

# Supplementary Material to “Gene set analysis using spatial statistics”

Angela L. Riffo-Campos      Guillermo Ayala  
Amelia Simó      Francisco Montes

February 18, 2021

## Contents

<b>1</b>	<b>Data</b>	<b>2</b>
1.1	PRJNA218851 and TCGA	2
1.1.1	TCGA dataset	2
1.1.2	PRJNA218851 dataset	2
1.1.3	Creating the SummarizedExperiment	4
1.1.4	PRJNA218851 and TCGA	5
1.2	PRJNA411984	5
1.3	PRJNA413956	6
<b>2</b>	<b>Gene set collections</b>	<b>6</b>
2.1	GO	6
2.2	KEGG	7
2.3	Important gene sets	7
<b>3</b>	<b>Proportion test</b>	<b>7</b>
3.1	Generating the matrices of p-values	7
3.1.1	PRJNA218851	8
3.1.2	TCGA	8
3.1.3	PRJNA411984	9
3.1.4	PRJNA413956	9
3.2	Generating the randomization p-values	10
3.2.1	PRJNA218851	10
3.2.2	TCGA	11
3.2.3	PRJNA411984	12
3.2.4	PRJNA413956	12
3.3	Joining results	13
3.3.1	PRJNA218851	13
3.3.2	TCGA	14
3.3.3	PRJNA411984	15
3.3.4	PRJNA413956	16
<b>4</b>	<b>Global analysis</b>	<b>17</b>
4.1	Using each data set and all gene sets	17
4.2	Analyzing p-values for relevant gene sets	20
4.3	Orderings	21
<b>5</b>	<b>edgeR-GOseq pipeline</b>	<b>23</b>
<b>6</b>	<b>Groups and test names</b>	<b>27</b>

<b>7</b>	<b>Simulation study</b>	<b>29</b>
7.1	Code	30
7.1.1	Negative binomial counts	30
7.1.2	Poisson counts	34

## 1 Data

This section contains the code to produce the `SummarizedExperiments`. The code can be reproduced with minor changes concerning with paths that should be easily modified.

### 1.1 PRJNA218851 and TCGA

This section is reproduced from the supplementary material of [1]. It is included in order to show the whole process to generate the data set used in the experimental results.

We will show how to download the data from public repositories and the preprocessing. It ends with the count matrix and metadata included in a `RangedSummarizedExperiment` object.

#### 1.1.1 TCGA dataset

The RNA-seq (HTSeq - Counts) data of paired samples (solid tissue normal and primary tumor) from 50 patients with colorectal cancer were obtained from TCGA dataset. The dataset was downloaded from the Genomic Data Commons Data Portal (GDC: <https://portal.gdc.cancer.gov/>) using the following shell code:

```
./gdc-client download -m CRC_pairs_manifest.txt
```

The clinical data were downloaded too.

```
setwd("../TCGA_CRC")
indir = getwd()
files = list.files(indir, pattern='\\counts.gz')
Metadata = read.csv(file= "ALL_data_TCGA_colonyrectum.csv",
                    header = TRUE, sep=",")
TCGA_CRC_Data = readDGE(files, path=indir, columns=c(1,2),
                       group=Metadata$Sample_Type)
save(TCGA_CRC, file = "TCGA_CRC.rda")
```

#### 1.1.2 PRJNA218851 dataset

In addition, the raw RNA-seq data of paired samples (normal colon and primary tumor) from 18 patients with colorectal cancer were added, this were obtained from the PRJNA218851 BioProject. The dataset was downloaded in FASTQ format from the Sequence Read Archive (SRA: <https://www.ncbi.nlm.nih.gov/sra>) using the SRA toolkit following shell code.

```
fastq-dump --readids --split-files SRA_sample_code
```

The quality of the PRJNA218851 raw dataset was checked using the FASTQC tool and the first 10 low-quality bases, reads shorter than 30 nucleotides and global low quality reads were discarded using `fastx-toolkit`.

```

for i in *.fastq; do fastqc "$i";done &
for i in *.fastq; do fastx_trimmer -i "$i" -o "$i".trim
    ↪ -f 11 -Q 33 -v; done &
for i in *.trim;do fastx_clipper -i "$i" -o "$i".clip -
    ↪ l 30 -v; done &
for i in *.clip;do fastq_quality_filter -i "$i" -o "$i"
    ↪ .filtered -q 20 -p 80 -Q 33 -v; done &

```

Later, the reads were mapped with STAR [Dobin12], first the GRCh38 reference human genome was indexed.

```

./STAR --runMode genomeGenerate --runThreadN 8 --
    ↪ genomeDir ./ --genomeFastaFiles /home4/GRCh38.fa

```

Then, the files were mapped, generating one SAM file per sample, using the following script.

```

#!/bin/bash
A="_1.fastq"
B="_2.fastq"
for line in $(cat list.txt);
do
echo "$line"
./STAR --genomeDir ./ --runThreadN 8 starIndex --
    ↪ readFilesIn /home4/Data_SRA/$line$A /home4/
    ↪ Data_SRA/$line$B --outFileNamePrefix /home4/
    ↪ BAM_STAR/$line;
done

```

After that, the SAM file were converted to sorted BAM file.

```

setwd("/home4/BAM_STAR/") ##Where the files can be found
x = read.csv("files.txt",stringsAsFactors = FALSE)
foutput = "lanza_STAR"
file.create(foutput)
dir_aligned = "/home4/BAM_STAR/"
for(i in 1:nrow(x)){
  file_sam = paste0(dir_aligned,x[i,1])
  file_bam = paste0(dir_aligned,x[i,1],".bam")
  chunk3 = paste("samtools view -bS ",file_sam,
    " | samtools sort - ",file_bam,"\n")
  cat(chunk3,file=foutput,append = TRUE)
}

```

Finally, the count matrix was generated.

```

library(Rsamtools)
library(GenomicFeatures)
library(GenomicAlignments)
gtfFile = "/home4/Data_SRA/GRCh38.gtf"
txdb = makeTxDbFromGFF(gtfFile, format="gtf")
genes = exonsBy(txdb, by="gene")
dirActualData = paste(getwd(),"/",sep="")
sampleTable = read.table("file.txt")
fls = paste(dirActualData,sampleTable[,1],sep="")

```

```

bamLst = BamFileList(fls, index=character(),
                     yieldSize=100000, obeyQname=TRUE)
PRJNA218851_CRC =
  summarizeOverlaps(features = genes, read=bamLst,
                    mode="Union",
                    singleEnd=FALSE,
                    ignore.strand=TRUE,
                    fragments=FALSE)
Metadata = read.csv(file= "PRJNA218851_CRC.csv",
                    header = TRUE, sep=",")
SampleName = Metadata$name_file
Stage = Metadata$Stage
colData(PRJNA218851_CRC) = DataFrame(SampleName, Stage)
save(PRJNA218851_CRC, file="PRJNA218851_CRC.rda")

```

### 1.1.3 Creating the SummarizedExperiment

```

pacman::p_load(SummarizedExperiment, edgeR)
load("PRJNA218851_CRC.rda")
dim(PRJNA218851_CRC)
colData(PRJNA218851_CRC)[, "Stage"]
x1 = PRJNA218851_CRC[, 1:36]
dim(x1)
colnames_x1 = colnames(assay(x1))
colData(x1)[, "Stage"] = factor(colData(x1)[, "Stage"])
levels(colData(x1)[, "Stage"]) = c("case", "control")
colData(x1) = DataFrame(tissue = colData(x1)[, "Stage"],
                       pair = c(51:68, 51:68))
levels(colData(x1)[, "tissue"])
load("TCGA_CRC.rda")
dim(TCGA_CRC)
meta = read.csv("metadata.csv", header=TRUE, sep=";")
levels(meta$Group) = c("control", "case")
match(TCGA_CRC$samples$files, meta[, "File_name"])
x2 = SummarizedExperiment(assays=TCGA_CRC$counts,
                        rowData=rowNames(TCGA_CRC$counts))
colData(x2) = DataFrame(tissue=meta$Group, pair = meta$pair)
x2 = x2[1:60482]
dim(x2)
rowData(x2) = unlist(lapply(rowData(x2)$X, function(x0)
  unlist(strsplit(x0, "[.]"))[[1]]))

cc = match(rowData(x1)[, "ENSEMBL"], rowData(x2)[, "X"])
x11 = x1[!is.na(cc), ]
cc = match(rowData(x2)[, "X"], rowData(x1)[, "ENSEMBL"])
x22 = x2[!is.na(cc), ]

cc = match(rowData(x11)[, "ENSEMBL"], rowData(x22)[, "X"])
x3 = cbind(assay(x11[cc, ]), assay(x22))
colnames(x3) = c(colnames_x1, as.character(meta$name_sample))

se = SummarizedExperiment(assays=x3,

```

```

                                rowData=rowNames(x3))
colData(se) = rbind(colData(x1),colData(x2))
temp = rep(0,nrow(colData(se)))
temp[which(colData(se)[,"tissue"] == "case")] = 1
colData(se) = DataFrame(pair = colData(se)[,"pair"],tissue = temp)
save(se,file="se.rda")

```

This `RangedSummarizedExperiment` object contains all the necessary information for later analysis. Note that the first 18 samples correspond to the study with accession number PRJNA218851 and the last 50 samples correspond to colorectal cancer data in TCGA.

#### 1.1.4 PRJNA218851 and TCGA

First, we load the packages needed later. It is necessary to install the package `pacman`.

```

pacman::p_load(OMICfp2)
pacman::p_load(SummarizedExperiment,org.Hs.eg.db,
               ggplot2,latex2exp,ReportingTools,GSEABase)

```

## 1.2 PRJNA411984

This is the pipeline used.

```

## Trimming
fastqc SRR6067723*
fastx_trimmer -i SRR6067723_1.fastq -o SRR6067723_1_T10
    ↪ .fastq -f 12 -Q 33 -v
## Mapping
./STAR --runMode genomeGenerate --runThreadN 10 --
    ↪ genomeDir ./ --genomeFastaFiles /media/scratch/
    ↪ GRCh38.p13.genome.fa
./STAR --genomeDir ./ --runThreadN 14 starIndex --
    ↪ readFilesIn /media/scratch/SRA_PRJNA411984/
    ↪ SRR6067723_1_T10.fastq /media/scratch/
    ↪ SRA_PRJNA411984/SRR6067723_2_T10.fastq --
    ↪ outFileNamePrefix /media/scratch/SRA_PRJNA411984
    ↪ /SRR6067723
## Convert to BAM files and sort it
sudo samtools view -S -b SRR6067723Aligned.out.sam >
    ↪ SRR6067723.bam
samtools sort SRR6067723.bam -o SRR6067723_sort.bam

```

Include the BAM files in R.

```

pacman::p_load(Rsamtools,GenomicFeatures,GenomicAlignments,org.Hs.eg.db)
gtfFile = "/media/scratch/gencode.v35.chr_patch_hapl_scaff.annotation.gtf"
txdb = makeTxDbFromGFF(gtfFile, format="gtf")
genes = exonsBy(txdb, by="gene")
indir = getwd()
files = list.files(indir, pattern = '*sort.bam')
bamLst = BamFileList(files, index=character(),obeyQname=TRUE)
PRJNA411984 = summarizeOverlaps(features = genes,

```

```

read=bamLst,
mode="Union",
singleEnd=FALSE,
ignore.strand=TRUE,
fragments=FALSE)

colnames(PRJNA411984)
save(PRJNA411984,file="PRJNA411984.rda")

load("../data/PRJNA411984/PRJNA411984.rda")
colData(PRJNA411984) = DataFrame(pair = c(1,2,1,2),tissue = c(1,1,0,0))
metadata(PRJNA411984) = list(name="PRJNA411984")
rowData(PRJNA411984) = gsub("\\\\.*", "", rownames(assay(PRJNA411984)))
names(rowData(PRJNA411984)) = "X"
save(PRJNA411984,file="../data/PRJNA411984.rda")

```

### 1.3 PRJNA413956

```

gtfFile = "/media/scratch/gencode.v35.chr_patch_hapl_scaff.annotation.gtf"
txdb = makeTxDbFromGFF(gtfFile, format="gtf")
genes = exonsBy(txdb, by="gene")
indir = getwd()
files = list.files(indir, pattern = '*sort.bam')
bamLst = BamFileList(files, index=character(),obeyQname=TRUE)
PRJNA413956 = summarizeOverlaps(features = genes,
read=bamLst,
mode="Union",
singleEnd=FALSE,
ignore.strand=TRUE,
fragments=FALSE)

SampleName = c("SRR6159233", "SRR6159234", "SRR6191641", "SRR6191642",
               "SRR6191645", "SRR6191646", "SRR6191647", "SRR6191648",
               "SRR6191649", "SRR6191650", "SRR6191651", "SRR6191652",
               "SRR6191653", "SRR6191654")

tissue = rep(c(1,0),7)
pair = rep(1:7,each=2)
colData(PRJNA413956) = DataFrame(SampleName,pair,tissue)
save(PRJNA413956,file="../data/PRJNA413956.rda")

```

## 2 Gene set collections

### 2.1 GO

```

pacman::p_load(EnrichmentBrowser,org.Hs.eg.db)
hsa_go = EnrichmentBrowser::getGenesets("hsa",db="go",onto="BP")
save(hsa_go,file="hsa_go.rda")

```

## 2.2 KEGG

```
pacman::p_load(EnrichmentBrowser, org.Hs.eg.db)
hsa_kegg = getGenesets("hsa", db="kegg")
save(hsa_kegg, file="hsa_kegg.rda")
```

## 2.3 Important gene sets

We are going to evaluate the ranks of previously known gene sets related with cancer when they are evaluated using our method.

```
gsc1 = readODS::read_ods("../data/Pathways_cancer.ods")
```

```
load("hsa_kegg.rda")
kegg.id = na.omit(gsc1$KEGG_ID)
x1 = unlist(lapply(names(hsa_kegg), function(y) unlist(strsplit(y, "_"))[1]))
kegg_relevant = hsa_kegg[match(kegg.id, x1)]
length(kegg_relevant)
save(kegg_relevant, file="kegg_relevant.rda")
```

```
load("hsa_go.rda")
go.id = na.omit(gsc1$GO_ID)
x1 = unlist(lapply(names(hsa_go), function(y) unlist(strsplit(y, "_"))[1]))
go_relevant = hsa_go[match(go.id, x1)]
length(go_relevant)
save(go_relevant, file="go_relevant.rda")
```

```
load("hsa_go.rda")
go.id.short = na.omit(gsc1$GO_Shortlist)
x1 = unlist(lapply(names(hsa_go), function(y) unlist(strsplit(y, "_"))[1]))
go_relevant_short = hsa_go[match(go.id.short, x1)]
length(go_relevant_short)
save(go_relevant_short, file="go_relevant_short.rda")
```

## 3 Proportion test

### 3.1 Generating the matrices of p-values

We use p-values corresponding to the proportion test (see paper) using the randomization distributions. First, the following function is needed.

```
se.gsc = function(se, db = hsa_go, msize=10, name = "go",
                  fractionRemain = .15){
  ## Selecting rows using fractionRemain
  se = se[matrixStats::rowSums2(assay(se)) >=
          ceiling(fractionRemain*ncol(se)),]
  res = AnnotationDbi::select(org.Hs.eg.db, keys=rowData(se)$X,
                              column=c("ENTREZID"), keytype="ENSEMBL")
}
```

```

b = BiocGenerics::match(rowData(se)$X,res[, "ENSEMBL"])
gsc = lapply(db,function(y) which(is.element(res[b, "ENTREZID"],y)))
res2 = sort(unique(unlist(gsc)))
se2 = se[res2,]
res2 = res[res2,]
b2 = BiocGenerics::match(rowData(se2)$X,res2[, "ENSEMBL"])
gsc2 = lapply(db,function(y) which(is.element(res2[b2, "ENTREZID"],y)))
gsc2 = gsc2[which(sapply(gsc2,length)>=msize)]
gsc2 = list(set=gsc2,name=name)
list(se = se2,gsc = gsc2)
}

```

### 3.1.1 PRJNA218851

```

data(PRJNA218851,package="OMICfpp2")
##load("../data/PRJNA218851.rda")
load("hsa_go.rda")
sg = se.gsc(se=PRJNA218851,db = hsa_go,name = "go")

pointgene(se=sg$se,
  tests = list(
    list(test=proportion.p,name="proportion.p"),
    list(test=CuzickEdwards,name="CuzickEdwards")),
  nsim = rep(100,2),
  type="complete",
  dir="/home/gag/Nextcloud/pointgene20_Output2/",
  steps = c(1,0,0,0,0))
pointgene(se=sg$se,
  tests = list(
    list(test=proportion.p,name="proportion.p"),
    list(test=CuzickEdwards,name="CuzickEdwards")),
  nsim = rep(100,2),
  type="between-pair",
  dir="/home/gag/Nextcloud/pointgene20_Output2/",
  steps = c(1,0,0,0,0))

```

### 3.1.2 TCGA

```

data(TCGA,package="OMICfpp2")
##load("../data/TCGA.rda")
load("hsa_go.rda")
sg = se.gsc(se=TCGA,db = hsa_go,name = "go")
pointgene(se=sg$se,
  tests = list(
    list(test=proportion.p,name="proportion.p"),
    list(test=CuzickEdwards,name="CuzickEdwards")),
  nsim = rep(100,2),
  type="complete",
  dir="/home/gag/Nextcloud/pointgene20_Output2/",
  steps = c(1,0,0,0,0))

```



```
pointgene(se=sg$se,
  tests = list(
    list(test=proportion.p,name="proportion.p"),
    list(test=CuzickEdwards,name="CuzickEdwards")),
  nsim = rep(100,2),
  type="between-pair",
  dir="/home/gag/Nextcloud/pointgene20_Output2/",
  steps = c(1,0,0,0,0))
```

### 3.1.3 PRJNA411984

```
data(PRJNA411984,package="OMICfpp2")
##load("../data/PRJNA411984.rda")
load("hsa_go.rda")
sg = se.gsc(se=PRJNA411984,db = hsa_go,name = "go")

pointgene(se=sg$se,
  tests = list(
    list(test=proportion.p,name="proportion.p"),
    list(test=CuzickEdwards,name="CuzickEdwards")),
  nsim = c(4,100),
  type="complete",
  dir="/home/gag/Nextcloud/pointgene20_Output2/",
  steps = c(1,0,0,0,0))

pointgene(se=sg$se,
  tests = list(
    list(test=proportion.p,name="proportion.p"),
    list(test=CuzickEdwards,name="CuzickEdwards")),
  nsim = c(4,100),
  type="between-pair",
  dir="/home/gag/Nextcloud/pointgene20_Output2/",
  steps = c(1,0,0,0,0))
```

### 3.1.4 PRJNA413956

```
data(PRJNA413956,package="OMICfpp2")
##load("../data/PRJNA413956.rda")
load("hsa_go.rda")
sg = se.gsc(se=PRJNA413956,db = hsa_go,name = "go")

pointgene(se=sg$se,
  tests = list(
    list(test=proportion.p,name="proportion.p"),
    list(test=CuzickEdwards,name="CuzickEdwards")),
  nsim = c(14,100),
  type="complete",
  dir="/home/gag/Nextcloud/pointgene20_Output2/",
  steps = c(1,0,0,0,0))
```

```
pointgene(se=sg$se,
  tests = list(
    list(test=proportion.p,name="proportion.p"),
    list(test=CuzickEdwards,name="CuzickEdwards")),
  nsim = c(14,100),
  type="between-pair",
  dir="/home/gag/Nextcloud/pointgene20_Output2/",
  steps = c(1,0,0,0,0))
```

## 3.2 Generating the randomization p-values

In this section, the tests proposed in the paper are evaluated. The detailed description can be found in the paper.

### 3.2.1 PRJNA218851

Let us start with the GO collection.

```
data(PRJNA218851,package="OMICfpp2")
##load("../data/PRJNA218851.rda")
load("hsa_go.rda")
sg = se.gsc(se=PRJNA218851,db = hsa_go,name = "go")
test2 = list(list(test =CuzickEdwards ,name = "CuzickEdwards"),
  list(test = DiggleMorrisMorton,name = "DiggleMorrisMorton"),
  list(test = diggle90,name = "diggle90"))

for(type in c("complete","between-pair")){
  cat("type ",type,"\n")
  for(i in 1:3){
    cat("test ",test2[[i]]$name,"\n")
    pointgene(se=sg$se,
      tests = list(
        list(test=proportion.p,name="proportion.p"),
        list(test=test2[[i]]$test,name=test2[[i]]$name)),
      nsim = rep(100,2),
      type=type,
      dir="/home/gag/Nextcloud/pointgene20_Output2/",
      steps = c(0,1,1,0,0),
      gsc = sg$gsc)
  }
}
```

Second, we repeat the analysis for the KEGG collection.

```
data(PRJNA218851,package="OMICfpp2")
##load("../data/PRJNA218851.rda")
load("hsa_kegg.rda")
sg = se.gsc(se=PRJNA218851,db = hsa_kegg,name = "kegg")
test2 = list(list(test =CuzickEdwards ,name = "CuzickEdwards"),
  list(test = DiggleMorrisMorton,name = "DiggleMorrisMorton"),
  list(test = diggle90,name = "diggle90"))

for(type in c("complete","between-pair")){
```

```

cat("type ",type,"\n")
for(i in 1:3){
  cat("test ",test2[[i]]$name,"\n")
  pointgene(se=sg$se,
    tests = list(
      list(test=proportion.p,name="proportion.p"),
      list(test=test2[[i]]$test,name=test2[[i]]$name)),
    nsim = rep(100,2),
    type=type,
    dir="/home/gag/Nextcloud/pointgene20_Output2/",
    steps = c(0,1,1,0,0),
    gsc = sg$gsc)
}
}

```

### 3.2.2 TCGA

First we use the GO collection.

```

sg = se.gsc(se=TCGA,db = hsa_go,name = "go")
test2 = list(list(test =CuzickEdwards ,name = "CuzickEdwards"),
  list(test = DiggleMorrisMorton,name = "DiggleMorrisMorton"),
  list(test = diggle90,name = "diggle90"))

for(type in c("complete","between-pair")){
  for(i in 1:3){
    pointgene(se=sg$se,
      tests = list(
        list(test=proportion.p,name="proportion.p"),
        list(test=test2[[i]]$test,name=test2[[i]]$name)),
      nsim = rep(100,2),
      type=type,
      dir="/home/gag/Nextcloud/pointgene20_Output2/",
      steps = c(0,1,1,0,0),
      gsc = sg$gsc)
  }
}

```

Second, we repeat the analysis for the KEGG collection.

```

sg = se.gsc(se=TCGA,db = hsa_kegg,name = "kegg")
test2 = list(list(test =CuzickEdwards ,name = "CuzickEdwards"),
  list(test = DiggleMorrisMorton,name = "DiggleMorrisMorton"),
  list(test = diggle90,name = "diggle90"))

for(type in c("complete","between-pair")){
  for(i in 1:3){
    pointgene(se=sg$se,
      tests = list(
        list(test=proportion.p,name="proportion.p"),
        list(test=test2[[i]]$test,name=test2[[i]]$name)),
      nsim = rep(100,2),
      type=type,

```

```

        dir="/home/gag/Nextcloud/pointgene20_Output2/",
        steps = c(0,1,1,0,0),
        gsc = sg$gsc)
    }
}

```

### 3.2.3 PRJNA411984

```

data(PRJNA411984,package="OMICfpp2")
##load("../data/PRJNA411984.rda")
sg = se.gsc(se=PRJNA411984,db = hsa_go,name = "go")
test2 = list(list(test =CuzickEdwards ,name = "CuzickEdwards"),
              list(test = DiggleMorrisMorton,name = "DiggleMorrisMorton"),
              list(test = diggle90,name = "diggle90"))

for(type in c("complete","between-pair")){
  for(i in 1:3){
    pointgene(se=sg$se,
              tests = list(
                list(test=proportion.p,name="proportion.p"),
                list(test=test2[[i]]$test,name=test2[[i]]$name)),
              nsim = c(4,100),
              type=type,
              dir="/home/gag/Nextcloud/pointgene20_Output2/",
              steps = c(0,1,1,0,0),
              gsc = sg$gsc)
  }
}

```

Second, we repeat the analysis for the KEGG collection.

```

sg = se.gsc(se=PRJNA411984,db = hsa_kegg,name = "kegg")
test2 = list(list(test =CuzickEdwards ,name = "CuzickEdwards"),
              list(test = DiggleMorrisMorton,name = "DiggleMorrisMorton"),
              list(test = diggle90,name = "diggle90"))

for(type in c("complete","between-pair")){
  for(i in 1:3){
    pointgene(se=sg$se,
              tests = list(
                list(test=proportion.p,name="proportion.p"),
                list(test=test2[[i]]$test,name=test2[[i]]$name)),
              nsim = c(4,100),
              type=type,
              dir="/home/gag/Nextcloud/pointgene20_Output2/",
              steps = c(0,1,1,0,0),
              gsc = sg$gsc)
  }
}

```

### 3.2.4 PRJNA413956

```

data<PRJNA413956,package="OMICfpp2">
##load("../data/PRJNA413956.rda")
load("hsa_go")
sg = se.gsc(se=PRJNA413956,db = hsa_go,name = "go")
test2 = list(list(test =CuzickEdwards ,name = "CuzickEdwards"),
             list(test = DiggleMorrisMorton,name = "DiggleMorrisMorton"),
             list(test = diggle90,name = "diggle90"))

for(type in c("complete","between-pair")){
  for(i in 1:3){
    pointgene(se=sg$se,
              tests = list(
                list(test=proportion.p,name="proportion.p"),
                list(test=test2[[i]]$test,name=test2[[i]]$name)),
              nsim = c(4,100),
              type=type,
              dir="/home/gag/Nextcloud/pointgene20_Output2/",
              steps = c(0,1,1,0,0),
              gsc = sg$gsc)
  }
}

```

Second, we repeat the analysis for the KEGG collection.

```

load("hsa_kegg.rda")
sg = se.gsc(se=PRJNA413956,db = hsa_kegg,name = "kegg")
test2 = list(list(test =CuzickEdwards ,name = "CuzickEdwards"),
             list(test = DiggleMorrisMorton,name = "DiggleMorrisMorton"),
             list(test = diggle90,name = "diggle90"))

for(type in c("complete","between-pair")){
  for(i in 1:3){
    pointgene(se=sg$se,
              tests = list(
                list(test=proportion.p,name="proportion.p"),
                list(test=test2[[i]]$test,name=test2[[i]]$name)),
              nsim = c(4,100),
              type=type,
              dir="/home/gag/Nextcloud/pointgene20_Output2/",
              steps = c(0,1,1,0,0),
              gsc = sg$gsc)
  }
}

```

### 3.3 Joining results

#### 3.3.1 PRJNA218851

```

data<PRJNA218851,package="OMICfpp2">
##load("../data/PRJNA218851.rda")
test2 = list(list(test =CuzickEdwards ,name = "CuzickEdwards"),
             list(test = DiggleMorrisMorton,name = "DiggleMorrisMorton"),
             list(test = diggle90,name = "diggle90"))

```

```

load("hsa_go.rda")
sg = se.gsc(se=PRJNA218851,db = hsa_go,name = "go")
for(type in c("complete","between-pair")){
  cat("type ",type,"\n")
  for(i in 1:3){
    cat("i ",i,"\n")
    pointgene(se=sg$se,
              tests = list(
                list(test=proportion.p,name="proportion.p"),
                list(test=test2[[i]]$test,name=test2[[i]]$name)),
              nsim = c(100,100),
              type=type,
              dir="/home/gag/Nextcloud/pointgene20_Output2/PRJNA218851/",
              steps = c(0,0,0,1,0),
              gsc = sg$gsc)
  }
}
load("hsa_kegg.rda")
sg = se.gsc(se=PRJNA218851,db = hsa_kegg,name = "kegg")
for(type in c("complete","between-pair")){
  cat("type ",type,"\n")
  for(i in 1:3){
    cat("i ",i,"\n")
    pointgene(se=sg$se,
              tests = list(
                list(test=proportion.p,name="proportion.p"),
                list(test=test2[[i]]$test,name=test2[[i]]$name)),
              nsim = c(100,100),
              type=type,
              dir="/home/gag/Nextcloud/pointgene20_Output2/PRJNA218851/",
              steps = c(0,0,0,1,0),
              gsc = sg$gsc)
  }
}

```

### 3.3.2 TCGA

```

data(TCGA,package="OMICfpp2")
##load("../data/TCGA.rda")
test2 = list(list(test =CuzickEdwards ,name = "CuzickEdwards"),
             list(test = DiggleMorrisMorton,name = "DiggleMorrisMorton"),
             list(test = diggle90,name = "diggle90"))
load("hsa_go.rda")
sg = se.gsc(se=TCGA,db = hsa_go,name = "go")
for(type in c("complete","between-pair")){
  cat("type ",type,"\n")
  for(i in 1:3){
    cat("i ",i,"\n")
    pointgene(se=sg$se,
              tests = list(
                list(test=proportion.p,name="proportion.p"),
                list(test=test2[[i]]$test,name=test2[[i]]$name)),
              nsim = c(100,100),
              type=type,
              dir="/home/gag/Nextcloud/pointgene20_Output2/PRJNA218851/",
              steps = c(0,0,0,1,0),
              gsc = sg$gsc)
  }
}

```

```

        nsim = c(100,100),
        type=type,
        dir="/home/gag/Nextcloud/pointgene20_Output2/TCGA/",
        steps = c(0,0,0,1,0),
        gsc = sg$gsc)
    }
}
load("hsa_kegg.rda")
sg = se.gsc(se=TCGA,db = hsa_kegg,name = "kegg")
for(type in c("complete","between-pair")){
  cat("type ",type,"\n")
  for(i in 1:3){
    cat("i ",i,"\n")
    pointgene(se=sg$se,
      tests = list(
        list(test=proportion.p,name="proportion.p"),
        list(test=test2[[i]]$test,name=test2[[i]]$name)),
      nsim = c(100,100),
      type=type,
      dir="/home/gag/Nextcloud/pointgene20_Output2/TCGA/",
      steps = c(0,0,0,1,0),
      gsc = sg$gsc)
  }
}

```

### 3.3.3 PRJNA411984

```

data(PRJNA411984,package="OMICfpp2")
##load("../data/PRJNA411984.rda")
test2 = list(list(test =CuzickEdwards ,name = "CuzickEdwards"),
  list(test = DiggleMorrisMorton,name = "DiggleMorrisMorton"),
  list(test = diggle90,name = "diggle90"))
load("hsa_go.rda")
sg = se.gsc(se=PRJNA411984,db = hsa_go,name = "go")
for(type in c("complete","between-pair")){
  cat("type ",type,"\n")
  for(i in 1:3){
    cat("i ",i,"\n")
    pointgene(se=sg$se,
      tests = list(
        list(test=proportion.p,name="proportion.p"),
        list(test=test2[[i]]$test,name=test2[[i]]$name)),
      nsim = c(4,100),
      type=type,
      dir="/home/gag/Nextcloud/pointgene20_Output2/PRJNA411984/",
      steps = c(0,0,0,1,0),
      gsc = sg$gsc)
  }
}
load("hsa_kegg.rda")
sg = se.gsc(se=PRJNA411984,db = hsa_kegg,name = "kegg")
for(type in c("complete","between-pair")){

```

```

cat("type ",type,"\n")
for(i in 1:3){
  cat("i ",i,"\n")
  pointgene(se=sg$se,
    tests = list(
      list(test=proportion.p,name="proportion.p"),
      list(test=test2[[i]]$test,name=test2[[i]]$name)),
    nsim = c(4,100),
    type=type,
    dir="/home/gag/Nextcloud/pointgene20_Output2/PRJNA411984/",
    steps = c(0,0,0,1,0),
    gsc = sg$gsc)
}
}

```

### 3.3.4 PRJNA413956

```

data(PRJNA413956,package="OMICfpp2")
##load("../data/PRJNA413956.rda")
test2 = list(list(test =CuzickEdwards ,name = "CuzickEdwards"),
  list(test = DiggleMorrisMorton,name = "DiggleMorrisMorton"),
  list(test = diggle90,name = "diggle90"))
load("hsa_go.rda")
sg = se.gsc(se=PRJNA413956,db = hsa_go,name = "go")
for(type in c("complete","between-pair")){
  cat("type ",type,"\n")
  for(i in 1:3){
    cat("i ",i,"\n")
    pointgene(se=sg$se,
      tests = list(
        list(test=proportion.p,name="proportion.p"),
        list(test=test2[[i]]$test,name=test2[[i]]$name)),
      nsim = c(4,100),
      type=type,
      dir="/home/gag/Nextcloud/pointgene20_Output2/PRJNA413956/",
      steps = c(0,0,0,1,0),
      gsc = sg$gsc)
  }
}
load("hsa_kegg.rda")
sg = se.gsc(se=PRJNA413956,db = hsa_kegg,name = "kegg")
for(type in c("complete","between-pair")){
  cat("type ",type,"\n")
  for(i in 1:3){
    cat("i ",i,"\n")
    pointgene(se=sg$se,
      tests = list(
        list(test=proportion.p,name="proportion.p"),
        list(test=test2[[i]]$test,name=test2[[i]]$name)),
      nsim = c(4,100),
      type=type,
      dir="/home/gag/Nextcloud/pointgene20_Output2/PRJNA413956/",

```



```

        steps = c(0,0,0,1,0),
        gsc = sg$gsc)
    }
}

```

## 4 Global analysis

### 4.1 Using each data set and all gene sets

```

data(PRJNA218851,package="OMICfpp2")
data(PRJNA411984,package="OMICfpp2")
data(PRJNA413956,package="OMICfpp2")
data(TCGA,package="OMICfpp2")

sens = list(
  list(data = PRJNA218851,name= "PRJNA218851"),
  list(data = PRJNA411984,name= "PRJNA411984"),
  list(data = PRJNA413956,name= "PRJNA413956"),
  list(data = TCGA,name= "TCGA"))

testRLs = list(
  list(testRL = CuzickEdwards,testRLn = "CuzickEdwards"),
  list(testRL = DiggleMorrisMorton,testRLn = "DiggleMorrisMorton"),
  list(testRL = diggle90,testRLn = "diggle90"))

types = c("between-pair","complete")

load("hsa_kegg.rda")
p.kegg = matrix(NA,nrow=length(hsa_kegg),ncol=24)
rownames(p.kegg) = names(hsa_kegg)
names0 = NULL
col0 = 0
for(i in 1:4){
  sg = se.gsc(se=sens[[i]]$data,db = hsa_kegg,name = "kegg")
  for(j in 1:length(testRLs)){
    for(k in 1:2){
      col0 = col0 +1
      tests0 = list(list(test=proportion.p,name="proportion.p"),
                    list(test=testRLs[[j]]$testRL,
                        name=testRLs[[j]]$testRLn))
      dir0 = paste0("/home/gag/Nextcloud/pointgene20_Output2/",
                    sens[[i]]$name, "/")
      p.value = pointgene(se = sg$se,
                          tests = tests0,
                          nsim = rep(100,2),type=types[k],
                          dir=dir0,
                          steps = c(0,0,0,1,1),gsc=sg$gsc,aggr=mean)
      aa = match(rownames(p.value),rownames(p.kegg))
      p.kegg[aa,col0] = p.value
      names0 = c(names0,paste(sens[[i]]$name,types[k],
                              testRLs[[j]]$testRLn,sep="_"))
    }
  }
}

```

```

    }
  }
}
p.kegg = data.frame(p.kegg)
names(p.kegg) = names0
save(p.kegg, file="p.kegg.rda")

```

Selecting the first p-value for all experimental setup.

```

select_one = function(x, to_select=1, na.rm=TRUE){
  if(na.rm) x = na.omit(x)
  x[to_select]
}

```

Now, we select just one p-value in order to analyze one realization.

```

load("hsa_kegg.rda")
p.kegg_one = matrix(NA, nrow=length(hsa_kegg), ncol=24)
rownames(p.kegg_one) = names(hsa_kegg)
names0 = NULL
col0 = 0
for(i in 1:4){
  sg = se.gsc(se=sens[[i]]$data, db = hsa_kegg, name = "kegg")
  for(j in 1:length(testRLs)){
    for(k in 1:2){
      col0 = col0 + 1
      tests0 = list(list(test=proportion.p, name="proportion.p"),
                    list(test=testRLs[[j]]$testRL,
                        name=testRLs[[j]]$testRLn))
      dir0 = paste0("/home/gag/Nextcloud/pointgene20_Output2/",
                    sens[[i]]$name, "/")
      p.value = pointgene(se = sg$se,
                          tests = tests0,
                          nsim = rep(100, 2), type=types[k],
                          dir=dir0,
                          steps = c(0, 0, 0, 1, 1), gsc=sg$gsc, aggr=select_one)
      aa = match(rownames(p.value), rownames(p.kegg_one))
      p.kegg_one[aa, col0] = p.value
      names0 = c(names0, paste(sens[[i]]$name, types[k],
                               testRLs[[j]]$testRLn, sep="_"))
    }
  }
}
p.kegg_one = data.frame(p.kegg_one)
names(p.kegg_one) = names0
save(p.kegg_one, file="p.kegg_one.rda")

```

```

load("hsa_go.rda")
p.go = matrix(NA, nrow=length(hsa_go), ncol=24)
rownames(p.go) = names(hsa_go)
names0 = NULL
col0 = 0
for(i in 1:4){

```

```

sg = se.gsc(se=sens[[i]]$data,db = hsa_go,name = "go")
for(j in 1:length(testRLs)){
  for(k in 1:2){
    col0 = col0 +1
    tests0 = list(list(test=proportion.p,name="proportion.p"),
                  list(test=testRLs[[j]]$testRL,
                      name=testRLs[[j]]$testRLn))
    dir0 = paste0("/home/gag/Nextcloud/pointgene20_Output2/",
                  sens[[i]]$name, "/")
    p.value = pointgene(se = sg$se,
                       tests = tests0,
                       nsim = rep(100,2),type=types[k],
                       dir=dir0,
                       steps = c(0,0,0,1,1),gsc=sg$gsc,aggr=mean)
    aa = match(rownames(p.value),rownames(p.go))
    p.go[aa,col0] = p.value
    names0 = c(names0,paste(sens[[i]]$name,types[k],
                           testRLs[[j]]$testRLn,sep="_"))
  }
}

p.go = data.frame(p.go)
names(p.go) = names0
save(p.go,file="p.go.rda")

```

```

load("hsa_go.rda")
p.go_one = matrix(NA,nrow=length(hsa_go),ncol=24)
rownames(p.go_one) = names(hsa_go)
names0 = NULL
col0 = 0
for(i in 1:4){
  sg = se.gsc(se=sens[[i]]$data,db = hsa_go,name = "go")
  for(j in 1:length(testRLs)){
    for(k in 1:2){
      col0 = col0 +1
      tests0 = list(list(test=proportion.p,name="proportion.p"),
                    list(test=testRLs[[j]]$testRL,
                        name=testRLs[[j]]$testRLn))
      dir0 = paste0("/home/gag/Nextcloud/pointgene20_Output2/",
                    sens[[i]]$name, "/")
      p.value = pointgene(se = sg$se,
                         tests = tests0,
                         nsim = rep(100,2),type=types[k],
                         dir=dir0,
                         steps = c(0,0,0,1,1),gsc=sg$gsc,aggr=mean)
      aa = match(rownames(p.value),rownames(p.go_one))
      p.go_one[aa,col0] = p.value
      names0 = c(names0,paste(sens[[i]]$name,types[k],
                              testRLs[[j]]$testRLn,sep="_"))
    }
  }
}

```

```
p.go_one = data.frame(p.go_one)
names(p.go_one) = names0
save(p.go_one, file="p.go_one.rda")
```

## 4.2 Analyzing p-values for relevant gene sets

We evaluate the minima for all procedures and data sets for each relevant gene set.

```
load("p.go.rda")
load("go_relevant.rda")
toremove = which(apply(p.go,1,function(x) all(is.na(x))))
p_go = p.go[-toremove,]
save(p_go, file="p_go.rda") ## p.go without NAs
relevant = match(names(go_relevant), rownames(p_go))
relevant = na.omit(relevant)
p_go_relevant = p_go[relevant,]
save(p_go_relevant, file="p_go_relevant.rda")
df_go = data.frame(rownames(p_go_relevant), apply(p_go_relevant, 1, min,
                                                    na.rm=TRUE))
save(df_go, file="df_go.rda")
```

```
load("p.go_one.rda")
load("go_relevant.rda")
toremove = which(apply(p.go_one,1,function(x) all(is.na(x))))
p_go_one = p.go_one[-toremove,]
save(p_go_one, file="p_go_one.rda") ## p.go_one without NAs
relevant = match(names(go_relevant), rownames(p_go_one))
relevant = na.omit(relevant)
p_go_one_relevant = p_go_one[relevant,]
save(p_go_one_relevant, file="p_go_one_relevant.rda")
df_go_one = data.frame(rownames(p_go_one_relevant),
                       apply(p_go_one_relevant, 1, min, na.rm=TRUE))
save(df_go_one, file="df_go_one.rda")
```

```
load("p.kegg.rda")
load("kegg_relevant.rda")
toremove = which(apply(p.kegg,1,function(x) all(is.na(x))))
p_kegg = p.kegg[-toremove,]
save(p_kegg, file="p_kegg.rda") ## p.kegg without NAs
relevant = match(names(kegg_relevant), rownames(p_kegg))
relevant = na.omit(relevant)
p_kegg_relevant = p_kegg[relevant,]
save(p_kegg_relevant, file="p_kegg_relevant.rda")
df_kegg = data.frame(rownames(p_kegg_relevant), apply(p_kegg_relevant, 1, min,
                                                         na.rm=TRUE))
save(df_kegg, file="df_kegg.rda")
View(df_kegg)
```

```

load("p.kegg_one.rda")
load("kegg_relevant.rda")
toremove = which(apply(p.kegg_one,1,function(x) all(is.na(x))))
p_kegg_one = p.kegg_one[-toremove,]
save(p_kegg_one,file="p_kegg_one.rda") ## p.kegg_one without NAs
relevant = match(names(kegg_relevant),rownames(p_kegg_one))
relevant = na.omit(relevant)
p_kegg_one_relevant = p_kegg_one[relevant,]
save(p_kegg_one_relevant,file="p_kegg_one_relevant.rda")
df_kegg_one = data.frame(rownames(p_kegg_one_relevant),
                        apply(p_kegg_one_relevant,1,min,na.rm=TRUE))
save(df_kegg_one,file="df_kegg_one.rda")
View(df_kegg_one)

```

### 4.3 Orderings

```

load("p_go.rda")
id = unlist(lapply(rownames(p_go),function(y) unlist(strsplit(y,"_"))[1]))
GO = ifelse(id == "NA", NA,
            paste("<a href='http://amigo.geneontology.org/amigo/term/",
                  id, "'>", id, "</a>", sep = ""))
df = data.frame(GO,p_go)
htmlRep = HTMLReport(shortName = "p_go",
                     reportDirectory = "./reports")
publish(df, htmlRep)
finish(htmlRep)

```

```

load("p_go_relevant.rda")
id = unlist(lapply(rownames(p_go_relevant),function(y)
  unlist(strsplit(y,"_"))[1]))
GO = ifelse(id == "NA", NA,
            paste("<a href='http://amigo.geneontology.org/amigo/term/",
                  id, "'>", id, "</a>", sep = ""))
df = data.frame(GO,p_go_relevant)
htmlRep = HTMLReport(shortName = "p_go_relevant",
                     reportDirectory = "./reports")
publish(df, htmlRep)
finish(htmlRep)

```

```

load("p_go_one_relevant.rda")
id = unlist(lapply(rownames(p_go_one_relevant),function(y)
  unlist(strsplit(y,"_"))[1]))
GO = ifelse(id == "NA", NA,
            paste("<a href='http://amigo.geneontology.org/amigo/term/",
                  id, "'>", id, "</a>", sep = ""))
df = data.frame(GO,p_go_one_relevant)
htmlRep = HTMLReport(shortName = "p_go_one_relevant",
                     reportDirectory = "./reports")
publish(df, htmlRep)
finish(htmlRep)

```

```

load("p_kegg.rda")
id = unlist(lapply(rownames(p_kegg),function(y) unlist(strsplit(y,"_"))[1]))
KEGG = ifelse(id == "NA", NA,
              paste("<a href='http://www.genome.jp/dbget-bin/www_bget?",
                    id, "'>", id, "</a>", sep = ""))

df = data.frame(KEGG,p_kegg)
htmlRep = HTMLReport(shortName = "p_kegg",
                     reportDirectory = "./reports")
publish(df, htmlRep)
finish(htmlRep)

```

```

load("p_kegg_relevant.rda")
id = unlist(lapply(rownames(p_kegg_relevant),function(y)
  unlist(strsplit(y,"_"))[1]))
KEGG = ifelse(id == "NA", NA,
              paste("<a href='http://www.genome.jp/dbget-bin/www_bget?",
                    id, "'>", id, "</a>", sep = ""))

df = data.frame(KEGG,p_kegg_relevant)
htmlRep = HTMLReport(shortName = "p_kegg_relevant",
                     reportDirectory = "./reports")
publish(df, htmlRep)
finish(htmlRep)

```

```

load("p_kegg_one.rda")
id = unlist(lapply(rownames(p_kegg_one),function(y) unlist(strsplit(y,"_"))[1]))
KEGG = ifelse(id == "NA", NA,
              paste("<a href='http://www.genome.jp/dbget-bin/www_bget?",
                    id, "'>", id, "</a>", sep = ""))

df = data.frame(KEGG,p_kegg_one)
htmlRep = HTMLReport(shortName = "p_kegg_one",
                     reportDirectory = "./reports")
publish(df, htmlRep)
finish(htmlRep)

```

```

load("p_kegg_one_relevant.rda")
id = unlist(lapply(rownames(p_kegg_one_relevant),function(y)
  unlist(strsplit(y,"_"))[1]))
KEGG = ifelse(id == "NA", NA,
              paste("<a href='http://www.genome.jp/dbget-bin/www_bget?",
                    id, "'>", id, "</a>", sep = ""))

df = data.frame(KEGG,p_kegg_one_relevant)
htmlRep = HTMLReport(shortName = "p_kegg_one_relevant",
                     reportDirectory = "./reports")
publish(df, htmlRep)
finish(htmlRep)

```

## 5 edgeR-GOseq pipeline

```
#####

library(Rsamtools)
library(GenomicFeatures)
library(GenomicAlignments)
library(org.Hs.eg.db)
library(edgeR)
library(VennDiagram)
library(goseq)

#####
# R code used for edgeR-GOseq pipeline (example with the PRJNA411984 dataset)
#####

gtfFile = "/media/scratch/genocode.v35.chr_patch_hapl_scaff.annotation.gtf"
txdb = makeTxDbFromGFF(gtfFile, format="gtf")
genes = exonsBy(txdb, by="gene")
indir = getwd()
files = list.files(indir, pattern = '*sort.bam')
bamLst = BamFileList(files, index=character(),obeyQname=TRUE)
PRJNA411984 = summarizeOverlaps(features = genes,
                                read=bamLst,
                                mode="Union",
                                singleEnd=FALSE,
                                ignore.strand=TRUE,
                                fragments=FALSE)

colnames(PRJNA411984)
# save(PRJNA411984,file="PRJNA411984.rda")

load("PRJNA411984.rda")
colnames(PRJNA411984)
# [1] "SRR6067723_sort.bam" "SRR6067725_sort.bam" "SRR6067729_sort.bam" "SRR6067731_sort.bam"
SampleName = c("SRR6067723", "SRR6067725", "SRR6067729", "SRR6067731")
tissue = c("cancer", "cancer", "normal", "normal")
replication = c("1", "2", "1", "2")
colData(PRJNA411984) = DataFrame(SampleName, tissue, replication)
PRJNA411984_met = PRJNA411984
# save(PRJNA411984_met,file="PRJNA411984_met.rda")

## Convert to edgeR format:
edgeRmatrix_STAR = assay(PRJNA411984)
groups = factor(tissue,
                levels=c("cancer", "normal"))
STAR_DATA_y = DGEList(counts=edgeRmatrix_STAR,group=groups)
dim(STAR_DATA_y)

#####
##### Filtering and normalization

keep <- filterByExpr(STAR_DATA_y)
```

```

summary(keep)
#   Mode   FALSE   TRUE
# logical 46841 20212

STAR_DATA_y_filt = STAR_DATA_y[keep, , keep.lib.sizes=FALSE]
dim(STAR_DATA_y_filt)

## annotation:
tmp = gsub("\\\\.\\.", "", row.names(STAR_DATA_y_filt))
row.names(STAR_DATA_y_filt) = tmp
STAR_Data = STAR_DATA_y_filt
preIDSymbol = AnnotationDbi::select(org.Hs.eg.db,
                                   keys=as.character(row.names(STAR_Data)),
                                   column=c("SYMBOL", "ENTREZID"),
                                   keytype="ENSEMBL")

STAR_Data$genes$ENSEMBL = row.names(STAR_Data)
m = match(STAR_Data$genes$ENSEMBL, preIDSymbol$ENSEMBL)
STAR_Data$genes$SYMBOL = preIDSymbol$SYMBOL[m]
STAR_Data$genes$ENTREZID = preIDSymbol$ENTREZID[m]
STAR_DATA_f = STAR_Data
STAR_DATA_f$genes = as.matrix(DataFrame(STAR_Data$genes$ENSEMBL,
                                         STAR_Data$genes$SYMBOL,
                                         STAR_Data$genes$ENTREZID))

colnames(STAR_DATA_f$genes) = c("ENSEMBL", "SYMBOL", "ENTREZID")
head(STAR_DATA_f$genes)
dim(STAR_DATA_f)

# Remove genes that have duplicate symbol:
d = duplicated(STAR_DATA_f$genes[,2])
summary(d)
#   Mode   FALSE   TRUE
# logical 14769  5443

#duplicates are removed and lib size is recalculated:
STAR_DATA_f = STAR_DATA_f[!d, , keep.lib.sizes=FALSE]
dim(STAR_DATA_f)

# The most used method (recommended) to normalize is:
PRJNA411984_norm_filt_annot = calcNormFactors(STAR_DATA_f)

# save(PRJNA411984_norm_filt_annot, file="PRJNA411984_norm_filt_annot.rda")

load("PRJNA411984_norm_filt_annot.rda")

#####
load("PRJNA413956_norm_filt_annot.rda")
View(PRJNA413956_norm_filt_annot)
summarized = PRJNA413956_norm_filt_annot
#####

disp=estimateCommonDisp(summarized)
disp$common.dispersion

```



```

tested=exactTest(dis)
topTags(tested)

genes=as.integer(p.adjust(tested$table$PValue[tested$table$logFC!=0],
                          method="BH")<.05)
names(genes)=row.names(tested$table[tested$table$logFC!=0,])
table(genes)

View(supportedOrganisms())
# hg38 --> hg19

supportedOrganisms()[supportedOrganisms()$Genome=="hg19",]
pwf=nullp(genes,"hg19","ensGene")
head(pwf)
GO.wall=goseq(pwf,"hg19","ensGene")
head(GO.wall)

GO.samp=goseq(pwf,"hg19","ensGene",method="Sampling",repcnt=1000, test.cats=c("GO:BP"))
head(GO.samp)

write.csv(GO.samp, file = "GO_PRJNA413956.csv")

#####
### KEGG_goseq
#####
pwf=nullp(genes,'hg19','ensGene')
KEGG=goseq(pwf,'hg19','ensGene',test.cats="KEGG")
head(KEGG)

write.csv(KEGG, file = "KEGG_PRJNA413956.csv")

#####
load("PRJNA411984_norm_filt_annot.rda")
View(PRJNA411984_norm_filt_annot)
summarized = PRJNA411984_norm_filt_annot
#####

disp=estimateCommonDisp(summarized)
disp$common.dispersion
tested=exactTest(disp)
topTags(tested)

genes=as.integer(p.adjust(tested$table$PValue[tested$table$logFC!=0],
                          method="BH")<.05)
names(genes)=row.names(tested$table[tested$table$logFC!=0,])
table(genes)

supportedOrganisms()[supportedOrganisms()$Genome=="hg19",]
pwf=nullp(genes,"hg19","ensGene")
head(pwf)
GO.wall=goseq(pwf,"hg19","ensGene")

```

```

head(GO.wall)

GO.samp=goseq(pwf,"hg19","ensGene",method="Sampling",repcnt=1000, test.cats=c("GO:BP"))
head(GO.samp)

write.csv(GO.samp, file = "GO_PRJNA411984.csv")

#####
### KEGG_goseq
#####
pwf=NULLp(genes,'hg19','ensGene')
KEGG=goseq(pwf,'hg19','ensGene',test.cats="KEGG")
head(KEGG)

write.csv(KEGG, file = "KEGG_PRJNA411984.csv")

#####
load("PRJNA218851_norm_filt_annot.rda")
View(PRJNA218851_norm_filt_annot)
summarized = PRJNA218851_norm_filt_annot
#####

disp=estimateCommonDisp(summarized)
disp$common.dispersion
tested=exactTest(disp)
topTags(tested)

genes=as.integer(p.adjust(tested$table$PValue[tested$table$logFC!=0],
                           method="BH")<.05)
names(genes)=row.names(tested$table[tested$table$logFC!=0,])
table(genes)

supportedOrganisms()[supportedOrganisms()$Genome=="hg19",]
pwf=NULLp(genes,"hg19","ensGene")
head(pwf)
GO.wall=goseq(pwf,"hg19","ensGene")
head(GO.wall)

GO.samp=goseq(pwf,"hg19","ensGene",method="Sampling",repcnt=1000, test.cats=c("GO:BP"))
head(GO.samp)

write.csv(GO.samp, file = "GO_PRJNA218851.csv")

#####
### KEGG_goseq
#####
pwf=NULLp(genes,'hg19','ensGene')
KEGG=goseq(pwf,'hg19','ensGene',test.cats="KEGG")
head(KEGG)

write.csv(KEGG, file = "KEGG_PRJNA218851.csv")

```

```
#####
load("TCGA_norm_filt_annot.rda")
View(TCGA_norm_filt_annot)
summarized = TCGA_norm_filt_annot
#####

disp=estimateCommonDisp(summarized)
disp$common.dispersion
tested=exactTest(disp)
topTags(tested)

genes=as.integer(p.adjust(tested$table$PValue[tested$table$logFC!=0],
                          method="BH")<.05)
names(genes)=row.names(tested$table[tested$table$logFC!=0,])
table(genes)

View(supportedOrganisms())
# hg38 --> hg19

supportedOrganisms()[supportedOrganisms()$Genome=="hg19",]
pwf=NULLp(genes,"hg19","ensGene")
head(pwf)
GO.wall=goseq(pwf,"hg19","ensGene")
head(GO.wall)

GO.samp=goseq(pwf,"hg19","ensGene",method="Sampling",repCnt=1000, test.cats=c("GO:BP"))
head(GO.samp)

write.csv(GO.samp, file = "GO_TCGA.csv")

#####
### KEGG_goseq
#####
pwf=NULLp(genes,'hg19','ensGene')
KEGG=goseq(pwf,'hg19','ensGene',test.cats="KEGG")
head(KEGG)

write.csv(KEGG, file = "KEGG_TCGA.csv")
```

## 6 Groups and test names

codification	name
GO:0038128	ERBB2_signaling_pathway
GO:0000165	MAPK_cascade
GO:0000187	activation_of_MAPK_activity
GO:0000185	activation_of_MAPKKK_activity
GO:0000186	activation_of_MAPKK_activity
GO:0000188	inactivation_of_MAPK_activity
GO:0032874	positive_regulation_of_stress-activated_MAPK_cascade
GO:0043408	regulation_of_MAPK_cascade
GO:0043409	negative_regulation_of_MAPK_cascade

*Continued on next page*

Table 1 – *Continued from previous page*

codification	name
GO:0043410	positive_regulation_of_MAPK_cascade
GO:0051403	stress-activated_MAPK_cascade
GO:0007219	Notch_signaling_pathway
GO:0007221	positive_regulation_of_transcription_of_Notch_receptor_target
GO:0008593	regulation_of_Notch_signaling_pathway
GO:0061314	Notch_signaling_involved_in_heart_development
GO:0045746	negative_regulation_of_Notch_signaling_pathway
GO:0045747	positive_regulation_of_Notch_signaling_pathway
GO:0060070	canonical_Wnt_signaling_pathway
GO:0007223	Wnt_signaling_pathway,calcium_modulating_pathway
GO:0016055	Wnt_signaling_pathway
GO:0030111	regulation_of_Wnt_signaling_pathway
GO:0030177	positive_regulation_of_Wnt_signaling_pathway
GO:0030178	negative_regulation_of_Wnt_signaling_pathway
GO:0035567	non-canonical_Wnt_signaling_pathway
GO:0060071	Wnt_signaling_pathway,planar_cell_polarity_pathway
GO:0060828	regulation_of_canonical_Wnt_signaling_pathway
GO:0090090	negative_regulation_of_canonical_Wnt_signaling_pathway
GO:0090263	positive_regulation_of_canonical_Wnt_signaling_pathway
GO:1904837	beta-catenin-TCF_complex_assembly
GO:1904886	beta-catenin_destruction_complex_disassembly
GO:0046427	positive_regulation_of_receptor_signaling_pathway_via_JAK-STAT
GO:0007259	receptor_signaling_pathway_via_JAK-STAT
GO:0046426	negative_regulation_of_receptor_signaling_pathway_via_JAK-STAT
GO:0060397	growth_hormone_receptor_signaling_pathway_via_JAK-STAT
GO:0060395	SMAD_protein_signal_transduction
GO:0006978	DNA_damage_response,signal_transduction_by_p53_class_mediator_resulting_in_transcription_of_p21_class_mediator
GO:0042771	intrinsic_apoptotic_signaling_pathway_in_response_to_DNA_damage_by_p53_class_mediator
GO:0006977	DNA_damage_response,signal_transduction_by_p53_class_mediator_resulting_in_cell_cycle_arrest
GO:0030330	DNA_damage_response,signal_transduction_by_p53_class_mediator
GO:0043517	positive_regulation_of_DNA_damage_response,signal_transduction_by_p53_class_mediator
GO:0043518	negative_regulation_of_DNA_damage_response,signal_transduction_by_p53_class_mediator
GO:0072332	intrinsic_apoptotic_signaling_pathway_by_p53_class_mediator
GO:1901796	regulation_of_signal_transduction_by_p53_class_mediator
GO:1902166	negative_regulation_of_intrinsic_apoptotic_signaling_pathway_in_response_to_DNA_damage_by_p53_class_mediator
GO:0032088	negative_regulation_of_NF-kappaB_transcription_factor_activity
GO:0007249	I-kappaB_kinase/NF-kappaB_signaling
GO:0007250	activation_of_NF-kappaB-inducing_kinase_activity
GO:0038061	NIK/NF-kappaB_signaling
GO:0043122	regulation_of_I-kappaB_kinase/NF-kappaB_signaling
GO:0043123	positive_regulation_of_I-kappaB_kinase/NF-kappaB_signaling
GO:0043124	negative_regulation_of_I-kappaB_kinase/NF-kappaB_signaling
GO:0051092	positive_regulation_of_NF-kappaB_transcription_factor_activity
GO:1901222	regulation_of_NIK/NF-kappaB_signaling

*Continued on next page*

Table 1 – Continued from previous page

codification	name
GO:1901223	negative_regulation_of_NIK/NF-kappaB_signaling
GO:1901224	positive_regulation_of_NIK/NF-kappaB_signaling

Table 1: Abbreviated and complete names of GO groups

abbreviated	complete
hsa01521	EGFR_tyrosine_kinase_inhibitor_resistance
hsa04012	ErbB_signaling_pathway
hsa04010	MAPK_signaling_pathway
hsa04330	Notch_signaling_pathway
hsa04151	PI3K-Akt_signaling_pathway
hsa04350	TGF-beta_signaling_pathway
hsa04310	Wnt_signaling_pathway
hsa04630	JAK-STAT_signaling_pathway
hsa04340	Hedgehog_signaling_pathway
hsa04210	Apoptosis
hsa04115	p53_signaling_pathway
hsa04370	VEGF_signaling_pathway
hsa04064	NF-kappa_B_signaling_pathway
hsa05210	Colorectal_cancer

Table 2: Abbreviated and complete names of KEGG groups

## 7 Simulation study

Two stochastic models have been used. The first model uses the Poisson distribution to simulate random counts and the second model replace it with the negative binomial distribution. The mean of the Poisson count will be denoted as  $\lambda$ . The negative binomial count will have parameters  $(\mu, \phi)$  in such a way the variance of the distribution is  $\mu + \mu^2/\phi$ . The two models are equal except the chosen (Poisson or negative binomial) distribution to generate random counts.

For each model we have control and case distributions. Really, one control distribution and different case distributions where the mean of the case distribution varies. When the Poisson distribution is used then the control distribution is a Poisson with  $\lambda = 20$  i.e. a mean of 20. The case distributions will have  $\lambda = \delta$  where varies,  $\delta$  goes from 1 to 40 with increments of 2. If the negative binomial distribution is used then the control distribution has parameters  $(\mu, \phi) = (20, 150)$ . The case distributions will have means from 1 to 40 with increments of 2. The dispersion parameter  $\phi$  is constantly equal to 150.

A non significant gene will have independent realizations of the same distribution chosen for the control and case sample within each pair i.e. there is no difference between the means of different conditions. The case count will be equal to the just generated count in the control component plus a random count generated using the case distribution. Note that the mean of the case distribution is incremented.

A total number of 1000 genes are considered. The gene sets are non

code	name
1	PRJNA218851_between-pair_CuzickEdwards
2	PRJNA218851_complete_CuzickEdwards
3	PRJNA218851_between-pair_DiggleMorrisMorton
4	PRJNA218851_complete_DiggleMorrisMorton
5	PRJNA218851_between-pair_diggle90
6	PRJNA218851_complete_diggle90
7	PRJNA411984_between-pair_CuzickEdwards
8	PRJNA411984_complete_CuzickEdwards
9	PRJNA411984_between-pair_DiggleMorrisMorton
10	PRJNA411984_complete_DiggleMorrisMorton
11	PRJNA411984_between-pair_diggle90
12	PRJNA411984_complete_diggle90
13	PRJNA413956_between-pair_CuzickEdwards
14	PRJNA413956_complete_CuzickEdwards
15	PRJNA413956_between-pair_DiggleMorrisMorton
16	PRJNA413956_complete_DiggleMorrisMorton
17	PRJNA413956_between-pair_diggle90
18	PRJNA413956_complete_diggle90
19	TCGA_between-pair_CuzickEdwards
20	TCGA_complete_CuzickEdwards
21	TCGA_between-pair_DiggleMorrisMorton
22	TCGA_complete_DiggleMorrisMorton
23	TCGA_between-pair_diggle90
24	TCGA_complete_diggle90

Table 3: Code and names for tests

overlapping gene sets with 20 genes. A total number of 50 groups are considered and all gene sets are significant.

We test if each gene set is significant. Really, we will obtain the p-value. Note that three different tests have been proposed and two randomization distributions have been used. We have six p-values per simulation.

Figures 1 and 2 display the mean p-values for the 50 significant gene sets considered. Figure 1 shows how the differences up to 20 units are clearly detected. Greater differences are not so clearly detected. This behaviour should be expected because the mean and the variance are equal for random Poisson counts.

Figure 2 displays a similar behaviour but a worse performance. The p-values are higher for the same difference of means. Note that the variance is the original mean plus its square divided by a fixed dispersion i.e. the variance is greater than the Poisson distribution. This can be shown in the observed p-values. However, differences up to almost the double of the original means are detected in both cases using the six different p-values.

The code used can be found in the last section of the file `SupplementaryMaterialMethods_pointgene.pdf`.

## 7.1 Code

### 7.1.1 Negative binomial counts

Generate a count matrix.

```
setwd("~/Nextcloud/pointgene17/SupplementaryMaterial/")
library(OMICfp2)
```

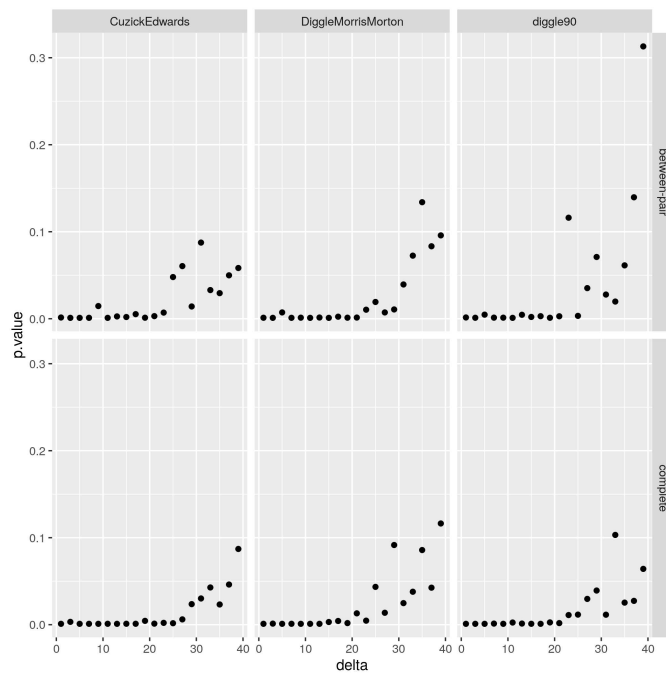


Figure 1: Observed mean p-values for different differences of means ( $\delta$  using Poisson distributions).

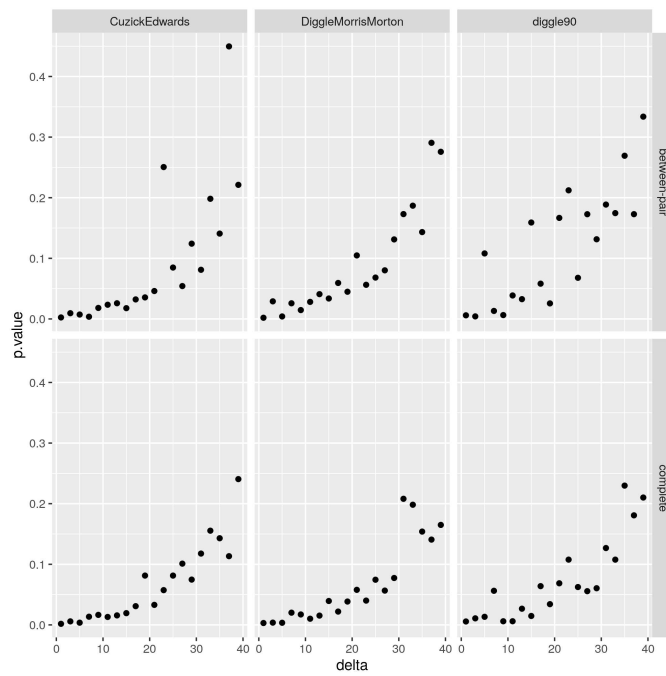


Figure 2: Observed mean p-values for different differences of means ( $\delta$  using negative binomial distributions).



```

n0 = 20
N0 = 1000
ngs = 20
ngsc = 50
gsc0 = vector("list",ngsc)
row = rep(1:ngsc,each=ngs)
for(i in 1:ngsc)
  gsc0[[i]] = which(row == i)

m0 = 30
deltas = seq(1,40,2)
size = 150

tests = c("CuzickEdwards", "DiggleMorrisMorton", "diggle90")
types = c("between-pair", "complete")

for(testLabelling in tests){
  for(type in types){
    result = NULL
    for(delta in deltas){
      mu0 = c(30,rep(30+delta,50))
      param0 = cbind(mu0,rep(size,51))
      x0 = rPairedNB2(n=n0,N=N0,param = param0,gsc = gsc0)
      y0 = cbind(rep(1:n0,each=2),rep(0:1,n0))
      pp = ppgene(x=x0,y=y0,testPair=proportion.p,nsim = 1,type,
                  gsc = gsc0,testLabelling,
                  nmax=1000,file.labelling = NULL)
      result = rbind(result,c(delta,mean(1-unlist(pp[[2]]))))
    }
    save(result,file=paste0("results/SimulationStudy_NB_30_",
                           testLabelling,"_",type,".rda"))
  }
}

```

Now some plots to explore the results.

```

deltas = seq(1,40,2)
tests = c("CuzickEdwards", "DiggleMorrisMorton", "diggle90")
types = c("between-pair", "complete")
nrow.df = length(deltas)*length(tests)*length(types)
df = data.frame(matrix(NA,nrow = nrow.df,ncol = 4))
names(df) = c("test", "type", "delta", "p.value")

df = NULL
for(i in 1:length(tests)){
  for(j in 1:length(types)){
    load(paste0("results/SimulationStudy_NB_30_",tests[i],"_",
               types[j],".rda"))
    r0 = result
    df = rbind(df,cbind(rep(i,length(deltas)),rep(j,length(deltas)),r0))
  }
}
df = data.frame(df)

```

```

names(df) = c("test", "type", "delta", "p.value")
df$test = factor(df$test, levels=1:3,
                 labels=c("CuzickEdwards", "DiggleMorrisMorton", "diggle90"))
df$type = factor(df$type, levels=1:2, labels=c("between-pair", "complete"))

p = ggplot(df, aes(x=delta, y=p.value)) + geom_point()
p = p + facet_grid(type ~ test)
ggsave(p, file = "results/SimulationStudyNB1.png")

```

### 7.1.2 Poisson counts

```

library(OMICfpp2)
n0 = 20
N0 = 1000
ngs = 20
ngsc = 50
gsc0 = vector("list", ngsc)
row = rep(1:ngsc, each=ngs)
for(i in 1:ngsc)
  gsc0[[i]] = which(row == i)

lambdas = seq(1, 40, 2)

tests = c("CuzickEdwards", "DiggleMorrisMorton", "diggle90")
types = c("between-pair", "complete")

for(testLabelling in tests){
  for(type in types){
    result = NULL
    for(lambda in lambdas){
      lambda0 = c(20, rep(lambda, ngsc))
      x0 = rPairedPoisson2(n=n0, N=N0, lambda=lambda0, gsc=gsc0)
      y0 = cbind(rep(1:n0, each=2), rep(0:1, n0))
      pp = ppgene(x=x0, y=y0, testPair=proportion.p, nsim = 1, type,
                  gsc = gsc0, testLabelling,
                  nmax=1000, file.labelling = NULL)
      result = rbind(result, c(lambda, mean(1-unlist(pp[[2]]))))
    }
    save(result, file=paste0("results/SimulationStudy_Poisson_30_",
                             testLabelling, "_", type, ".rda"))
  }
}

```

```

lambdas = seq(1, 40, 2)
tests = c("CuzickEdwards", "DiggleMorrisMorton", "diggle90")
types = c("between-pair", "complete")
nrow.df = length(lambdas)*length(tests)*length(types)
df = data.frame(matrix(NA, nrow = nrow.df, ncol = 4))
names(df) = c("test", "type", "delta", "p.value")

df = NULL

```

```

for(i in 1:length(tests)){
  for(j in 1:length(types)){
    load(paste0("results/SimulationStudy_Poisson_30_",tests[i],"_",
               types[j],".rda"))
    r0 = result
    df = rbind(df,cbind(rep(i,length(lambdas)),rep(j,length(lambdas)),r0))
  }
}
df = data.frame(df)
names(df) = c("test","type","delta","p.value")
df$test = factor(df$test,levels=1:3,
                 labels=c("CuzickEdwards","DiggleMorrisMorton","diggle90"))
df$type = factor(df$type,levels=1:2,labels=c("between-pair","complete"))

p = ggplot(df,aes(x=delta,y=p.value)) + geom_point()
p = p + facet_grid(type ~ test)

ggsave(p,file = "results/SimulationStudyPoisson1.png")

```

## References

- [1] Alberto Berral-Gonzalez, Angela L. Riffo-Campos, and Guillermo Ayala. "OMICfpp: a fuzzy approach for paired RNA-Seq counts". In: *BMC Genomics* 20.1 (Apr. 2019), p. 259. ISSN: 1471-2164. URL: <https://doi.org/10.1186/s12864-019-5496-5>.