A R Package for Changepoint Detection

Thales Melo
Florencia Leonardi

2018-11-23

Contents

Abstract

We describe the implementation of an R package focused in detecting changepoints in a matrix dataset, according to segments that maximize the system's segment likelihoods according to a given likelihood function, which dictate how we want to segment the data, and a penalty function, which can be used to do changes.

This is the second paragraph.

Resumo

Resumo do pacote em português.

Chapter 1

Introduction

This describes the development of the R package.

1.1 Motivation

To make it easy to detect changepoints in R by researchers.

1.2 Main References

This packages was bases on ???

1.3 Similar Work

There is similar work done by `fpop` package

Chapter 2

Main Definition

Consider a matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ of real data (discrete or continous), where p is the number of columns (variables) and n is the sample size (number of individuals). We assume the individuals are not related and therefore the rows of $\mathbf{X}$ are independent.

For $j \in \{1, \ldots, p\}$ we denote by $\mathbf{X}^j$ the jth column of $\mathbf{X}$ and given an interval of contiguous variables $I \subset \{1, \ldots, p\}$ we denote by $\mathbf{X}_I$ the sub-matrix with columns corresponding to elements of I.

In this context we assume there is a sequence of intervals (a partition) $S_0 = \{I_1, \ldots, I_k\}$ covering the entire range of variables such that the distribution of $\mathbf{X}$ is `homogeneous` inside each block I, that is $\mathbf{X}_i^j \sim f_{I(j)} \quad$ for all $i = 1, \ldots, n$, where $I(j)$ is the unique interval in S_0 containing marker j.

The function $f_{I(j)}$ represents a density function in the case of continuous random variables or a probability mass function in the case of discrete random variables. Moreover we assume the sequence $S_0 = \{I_1, \ldots, I_k\}$ is `minimal` in the sense that we can not merge two sub-sequent intervals and obtain an equivalent model. This is equivalent to assume that $f_{I_\ell} \neq f_{I_{\ell+1}}$ for all $\ell = 1, \ldots, k-1$.

Observe that we are not assuming independence between the columns of $\mathbf{X}$, this is an additional assumption that depends on the specific application. Moreover, the independence between the rows of $\mathbf{X}$ could also be ignored, but we assume it by simplicity.

Chapter 3

Model Selection Criterion

We propose a method to estimate this minimal set of intervals % $S = \{I_1, \ldots, I_k\}$ by optimising a criterion over all possible sets of intervals, namely

$$\hat{S} \;=\; \arg\max_{S} \left\{ \sum_{I \in S} (\log \hat{\mathbb{P}}(\mathbf{X}_I) - \text{pen}(I)) \right\}, \tag{3.1}$$

where pen is a non-decreasing function of $|I|$, the size of the interval I.

The R package `segmentr` solves the optimisation problem (3.1) $\hat{s} \;=\; \arg\min_{s} \sum_s (f(\mathbf{X}_s) + h(s))$, where $\mathbf{X} \in \mathbb{R}^{n \times p}$ and s is a set of `segments` over the set $1, \ldots, p$. Basically, s can be viewed as an ordered subset $s = \{0, s_1, \ldots, s_k\} \subset \{0, 1, \ldots, p\}$, such that the segments (intervals) are given by $I_j = \{s_{j-1} + 1, \ldots, s_j\} \qquad j = 1, \ldots, k\,.$

The function $c(\mathbf{X}, s) = \sum_s f(\mathbf{X}_s) + h(s)$ is a `cost` function that tries to be optimised over the possible segmentations s. It is defined as a sum because in general f defines a measure of `fitting` to the dataset and h defines a measure of complexity of the segmentation s.

Chapter 4

Efficient Algorithms

The algorithm described in 3 gives us the exact answer to the dataset provided. The complexity of the algithm, however, is $O(n^2)$, which means the execution time will be prohibitive for very large datasets.

Taking that into consideration, two different algorithms are proposed in order to resuce the complexity of the algorithm.

4.1 Hierarchical algorithm

The `hierarchical algorithm` assumes the changepoints have a hierarchical structure. The way it works is the algorithm goes throgh the data once, identify a point that maximizes the likelihood function, and then recursively calls the function on the segments divided by the newly found changepoint.

4.2 Hybrid algorithm

The `hybrid algorithm` uses a mixed approach, in which it used the hierarchical approach for when the segments are larger than a specified `threshold`, and uses the exact approach for when the segments are smaller than the threshold.

Chapter 5

Package Usage

This section shows how to use the package.

5.1 Instalation

The package can be installed using `devtools`.

```
install.packages("devtools")
require("devtools")
install_github("thalesmello/segmentr")
```

5.2 Usage

The package can be used with the `segment` function.

```r
require("segmentr")
```

```
## Loading required package: segmentr
```

```r
data <- rbind(
  c(1, 1, 0, 0, 0, 1, 1, 1),
  c(1, 1, 0, 0, 0, 1, 1, 1),
  c(1, 1, 0, 0, 0, 1, 1, 1),
  c(1, 1, 0, 0, 0, 1, 1, 1),
  c(1, 1, 0, 0, 0, 1, 1, 1),
  c(1, 1, 0, 0, 0, 1, 1, 1)
)

segment(
  data,
  algorithm = "exact",
  penalty = function(X) (0.1 * 2^ncol(X)) * log(nrow(X))
)$changepoints
```

```
## [1] 2 4 6
```

Chapter 6

Simulations

This section shows simulations using the package.

Chapter 7

Real Data Examples

This shows how the package can be used to analyze realworld data.

Chapter 8

Considerations on performance

This code shows a handful of attempts that were made in actually trying to make the code run faster.

8.1 Benchmark the code

The first step is to benchmark a first slow version of the code.

8.2 Native Code

After identifying the bottleneck of the code, we can try to rewrite that function in a more optimized manner. For that, we use RCPP to implement the commonly used `multivariate` function in C++. That allows us to get better in the R code execution.

8.3 Parallelization

Another approach to try to optimize the problem is to parallelize the computation. This is rather complicated in this scenario because, as a general rule, a stage of the computation depends on the previous stage. That means the computation can't be easily split across multiple nodes, as they frequently would have to wait for the computations in other nodes to finish.

Still, we are able to introduce some parallelization wherever possible as described in the equation ???.

Using the package `foreach`, we implemented parallel computation in the package, for which some performance gains can be observed in a handful of scenarios.

In general, it's useful to parallelize when the likelihood function takes a considerable amount of time to compute the likelihood of a segment. Because each computing node takes a considerable amount of time computing, it's worth it to split the task in multiple workers. However, when that same time is fast, due to a small number of samples in the segment, for example, the log likelihood function tends to compute in a fairly smaller time.

8.4 Benchmark

In the table we show the code snippets for the benchmark and a table comparing the results.

Chapter 9

Conclusion

This paper describes a method for identifying changepoints in a given dataset, and then goes on to implement it. Many attempts were also made to make the code run faster.

Chapter 10

References

- fpop
- RStudio
- RCPP
- original research paper from Florencia
- benchmark package