HCsnip: An R Package for Semi-supervised Snipping of the Hierarchical Clustering Tree

Askar Obulkasim[1] and Mark A. van de Wiel[1,2]

[1]Department of Epidemiology and Biostatistics, Vrije Universiteit Medical Center. [2]Department of Mathematics, Vrije Universiteit, Amsterdam, The Netherlands.

ABSTRACT: Hierarchical clustering (HC) is one of the most frequently used methods in computational biology in the analysis of high-dimensional genomics data. Given a data set, HC outputs a binary tree leaves of which are the data points and internal nodes represent clusters of various sizes. Normally, a fixed-height cut on the HC tree is chosen, and each contiguous branch of data points below that height is considered as a separate cluster. However, the fixed-height branch cut may not be ideal in situations where one expects a complicated tree structure with nested clusters. Furthermore, due to lack of utilization of related background information in selecting the cutoff, induced clusters are often difficult to interpret. This paper describes a novel procedure that aims to automatically extract meaningful clusters from the HC tree in a semi-supervised way. The procedure is implemented in the R package **HCsnip** available from Bioconductor. Rather than cutting the HC tree at a fixed-height, **HCsnip** probes the various way of snipping, possibly at variable heights, to tease out hidden clusters ensconced deep down in the tree. The cluster extraction process utilizes, along with the data set from which the HC tree is derived, commonly available background information. Consequently, the extracted clusters are highly reproducible and robust against various sources of variations that ``haunted" high-dimensional genomics data. Since the clustering process is guided by the background information, clusters are easy to interpret. Unlike existing packages, no constraint is placed on the data type on which clustering is desired. Particularly, the package accepts patient follow-up data for guiding the cluster extraction process. To our knowledge, **HCsnip** is the first package that is able to decomposes the HC tree into clusters with piecewise snipping under the guidance of patient time-to-event information. Our implementation of the semi-supervised HC tree snipping framework is generic, and can be combined with other algorithms that operate on detected clusters.

KEYWORDS: hierarchical clustering, semi-supervised clustering, data integration, high-dimensional data, R package

CITATION: Obulkasim and van de Wiel. HCsnip: An R Package for Semi-supervised Snipping of the Hierarchical Clustering Tree. *Cancer Informatics* 2015:14 1–19 doi: 10.4137/CIN.S22080.

RECEIVED: November 21, 2014. **RESUBMITTED:** January 05, 2015. **ACCEPTED FOR PUBLICATION:** January 06, 2015.

ACADEMIC EDITOR: JT. Efird, Editor in Chief

TYPE: Software or Database Review

FUNDING: This study was performed within the framework of the Center for Translational Molecular Medicine, DeCoDe project (grant 03O-101). The authors confirm that the funder had no influence over the study design, content of the article, or selection of this journal.

COMPETING INTERESTS: Authors disclose no potential conflicts of interest.

CORRESPONDENCE: a.wubulikasimu@erasmusmc.nl, mark.vdwiel@vumc.nl

Background

The increasing affordability of high-throughput molecular data is enabling the simultaneous measurement of several genomic features in the same biological samples. With the consequent explosion of various types of molecular data, a general question facing biologists and statisticians is how to organize the observed molecular data into meaningful structures. Clustering has been widely explored for this purpose. Hierarchical clustering (HC) is probably the most ubiquitous unsupervised clustering technique for analyzing molecular data,[1] probably due to its simplicity and yet ability to represent data at different resolutions. The HC algorithms organize objects into a hierarchical cluster tree (called dendrogram) whose branches are the desired clusters. However, there are no explicit clusters in the HC algorithm output. Normally, a fixed-height cutoff is chosen and each contiguous branch of objects below that height is considered a separate cluster. This method of cutting poses an impediment on the ability of HC algorithms to fully exploit the data. There are at least two conceivable drawbacks associated with this method of cutting. First, the manner of choosing a fixed-height cut, which is in general difficult to determine, is heuristic. The user has to visually observe the HC tree structure to decide upon the cutoff, which is problematic for nonprofessional users. Until now, there is no known and reliable heuristic method to choose an optimal cutoff. In some cases, no single fixed-height cut can identify clusters deemed to underlie the data correctly.[2] Second, the fixed-height cut may identify cancer molecular subtypes that are unrelated to patients' clinical information.[3] Because it does not utilize the available clinical data to identify subtypes, there is no guarantee that the returned subtypes will exhibit significant functional coherence.

To address the aforementioned issues, we present a semi-supervised HC tree-snipping framework called **HCsnip** for the open source statistical computing environment R, available from Bioconductor. This open source package includes a novel procedure to induce cut(s) on the HC tree by snipping at (possibly) variable heights, subsequently extracting hidden clusters ensconced deep down in the tree. The proposed algorithm first extracts all possible partitions in a given HC tree to

construct the searching space. Here, a partition corresponds to a possible clustering that consists of nonoverlapping clusters. Then, partitions are evaluated using the data type from which the HC tree is derived and the available background information. Finally, an optimal partition is selected based on its aggregated quality scores from the two data sets. One may expect that the selected partition at this step captures the data structures deemed in the two data sets. Unlike exiting packages, our implementation is flexible enough to accommodate any type of molecular data and does not have any constraint on the format of background information. Especially, our approach accepts commonly available, clinically relevant patient follow-up data as background information. To our knowledge, **HCsnip** is the first implementation of the cluster extraction procedure from a HC tree under the partial "guidance" of time-to-event data. To sum up, **HCsnip** is a data-driven, rather than heuristic, exploratory data analysis tool that has high potentials in exploiting the enormous and yet mostly untapped information latent in the data. Detailed descriptions of the methodology implemented in the **HCsnip** package are provided in Obulkasim et al.[4]

The remainder of the article is organized as follows. The section "Semi-supervised HC tree snipping" contains succinct descriptions of important ideas and tasks that an optimal HC tree-snipping framework should address and our motivation to write this package. The section on "The Semi-supervised HC tree-snipping routines" elaborates clustering routines such as how to snip a given HC tree within our framework, extraction of an optimal partition among large number of candidates, unbiased *P*-value calculation for the optimal partition using permutation argument, prediction of cluster membership of a new sample using the semi-supervised method, visualization of clusters using samples' molecular entropy, and optimal treatment assignment using multiple HC trees. Illustrations of the use of package with built-in data sets are given in the "Examples" section. The section "Illustrations with real-world data sets" presents applications of **HCsnip** to real-world data sets that are not included in the package. The article closes with a discussion in the "Conclusion" section.

Semi-supervised HC Tree Snipping

One of the reasons that make HC algorithms popular is that they are able to represent nested clusters. However, the nested clusters are likely to be occluded when the standard fixed-height cut approach is employed.[2,5] A series of works has been proposed to surmount this problem. For instance, the Dynamic Tree Cut approach (DTC)[6] is designed to detect clusters in the HC tree according to their shapes. Although the DTC is able to extract clusters without using the fixed-height cutoff, it inherits the unsupervised nature of the HC algorithms. Thus, extracted clusters often exhibit weak (if any) clinical relevance. By including partial background information provided by the user, the cluster extraction process may produce results that come closer to the user's expectations. To our knowledge, so far, there are only two methods available for extracting clusters from the HC tree by integrating prior knowledge into the cluster extraction process. They are the Variation Information cut (VI-cut) by Navlakha et al (2010)[7] and a HC tree-snipping approach by Dotan-Cohen et al (2007).[8] The former tries to induce clusters in the HC tree decompositions that best match a partial set of known annotations. The latter seeks a partition on the HC tree that requires the minimum number of edge snips such that all genes in the cluster have a label in common.

These methods, however, are not immune to problems. First, both methods were designed for gene clustering, which is very different from clustering samples.[9] Second, both placed a constraint on the type of background information, viz background information must be in discrete format, eg, gene annotation labels. In practice, however, continuous scaled clinical data are commonly available, and usually, conversion of these into a discrete format is associated with information loss. These shortcomings hamper their utility in the situation in which the HC tree is derived for sample clustering and a direct application of an arbitrary type of background information (without preprocessing) in the tree snipping process so that the resulting clusters are informative, as well as robust enough, to extrapolate to new samples as wished for.

Alleviating the hurdles in the aforementioned situation is directly linked to our motivation to write the **HCsnip** package: to develop a general adaptive-height HC tree-snipping framework for semi-supervised sample clustering. Besides inheriting the advantages of the HC algorithms, **HCsnip** enjoys the following merits that existing packages lack:

- **HCsnip** is designed for sample clustering.
- Flexible implementation accepts any data type for clustering and lifts the constraint on the format of background information. Most notably, it allows clinically relevant patient time-to-event data to "guide" the HC tree-snipping process.
- The optimal partition selection process aggregates the clustering quality scores from the two data sets to dampen the effect of noise in the data so that a more stable clustering result is warranted.
- A novel application of **HCsnip** is in optimal treatment assignment, which we believe has potentials in personalized medicine paradigm, if implemented.

Functions in **Hcsnip** have input parameters that are easy to understand and determine. The user may also choose to call functions with default values that are likely to produce reasonable results in most cases.

The Semi-supervised HC Tree-Snipping Routines

HCsnip can be regarded as a tool to integrate multiple data sets for clustering purpose. It comprises many novel functions

such as an efficient procedure to extract all possible partitions from a given HC tree and a permutation test that is specially designed for testing the significance of the association of the extracted clusters with data on patient follow-up in an unbiased manner. Besides visualizing the cluster differences in terms of follow-up, an ancillary visualization scheme that uses the samples' molecular entropy to display the cluster difference at the molecular level has been incorporated in **HCsnip** as well. We also implemented a novel procedure to assign new sets of samples to given clusters in a semi-supervised way. Two procedures exclusively designed for optimal treatment assignment application are implemented in a computationally efficient way. To well integrate with the Bioconductor structure and make the package more accessible, **HCsnip** is implemented to accept an *ExpressionSet* type class (**Biobase** package), which is one of the commonly used Bioconductor classes. In the following sections, we present in-depth descriptions of each package feature.

The search space construction. The first step in extracting the optimal clustering from a given HC tree is to extract all possible partitions from it. Before this step, we presume a HC tree that expresses similarities among samples in terms of their molecular profiles, which are readily available, reasonably well. The user has the freedom to choose a distance metric and a linkage method from among a large number of options. Herein, we illustrate our piecewise cutting scheme with a toy example.

Say we are given a HC tree as in Figure 1. We first split the HC tree into two branches. Then, generate the following list of partitions by applying the fixed-height cut method to cut each branch at all possible heights separately:

on the left branch:

{4, 5}	{9, 6, 10}
{4, 5, 9}	{6, 10}
{4, 5, 9, 6, 10}	

on the right branch:

{1, 8}	{2, 3, 7}
{1, 8, 2}	{3, 7}
{1, 8, 2, 3, 7}	

This way of cutting ensures that the HC tree structure shall not be contravened. Hence, the extracted clusters are faithful to the data set from which the HC is derived. Because we do not allow for singletons (to avoid generating too many spurious partitions and to reliably estimate partition quality in later stages), we permit one exception to this rule: if a snip creates singleton(s), then they are assigned to the closest cluster that is not involved in the particular snip. In the example above, if we snip the cluster {9, 6, 10} just below "9," {9} is then merged with {4, 5} to form {4, 5, 9}.

Finally, we make all possible clustering combinations between branches. We then receive the following set of unique partitions:

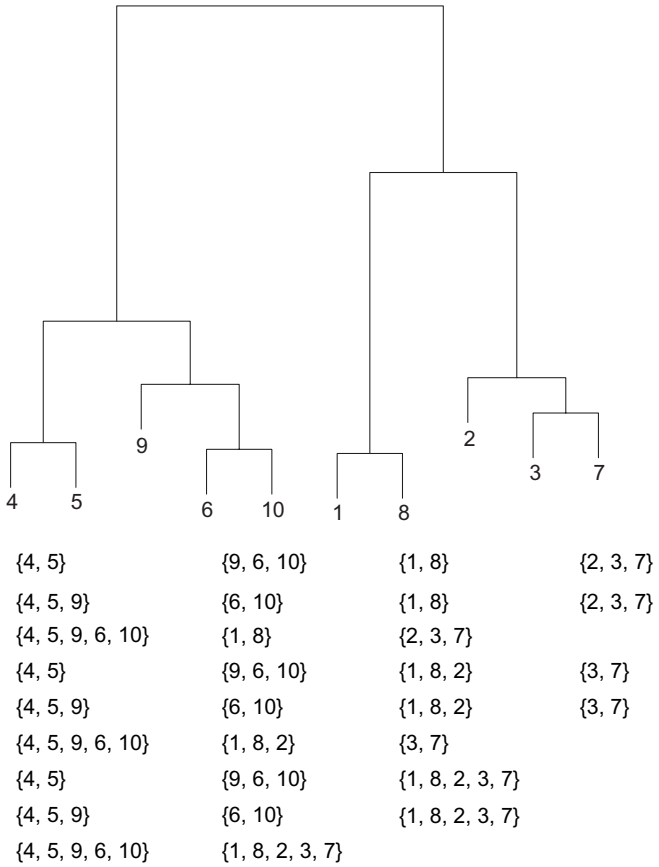

Figure 1. An example HC tree.

This set will be the search space in later stages for extracting the optimal partition (clustering). To reduce the effect of outliers and reliably estimate the quality of clusters in each partition, we introduce a threshold parameter on the minimum number of samples in each cluster. The number of partitions returned from this step is inversely proportional to this threshold.

A function that executes the aforementioned procedures is *HCsnipper*, which has the following arguments:

```
HCsnipper(X, hc = NULL, dis = NULL,
dis.method = "cor", link.method = "ward", minclus = 4,
maxmiss = 30, …)
```

The available arguments are as follows:

- **X:** An object of class *ExpressionSet* or data matrix from which the HC tree needs to be derived. Columns are assumed to represent the samples, and rows represent the samples' features (eg, genes). Missing values are allowed.
- **hc:** HC tree from which partitions needs to be extracted. This must be an object of class hclust (from the **stats** package). This is an optional argument, and if given, "X" and "dis" will be ignored.
- **dis:** A square distance matrix or object class of *dist* (from the package **stats**) from which the HC tree needs to be

derived. This is an optional argument, and if given, "X" will be ignored.

- **dis.method:** This is the distance measure to be used. This must be one of the methods acceptable for the function *dist* in the **stats** package or the Pearson correlation "cor" (default).
- **link.method:** The agglomeration method to be used. This should be one of "ward" (default), "single," "complete," "average," "mcquitty," "median," or "centroid."
- **minclus:** The minimum number of samples allowed to form a cluster. This parameter is inversely proportional to the number of partitions returned, ie, a large value returns fewer partitions, and vice versa.
- **maxmiss:** Allowed maximum percentage of missing values per row in "X."
- **…** Additional arguments for the *impute.knn* function from the **impute** package to impute missing values in "X."

The number of partitions returned by the function depends not only on the "minclus" argument but also on the shape of the HC tree. A balanced HC tree results in more partitions than a skewed one.

A call to *HCsnipper* function should at least contain one of the first three arguments. This function returns an object-of-class list. Besides containing the slots that report most of the arguments provided in the function call, this list object has a slot, "partitions," which contains a matrix in which rows represent the partitions – each of which is composed of exhaustive and mutually exclusive clusters – and columns represent the samples.

Partition quality assessing. Once the search space is ready, the next step is to objectively evaluate the quality of each partition. The usefulness of the optimal partition shall be returned in a later stage depending on the output of this stage. Hence, correct measurement of the quality of partition is of vital importance. A cluster quality measure is a function that uses the given data set and the corresponding partition to assign a nonnegative real number to the partition, which reflects how "good" or "cogent" the clustering is. A plethora of cluster quality measures have been proposed, but none of them has been proven to perform uniformly better than the rest. It is probably impossible to make all exiting measures available in this package. However, many well-known ones are included for the user to choose from. They are subsumed under the following three categories based on the input data type:

1. When the input is molecular data: Three well-known cluster quality measures for high-dimensional molecular data, which have been extensively investigated by Bolshakova et al.[10], are available for this data type. They are Within-cluster sum of squares (WSS), C-index,[11] and Goodman–Kruskal index.[12] On simply supplying a partition under consideration and the distance matrix, the function produces a score that indicates the optimality of solution that the partition represents. Except for the C-index, the remaining measures are calculated using the **fpc** package.[13]
2. When the input data is in discrete format: If the available data are in the discrete format (eg, sex, binary tumor remission status etc), the user may consider using the in-group proportion criteria.[14]
   A function that generates cluster quality score in the above two cases is *measure*. It has the following input arguments:
   - **parti:** A partition to be evaluated must comprise exhaustive and mutually exclusive clusters.
   - **dis:** A square distance matrix or object class of *dist* (from the **stats** package) corresponds to "parti."
   - **X:** An object of class *ExpressionSet* or data matrix from which the HC tree has been derived. Columns are assumed to represent the samples, and rows represent the samples' features (genes). Missing values are allowed. Only needs to be supplied when "method" is set to "igp" (in-group proportion).
   - **method:** The type of evaluation measure to be used for assessing the quality of "parti." Available options are "c-index," "igp," and all other measures made available in the *cluster.stats* function from the **fpc** package. Note that values returned by different measures have different meanings. For instance, if the silhouette coefficient is chosen, a large value indicates a high-quality partition, whereas a small value denotes a high-quality partition when the C-index is used. Thus, interpret the returned values accordingly.
   - **maxmiss:** Maximum allowed percentage of missing values per row in "X."
   - **…** Additional argument for the function *cluster.stats* from the **fpc** package.

   measure returns a numeric value representing the quality of partition under consideration.
3. When the input is follow-up data: Here, a partition is used as a covariate and the time-to-event information as the response variable to check their association. The following two measures tailored to time-to-event tasks are included: modified Akaike information criterion (AIC)[15] and modified Bayesian information criterion (BIC).[16] Unlike *P*-values, these information criteria allow comparison across partitions that include different numbers of clusters. Our experiments show that, qualitatively, the ordinary AIC and BIC do not lead to considerable differences.

In the follow-up data setting, *surv_measure* can be called to calculate the quality of a partition. Next to the inputs required in the function measure, this function requires survival time

of patients "surv.time" and patients' survival status indicator "status" to be supplied. The former has to be a numeric vector, and the latter has to be in binary format (0, 1), where 1 denotes an event (dead). Note that, for convenience, the output is multiplied by –1 inside the function so that a large value denotes a high-quality partition.

If the user has precalculated quality scores of partitions, this step can be skipped. It is worthwhile to mention that the optimal partition selected in later stages depends, to some extent, on the type of cluster quality measure used. Which measure suits which data type is beyond the scope of this package. However, it may be useful to evaluate a partition with more than one method and take their average.

The optimal partition selection and bias-corrected *P*-value calculation. The previous two steps are designed to generate inputs required for this step. Here, the optimal partition is selected from among a large number of candidates based on their quality scores from the two input data sets. The optimal one should be selected in such a way that it unravels the structure deemed to underlay the data, and most importantly, is stable enough to extrapolate to new samples.

We presume that besides the molecular data set (from which the HC has been derived), we may also have (i) partial clinical data of the same patients (normally obtained before measuring the former) such as follow-up data[3] or (ii) another type of molecular data measured on the same genome. Assume that the available extra data comprises patient follow-up. If the optimal partition is selected solely based on the quality scores from molecular data, returned clusters may not exhibit relevance to the follow-up information. Whereas, yield clusters may not be faithful to the molecular data if only the quality scores from follow-up data are utilized. To obtain a robust clustering and to represent the data structure in both data spaces, **HCsnip** selects a partition that is ranked high by both data types as the optimal one (Fig. 2). Specifically, it ranks the quality scores from the two data sets from good to bad, separately. The final score of a partition is the summation of the two ranking values. A partition with the smallest ranking value (after summation) is returned as the optimal clustering in the data. We argue that, this way of judging the cluster quality may dampen the effect of noise in the molecular data used, and eases the interpretation of the optimal clustering.

Next to the quality score generated earlier, perhaps it is helpful to express the quality of the optimal partition in terms of probability, such as the statistical lingua franca of biomedical research, the *P*-value. However, when studying the association of cluster membership with clinical follow-up information, such as survival data, we cannot use the standard testing procedures when our semi-supervised approach has been applied: we would use the follow-up data twice, and the resulting *P*-value is likely to be too small. **HCsnip** avoids this bias by also applying the semi-supervised cluster construction

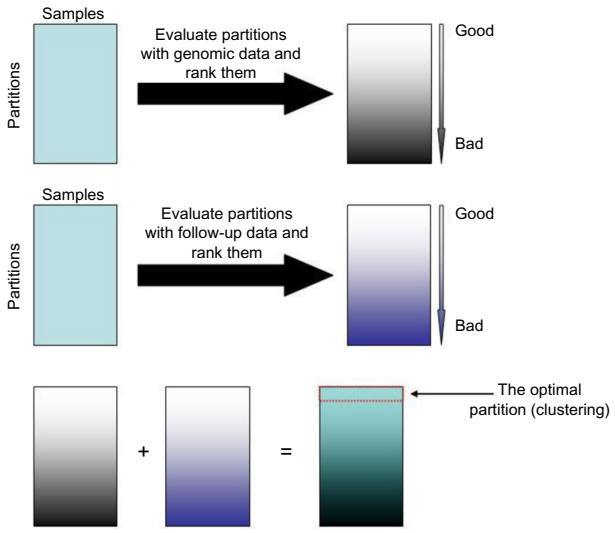

Figure 2. Schemata of the optimal partition selection procedure.

under the null hypothesis. This null hypothesis is simply the absence of association between the samples' molecular information and the follow-up data. Then, our cluster construction in combination with a suitable test statistic is designed to detect associations that can be represented by groups of samples. We adapt the *P*-value computation as follows.

1. For the observed data, compute the clusters using a semi-supervised approach (piecewise or fixed-height cut).
2. Use a suitable test statistic (eg, log-rank for time-to-event data and chi-square for nominal data) to compute the conditional *P*-value given the resulting clusters: p_{obs}.
3. For $i = 1,\ldots, B$ (eg, $B = 1000$):
   a. Randomly permute the follow-up data among the samples.
   b. Apply exactly the same semi-supervised approach for each instance.
   c. Conditional on the resulting clusters, compute *P*-value p_i.
4. Finally, compute the *P*-value of interest: $p = P(p_{obs} \geq p_i) = \{\#i | p_{obs} \geq p_i \text{ for } i = 1,\ldots, B\}/B$.

Here, p satisfies a crucial property of *P*-values: it is uniformly distributed when the null hypothesis is true, because then p_{obs} and p_i are exchangeable random variables. The exchangeability is a result from the null hypothesis and the use of exactly the same procedures to compute p_{obs} and p_i.

The function *perm_test* in this package is designed to perform the optimal partition selection and an unbiased *P*-value calculation. It has the following input arguments:

- **partitions:** A matrix in which rows represent the partitions and columns represent the samples. Each row must contain exhaustive and mutually exclusive clusters.

- **surv.time:** A numeric vector that contains patients' survival times.
- **status:** A numeric binary vector (0, 1) that contains patients' survival status indicator, where 1 denotes an event (dead).
- **score1:** A numeric vector that includes the quality scores of partitions calculated using the data type from which the HC tree has been derived. This vector must be prepared in such a way that a large value indicates a high-quality partition.
- **score2:** A numeric vector that includes the quality scores of partitions calculated using the follow-up data. This vector must be prepared in such a way that a large value indicates a high-quality partition.
- **method:** The type of partition quality measure that has been used to calculate "score1."
- **nperm:** The number of permutations to be performed. This function returns an object-of-class list with the following slots: "obs.p," observed *P*-value of the optimal partition; "perm.p," containing the *P*-values generated by permutations; "best," the optimal partition.

Clustering on new sets of samples. HCsnip includes a novel method (*PNN+ Concordance*) to assign new samples to one of the predefined clusters in a semi-supervised manner. *PNN+ Concordance* attempts to directly assess the similarity between a new sample and a given partition using their molecular profiles, as well as indirectly assess their similarity in terms of time-to-event information using the pseudo nearest neighbor (PNN).[17] Herein, we illustrate the working principle of *PNN+ Concordance* with an example of a toy.

Say we have 10 samples for which both gene expression data and time-to-event information are available. Going through the aforementioned multiple steps, a partition with two clusters is selected as the optimal one. Suppose the cluster labels and time-to-event information of these 10 samples are as follows:

cluster = (1, 1, 1, 1, 2, 2, 2, 2, 2, 2),
surv.time = (2.6, 4.0, 4.5, 4.9, 5.4, 25.3, 26.8, 28.2, 28.3, 29.4),
event = (0, 1, 0, 0, 0, 1, 1, 1, 0, 0).

A new sample arrives, of which only the gene expression profile is available. We project this new sample onto the expression data space and measure its similarity with each of the 10 training samples. Say after ordering (descending) the similarities, we obtain the following order indexes:

$$\text{ord}^{\text{test}} = (10, 3, 6, 4, 1, 9, 5, 2, 7, 8).$$

Save the indexes of the first $k = 3$ samples (nearest neighbors) in the list. Note that both expression profiles and time-to-event information are available for these samples. We regard these samples as the PNNs of the new sample in the patient time-to-event data space (Fig. 3). The logic behind this is that when two samples have similar molecular characteristics, they may also share clinical characteristics due to the potential association between the two types of features.

Select the first sample from the PNN list (blue), make all possible pairs of cases with samples in the first cluster (black). The results are as follows:

(29.4, 2.6)	(0, 0)
(29.4, 4.0)	(0, 1)
(29.4, 4.5)	(0, 0)
(29.4, 4.9)	(0, 0)

Take the first pair from the pairing list, label them as the active set and the remaining eight samples (nine if the active set pairs are the same) as the reference set (ω). Select a sample from the reference set and combine it with the active set to form a triplet. Calculate the similarity of the active set pair according to the censoring status of the triplet (Fig. 4) using the Concordance index,[18] which is defined as follows:

1. If the reference sample has an event and the active set pairs are censored

$$c^1 = \sum_{i \in \omega} I\left\{ t_1 \geq t_i^{ref} \; AND \; t_2 \geq t_i^{ref} \right\} \tag{1}$$

2. If both the reference sample and the active set have events

$$c^2 = \sum_{i \in \omega} I\left\{ t_1 \geq t_i^{ref} \; AND \; t_2 \geq t_i^{ref} \right\} OR \left\{ t_1 < t_i^{ref} \; AND \; t_2 < t_i^{ref} \right\} \tag{2}$$

3. If the reference sample is censored and the active set has events

$$c^3 = \sum_{i \in \omega} I\left\{ t_1 \leq t_i^{ref} \; AND \; t_2 \leq t_i^{ref} \right\} \tag{3}$$

where $I\{\cdot\} = 1$ if the argument is true and $I\{\cdot\} = 0$ otherwise. Here, t_1 is the survival time of the first sample in the active set, t_i^{ref} is the survival time of the i^{th} reference sample. The final similarity of the active set is

$$\text{sim} = \frac{\sum_{i=1}^{3} c^i}{|\Phi|} \tag{4}$$

where Φ is the set that consists of all triplets that satisfy one of the aforementioned conditions. Following these calculation steps, we obtain the following similarities between the samples in the PNN list and the two clusters under consideration:

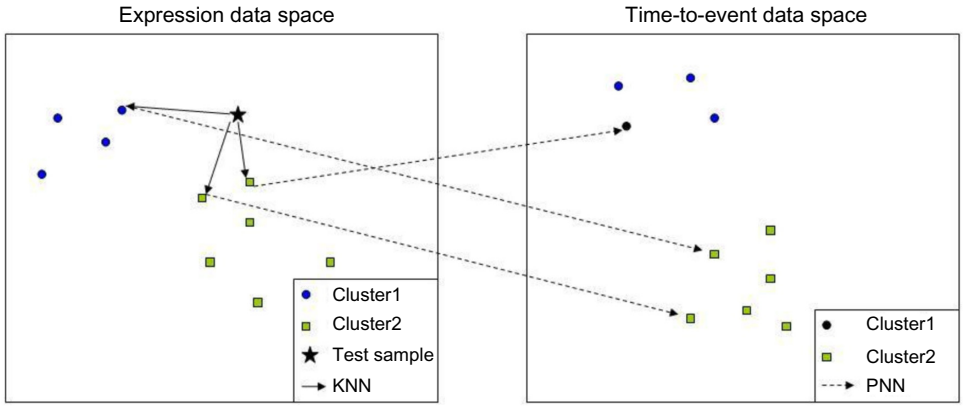

Figure 3. Illustration of the indirect projection principle.

PNN	*cluster*[1]	*cluster*[2]
10	0.357	0.516
3	0.727	0.667
6	0.571	0.08

Weight the similarities with cluster[1] by the inverse of the ordering indexes, we obtain

$$\frac{0.357}{1}+\frac{0.727}{2}+\frac{0.571}{3}=0.911 \quad (5)$$

Analogously, we obtain 0.796 for cluster[2]. Thus, *PNN+Concordance* assigns the new sample to cluster[1].

Note that the value of k varies according to the cluster size. This may bias toward the big cluster. One may reweight the similarities according to the cluster size. We do not do so because 1) we apply the inverse weighting scheme. This way of weighting diminishes the effect of a neighbor located far from the test sample. Thus, large k (equals more summation terms in Equation 5) does not affect the final similarity value much. 2) The prior probability for a test sample to be assigned to a large cluster should be somewhat larger than that for a small cluster.

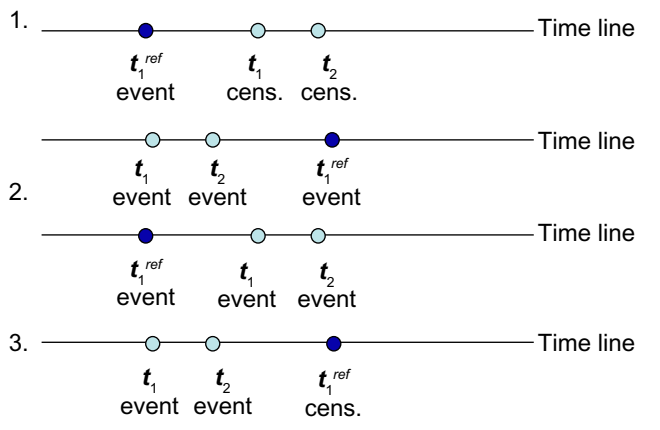

Figure 4. Illustration of the three scenarios considered in the Concordance index calculation.

The function that performs semi-supervised clustering on a new set of samples is *cluster_pred*, which has the following arguments:

```
cluster_pred(X, partition, surv.time = NULL,
  status = NULL, te.index, minclus = 4, te.surv.
  time = NULL, te.status = NULL, method = "conc",
  maxmiss = 30, plot.it = FALSE, …)
```

- **X:** An object of class *ExpressionSet* or data matrix in which columns are assumed to represent the samples and rows represent the samples' features. "X" can also be a square distance matrix or an object class of *dist*. It must comprise the data set from which "partition" has been obtained and the data set corresponds to the new set of samples.
- **partition:** A numeric vector containing exhaustive and mutually exclusive clusters that have been induced on the training set.
- **surv.time:** An optional numeric vector that contains patients' survival times. Only needs to be specified when "method = conc."
- **status:** An optional numeric binary vector (0, 1) that contains patients' survival status indicator, where 1 denotes an event (dead). Only needs to be specified when "method = conc."
- **te.index:** Indexes of the columns in "X" that correspond to the new set of samples.
- **minclus:** The minimum number samples allowed to form a cluster on the new set. This is to avoid returning tiny clusters and to reduce the effect of outliers.
- **te.surv.time:** An optional numeric vector that contains new patients' survival times. Only needs to be specified when "plot.it = TRUE."
- **te.status:** An optional numeric binary vector (0, 1) that contains new patients' survival status indicator, where 1 denotes an event (dead). Only needs to be specified when "plot.it = TRUE."
- **method:** The type of method desired to assign the new set to one of the clusters in "partition." Must be either

Ward distance "ward" or *PNN + Concordance* "conc" (default). The latter takes advantage of the correlation between molecular ("X") and follow-up ("surv.time" and "status") data of the training set. If this correlation is weak, the two options, then, are equal. We suggest using the *globaltest*[19] as pretest to check this correlation before choosing one of them.
- **maxmiss:** Maximum allowed percentage of missing values per row in "X."
- **plot.it:** If "TRUE" and the follow-up data of the new set are given, a Kaplan–Meier plot will be generated for each cluster induced on the new set.
- **…** Additional arguments for the function *impute.knn* from the impute package to impute missing values in "X."

When the follow-up data of the new set is not given, the function returns a vector of cluster labels of the new set. If supplied, an object of class list with the following slots will be returned instead:

- **St:** A data frame with five columns. The first column contains the unique survival time in "te.surv.time." The second column includes the cluster labels of the new set, and the number of patients at risk at each unique time point is given in the third column. The survival probability and the variance of the survival probability at each unique time point are provided in the last two columns.
- ***P*-value:** The log-rank test *P*-value of the new set.

Error rate calculation using the Random Survival Forest. To objectively quantify the "goodness" of clustering on the new set of samples, the association between the test samples' cluster labels and the actual survival time is evaluated. **HCsnip** uses the Random Survival Forest (RSF)[20] for this purpose. The parametric model, eg, the Cox model, seems inappropriate here because, if a cluster is purely composed of nonevent patients, which is what actually we are after, the variance of the coefficient will be inflated, thus making the *P*-value unreliable.

The function *RSF_eval* in this package is designed to serve this end. This function constructs the survival forest (SF) using the training samples' follow-up as response and the cluster labels as covariates. The constructed forest is used to predict the test samples' survival time based on that of the cluster labels. The predicted survival times are compared with the actual ones by *C.index* to calculate the error rate. "Calculating the error rate on the test set using the Harrell's concordance index, which is related to the area under the receiver operating characteristic curve, can be interpreted as a misclassification probability."[20] The error rate ranges from 0 to 1, 0 being the perfect match. A low error rate, ie, high concordance between the predicted survival and actual ones, indicates that the partition found in the training phase is highly extrapolatable to new patients.

RSF_eval has the following input arguments:

- **partition:** Cluster labels of the training set samples.
- **surv.time:** A numeric vector that contains patients' survival times.
- **status:** A binary vector (0, 1) that contains patients' survival status indicator, where 1 denotes an event (dead).
- **te.partition:** Cluster labels of the test samples for which error rate is to be calculated.
- **te.surv.time:** Actual survival time of the test samples.
- **te.status:** Actual survival outcomes of the test samples.
- **…** Additional argument for the *rfsrc* function from the **randomForestSRC** package.[21]

This function returns a numeric vector of error rates with lengths equaling the number of trees constructed. Default is 1000.

Visualization of cluster differences using samples' molecular entropy. To visually delineate the cluster differences in terms of the samples' molecular profiles, the function *EnvioPlot* in this package is designed to visualize the samples' molecular entropy using a violin plot. In the study of van Wieringen and van der Vaart (2010),[22] the effect of DNA copy number aberrations on the transcriptome has been investigated via the genomic and transcriptomic entropy of a cancer cell. In our context, entropy is used to measure the diversity (at the molecular level) of samples in a cluster. If a cluster is composed of samples with similar molecular signatures, one may, then, expect smaller entropy. Whereas in a cluster with samples of heterogeneous origin (high degree of dissimilarity), one may, then, expect large entropy. Interpretation of the entropy estimates depends on the input data type. If the input is gene expression data, large entropy corresponds to high level of overall expression of genes in the profile. If the input data comprise DNA copy number, large entropy corresponds to a large number of copy number aberrations in the profile.[22]

EnvioPlot has the following input arguments:

- **X:** An object of class *ExpressionSet* or high-dimensional molecular data matrix from which "parti" has been obtained, eg, gene expression. Columns are assumed to represent the samples, and rows represent the samples' features. Missing values are allowed.
- **method:** The type of method used to calculate the sample's molecular entropy. Either "knn" (default) or "normal." See the *hdEntropy* function in the **sigaR** package.
- **parti:** A partition for which the violin plot is to be made.
- **horizontal:** Should boxes be organized horizontally? Its default value is set to "FALSE."
- **col:** A vector of colors for each cluster. Length should be equal to the number of clusters in "parti."
- **names:** A vector of cluster labels.

- **…** Additional arguments for the *hdEntropy* function in **sigaR** package.

This function returns a numeric vector containing the entropy estimate for each sample.

Optimal treatment assignment using HCsnip. Besides being a semi-supervised tool that extracts clusters from the HC tree, **HCsnip** package can be used in optimal treatment assignment. The setting is as follows: molecular profiles and survival information of two groups of patients treated with two different drugs are available. We wish to evaluate the potential use of our method for better assignment of one of the two treatments to new patients using cross-validation. Specifically, for each left-out test sample (representing a new patient), a HC tree is constructed for both treatment groups separately, and the new patients' molecular profiles (assuming that new patients' group labels and survival information are missing) are compared with the optimal partitions extracted from each of the two HC trees (Fig. 5). The new patient is then assigned to a cluster in each of the two trees separately using the previously described approach. Finally, the actual survival data of the members of the two clusters (eg, in Figure 5 *clust*1 and *clust*3) are used to decide which of the two treatments is predicted to be the most beneficial for the new patient.

HCsnip includes the following two functions to assign new patients to one of the multiple groups in an optimal manner and to objectively evaluate the quality of overall assignment.

TwoHC_assign. This function constructs a HC tree and extracts the optimal partition for each treatment group separately using the semi-supervised approach previously described. When a new patient arrives for whom only the molecular profile is given, *PNN* + *Concordance* is utilized to find a cluster (competing cluster) in each optimal partition with which the new sample is most similar. Finally, the function examines the follow-up information of the two competing clusters and assigns the new sample to the cluster that has better overall survival. The new sample is recommended to receive the treatment that has been administered to the group from which the winner cluster is obtained. A complete workflow is portrayed in Figure 5.

The *TwoHC_assign* function has the following input arguments:
TwoHC_assign (X, index1, index2, new.X, dis.method = "cor", link.method = "ward", minclus = 4, maxmiss = 30, surv.time, status, method1 = "BIC", method2 = "g2")

- **X:** An object of class *ExpressionSet* or data matrix in which columns are assumed to represent the samples, and rows represent the samples' features. It must include the data sets from which the two HC trees need to be derived. Missing values are allowed.
- **Index1:** Indexes of the columns in "X"; corresponds to the samples in the first treatment group.
- **Index2:** Indexes of the columns in "X"; corresponds to the samples in the second treatment group.
- **new.X:** An object of class *ExpressionSet* or data matrix in which columns are assumed to represent the samples and rows represent the samples' features. It must include the data set that corresponds to new samples. Missing values are allowed.
- **dis.method:** The distance measure to be used. This must be one of the methods acceptable for the *dist* function from the **stats** package or the Pearson correlation "cor" (default).
- **link.method:** The agglomeration method to be used. This should be one of "ward" (default), "single," "complete," "average," "mcquitty," "median," or "centroid."
- **minclus:** The minimum number of samples allowed to form a cluster.
- **maxmiss:** Maximum allowed percentage of missing values per row in "X."
- **surv.time:** A numeric vector that contains patients' survival times in "X."
- **status:** A numeric binary vector (0, 1) that contains patients' survival status indicator in "X," where 1 denotes an event (dead).
- **method1:** The type of measure desired to assess the quality of a partition in terms of follow-up. Default is "BIC."
- **method2:** The type of measure desired to assess the quality of a partition in terms of data matrix "X." Default is Goodman–Kruskal index "g2."

This function returns an object-of-class list. The list contains slots that report most of the arguments provided in the function call, as well as the following slots:

- **hc1:** HC tree corresponding to the first treatment group.
- **hc2:** HC tree corresponding to the second treatment group.
- **partitions.hc1:** A matrix containing the partitions extracted from "hc1"; rows represent the partitions, each of which is composed of exhaustive and mutually exclusive clusters, and columns represent the samples.
- **partitions.hc2:** A matrix containing the partitions extracted from "hc2"; rows represent the partitions, each of which is composed of exhaustive and mutually exclusive clusters, and columns represent the samples.
- **best.hc1:** The optimal partition retrieved from "hc1."
- **best.hc2:** The optimal partition retrieved from "hc2."
- **score.hc1:** A matrix with two columns; the first column contains the quality scores of partitions in "partitions.hc1," which have been calculated using the follow-up data; the second column contains the quality scores calculated using "X."
- **score.hc2:** A matrix with two columns. The first column contains the quality scores of partitions in "partitions.hc2," which have been calculated using the follow-up

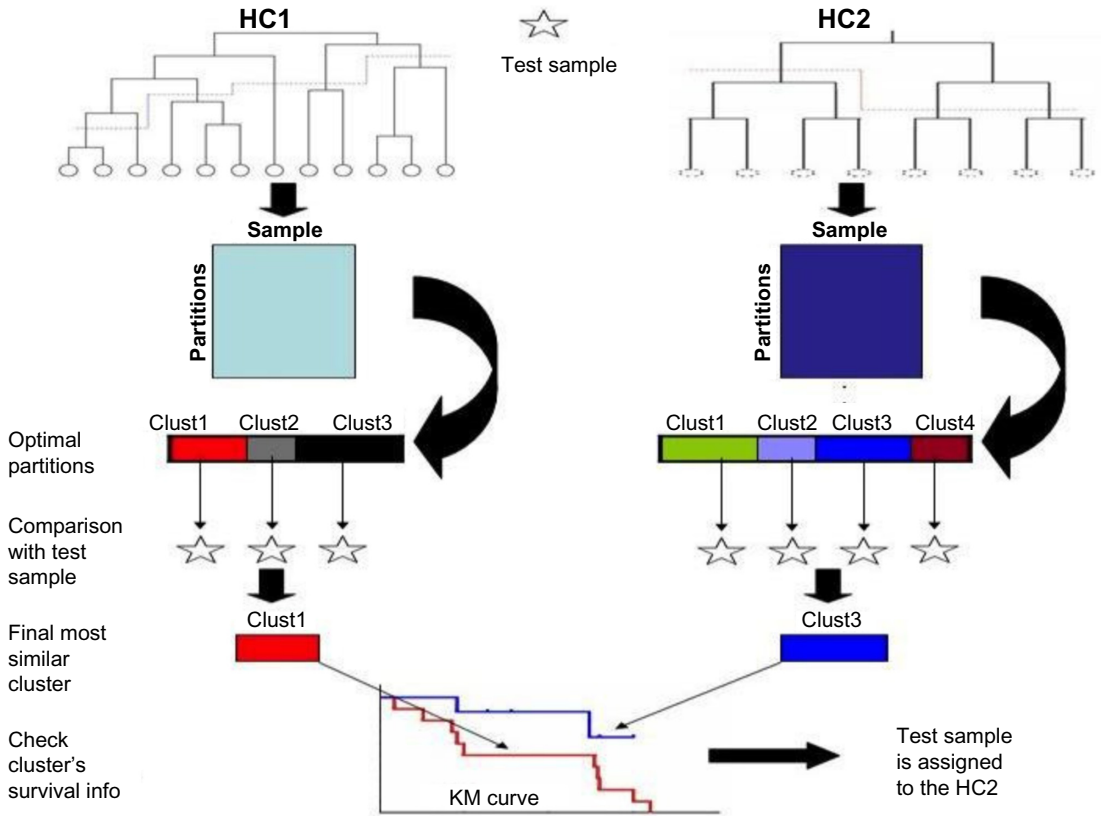

Figure 5. Schemata of group assignment from *TwoHC_assign* function for a new sample.

data; the second column contains the quality scores calculated using "X."

- **Assign:** A matrix with three columns. The first column contains the indexes of groups to which new samples have been assigned; the second column contains the indexes of the competing clusters in "best.hc1"; the third column contains the indexes of the competing clusters in "best.hc2."

TwoHC_perm. This function is designed to assess whether the predicted gain in survival is significant when the treatment assignment from *TwoHC_assign* is taken. For a new patient, the predicted gain in survival is quantified by the absolute value of the estimated treatment parameter $\hat{\beta}_{\text{obs}}$ in the Cox model, which compares the survival times of the patients in the two competing clusters. On its turn, it is transformed to $rr_{\text{obs}} = \exp\left(-|\hat{\beta}_{\text{obs}}|\right)$, which quantifies the gain in the relative risk in the Cox model. However, this results in a biased estimate, as *TwoHC_assign* already has been used $\hat{\beta}_{\text{obs}}$ in the clustering on the new samples. Thus, even when the two treatments would be equally good for the genomic profiles in the two competing clusters, we obtain $\text{rr}^i_{\text{obs}} < 1$. To resolve this issue, the permutation argument is utilized to correct for this bias. Specifically, for each new sample i, the survival data of the two competing clusters are the permuted p times. Then, for each permutation, j $\text{rr}^i_{\text{perm},j}$, which contains the same bias as rr^i_{obs}, is recalculated. After p permutations, the P-value is calculated in the following manner:

1. Calculate the test statistics for the observed relative risk ratio using their geometric mean (suits well for the ratio scaled data)

$$\text{rr}^i_{\text{obs}} = \frac{\exp\left(-|\hat{\beta}^i_{\text{obs}}|\right)}{Z_i}$$

$$T_{\text{obs}} = \frac{\frac{1}{n}\sum_{i=1}^{n}\log\left(\text{rr}^i_{\text{obs}}\right)}{\text{stdev}\left(\log\left(\text{rr}^1_{\text{obs}}, \text{rr}^2_{\text{obs}}, \ldots, \text{rr}^n_{\text{obs}}\right)\right)}$$

where

$$Z_i = \exp\left(-\text{median}\left(|\hat{\beta}^{1i}_{\text{perm}}|, |\hat{\beta}^{2i}_{\text{perm}}|, \ldots, |\hat{\beta}^{pi}_{\text{perm}}|\right)\right)$$

$\hat{\beta}_{\text{obs}}$ is a vector of length n containing the observed coefficients. $\hat{\beta}_{\text{perm}} \in R^{p\times n}$ is a matrix containing the coefficients generated by permutations.

2. Generate test statistics from the null distribution. For $j = 1, \ldots, p$:

$$\text{rr}^j_{\text{perm}} = \frac{\exp\left(-|\hat{\beta}^{ji}_{\text{perm}}|\right)}{Z_i}$$

$$T^{j}_{\text{perm}} = \frac{\frac{1}{n}\sum_{i=1}^{n}\log\left(\text{rr}^{ji}_{\text{perm}}\right)}{\text{stdev}\left(\log\left(\text{rr}^{j1}_{\text{perm}}, \text{rr}^{j2}_{\text{perm}}, \ldots, \text{rr}^{jn}_{\text{perm}}\right)\right)}$$

3. Finally, compute the *P*-value of interest using the following equation:

$$P\text{-value} = \frac{\sum_{j=1}^{p} I\left(T_{\text{obs}} \geq T^{j}_{\text{perm}}\right)}{p}$$

where $I(\cdot) = 1$ if the argument is true and = 0 otherwise.

A call to this function requires the output from *TwoHC_assign* and the number of permutations desired ("nperm"). The function returns an object-of-class list with the following slots:

- **Obs.betas:** A numeric vector that contains the observed coefficients corresponding to the new samples.
- **Perm.betas:** A matrix that contains the coefficients that have been generated by the permutations. The columns correspond to samples, rows correspond to permutations.
- **Ranks:** A numeric vector that contains the ranks of the observed coefficients among the "nperm" coefficients that have been generated by permutations.
- **RiskRatios:** A numeric vector that contains the ratio of relative risks of the new samples.
- ***P*-value:** The *P*-value for the overall group assignment.

Examples

This section is dedicated to the illustration of the use of functions in **HCsnip** with built-in data sets. The illustrations are geared toward such a case: the HC is derived from molecular data and patients' time-to-event data are used as background information. Two publicly available data sets are included in this package. The leukemia data set[23] contains the subset of gene expression profiles of adult acute myeloid leukemia patients. It contains the measurements of 116 samples on 1571 genes, and follow-up data are also included. The complete data set is available at the Gene Expression Omnibus (www.ncbi.nlm.nih.gov/geo/) with accession number GSE425. The second data set is a subset of the latest version of The Cancer Genome Atlas (TCGA) glioblastoma multiforme (GBM) level 3 gene expression data with partial clinical info.[24] It contains the expression data of 120 samples on 3000 genes. The clinical data include patient follow-up and the types of drugs that patients have been administered. The complete data set is available at the TCGA data portal (https://tcga-data.nci.nih.gov/tcga/).

HCsnipper. Here, we constructed a HC tree using the first 30 samples in the leukemia data set and retrieved all possible partitions using *HCsnipper*. Except for *minclus* = 5, default settings are applied to the remaining parameters. We use R package **WGCNA** to display the partitions. This package can be directly installed using install.packages("WGCNA").

```
library("HCsnip")
library("WGCNA")
data(BullingerLeukemia)
attach(BullingerLeukemia)
res <- HCsnipper(em[, 1:30], minclus = 5)
cl <- res$partitions
plotDendroAndColors(res$hc, t(cl), hang = –1, main = "",
dendroLabels = FALSE)
```

The function yielded eight partitions that are shown in Figure 6. It is clear that the partitions 4–7 cannot be discovered if the standard fixed-height cut is applied.

Generate partition quality scores. In order to select the optimal partition, here, each partition is evaluated using the given expression and the follow-up data, separately.

```
a <- apply(cl, 1, function(x) measure(parti = x,
dis = 1–cor(em[, 1:30])))
b <- apply(cl, 1, function(x) surv_measure
(x, surv.time[1:30], status[1:30]))
```

Select an optimal partition. Now, the optimal partition will be obtained using the perm_test function. The perm_test selected a partition with two clusters, the eighth partition in Figure 6, as the optimal one.

```
result <- perm_test(cl, surv.time[1:30], status[1:30], score1
= a, score2 = b, nperm = 10)
table(result$best)
```

Clustering on the new samples using the predefined partition. Here, we take 50 new samples and assign them to one of the two clusters in the optimal partition obtained in the previous step. The function *cluster_pred* induced two clusters

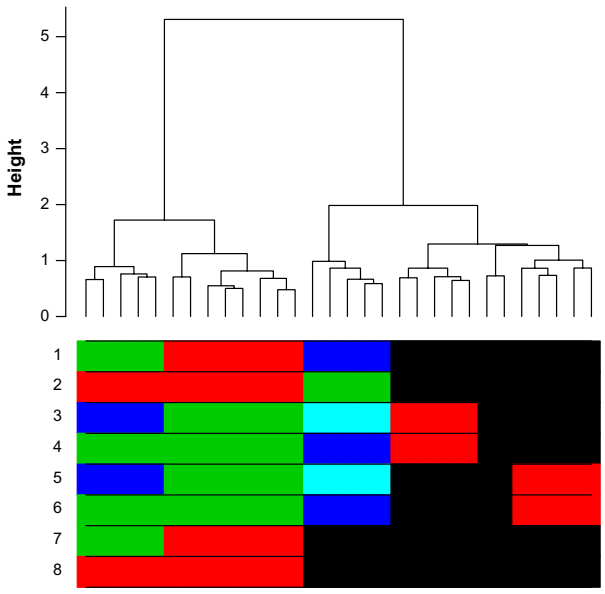

Figure 6. The HC tree that is derived from the first 30 samples' expression profiles in the Leukemia data set. Partitions induced by the piecewise snipping are displayed at the bottom of the HC tree.

on the new samples, and their difference in terms of follow-up is visualized in Figure 7. Observe that the two clusters are composed of patients with substantially different survival outcomes (P-value = 0.01). This shows that the optimal partition is highly extrapolatable for new samples.

```
pred <- cluster_pred(X = em[, 1:80], partition = result
$best,
surv.time = surv.time[1:30], status = status[1:30],
te.index = 31:80, te.surv.time = surv.time[31:80],
te.status = status[31:80], plot.it = TRUE)
```

Visualization of cluster differences using samples' molecular entropy. For visual inspection, the difference between the two clusters induced on the 50 new samples in terms of their expression profiles is visualized via the *Envio Plot* function. As we observe from Figure 8, the samples in the first cluster exhibit somewhat higher entropy (although nonsignificant) than the samples in the second cluster, which indicates that the two clusters are relatively similar at the molecular level. In order to visualize the observed difference in the entropy levels, we project the gene expression profiles of these 50 samples on the first two principal components. The projected data are plotted in Figure 9. It is apparent that the samples in the first cluster are much more scattered out, indeed indicating a large entropy. Note that, albeit including samples with similar expression profiles, the two clusters exhibit considerably different survival outcomes. Due to its semi-supervised nature, **HCsnip** successfully identified these two groups.

```
H <- EnvioPlot(X = em[, 31:80], parti = pred$St[, 2])
gr <- pred$St[, 2]
pc <- princomp(em[, 31:80])$loadings[, 1:2]
a <- pc[which(cls = = 1), 1]
b <- pc[which(cls = = 1), 2]
par(mfrow = c(1, 2), mar = c(10, 4, 10, 1))
plot(a, b, xlim = range(pc[, 1]), ylim = range(pc[, 2]),
pch = 19, col = 2, xlab = "PC1", ylab = "PC2")
ch <- chull(a, b)
ch <- c(ch, ch[1])
lines(a[ch], b[ch])
a <- pc[which(cls = = 2), 1]
b <- pc[which(cls = = 2), 2]
plot(a, b, xlim = range(pc[, 1]), ylim = range(pc[, 2]),
pch = 15, col = 3, xlab = "PC1", ylab = "PC2")
ch <- chull(a, b)
ch <- c(ch, ch[1])
lines(a[ch], b[ch])
```

Error rate calculation for the new set. Finally, we investigate whether the clustering induced on the test set yields survival estimates that are reasonably concordant with the actual survival data. If the clustering on the training set well represents the input data structure, which is subsequently reflected on the clustering on the test set, one may expect smaller error rates from the *RSF_eval* function. The error rate 0.48 indicates that the survival estimates are relatively concordant with their actual data.

```
Err <- RSF_eval(result$best, surv.time[1:30], status[1:30],
pred$St[, 2], surv.time[31:80], status[31:80])
mean(Err)
```

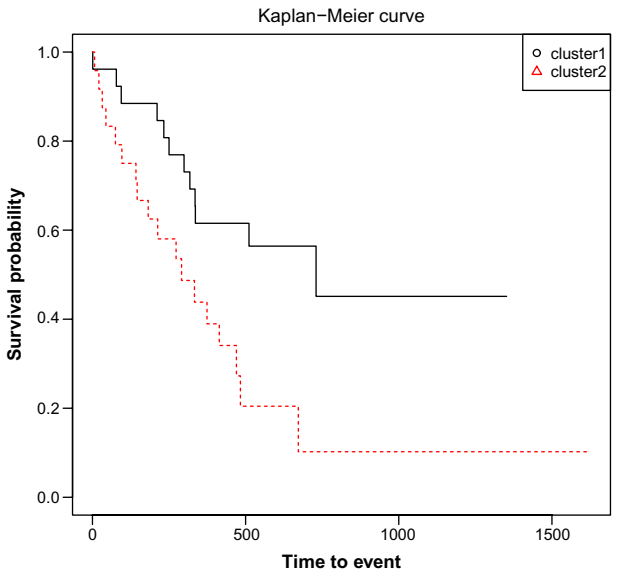

Figure 7. Kaplan–Meier survival analysis by clusters that were induced on the new samples.

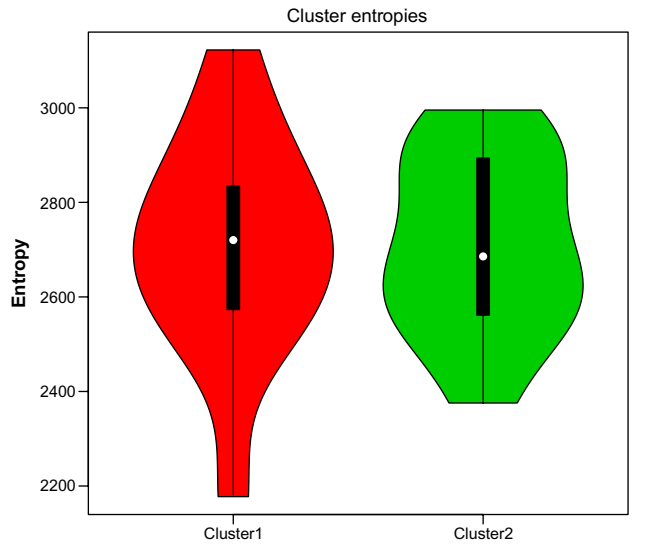

Figure 8. The gene expression entropy distributions of the two clusters that were induced on the 50 new samples. The shape of the violin plot shows the distribution of the gene expression entropy values within each cluster and the white dot in each denotes the mean entropy level. Distributions are relatively different between cluster1 and cluster2, with a larger mean gene expression entropy value observed in cluster1.

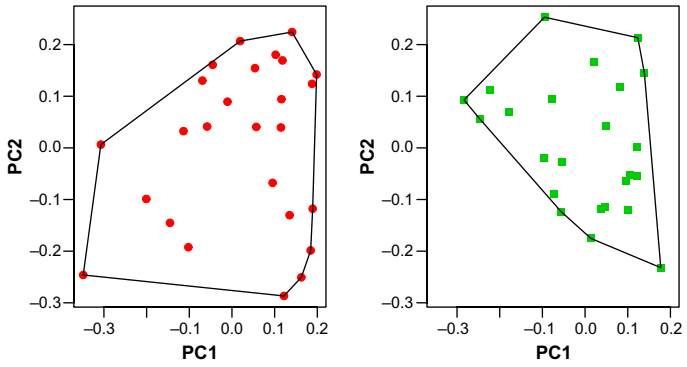

Figure 9. PCA plot. The 50 samples shown in the 2D plane spanned by their first two principal components. This type of plot is useful for visualizing the cluster homogeneity. The smaller the connected area, the more similar are the samples within the clusters. The scatter plots show that there is less within-cluster variation in cluster2 (right), which is indicative of a higher level of within-cluster homogeneity resulting from low within-cluster gene expression entropy. In cluster1(left), the samples are more spread out, indicative of a higher degree of within-cluster variation resulting from a higher within-cluster gene expression entropy.

Optimal treatment assignment using HCsnip. We close the "Examples" section by applying **HCsnip** to an application outside the realm of clustering: optimal treatment assignment. The latest version of TCGA GBM[24] data set included in the package is used for this illustration. GBM is the most common and most aggressive type of malignant primary brain tumor in humans. The level 3 gene expression and clinical data up to January 2013 were downloaded from the TCGA data portal. We selected patients treated with Avastin® (bevacizumab) and Temodar® (temozolomide). The setup for this illustration is as follows: we select 30 samples from each treatment group (training set) and derive two HC trees separately. **HCsnip** is applied to each HC tree separately to extract an optimal partition. Then, 60 new samples are selected (30 from each group) to form a test set. The function *TwoHC_assign* is used to assign the test set to one of the treatment groups in the semi-supervised manner. The significance of treatment assignment on the test set is evaluated using *TwoHC_perm* with 100 permutations.

From Figure 10A, we observe that large numbers of samples have bias-corrected relative risk ratios smaller than one. This means these samples have reduced relative risks that are better than that of random. The reduction of risk with respect to current treatment assignment, as quantified by rr_{obs}, is highly significant (Fig. 10B). The R scripts for realization of this illustration are provided in the Supplementary File.

Illustrations with Real-World Data Sets

In this section, we illustrate the performances of **HCsnip** with multiple real-world data sets that are not included in the package. These illustrations aim to (1) compare the quality of the optimal partitions produced by **HCsnip** and the standard fixed-height cut method; (2) test the value of utility of **HCsnip** in optimal treatment assignment application. The results reported herein are meant to offer, to some extent, unique insights into the novel aspects of semi-supervised snipping, central to **HCsnip**.

HCsnip vs fixed-height cut. The latest version of GBM[24] gene expression data and DNA copy number data with partial clinical information are used for this comparison. Details of array platforms, preprocessing, etc are provided in Appendix A. To make a thorough comparison, the following three scenarios are considered:

- The HC tree is derived from DNA copy number data, and gene expression data are used as background information.
- The HC tree is derived from DNA copy number data, and time-to-event data are used as background information.
- The HC tree is derived from gene expression data, and time-to-event data are used as background information.

Main data: DNA copy number; background info: gene expression. We derived a HC tree from DNA copy number data using the dedicated R package **WECCA**[25] with parameters "ordinal" and "heterogeneity." We use "GK" to measure

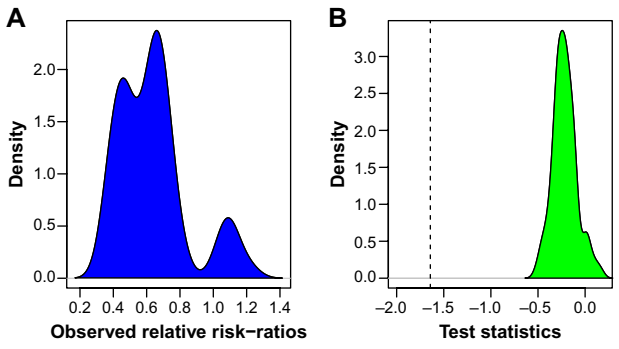

Figure 10. (**A**) The density of the observed relative risk ratios (rr_{obs}). (**B**) The distribution of the test statistics from the null model obtained by permutation (T_{perm}).
Note: The black broken line denotes the location of the observed test statistics.

the cluster quality in both data settings. The HC tree and the optimal partitions retrieved by the two approaches are displayed in Figure 11. **HCsnip** returned an optimal partition that comprises 13 clusters (*cluster.p*), the fixed-height cut approach; on the other hand, **HCsnip** generated a partition with two clusters (*cluster.s*). To examine the usefulness of the extracted clusters from the two approaches, their correlations with follow-up data are investigated. Analysis shows that the differences in survival between *cluster.p* are more significant (unbiased *P*-value = 0.093) than those in *cluster.p* (unbiased *P*-value = 0.439).

To assist in visually inspecting the differences among clusters, we summarize them 1) in terms of follow-up data; 2) in terms of samples' molecular entropy values. As observed from Figures 12 and 13, the clusters that are induced by the fixed-height cut approach contain many subclusters that are different in terms of both their survival outcomes (some of them are highly significant) as well as their molecular profiles. Particularly, *cluster.p*6 and its two neighbors *cluster.p*7 and *cluster.p*8 are noteworthy. The samples in *cluster.p*6 exhibit better overall survival than the samples in the latter two. The entropy distributions in these three clusters (Fig. 13) further confirm their dissimilarities. However, the fixed-height cut approach fails to "see" them and merge them into one cluster.

Main data: DNA copy number; background info: time-to-event. Performances of the two approaches are evaluated and compared through a split-sample validation procedure. Specifically, the original data set size *N* becomes a parent population from which 70% samples are randomly drawn without replacement and used for training, and the remaining samples are used for testing purpose. HC is derived and the optimal partition is selected using the training set. Each test sample is assigned to one of the clusters found in the training phase. To obtain reliable results, we repeat this procedure 100 times. Note that, to make the comparison evenhanded, the optimal partition from the fixed-height cut approach is selected in the same manner as in **HCsnip**. Results from the two approaches when different cluster quality measures were used are presented in Table 1.

Observe that, irrespective of the cluster quality measures used, **HCsnip** produced clusterings in which clusters are superior to those produced by the fixed-height cut.

Main data: gene expression; background info: time-to-event. Similar phenomena are observed in this setting as well. The superiority is more pronounced when *PNN + Concordance*

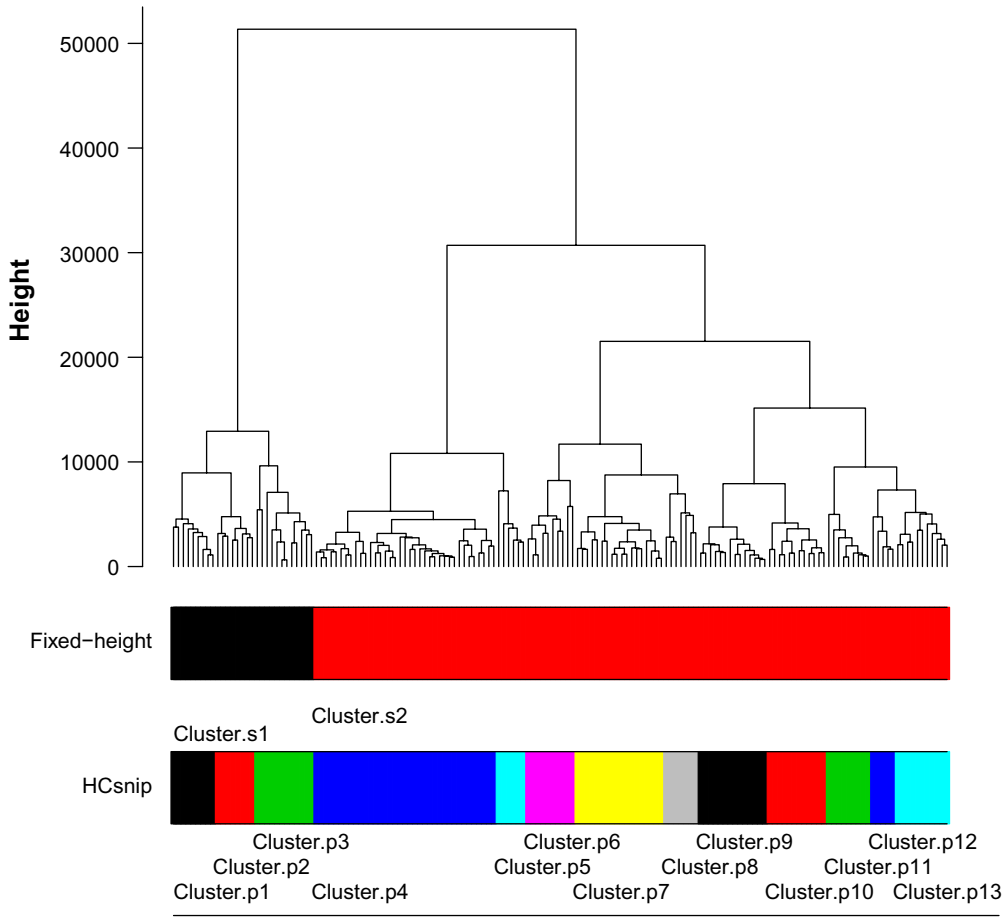

Figure 11. The HC tree that has been derived by using the DNA copy number profiles in the GBM data set. The optimal partition extracted by the two approaches is displayed at the bottom.

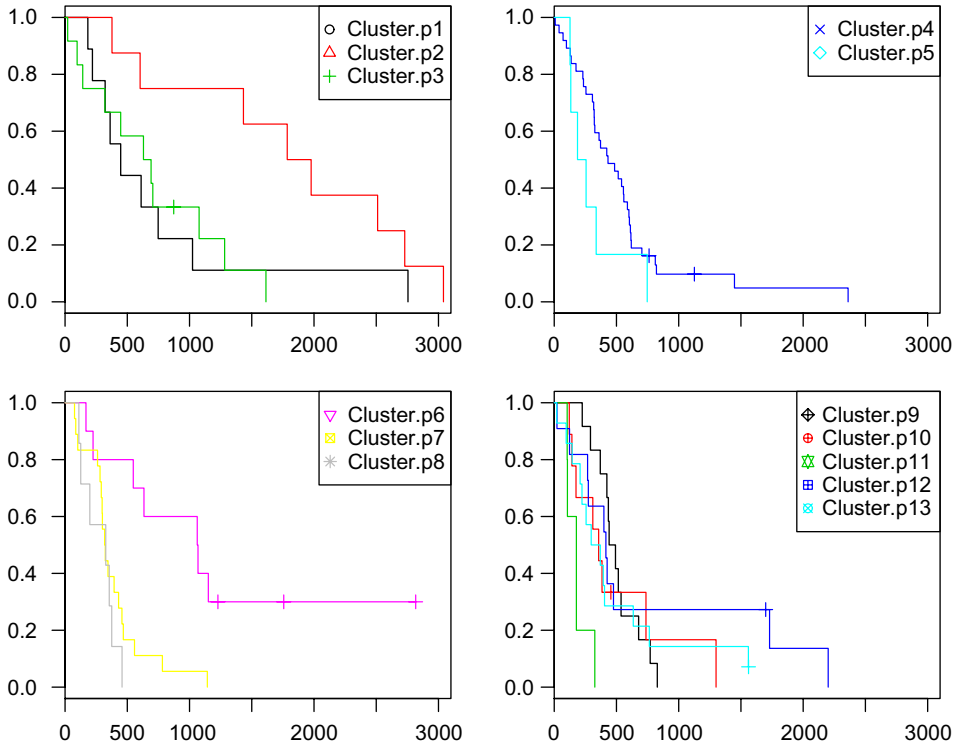

Figure 12. Kaplan–Meier survival analysis by clusters displayed in Figure 11. For visualization purpose, clusters in each branch are plotted separately.

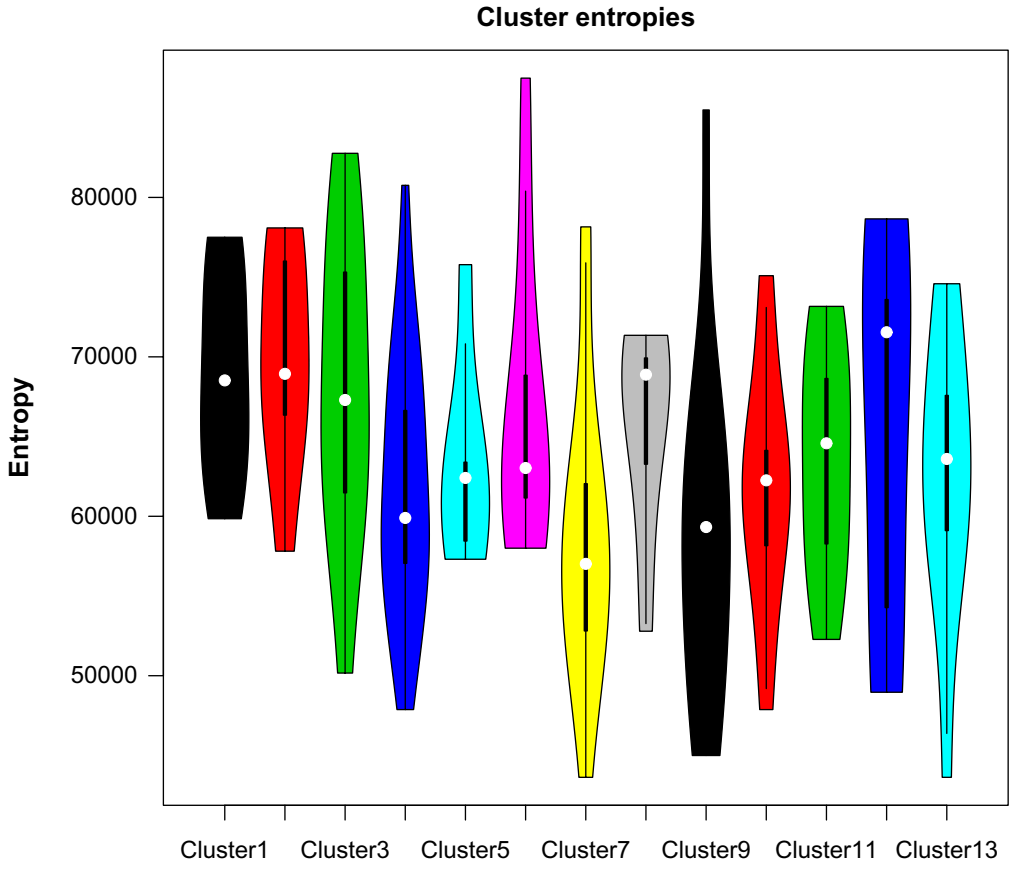

Figure 13. Gene expression entropy distribution by clusters displayed in Figure 11. The shape of the violin plot shows the distribution of the gene expression entropy values within each cluster, and the white dot in each denotes the mean entropy level.

Table 1. Cross-validation results when the main data comprise DNA copy number. The column "Measure" specifies the measure used for the main data; the last two columns denote the results when applying the two different methods used in clustering on the test set; the subcolumns "AIC" and "BIC" denote the results when two different measures are used for the follow-up data. Numbers denote the number of times **HCsnip** produces smaller error rates than that produced by the fixed-height cut in 100 repetitions, and the opposite holds for numbers in parentheses. Note that, for some values, the two numbers do not sum to 100. This is because in some repetitions, the two approaches produce ties.

MEASURE	WARD		PNN + CONCORDANCE	
	AIC	*BIC*	*AIC*	*BIC*
WSS	66(34)	69(31)	63(37)	62(38)
C-index	65(35)	72(28)	60(39)	55(45)
GK	59(35)	46(31)	47(46)	48(30)

Table 2. Cross-validation results when the main data comprise gene expression profiles. The column "Measure" specifies the measure used for the main data; the last two columns denote the results when applying the two different methods used in clustering on the test set; the subcolumns "AIC" and "BIC" denote the results when two different measures are used for the follow-up data. Numbers denote the number of times **HCsnip** produces smaller error rates than the fixed-height cut approach in 100 repetitions, and the opposite holds for numbers in parentheses. Note that, for some values, the two numbers do not sum to 100. This is because in some repetitions, the two approaches produce ties.

MEASURE	WARD		PNN + CONCORDANCE	
	AIC	*BIC*	*AIC*	*BIC*
WSS	63(37)	62(38)	65(35)	84(16)
C-index	52(48)	56(43)	66(33)	79(21)
GK	60(39)	48(51)	63(36)	80(19)

is used for the test sample assignment (Table 2). This is not unexpected because the follow-up data exhibit strong correlation with the expression data (P-value = 0.005) than with the copy number data.

The comparisons with the standard fixed-height cut approach in different scenarios show that the superior performances of **HCsnip** are not an artifact of the larger search space, and the superior performance remains when evaluated on independent test samples.

Optimal treatment assignment. Besides being a tool for decomposing the HC into meaningful clusters, **HCsnip** may be used in clinical practice, especially in making medical decisions that need to be tailored to the individual patient, known as "personalized medicine." We illustrate this by the ovarian serous cystadenocarcinoma (OV) data sets,[26] which can be the basis for further research. The latest version of OV gene expression data with partial clinical information has been downloaded from TCGA data base. Details of the data set are provided in Appendix A (including preprocessing details). We compare our optimized strategy with the original treatment assignment. Because TCGA does not require the Tissue Source Sites to complete the treatment form, we were not able to retrieve information regarding why a patient has been administered a particular drug. Still, we may assume that the original treatment assignment was done under usual care. Hence, comparison of the predicted performance under our treatment assignment with the original one is relevant.

The performance of **HCsnip** is evaluated through leave-one-out cross-validation, ie, we leave out one patient and construct a HC tree for each treatment group using the remaining patients' expression profiles. Partitions are extracted and, using the survival information, the optimal partition is selected for each HC tree. In each of the two partitions, the cluster that matches the expression profile of the left-out patient best is determined using *TwoHC_assign*. Finally, the survival patterns of the other patients in those two clusters are compared with a two-group Cox model, and the left-out sample is given the group label of the HC tree corresponding to the best (in terms of survival) cluster. Note that only the expression profile of the left-out sample is used during the whole process.

We observe that 107 (49%) samples have assigned labels that are discordant with the actual ones. Among these, 24 actually received carboplatin and 83 received taxol. The predicted improvement in survival when using the treatment assignment from *TwoHC_assign* instead of the actual one is tested through *TwoHC_perm*. For this illustration, we use $n = 107$ discordant samples, with $p = 10000$ permutations. Figure 14A shows the empirical distribution of $\mathrm{rr}^i_{\mathrm{obs}}$ for the 107 discordant samples. Considerable amount of samples have relative risk <1, which means, for these samples, the decrease in relative risk is smaller than the one in which patients are assigned to different treatment groups in random. Figure 14B shows T_{obs} against the background of its null distribution, as obtained by the permutations. The reduction of risk with respect to the current treatment assignment, as quantified by the bias-corrected risk ratio, is highly significant: P-value $<< 0.001$.

Note that we are not trying to overrule the previously made treatment decisions. These decisions may have been based on grounds other than efficacy alone, eg, toxicity. We simply use these data to illustrate the potential of our proposed methodology in a real-life application outside the realm of clustering. Because treatment is irreversible and the original decision strategy is unclear, a formal clinical trial-based comparison is not possible here. But, statistically, we demonstrate that, based on predicted survival, group assignments are better than the original one. Hence, this illustration, to some extent, does provide information on the usefulness of **HCsnip** in this type of application.

Conclusion

Nowadays, almost all major statistical computing products offer HC functionality. Consequently, desiderata for an

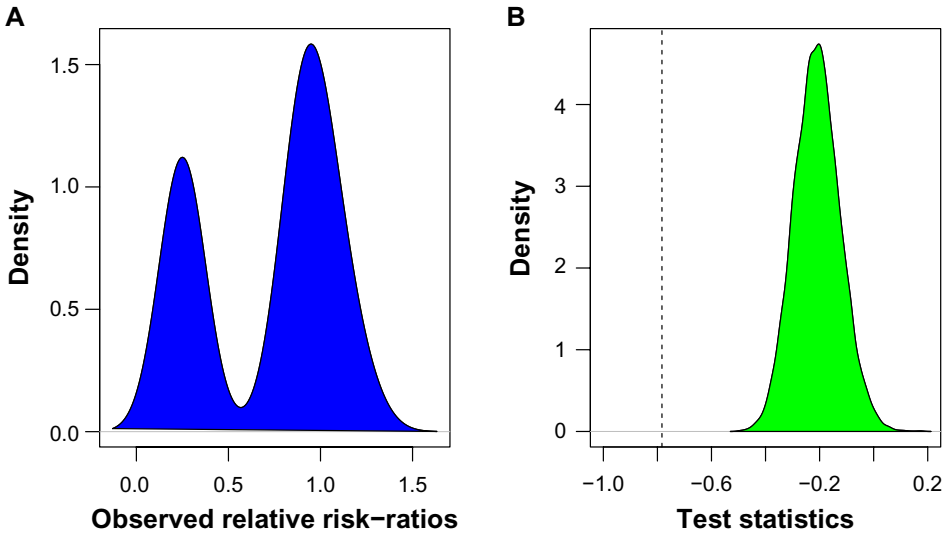

Figure 14. (**A**) The empirical distribution of rr_{obs}. (**B**) The distribution of the test statistics from the null model obtained by permutation T_{perm}.
Note: The black broken line denotes the location of the observed test statistics.

automatic procedure, which requires no user interaction, to extract clusters from a HC representation are surged.

We have described and implemented the **HCsnip** package, a novel semi-supervised HC tree-snipping procedure that combines available background information with molecular data to tease out clinically relevant cancer subtypes located deep down in the tree. To our knowledge, **HCsnip** is the first package that accepts patient follow-up data as background information without discretization. Application of **HCsnip** to high-dimensional molecular data seems to originate with the large amount of valuable information latent in data because 1) piecewise snipping is able to extract hidden clusters located deep down in the tree to unravel the true data structure; 2) fully supervised learning cannot, in practice, learn models with deep hierarchies. Say the data set under consideration has some patients who have relatively different survival outcomes than the remaining patients and located deep in the HC tree, and that their molecular profiles are characterized by only small numbers of features (common case in molecular data). Because of linearity, they may not be able to capture small changes in expression values of these features, thus failing to "see" this group of patients. Whereas cutting the HC tree at variable heights can be regarded as a nonlinear clustering that is, therefore, highly likely to identify this group.[4] Consequently, clustering returned by the piecewise cut may be a better predictor, although this is not the package is designed for, of survival prediction than the input data.

Consistent promising performances on built-in data sets and other multiple data sets with different setups corroborate the superiority of **HCsnip**. Usage of **HCsnip** in optimal treatment assignment perhaps does not exhaust the range of applications the package is capable of, which deserves to be explored. We believe the flexibility in handling different data types and many novel functions that the package comprises, make this package not just a respectable alternative to existing packages but, more often, one regarded as a preferred choice.

There are, however, still many areas open for further improvement. For instance, the cluster quality evaluation procedure currently does not take into account the heights at which clusters are formed. Clusters that are formed deep in the HC tree indicate that there is strong evidence in the data that samples in these clusters are most similar. One may incorporate this information into the partition evaluation process by giving a higher score to a partition in which clusters are composed of samples that are merged at comparatively lower height differences. If the two input data types (main data and background information) are of different importance levels, instead of simply summing the ranks of the two quality scores, one may combine them by giving different weights. Another extension would be to apply the instance-level constraints[27] to prefilter "dull" partitions (exploiting prior knowledge), which may prevent the algorithm from generating an optimal partition that is too much deviated from the user's expectation.

With ongoing research in analyzing heterogeneous high-dimensional molecular data sets, we believe **HCsnip** will be among the first to take advantage of new technology.

Author Contributions

AO performed data analysis and wrote both the R package and the manuscript. MvdW conceived this study and critically revised the manuscript. Both authors reviewed and approved the final manuscript.

Supplementary Material

Supplementary File 1. This file contains the R scripts for realization of Figure 10.

REFERENCES

1. de Souto M, Costa IG, de Araujo D, Ludermir TB, Schliep A. Clustering cancer gene expression data: a comparative study. *BMC Bioinformatics*. 2008;9:497.
2. Sørlie T, Perou C-M, Tibshirani R, et al. Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications. *Proc Natl Acad Sci USA*. 2001;98:10869–74.
3. Bair E, Tibshirani R. Semi-supervised methods to predict patient survival from gene expression data. *PLoS Biol*. 2004;2:511–22.
4. Obulkasim A, Meijer AG, van de Wiel AM. Semi-supervised adaptive-height snipping of the hierarchical clustering tree. *BMC Bioinformatics*. 2015;16(1):15.
5. Sboner A, Demichelis F, Calza S, et al. Molecular sampling of prostate cancer: a dilemma for predicting disease progression. *BMC Med Genomics*. 2010;3:8.
6. Langfelder P, Zhang B, Horvath S. Defining clusters from a hierarchical cluster tree: the dynamic tree cut package for R. *Bioinformatics*. 2008;24:719–20.
7. Navlakha S, White J, Nagarajan N, Pop M, Kingsford C. Finding biologically accurate clusterings in hierarchical tree decompositions using the variation of information. *J Comput Biol*. 2010;17:503–16.
8. Dotan-Cohen D, Melkman AA, Kasif S. Hierarchical tree snipping: clustering guided by prior knowledge. *Bioinformatics*. 2007;23:3335–42.
9. Bayá AE, Granitto PM. Clustering gene expression data with a penalized graph-based metric. *BMC Bioinformatics*. 2011;12:2.
10. Bolshakova N, Azuaje F. Machaon CVE: cluster validation for gene expression data. *Bioinformatics*. 2003;19:2494–5.
11. Hubert L, Schultz J. Quadratic assignment as a general data-analysis strategy. *Br J Math Stat Psychol*. 1976;29:190–241.
12. Goodman L, Kruskal W. Measures of associations for cross-validations. *J Am Stat Assoc*. 1954;49:732–64.
13. Hennig C. *fpc: Flexible procedures for clustering*. R package version 2.1–5. Available at: http://CRAN.R-project.org/package=fpc. Published 2013.
14. Kapp AV, Tibshirani R. Are clusters found in one dataset present in another dataset? *Biostatistics*. 2007;8:9–31.
15. Liang H, Zou G. Improved AIC selection strategy for survival analysis. *Comput Stat Data Anal*. 2008;52:2538–48.
16. Volinsky TC, Raftery AE. Baysian information criteria for censored survival models. *Biometrics*. 2000;56:256–62.
17. Obulkasim A, Meijer GA, van de Wiel MA. Stepwise classification of cancer samples using clinical and molecular data. *BMC Bioinformatics*. 2011;12:422.
18. Harrell FE, Califf RM, Pryor DB, Lee KL, Rosati RA. Evaluating the yield of medical tests. *JAMA*. 1982;247:2543–6.
19. Goeman J, Oosting J. *globaltest: Testing groups of covariates/features for association with a response variable, with applications to gene set testing*. R package version 5.14.0. Available at: http://bioconductor.org/packages/2.12/bioc/html/globaltest.html. Published 2012.
20. Ishwaran H, Kogalur UB, Blackstone EH, Lauer MS. Random survival forest. *Ann Appl Stat*. 2008;2:841–60.
21. Ishwaran H, Kogalur UB. *randomForestSRC: Random survival forests*. R package version 3.6.4. Available at: http://CRAN.R-project.org/package=randomForestSRC. Published 2013.
22. van Wieringen WN, van der Vaart AW. Statistical analysis of the cancer cells molecular entropy using high-throughput data. *Bioinformatics*. 2010;27:556–63.
23. Bullinger L, Döhner K, Bair E, Fröhling S, Schlenk RF, Tibshirani R. Use of gene-expression profiling to identify prognostic subclasses in adult acute myeloid leukemia. *N Eng J Med*. 2004;350:1605–16.
24. Verhaak R-G, Hoadley K-A, Purdom E, et al. Integrated genomic analysis identifies clinically relevant subtypes of glioblastoma characterized by abnormalities in PDGFRA, IDH1, EGFR, and NF1. *Cancer Cell*. 2010;17:98–110.
25. van Wieringen WN, van de Wiel MA, Ylstra B. Weighted clustering of array CGH data. *Biostatistics*. 2008;9:484–500.
26. Cancer Genome Atlas Research Network. Integrated genomic analyses of ovarian carcinoma. *Nature*. 2011;474:609–15.
27. Wagstaff K, Cardie C. Clustering with instance-level constraints. 2000. In: Proceedings of the Seventeenth International Conference on Machine Learning, 1103–10.
28. van de Wiel MA, Vosse S. *CGHcall: Calling aberrations for array CGH tumor profiles*. R package version 2.20.0. Available at: http://bioconductor.org/packages/2.12/bioc/html/CGHcall.html. Published 2012.
29. Yu T, Li J, Ma S. Adjusting confounders in ranking biomarkers: a model-based ROC approach. *Brief Bioinform*. 2011;13:513–23.

Appendix A: Data Sets

Glioblastoma multiforme data set

- *Sample type:* Glioblastome multiforme.
- *Molecular levels:* DNA copy number and gene expression.
- *Reference:* Verhaak et al (2010).[24]
- *DNA copy number platform:* 244 K Agilent Memorial Sloan–Kettering Cancer Center.
- *Gene expression platform:* Affymetrix 133A.
- *Number of samples*: 158.
- *Availability:* The Cancer Genome Atlas (TCGA) (http://cancergenome.nih.gov/)
- *Preprocessing:* Level 1 DNA copy number data with 160 samples with partial clinical information were downloaded. The Agilent copy number platform consists of 235834 probes; 223554 were available after preprocessing by using the package **CGHcall**.[28] The preprocessed data matrix was segmented and called using the same package. The called data matrix was regioned (to prepare a computer-generated hologram region object, which is the required input type), and the HC tree was obtained by using the package WECCA.[25] The functions used in different steps were called with default settings. The Affymetrix gene expression array contains 62980 probes. We followed the filtering approach described in Yu et al, (2011)[29] to reduce the probe set to 15750. Due to lack of follow-up information, two samples were deleted from the downstream analysis.

Ovarian cancer data set

- *Sample type:* Ovarian serous cystadenocarcinoma (OV).
- *Molecular levels:* gene expression.
- *Reference:* Spellman et al (2011).[26]
- Gene expression platform: 244 K Agilent.
- *Number of samples:* 218.
- *Availability:* The Cancer Genome Atlas (TCGA) (http://cancergenome.nih.gov/)
- *Preprocessing:* Level 3 gene expression data and full clinical data were downloaded. The Agilent gene expression platform consists of 17811 probes; 17440 were available after deleting missing values. We selected patients treated with carboplatin (109) and taxol (109). The former is a random selection of a total of 254 available samples to match group sizes of the two treatment groups.