# Implementation correctness

Here we demonstrate the implementation correctness by comparing the result to exist implementation in R or C lang.

## ORA and SPIA

The SPIA method including the result of ORA analysis thus we check the implementation of SPIA and ORA together in this section.

### The result from R package SPIA

The code below could be found in SPIA's docs.

```
> library(SPIA)
> library(hgu133plus2.db)
> x <- hgu133plus2ENTREZID
> top$ENTREZ<-unlist(as.list(x[top$ID]))
> top<-top[!is.na(top$ENTREZ),]
> top<-top[!duplicated(top$ENTREZ),]
> tg1<-top[top$adj.P.Val<0.1,]
> DE_Colorectal=tg1$logFC
> names(DE_Colorectal)<-as.vector(tg1$ENTREZ)
> ALL_Colorectal=top$ENTREZ

>
res=spia(de=DE_Colorectal,all=ALL_Colorectal,organism="hsa",nB=2000,plots=FALSE,beta=NULL,combine="fish
```

And the top10 result is listed in the table below. ( `Id` , `pSize` , `NDE` is removed for the space consideration)

| | Name | pNDE | pPERT | pG | pGFdr | pGFWER | Status |
|---|---|---|---|---|---|---|---|
| **0** | Focal adhesion | 1.009186e-07 | 0.000005 | 1.479215e-11 | 2.026525e-09 | 2.026525e-09 | Activated |
| **1** | Alzheimer's disease | 2.503714e-11 | 0.221000 | 1.489554e-10 | 1.020344e-08 | 2.040688e-08 | Inhibited |
| **2** | ECM-receptor interaction | 4.058570e-06 | 0.000005 | 5.199181e-10 | 2.374293e-08 | 7.122878e-08 | Activated |
| **3** | Parkinson's disease | 6.435720e-10 | 0.062000 | 9.953264e-10 | 3.408993e-08 | 1.363597e-07 | Inhibited |
| **4** | Pathways in cancer | 4.194045e-05 | 0.003000 | 2.124922e-06 | 5.822286e-05 | 2.911143e-04 | Activated |

### The Result from PyPathway

```
from pypathway import *
c = ColorectalCancer()
r2 = SPIA.run(c.deg, c.background, organism="hsa")
```

The top5 result is list in the table below.

- the `seed` in the SPIA package can not be settled so in the result of `pG`, `pGfdr` and `pGFWER` there are some deviation but still in same order of magnitude。
- the result of `pNDE` indicate that the implementation of `ORA` is correct (In `PyPathway`, the `SPIA` and `ORA` use same implementation of `ORA`).

| | name | pNDE | pPERT | pG | pGfdr | pGFWER | status |
|---|---|---|---|---|---|---|---|
| **04510** | Focal adhesion | 1.00919e-07 | 5e-06 | 1.47922e-11 | 2.02653e-09 | 2.02653e-09 | Activated |
| **05010** | Alzheimer's disease | 2.50371e-11 | 0.232 | 1.56087e-10 | 1.0692e-08 | 2.1384e-08 | Inhibited |
| **04512** | ECM-receptor interaction | 4.05857e-06 | 5e-06 | 5.19918e-10 | 2.37429e-08 | 7.12288e-08 | Activated |
| **05012** | Parkinson's disease | 6.43572e-10 | 0.058 | 9.33601e-10 | 3.19758e-08 | 1.27903e-07 | Inhibited |
| **05200** | Pathways in cancer | 4.19405e-05 | 0.007 | 4.7094e-06 | 0.000129038 | 0.000645188 | Activated |

## GSEA

We use the Java implementation from board institute to check the implementation correctness. The `class vector` and the `expression data` is available at Github. The GSEA algorithm, use random permutation to calculate NES and we can not use same random seed and random algorithm in `Java` and `Python`. So we compare the es of the top five pathway in KEGG enrichment and find that the result is identical to the original GSEA Java implementation

```
Term
Valine, leucine and isoleucine degradation_Homo sapiens_hsa00280   -0.836357
Glycerolipid metabolism_Homo sapiens_hsa00561                      -0.674108
Peroxisome_Homo sapiens_hsa04146                                   -0.672286
Fatty acid degradation_Homo sapiens_hsa00071                       -0.659410
Fatty acid metabolism_Homo sapiens_hsa01212                        -0.631014
Name: es, dtype: float64
```

## Enrichnet

We use the we interface provided by `Enrichnet` to check the correctness of the method.

### Result of the Enrichnet web interface

We perform the analysis with following parameters * Choose a molecular network: STRING * Identifier format: HGNC SYMBOL * gene identifier: we use the gene list derive from `ColorectalCancer` dataset

#### Derive

```
c = ColorectalCancer()
sym = IdMapping.convert(input_id=c.deg_list, source='ENTREZID', target="SYMBOL", species='hsa') sym =
[x[1][0] for x in sym if x[1]]
```

#### Result

| | | | | | | |
|---|---|---|---|---|---|---|

| | Annotation (pathway/process) | XD-score | Fisher q-value | Gene set size | Pathway size | Overlap size |
|---|---|---|---|---|---|---|
| **0** | hsa00280:Valine, leucine and isoleucine degrad... | 4.334854 | 4.772740e-09 | 4342 | 44 | 35 |
| **1** | hsa00603:Glycosphingolipid biosynthesis - glob... | 3.895005 | 2.245753e-02 | 4342 | 14 | 10 |
| **2** | hsa00640:Propanoate metabolism | 3.673330 | 8.447069e-05 | 4342 | 32 | 23 |
| **3** | hsa00410:beta-Alanine metabolism | 3.316433 | 4.982290e-03 | 4342 | 22 | 15 |
| **4** | hsa00900:Terpenoid backbone biosynthesis | 3.252147 | 3.759063e-02 | 4342 | 15 | 10 |

### Result of PyPathway

#### Code

```
c = ColorectalCancer()
# convert ENTREZID to SYMBOL
sym = IdMapping.convert(input_id=c.deg_list, source='ENTREZID', target="SYMBOL", species='hsa')
sym = [x[1][0] for x in sym if x[1]]
# start analysis
en = Enrichnet.run(genesets=sym, graph='string')
```

#### Result

| | Annotation (pathway/process) | XD-score | Fisher q-value | Gene set size | Pathway size | Overlap size |
|---|---|---|---|---|---|---|
| **0** | hsa00280:Valine, leucine and isoleucine degrad... | 4.334854 | 4.772740e-09 | 4342 | 44 | 35 |
| **1** | hsa00603:Glycosphingolipid biosynthesis - glob... | 3.895005 | 2.245753e-02 | 4342 | 14 | 10 |
| **2** | hsa00640:Propanoate metabolism | 3.673330 | 8.447069e-05 | 4342 | 32 | 23 |
| **3** | hsa00410:beta-Alanine metabolism | 3.316433 | 4.982290e-03 | 4342 | 22 | 15 |
| **4** | hsa00900:Terpenoid backbone biosynthesis | 3.252147 | 3.759063e-02 | 4342 | 15 | 10 |

## Network Propagation

### Heat diffuse

We use Cytoscape Diffusion APP to check the correctness of the heat diffusion.

#### Result of Diffusion APP

```
* Image here
```

#### Result of PyPathway

- code

```
from pypathway import diffusion_kernel
G = nx.Graph([[1, 2], [2, 3], [3, 5], [2, 5], [1, 4], [4, 5]])
h = np.array([0, 1, 0, 1, 0])
diffusion_kernel(G, h, rp=0.7, n=100).node
```

- result

```
{1: {'heat': 0.41720485133090102},
 2: {'heat': 0.39506307105908312},
 3: {'heat': 0.30896131580088793},
 4: {'heat': 0.49570801546580717},
 5: {'heat': 0.38306274634330961}}
```

# MAGI

## Pathway select

The test file of MAGI could be found in Github, and we use a modified C version of MAGI only set the random seed to 10 and compile it in `macOS 10.12.6` and compiler `LLVM 9.0` to generate a random free result.

### original

- **pathway select**

```
./Pathway_Select -p StringNew_HPRD.txt -c ID_2_Autism_4_Severe_Missense.Clean_WithNew.txt -h
GeneCoExpresion_ID.txt -e adj1.csv.Tab.BinaryFormat -d New_ESP_Sereve.txt -l Gene_Name_Length.txt -i
0
```

- **cluster**

```
-c RandomGeneList.0 -a 0.3 -s seeds -avgCoExpr 0.415 -avgDensity 0.08 -e adj1.csv.Tab.BinaryFormat -l
5 -i cluster -p StringNew_HPRD.txt -h GeneCoExpresion_ID.txt -m 1 -u 10 -minCoExpr 0.01
```

### PyPathway

- **pathway select**

```
MAGI.select_pathway(
    path + 'StringNew_HPRD.txt',
    path + 'ID_2_Autism_4_Severe_Missense.Clean_WithNew.txt',
    path + 'GeneCoExpresion_ID.txt', path + 'adj1.csv.Tab.BinaryFormat',
    path + 'New_ESP_Sereve.txt',
    path + 'Gene_Name_Length.txt',
    rand_seed = 10
)
```

- **cluster**

```
r = MAGI.cluster(
    path + 'StringNew_HPRD.txt', path + 'GeneCoExpresion_ID.txt',
    path + 'adj1.csv.Tab.BinaryFormat', 10, 5, 10, 0.3
```

```
  )
```

## Result

We get same highest scored submodule: `{'GTF2IRD1': {'numSevereMutInCases': '0', 'prob': '0.000000', 'numMissenseMutInCases': '1', 'weightCases': '3.045591', 'numSevereMutInControl': '0', 'weightControl': '0'}, 'CTNNB1': {'numSevereMutInCases': '1', 'prob': '0.000000', 'numMissenseMutInCases': '1', 'weightCases': '6.503305', 'numSevereMutInControl': '0', 'weightControl': '0'}, 'SMAD4': {'numSevereMutInCases': '0', 'prob': '0.000000', 'numMissenseMutInCases': '0', 'weightCases': '0.000000', 'numSevereMutInControl': '0', 'weightControl': '0'}, 'CDC73': {'numSevereMutInCases': '0', 'prob': '0.000000', 'numMissenseMutInCases': '0', 'weightCases': '0.000000', 'numSevereMutInControl': '0', 'weightControl': '0'}, 'HSPA4': {'numSevereMutInCases': '0', 'prob': '0.000000', 'numMissenseMutInCases': '1', 'weightCases': '3.139670', 'numSevereMutInControl': '0', 'weightControl': '0'}, 'SMARCC2': {'numSevereMutInCases': '1', 'prob': '0.000000', 'numMissenseMutInCases': '0', 'weightCases': '3.808325', 'numSevereMutInControl': '1', 'weightControl': '0'}, 'PIAS1': {'numSevereMutInCases': '0', 'prob': '0.000000', 'numMissenseMutInCases': '1', 'weightCases': '3.515701', 'numSevereMutInControl': '0', 'weightControl': '0'}, 'SMAD2': {'numSevereMutInCases': '0', 'prob': '0.000000', 'numMissenseMutInCases': '1', 'weightCases': '2.088150', 'numSevereMutInControl': '0', 'weightControl': '0'}, 'ZMYND11': {'numSevereMutInCases': '1', 'prob': '0.000000', 'numMissenseMutInCases': '0', 'weightCases': '3.822162', 'numSevereMutInControl': '0', 'weightControl': '0'}, 'GATAD2B': {'numSevereMutInCases': '1', 'prob': '0.000000', 'numMissenseMutInCases': '0', 'weightCases': '3.588371', 'numSevereMutInControl': '2', 'weightControl': '0'}}`