Scale4C: an R/Bioconductor package for scale-space transformation of 4C-seq data

Carolin Walter

June 27, 2025

Contents

1 Introduction

Scale4C uses Witkin's scale-space filtering to describe a 4C-seq signal (chromosome conformation capture combined with sequencing) qualitatively [Witkin, 1983]. To this extend, the original data signal is smoothed with Gauss kernels of increasing σ ("sigma"). Inflection points (i.e. the borders between different features) can be tracked through the resulting 2D matrix, or 'fingerprint map', which shows the position of a fragment on the x-axis and a range of smoothing σ on the y-axis. Each entry in the matrix encodes whether there is an inflection point at a certain position for a given σ (and if so, whether it's a switch from convex to concave or vice versa), or not. In this context, a 'singularity' is defined as a singular point in the fingerprint map, which means that both borders of the corresponding feature are identical for the singular point's σ. In case of Gauss kernel smoothing, this is only possible when a feature ('peak' / 'hill' or 'valley') appears or disappears, depending on whether the smoothing factor σ is decreased or increased [Witkin, 1983]. Tracing singularities down their inflection point contours through the fingerprint map allows for a precise localization of distorted, strongly smoothed features. The singularities and localization information can then be transformed into a simple tree structure. Each singularity hereby marks the point where three children spawn from one parent feature. Intuitively (and, in case of Gauss kernels, correctly [Witkin, 1983, Lee *et al.*, 2013]), one would assume that a peak is surrounded by two valleys and vice versa, and that when a smaller peak is smoothed out, the valleys around its former position will form one new valley, and that's what is reflected in the tree. All children of a chosen parent start at the same σ, however, their range of σ for which they persist can vary among the children.

These ternary trees can be visualized in a 2D tesselation map, which allows to assess prominent features with a high degree of stability for multiple values of σ visually.

1.1 Loading the package

After installation, the package can be loaded into R by typing

```
> library(Scale4C)
```

into the R console.

1.2 Provided functionality

Scale4C requires the R-packages *smoothie*, *IRanges*, *SummarizedExperiment* and *GenomicRanges*.

This package provides the following basic functions for the scale-space transformation of 4C-seq data:

- `calculateFingerprintMap`: first of two central functions; computes the inflection points for the signal when smoothed with different Gauss kernels
- `findSingularities`: second of two central functions; identifies and tracks singularities through the fingerprint map
- `plotTesselation`: plots the final tesselation of the scale space for the given 4C-seq signal, as provided by the singularities and tracking results

Optional functions include:

- `importBasic4CseqData`: transforms *Basic4Cseq*'s fragment-based table output for use with *Scale4C*
- `plotTraceback`: plots the original fingerprint map and traceback results for a given 4C-seq signal
- `plotInflectionPoints`: plots the smoothed 4C-seq signal for one chosen σ, with or without matching inflection points
- `outputScaleSpaceTree`: outputs a *GenomicRanges* object, including the size (i.e. range of σ) for detected features, based on the singularity-derived scale-space tree data

In addition to the examples presented in this vignette, more detailed information on the functions' parameters and additional examples are presented in the corresponding manual pages.

2 Scale-space transformation of 4C-seq data

Similar to *Basic4Cseq*, *Scale4C* includes fetal liver data of [Stadhouders *et al*., 2012] to demonstrate central functions. Due to size limits and performance issues, only a subset of its near-cis fragment data is included.

2.1 Import of raw fragment data

Scale4C expects a *GRanges* object as input, which includes chromosome name, start, end, "reads" and "meanPosition" (mean of start and end). If you have got an appropriate data frame or similar, the *GenomicRanges* contains the relevant conversion functions.

Since *Basic4Cseq* already offers an output function for fragment data, albeit with more information than *Scale4C* will ever need, a converter function is included in this package that can read *Basic4Cseq*'s output tables and extract the relevant data. Per default, this is the fragment data within a certain distance from the viewpoint of the experiment.

```
> csvFile <- system.file("extdata", "liverData.csv", package="Scale4C")
> liverReads <- importBasic4CseqData(csvFile, viewpoint = 21160072,
     viewpointChromosome = "chr10", distance = 1000000)
> head(liverReads)
```

```
GRanges object with 6 ranges and 2 metadata columns:
      seqnames            ranges strand |     reads meanPosition
         <Rle>         <IRanges>  <Rle> | <integer>    <numeric>
  [1]    chr10 21129612-21129613      * |      3074     21129612
  [2]    chr10 21132470-21132471      * |       400     21132470
  [3]    chr10 21132477-21132478      * |         2     21132478
  [4]    chr10 21141442-21141443      * |      3488     21141442
  [5]    chr10 21143414-21143415      * |      1295     21143414
  [6]    chr10 21143421-21143422      * |        10     21143422
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

Note: The *system.file()* expression is used to locate the example data from Stadhouders et al. For custom data, simply entering the full or relative path as a string is sufficient, e.g.

csvFile <- "../myFolder/myFragmentData.csv".

2.2 Initialisation of a Scale4C object

As soon as the data is available in R, we can create the raw *Scale4C* object.

```
> liverData = Scale4C(rawData = liverReads, viewpoint = 21160072, viewpointChromosome = "
> liverData
```

```
4C-seq scale space data
Type: Scale4C
Viewpoint:  chr10 : 21160072
Number of total fragments:  286
Points of interest:  0
Maximum sigma of fingerprint map:  -1
Number of singularities:  0
```

While the viewpoint of a 4C-seq experiment is usually clearly visible as its characteristic overrepresentation causes a visible peak in the data, including further reference points into near-cis plots may facilitate orientation.

```
> poiFile <- system.file("extdata", "vp.txt", package="Scale4C")
> pointsOfInterest(liverData) <- addPointsOfInterest(liverData,
     read.csv(poiFile, sep = "\t", stringsAsFactor = FALSE))
> head(pointsOfInterest(liverData))
```

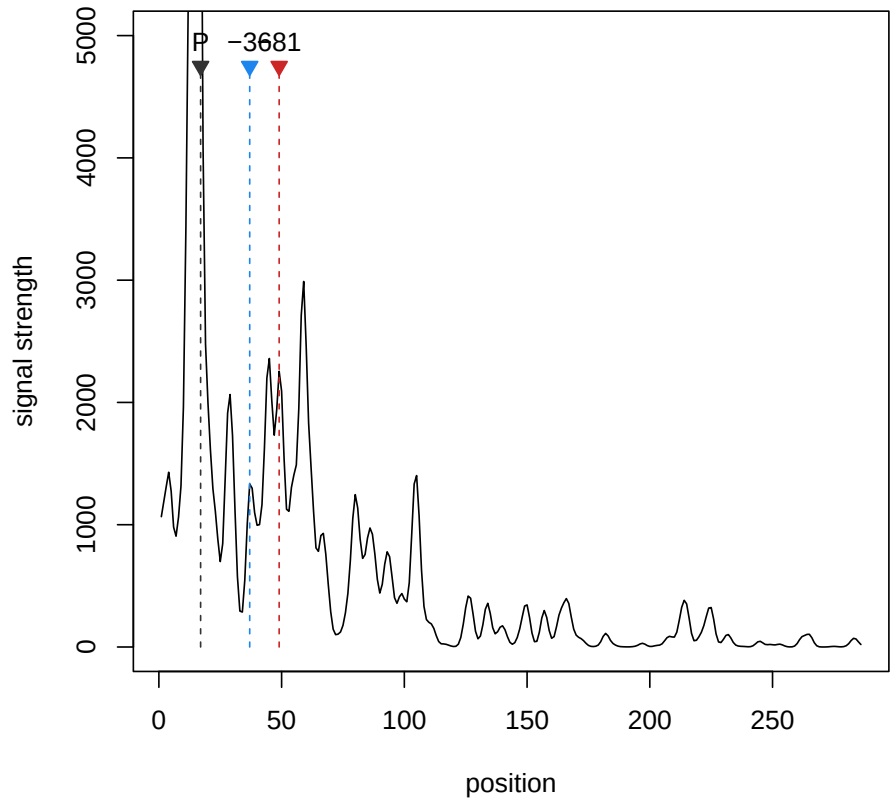
```
GRanges object with 3 ranges and 3 metadata columns:
      seqnames            ranges strand |        name      colour     index
         <Rle>         <IRanges>  <Rle> | <character> <character> <integer>
  [1]    chr10 21160072-21160073      * |           P      grey20        17
  [2]    chr10 21197709-21197710      * |         -36 dodgerblue2        37
  [3]    chr10 21241139-21241140      * |         -81  firebrick3        49
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

2.3 Smoothing, inflection points, fingerprint map, and singularities

Scale4C relies on two central functions to perform the scale-space transformation, `calculateFingerprintMap` and `findSingularities`. To create a fingerprint map, we smooth the 4C-seq signal with increasing values of σ for the Gauss kernel.

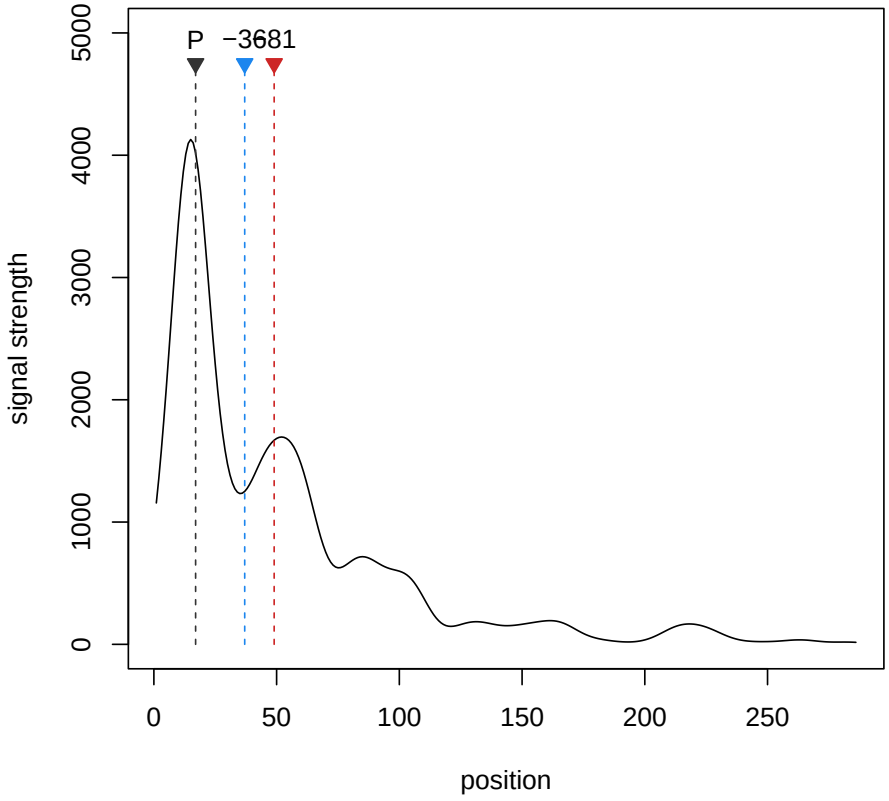
`plotInflectionPoints` can plot the smoothed data for a given σ, effectively showing a non-condensed 'slice' of the scale space and fingerprint map.

```
> plotInflectionPoints(liverData, 2, fileName = "", plotIP = FALSE)
```



For a σ of 2, both marked elements aside from the viewpoint are clearly located on peak regions.

```
> plotInflectionPoints(liverData, 50, fileName = "", plotIP = FALSE)
```



We can see that for a σ of 50, the -36 element of Stadhouders et al. has been smoothed over; the former peak is now located in a larger valley. The -81 element, however, is still located in a peak region.

Note: Since we haven't actually calculated the fingerprint map and inflection points up to now, *plotIP* should be set to FALSE.

In a similar manner, `calculateScaleSpace` and `calculateFingerprintMap` compute the whole fingerprint map, up to a provided maximum square σ. The fingerprint data is stored in the second assay of the *scaleSpace* slot of a *Scale4C* object. `findSingularities` then identifies singular points of the inflection point contours in scale-space [Witkin, 1983].

```
> scaleSpace(liverData) = calculateScaleSpace(liverData, maxSQSigma = 10)
> head(t(assay(scaleSpace(liverData), 1))[,1:5])
```

```
          [,1]     [,2]     [,3]     [,4]     [,5]
[1,] 1817.604  900.245  853.245 2241.585 1463.276
[2,] 1338.876 1097.941 1177.597 1744.064 1405.163
[3,] 1065.672 1189.540 1312.445 1429.122 1274.580
[4,]  986.730 1174.141 1291.356 1322.246 1221.554
[5,]  946.830 1136.560 1246.503 1265.563 1199.958
[6,]  918.388 1097.436 1202.648 1227.564 1191.668
```

To speed up the examples, we use a small maximum σ for the given example.

```
> liverData = calculateFingerprintMap(liverData, maxSQSigma = 10)
> singularities(liverData) = findSingularities(liverData, 1, guessViewpoint = FALSE)
```

The resulting fingerprint map and singularity trace for a maxSQSigma of 2000 are presented in figure 1.

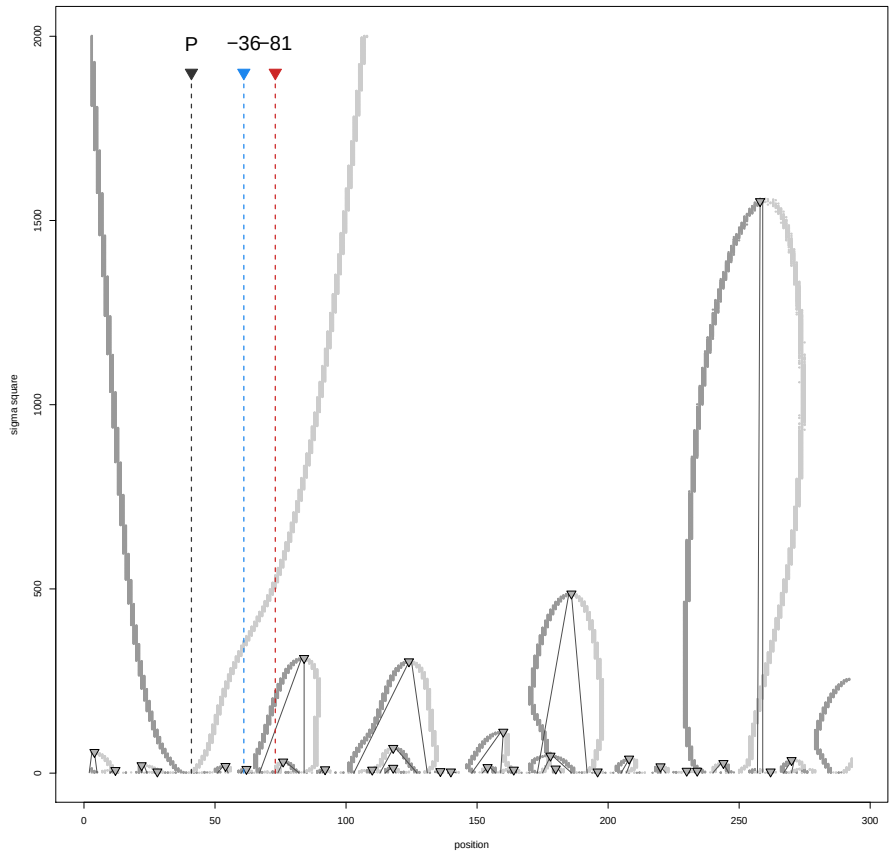


Figure 1: Fingerprint map with traced singularities for Stadhouders et al.'s example liver data [Stadhouders *et al.*, 2012]. The viewpoint wasn't tracked, since the corresponding contours do not meet for the chosen σ, and the largest remaining singularity's contours were not traced correctly.

There are two issues with the fingerprint map: First, the viewpoint is not identified, because the σ used is far too small for the inflection point contours to meet. Second, the largest identified singularity has wrong traces for both the left and right side: the dark gray tracking line doesn't match the corresponding inflection point curve or contour. Both problems should be corrected before tesselation. We can easily manipulate the list of singularities with R's usual data frame options, and load the corrected list back into our *Scale4C* object.

```
> data(liverData)
> tail(singularities(liverData))

GRanges object with 6 ranges and 4 metadata columns:
      seqnames    ranges strand |   sqsigma      left     right            type
         <Rle> <IRanges>  <Rle> | <numeric> <numeric> <numeric>      <character>
  [1]    chr10   118-119      * |        66       113       127           valley
```

```
  [2]    chr10   159-160      * |       110       148       159             peak
  [3]    chr10   124-125      * |       301       103       131             peak
  [4]    chr10     83-84      * |       310        67        84             peak
  [5]    chr10   185-186      * |       485       173       192             peak
  [6]    chr10   258-259      * |      1550       257       259 tracking problem
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths
> # add viewpoint singularity
> singularities(liverData) = c(singularities(liverData), GRanges("chr10", IRanges(39, 42)
    "*", "sqsigma" = 2000, "left" = 40, "right" = 41, "type" = "peak"))
> # correct singularity's coordinates
> tempSingularities = singularities(liverData)
> tempSingularities$left[30] = 235
> tempSingularities$right[30] = 250
> tempSingularities$type[30] = "peak"
> singularities(liverData) = tempSingularities
> tail(singularities(liverData))
GRanges object with 6 ranges and 4 metadata columns:
      seqnames    ranges strand |   sqsigma      left     right        type
         <Rle> <IRanges>  <Rle> | <numeric> <numeric> <numeric> <character>
  [1]    chr10   159-160      * |       110       148       159        peak
  [2]    chr10   124-125      * |       301       103       131        peak
  [3]    chr10     83-84      * |       310        67        84        peak
  [4]    chr10   185-186      * |       485       173       192        peak
  [5]    chr10   258-259      * |      1550       235       250        peak
  [6]    chr10     39-42      * |      2000        40        41        peak
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

Note: There are a number of possible problems with the identification and the tracking of singularities.

First, singularities can simply be missed altogether. This typically happens for few singularities with a high σ, i.e. huge remaining basic features of the data, when the upper part of the contours are so stretched that holes appear in the fingerprint map. While it is possible to adapt `calculateFingerprintMap` for a chance to catch these features (cp. manual page), it is often faster to just add a singularity and its corresponding left and right fragment at base σ level manually. If the 4C-seq data shows the typical peak at the viewpoint, this allows to speed up the calculations for the fingerprint map a bit: By ignoring the viewpoint singularity at extremely high σ and using its contours to visually identify its left and right extent, we can stop at a reasonable σ and get a feasible tesselation. The example code above adds the viewpoint peak to the example data's list of singularities.

Second, singularities can also be marked erroneously multiple times. This also tends to happen for very large and frazzled contours; either adapting `calculateFingerprintMap`'s *epsilon* or simply deleting the false singularities manually solves this problem. Identifying the correct singularity visually should be easy enough with `plotTraceback`'s output plots: A valid singularity is located on the top of a inflection point curve, and nowhere else.

Third, $\sigma = 1$ tends to be difficult to trace due to the number of inflection point curves in close proximity and identified singularities (part of which are usually wrong).

The results of the example's fingerprint map and singularity search after correction are shown in figure 2.

If the data looks 'normal', i.e. the 4C-seq signal creates mainly large 'peaks' in the fingerprint map, and no large 'valleys' are present, it is also possible to stop at a lower σ and let the function combine the remaining single contours by setting *guessViewpoint = TRUE*. In this case, it is assumed that all remaining curves in the fingerprint map belong to peaks, and singularities are formed accordingly. Results for such a combination ($\sigma = 500$) are shown in figure 3.

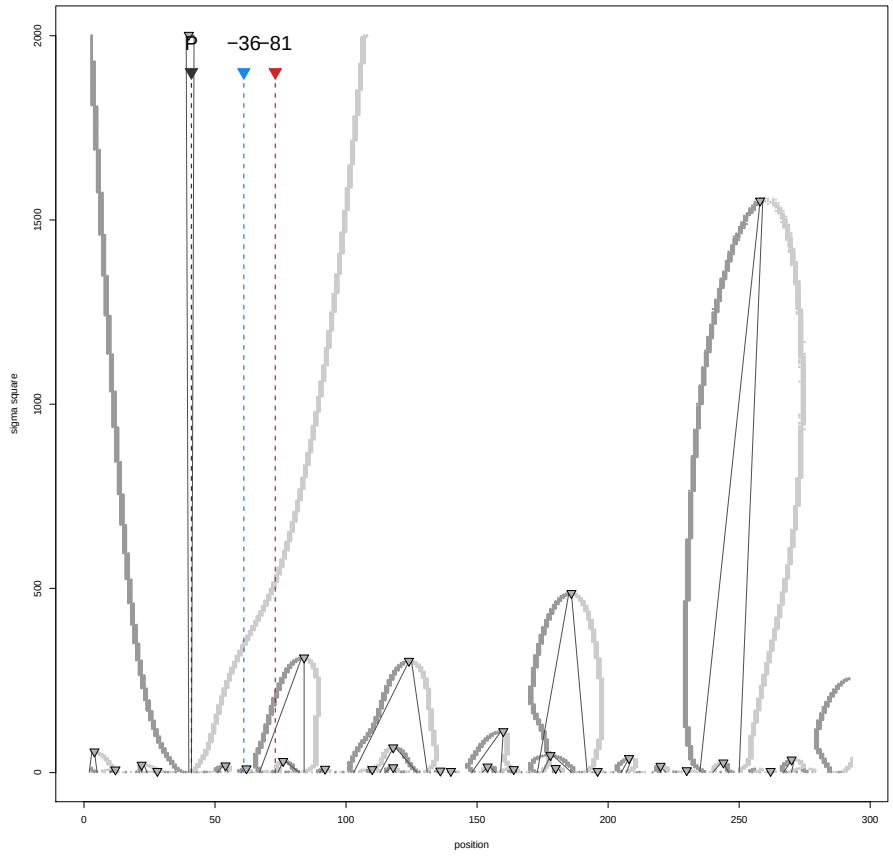


Figure 2: Fingerprint map after manual correction: The viewpoint is added, and the remaining singularities for Stadhouders et al.'s example liver data [Stadhouders *et al.*, 2012] are traced correctly.

2.4 Tesselation

If the fingerprint map and the singularity traceback appear to be satisfactory, the actual tesselation can be plotted. The example plot in figure 4 uses different colours to depict peaks (brown) and valleys (blue). Each singularity causes one parent feature to split into three child features (e.g. a single peak is split into peak-valley-peak); `plotTesselation` per default marks the central feature of a trio in a darker shade, and adjacent in a lighter one. Different colours for 'central' and 'adjacent' features allow for optical quality control of the tesselation, cp. manual page for `plotTesselation` and [Witkin, 1983, Lee *et al.*, 2013].

```
> # use pre-calculated example data with VP and correction
> data(liverDataVP)
> plotTesselation(liverDataVP, fileName = "")
```

The tesselation allows to assess the qualitative structure of the provided 4C-seq signal. The development of features through different smoothing scales can be traced, and comparisons between samples regarding their complexity become possible by comparing the number of singularities in a certain range of choice. Since 4C-seq is about contact intensities and chromosomal interactions, the most interesting regions tend to be peaks that persist for larger intervals of σ.

We can also output the underlying structure as a simple table, with or without σ $log2$ transformation. If

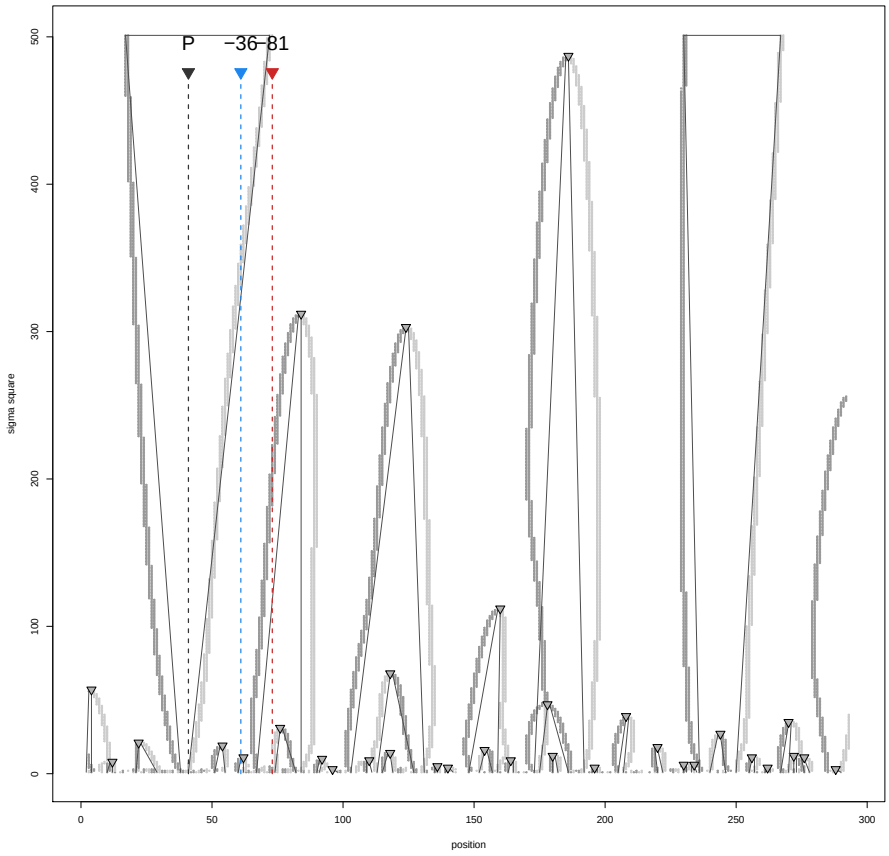


Figure 3: Fingerprint map for a smaller maximum σ of 500 with automatically matched contours.

outputPeaks is set to FALSE, the output table shows a list of all central features, as specified by singularities in the fingerprint map, and their corresponding adjacent left and adjacent right features with their range of persistence in σ, and left and right extent in fragments / genomic position. Otherwise, valleys are omitted from the output, and the data is exported as *GenomicRanges* object. The features' signals, i.e. the mean read counts for the identified intervals, are also provided.

```
> head(outputScaleSpaceTree(liverDataVP, useLog = FALSE))
```

```
GRanges object with 6 ranges and 1 metadata column:
      seqnames    ranges strand |     total
         <Rle> <IRanges>  <Rle> | <numeric>
  [1]    chr10     28-29      * |      3488
  [2]    chr10   139-142      * |        73
  [3]    chr10   195-197      * |       159
  [4]    chr10   261-263      * |        13
  [5]    chr10   135-136      * |       275
  [6]    chr10   230-231      * |         0
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

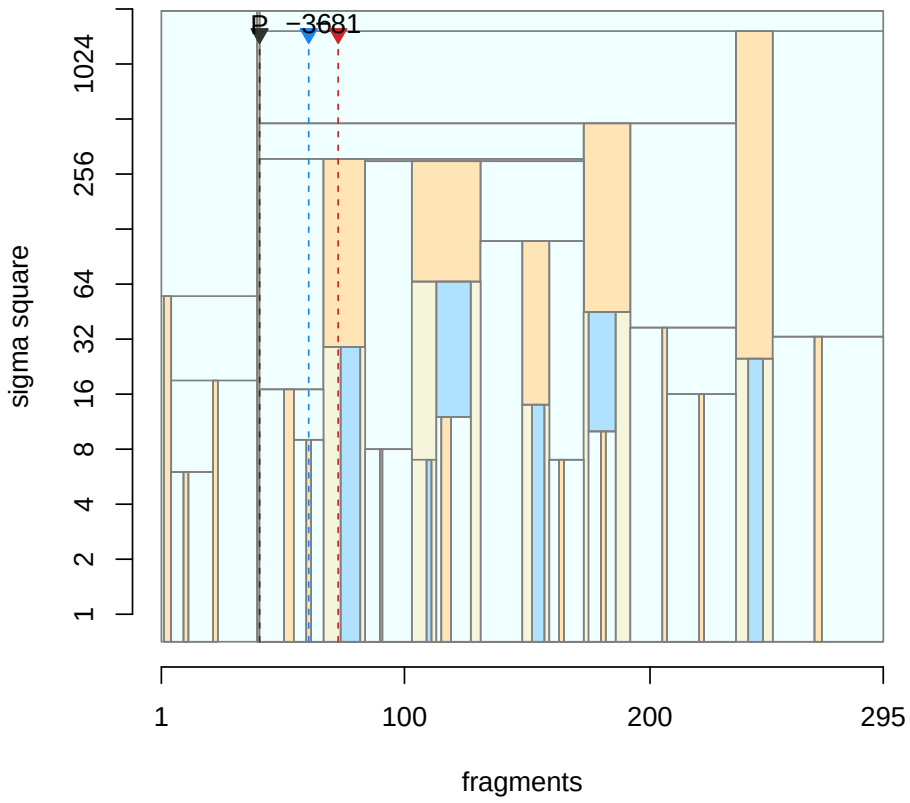


Figure 4: Tesselation of the x-σ-plane, as provided by the traced singularities in the fingerprint map. A slice at $\sigma = 50$ corresponds to the smoothed data plot from before, with the -36 element located in a valley and -81 at the neighbouring peak.

References

[Stadhouders *et al*., 2012] Stadhouders, R., Thongjuea, S., *et al* (2012) Dynamic long-range chromatin interactions control Myb proto-oncogene transcription during erythroid development, *EMBO*, **31**, 986-999.

[Witkin, 1983] Witkin, A. (1983) Scale-space Filtering, *Proceedings of the Eighth International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'83, 1019-1022.

[Lee *et al*., 2013] Lee, J., Lee, U., Kim, B., et al. (2013) A computational method for detecting copy number variations using scale-space filtering, *BMC Bioinformatics*, **14**, 57.

3 Session Information

```
R version 4.5.1 Patched (2025-06-14 r88325)
Platform: aarch64-apple-darwin20
```

```
Running under: macOS Ventura 13.7.1

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRlapack.dylib

locale:
[1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: America/New_York
tzcode source: internal

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

other attached packages:
 [1] Scale4C_1.31.1              SummarizedExperiment_1.39.1 Biobase_2.69.0
 [4] MatrixGenerics_1.21.0       matrixStats_1.5.0           GenomicRanges_1.61.1
 [7] Seqinfo_0.99.1              IRanges_2.43.0              S4Vectors_0.47.0
[10] BiocGenerics_0.55.0         generics_0.1.4              smoothie_1.0-4

loaded via a namespace (and not attached):
 [1] SparseArray_1.9.0   Matrix_1.7-3        lattice_0.22-7      abind_1.4-8
 [5] S4Arrays_1.9.1      XVector_0.49.0      grid_4.5.1          DelayedArray_0.35.2
 [9] compiler_4.5.1      tools_4.5.1         crayon_1.5.3
```