

TCseq: time course sequencing data analysis

Mengjun, Lei Gu

May 1, 2024

The TCseq package provides a unified suite for analysis of different types of time course sequencing data. It can be applied to transcriptomic time course data such as RNA-seq as well as epigenomic time course data such as ATAC-seq, ChIP-seq. The main focuses of this package are on differential analysis between different time points and temporal pattern analysis and visualization.

Unlike RNA-seq, the genomic regions of interest of sequencing data like ATAC-seq, ChIP-seq are not pre-defined and are specific to each experimental conditions, which limits the subsequential differential analysis between conditions. For those data type, the TCseq package provides functions to combine and merge conditionally specific genomic regions and generate a reference genomic regions for all conditions. This package then uses the negative binomial generalized linear model implemented in edgeR to provide differential analysis [1]. To capture the temporal patterns of the time course data, the package includes several unsupervised clustering methods to identify and a function to visualize the patterns.

This vignette uses an example ATAC-seq time course data to illustrate how to use the TCseq package.

1 Input data

The minimal input data for the TCseq are experiment design and reference genomic regions.

1.1 Generate reference genomic regions

For RNA-seq, the reference genomic regions are predefined (genes or exons). While for epigenome sequencing data, genomic regions of interest are usually defined as reads enriched regions which are also called peaks. peaks set for a given condition can be identified by peak callers such as MACs and is specific to that condition. The TCseq package provides a function to read in a set of peak set file in BED format, combines these files in to a single data frame, merges overlapping regions according use defined criteria and takes the largest bound as the reference region for all the overlapping regions. The merge criteria can be either absolute overlapping bases or overlapping ration (absolute overlapping bases divide minimum length of the regions to be merged).

If a set of BED files are available under certain directory, say `dir.peaks`, the file names of the BED files to be merged have common substring "narrowpeaks", then the reference genomic regions can be generated by:

```
> library(TCseq)

> dir <- dir.peaks
> gf <- peakreference(dir = dir, pattern = "narrowpeaks")
```

The resulting data frame have four columns as follows:

```
> data("genomicIntervals")
> head(genomicIntervals)
```

	chr	start	end	id
1	chr1	6453992	6454731	peak1
2	chr1	7823890	7824372	peak2
3	chr1	8029820	8030138	peak3
4	chr1	8030317	8030627	peak4
5	chr1	10880000	10880701	peak5
6	chr1	13154468	13154786	peak6

1.2 Create a TCA object

The TCseq uses an S4 class TCA to store all input data for subsequential analysis. When read counts table is not available, only data frames of experiment design and reference genomic regions are required to create a TCA object, TCseq also provides a function to generate counts table, to use the function, file names of BAM files for each sample/library have to be provided in the data frame of experiment design:

```
> # Experiment design
> data("experiment_BAMfile")
> head(experiment_BAMfile)
```

	sampleid	timepoint	group	bamfile
1	s1	0h	1	rep1_0h.bam
2	s2	24h	2	rep1_24h.bam
3	s3	40h	3	rep1_40h.bam
4	s4	56h	4	rep1_56h.bam
5	s5	72h	5	rep1_72h.bam
6	s6	120h	6	rep1_120h.bam

```
> # create a TCA object
> tca <- TCA(design = experiment_BAMfile, genomicFeature = genomicIntervals)
> tca
```

An object of class "TCA"

@design

	sampleid	timepoint	group	bamfile
1	s1	0h	1	rep1_0h.bam
2	s2	24h	2	rep1_24h.bam
3	s3	40h	3	rep1_40h.bam
4	s4	56h	4	rep1_56h.bam
5	s5	72h	5	rep1_72h.bam

7 more rows ...

@genomicFeature

	chr	start	end	id
1	chr1	6453992	6454731	peak1
2	chr1	7823890	7824372	peak2
3	chr1	8029820	8030138	peak3
4	chr1	8030317	8030627	peak4
5	chr1	10880000	10880701	peak5

2746 more rows ...

@clusterRes

An object of class "clust"

The count table then can be created (suppose the BAM files are store in the directory dir.BAM):

```
> tca <- countReads(tca, dir = dir.BAM)
```

When the counts table is available, BAM file information is not mandatory in the experiment design. Counts table can be provided when creating a TCA object:

```
> #Experiment design without BAM file information
> data("experiment")
> #Counts table
> data("countsTable")
> tca <- TCA(design = experiment, genomicFeature = genomicIntervals,
+           counts = countsTable)
> tca
```

An object of class "TCA"

@design

	sampleid	timepoint	group
1	s1	0h	1
2	s2	24h	2
3	s3	40h	3
4	s4	56h	4
5	s5	72h	5

7 more rows ...

@counts

	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12
peak1	344	243	169	70	57	20	298	199	135	63	54	34
peak2	72	114	91	93	133	164	55	71	93	116	150	191
peak3	28	50	109	115	94	109	60	89	75	129	85	101
peak4	28	49	110	113	103	108	59	89	77	131	83	104
peak5	464	344	280	154	108	33	444	255	259	155	97	32

2746 more rows ...

@genomicFeature

	chr	start	end	id
1	chr1	6453992	6454731	peak1
2	chr1	7823890	7824372	peak2
3	chr1	8029820	8030138	peak3
4	chr1	8030317	8030627	peak4
5	chr1	10880000	10880701	peak5

2746 more rows ...

@clusterRes

An object of class "clust"

The counts table can also be assigned to an existing TCA object:

```
> counts(tca) <- countsTable
```

In addition, a TCA object can also be created from an existing RangedSummarizedExperiment or SummarizedExperiment. For summarizedExperiment, additional reference genomic regions information must be provided, while for RangedSummarizedExperiment object, the reference genomic regions will be extracted directly from the object. For a SummarizedExperiment object:

```
> suppressWarnings(library(SummarizedExperiment))
> se <- SummarizedExperiment(assays=list(counts = countsTable), colData = experiment)
> tca <- TCAFromSummarizedExperiment(se = se, genomicFeature = genomicIntervals)
```

The TCA object with experiment design, read counts, reference genomic regions can be used for following differential analysis.

2 Differential Analysis

The differential event is detected by using the generalized linear model (GLM) methods [2] implemented in edgeR package.

```
> tca <- DBanalysis(tca)
```

Low quality genomic regions (read counts are low for all the time points) can also be filtered out. The following step only keeps genomic regions with two or more samples that have read counts more than 10.

```
> tca <- DBanalysis(tca, filter.type = "raw", filter.value = 10, samplePassfilter = 2)
```

Differential analysis results between given timepoints can be extracted by:

```
> DBres <- DBresult(tca, group1 = "0h", group2 = c("24h", "40h", "72h"))
> str(DBres, strict.width = "cut")
```

```
Formal class 'CompressedGRangesList' [package "GenomicRanges"] with 5 slots
..@ unlistData      :Formal class 'GRanges' [package "GenomicRanges"] with 7 ..
.. .. ..@ seqnames   :Formal class 'Rle' [package "S4Vectors"] with 4 sl..
.. .. ..@ values     : Factor w/ 21 levels "chr1","chr2",...: 1 10 ..
.. .. ..@ lengths    : int [1:180] 63 42 56 35 44 33 42 35 42 28 ...
.. .. ..@ elementMetadata: NULL
.. .. ..@ metadata   : list()
.. .. ..@ ranges     :Formal class 'IRanges' [package "IRanges"] with 6 ..
.. .. ..@ start      : int [1:8253] 6453992 7823890 8029820 803031..
.. .. ..@ width      : int [1:8253] 740 483 319 311 702 319 491 39..
.. .. ..@ NAMES      : chr [1:8253] "peak1" "peak2" "peak3" "peak"..
.. .. ..@ elementType : chr "ANY"
.. .. ..@ elementMetadata: NULL
.. .. ..@ metadata   : list()
.. .. ..@ strand     :Formal class 'Rle' [package "S4Vectors"] with 4 sl..
.. .. ..@ values     : Factor w/ 3 levels "+","-","*": 3
.. .. ..@ lengths    : int 8253
.. .. ..@ elementMetadata: NULL
.. .. ..@ metadata   : list()
.. .. ..@ seqinfo     :Formal class 'Seqinfo' [package "GenomeInfoDb"] wi..
.. .. ..@ seqnames   : chr [1:21] "chr1" "chr2" "chr3" "chr4" ...
.. .. ..@ seqlengths : int [1:21] NA NA NA NA NA NA NA NA NA NA ...
.. .. ..@ is_circular: logi [1:21] NA NA NA NA NA NA NA ...
.. .. ..@ genome     : chr [1:21] NA NA NA NA ...
.. .. ..@ elementMetadata:Formal class 'DFrame' [package "S4Vectors"] with 6..
.. .. ..@ rownames   : NULL
.. .. ..@ nrow       : int 8253
.. .. ..@ elementType : chr "ANY"
.. .. ..@ elementMetadata: NULL
.. .. ..@ metadata   : list()
.. .. ..@ listData    :List of 4
.. .. .. ..$ logFC : num [1:8253] -0.232 0.836 0.984 0.992 -0.304 ...
.. .. .. ..$ PValue: num [1:8253] 0.156923 0.000133 0.000897 0.000831 ..
.. .. .. ..$ paj   : num [1:8253] 0.37247 0.00377 0.01378 0.01314 0.14..
.. .. .. ..$ id    : chr [1:8253] "peak1" "peak2" "peak3" "peak4" ...
.. .. ..@ elementType : chr "ANY"
.. .. ..@ metadata     : list()
..@ elementMetadata:Formal class 'DFrame' [package "S4Vectors"] with 6 slots
```

```

.. .. ..@ rownames      : NULL
.. .. ..@ nrows         : int 3
.. .. ..@ elementType   : chr "ANY"
.. .. ..@ elementMetadata: NULL
.. .. ..@ metadata      : list()
.. .. ..@ listData      : Named list()
..@ elementType        : chr "GRanges"
..@ metadata           : list()
..@ partitioning       :Formal class 'PartitioningByEnd' [package "IRanges"] wit..
.. .. ..@ end           : int [1:3] 2751 5502 8253
.. .. ..@ NAMES         : chr [1:3] "24hvs0h" "40hvs0h" "72hvs0h"
.. .. ..@ elementType   : chr "ANY"
.. .. ..@ elementMetadata: NULL
.. .. ..@ metadata      : list()

```

```
> head(DBres$`24hvs0h`)
```

GRanges object with 6 ranges and 4 metadata columns:

	seqnames	ranges	strand	logFC	PValue	paj
	<Rle>	<IRanges>	<Rle>	<numeric>	<numeric>	<numeric>
peak1	chr1	6453992-6454731	*	-0.232273	0.156922944	0.37247197
peak2	chr1	7823890-7824372	*	0.835753	0.000132918	0.00376967
peak3	chr1	8029820-8030138	*	0.984173	0.000896762	0.01378418
peak4	chr1	8030317-8030627	*	0.991673	0.000830989	0.01313822
peak5	chr1	10880000-10880701	*	-0.304292	0.033572917	0.14476347
peak6	chr1	13154468-13154786	*	-0.188395	0.404095979	0.65507840

id

	<character>
peak1	peak1
peak2	peak2
peak3	peak3
peak4	peak4
peak5	peak5
peak6	peak6

```
-----
seqinfo: 21 sequences from an unspecified genome; no seqlengths
```

Significant differential events ($\log_2\text{-fold} > 2$ or $\log_2\text{-fold} < -2$, adjusted p-value < 0.05) can be further extracted by:

```
> DBres.sig <- DBresult(tca, group1 = "0h", group2 = c("24h","40h","72h"), top.sig = TRUE)
> str(DBres.sig, strict.width = "cut")
```

Formal class 'CompressedGRangesList' [package "GenomicRanges"] with 5 slots

```

..@ unlistData      :Formal class 'GRanges' [package "GenomicRanges"] with 7 ..
.. .. ..@ seqnames  :Formal class 'Rle' [package "S4Vectors"] with 4 sl..
.. .. .. ..@ values   : Factor w/ 21 levels "chr1","chr2",...: 14 19..
.. .. .. ..@ lengths  : int [1:151] 1 1 1 2 1 1 3 2 1 1 ...
.. .. .. ..@ elementMetadata: NULL
.. .. .. ..@ metadata  : list()
.. .. ..@ ranges     :Formal class 'IRanges' [package "IRanges"] with 6 ..
.. .. .. ..@ start    : int [1:1066] 60749878 28531781 105257039 26..
.. .. .. ..@ width    : int [1:1066] 399 369 954 310 211 411 439 46..
.. .. .. ..@ NAMES    : chr [1:1066] "peak767" "peak1298" "peak166"..
.. .. .. ..@ elementType : chr "ANY"

```

```

.. .. .. .. ..@ elementMetadata: NULL
.. .. .. .. ..@ metadata      : list()
.. .. ..@ strand      :Formal class 'Rle' [package "S4Vectors"] with 4 sl..
.. .. .. .. ..@ values      : Factor w/ 3 levels "+","-","*": 3
.. .. .. .. ..@ lengths     : int 1066
.. .. .. .. ..@ elementMetadata: NULL
.. .. .. .. ..@ metadata      : list()
.. .. ..@ seqinfo      :Formal class 'Seqinfo' [package "GenomeInfoDb"] wi..
.. .. .. .. ..@ seqnames     : chr [1:21] "chr1" "chr2" "chr3" "chr4" ...
.. .. .. .. ..@ seqlengths   : int [1:21] NA NA NA NA NA NA NA NA NA NA ...
.. .. .. .. ..@ is_circular: logi [1:21] NA NA NA NA NA NA NA ...
.. .. .. .. ..@ genome       : chr [1:21] NA NA NA NA ...
.. .. ..@ elementMetadata:Formal class 'DFrame' [package "S4Vectors"] with 6..
.. .. .. .. ..@ rownames     : NULL
.. .. .. .. ..@ nrows        : int 1066
.. .. .. .. ..@ elementType   : chr "ANY"
.. .. .. .. ..@ elementMetadata: NULL
.. .. .. .. ..@ metadata      : list()
.. .. .. .. ..@ listData      :List of 4
.. .. .. .. .. ..$ logFC : num [1:1066] 2.54 2.27 2.16 2.74 2.01 ...
.. .. .. .. .. ..$ PValue: num [1:1066] 3.74e-11 1.68e-07 7.59e-20 1.66e-06 ..
.. .. .. .. .. ..$ paj      : num [1:1066] 3.43e-08 3.08e-05 2.09e-16 1.50e-04 ..
.. .. .. .. .. ..$ id       : chr [1:1066] "peak767" "peak1298" "peak1667" "pe"..
.. .. .. .. ..@ elementType   : chr "ANY"
.. .. .. .. ..@ metadata      : list()
.. ..@ elementMetadata:Formal class 'DFrame' [package "S4Vectors"] with 6 slots
.. .. .. .. ..@ rownames     : NULL
.. .. .. .. ..@ nrows        : int 3
.. .. .. .. ..@ elementType   : chr "ANY"
.. .. .. .. ..@ elementMetadata: NULL
.. .. .. .. ..@ metadata      : list()
.. .. .. .. ..@ listData      : Named list()
.. ..@ elementType       : chr "GRanges"
.. ..@ metadata          : list()
.. ..@ partitioning       :Formal class 'PartitioningByEnd' [package "IRanges"] wit..
.. .. .. .. ..@ end          : int [1:3] 357 714 1066
.. .. .. .. ..@ NAMES        : chr [1:3] "24hvs0h" "40hvs0h" "72hvs0h"
.. .. .. .. ..@ elementType   : chr "ANY"
.. .. .. .. ..@ elementMetadata: NULL
.. .. .. .. ..@ metadata      : list()

```

3 Temporal pattern analysis

3.1 Construct time course table

To detect temporal patterns of the time course sequencing data, the TCseq package uses unsupervised clustering methods. First, a time course table is created for clustering analysis. The rows of the time course table are genomic regions, and the columns are time points, the values can be chosen from normalized read counts or logFC of all time points compared to a given group. Here we compare each time point with the initial time point. Such table can be created as follows:

```

> # values are logFC
> tca <- timecourseTable(tca, value = "FC", control.group = "0h", norm.method = "rpkm", filter =
or

```

```
> # values are normalized read counts
> tca <- timecourseTable(tca, value = "expression", norm.method = "rpkm", filter = TRUE)
```

When the "filter" parameter is set to be TRUE, the time course table will filter out all genomic regions with no significant changes between any two time points. The table can be accessed by:

```
> t <- tcTable(tca)
> head(t)
```

	0h	24h	40h	56h	72h	120h
peak767	20.602720	122.53103	215.45264	457.1920	679.5218	1001.9488
peak1298	19.677065	100.91683	297.78073	668.2194	779.7382	250.8255
peak1667	31.716975	142.26128	387.47324	666.9718	800.7809	634.0918
peak2103	46.542402	313.97751	362.09517	857.1818	979.5473	976.4725
peak2129	17.421163	72.14858	237.62324	490.9435	781.7799	448.2977
peak2369	2.523052	22.77127	65.60794	189.1049	388.9364	1090.6255

3.2 Clustering analysis

Two types of clustering algorithms are included in the package: hard clustering (hierachical, pam, kmeans) and soft clustering (fuzzy cmeans [3]). The temporal patterns are analyzed using the following function:

```
> tca <- timeclust(tca, algo = "cm", k = 6, standardize = TRUE)
```

Instead of absolute value of different time series, one might only focus on the change patterns and expect time series with similar pattern to be cluster in same group. In this case, "standardize" parameter gives an option to perform z-score transformation on the data to be clustered, which reduces the noises introduced by the difference in the absolute values.

3.3 Visualize the clustering results

The clustering results can be visualized as follows:

```
> p <- timeclustplot(tca, value = "z-score(PRKM)", cols = 3)
```

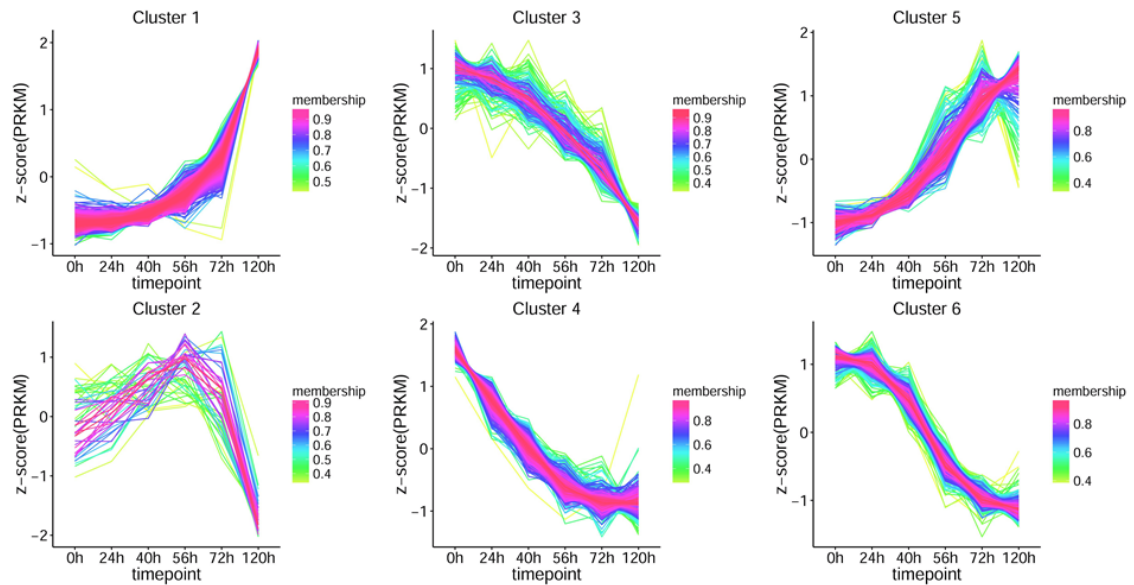


Figure 1: Visualization of clustering results

Individual clusters can also be plotted:

```
> #plot cluster 1:  
> print(p[[1]])
```



Figure 2: Visualization of cluster 1

To plot the cmeans clustering results, the TCseq provides several color schemes to color code the membership values which indicate the degree to which data points belong to a cluster.

References

- [1] Robinson, M.D., McCarthy, D.J. and Smyth, G.K. edgeR: a Bioconductor package for differential expression analysis of digital gene expression data, *Bioinformatics*, 26, 139-140,2010.
- [2] McCarthy,D.J.,Chen, Y., Smyth, G. K. Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. *Nucleic acids research* 40, 4288-4297,2012.
- [3] Futschik, M.E. and Carlisle, B. Noise-robust soft clustering of gene expression time-course data, *Journal of bioinformatics and computational biology*, 3, 965-988, 2005.
- [4] L. Kumar and M. Futschik, Mfuzz: a software package for soft clustering of microarray data, *Bioinformation*, 2(1),5-7,2007