
PEA Manual

An integrated R toolkit for plant epitranscriptome analysis

Version 1.0

April 12th, 2018

Authors: Jingjing Zhai, Jie Song, Qian Cheng, Yunjia Tang, Chuang Ma

Contact:

Jingjing Zhai: zhaijingjing603@gmail.com

Dr. Chuang Ma: chuangma2006@gmail.com

Table of Contents

Brief introduction	1
1. PEA installation	1
1.1 Docker installation and start.....	1
1.2 Verify if Docker is installed correctly	2
1.3 PEA installation from Docker Hub	3
1.4 Quickly start	3
2. CMR calling.....	3
2.1 Reads mapping using tophat	4
2.2 Peak calling methods implemented in PEA	4
2.2.1 Peak calling using SlidingWindow	5
2.2.2 Peak calling using exomePeak.....	6
2.2.3 Peak calling using MetPeak.....	6
2.2.4 Peak calling using MACS2.....	6
2.2.5 Peak calling using BayesPeak	6
3. CMR prediction	7
3.1 Samples organization	7
3.2 Samples vectorization with three feature encoding schemes	7
3.3 Construction of a CMR predictor using random forest (RF) and PSOL algorithm	8
3.4 CMR prediction using predictor generated by PSOL	9
4. CMR annotation	10
4.1 CMR location distribution.....	10
4.2 Motif scanning and discovery	11
4.3 Functional enrichment analysis of CMR corresponded genes	11
5. Source codes availability	11
6. How to access help	11
7. References.....	12

Brief introduction

PEA is an integrated R toolkit that aims to facilitate the plant epitranscriptome analysis. This toolkit contains a comprehensive collection of functions required for read mapping, CMR calling, motif scanning and discovery, and gene functional enrichment analysis. PEA also takes advantage of machine learning technologies for transcriptome-scale CMR prediction, with high prediction accuracy, using the Positive Samples Only Learning algorithm, which addresses the two-class classification problem by using only positive samples (CMRs), in the absence of negative samples (non-CMRs). Hence PEA is a versatile epitranscriptome analysis pipeline covering CMR calling, prediction, and annotation. The R source codes of PEA and related high-throughput sequencing analysis tools have been packaged into a Docker image for convenient usage.

1. PEA installation

1.1 Docker installation and start

- For Windows (Test on Windows 10 Enterprise version):
 - a) Download the installer from following URL:
<https://download.docker.com/win/stable/Docker%20for%20Windows%20Installer.exe>;
 - b) Double click the EXE file to open it;
 - c) Follow the wizard instruction and complete installation;
 - d) Search docker, select **Docker for Windows** in the search results and click it.
- On Mac OS X (Test on macOS Sierra version 10.12.6 and macOS High Sierra version 10.13.3):
 - a) Download the installer from following URL:
<https://download.docker.com/mac/stable/Docker.dmg>;
 - b) Double click the DMG file to open it;
 - c) Drag the docker into Applications and complete installation;
 - d) Start docker from Launchpad by click it.

- On Ubuntu (Test on Ubuntu 14.04 LTS and Ubuntu 16.04 LTS):
 - a) Go to <https://download.docker.com/linux/ubuntu/dists/>, choose your Ubuntu version, browse to [pool/stable](#) and choose [amd64](#), [armhf](#), [ppc64el](#) or [s390x](#). Download the DEB file for the Docker version you want to install;
 - b) Install Docker, supposing that the DEB file is download into following path:
/home/docker-ce_<version-XXX>~ubuntu_amd64.deb

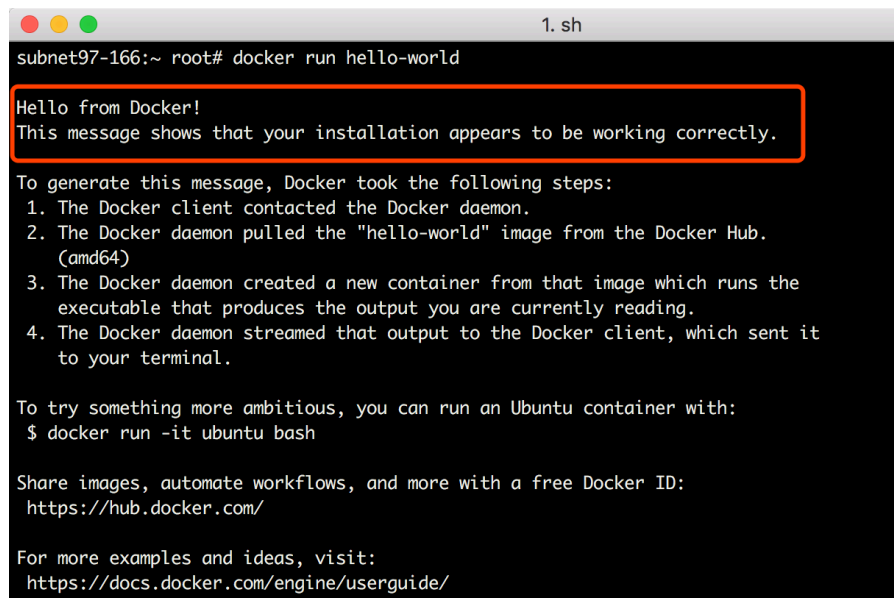
```
1. sudo dpkg -i /home/docker-ce-<version-XXX>~ubuntu_amd64.deb
2. sudo apt-get install -f
```

1.2 Verify if Docker is installed correctly

Once Docker installation is completed, we can run **hello-world** image to verify if Docker is installed correctly. Open terminal in Mac OS X and Linux operating system or CMD for Windows operating system, then type the following command:

```
$ docker run hello-world
```

If docker is installed successfully, you can see the following messages.



```
1. sh
subnet97-166:~ root# docker run hello-world
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/engine/userguide/
```

- **Notes**

- 1) Root permission is required for Linux operating system.
- 2) Considering that differences between different computers may exist, please refer to official Docker installation manual (<https://docs.docker.com/install>) if the

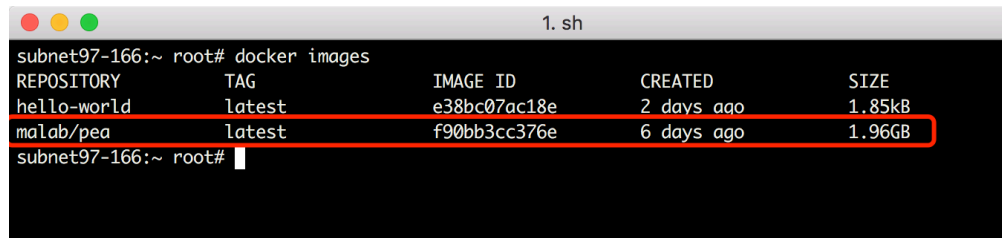
instructions above don't work.

1.3 PEA installation from Docker Hub

For Mac OS X and Linux operating systems, open the terminal, for Windows operating system, open CMD. Typing the following command:

```
1. # Pulling PEA from Docker PEA
2. $ docker pull malab/pea
```

If PEA is installed correctly, type “docker images”, you will see the following messages:



```
subnet97-166:~ root# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hello-world	latest	e38bc07ac18e	2 days ago	1.85kB
malab/pea	latest	f90bb3cc376e	6 days ago	1.96GB

```
subnet97-166:~ root#
```

1.4 Quickly start

Once PEA is installed successfully, type the following command to start PEA:

```
1. $ docker run -it -v /host directory of dataset:/home/data malab/pea R
2. # Supposing that users' private dataset is located in the directory "/home/test", then change the red colored words above (/host directory of dataset) to host directory (/home/test)
3. library(PEA)
4. setwd("/home/data/")
```

Note: the directory ("/home/data/") is a virtual directory in PEA Docker image. In order to use private dataset more easily, the parameter “-v” is strongly recommended to mount host directory of dataset to PEA image.

2. CMR calling

2.1 Reads mapping using tophat

```

1. #Loading sample data for reads mapping
2. fq <- system.file("extdata/test.fq", package = "PEA")
3. referenceGenome <- system.file("extdata/chromosome1.fa", package = "PEA")
4. #Reads mapping using tophat with default parameter, the alignment results
   will be saved to the working directory (/host directory of dataset)
5. test.bam <- readMapping(alignment = "tophat", fq = fq,
6.
                           refGenome = referenceGenome, paired = F)
7. #Reads mapping using tophat with 2 threads
8. test.bam <- readMapping(alignment = "tophat", fq = fq,
9.
                           refGenome = referenceGenome, paired = F,
10.
                           ... = "-p 2")

```

Note: other alignment toolkits such as Bowtie, Bowtie 2, TopHat 2, Hisat and Hisat 2 can be easily invoked by specifying “alignment” parameter in “readMapping” function.

2.2 Peak calling methods implemented in PEA

PEA integrated five peak calling methods including “SlidingWindow”, “exomePeak”, “MetPeak”, “MACS2” and “BayesPeak”. each peak calling method can be easily invoked by specifying parameter “method” in “CMRCalling” function.

How to choose:

- a) “SlidingWindow” is used to examine the significant enriched regions (peaks) using Fisher’s exact test-based sliding window method between RIP and input samples.
- b) “exomePeak” is an exome-based peak calling method designed specifically for analysis of MeRIP-seq (m⁶A-seq).

- c) “MetPeak” is a peak calling method used for transcriptome-wide m⁶A detection designed for MeRIP-seq data which is developed based on graphical model.
- d) “MACS2” is firstly designed to identify transcription binding sites in ChIP-seq dataset and can also be used for peak calling in MeRIP-seq because of the similar principle.
- e) “BayesPeak” is specially designed to detect genomic enriched regions in ChIP-seq dataset using Bayesian hidden Markov model and Markov chain Monte Carlo algorithms.

Summary: Both “SlidingWindow” and “exomePeak” are sliding window-based methods to detect significant enriched regions and perform well on CMR prediction, while “MACS2” and “BayesPeak” are peak calling methods designed for ChIP-seq dataset, but “MACS2” can strictly control false positives, in contrast, BayePeak is weaker in controlling false positives. In addition, methods “exomePeak”, “MeTPeak” and “MACS2” can accept biological replicates using statistical methods.

2.2.1 Peak calling using SlidingWindow

```

1. #Loading sample data for peak calling
2. input.bam <- system.file("extdata/chr1_input_test.bam", package = "PEA")
3. RIP.bam <- system.file("extdata/chr1_RIP_test.bam", package = "PEA")
4. refGenome <- system.file("extdata/chromosome1.fa", package = "PEA")
5. GTF <- system.file("extdata/chromosome1.gtf", package = "PEA")
6. #Peak calling using sliding window-based method
7. cmrMat <- CMRCalling(CMR = "m6A", IPBAM = RIP.bam, inputBAM = input.bam,
8.                      method = "SlidingWindow", mappedInput = 17472,
9.                      mappedRIP = 20072, refGenome = refGenome)
10. #Saving results into working directory
11. write.table(cmrMat, file = "SlidingWindow_peaks.txt", sep = "\t",
12.             quote = F, row.names = F, col.names = F)
13. #Note: parameters "mappedInput" and "mappedRIP" represent the number of reads
    aligned to reference genome in input and RIP samples, respectively

```

2.2.2 Peak calling using exomePeak

```

1. #m6A peak calling using exomePeak
2. cmrMat <- CMRCalling(CMR = "m6A", method = "exomePeak", IPBAM = RIP.bam,
3.                     inputBAM = input.bam, GTF = GTF)
4. write.table(cmrMat, file = "exomePeak_peaks.txt", sep = "\t",
5.             quote = F, row.names = F, col.names = F)

```

2.2.3 Peak calling using MetPeak

```

1. #m6A peak calling using MetPeak
2. cmrMat <- CMRCalling(CMR = "m6A", method = "MetPeak", IPBAM = RIP.bam,
3.                     inputBAM = input.bam, GTF = GTF)
4. write.table(cmrMat, file = "MetPeak_peaks.txt", sep = "\t",
5.             quote = F, row.names = F, col.names = F)

```

2.2.4 Peak calling using MACS2

```

1. #m6A peak calling using MACS2
2. cmrMat <- CMRCalling(CMR = "m6A", method = "MACS2", IPBAM = RIP.bam,
3.                     inputBAM = input.bam, GTF = GTF, ...="--nomodel")
4. #Note: futher parameters recognized by MACS2 can be specified in "...
5. write.table(cmrMat, file = "MACS2_peaks.txt", sep = "\t",
6.             quote = F, row.names = F, col.names = F)

```

2.2.5 Peak calling using BayesPeak

```

1. #m6A peak calling using BayesPeak
2. cmrMat <- CMRCalling(CMR = "m6A", method = "BayesPeak", IPBAM = RIP.bam,
3.                     inputBAM = input.bam, GTF = GTF)
4. write.table(cmrMat, file = "BayesPeak_peaks.txt", sep = "\t",
5.             quote = F, row.names = F, col.names = F)

```


3. CMR prediction

3.1 Sample organization

Before performing CMR prediction, positive and negative samples are required to be organized. PEA selected m⁶A modifications as positives if peaks contain only one RRA*CH (R denotes A or G, A* denotes methylated A, and H denotes A, C or U) signal, and the remaining RRACH signals are considered as unlabeled samples.

```

1. #Converting genomic position to cDNA position
2. GTF <- system.file("extdata/chromosome1.gtf", package = "PEA")
3. peaks <- G2T(bedPos = cmrMat, GTF = GTF)
4. #Searching RRACH motif from cDNA sequence
5. cDNA <- system.file("extdata/chr1_cdna.fa", package = "PEA")
6. motifPos <- searchMotifPos(sequence = cDNA)
7. #Finding confident positive samples and unlabeled samples
8. posSamples <- findConfidentPosSamples(peaks = peaks, motifPos = motifPos)
9. unlabelSamples <- findUnlabelSamples(cDNAID = posSamples$cDNAID, motifPos
   = motifPos, posSamples = posSamples$positives)

```

3.2 Samples vectorization with three feature encoding schemes

In order to be recognized by the ML-based classification models, PEA can transform each sample (L -nt flanking sequence centered on m⁶A or non-m⁶A modifications) into a $(L*4+20+22)$ -dimensional vector which including $L*4$ binary-based features, 20 k -mer based features ($k = 1$ and $k = 2$) and 22 PseDNC-based features.

```

1. #Extracting flanking sequence of 101-nt centered on m6A or non-m6A
2. positives <- posSamples$positives
3. posSeq <- extractSeqs(RNAseq = cDNA, samples = positives,
4.                       seqLen = 101)
5. unlabelSeq <- extractSeqs(RNAseq = cDNA, samples = unlabelSamples,

```

```

6.                               seqLen = 101)
7. #Feature encoding using binary, k-mer and PseDNC encoding schemes
8. posFeatureMat <- featureEncoding(RNAseq = posSeq)
9. unlabelFeatureMat <- featureEncoding(RNAseq = unlabelSeq)
10. featureMat <- rbind(posFeatureMat, unlabelFeatureMat)

```

3.3 Construction of a CMR predictor using random forest (RF) and PSOL algorithm

The RF is a powerful ML algorithm that has been widely used for the binary classification problems in bioinformatics and computational biology (Cui, et al., 2015; Ma, et al., 2014; Touw, et al., 2013). To build a RF-based classifier (predictor), positive samples and negative samples are required to be specified by the user. Existing m⁶A prediction approach typically use the known m⁶A modifications as the positive sample set and the unlabeled samples as the negative sample set to build predictors to identify new m⁶A modifications from unlabeled samples (Chen, et al., 2016; Zhou, et al., 2016). Such kind of m⁶A predictors is actually built from a noisy negative sample set. As a result, the predictors do not perform well as they could in identifying new m⁶A modifications. To address this challenge, PSOL algorithm was introduced into the PEA package, which can be used to build an ML-based binary classification system with high prediction accuracy with only positive samples and without pre-specified negative samples. This algorithm has been previously applied to predict abiotic stress-response genes and genomic loci encoding functional noncoding RNAs (Ma, et al., 2014; Wang, et al., 2006).

To implement the PSOL algorithm, an initial negative sample set with the same size as the positive sample set was firstly constructed, which contained samples that were selected from the unlabeled sample set based on the maximal Euclidean distances to positive samples. The negative sample set was then expanded iteratively using the RF-based classifier until the designated iteration number was reached. In each iteration, a ten-fold cross-validation method and the receiver operating characteristic (ROC) curve analysis were performed to evaluate the predictive performance of current classifier in distinguishing between positive samples and negative samples. In order to get the optimal RF classifier, the RF classifier with max AUC value was selected from the ten-fold

cross-validation and was subsequently used for scoring the unlabeled samples with a user-adjustable threshold (TPR = 0.995) to ensure a large fraction of positive samples can be correctly identified.

In each iteration, an m⁶A predictor was built using the RF algorithm with positive samples and selected negative samples. PSOL algorithm can be easily implemented using the function “PSOL” in the PEA package.

```

1. #Creating a directory to save psol results
2. dir.create("./psol")
3. psolResDic <- "./psol/"
4. #Starting running PSOL
5. psolRes <- PSOL(featureMatrix = featureMat, positives = positives,
6.                 unlabels = unlabelSamples, PSOLResDic = psolResDic,
7.                 cpus = 1)
8. #Extracting negative samples generated by PSOL
9. negatives <- psolRes$finalNegatives
10. #Performing 5-fold cross-validation experiment to evaluate the performance
    of predictor
11. cvRes <- cross_validation(featureMat = featureMat, positives = positives,
12.                           negatives = negatives, cross = 5)
13. #Plotting ROC curves for 5-fold cross-validation
14. pdf("cross_validation.pdf", height = 5, width = 5)
15. plotROC(cvRes = cvRes)
16. dev.off()

```

3.4 CMR prediction using predictor generated by PSOL

After performing PSOL, an m⁶A predictor would be generated in the “psolRes” object, users can easily predict novel candidate CMRs by following command.

```

1. #Predicting novel candidate m6A modifications
2. predSeq <- system.file("extdata/test_pred.fa", package = "PEA")
3. predMat <- predCMR(predSeq, model = psolRes$model)

```

4. CMR annotation

PEA also provide CMR annotation to provide insights into spatial and functional associations of CMRs through function “CMRAnnotation”. Using this function, the manner of distribution of CMRs in the transcriptome is statistically analyzed, including the spatial distribution of CMRs, and the regions of enrichment of CMRs within transcripts. In addition, motif scanning and *de novo* motif discovery are also provided to investigate the potential regulatory mechanisms leading by CMRs. Moreover, gene functional (Gene Ontology) enrichment analysis is also performed to characterize the enriched functions of CMR-corresponding transcripts using R package “topGO”.

4.1 CMR location distribution

```

1. #Loading sample data
2. GTF <- system.file("extdata/chromosome1.gtf", package = "PEA")
3. input.bam <- system.file("extdata/chr1_input_test.bam", package = "PEA")
4. RIP.bam <- system.file("extdata/chr1_RIP_test.bam", package = "PEA")
5. refGenome <- system.file("extdata/chromosome1.fa", package = "PEA")
6. cDNA <- system.file("extdata/chr1_cdna.fa", package = "PEA")
7. #Extracting the UTR position from GTF file
8. UTRMat <- getUTR(GTF = GTF)
9. cmrMat <- CMRCalling(CMR = "m6A", IPBAM = RIP.bam, inputBAM = input.bam,
10.                      method = "SlidingWindow", mappedInput = 17472,
11.                      mappedRIP = 20072, refGenome = refGenome)
12. #Performing CMR location distribution analysis
13. pdf("CMR_location.pdf", height = 10, width = 10)
14. results <- CMRAnnotation(cmrMat = cmrMat, SNR = F, UTRMat = UTRMat,
15.                           genomic = T, annotation = "location", GTF = GTF,
16.                           RNAseq = cDNA)
17. dev.off()
```

4.2 Motif scanning and discovery

```
1. #Searching motif
2. testSeq <- system.file("extdata/test.fa", package = "PEA")
3. pdf("motifScan.pdf", height = 5, width = 5)
4. results.scan <- CMRAnnotation(cmrSeq = testSeq,
5.                               annotation = "motifScan")
6. dev.off()
7. #De-novo motif detection
8. pdf("motifDetect.pdf", height = 5, width = 5)
9. results.detect <- CMRAnnotation(cmrSeq = testSeq,
10.                                annotation = "motifDetect")
11. dev.off()
```

4.3 Functional enrichment analysis of CMR corresponded genes

```
1. #GO enrichment analysis analysis
2. library(topGO)
3. peaks <- G2T(bedPos = cmrMat, GTF = GTF)
4. pdf("GOenrichment.pdf", height = 5, width = 5)
5. enrichment <- CMRAnnotation(cmrMat = peaks, GTF = GTF, annotation = "GO",
6.                              topNodes = 20, dataset = "athaliana_eg_gene")
7. dev.off()
```

5. Source codes availability

The source codes and user manual of PEA are freely available to academic users at <https://github.com/cma2015/PEA>.

6. How to access help

- If users encounter any bugs or issues, feel free to leave a message at Github issues: <https://github.com/cma2015/PEA/issues>. We will try our best to deal with all issues as soon as possible.
- For comments/suggestions/error reports are available, please contact: zhaijingjing603@gmail.com or chuangma2006@gmail.com

7. References

- Chen, W., *et al.* Identifying N⁶-methyladenosine sites in the *Arabidopsis thaliana* transcriptome. *Mol Genet Genomics* 2016;291(6):2225-2229.
- Cui, H., Zhai, J. and Ma, C. miRLocator: machine learning-based prediction of mature microRNAs within plant pre-miRNA sequences. *PLoS One* 2015;10(11):e0142753.
- Ma, C., *et al.* Machine learning-based differential network analysis: a study of stress-responsive transcriptomes in *Arabidopsis*. *Plant Cell* 2014;26(2):520-537.
- Touw, W.G., *et al.* Data mining in the life sciences with random forest: a walk in the park or lost in the jungle? *Brief Bioinform* 2013;14(3):315-326.
- Wang, C.L., *et al.* PSOL: a positive sample only learning algorithm for finding non-coding RNA genes. *Bioinformatics* 2006;22(21):2590-2596.
- Zhou, Y., *et al.* SRAMP: prediction of mammalian N⁶-methyladenosine (m⁶A) sites based on sequence-derived features. *Nucleic Acids Res* 2016;44(10):e91.