Prepare data for DeepIP model training and testing – yeast

Xiaohui Wu and Wenbin Ye

Last modified 2025-03-30

Contents

1 Overview

This documentation describes how to generate training and test fasta files for training and testing DeepIP models. Here we used two small yeast species – Saccharomyces cerevisiae (brewer's yeast) and Schizosaccharomyces pombe (fission yeast) for demonstration.

The differences between the yeast demo and the Arabidopsis (Ath) demo are following:

1) The yeast genome annotation is stored in GFF3 file, while the Ath genome is stored in TXDB package.
2) A BSgenome object was built from the genome assembly of yeast, while the Ath genome is already stored in BSgenome.Athaliana.TAIR.TAIR9.
3) PAs from Arabidopsis were obtained from public databases including PolyADB 3 and PolyASite 2.0, while the yeast PAs were identified by PolyASeqTrap from 3'READS data (SRR5276077 for S. cerevisiae; SRR5276080 for S. pombe)
4) Only high quality PAs (level=V1 from the PolyAseqTrap result) were used for S.pombe, while in the Arabidopsis demo, both high and low quality PAs were used.
5) The yeast genome is small, we need to adjust the parameters for `getFreePARangesIn3UTR` to get enough PA-free regions.
6) PolyA signal distribution was not considered in this demo, while it is easy to use mouse/human polyA signals for yeast as well.

2 Input data preparation

Please follow the BSgenome tutorial to build BSgenome package for `Saccharomyces_cerevisiae.R64-1-1.dna.toplevel.fa` and `Schizosaccharomyces_pombe.ASM294v2.dna.toplevel.fa`.

The input data of S.cerevisiae includes:

- PA data: SRR5276077_PAC_table.Rdata, the PA list abtained from PolyAseqTrap. We considered coord_level=V1 as high quality PAs.
- Genome assembly: BSgenome.Scerevisia.ENSEMBL.R64
- Genome annotation: Saccharomyces_cerevisiae.R64-1-1.113.gtf

The input data of S. pombe includes:

- PA data: SRR5276080_PAC_table.Rdata, the PA list abtained from PolyAseqTrap. We considered coord_level=V1 as high quality PAs.
- Genome assembly: BSgenome.Spombe.ENSEMBL.ASM294v2
- Genome annotation: Schizosaccharomyces_pombe.ASM294v2.60.gff3

```
setwd('D:/data/projectData/polyAseqTrap/yeast/')

library(movAPA)
library(DeepIP)

#install.packages("BSgenome.Scerevisia.ENSEMBL.R64_1.4.2.tar.gz",
#                 repos = NULL, type = "source")
#install.packages("BSgenome.Spombe.ENSEMBL.ASM294v2_1.4.2.tar.gz",
#                repos = NULL, type = "source")
```

```
## Scerevisia
library(BSgenome.Scerevisia.ENSEMBL.R64)
gffFile='Saccharomyces_cerevisiae.R64-1-1.113.gtf'
bsgenome=BSgenome.Scerevisia.ENSEMBL.R64
PAfile='SRR5276077_PAC_table.Rdata'
seqlengths(bsgenome)
opre='sce'
```

```
library(BSgenome.Spombe.ENSEMBL.ASM294v2)
gffFile='Schizosaccharomyces_pombe.ASM294v2.60.gff3'
bsgenome=BSgenome.Spombe.ENSEMBL.ASM294v2
PAfile='SRR5276080_PAC_table.Rdata'
opre='spo'
```

Please run the above code chunk separately. Then the following code chunk will process the given gff file and PA file to obtain high quality (HQ) and low quality (LQ) PAs from the PA data, and 3'UTR regions from the GFF/GTF file.

The output files (e.g., sceHQV1.bed, sceLQ.bed, and sceTXDB_UTR.bed) will be used for DeepIP data preparation.

```r
## read PA list from polyAseqTrap
PA=readRDS(PAfile)
head(PA)
colnames(PA)
table(PA$coord_level)
table(PA$coord_level, PA$isArich_coord)

PA=PA[, c('seqnames','strand','coord','coord_level', 'isArich_coord')]
colnames(PA)=c('chr','strand','coord','level','isArich')

## plot PA list in different levels (V1 is HQ)
PACds=movAPA::readPACds(PA)
PAFAfiles=movAPA::faFromPACds(PACds, bsgenome=bsgenome,
                              what='updn', fapre=PAfile, byGrp = 'level')
movAPA::plotATCGforFAfile(faFiles=PAFAfiles, ofreq=FALSE,
                          opdf=TRUE, mergePlots = TRUE,
                          filepre=paste0(opre, 'PAs'))

## output HQ and LQ PA.bed files
write.table(PA[PA$level=='V1', c('chr','strand','coord')],
            file=paste0(opre,'HQV1.bed'), col.names=F, row.names = F, sep="\t", quote=F)
write.table(PA[PA$level!='V1', c('chr','strand','coord')],
            file=paste0(opre,'LQ.bed'), col.names=F, row.names = F, sep="\t", quote=F)

## read 3UTR annotation from gff3
gff=movAPA::parseGff(gffFile)
table(gff$anno.need$type)

UTR=gff$anno.need[gff$anno.need$type=='three_prime_UTR',
                  c('seqnames', 'start','end', 'strand')]
dim(UTR)
colnames(UTR)=c('chr','start','end','strand')
UTR$coord=UTR$end
UTR$coord[UTR$strand=='-']=UTR$start[UTR$strand=='-']

UTR=UTR[UTR$chr %in% seqnames(bsgenome), ]
table(UTR$chr)

## plot 3UTR end of gff3 (sce UTR end is not correct!)
utrPACds=movAPA::readPACds(UTR)
utrFaFile=movAPA::faFromPACds(utrPACds, bsgenome=bsgenome, what='updn',
                              fapre=paste0(opre, 'UTR'), byGrp = 'strand')
movAPA::plotATCGforFAfile(faFiles=utrFaFile, ofreq=FALSE,
                          opdf=TRUE, mergePlots = TRUE, filepre=paste0(opre, 'UTR'))

write.table(UTR[, 1:4], file=paste0(opre,'TXDB_UTR.bed'),
            col.names=F, row.names = F, sep="\t", quote=F)
```

3 Prepare DeepIP data for S.pombe

The pipeline for preparing DeepIP training and test data for S.pombe is the same as that for Arabidopsis. But here we did not consider polyA siganl (`grams=NULL`). And, due to the small genome size, we did not use

LQ data in generating `utrsNoPA.RDS`.

```
hqPAfiles='spoHQV1.bed'
allPAfiles=NULL # not using LQ in S.pombe to make utrsNoPA larger

bsgenome=BSgenome.Spombe.ENSEMBL.ASM294v2
seqnames(bsgenome)
chrs=seqnames(bsgenome)

txdb=read.table('spoTXDB_UTR.bed', header=F)
colnames(txdb)=c('seqnames','start','end','strand')

grams=NULL
gramsPriority=NULL

outputDir='spo_deepIP_data/'

getArichPAseqsIn3UTR(paBedFiles=hqPAfiles, txdb=txdb, extUTRLen=5000,
                     bsgenome=bsgenome, chrs=chrs, grams=grams,
                     gramsPriority=gramsPriority,
                     outputSeqPre=paste0(outputDir,
                                         "realPA_Arich_seq/realPA_Arich_seq"))
```

This step is the only step that need manual inspection.

We need to adjust the `extUTRLen` and `paBedFiles` to obtain enough PA-free 3'UTR regions for searching IP sites.

For example, we can set `extUTRlen` from 5000 to 50,000, and set `allPAfiles=NULL` or `allPAfiles=spoLQ.bed` to check the results. Finally, in this case, we used the `allPAfiles=NULL` and `extUTRlen=20000`.

```
## larger extUTRlen to get more utrsNoPA for IP sites
## extUTRlen=10,000 --> 6290 utrs --> 3819 IPs
## extUTRlen=20,000 --> 6821 utrs --> 4162 IPs
## extUTRlen=50,000 --> 6821 utrs --> 4162 IPs
## no LQPA & extUTRlen=20,000 --> 7893 utrs --> 4834 IPs
utrsNoPA=getFreePARangesIn3UTR(paBedFiles=c(hqPAfiles, allPAfiles),
                               txdb=txdb, extUTRLen=20000, extPA=200,
                               outputRds='utrsNoPA.RDS', chrs=chrs,
                               plotFA=TRUE, N=5000, bsgenome=bsgenome,
                               outputDir=outputDir)


getArichIPseqs(regionsNoPARDS=paste0(outputDir,'utrsNoPA.RDS'),
               bsgenome=bsgenome,
               grams=grams,
               N=NULL,
               outputSeqPre=paste0(outputDir, "IP_Arich_seq/IP_Arich_seq"))

# max real num: 17624
fas=statCntFas(path=paste0(outputDir, 'realPA_Arich_seq/'),
               filePre='realPA_Arich_seq.')

# max IP num: 4834
fas=statCntFas(path=paste0(outputDir, 'IP_Arich_seq'),
```

```
               filePre='IP_Arich_seq.')

## These numbers are the total number of real and false samples.
```

```
dir.create(paste0(outputDir,'modelDataSplits'))

## true train and test files
split2TrainTest(path=paste0(outputDir,'realPA_Arich_seq/'),
                filePre='realPA_Arich_seq.',
                dir1=paste0(outputDir,
                            'modelDataSplits/realPA_Arich_seq_train_per70'),
                dir2=paste0(outputDir,
                            'modelDataSplits/realPA_Arich_seq_test_per30'),
                per1=0.7,
                label=":1")

## false train and test files
split2TrainTest(path=paste0(outputDir,'IP_Arich_seq'),
                filePre='IP_Arich_seq.',
                dir1=paste0(outputDir,
                            'modelDataSplits/IP_Arich_seq_train_per70'),
                dir2=paste0(outputDir,
                            'modelDataSplits/IP_Arich_seq_test_per30'),
                per1=0.7,
                label=":0")

REALPA_PERC=statCntFas(path=paste0(outputDir, 'realPA_Arich_seq/'),
                       filePre='realPA_Arich_seq.')

seqDir=paste0(outputDir,'modelDataSplits/')

## randomly sample 3,000 sequences as real (positive) training data
files=getNseqs(seqDir=paste0(seqDir, 'realPA_Arich_seq_train_per70'),
               N=3000,
               nsplits=1,
               outputPre=paste0(seqDir,
                                "realPA_Arich_seq_train_per70_3000s_1splits"),
               perc=REALPA_PERC)


## plot the real training fa file for validation
movAPA::plotATCGforFAfile(faFiles=files, ofreq=FALSE,
                          opdf=T, refPos=101)

## randomly sample 2,000 sequences as false (IP, negative) training data
files=getNseqs(seqDir=paste0(seqDir, 'IP_Arich_seq_train_per70'),
               N=3000,
               nsplits=1,
               outputPre=paste0(seqDir,
                                'IP_Arich_seq_train_per70_3000s_1splits'),
               perc=REALPA_PERC)

## plot the IP training fa file for validation
```

```
movAPA::plotATCGforFAfile(faFiles=files, ofreq=FALSE,
                          opdf=T, refPos=101)


## test
files=getNseqs(seqDir=paste0(seqDir, 'realPA_Arich_seq_test_per30'),
               N=1400,
               nsplits=1,
               outputPre=paste0(seqDir,
                                'realPA_Arich_seq_test_per30_1400s_1splits'),
               perc=REALPA_PERC)


movAPA::plotATCGforFAfile(faFiles=files, ofreq=FALSE, opdf=T, refPos=101)

files=getNseqs(seqDir=paste0(seqDir, 'IP_Arich_seq_test_per30'),
               N=1400,
               nsplits=1,
               outputPre=paste0(seqDir,
                                'IP_Arich_seq_test_per30_1400s_1splits'),
               perc=REALPA_PERC)

movAPA::plotATCGforFAfile(faFiles=files, ofreq=FALSE, opdf=T, refPos=101)


combineFaFiles_fast(paste0(seqDir,
                     c('realPA_Arich_seq_train_per70_3000s_1splits.split.1.fa',
                      'IP_Arich_seq_train_per70_3000s_1splits.split.1.fa')),
                    ofile = paste0(seqDir, 'train.3000T.3000F.fa'),
                    verbose = TRUE)


combineFaFiles_fast(paste0(seqDir,
                     c('realPA_Arich_seq_test_per30_1400s_1splits.split.1.fa',
                      'IP_Arich_seq_test_per30_1400s_1splits.split.1.fa')),
                    ofile = paste0(seqDir, 'test.1400T.1400F.fa'),
                    verbose = TRUE)
```

4 Train and test DeepIP model for S.pombe

```
setwd('spo_deepIP_data/modelDataSplits/')

## train
trainDeepIP(condaEnv="C:/Users/LENOVO/anaconda3/envs/DeepIP/",
            inTrainSeq='train.3000T.3000F.fa',
            outTrainedModel='train.3000T.3000F.epoch100.hdf5',
            epoch=100)

## test
testDeepIP(condaEnv="C:/Users/LENOVO/anaconda3/envs/DeepIP/",
           inTestSeq='test.1400T.1400F.fa',
```

```
           inTrainedModel='train.3000T.3000F.epoch100.hdf5',
           outTestCsv='train.3000T.3000F.epoch100_ON_test.1400T.1400F.csv',
           seqLabel='')
# metrics
statDeepRes('train.3000T.3000F.epoch100_ON_test.1400T.1400F.csv',
            ofile='train.3000T.3000F.epoch100_ON_test.1400T.1400F.stat.csv')

# TP/FP fa profile
plotFaDeepRes('train.3000T.3000F.epoch100_ON_test.1400T.1400F.csv',
              fafile='test.1400T.1400F.fa')
```

5 Prepare DeepIP data for S.cerevisiae

The pipeline is the same as S.pombe, with the only difference is we also use the low quality PAs `allPAfiles='sceLQ.bed'`.

```
setwd('D:/data/projectData/polyAseqTrap/yeast/')
hqPAfiles='sceHQV1.bed'
allPAfiles='sceLQ.bed'

bsgenome=BSgenome.Scerevisia.ENSEMBL.R64
seqnames(bsgenome)
chrs=seqnames(bsgenome)

txdb=read.table('sceTXDB_UTR.bed', header=F)
colnames(txdb)=c('seqnames','start','end','strand')

grams=NULL
gramsPriority=NULL

outputDir='sce_deepIP_data/'

getArichPAseqsIn3UTR(paBedFiles=hqPAfiles, txdb=txdb, extUTRLen=5000,
                     bsgenome=bsgenome, chrs=chrs, grams=grams,
                     gramsPriority=gramsPriority,
                     outputSeqPre=paste0(outputDir,
                                         "realPA_Arich_seq/realPA_Arich_seq"))

## larger extUTRlen to get more utrsNoPA for IP sites
## extUTRlen=10000 --> 8360 utrs --> 4593 IPs
## extUTRlen=10000 --> 8690 utrs --> 4811 IPs
utrsNoPA=getFreePARangesIn3UTR(paBedFiles=c(hqPAfiles, allPAfiles),
                               txdb=txdb, extUTRLen=20000, extPA=200,
                               outputRds='utrsNoPA.RDS', chrs=chrs,
                               plotFA=TRUE, N=5000, bsgenome=bsgenome,
                               outputDir=outputDir)


getArichIPseqs(regionsNoPARDS=paste0(outputDir,'utrsNoPA.RDS'),
               bsgenome=bsgenome,
               grams=grams,
               N=NULL,
```

```
                outputSeqPre=paste0(outputDir, "IP_Arich_seq/IP_Arich_seq"))

# max real num: 10557
fas=statCntFas(path=paste0(outputDir, 'realPA_Arich_seq/'),
                filePre='realPA_Arich_seq.')

# max IP num: 4805
fas=statCntFas(path=paste0(outputDir, 'IP_Arich_seq'),
                filePre='IP_Arich_seq.')

dir.create(paste0(outputDir,'modelDataSplits'))

## true train and test files
split2TrainTest(path=paste0(outputDir,'realPA_Arich_seq/'),
                 filePre='realPA_Arich_seq.',
                 dir1=paste0(outputDir,
                             'modelDataSplits/realPA_Arich_seq_train_per70'),
                 dir2=paste0(outputDir,
                             'modelDataSplits/realPA_Arich_seq_test_per30'),
                 per1=0.7,
                 label=":1")

## false train and test files
split2TrainTest(path=paste0(outputDir,'IP_Arich_seq'),
                 filePre='IP_Arich_seq.',
                 dir1=paste0(outputDir,
                             'modelDataSplits/IP_Arich_seq_train_per70'),
                 dir2=paste0(outputDir,
                             'modelDataSplits/IP_Arich_seq_test_per30'),
                 per1=0.7,
                 label=":0")

REALPA_PERC=statCntFas(path=paste0(outputDir, 'realPA_Arich_seq/'),
                       filePre='realPA_Arich_seq.')

seqDir=paste0(outputDir,'modelDataSplits/')

## randomly sample 3,000 sequences as real (positive) training data
files=getNseqs(seqDir=paste0(seqDir, 'realPA_Arich_seq_train_per70'),
                N=3000,
                nsplits=1,
                outputPre=paste0(seqDir,
                                 "realPA_Arich_seq_train_per70_3000s_1splits"),
                perc=REALPA_PERC)


## plot the real training fa file for validation
movAPA::plotATCGforFAfile(faFiles=files, ofreq=FALSE,
                          opdf=T, refPos=101)

## randomly sample 2,000 sequences as false (IP, negative) training data
files=getNseqs(seqDir=paste0(seqDir, 'IP_Arich_seq_train_per70'),
                N=3000,
```

```
                nsplits=1,
                outputPre=paste0(seqDir,
                                 'IP_Arich_seq_train_per70_3000s_1splits'),
                perc=REALPA_PERC)

## plot the IP training fa file for validation
movAPA::plotATCGforFAfile(faFiles=files, ofreq=FALSE,
                          opdf=T, refPos=101)


## test
files=getNseqs(seqDir=paste0(seqDir, 'realPA_Arich_seq_test_per30'),
                N=1400,
                nsplits=1,
                outputPre=paste0(seqDir,
                                 'realPA_Arich_seq_test_per30_1400s_1splits'),
                perc=REALPA_PERC)


movAPA::plotATCGforFAfile(faFiles=files, ofreq=FALSE, opdf=T, refPos=101)

files=getNseqs(seqDir=paste0(seqDir, 'IP_Arich_seq_test_per30'),
                N=1400,
                nsplits=1,
                outputPre=paste0(seqDir,
                                 'IP_Arich_seq_test_per30_1400s_1splits'),
                perc=REALPA_PERC)

movAPA::plotATCGforFAfile(faFiles=files, ofreq=FALSE, opdf=T, refPos=101)


combineFaFiles_fast(paste0(seqDir,
                      c('realPA_Arich_seq_train_per70_3000s_1splits.split.1.fa',
                      'IP_Arich_seq_train_per70_3000s_1splits.split.1.fa')),
                    ofile = paste0(seqDir, 'train.3000T.3000F.fa'),
                    verbose = TRUE)


combineFaFiles_fast(paste0(seqDir,
                     c('realPA_Arich_seq_test_per30_1400s_1splits.split.1.fa',
                     'IP_Arich_seq_test_per30_1400s_1splits.split.1.fa')),
                    ofile = paste0(seqDir, 'test.1400T.1400F.fa'),
                    verbose = TRUE)
```

6 Train and test DeepIP model for S.cerevisiae

```
############## sce: train ##################
setwd('sce_deepIP_data/modelDataSplits/')
trainDeepIP(condaEnv="C:/Users/LENOVO/anaconda3/envs/DeepIP/",
            inTrainSeq='train.3000T.3000F.fa',
            outTrainedModel='train.3000T.3000F.epoch100.hdf5',
```

```
            epoch=100)


############## sce: test ##################
testDeepIP(condaEnv="C:/Users/LENOVO/anaconda3/envs/DeepIP/",
           inTestSeq='test.1400T.1400F.fa',
           inTrainedModel='train.3000T.3000F.epoch100.hdf5',
           outTestCsv='train.3000T.3000F.epoch100_ON_test.1400T.1400F.csv',
           seqLabel='')

# metrics
statDeepRes('train.3000T.3000F.epoch100_ON_test.1400T.1400F.csv',
            ofile='train.3000T.3000F.epoch100_ON_test.1400T.1400F.stat.csv')

# TP/FP fa profile
plotFaDeepRes('train.3000T.3000F.epoch100_ON_test.1400T.1400F.csv',
              fafile='test.1400T.1400F.fa')
```