The cleanUpdTSeq user's guide

Sarah Sheppard, Nathan Lawson, Lihua Julie Zhu*

October 13, 2014

Contents

1 Introduction

3' ends of transcripts have generally been poorly annotated. With the advent of deep sequencing, many methods have been developed to identify 3' ends. The majority of these methods use an oligo-dT primer, which can bind to internal adenine-rich sequences, and lead to artifactual identification of polyadenylation sites. Heuristic filtering methods rely on a certain number of adenines in the genomic sequence downstream of a putative polyadenylation site to remove internal priming events. We introduce a package to provide a robust method to classify putative polyadenylation sites. cleanUpdTSeq uses a naïve Bayes classifier, implemented through the *e1071* [1], and sequence features surrounding the putative polyadenylation sites for classification.

The package includes a training dataset constructed from 6 different Zebrafish sequencing dataset, and functions for fetching surrounding sequences using BSgenome [2], building

*sarah.sheppard@umassmed.edu, julie.zhu@umassmed.edu

feature vectors and classifying whether the putative polyadenylations site is a true polyadenylation site or a mis-primed false site.

A paper has been submitted to Bioinformatics and currently under revision [3].

2 step-by-step guide

Here is a step-by-step guide on using cleanUpdTSeq to classify a list of putative polyadenylation sites

2.1 Step 1. Load the package cleanUpdTSeq, read in the test dataset and then use the function BED2GRangesSeq to convert it to GRanges.

```
> library(cleanUpdTSeq)
> testFile <- system.file("extdata", "test.bed", package="cleanUpdTSeq")
> testSet <- read.table(testFile, sep="\t", header=TRUE)
> peaks <- BED2GRangesSeq(testSet, withSeq=FALSE)
```

If test dataset contains sequence information already, then use the following command instead.

```
> peaks <- BED2GRangesSeq(testSet, upstream.seq.ind=7,
+                         downstream.seq.ind=8, withSeq=TRUE)
```

To work with your own test dataset, please set testFile to the file path that contains the putative sites.

Here is how the test dataset look like.

```
> head(testSet)

    chr    start     stop        name score strand
1 chr10  2965327  2965327 6hpas-22249     1      -
2 chr10  2966558  2966558 6hpas-22250     1      -
3 chr10  2974251  2974251 6hpas-22251     2      -
4 chr10  2978441  2978441 6hpas-22252     1      -
5 chr11 16772291 16772291 6hpas-33204     1      -
6 chr11 16777848 16777848 6hpas-33205     1      -
                                  upstream                     downstream
1 TCTTCATCATGGTCATCTCGCACCAGAGAGTGTGCCAGGG CAGGAAGTTTTACCTGTCTGTCATTATCGT
2 ACCCTGGTGAGGGTATAGAGCTGGTCCAGTGTGCCACGGC AAAGAGGAAAACAGCATTGTTCCTCCTGGA
3 TGATTTGTTTGTAACTGATTTTATCTTTTAATAAAAAAGA AAAAAGAAAGTCAAGCCAAGAGGCAAATAC
4 GGAGCGCGACCGCATCAACAAAATCTTGCAGGATTATCAG AAGAAAAAGATGGTGAGTTATTATCATTCA
5 AGGGAAATAAATACAAAAGAATAAAAATATGATTCATTGT AAGAAAAACACTTTAGCTACAAAAGTCCTT
6 ATTTAGTTGGGTATTATTTCAAATAAAGAGAGAGAGAGAC ACAAAAACTACATCAAATTTGAGGACAAAA
```

2.2 Step2. Build feature vectors for the classifier using the function buildFeatureVector.

The zebrafish genome from BSgenome is used in this example for obtaining surrounding sequences. For a list of other genomes available through BSgenome, please refer to the BSgenome package documentation [2].

```
> testSet.NaiveBayes <- buildFeatureVector(peaks, BSgenomeName=Drerio,
+                                          upstream=40, downstream=30,
+                                          wordSize=6, alphabet=c("ACGT"),
+                                          sampleType="unknown",
+                                          replaceNAdistance=30,
+                                          method="NaiveBayes",
+                                          ZeroBasedIndex=1, fetchSeq=TRUE)
```

If sequences are present in the test dataset already, then set fetchSeq=FALSE.

2.3 Step 3. Load the training dataset and classify putative polyadenylation sites.

```
> data(data.NaiveBayes)
> if(interactive()){
+     predictTestSet(data.NaiveBayes$Negative, data.NaiveBayes$Positive,
+                   testSet.NaiveBayes=testSet.NaiveBayes,
+                   outputFile="test-predNaiveBayes.tsv",
+                   assignmentCutoff=0.5)
+ }
```

The output file is a tab-delimited file containing the name of the putative polyadenylation sites, the probability that the putative polyadenylation site is false/oligodT internally primed, the probability the putative polyadenylation site is true, the predicted class based on the assignment cutoff and the sequence surrounding the putative polyadenylation site.

3 References

1. Meyer, D., et al., e1071: Misc Functions of the Department of Statistics (e1071), TU Wien. 2012.
2. Pages, H., BSgenome: Infrastructure for Biostrings-based genome data packages.
3. Sarah Sheppard, Nathan D. Lawson, and Lihua Julie Zhu. 2013. Accurate identification of polyadenylation sites from 3' end deep sequencing using a naïve Bayes classifier. Bioinformatics. Under revision

4 Session Info

```
> sessionInfo()

R version 3.1.1 Patched (2014-09-25 r66681)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats4    parallel  stats     graphics  grDevices utils     datasets
[8] methods   base

other attached packages:
 [1] cleanUpdTSeq_1.4.0                 e1071_1.6-4
 [3] seqinr_3.0-7                       BSgenome.Drerio.UCSC.danRer7_1.4.0
 [5] BSgenome_1.34.0                    rtracklayer_1.26.0
 [7] Biostrings_2.34.0                  XVector_0.6.0
 [9] GenomicRanges_1.18.0               GenomeInfoDb_1.2.0
[11] IRanges_2.0.0                      S4Vectors_0.4.0
[13] BiocGenerics_0.12.0

loaded via a namespace (and not attached):
 [1] BBmisc_1.7              BatchJobs_1.4           BiocParallel_1.0.0
 [4] DBI_0.3.1               GenomicAlignments_1.2.0 RCurl_1.95-4.3
 [7] RSQLite_0.11.4          Rsamtools_1.18.0        XML_3.98-1.1
[10] base64enc_0.1-2         bitops_1.0-6            brew_1.0-6
[13] checkmate_1.4           class_7.3-11            codetools_0.2-9
[16] digest_0.6.4            fail_1.2                foreach_1.4.2
[19] iterators_1.0.7         sendmailR_1.2-1         stringr_0.6.2
[22] tools_3.1.1             zlibbioc_1.12.0
```