Using clusterStab

James W. MacDonald

April 22, 2010

1 Overview

This document provides a brief guide to the *clusterStab* package, which is intended to be used for two things:

- Determining the number of clusters in a set of microarray data.
- Deciding how 'good' these clusters are.

Clustering microarray data and producing so-called heatmaps is a very common thing to do. However, there are some significant drawbacks to this technique. First, clustering is not an inferential technique, so there are no p-values associated with the heatmap to indicate how likely it is that the results arose simply by chance. Second, most clustering techniques are quite sensitive to the input data, so it is possible to get a very different result by simply adding or removing a single sample (or by adding or removing genes). This, of course, would result in a completely different interpretation. Third, clustering methods are intended to show previously unknown patterns in high dimensional data, but are often used to show *expected* patterns. This can result in a self-fulfilling prophecy, if for instance, the data are filtered based on an expected pattern. An example would be filtering the data based on the difference in expression between two sample types and then clustering the data. Since the filtered genes by definition are those that are different between the two groups, the heatmap will clearly show this difference. This is usually an uninteresting result, as permuting the sample labels, re-filtering the data and re-clustering will almost always result in a heatmap as striking as the original, yet will be based on sample groups that are not expected to be different.

This third point is quite important. In order to create a heatmap that is small enough to be useful, it is often necessary to filter out genes that are

not differentially expressed in any samples without unintentionally selecting genes that fulfill an *a priori* assumption about the number of clusters. A good approach is to select genes that have a high coefficient of variation (CV), which implies that there is a high variation in expression relative to the average expression for that gene.

When using hierarchical clustering to detect *a priori* unknown groups in data, there are thus two questions that arise; how many clusters are there, and given a number of clusters, how likely is it that they simply arose by chance? We answer both questions by relying on the fact that clustering solutions can be sensitive to changes in the input data. To test for the likely number of clusters, we repeatedly select subsets of the samples, cluster them, and then see for each number of clusters (from 2 to N) how often we get similar results. If the results are not often similar for a given number of clusters, it is not likely that there are that many clusters in our data. We assess similarity using a *Jaccard coefficient*, and use histograms to compare different numbers of clusters.

Once we have decided how many clusters we think there are, we can test to see how likely it is that this number came about simply by chance. Again, this is done by selecting subsets of the data and clustering. However, in this case we are selecting subsets of genes (using all samples). The basic idea here is the same; repeatedly subset the data and compare the resulting clustering solutions. If we keep getting the same clusters over and over, then this gives evidence that the cluster is stable, which implies that it is not likely that this result arose simply by chance.

Note that these functions are based on hierarchical clustering using the `hclust()` function, although they can be generalized to most clustering techniques. Incorporation of different clustering algorithms will be based on user requests.

2 A Simple Example

For this example we will be using the *fibroEset* package, which contains an `ExpressionSet` object with 46 samples and 12625 genes. There are 11 bonobo samples, 12 gorilla samples, and 23 human samples, so I would expect either two or three clusters, depending on how well the bonobo and gorilla samples cross-hybridize to Affymetrix HG-U95Av2 chips.

This is of course too many genes to cluster, so we will start by selecting only those genes that appear to be differentially expressed. Here I will use those genes that have a CV greater than 0.1. However, there are

many other possibilites that can be used; see the *genefilter* package for more suggestions.

```
> library(clusterStab)
> library(genefilter)
> library(fibroEset)
> data(fibroEset)
> exprs(fibroEset) <- log2(exprs(fibroEset))
> filt <- cv(0.1, Inf)
> index <- genefilter(fibroEset, filt)
> fb <- fibroEset[index, ]
> bh <- benhur(fb, 0.7, 6, seednum = 12345)
```

There are 721 genes for which the CV is greater than 0.1. We will now use these genes to decide how many different sample types (clusters) there are in our data, using the `benhur` function. The number of clusters is determined by taking random samples of the microarrays and then clustering. Similarity between pairwise clusters is estimated using a Jaccard coefficient; stable clusters will tend to have similar results on repeated sampling, whereas unstable clusters will tend to have very different results. This can be visualized using either histograms or empirical cumulative distribution function (eCDF) plots. We first look at the histograms using the `hist()` function.

Figure 1 shows individual histograms for each of the 2 - 6 possible clusters that were tested. Each histogram shows the distribution of the Jaccard coefficients for a particular number of clusters. What we are looking for is the histogram with the majority of the data at or near one. This gives the most likely number of clusters in our data. It appears that the most likely number of clusters is one, as each of the other histograms have data that are not clustered near one.

Figure 2 shows the dendrogram for this clustering result. The fibroEset data are Affy HG-U95av2 chips that were run on human, bonobo, and gorilla samples (labeled h, b, and g in the dendrogram). It is not surprising that the bonobo and gorilla samples cluster together considering that they were analyzed using a human chip. In addition, there are at least two smaller clusters within the human samples

We can also look at the eCDF plot, using the `ecdf()` function.

Figure 3 shows the eCDFs for each number of clusters. Again, it appears that there are probably only two real clusters here.

Once we have decided how many clusters there are in our sample, we

```
> hist(bh)
```

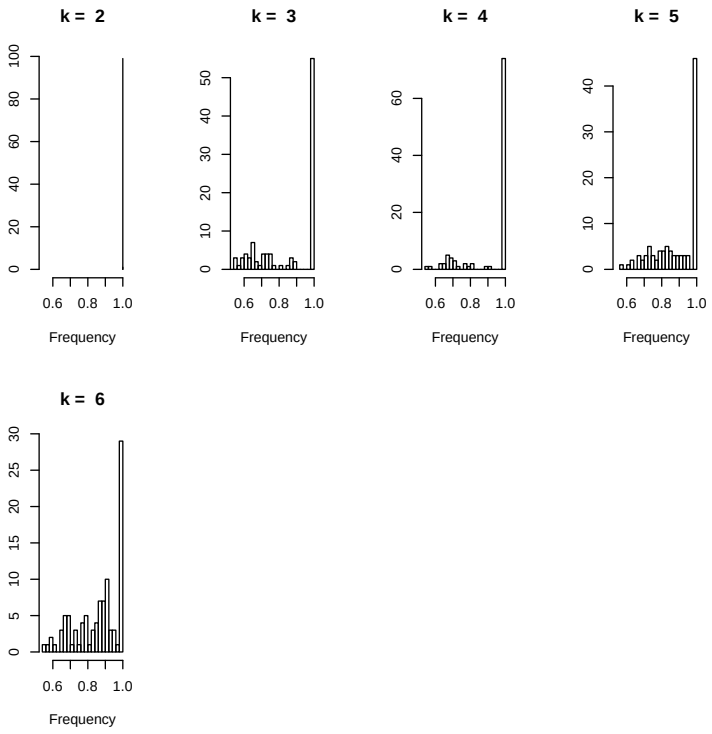

Figure 1: Histograms of the fibroEset data

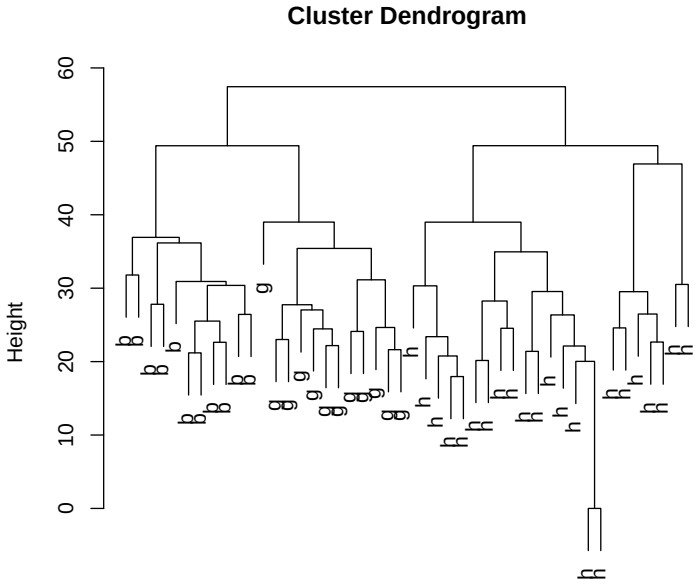

Figure 2: Cluster of the fibroEset data

```
> ecdf(bh)
```

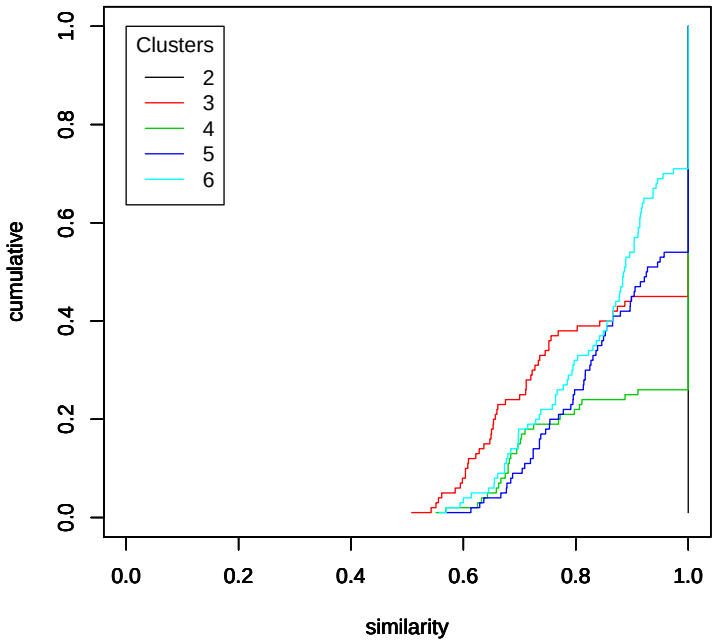

Figure 3: Empirical CDF plots of the fibroEset Data

can test to see how stable the clusters are. For this, we use the `clusterComp` function.

```
> cmp <- clusterComp(fb, 2)
> cmp

Results from running clusterComp:
                     1    2
Cluster stability:  100% 100%
Iterations: 100
Subsampling frequency: 80%
Agglomeration method: average
Original cluster membership:
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```