

## **fastseg**

### **An R Package for fast segmentation**

**Günter Klambauer and Andreas Mitterecker**

Institute of Bioinformatics, Johannes Kepler University Linz  
Altenberger Str. 69, 4040 Linz, Austria  
*klambauer@bioinf.jku.at*

**Version 1.18.0, May 3, 2016**

## **Scope and Purpose of this Document**

This document is a user manual for the R package `fastseg`. It is only meant as a gentle introduction into how to use the basic functions implemented in this package. Not all features of the R package are described in full detail. Such details can be obtained from the documentation enclosed in the R package. Further note the following: (1) this is neither an introduction to segmentation algorithms; (2) this is not an introduction to R. If you lack the background for understanding this manual, you first have to read introductory literature on these subjects.

## **Contents**

## 1 Introduction

`fastseg` implements a very fast and efficient segmentation algorithm. It has similar functionality as `DNACopy` (?) but is considerably faster and more flexible. `fastseg` can segment data stemming from DNA microarrays and data stemming from next generation sequencing for example to detect copy number segments. Further it can segment data stemming from RNA microarrays like tiling arrays to identify transcripts. Most generally, it can segment data given as a matrix or as a vector. Various data formats can be used as input to `fastseg` like expression set objects for microarrays or `GRanges` for sequencing data.

The segmentation criterion of `fastseg` is based on a statistical test in a Bayesian framework, namely the cyber t-test (?). The speed-up stems from the facts, that sampling is not necessary in for `fastseg` and that a dynamic programming approach is used for calculation of the segments' first and higher order moments.

For further information regarding the algorithm and its assessment see the `fastseg` homepage at <http://www.bioinf.jku.at/software/fastseg/fastseg.html>

## 2 Getting started

To load the package, enter the following in your R session:

```
> library(fastseg)
```

### 2.1 Data

According to the `DNACopy` package from bioconductor we selected a subset of the data set presented in (?). This data set will be called `coriell`. The data correspond to two array CGH studies of fibroblast cell strains.<sup>1</sup> In particular, the studies **GM05296** and **GM13330** were chosen. After selecting only the mapped data from chromosomes 1-22 and X, there are 2271 data points.

To prepare the data for our examples we execute the following code:

```
> data(coriell)
> head(coriell)
```

	Clone	Chromosome	Position	Coriell.05296	Coriell.13330
1	GS1-232B23	1	1	0.000359	0.207470
2	RP11-82d16	1	469	0.008824	0.063076
3	RP11-62m23	1	2242	-0.000890	0.123881
4	RP11-60j11	1	4505	0.075875	0.154343
5	RP11-111005	1	5441	0.017303	-0.043890
6	RP11-51b04	1	7001	-0.006770	0.094144

<sup>1</sup>[http://www.nature.com/ng/journal/v29/n3/supinfo/ng754\\_S1.html](http://www.nature.com/ng/journal/v29/n3/supinfo/ng754_S1.html)

```
> samplenames <- colnames(coriell)[4:5]
> data <- as.matrix(coriell[4:5])
> #data[is.na(data)] <- median(data, na.rm=TRUE)
> chrom <- coriell$Chromosome
> maploc <- coriell$Position
```

The main functions of the package are `fastseg` and `toDNACopyObj`. The first one runs the segmentation algorithm and the latter converts the segmentation results to a `DNACopy` object which will be quite helpful for plot functions.

## 2.2 File formats

The package can handle different file formats: `GRanges`, `ExpressionSet` objects, matrix or a vector.

### 2.2.1 Vector

```
> data2 <- data[, 1]
> res <- fastseg(data2)
> head(res)
```

`GRanges` object with 6 ranges and 5 metadata columns:

	seqnames	ranges	strand	ID	num.mark	seg.mean
	<Rle>	<IRanges>	<Rle>	<character>	<numeric>	<numeric>
[1]	1	[ 1, 1227]	*	sample1	1227	-0.003604316
[2]	1	[1228, 1270]	*	sample1	43	0.461622814
[3]	1	[1271, 1357]	*	sample1	87	0.004317655
[4]	1	[1358, 1372]	*	sample1	15	-0.651081333
[5]	1	[1373, 2214]	*	sample1	842	0.014980804
[6]	1	[2215, 2271]	*	sample1	57	0.614116421

	startRow	endRow
	<integer>	<integer>
[1]	1	1227
[2]	1228	1270
[3]	1271	1357
[4]	1358	1372
[5]	1373	2214
[6]	2215	2271

-----

seqinfo: 1 sequence from an unspecified genome; no seqlengths

```
>
>
```

### 2.2.2 Matrix

```
> data2 <- data[1:400, ]
> res <- fastseg(data2)
> head(res)
```

GRanges object with 6 ranges and 5 metadata columns:

	seqnames	ranges	strand	ID	num.mark	seg.mean
	<Rle>	<IRanges>	<Rle>	<character>	<numeric>	<numeric>
[1]	1	[ 1, 80]	*	Coriell.05296	80	0.016815675
[2]	1	[ 81, 84]	*	Coriell.05296	4	0.134374750
[3]	1	[ 85, 400]	*	Coriell.05296	316	0.003267547
[4]	1	[ 1, 91]	*	Coriell.13330	91	0.016150637
[5]	1	[ 92, 140]	*	Coriell.13330	49	0.485243673
[6]	1	[141, 400]	*	Coriell.13330	260	-0.032339504

	startRow	endRow
	<integer>	<integer>
[1]	1	80
[2]	81	84
[3]	85	400
[4]	1	91
[5]	92	140
[6]	141	400

-----

seqinfo: 1 sequence from an unspecified genome; no seqlengths

### 2.2.3 GRanges objects

```
> library("GenomicRanges")
> ## with both individuals
> gr <- GRanges(seqnames=chrom,
+               ranges=IRanges(maploc, end=maploc))
> mcols(gr) <- data
> colnames(mcols(gr)) <- samplenames
> res <- fastseg(gr)
> head(res)
```

GRanges object with 6 ranges and 5 metadata columns:

	seqnames	ranges	strand	ID	num.mark	seg.mean
	<Rle>	<IRanges>	<Rle>	<character>	<integer>	<numeric>
[1]	1	[ 1, 240001]	*	Coriell.05296	141	0.01973119
[2]	10	[ 1, 65001]	*	Coriell.05296	57	-0.01061286
[3]	10	[ 66906, 108904]	*	Coriell.05296	43	0.45160927
[4]	10	[110001, 142001]	*	Coriell.05296	34	0.00403140
[5]	11	[ 1, 34421]	*	Coriell.05296	52	0.01163838
[6]	11	[ 35417, 39624]	*	Coriell.05296	14	-0.65108133

```

      startRow    endRow
    <integer> <integer>
[1]         1      142
[2]         1       58
[3]        59      102
[4]       103      137
[5]         1       53
[6]        54       68
-----
seqinfo: 23 sequences from an unspecified genome; no seqlengths

> ## with one individual
> gr2 <- gr
> data2 <- as.matrix(data[, 1])
> colnames(data2) <- "sample1"
> mcols(gr2) <- data2
> res <- fastseg(gr2)
> head(res)

GRanges object with 6 ranges and 5 metadata columns:
      seqnames      ranges strand |          ID  num.mark  seg.mean
      <Rle>      <IRanges> <Rle> | <character> <integer>  <numeric>
[1]         1 [ 1, 240001]    * |    sample1     141  0.01973119
[2]        10 [ 1, 65001]    * |    sample1      57 -0.01061286
[3]        10 [ 66906, 108904] * |    sample1      43  0.45160927
[4]        10 [110001, 142001] * |    sample1      34  0.00403140
[5]        11 [ 1, 34421]    * |    sample1      52  0.01163838
[6]        11 [ 35417, 39624] * |    sample1      14 -0.65108133
      startRow    endRow
    <integer> <integer>
[1]         1      142
[2]         1       58
[3]        59      102
[4]       103      137
[5]         1       53
[6]        54       68
-----
seqinfo: 23 sequences from an unspecified genome; no seqlengths

>

```

### 2.2.4 ExpressionSet objects

```

> library(oligo)
> eSet <- new("ExpressionSet")

```

```

> assayData(eSet) <- list(intensity=data)
> featureData(eSet) <- new("AnnotatedDataFrame",
+   data=data.frame(
+     chrom = paste("chr",chrom,sep=""),
+     start = maploc,
+     end   = maploc,stringsAsFactors=FALSE))
> phenoData(eSet) <- new("AnnotatedDataFrame",
+   data=data.frame(samples=samplenames))
> sampleNames(eSet) <- samplenames
> res <- fastseg(eSet)
> head(res)

```

GRanges object with 6 ranges and 5 metadata columns:

	seqnames	ranges	strand	ID	num.mark	seg.mean
	<Rle>	<IRanges>	<Rle>	<character>	<integer>	<numeric>
[1]	chr1	[ 1, 240001]	*	Coriell.05296	141	0.01973119
[2]	chr10	[ 1, 65001]	*	Coriell.05296	57	-0.01061286
[3]	chr10	[ 66906, 108904]	*	Coriell.05296	43	0.45160927
[4]	chr10	[110001, 142001]	*	Coriell.05296	34	0.00403140
[5]	chr11	[ 1, 34421]	*	Coriell.05296	52	0.01163838
[6]	chr11	[ 35417, 39624]	*	Coriell.05296	14	-0.65108133

	startRow	endRow
	<integer>	<integer>
[1]	1	142
[2]	1	58
[3]	59	102
[4]	103	137
[5]	1	53
[6]	54	68

-----  
seqinfo: 23 sequences from an unspecified genome; no seqlengths

### 2.2.5 Vector

```

> data2 <- data[, 1]
> res <- fastseg(data2)
> head(res)

```

GRanges object with 6 ranges and 5 metadata columns:

	seqnames	ranges	strand	ID	num.mark	seg.mean
	<Rle>	<IRanges>	<Rle>	<character>	<numeric>	<numeric>
[1]	1	[ 1, 1227]	*	sample1	1227	-0.003604316
[2]	1	[1228, 1270]	*	sample1	43	0.461622814
[3]	1	[1271, 1357]	*	sample1	87	0.004317655
[4]	1	[1358, 1372]	*	sample1	15	-0.651081333
[5]	1	[1373, 2214]	*	sample1	842	0.014980804

```

[6]      1 [2215, 2271]      * |      sample1      57 0.614116421
      startRow      endRow
      <integer> <integer>
[1]      1      1227
[2]     1228      1270
[3]     1271      1357
[4]     1358      1372
[5]     1373      2214
[6]     2215      2271
-----
seqinfo: 1 sequence from an unspecified genome; no seqlengths

>
>

```

### 2.2.6 Matrix

```

> data2 <- data[1:400, ]
> res <- fastseg(data2)
> head(res)

```

GRanges object with 6 ranges and 5 metadata columns:

```

      seqnames      ranges strand |      ID num.mark      seg.mean
      <Rle>      <IRanges> <Rle> | <character> <numeric>      <numeric>
[1]      1 [ 1, 80]      * | Coriell.05296      80 0.016815675
[2]      1 [ 81, 84]      * | Coriell.05296      4 0.134374750
[3]      1 [ 85, 400]      * | Coriell.05296     316 0.003267547
[4]      1 [ 1, 91]      * | Coriell.13330      91 0.016150637
[5]      1 [ 92, 140]      * | Coriell.13330      49 0.485243673
[6]      1 [141, 400]      * | Coriell.13330     260 -0.032339504
      startRow      endRow
      <integer> <integer>
[1]      1      80
[2]     81      84
[3]     85     400
[4]      1      91
[5]     92     140
[6]    141     400
-----
seqinfo: 1 sequence from an unspecified genome; no seqlengths

```

## 2.3 Plotting the segmentation results

For plotting the data we have to generate an DNACopy object out of the segmentation results:



```
> ## with both individuals
> gr <- GRanges(seqnames=chrom,
+               ranges=IRanges(maploc, end=maploc))
> mcols(gr) <- data
> colnames(mcols(gr)) <- samplenames
> res <- fastseg(gr, segMedianT=0.2)
```

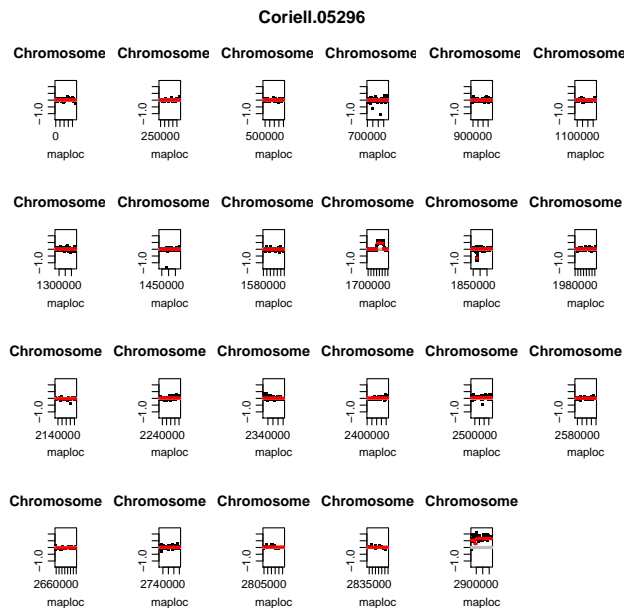
The plotting is done via the `plot` function of `DNAcopy`:

```
> segPlot(gr, res, plot.type="w")
```



Or alternatively:

```
> segPlot(gr, res, plot.type="s")
```



## 2.4 Performance of the method

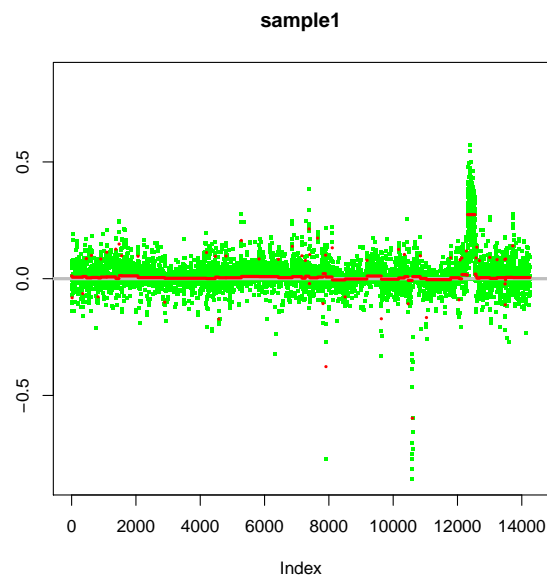
Here we show that `fastseg` outperforms `DNACopy` with respect to computational time on summarized microarray data. The quality of the segmentation result of both `fastseg` and `DNACopy` depends strongly on the methods' parameters.

The data is a small subset of copy number calls which were produced by the `cn.farms` algorithm ? from an Affymetrix SNP microarray experiment of a HapMap sample.

```
> data(fastsegData)
> system.time(res <- fastseg(fastsegData))
```

```
user  system elapsed
0.229   0.002   0.231
```

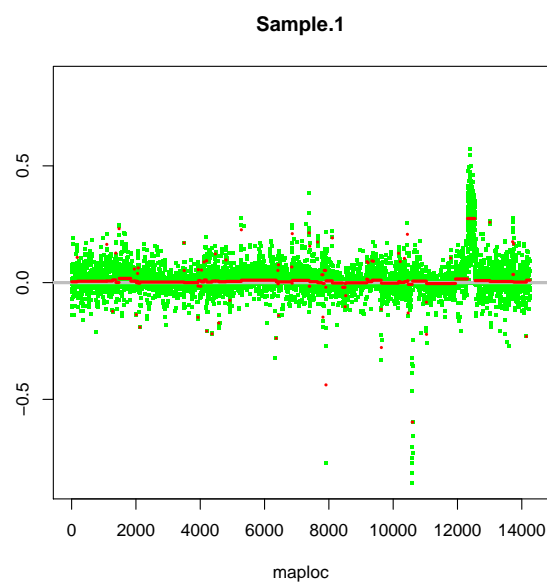
```
> segPlot(fastsegData,res, plot.type="w")
```



```
> library(DNACopy)
> cna <- DNACopy::CNA(fastsegData,chrom="chr1",maploc=1:length(fastsegData))
> system.time(res2 <- DNACopy::segment(cna))
```

```
Analyzing: Sample.1
  user  system elapsed
5.701   0.006   5.710
```

```
> plot(res2, plot.type="w", xmaploc=TRUE)
```



### 3 Future Extensions

We are planning to program a parallelized version of this package. Furthermore we will enhance the plot functions by our own.

### 4 How to cite this package

If you use this package for research that is published later, you are kindly asked to cite it as follows: (?).

To obtain Bib<sub>T</sub><sub>E</sub>X entries of the two references, you can enter the following into your R session:

```
> toBibtex(citation("fastseg"))
```