Prepare data for DeepIP model training and testing

Xiaohui Wu

Last modified 2025-03-30

Contents

1 Overview

This documentation describes how to generate training and test fasta files for deep learning models. We used the model species – Arabidopsis for demonstration.

The data preparation of the DeepIP model is as following:

A. Construction of positive data for DeepIP:

1) We first filter A-rich real (high-quality) polyA sites, requiring at least one 10 nt window from upstream 10 nt to downstream 10 nt of each site containing at least 6 consecutive As or a total of 7 As. These sites will normally be considered as internal priming sites and removed in most studies. Here we used 3'seq data (e.g., 3¡äREADS, or DRS,) not suffering internal priming issue to select A-rich but real sites.

2) We compute the proportions of polyA signals in the sites from step 1, and each polyA site would be assigned one polyA signal. If there is no signal, then the polyA site will be assigned “NOPAS”. If there are multiple signals, then the polyA site could be assigned in the order “AATAAA > ATTAAA > other motifs”. Therefore, a polyA signal distribution of real polyA sites would be obtained, like AAUAAA=50%, ATTAAA=20%, etc.

3) We randomly selected real A-rich polyA sites *following the polyA signal distribution* for training and testing. This is to avoid potentially under-representing or over-representing of polyA sites with or without some signals.

B. Construction of negative data for DeepIP:

1) We first obtained regions from the entire genome that are completely free of polyA sites, which do not contain any polyA sites and are not within the 200 nt range of annotated 3¡ä UTRs.

2) Then we scanned A stretches (fragment with at least six As) in these polyA site free regions and searched for hexamers in the corresponding species within 10-50 nt upstream of the A stretch.

3) Finally, a random position within the 10 nt upstream and downstream range of the center of the A stretch was selected as the false polyA site. The final negative dataset contains sequences of 200 nt, with the center of the sequence surrounded by a stretch, and the hexamer signal for each sequence is recorded.

4) We randomly samples negative sequences *following the polyA signal distribution of real polyA sites* for training and testing.

It should be noted that, we have considered the distribution of polyA signals during both model training and testing. This is because in plants, especially animals, polyA sites with signals such as AATAAA have a higher proportion and more distinct sequence characteristics. If the signal distribution is not considered for random sampling of polyA sites for training, it may lead to biased distribution of polyA sites with different polyA signals, resulting in an erroneous overestimation of the model’s performance.

In fact, implementing a model without considering signal distribution is easier, but considering signal distribution is more reasonable. For Arabidopsis with only AAUAAA is relatively dominant (in ~10% of polyA sites), we only considered AAUAAA and NOPAS.

For species with sparse APA research or unclear signal distribution, DeepIP is also fully applicable and even easier to implement (just set `grams=NULL`). Therefore, the strict prioritization of AATAAA > ATTAAA > other motifs would definitely not introduce bias in the dataset or favor well-characterized polyA sites. Instead, it considered the usage of different polyA signals and the varied prevalence of hexamers across species.

2 Data input: PA list and genome annotation

For this demo, the input data include:

- **genome assembly**: BSgenome.Athaliana.TAIR.TAIR9
- **genome annotation**: TxDb.Athaliana.BioMart.plantsmart51
- **high quality PAs**: DRS_Ath.bed, this file contains PAs from DRS sequencing, which is not affected by internal priming. It was stored in the DeepIP package.
- **other potential PAs**: PlantAPAdb_Ath.bed, this file contains PAs downloaded from PlantAPAdb. This file is optional. It was stored in the DeepIP package.

```r
library(DeepIP, warn.conflicts = FALSE, quietly=TRUE)

## genome assembly
library(BSgenome.Athaliana.TAIR.TAIR9,
        warn.conflicts = FALSE, quietly=TRUE)
bsgenome=BSgenome.Athaliana.TAIR.TAIR9

## genome annotation
library(TxDb.Athaliana.BioMart.plantsmart51,
        warn.conflicts = FALSE, quietly=TRUE)
txdb=TxDb.Athaliana.BioMart.plantsmart51
```

```r
## high quality PAs (HQ) -- for getting real A-rich PAs
hqPAfiles=system.file("demo_data_ath", "DRS_Ath.bed", package = "DeepIP")

## other potential PAs (LQ)
## -- together with HQ PAs and TXDB for getting free-PA regions for IP data.
allPAfiles=system.file("demo_data_ath", "PlantAPAdb_Ath.bed",
                       package = "DeepIP")

## check chr names in bsgenome
## it is better to make chr names in PA files, TXDB, and bsgenome consistent
## although DeepIP will check and make the chr consistency.
seqnames(bsgenome)
#> [1] "Chr1" "Chr2" "Chr3" "Chr4" "Chr5" "ChrM" "ChrC"

## used chrs
chrs=paste0('Chr', c(1:5))

## known PA signals for hierarchical screening sequences
## For Arabidopsis, only AATAAA is the most dominant polyA signal,
## accouting for ~10% PAs.
## If the polyA signal is unknown for the species, simply set grams=NULL.
grams='AATAAA'
gramsPriority=NULL

## for animal species, we can use:
# grams=c('AATAAA','ATTAAA','TATAAA','AGTAAA','AATACA','CATAAA','AATATA',
#         'GATAAA','AATGAA','AATAAT','AAGAAA','ACTAAA','AATAGA','ATTACA',
#         'AACAAA','ATTATA','AACAAG','AATAAG')
# gramsPriority=c(1, 2, rep(3, length(gramsMM)-2))


## output directory
## all files will be generated in this folder
outputDir='D:/DeepIP_data_ath/'

if (!file.exists(outputDir)) dir.create(outputDir)
```

3 Step by step data preparation

3.1 Step 1. get real A-rich PA sequences

`getArichPAseqsIn3UTR` will get **real** A-rich PA sequences (200nt, PA at 101st).

We use HQ PA file, not using the LQ PA file, to get **high-confidence A-rich real PA**.

If `grams` is provided, then the polyA siganl upstream -10 to -50 region of PAs will be scanned for each gram, and individual files with a respective polyA signal will be output.

The output files are like:

- realPA_Arich_seq.AATAAA.fa – this is the sequence file for A-rich **real** PAs with AATAAA signal.
- realPA_Arich_seq.NOPAS.fa – same as above but without any polyA signal.
- realPA_Arich_seq_2grams_base_profile.pdf – this is the single nucleotide plot for the two fa files.

```
getArichPAseqsIn3UTR(paBedFiles=hqPAfiles, txdb=txdb, extUTRLen=5000,
                     bsgenome=bsgenome, chrs=chrs, grams=grams,
                     gramsPriority=gramsPriority,
                     outputSeqPre=paste0(outputDir,
                                   "realPA_Arich_seq/realPA_Arich_seq"))
#> ###### getTxdbUTRs...
#> Add [Chr] to chr names
#> utrsPA: 41099
#> utrsExt (reduced): 12791
#> ###### readBedFiles2GR...
#> Read BED file C:/Users/LENOVO/AppData/Local/R/win-library/4.4/DeepIP/demo_data_ath/DRS_Ath.bed: 4991
#> ###### subset PAs in granges...
#> 49784 PACs
#> ###### get Arich PAs by removePACdsIP...
#> 19736 IP PACs; 30048 real PACs
#> ###### search upstream PA signals...
#>
#>  NOPAS AATAAA
#>  17961   1775
#> ###### Output 200nt seq (PA is 101nt) for each gram (or all to NOPAS if gram=NULL) ...
#> 17961 >>> D:/DeepIP_data_ath/realPA_Arich_seq/realPA_Arich_seq.NOPAS.fa
#> 1775 >>> D:/DeepIP_data_ath/realPA_Arich_seq/realPA_Arich_seq.AATAAA.fa
#> >>> D:/DeepIP_data_ath/realPA_Arich_seq/realPA_Arich_seq_2grams_base_profile.pdf
#> [1] "D:/DeepIP_data_ath/realPA_Arich_seq/realPA_Arich_seq.NOPAS.fa"
#> [2] "D:/DeepIP_data_ath/realPA_Arich_seq/realPA_Arich_seq.AATAAA.fa"
```

3.2 Step 2. get PA-free 3'UTR regions

First, the 3'UTR regions without any HQ, LQ PAs, or TXDB 3' UTR ends can be obtained.

This function will also check the base composition of each PA source. All temporary files and `utrsNoPA.RDS` will be output to the `outputDir`.

Temporary files include the following:

- .freq (a file containings base frequency around PAs)

- .fa (a fa file containing randomly selected 5000 sequences)
- .pdf (plot of the single nucleotide profile)

Note:

For species with small genome (e.g., yeast), if there are too few regions in the output `utrsNoPA.RDS`, we then may only get very few IPs in Step 3.
To generate more PA-free regions for IP searching, we can set larger `extUTRLen` or use less `paBedFiles`. Larger `extUTRLen` can generate more 3'UTR regions for searching IP (A-rich sequences). Fewer PAs in `paBedFiles` can avoid removing too many 3'UTR regions with potential (not very confident) PAs. For example, we can set `allPAfiles=NULL` to not considering non-HQ PAs.

```
## The output file utrsNoPA.RDS contains regions without any potential PAs
utrsNoPA=getFreePARangesIn3UTR(paBedFiles=c(hqPAfiles, allPAfiles),
                         txdb=txdb, extUTRLen=5000, extPA=200,
                         outputRds='utrsNoPA.RDS', chrs=chrs,
                         plotFA=TRUE, N=5000, bsgenome=bsgenome,
                         outputDir=outputDir)
#> ###### readBedFiles2GR...
#> Read BED file C:/Users/LENOVO/AppData/Local/R/win-library/4.4/DeepIP/demo_data_ath/DRS_Ath.bed: 4991
#> 5000 PACs
#> 5000 >>> D:/DeepIP_data_ath/DRS_Ath.fa
#> Ncol=5, use the 1/2/3/5 columns in C:/Users/LENOVO/AppData/Local/R/win-library/4.4/DeepIP/demo_data_
#> Add [Chr] to chr names
#> Read BED file C:/Users/LENOVO/AppData/Local/R/win-library/4.4/DeepIP/demo_data_ath/PlantAPAdb_Ath.be
#> 5000 PACs
#> 5000 >>> D:/DeepIP_data_ath/PlantAPAdb_Ath.fa
#> Read BED file total: 140975 PAs
#> >>> D:/DeepIP_data_ath/DRS_Ath.freq
#> >>> D:/DeepIP_data_ath/DRS_Ath.pdf
#> >>> D:/DeepIP_data_ath/PlantAPAdb_Ath.freq
#> >>> D:/DeepIP_data_ath/PlantAPAdb_Ath.pdf
#> ###### expandGRPA...
#> ###### getTxdbUTRs...
#> Add [Chr] to chr names
#> 5000 PACs
#> 5000 >>> D:/DeepIP_data_ath/TXDB_3UTR.fa
#> >>> D:/DeepIP_data_ath/TXDB_3UTR.TXDB_3UTR_base_profile.freq
#> >>> D:/DeepIP_data_ath/TXDB_3UTR.TXDB_3UTR_base_profile.pdf
#> utrsPA: 41099
#> utrsExt (reduced): 12791
#> ###### reduce PAs+UTRPAs...
#> ###### disjoin PAs+UTRPAs+UTRExts...
#> ###### output final PA-free UTRs >500nt...
#> utrsNoPA: 31041
```

3.3 Step 3. get internal priming (IP) A-rich PA sequences

`getArichIPseqs` will get A-rich IP sequences, providing `utrsNoPA.RDS` obtained from `getFreePARangesIn3UTR`. This function scans As and polyA siganls (if `grams` is provided) in PA-free 3' UTR regions to get IP sequences as negative samples.

The output files are like:

- IP_Arich_seq.AATAAA.fa – this is the sequence file for A-rich **IP** PAs with AATAAA signal.
- IP_Arich_seq.NOPAS.fa – same as above but without any polyA signal.
- IP_Arich_seq_2grams_base_profile.pdf – this is the single nucleotide plot for the two fa files.

```
## It may take about 0.5H for 90,000 sequences.
## Here for demonstration, we set `N=3000` to speed up,
## which randomly selects 3000 A-rich sequences for processing.
## In practive, we can set `N=NULL` to process all sequences.
getArichIPseqs(regionsNoPARDS=paste0(outputDir,'utrsNoPA.RDS'),
               bsgenome=bsgenome,
               grams=grams,
               N=3000,
               outputSeqPre=paste0(outputDir, "IP_Arich_seq/IP_Arich_seq"))
#> ###### load regionsNoPARDS: 31041
#> 31041 PACs
#> ###### get regionsNoPA seqs...
#> ###### scan PAS (grams)...
#> ###### scan An...
#> Output N=3000 from total 31036 seqs
#> 500 of 3000 done ...
#> 1000 of 3000 done ...
#> 1500 of 3000 done ...
#> 2000 of 3000 done ...
#> 2500 of 3000 done ...
#> 3000 of 3000 done ...
#> NOPAS nseq= 1705
#> AATAAA nseq= 514
#> >>> D:/DeepIP_data_ath/IP_Arich_seq/IP_Arich_seq_2grams_base_profile.pdf
```

3.4 Step 4. get separate train/test sequences

Above steps will generate real and IP sequence fa files with distince polyA siganls in `realPA_Arich_seq` and `IP_Arich_seq` folders.

Then we will randomly split each fa file (e.g., IP_Arich_seq.AATAAA.fa), 70% for training and 30% for testing.

```
## count the number of real and IP sequences
fas=statCntFas(path=paste0(outputDir, 'realPA_Arich_seq/'),
               filePre='realPA_Arich_seq.')
#> [1] "realPA_Arich_seq."
#>  NOPAS AATAAA
#>  17961   1775
#> Total 19736

fas=statCntFas(path=paste0(outputDir, 'IP_Arich_seq'),
               filePre='IP_Arich_seq.')
#> [1] "IP_Arich_seq."
#>  NOPAS AATAAA
#>   1705    514
#> Total 2219
```

```r
dir.create(paste0(outputDir,'modelDataSplits_ath'))

## true train and test files
split2TrainTest(path=paste0(outputDir,'realPA_Arich_seq/'),
                filePre='realPA_Arich_seq.',
                dir1=paste0(outputDir,
                            'modelDataSplits_ath/realPA_Arich_seq_train_per70'),
                dir2=paste0(outputDir,
                            'modelDataSplits_ath/realPA_Arich_seq_test_per30'),
                per1=0.7,
                label=":1")
#> split2TrainTest: the new sequence title will like <old title>;AATAAA:1
#> AATAAA dir1=1242 (0.700000); dir2=533; total=1775
#> NOPAS dir1=12572 (0.700000); dir2=5389; total=17961

## false train and test files
split2TrainTest(path=paste0(outputDir,'IP_Arich_seq'),
                filePre='IP_Arich_seq.',
                dir1=paste0(outputDir,
                            'modelDataSplits_ath/IP_Arich_seq_train_per70'),
                dir2=paste0(outputDir,
                            'modelDataSplits_ath/IP_Arich_seq_test_per30'),
                per1=0.7,
                label=":0")
#> split2TrainTest: the new sequence title will like <old title>;AATAAA:0
#> AATAAA dir1=359 (0.700000); dir2=155; total=514
#> NOPAS dir1=1193 (0.700000); dir2=512; total=1705
```

Above steps will generate real and IP sequence fa files for training and testing.

The following directories are generated:

- IP_Arich_seq_test_per30 – This folder contains 30% **IP** sequences for subsequent **test** data generation.
- IP_Arich_seq_train_per70 – This folder contains 70% **IP** sequences for subsequent **training** data generation.
- realPA_Arich_seq_test_per30 – This folder contains 30% **real PA** sequences for subsequent **test** data generation.
- realPA_Arich_seq_train_per70 – This folder contains 70% **real PA** sequences for subsequent **training** data generation.

Each folder includes fa files with different polyA signals like:

- AATAAA.fa
- NOPAS.fa

3.5 Step 5. get combined training and test sequences

3.5.1 Get training true and false sequences

```
## get the polyA signal distributions for the subsequent random sampling
REALPA_PERC=statCntFas(path=paste0(outputDir, 'realPA_Arich_seq/'),
                       filePre='realPA_Arich_seq.')
#> [1] "realPA_Arich_seq."
#>  NOPAS AATAAA
#>  17961   1775
#> Total 19736

seqDir=paste0(outputDir,'modelDataSplits_ath/')

## randomly sample 10,000 sequences as real (positive) training data
files=getNseqs(seqDir=paste0(seqDir, 'realPA_Arich_seq_train_per70'),
               N=10000,
               nsplits=1,
               outputPre=paste0(seqDir,
                                "realPA_Arich_seq_train_per70_10000s_1splits"),
               perc=REALPA_PERC)
#> ###### Read PA fa
#> Split 1: Get 1 seqs from NOPAS to make N=10000
#> >>> D:/DeepIP_data_ath/modelDataSplits_ath :
#> [1] "realPA_Arich_seq_train_per70_10000s_1splits.split.1.fa"


## plot the real training fa file for validation
movAPA::plotATCGforFAfile(faFiles=files, ofreq=FALSE,
                          opdf=T, refPos=101)
#> >>> D:/DeepIP_data_ath/modelDataSplits_ath/realPA_Arich_seq_train_per70_10000s_1splits.split.1.pdf

## randomly sample 10,000 sequences as false (IP, negative) training data
## Here for demonstration, we set `N=1000`.
files=getNseqs(seqDir=paste0(seqDir, 'IP_Arich_seq_train_per70'),
               N=1000,
               nsplits=1,
               outputPre=paste0(seqDir,
                                'IP_Arich_seq_train_per70_10000s_1splits'),
               perc=REALPA_PERC)
#> ###### Read PA fa
#> Split 1: Get 1 seqs from NOPAS to make N=1000
#> >>> D:/DeepIP_data_ath/modelDataSplits_ath :
#> [1] "IP_Arich_seq_train_per70_10000s_1splits.split.1.fa"

## plot the IP training fa file for validation
movAPA::plotATCGforFAfile(faFiles=files, ofreq=FALSE,
                          opdf=T, refPos=101)
#> >>> D:/DeepIP_data_ath/modelDataSplits_ath/IP_Arich_seq_train_per70_10000s_1splits.split.1.pdf
```

3.5.2 Get test true and false sequences

```
## randomly sample 5,000 sequences as real (positive) test data
files=getNseqs(seqDir=paste0(seqDir, 'realPA_Arich_seq_test_per30'),
               N=5000,
               nsplits=1,
               outputPre=paste0(seqDir,
                                'realPA_Arich_seq_test_per30_5000s_1splits'),
               perc=REALPA_PERC)
#> ###### Read PA fa
#> Split 1: Get 1 seqs from NOPAS to make N=5000
#> >>> D:/DeepIP_data_ath/modelDataSplits_ath :
#> [1] "realPA_Arich_seq_test_per30_5000s_1splits.split.1.fa"


movAPA::plotATCGforFAfile(faFiles=files, ofreq=FALSE, opdf=T, refPos=101)
#> >>> D:/DeepIP_data_ath/modelDataSplits_ath/realPA_Arich_seq_test_per30_5000s_1splits.split.1.pdf

## randomly sample 5,000 sequences as IP (negative) test data
## Here for demonstration, we set `N=500`.
files=getNseqs(seqDir=paste0(seqDir, 'IP_Arich_seq_test_per30'),
               N=500,
               nsplits=1,
               outputPre=paste0(seqDir,
                                'IP_Arich_seq_test_per30_5000s_1splits'),
               perc=REALPA_PERC)
#> ###### Read PA fa
#> Split 1: Get 1 seqs from NOPAS to make N=500
#> >>> D:/DeepIP_data_ath/modelDataSplits_ath :
#> [1] "IP_Arich_seq_test_per30_5000s_1splits.split.1.fa"

movAPA::plotATCGforFAfile(faFiles=files, ofreq=FALSE, opdf=T, refPos=101)
#> >>> D:/DeepIP_data_ath/modelDataSplits_ath/IP_Arich_seq_test_per30_5000s_1splits.split.1.pdf
```

Above steps will generate the following fa files:

- IP_Arich_seq_train_per70_10000s_1splits – This file combined **IP training** sequence files with different polyA signals.
- realPA_Arich_seq_train_per70_10000s_1splits – This file combined **real training** sequence files with different polyA signals.
- realPA_Arich_seq_test_per30_5000s_1splits – This file combined **real test** sequence files with different polyA signals.
- IP_Arich_seq_test_per30_5000s_1splits – This file combined **IP test** sequence files with different polyA signals.

These files are all following the same polyA signal distribution. For example, all files contain the same proportion of sequences with polyA signal=AATAAA.

3.5.3 Finally, get one training and one test file

Finally, we can get one fa file for training and one fa file for test, these two files can be used for DeepIP training and test.

Each file contains the same numbers of true (real) and false (IP) sequences, with real sequences labeled with :1 and IP sequences labeled with :0 in the sequence title.

The sequence title also include the polyA signal and polyA site information.

```
combineFaFiles_fast(paste0(seqDir,
                    c('realPA_Arich_seq_train_per70_10000s_1splits.split.1.fa',
                      'IP_Arich_seq_train_per70_10000s_1splits.split.1.fa')),
                    ofile = paste0(seqDir, 'train.10000T.10000F.fa'),
                    verbose = TRUE)
#> D:/DeepIP_data_ath/modelDataSplits_ath/realPA_Arich_seq_train_per70_10000s_1splits.split.1.fa     10
#> D:/DeepIP_data_ath/modelDataSplits_ath/IP_Arich_seq_train_per70_10000s_1splits.split.1.fa     1000
#> Total     11000
#> [1] 11000


combineFaFiles_fast(paste0(seqDir,
                    c('realPA_Arich_seq_test_per30_5000s_1splits.split.1.fa',
                      'IP_Arich_seq_test_per30_5000s_1splits.split.1.fa')),
                    ofile = paste0(seqDir, 'test.5000T.5000F.fa'),
                    verbose = TRUE)
#> D:/DeepIP_data_ath/modelDataSplits_ath/realPA_Arich_seq_test_per30_5000s_1splits.split.1.fa   5000
#> D:/DeepIP_data_ath/modelDataSplits_ath/IP_Arich_seq_test_per30_5000s_1splits.split.1.fa   500
#> Total     5500
#> [1] 5500
```

4 Train DeepIP model

4.1 Create conda env

Please first install the anaconda (https://www.anaconda.com/).

```
# create the conda env in the command window
conda create -n DeepIP python=3.7

# test the env
conda env list
conda activate DeepIP

# install the following modules
pip install Keras
pip install tensorflow
pip install pandas
pip install sklearn

# or use conda install, like
conda install -c anaconda protobuf
```

4.2 Train DeepIP model

Using the above train file, we can train a DeepIP model.

```r
setwd(seqDir)
trainDeepIP(condaEnv="A_CONDA_ENV",
            inTrainSeq='train.10000T.10000F.fa',
            outTrainedModel='train.10000T.10000F.epoch100.hdf5',
            epoch=100)
```

5 Test DeepIP model

We can also use above test file to test the performance of the trained DeepIP model.

```r
testDeepIP(condaEnv="A_CONDA_ENV",
           inTestSeq='test.5000T.5000F.fa',
           inTrainedModel='train.10000T.10000F.epoch100.hdf5',
           outTestCsv='train.10000T.10000F.epoch100_ON_test.5000T.5000F.csv',
           seqLabel='')

# calculate ROC/AUC/F1/... metrics
statDeepRes('train.10000T.10000F.epoch100_ON_test.5000T.5000F.csv',
            ofile='train.10000T.10000F.epoch100_ON_test.5000T.5000F.stat.csv')

# Plot single nucleotide profiles for TP/FP/TN/FN sequences
plotFaDeepRes('train.10000T.10000F.epoch100_ON_test.5000T.5000F.csv',
              fafile='test.5000T.5000F.fa')
```