Similarity-Based Segmentation of Multi-Dimensional Signals

Rainer Machne
Douglas B. Murray
Peter F. Stadler

SFI WORKING PAPER: 2017-04-013

SFI Working Papers contain accounts of scientific work of the author(s) and do not necessarily represent the views of the Santa Fe Institute. We accept papers intended for publication in peer-reviewed journals or proceedings volumes, but not papers that have already appeared in print. Except for papers by our external faculty, papers must be based on work done at SFI, inspired by an invited visit to or collaboration at SFI, or funded by an SFI grant.

©NOTICE: This working paper is included by permission of the contributing author(s) as a means to ensure timely distribution of the scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the author(s). It is understood that all persons copying this information will adhere to the terms and constraints invoked by each author's copyright. These works may be reposted only with the explicit permission of the copyright holder.

www.santafe.edu

SANTA FE INSTITUTE

Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Manuscript Category

OXFORD

Subject Section

Similarity-Based Segmentation of Multi-Dimensional Signals

Rainer Machné[1,6]*, Douglas B. Murray[2] and Peter F. Stadler[3-8]

[1]Institute for Synthetic Microbiology, Cluster of Excellence on Plant Sciences (CEPLAS), Heinrich Heine University Düsseldorf, Universitätsstraße 1, D-40225 Düsseldorf, Germany
[2]Institute for Advanced Biosciences, Keio University, Tsuruoka, Yamagata 997-0017, Japan
[3]Bioinformatics Group, Deptartment of Computer Science, Interdisciplinary Center for Bioinformatics, German Centre for Integrative Biodiversity Research (iDiv) Halle-Jena-Leipzig, Competence Center for Scalable Data Services and Solutions, and Leipzig Research Center for Civilization Diseases, University Leipzig, Härtelstrasse 16-18, D-04107 Leipzig, Germany
[4]Max Planck Institute for Mathematics in the Sciences, Inselstraße 22, D-04103 Leipzig, Germany
[5]Fraunhofer Institute for Cell Therapy and Immunology, Perlickstrasse 1, D-04103 Leipzig, Germany
[6]Department of Theoretical Chemistry of the University of Vienna, Währingerstrasse 17, A-1090 Vienna
[7]Center for RNA in Technology and Health, Univ. Copenhagen, Grønnegårdsvej 3, Frederiksberg C, Denmark
[8]Santa Fe Institute, 1399 Hyde Park Road, Santa Fe NM 87501, USA

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: The segmentation of time series and genomic data is a common problem in computational biology. With increasingly complex measurement procedures individual data points are often not just numbers or simple vectors in which all components are of the same kind. Analysis methods that capitalize on slopes in a single real-valued data track or that make explicit use of the vectorial nature of the data are not applicable in such scenaria.
Results: We develop here a framework for segmentation in arbitrary data domains that only requires a minimal notion of similarity. Using unsupervised clustering of (a sample of) the input yields an approximate segementation algorithm that is efficient enough for genome-wide applications. As a showcase application we segment a time series of transcriptome sequencing data from budding yeast, using a similarity measure focussing on relative expression profile across the metabolic cycle rather then coverage per time point.
Availability: The software is available as an R package from `https://github.com/raim/segmenTier`.
Contact: raim@tbi.univie.ac.at
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

The segmentation problem consists in finding a piece-wise constant approximation to a function defined on a 1-dimensional independent variable, which may be e.g. time or a genomic coordinate. The problem arose early in the -omic era in the context of detecting copy number variations in array comparative genomic hybridization (CGH) data and for determing transcripts from RNA expression data measured by tiling arrays. For a 1-dimensional readout, coverage, it can be solved by dynamic programming (Picard *et al.*, 2005; Huber *et al.*, 2006). With the rapidly increasing flood of high throughput data the problem was generalized to segmenting vector-valued data such as multiple transcriptomes or CGH data from an entire cohort of patients. Again a dynamic programming solution can be found (Picard *et al.*, 2011).

In many applications one is not only interested in a segmentation but at the same attempts to infer clusters of measurements that behave in similar way. This problem has typically been tackled by combining an expectation-maximization (EM) step to train a model that is then used to

© The Author 2017. Published by Oxford University Press. All rights reserved. For permissions, please e-mail: journals.permissions@oup.com

produce the segmentation. A hybrid of EM and the dynamic programing algorithm was used for multidimensional CGH data (Picard *et al.*, 2011), in a transcriptomics context hidden semi-Markov models were used to deal with the directional nature of strand-specific transcriptome data (Du *et al.*, 2014).

A particularly important version of the problem is chromatin segmentation, i.e., the assignment of biologically interpretable functional annotations based on a pattern of histone modifications typically measured by ChIP-seq. `ChromaSig` (Hon *et al.*, 2008) then uses a greedy algorithm to extend seed patterns to determine segments. `ChromHMM` (Ernst & Kellis, 2010, 2012) and `EpiCSeg` (Mammana & Chung, 2015) are based on a hidden Markov models (HMM), `Segway` (Hoffman *et al.*, 2012) uses Dynamic Bayesian Network, and bidirectional HMMs are proposed in (Zacher *et al.*, 2014) to model the assignment of binarized (presence/absence) data or ChiP-Seq signal intensities or read counts to chromatin states. The models are trained on a subset of the data and then a Viterbi algorithm or its equivalent is then used to decode the complete data track, i.e. to produce a segmentation of the complete data set.

A closely related segmentation problem is the detection of differentially methylated regions (DMRs) from whole genome bisulfite sequencing data. While some tools use simple heuristics to aggregate differentially methylated CpGs into DMRs (e.g. `BS-seq` (Wu *et al.*, 2015), `MOABS` (Sun *et al.*, 2014), or `MethylSig` (Park *et al.*, 2014)). More recently, HMM based methods such as `HMM-Fisher` (Sun & Yu, 2016) have been introduced.

Another important variation on the theme are reference-based transcriptome assemblers such as `cufflinks` (Trapnell *et al.*, 2010), `TruHMM` (Li *et al.*, 2013), `ORA` (Sardu *et al.*, 2014), or `TransComb` (Liu *et al.*, 2016). These tools, however, do not solve a general segmentation problem. The HMM of `TruHMM`, for instance only separates expressed from non-expressed states. Similarly, the determination of intro-exon boundaries can be viewed as a particular, specialized segmentation problem. We refer to (Cleynen *et al.*, 2014) for a recent review and comparison.

HMMs or similar models are attractive theoretically because (i) such models inherently define segmentations, (ii) they provide a consistent statistical framework, and (iii) they are capable of computing measures of confidence for the results. The EM training, however, can be computationally rather expensive for large data sets. Furthermore, it requires a user-defined number of labels (such as chromatin states) and it depends on the vector structure of the data (such as position-wise counts or presence/absence information for each position and each transcriptome or ChIP-seq experiment. It is of interest therefore, to investigate whether the expensive model training can be replaced by computationally more efficient heuristics. This avenue has been explored e.g. in `metilene` (Jühling *et al.*, 2016), which uses a fast heuristic to propose differentially methylated regions that are tested for significance in a second step.

In this contribution we describe a general purpose segmentation method for essentially arbitrary linearly ordered data. It relies only on a similarity function and proceeds in three separate steps: (i) the measurements are clustered using an unsupervised clustering method so that each measurement is replaced by membership to one of the clusters or a "garbage bin" and (ii) cluster-cluster and/or position-cluster similarity measures are calculated, and (iii) the resulting coarse grained series of cluster IDs is segmented using an exact dynamic programming algorithm.

The data set that motivated the work presented here is a time-series of 24 RNA-seq transcriptomes covering about two and a half cycles of oscillatory growth dynamics ('metabolic cycle') in budding yeast (*Saccharomyces cerevisiae*) continuous culture. In this experimental system, a large majority of protein-coding transcripts showed dynamic (oscillatory) abundance signals in earlier hybridization-based transcriptome measurement ('microarrays') that are reproducible across a variety of strains and conditions (Klevecz *et al.*, 2004; Slavov *et al.*, 2011; Machné & Murray, 2012).

The budding yeast genome is compact, i.e densely packed with transcripts (Xu *et al.*, 2009) with a low frequency of spliceosomal introns (Spingola *et al.*, 1999), and expresses a large number of antisense and long non-coding RNAs (**?**Tisseur *et al.*, 2011; Marquardt *et al.*, 2011; Yamashita *et al.*, 2016) that often partially overlap other transcriptional units. This makes inference of yeast transcripts difficult for the standard tools, which are optimized for mammalian transcriptomes and use splice junctions as an import source of information. Moreover, during the metabolic oscillation the transcriptome is highly dynamic, so that different time points cannot be simply pooled before transcripts are characterized. Taken together, segmentation thus would seem to be a more prudent approach.

The correlation between consecutive time points and the usually well-defined phase of the expression changes w.r.t. to the metabolic cycle on the one hand, and the large variations of coverage between different loci of the same annotated transcript on the other hand, strongly suggest not to focus on the expression levels in individual experiments but on the position-wise *pattern* of the time-course for the purpose of segmentation. A clustering strategy for such oscillatory transcriptome data based on selected and scaled components of the discrete Fourier Transform (DFT) of the time-series was introduced for corresponding microarray data (Machné & Murray, 2012), where it was used to identify groups of genes with consistent temporal profiles. Here we employ the same technique to quantify similarities of temporal expression changes at single nucleotide resolution. This amounts to abandoning the vector space structure of the data in favor of a more general view on the segmentation problem that assumes a position-wise signal that "lives" in a similarity space, a mathematical construct that further generalizes the more familiar metric spaces.

In this contribution we first summarize the necessary theoretical background to formulate the segmentation problem in arbitrary similarity spaces. We then show that it can be solved by means of dynamic programming. We then focus on the application to yeast transcriptome time series to demonstrate the practical applicability of our approach.

2 Theory

We start from a signal $x : [1, ..., n] \to \mathbb{X}$, $i \mapsto x_i$ that is measured on a finite sequence of consecutive points, which without loss of generality we can label $i = 1, \ldots, n$. For the domain $\mathbb{X}$ of measured data we assume only that the similarity between two measurements can be quantified. That is, there exists a similarity function $\sigma : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ such that for all $u, v \in \mathbb{X}$ holds

(s0) $\sigma(u, u) \geq 0$;
(s1) $\sigma(u, v) = \sigma(v, u)$;
(s2) $\sigma(u, u) \geq \sigma(u, v)$;

The pair $(\mathbb{X}, \sigma)$ satisfying (s1) and (s2) is a *similarity space*. For completness of the presentation we note that two additional axioms,

(s3) $\sigma(u, v) + \sigma(v, w) \leq \sigma(u, w) + \sigma(v, v)$;
(s4) $\sigma(u, v) = \sigma(u, u) = \sigma(v, v)$ iff $u = v$.

are satisfied for all $u, v, w \in \mathbb{X}$. Axiom (s3) corresponds to the triangle inequality. If axiom (s4) holds in addition, there is an equivalent metric space with distance function $d_\sigma(u, v) = \sigma(u, u) - \sigma(u, v)$. We will require neither condition here since similarity functions satisfying (s1) and (s2) already introduce sufficient structure on $\mathbb{X}$ for well-defined clustering and classification algorithms, see e.g. (Jacobs *et al.*, 2000; Pekalska & Duin, 2002; Balcan *et al.*, 2008; Ackermann *et al.*, 2010; Chang *et al.*, 2016). The k-medoids strategy (Vinod, 1969; Kaufman &

Rousseeuw, 1987), for example, is a generalization of the well-known k-means clustering that only requires similarity data, see also (Juan & Vidal, 1994; Smellie, 2004) for more recent developments.

A *segmentation* $\mathfrak{S} := (i_1, \ldots, i_{\ell-1})$ with $1 = i_0 < i_1 < \cdots < i_\ell = n$ is a sequence of $\ell - 1$ *jump points* or *break points* that partitions $[1, n]$ into ℓ non-empty, consecutive intervals $I_h := [i_{h-1}, i_h]$ for $1 \leq h \leq \ell$. The coarsest segmentation, for $\ell = 1$, consists only of the interval $[1, n]$ itself, while the finest segmentation assigns each point to a distinct interval, $I_h = \{h\}$. A given segmentation $\mathfrak{S}$ defines sets $\mathcal{J}_h := \{x_i \in \mathcal{X} | i \in I_h\}$ of signals observed in the corresponding interval.

Good segmentations are distinguished from bad ones by a measure of the similarities of signal values within each set $\mathcal{J}_h$. Since $\mathcal{J}_h$ is uniquely determined by the interval I_h we can think of the scores as a functions $f(I_h)$ of the intervals. The quality of a particular segmentation $\mathfrak{S}$ is then $f(\mathfrak{S}) = \sum_{I_h \in \mathfrak{S}} f(I_h)$. We now can define the segmentation problem in one of the following ways: (i) We fix a number ℓ of desired segments, or (ii) we fix a cost M for each jump. In first case we ask for the segmentation with ℓ non-empty intervals that maximizes $f(\mathfrak{S})$. In the second case we allow ℓ to vary and penalize the use for more distinct intervals then necessary by maximizing $\sum_h f(I_h) - M(\ell - 1)$.

It is easy to see that both problems can be solved by essentially the same dynamic programming approach. Denote by $S_{i,h}$ the optimal segementation of the sub-problem on the points $[1, ..., i]$ that consists of exactly h non-empty intervals. For brevity we write $f_{i,j} := f([i, ..., j])$. These values satisfy the recursion

$$\begin{aligned} S_{k,1} &= f_{1,k} \\ S_{k,h} &= \max_{h-1 \leq j < k} S_{j,h-1} + f_{j,k} \end{aligned} \tag{1}$$

In the first case, the optimal solution is found as $S_{n,\ell}$, for the other variant, one needs to termine $\max_h S_{n,h} - (h-1)M$. An optimal sementation is obtained by backtracing from these entries in either case. Equ.(1) provides a polynomial time solution in $O(n^2\ell)$ or $O(n^3)$ time and $O(n\ell)$ or $O(n^2)$ space, assuming that the scores $f_{j,k}$ can be computed fast enough.

The structure of the cost function f is not arbitrary. Most importantly breaking up an interval into two parts must not make the fit worse, i.e., $f_{i,j} \leq f_{i,u-1} + f_{u,j}$ must hold for all $i < u \leq j$. This condition implies $f_{i,j} \leq \sum_{u=i}^{j} f_{u,u}$ and strongly suggests to consider scores of form

$$f(I_h) := \sum_{i \in I_h} \sigma(x_i, \mathcal{J}_h) \tag{2}$$

where $\sigma(x, \mathcal{J})$ measures the similarity of $x \in \mathbb{X}$ with a set $\mathcal{J} \subseteq \mathbb{X}$. In general similarity spaces there are at least two natural choices for $\sigma(x, \mathcal{J})$. One may represent the set $\mathcal{J}$ by a "representative point" $x_\mathcal{J} \in \mathcal{J}$ and define $\sigma(x, \mathcal{J}) := \sigma_m(x, x_\mathcal{J})$. The most natural choice of a representative object is the medoid, which has the property that the total similarity $\sum_{y \in \mathcal{J}} \sigma(y, x_\mathcal{J})$ is maximal. Alternatively, the average similarity $\sigma_a(x, \mathcal{J}) := (1/|\mathcal{J}|) \sum_{y \in \mathcal{J}} \sigma(x, y)$ can be used. The naïve evaluation of these measures is quadratic in the length of the interval, and hence would slow down the evaluation by another order of magnitude. More efficient evaluations are easily available, however, see Appendix A.

Even with an efficient evaluation of $f_{i,j}$ the performance bounds on equ.(1) are expensive for genome-wide applications. The basic idea to devise a much faster approximate solution is to replace the sets $\mathcal{J}_h$ of measurements within an interval with a precomputed clustering of the measurement comprising N clusters $\mathcal{C}_\alpha \subseteq \mathbb{X}$, $\alpha = 1, \ldots N$.

Denote by $S_{i,\alpha}$ the optimal score for segmentation of the interval $[1, ..., i]$ that assigns cluster $\mathcal{C}_\alpha$ to the last position i, and let $s(i, j, \alpha)$ denote a score that measures the quality of the matching of the measurements in the interval $[i, ..., j]$ to the cluster α. Again we write M for the penalty incurred by the use a new fragment. Then $S_{k,\alpha}$ satisfied the recursion

$$S_{k,\alpha} = \max_{j \leq k} \max_{\beta \neq \alpha} S_{j-1,\beta} + s(j, k, \alpha) - M \tag{3}$$

with the initialization (basis case) $S(0, \beta) = M$ for all β, ensuring that $S_{i,\alpha} \geq s(1, i, \alpha)$. This can be evaluated in $O(n^2N^2)$ time and (nN) space. The the optimial segmentation is then obtained by standard backtracing from the entry realizing the optimal score $\max_\alpha S_{n,\alpha}$.

Several scoring functions $s(i, j, \alpha)$ can be conceived. On the one hand, one might want to match the original segmentation problem as closely as possible. This amounts to using a score of the form

$$s(i, k, \alpha) = \sum_{j=i}^{k} \tilde{\sigma}(x_j, \mathcal{C}_\alpha) \tag{4}$$

where $\tilde{\sigma}$ is either the average similarity between x_j and the cluster $\mathcal{C}_\alpha$ (corresponding to σ_a in the original problem) or the similarity between x_j and the medoid point of $\mathcal{C}_\alpha$ corresponding to the σ_m measure. The latter is particularly appealing if e.g. a k-medoid method was used for clustering. If a divisive clustering method was used, then usually no natural representative is available, and the average similarity score may be the natural choice. Of course, if σ is derived from a vector space, i.e., centroids $\bar{x}_\alpha := (1\|\mathcal{C}_\alpha|) \sum_{x \in \mathcal{C}_\alpha} x$ are well defined, these are the natural replacement for medoids.

Alternatively, we can determine the cluster $\mathcal{C}_i$ fits best to x_i, i.e., $\mathcal{C}_i = \arg\max_\alpha \sigma(x_i, \mathcal{C}_\alpha)$. It is not necessary to assign all measurements to a cluster. Instead, we may place a measurements x_i into a "garbage bin" or "nuissance cluster" $\mathcal{C}_0$ if $\sigma(x_i, \mathcal{C}_\alpha) < \theta$ for all regular clusters, i.e., $1 \leq \alpha \leq N$ and some threshold similarity value θ. In this setting, the scoring function is naturally constructed as

$$s(i, j, \mathcal{C}) = \sum_{k=i}^{j} Q(\mathcal{C}_k, \mathcal{C}) \tag{5}$$

where $Q(\mathcal{C}, \mathcal{D})$ is an, in principle arbitrary, similarity measure for the two clusters. The most basic choice is $Q(\mathcal{C}, \mathcal{C}) = 1$ and $Q(\mathcal{C}, \mathcal{D}) = a < 0$ for $\mathcal{C} \neq \mathcal{D}$. For the specific application scenario of transcriptome time series we found similarity measures derived from the Pearson correlation of the expression time series particularly useful (see Methods for details).

There are several possibilities to treat scattered outliers or special cases such a non-expressed locations. One approach is to use a smaller penalty value M_0 instead of M for jumps to the "nuissance cluster" to allow such values to accumulate in short 'special' segments. Another plausible parametrization for a "nuissance cluster" is to simply set $s(i, j, 0) = M$, compensating the penalty M for the jump to $\mathcal{C}_0$ thus adding no contribute to the similarity score at all for the nuisance cluster.

To make the paramters easier to interpret, it can be convenient to use a scaled version so that $\tilde{\sigma}$ takes values between 0 and 1. The jump penalty parameter M then serves as a lower bound on the length of a segment.

We observe, finally, that with any of the choice of the scoring functions discussed above, it not necessary to store or precompute all values of $s(i, j, \mathcal{C})$. For every real cluster and $i > 1$ we have

$$s(i, j, \alpha) = s(1, j, \alpha) - s(1, i - 1, \alpha) \tag{6}$$

Hence if suffices to compute the $O(nN)$ entries of $s(1, j, \alpha)$ for all j and α.

The quadratic running time suggests to employ a heuristic pre-segmentation step that reduces the problem to smaller pieces at positions where break points are obvious such as longer genomic regions without expression or histone modification signals. Alternatively, the running time can be reduced by cutting the input into overlapping pieces with a size a few times larger than some upper bound on the expected segement size.

3 Methods

Implementation. The algorithm was implemented in R, where the workhorse scoring routines are written in `C++` *via* `Rcpp` (Eddelbuettel & François, 2011), and is available as an R library from github (`https://github.com/raim/segmenTier`). The state of the code used herein is tagged as release `v0.1`. The library additionally implements the time-series preprocessing (data transformations, including the DFT) and clustering strategies used in this presentation. The implementation is specifically equipped for batch calculations over different input clusterings and parameter settings, to scan for optimal parameters for the given segmentation objective.

The package is geared towards use from command-line scripts in parallel computing environment, and these command-line scripts are available from the *segmenTools* code repository and R library (`https://github.com/raim/segmenTools`). This library provides all functionality used for the comparative evaluation of segmentation characteristics and quality.

Scoring Functions. The simple scoring $Q(\mathcal{C},\mathcal{D}) = 1$ or a depending on whether $\mathcal{C}$ and $\mathcal{D}$ are equal or distance is available as `ccls` scoring in `segmentier` with default value $a = -1$.

For vector-like data such as the time series of yeast transcriptomes that motivated this work, the Pearson correlation coefficient $\mathrm{corr}(\bar{x}_\mathcal{C}, \bar{x}_\mathcal{D})$ is a natural choice of the similarity measure. As it turned out, a transformation that reduced the influence of moderate positive or negative correlations improved the segmentation performance. To this end we used transformation $\varphi_\epsilon : [-1,+1] \to [-1,+1] : t \mapsto \mathrm{sign}(t)t^\epsilon$. Note that φ_1 is the identity. For large ϵ the transformation approaches $\varphi_\infty(-1) = -1, \varphi_\infty(1) = 1$, and $\varphi_\infty(t) = 0$ for $-1 < t < 1$.

`segmenTier` implements both $\sigma(x_i,\mathcal{C}_\alpha) = \varphi_\epsilon(\mathrm{corr}(x_i, \bar{x}_{\mathcal{C}_\alpha}))$ and $Q(\mathcal{C},\mathcal{D}) = \varphi_\epsilon(\mathrm{corr}(\mathcal{C},\mathcal{D}))$ as options `icor` and `ccor`. The transformation parameter ϵ is available as parameter `E`.

Nuissance positions x_0 or clusters $\mathcal{C}_0$ are assigned a fixed correlation coefficient ν, with $\sigma(x_0,\mathcal{C}) = \sigma(\mathcal{C}_0,\mathcal{C}) = -\nu$ and $\sigma(x_0,\mathcal{C}_0) = \sigma(\mathcal{C}_0,\mathcal{C}_0) = \nu$. For our test data $\nu > 1$ proved useful to enforce breaks between close adjacent transcripts and cut ends at the transition to noisy low-expressed regions (parameter `nui` in `segmenTier`).

Finally, the length-penalty M and the nuissance-specific M_0 are available as parameters `M` und `Mn` in `segmenTier`.

Two package demos demonstrate the usage of the package and effects of parameters. The first (`segment_test`) uses an artifical data set to demonstrate the low level interface to the algorithm, function `segmentClusters`, and how to construct similarity matrices for scoring functions `icor` and `ccor`. It further demonstrates that variations in scoring matrix calculation and back-tracing affect only the discrete scoring function `ccls` (see below & Suppl. Fig. S1). The second (`segment_data`) demonstrate usage of higher-level interfaces and S3 data structures for processing of time-series, including simple wrappers for time-series transformation, clustering, segmentation and plotting.

Data Acquisition. The yeast strain used in this study was IFO 0233 and continuous culture experiments were carried out as previously described (Murray *et al.*, 2007). Total RNA was extracted (Sasidharan *et al.*, 2012) from 24 samples taken every 4 min, covering ca. 2.5 cycles of the respiratory oscillation (Suppl. Fig. S2), and DNA removed (RNase-Free DNase Set, Qiagen, Japan). Total RNA had an RNA integrity number >7 and 260nm:230nm and 260nm:230nm ratios >2.14. Strand specific cDNA libraries were created using the dUTP method (Parkhomchuk *et al.*, 2009; **?**) and sequencing was carried out on an Illumina 1G sequencer. All cDNA libraries were generated and sequenced by BGI, China.

RNA-seq reads were mapped against yeast S288C genome (release R64-1-1) using `segemehl` (version 0.1.4) (Hoffmann *et al.*, 2014) with default parameters and spliced read mapping enabled. Initially unmatched reads were mapped again using the remapping tool from the `segemehl` package and the resulting files were merged. Coverage (read-counts per nucleotide) was normalized for total library size and was stored in a bedgraph file for further analysis.

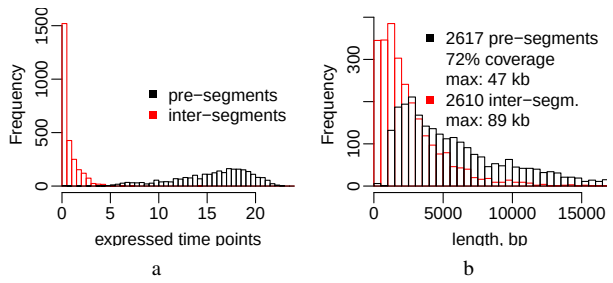


Fig. 1. Length distributions of the pre-segmentation. Distribution of mean number of expressed time points, read-count presence $\bar{n} = (i-j+1)^{-1}\sum_j^i n_k$, (a) and lengths (b) of genomic intervals of the final pre-segments (black) and inter-segment regions (red).

Pre-Segmentation. The algorithm performance was improved by using areas largely devoid of RNA-seq coverage as "safe anchors" for pre-segmentation. To this end we calculated for each genomic position i the number n_i of time-points with any signal at all, i.e., $x_i \neq \vec{o}$. A moving average $\tilde{n}_i/d$ of this signal, scaled by the total number of measured time points d (here 24), is then used to determine non-expressed region. For the yeast data we used a windowsize of `avg`$= 1000$ nt for the running average, and required an interval of length 1000 nt where the moving averages does not exceed the threshold $\tilde{n}_i/d \leq 1/3$ (achieved by option `minrd`$= 8$, indicating 8 of 24 time points) for the average number of nucleotides with any expression signal at all. To avoid the removal of weakly expressed regions close to non-expressed intervals, we extended pre-segments on both ends until the $\tilde{n}_i/d$ signal, computed with a much narrower moving average over only `favg`$= 100$ nt, dropped to 0. Finally, adjacent pre-segments with a length of less than `minsg`$= 1000$ nt were fused with their neighbors, and segments spanning chromosome ends were split there. The pre-segmentation heuristics is implemented in the R script wrapper `presegment.R` and distributed with the `segmenTools` package, and the indicated parameters for the pre-segmentation procedure can be set by the user.

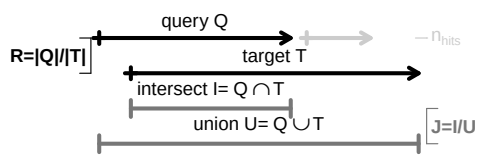
Application of the pre-segementation heuristic to the yeast data set produced 2617 pre-segments between 1 and 47 kb, and in total covering 72% of the genome (Fig. 1). The majority of 2610 inter-segment regions is small; the largest (89 kb) corresponds to the mitochondrial genome and was skipped due to the specific complex structure (large introns, overlapping ORF) of the mitochondrial transcriptome. Both the expressed pre-segments and the inter-segments, potentially containing lowly expressed smaller transcripts, were then used as input for the segmentation procedure. This allowed the efficient computation of the final segmentation using the DP approach described above.

Time-Series Processing & Nuissance Cluster. The time-series of read-counts for each nucleotide were processed as previously described for micro-array based data (Machné & Murray, 2012) with data- and problem-specific adaptations. The following processing pipeline is implemented in `segmenTier` as function `processTimeseries`. In short, a Discrete Fourier Transform was applied to the raw (`trafo=raw`) or `arcsinh`-transformed (`trafo=ash`) time-series, and all components were amplitude-scaled (`use.snr=TRUE`) as described (Machné & Murray, 2012). High-frequency components were discarded and only the informative components 1–7 (`dft.range=1:7`) were selected for clustering. Components 2–7 carry the main information of the time-series

profiles. Unlike previously, we included the first DFT component, which provides information on the temporal mean of expression. This allowed for a better separation of very close adjacent segments with otherwise similar temporal profiles. However, to avoid segmentation by the mean expression alone, this component was separately scaled by the `arcsinh` transformation (`dc.trafo=ash`). All non-expressed positions where skipped and automatically assigned to the specially treated nuissance cluster.

The transformed data was then clustered by the base R implementation of k-means using the `Hartigan-Wong` algorithm (Hartigan & Wong, 1979), *via* the package's `clusterTimeseries` function. This function also calculates the similarity matrices required for the scoring functions, specifically Pearson correlation coefficients between cluster means for scoring function `ccor` or between each position and cluster means for scoring function `icor`. For the latter we implemented the algorithm in `C++` (*via* `Rcpp`) for efficiency. After clustering another filtering step proved useful for scoring function `ccor`. Positions with a correlation coefficient below a threshold θ (`nui.thresh`=0.6) to their cluster center where re-assigned to the nuissance cluster.

Parameter Scan & Optimization. In order to test the quality of the segementation, we used a set of 5171 experimentally defined transcripts that were assigned to open reading frames (ORF-T) (Xu *et al.*, 2009), as available from the yeast genome database (SGD) (file `Xu_2009_ORF-Ts_V64.bed`). With our segmentation as the query Q and annotated ORF-T as the target T, we calculated three parameters for each target segment, graphically defined as follows:



(7)

For each target segment we counted all overlapping query segments (n_{hits}, then selected the query segment with the highest Jaccard index $J = |Q \cap T|/|Q \cup T|$ and calculated the query/target length ratio $R = |Q|/|T|$. To characterize a segmentation globally, we used the fraction R_{short} of segments with $R \leq 0.8$ and R_{long} with $R \leq 1.2$, the total Jaccard index J_{tot} of all best hits queries for all targets, and $\tilde{n}_{hits}$, the average number of query hits per target. A segmentation that recovers the test set well, will have high J_{tot} and R_{long}, and low $\tilde{n}_{hits}$ and R_{short}.

We tested 540 parameter combinations against the ORF-T test set, varying time-series pre-processing, clustering and segmentation parameters (Fig. 2). The segmentations were clustered by the four optimization parameters (coloring in Fig. 2) using 'partitioning around medoids' (PAM). Enrichment of these clusters with run parameters was established by Fisher's exact tests (Fig. 2c) and provides an overview of the effects of parameters. A detailed inspection of parameter combinations in the optimal clusters finally guided the choice of three optimal and two bad-scoring parameter sets for discussion (Fig. 2d and Fig. 3).

4 Results and Discussion

We have shown here that segmentation along a linear coordinate axis can be modelled as a combinatorial optimization problem independently of the data type or types. Indeed, the only mathematical structure required in the data domain is a symmetric similarity measure that favours identity, i.e., satisfies conditions (s1) and (s2). An efficient implementation is made available with the R package `segmenTier`.

As a showcase application we consider the segmentation of a time-series of 24 yeast transcriptomes. Using a pre-segmentation at large

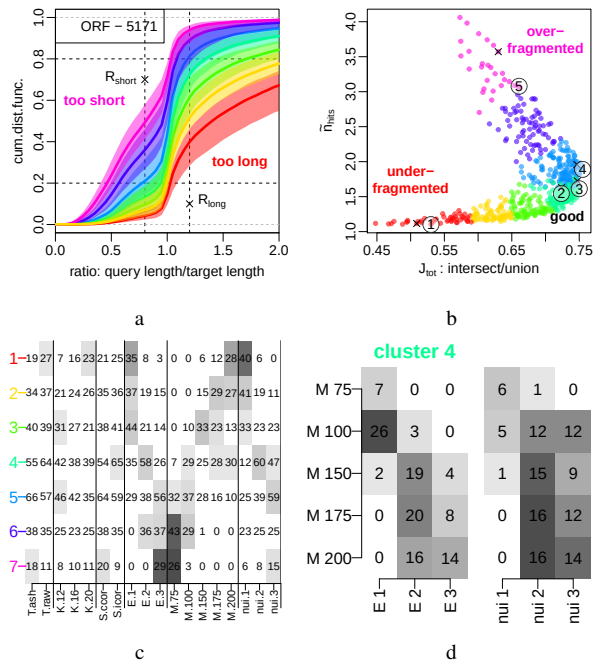


Fig. 2. Parameter Scan & Selection: ORF Transcript Recovery. 540 segmentations were calculated with varied parameters. Segments (queries) from each segmentations were tested against a set of annotated ORF transcripts. Only the best matching pairs for each target were used. (a) Empirical cumulative distribution function of query-target length ratios R. A segmentation where many query segments that are longer ($R > 1$) then their matching ORF transcript target is interpreted as under-fragmentation of the data (e.g. the 'too long' red segmentations), while segments which are shorter then their target ($R < 1$) point to over-fragmentation (e.g. the 'too short' magenta segmentations. (b) The total Jaccard index J_{tot} of best-matching pairs (x-axis) and $\tilde{n}_{hits}$, the number of fragment hits per target (y-axis). Numbers indicate the example segmentations in Fig. 3, the slightly shifted 'x' indicate the same parameters but with scoring function `ccor` instead of `icor`. The grouping and coloring of segmentations is derived from a PAM clustering of the ratios R_{short} and R_{long}, and J_{tot} and $\tilde{n}_{hits}$. (c) Frequency distribution of parameters (T: time-series processing, K: number of clusters, S: scoring function, E: similarity transformation exponent ϵ, M: length-penalty parameter M, nui: nuissance cluster correlation ν) in the PAM clusters; the gray-scale is derived from the p-values ($-log2(p)$) of Fisher's exact tests of the overlaps and indicates enrichment. This provides an overview of parameter effects on optimization criteria. Detailed effects for each parameters are shown in Suppl. Fig. S3a. (d) Frequency distributions of parameter combinations in the 'optimal' cluster 4, gray-scale is derived from the shown frequencies; see Suppl. Fig. S3b for other clusters.

non-expressed regions (Fig. 1) `segmenTier` computes segmentation of the complete yeast genome data set in about 15 hours on a desktop computer[1]. This performance made is possible to explore the parameter space systematically.

We systematically scanned 540 combinations and tested the resulting segments for recovery of a set of 5171 published transcripts (Xu *et al.*, 2009) annotated as Open Reading Frame transcripts (ORF-T) (Fig. 2). Over-fragmentation is observed when segments are shorter then target ORF-T and a given ORF-T comprises several segments, and under-fragmentation when adjacent segments are not split (Fig. 2a & 2b).

The test revealed the exponent ϵ, the nuissance cluster similarity ν and the length-penalty score M as major determinants of ORF-T recovery performance. Notably, a good recovery of ORF-T can be achieved with different opposing combinations, combining either higher $M \geq 150$ with $\epsilon > 1$ and $\nu > 1$, or low $M < 150$ and $\epsilon = \nu = 1$ (Fig. 2c & Fig. 2d). We also observed small but systematic effects of the of choice of data pre-processing, the number of clusters K, and scoring function S (Suppl.

[1] Lenovo ThinkCenter, Intel i7-4790 CPU (3.60 GHz), Fedora F22

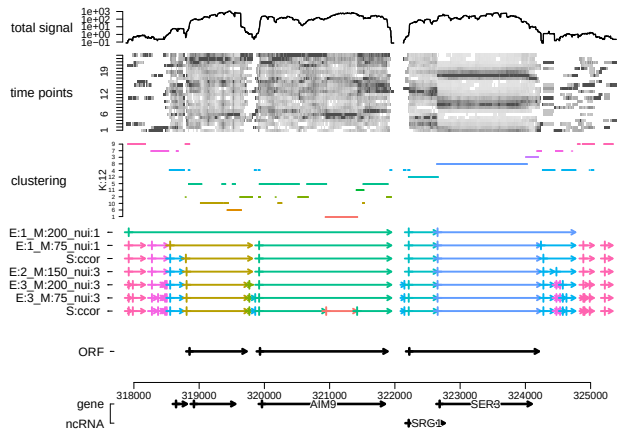


Fig. 3. Example Region: SRG1 vs. SER3 (chrV:317829..325452). All segmentations were calculated from the shown clustering (K=12, clusters sorted by similarity) of selcted DFT components of the raw read-count time-series (see Methods), with the indicated parameters `E`, `M` and `nui`, and with scoring function `icor` (corresponding from top to bottom to 1–6 in Fig. 2b), except where indicated ('S:ccor'), which were caculated with the same parameters as the segments directly above but using scoring function `ccor` ('x' in Fig. 2b). 'ORF' are annotated transcripts from the ORF-T test set (Xu et al., 2009), 'gene' and 'ncRNA' are annotations from the yeast genome release R64.1.1 of the actual ORF, from left to right YER078W-A, YER079W, AIM9 and SER3, and the ncRNA SRG1.

Fig. S3a) which contributed to our final choice of parameter sets used for further discussion and indicated in Figure 2b.

The detailed effects of parameters can be illustrated on specific examples: a small region from chromosome V comprises of four ORF and one non-coding RNA (Fig. 3 & S4), and a broader genomic region (Fig. 4). The non-coding RNA SRG1 regulates the transcription of the downstream overlapping SER3 ORF transcript via a transcription-interference mechanism (Martens *et al.*, 2004).

M allows to enforce minimal length requirements for the expected signal, here RNA transcripts. However, unlike most eukaryotic genomes the yeast genome is highly compact and densely packed with transcripts (Fig. 4). We often observe a gradient of signal towards the end of transcriptional units, indicating the distribution of actual transcription start and end sites or potential read-through events. Thus, the signal between distinct transcriptional units often remains above 0 and often will reflect contributions from both transcripts. For example the three transcripts of genes YER078W-A, YER079W, AIM9, upstream of the SRG1, remain un-segmented with parameters $\epsilon = \nu = 1$ at $M = 200$ (Fig. 3 &

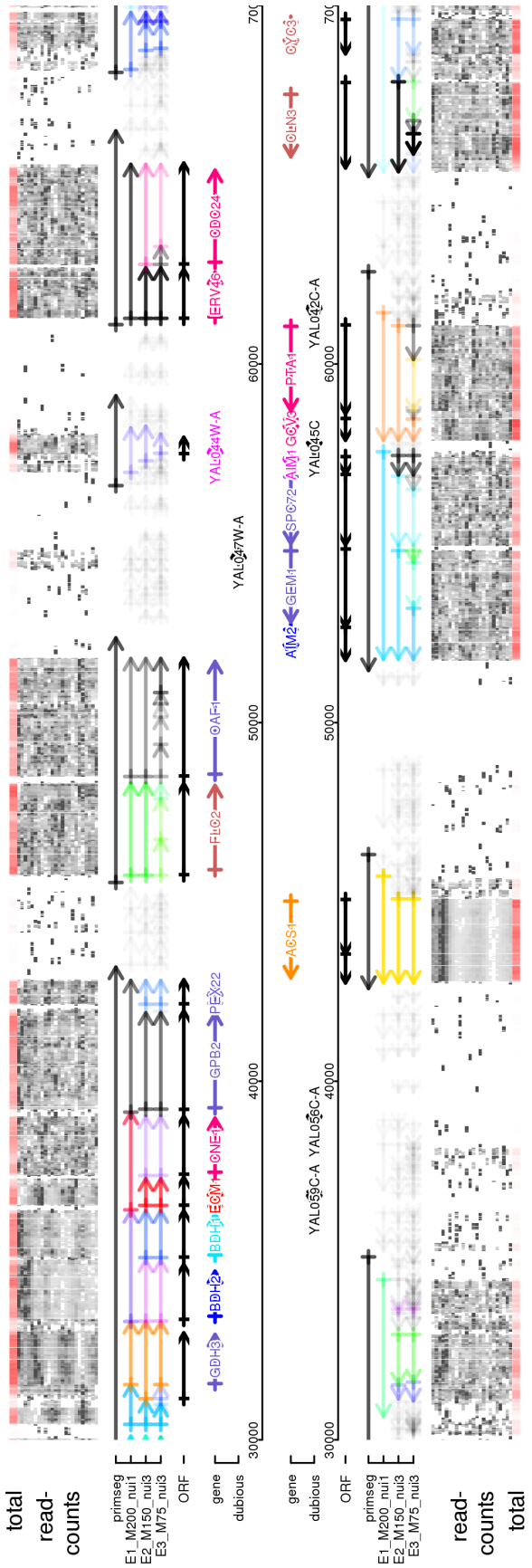


Fig. 4. Example Domain (chrI:300000..70000). A densely transcribed domain from chromosome I is shown, transcribed segments are shown separately and in reverse order for the forward (top) and reverse (bottom) strand. 'total' is a color-coded version of the total signal (sum over all time points), 'read-counts' are the individual time-points, 'primseg' are the primary segments which where used to split segmentation (Fig. 1), the shown segmentations are from top to bottom those indicated as 1, 3 and 5 in Fig. 2b. Segments were colored by calculating a cycle phase from the average segment read-count along a color wheel (Suppl. Fig. S5), and color alpha was derived from a p-value ($-\log_2(p)$) for oscillation calculated by the `rain` R package (Thaben & Westermark, 2014). Segments below $p < 0.05$ are shown in black, where the alpha is derived from their total expression level. 'ORF' are the experimentally defined ORF-T transcripts from (Xu et al., 2009) which were used in the parameter scan (Fig. 2), 'gene' and 'dubious' are annotated confirmed and dubious open reading frames in the yeast genome release R64.1.1. The 'gene' ORF are colored by a previous consensus definition of rhythmically co-transcribed gene cohorts in this experimental system (Machné & Murray, 2012). The plot was generated using the `genomeBrowser` R script collection at `https://gitlab.com/raim/genomeBrowser`.

S4a). Amplifying or scaling of the cluster-position and nuissance similarity measures by ϵ and ν allows to locally overcome length requirements set by M and cut-off small and mostly noisy segments between adjacent and at the end of transcripts. Several such examples can be observed in Figure 4.

In contrast, the overlapping transcriptional units of SRG1 ncRNA and SER3 ORF transcript both have a strong and distinct signal. Their detection is stable over the parameter range tested (Fig. 3). It is notable that this pair was not identified as separate transcripts in the ORF-T data set. The genome contains several known and expectably many more unknown examples of such overlapping transcripts ineracting by transcriptional interference mechanisms, and potentially un-detected in our ORF-T set. Such artefacts may in part explain why the total Jaccard measure J_{tot} remains well below its maximum (Fig. 2b).

Setting a lower M, combined with similarity scaling often leads to internal breaks that appear artefactual but do usually represent subtly distinct temporal profiles, e.g., the central portion of gene AIM9 is split at $\epsilon = \nu = 3$ and $M = 75$ with scoring function `ccor` (Fig. 3, bottom segmentation). It is for the present analysis and without further experimental tests impossible to judge all individual break points, several experimental and data processing sources of noise can bias local signals. But it is well understood that "cryptic" transcription can occur inside of and overlapping with more traditionally defined transcriptional units (Xu *et al.*, 2009; Yassour *et al.*, 2010). Optimal parameter settings will highly depend on the data structure and the purpose of analysis. Parameters could be geared to higher break sensitivity to analyse more subtle signals, e.g. from splice sites, or merely pinpoint methodological artefacts.

User-defined choices of parameters cannot be entirely avoided in segmentation problems. This is in particular true in application where the structure of the data domain is either unknown or deliberately modelled with little internal detail. In most cases the similarity measure will be defined heuristically based on the user's intuition about the data or based on a prior empirical exploration. For our show-case example, oscillatory transcriptome dynamics, we had previously observed that the phase and amplitude information describes the data better than the more obvious representation of the time-series as vectors in a euclidean space (Machné & Murray, 2012; Lehmann *et al.*, 2013). We note that in the context of the segmentation problem, it proved useful to additionally account for the total coverage to achieve better segmentation between close adjacent fragments. This seems to reflect the tendency of adjacent genes to show correlated expression patterns (Tsai *et al.*, 2007).

The *segmenTools* package therefore has been designed to make it easy to scan and evaluate the effects of parameters. It also implements optimization strategies for the parameters to be used in applications where an annotated set of test data is available.

The clustering techniques available for similarity spaces typically contain a random component, such as the choice of the initial medoids in the k-medoids approach. The efficiency of `segmenTier` makes is possible to compute multiple segmentations so that the robustness of the segmentation can be determined. This also suggests to investigate, in future research, systematic methods to define and compute consensus segmentations. A simple majority voting over several runs with identical parameters but exploiting the random effect of k-means clustering, may already yield good results (Suppl. Fig. S4b).

5 Concluding Remarks

The general 1D segmentation method outlined here poses several questions for future research. Although we have shown that the results are quite robust over a range of parameter values for data as complex as the yeast transcriptome, some parameters choice have to be left to user as matter of design: most importantly, the user has to supply an estimate of the typical segement size or an estimate of the noise level. Without any prior knowledge, these cannot be estimated from the data as a matter of principle. To see this, suppose that there is no noise at in the data. Then every jump in the data is real and segments are interval of exactly constant signal. In the other extreme, the input might be entirely noise and the correct answer is a single segment.

The second key parameter is the number of clusters K that are to be used. As we have seen, the segmentation problem can be phrased without a clustering step. For most applications, however, this is beyond the limits of computational resources. We have seen above that K only has a moderate influence on the results. If K is chosen much too large, this mostly infringes on the computational efficiency. Too few clusters, on the other hand, may lead to incomplete segementation. Classical information-theoretic methods (Sugar & James, 2003) or the silhouettes method (Rousseuw, 1987) are certainly applicable to estimate good values of K, or even assess the quality of the re-clustering of the data set by the segmentation. It remains a question for future research, however, whether values of K optimal in terms of clustering also yield the best segmentation results.

An important issue is the robustness of the segmentation. As the most simple test we have used independent instances of k-means clustering to assess this issue. One might also consider bootstrap-like resampling. The construction of multiple segmentations for the same data set, either due to randomized components in the clustering algorithm (here the random choice of the K initial centroids), or due to variations in the parameters, naturally leads to the question how segmentations should be compared and how to define a consensus set of segmentations. In particular the latter appears to be an interesting problem for future research that is not only of relevance for the future releases of `segmenTier` but also for problems such as the reconciliation of multiple chromatin segmentations.

Funding

DBM was funded by a partnering award from Japan Science and Technology Agency, Yamagata prefectural government and the City of Tsuruoka. RM was funded by the *Deutsche Forschungsgemeinschaft*, grants AX 84/4-1 and EXC 1028.

Acknowledgements

Computations were performed at Interdisciplinary Center for Bioinformatics at the Leipzig University and at the "Centre for Information and Media Technology" (ZIM) at the University of Düsseldorf (Germany). We thank Stephan Bernhart for support with genome mapping of the raw sequencing data.

References

Ackermann, M. R., Blömer, J. & Sohler, C. (2010) Clustering for metric and nonmetric distance measures. *ACM Trans Algorithms,* **6**, 59.

Balcan, M.-F., Blum, A. & Vempala, S. (2008) A discriminative framework for clustering via similarity functions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing (STOC '08)* pp. 671–680 ACM, New York.

Chang, C.-S., Liao, W., Chen, Y.-S. & Liou, L.-H. (2016) A mathematical theory for clustering in metric spaces. *IEEE Trans Network Sci Eng,* **3**, 2–16.

Cleynen, A., Dudoit, S. & Robin, S. (2014) Comparing segmentation methods for genome annotation based on RNA-Seq data. *JABES,* **19**, 101–118.

Du, Y., Murani, E., Ponsuksili, S. & Wimmers, K. (2014) biomvRhsmm: genomic segmentation with hidden semi-Markov model. *Biomed Res Int.,* **2014**, 910390.

Eddelbuettel, D. & François, R. (2011) Rcpp: seamless R and C++ integration. *Journal of Statistical Software,* **40** (8), 1–18.

Ernst, J. & Kellis, M. (2010) Discovery and characterization of chromatin states for systematic annotation of the human genome. *Nat Biotechnol,* **28**, 817–825.

Ernst, J. & Kellis, M. (2012) ChromHMM: automating chromatin-state discovery and characterization. *Nat Methods,* **9**, 215–216.

Hartigan, J. A. & Wong, M. A. (1979) Algorithm as 136: a k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics),* **28** (1), 100–108.

Hoffman, M. M., Buske, O. J., Wang, J., Weng, Z., Bilmes, J. A. & Noble, W. S. (2012) Unsupervised pattern discovery in human chromatin structure through genomic segmentation. *Nature Methods,* **9**, 473–476.

Hoffmann, S., Otto, C., Doose, G., Tanzer, A., Langenberger, D., Christ, S., Kunz, M., Holdt, L. M., Teupser, D., Hackermüller, J. & Stadler, P. F. (2014) A multi-split mapping algorithm for circular RNA, splicing, trans-splicing, and fusion detection. *Genome Biology,* **15**, R34.

Hon, G., Ren, B. & Wang, W. (2008) ChromaSig: a probabilistic approach to finding common chromatin signatures in the human genome. *PLoS Comput Biol,* **4**, e1000201.

Huber, W., Toedling, J. & Steinmetz, L. M. (2006) Transcript mapping with high-density oligonucleotide tiling arrays. *Bioinformatics,* **22**, 1963–1970.

Jacobs, D. W., Weinshall, D. & Gdalyahu, Y. (2000) Classification with nonmetric distances: image retrieval and class representation. *IEEE Trans Pattern Anal Machine Intel,* **22**, 583–560.

Juan, A. & Vidal, E. (1994) Fast k-means-like clustering in metric spaces. *Pattern Recognition Letters,* **15**, 19–25.

Jühling, F., Kretzmer, H., Bernhart, S. H., Otto, C., Stadler, P. F. & Hoffmann, S. (2016) `metilene`: fast and sensitive calling of differentially methylated regions from bisulfite sequencing data. *Genome Res.,* **26**, 256–262.

Kaufman, L. & Rousseeuw, P. J. (1987) Clustering by means of medoids. In *Statistical Data Analysis Based on the L_1-Norm and Related Methods*, (Dodge, Y., ed.),. North-Holland pp. 405–416.

Klevecz, R. R., Bolen, J., Forrest, G. & Murray, D. B. (2004) A genomewide oscillation in transcription gates DNA replication and cell cycle. *Proc Natl Acad Sci U S A,* **101** (5), 1200–5.

Lehmann, R., Machné, R., Georg, J., Benary, M., Axmann, I. M. & Steuer, R. (2013) How cyanobacteria pose new problems to old methods: challenges in microarray time series analysis. *BMC Bioinformatics,* **14** (1), 133.

Levin, J., Yassour, M., Adiconis, X., Nusbaum, C., Thompson, D., Friedman, N., Gnirke, A. & Regev, A. (2010) Comprehensive comparative analysis of strand-specific RNA sequencing methods. *Nat Methods,* **7** (9), 709–715.

Li, S., Dong, X. & Su, Z. (2013) Directional RNA-seq reveals highly complex condition-dependent transcriptomes in *E. coli* K12 through accurate full-length transcripts assembling. *BMC Genomics,* **14**, 520.

Liu, J., Yu, T., Jiang, T. & Li, G. (2016) TransComb: genome-guided transcriptome assembly via combing junctions in splicing graphs. *Genome Biol,* **17**, 213.

Machné, R. & Murray, D. (2012) The yin and yang of yeast transcription: elements of a global feedback system between metabolism and chromatin. *PLoS One,* **7** (6), e37906.

Mammana, A. & Chung, H.-R. (2015) Chromatin segmentation based on a probabilistic model for read counts explains a large portion of the epigenome. *Genome Biol.,* **16**, 151.

Marquardt, S., Hazelbaker, D. Z. & Buratowski, S. (2011) Distinct RNA degradation pathways and 3' extensions of yeast non-coding RNA species. *Transcription,* **2**, 145–154.

Martens, J. A., Laprade, L. & Winston, F. (2004) Intergenic transcription is required to repress the *Saccharomyces cerevisiae* SER3 gene. *Nature,* **429**, 571–574.

Murray, D., Beckmann, M. & Kitano, H. (2007) Regulation of yeast oscillatory dynamics. *Proc Natl Acad Sci U S A,* **104** (7), 2241–2246.

Park, Y., Figueroa, M. E., Rozek, L. S. & Sartor, M. A. (2014) `MethylSig`: a whole genome dna methylation analysis pipeline. *Bioinformatics,* **30**, 2414–2422.

Parkhomchuk, D., Borodina, T., Amstislavskiy, V., Ban aru, M., Hallen, L., Krobitsch, S., Lehrach, H. & Soldatov, A. (2009) Transcriptome analysis by strand-specific sequencing of complementary DNA. *Nucleic Acids Res,* **37** (18), e123.

Pekalska, E. & Duin, R. P. W. (2002) Dissimilarity representations allow for building good classifiers. *Pattern Recognition Let.,* **23**, 943–956.

Perocchi, F., Xu, Z., Clauder-Munster, S. & Steinmetz, L. (2007) Antisense artifacts in transcriptome microarray experiments are resolved by actinomycin D. *Nucleic Acids Res,* **35** (19), e128.

Picard, F., Lebarbier, E., Hoebeke, M., Rigaill, G., Thiam, B. & Robin, S. (2011) Joint segmentation, calling, and normalization of multiple CGH profiles. *Biostatistics,* **12**, 413–428.

Picard, F., Robin, S., Lavielle, M., Vaisse, C. & Daudin, J. (2005) A statistical approach for CGH microarray data analysis. *BMC Bioinformatics,* **6**, 27.

Rousseuw, P. J. (1987) "silhouettes": a graphical aid to the interpretation and validation of cluster analysis. *Comput. Appl. Math.,* **20**, 53–65.

Sardu, A., Treu, L. & Campanaro, S. (2014) Transcriptome structure variability in *Saccharomyces cerevisiae* strains determined with a newly developed assembly software. *BMC Genomics,* **15**, 1045.

Sasidharan, K., Amariei, C., Tomita, M. & Murray, D. (2012) Rapid DNA, RNA and protein extraction protocols optimized for slow continuously growing yeast cultures. *Yeast,* **29** (8), 311–322.

Slavov, N., Macinskas, J., Caudy, A. & Botstein, D. (2011) Metabolic cycling without cell division cycling in respiring ye ast. *Proc Natl Acad Sci U S A,* **108** (47), 19090–19095.

Smellie, A. (2004) Accelerated k-means clustering in metric spaces. *J Chem Inf Comput Sci,* **44**, 1929–1935.

Spingola, M., Grate, L., Haussler, D. & Ares Jr, M. (1999) Genome-wide bioinformatic and molecular analysis of introns in *Saccharomyces cerevisiae*. *RNA,* **5**, 221–234.

Sugar, C. A. & James, G. M. (2003) Finding the number of clusters in a data set: an information-theoretic approach. *J. Amer. Statistical Assoc.,* **98**, 750–763.

Sun, D., Xi, Y., Rodriguez, B., Park, H. J., Tong, P., Meong, M., Goodell, M. A. & Li, W. (2014) MOABS: model based analysis of bisulfite sequencing data. *Genome Biology,* **15**, R38.

Sun, S. & Yu, X. (2016) HMM-Fisher: identifying differential methylation using a hidden Markov model and Fisher's exact test. *Stat Appl Genet Mol Biol,* **15**, 55–67.

Thaben, P. & Westermark, P. (2014) Detecting rhythms in time series with RAIN. *J Biol Rhythms,* **29** (6), 391–400.

Tisseur, M., Kwapisz, M. & Morillon, A. (2011) Pervasive transcription – lessons from yeast. *Biochimie,* **93**, 1889–1896.

Trapnell, C., Williams, B. A., Pertea, G., Mortazavi, A., Kwan, G., van Baren, M. J., Salzberg, S. L., Wold, B. J. & Pachter, L. (2010) Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nat Biotechnol,* **28**, 511–515.

Tsai, H.-K., Su, C. P. C., Lu, M.-Y. J., Shin, C.-H. & Wang, D. (2007) Co-expression of adjacent genes in yeast cannot be simply attributed to

shared regulatory system. *BMC Genomics,* **8**, 352.

Vinod, H. D. (1969) Integer programming and the theory of grouping. *J. Amer. Stat. Assoc.,* **64**, 506–519.

Wu, H., Xu, T., Feng, H., Chen, L., Li, B., Yao, B., Qin, Z., Jin, P. & Conneely, K. N. (2015) Detection of differentially methylated regions from whole-genome bisulfite sequencing data without replicates. *Nucleic Acids Res.,* **43**, e141.

Xu, Z., Wei, W., Gagneur, J., Perocchi, F., Clauder-Munster, S., Camblong, J., Guffanti, E., Stutz, F., Huber, W. & Steinmetz, L. (2009) Bidirectional promoters generate pervasive transcription in yeast. *Nature,* **457** (7232), 1033–1037.

Yamashita, A., Shichino, Y. & Yamamoto, M. (2016) The long non-coding RNA world in yeasts. *Biochim Biophys Acta,* **859**, 147–154.

Yassour, M., Pfiffner, J., Levin, J. Z., Adiconis, X., Gnirke, A., Nusbaum, C., Thompson, D., Friedman, N. & Regev, A. (2010) Strand-specific RNA sequencing reveals extensive regulated long antisense transcripts that are conserved across yeast species. *Genome Biol,* **11**, R87.

Zacher, B., Lidschreiber, M., Cramer, P., Gagneur, J. & Tresch, A. (2014) Annotation of genomics data using bidirectional hidden Markov models unveils variations in Pol II transcription cycle. *Mol Syst Biol,* **10**, 768.

Appendix A

Let A be a symmetric $n \times n$ matrix. In order to compute $B_{k,l} := \sum_{i=k}^{l} \sum_{i=k}^{l} A_{ij}$ and $M_{k,l} := \max_{k \le p \le l} \sum_{i=k}^{l} A_{ij}$ efficiently, we my construct the auxiliary matrices U and V in quadratic time:

$$U_{ip} := \sum_{j=1}^{p} A_{ij} = U_{i,p-1} + A_{ip}$$

$$V_{rp} := \sum_{i=1}^{r} \sum_{j=1}^{p} A_{ij} = \sum_{i=1}^{r} U_{i,p} = V_{r-1,p} + U_{r,p}$$

with initial conditions $U_{i1} = A_{i1}$ and $V_{1p} = U_{1p}$. Then $B_{kl} = V_{ll} + V_{k-1,k-1} - 2V_{k-1,l}$ and $M_{kl} = \max_{k \le p \le l}(U_{pl} - U_{p,k-1})$. This B and M can be computed in quadratic and cubic time, respectively.

Similarity-Based Segmentation of Multi-Dimensional Signals
Supporting Figures S1–S5

Rainer Machné, Douglas B. Murray and Peter F. Stadler

April 21, 2017

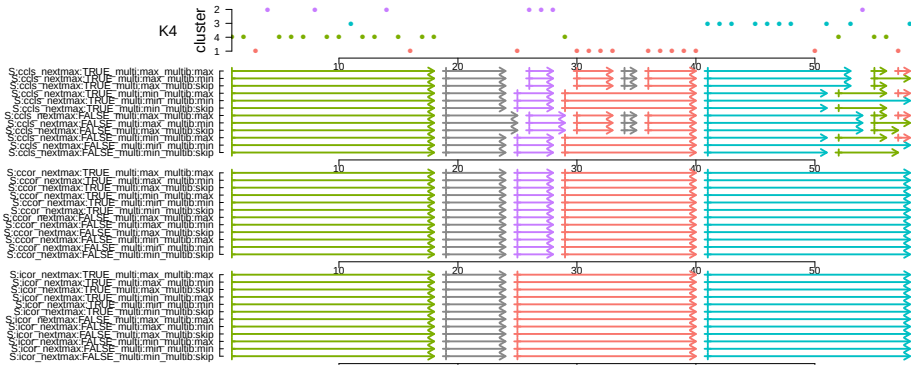


Figure S1: **Simulated Data.** Only scoring function `ccls` is sensitive to internal details of scoring function calculation and back-tracing.

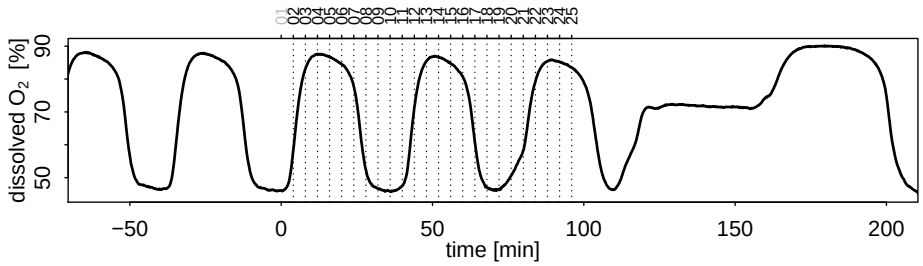


Figure S2: **Sampling of the RNA-seq Time-Series: Dissolved O_2.** The experimental system, respiratory oscillations during continuous growth of budding yeast strain IFO 0322, has been described extensively (Murray *et al.*, 2007, Machné & Murray, 2012) growth and dynamics of dissolved Oxygen in the reactor were equivalent to previous experimental conditions (Murray *et al.*, 2007), and oscillation appeared stable. Samples 01–25 were taken at the times indicated, sample 01 was discarded and only samples 02–25 were processed. Notably, the culture underwent a transient shortly after sampling, perhaps induced by sampling itself, then re-established oscillatory growth.

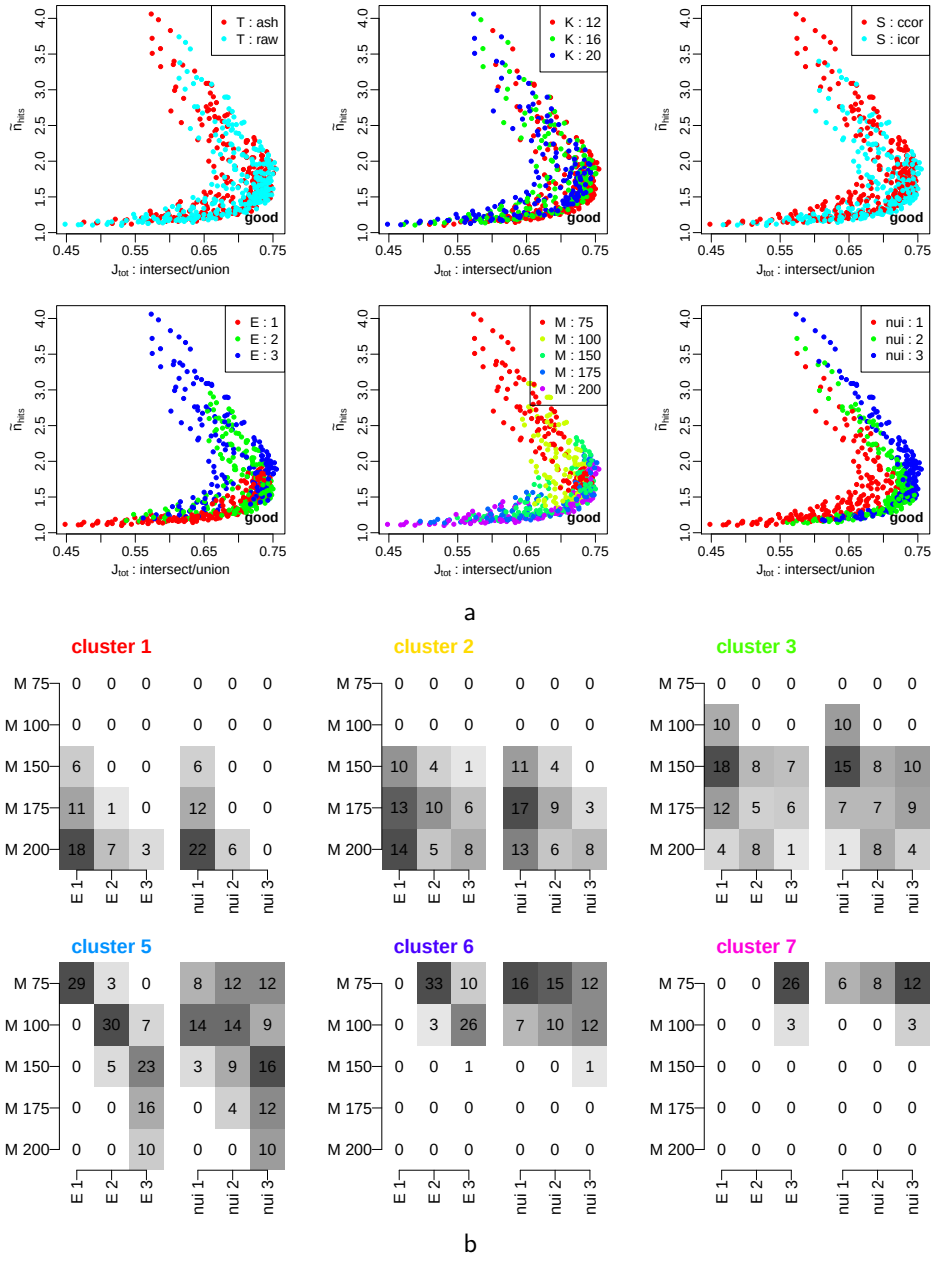


Figure S3: **Parameter Scan: Parameter Effects.** (a) As Fig. 2b, but colored by time-series processing, clustering and segmentation parameters. (b) As Fig. 2d but for all other PAM clusters shown in Fig. 2a–2c.

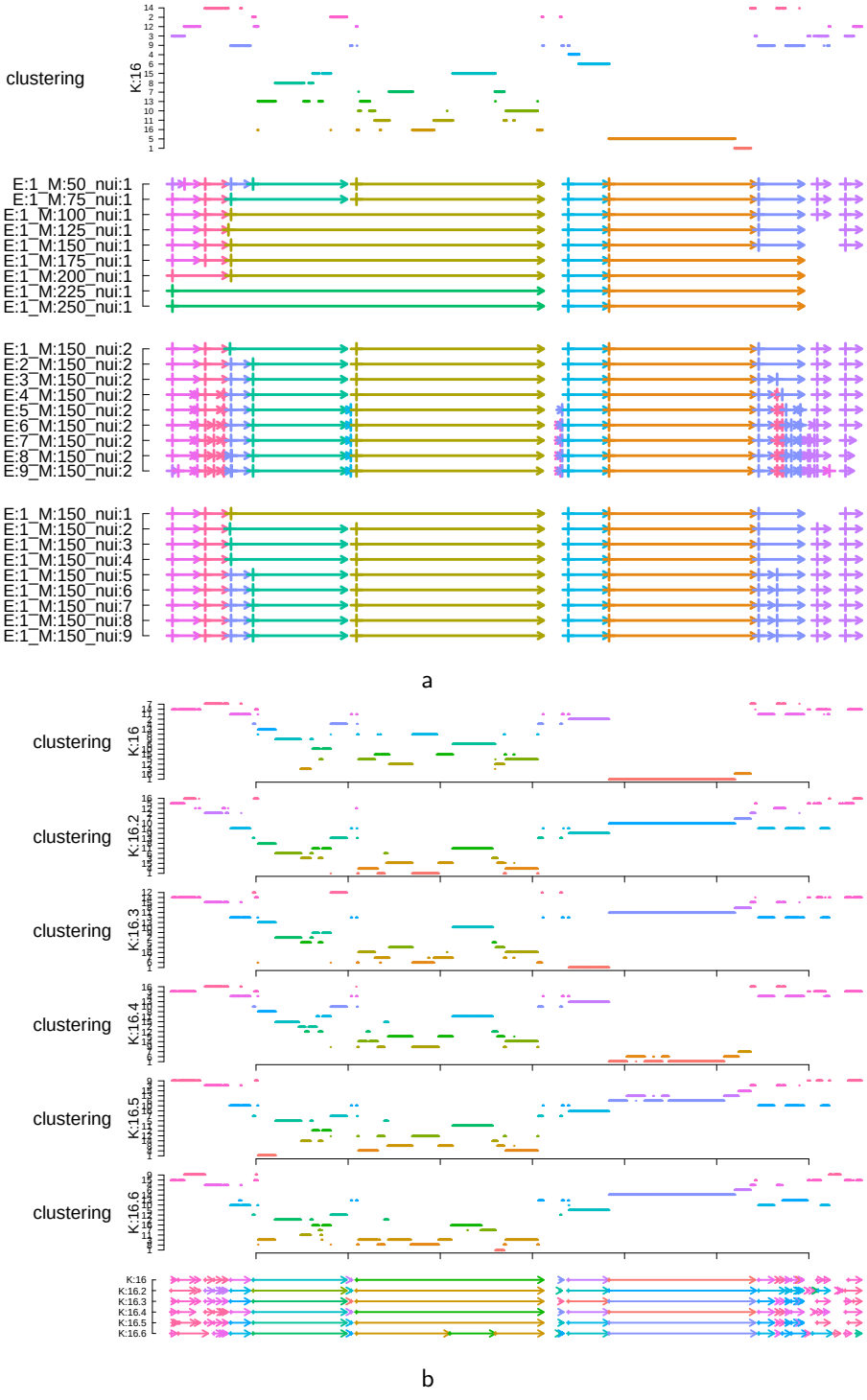


Figure S4: **Example Region: SRG1 vs. SER3.** Systematic scans of dominant parameters (a) and random effect from cluster initialization (b).

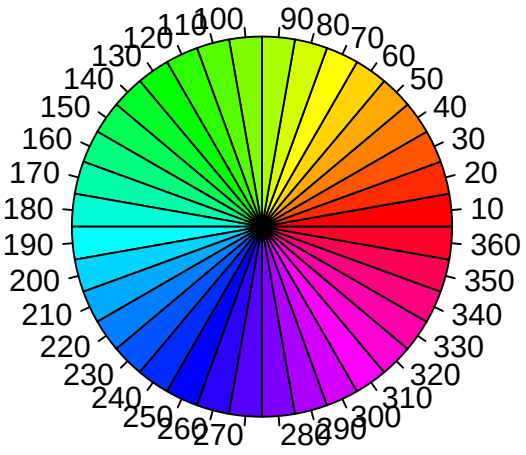


Figure S5: **Phase coloring scheme.** Phase coloring scheme used in Fig. 4 of the main paper.