300000                0.0900000
3              0.0200000                0.0000000
4              0.3100000                0.2300000
5              0.8148148                0.6666667
6              0.7600000                0.6000000
  PROMISE.perm.p nperms
1      0.0000000    100
2      0.0000000    100
3      0.0000000    100
4      0.3600000    100
```

```
5       0.7222222      54
6       0.7200000      25
```