M3C: Monte Carlo Reference-based Consensus Clustering

Christopher R John

25/09/2019

Contents

1 Abstract

Genome-wide expression data is used to stratify patients into classes using clustering algorithms for precision medicine. The Monti consensus clustering algorithm is a widely applied method to identify the number of clusters (K), however when combined with the PAC score (Senbabaoglu et al., 2014); it has inherant bias towards greater values of K. While the original delta K metric to find K is often subjective as it relies on finding an elbow point and has been demonstrated to be inferior to the PAC score (Senbabaoglu et al., 2014). Both prior methods do not test the null hypothesis K=1. As a solution, we developed Monte Carlo reference-based consensus clustering (M3C) (John et al., 2018). M3C uses a Monte Carlo simulation to generate null distributions of stability (PAC) scores along the range of K which are used to decide the optimal value. Using a reference, that maintains the correlation structure of the input features, improves on the limitations of consensus clustering. M3C uses the Relative Cluster Stability Index (RCSI) and p values to decide on the value of K and reject the null hypothesis, K=1.

2 Prerequisites

M3C recommended spec

For the Monte Carlo simulation (method=1), M3C is best run on a relatively new and fast multi-core computer or cluster. If the machine is not particuarly fast, the Monte Carlo iterations parameter may be reduced to 5-50x.

The second method (method=2) that uses a penalty term to eliminate overestimation of K is faster, but the user will not get p values for each value of K.

M3C requires

A matrix or data frame of normalised continuous expression data (e.g. microarray, RNA-seq, methylation arrays, protein arrays) where columns equal samples and rows equal features. M3C's reference will work better if the feature data is approximately normally distributed across biological replicates.

For example, for RNA-seq data, VST or rlog transformed count data, log2(CPM), log2(TPM), and log2(RPKM), are all acceptable forms of normalisation.

The data should be filtered to remove features with no or very low signal, and filtered using variance to reduce dimensionality (unsupervised), or p value from a statistical test (supervised). We include a feature filter function that uses variance which is demonstrated later in the vignette. This variance function should be applied after removing the tendency of the variance to increase with the mean (e.g. after VST or log2 transformation).

Outliers should be removed, we include an easy to use PCA function which has a text label parameter to help remove outliers (type ?pca for more information).

We recommend M3C only be used to cluster datasets with high numbers of samples (e.g. 60-1000). M3C is mainly aimed at large patient cohort datasets such as those produced by TCGA and other consortia. Because of the high complexity of the algorithm and the type of consensus matrix it makes, it is not well suited for single cell RNA-seq data, better to use SC3 or Spectrum, for example.

M3C also accepts optionally

Annotation data frame, where every row is a patient or sample and columns refer to meta-data, e.g. age, sex, time until death, etc. M3C will automatically rearrange the annotation to match the clustering output and add the consensus cluster grouping to it. Note, this only works if the IDs (column names in data) match the entries in a column called "ID" in the user supplied annotation data frame.

M3C also accepts clinical or biological data for a basic statistical analysis with the discovered classes. Again the 'ID' column must exist in the annotation data frame. If the data is continuous or categorical, a Kruskal-Wallis or chi-squared test are performed, respectively. For these two tests, a 'variable' parameter must be given which is a string that defines the dependent variable in the users annotation data frame. If the data is survival data for cancer, the annotation data frame must have an 'ID' column first, followed by a 'Time' and 'Death' column (where 0 is no death and 1 is death or time until last visit).

3 Example workflow I: TCGA glioblastoma dataset

The M3C package contains the glioblastoma (GBM) cancer microarray dataset for testing (reduced to 50 samples randomly). The original optimal cluster decision was 4. First, we load M3C which also loads the GBM data.

```
library(M3C)
library(NMF)
library(gplots)
library(ggsci)
# now we have loaded the mydata and desx objects (with the package automatically)
# mydata is the expression data for GBM
# desx is the annotation for this data
```

3.1 Exploratory data analysis

This is an important exploratory step prior to running M3C. It is best to remove extreme outliers. It is also important to be aware of the assumptions of PAM, K-means, and HC. K means, PAM, and HC assume the following;

i) Clusters are approximately spherical - not severely elongated in one direction (anisotropic) or non-linear
ii) Clusters are approximately equal in variance

There is more info about the assumptions of clustering algorithms here: (http://scikit-learn.org/stable/modules/clustering.html)

Spectral clustering may be used to cluster more unusual structures using M3C, but this normally is not necessary. Ward's hierarchical clustering (HC) is faster than these 3 algorithms and is an option included in M3C, although in practice PAM and KM usually perform better.

It is also important to check that batch effects and other uninteresting sources of variation do not drive the main variation in the data. The PCA function has a labels flag for adding categorical and continuous variables to the below plot (type ?pca). For a more quantitative analysis of these variables before clustering there is the 'plot_drivers' function from the bioplotr package which produces a nice plot (https://github.com/dswatson/bioplotr).

The data should have been transformed to be homoskedastic (or nearly so) by this point (e.g. vst or log2), otherwise the larger features will have larger variances and dominate the PCA and cluster analysis. If the data contains different kinds of features, e.g. both protein and miRNA, then they should also be z-score normalised to be comparable.

```
PCA1 <- pca(mydata)
```

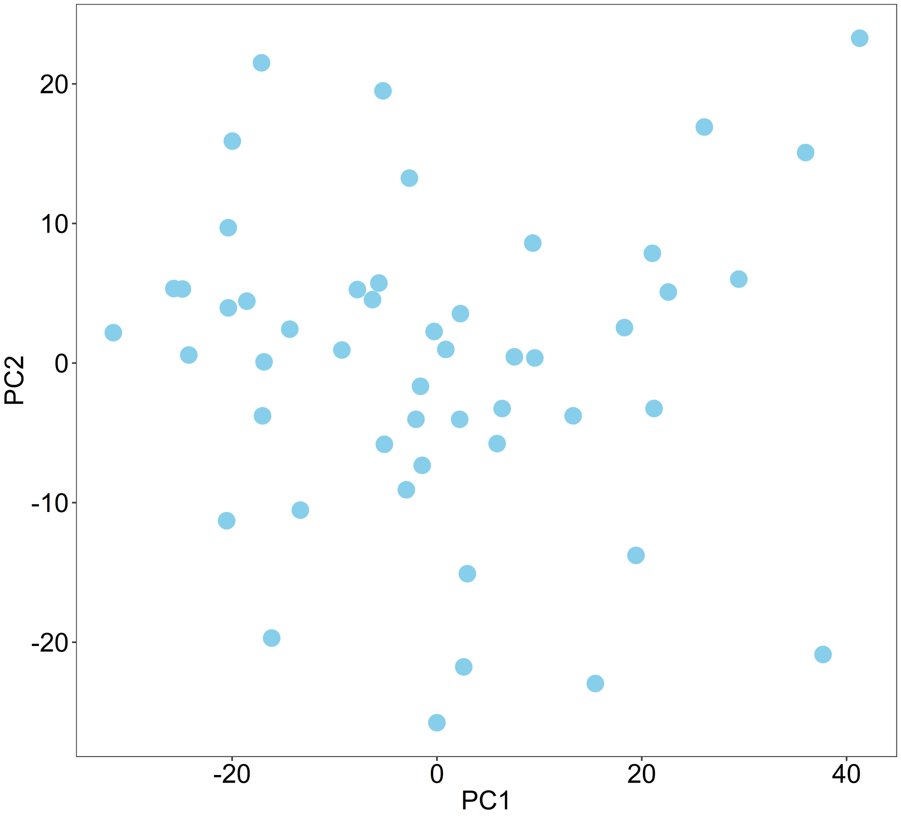

Figure 1: *Figure 1: Preliminary PCA to check structure*

3.2 Running M3C

In our example, we run the algorithm using the default settings (100x Monte Carlo iterations and 100x inner replications). We have found the results generally stable using these parameters. If the user is running a slow machine, has a massive dataset, or want results quickly, it is reasonable to lower the Monte Carlo iterations parameter to 5-50x. The inner replications needs to be at least 100x, changing this up to 250x will lead to increased result stability on some datasets.

Plots from the tool and an .csv file with the numerical outputs may be printed into the working directory (by adding printres = TRUE). We will set the seed in this example, incase the user wishes to repeat our results exactly (seed = 123). We will add an annotation file for streamlined downstream analyses (des = desx), and because we are comparing the classes with a categorical variable (tumour type) we shall set 'doanalysis' to TRUE and 'analysistype' to 'chi', to do a chi-squared test for significance. The 'variable' parameter is set to class (tumour histology in this data).

It is recommended to save the workspace after M3C if the user is working with a large dataset because the runtimes can be quite long. M3C uses by default PAM with Euclidean distance in the consensus clustering loop because we have found this runs fast with good results. We will set the 'removeplots' parameter to TRUE in this example to remove plots from the vignette, normally this is FALSE by default.

```r
# for vignette
res <- M3C(mydata, cores=1, seed = 123, des = desx, removeplots = TRUE,
           analysistype = 'chi', doanalysis = TRUE, variable = 'class')

# ordinary usage
# res <- M3C(mydata)
```

The scores and p values are contained within the res$scores object. We can see below the RCSI reaches a maxima at K = 4 (RSCI=0.33), the Monte Carlo p value supports this optimal K decision (p=0.033). This means the null hypothesis that K = 1 can be rejected for this dataset because we have achieved significance (alpha=0.05) versus a dataset with no clusters. For p values that extend beyond the lower limits imposed by the Monte Carlo simulation, M3C estimates parameters from the simulation to generate a beta distribution. The BETA_P in this case study is 0.033.

```r
res$scores
##    K  PAC_REAL   PAC_REF        RCSI MONTECARLO_P     BETA_P   P_SCORE
## 1  2 0.6408163 0.5847347 -0.09158464   0.58415842 0.61114800 0.2138536
## 2  3 0.4473469 0.5490776  0.20490525   0.15841584 0.17714343 0.7516749
## 3  4 0.3526531 0.4762531  0.30046461   0.02970297 0.03541014 1.4508724
## 4  5 0.3191837 0.4086122  0.24699993   0.05940594 0.04699842 1.3279167
## 5  6 0.3061224 0.3492082  0.13168302   0.18811881 0.17563554 0.7553876
## 6  7 0.2946939 0.3043755  0.03232506   0.41584158 0.41553858 0.3813886
## 7  8 0.2751020 0.2708571 -0.01555055   0.58415842 0.55673522 0.2543513
## 8  9 0.2563265 0.2439429 -0.04951814   0.62376238 0.64921812 0.1876094
## 9 10 0.2244898 0.2216327 -0.01280896   0.49504950 0.54812339 0.2611217
```

Also important is the relationship between the clinical variables and the discovered clusters. In this data we want to compare with a categorical variable so perform a chi-squared test. We are reassured to see below K=4 is highly significant (p=5.5x10-14). It is important to bear in mind the clinical or biological relationships as well as the structural nature of the data when deciding K.

```r
res$clinicalres
##    K          chi
## 1  2 8.977979e-07
## 2  3 2.961138e-12
## 3  4 5.479839e-14
## 4  5 7.124742e-13
## 5  6 1.294080e-11
## 6  7          NaN
## 7  8          NaN
## 8  9          NaN
## 9 10          NaN
```

Now we will take a look at some of the plots M3C generates.

This is a CDF plot of the consensus matrices for our test data. We are looking for the value of K with the flattest curve and this can be quantified using the PAC metric (Senbabaoglu et al., 2014). In the CDF and following PAC plot we can see the inherant bias of consensus clustering where as K increases so does the apparent stability of the results (or CDF plot flatness) for any given dataset, this we correct for by using a reference. This makes the method more sensitive to detection of structure in noisy data.

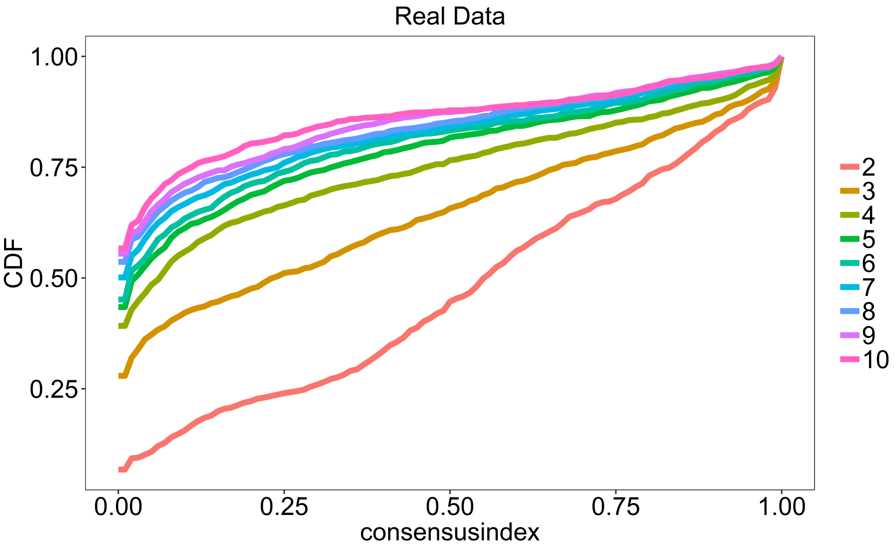

Figure 2: *Figure 2: CDF plot for real data*

This figure below shows the PAC score (Senbabaoglu et al., 2014), we can see an elbow at K = 4 which is suggestive this is the best K. However, the bias of the PAC score can be seen here as it naturally tends towards lower values as K increases (see above plot), making selecting K without taking this into account subject to this inherant bias. Selecting the minimal PAC score will only work when the clusters are well seperated.

Another option would be to use the original delta K metric here from the Monti method, however, identifying the elbow in this plot or 'value before the floor' is often very subjective, especially on noisy datasets like commonly occur in precision medicine. This metric has been shown to perform poorly on real data (Senbabaoglu et al., 2014) and like the PAC score has the theoretical issue of not accounting for CDF convergence as K increases.

In addition, these methods cannot reject the null hypothesis K=1, for that we have introduced p values into the consensus clustering methodology.

We then derive the Relative Cluster Stability Index (RCSI) which takes into account the reference PAC scores using the reference mean. This metric is better than the PAC score for deciding class number, where the maximum value corresponds to the optimal K. In this example the RCSI has an optima at K=4. We recommend the RCSI be used to select K, and the p values to reject the null in most cases.

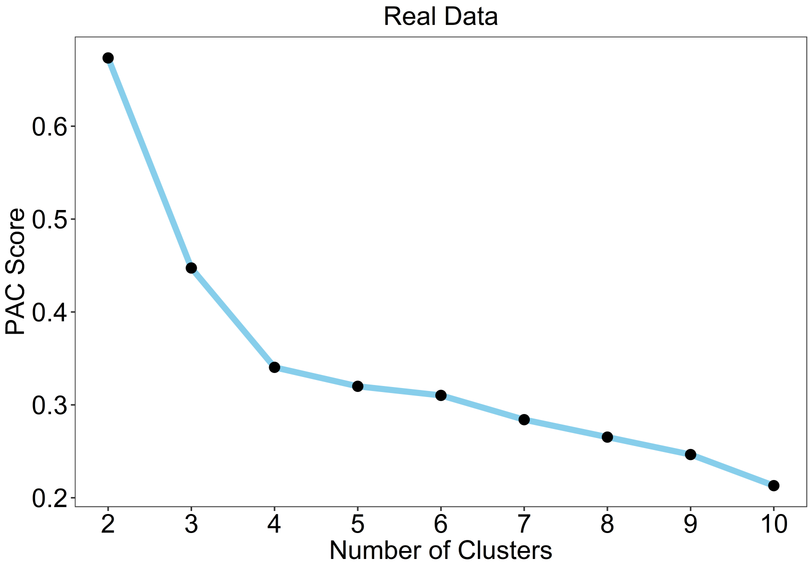

Figure 3: *Figure 3: PAC score for real data*

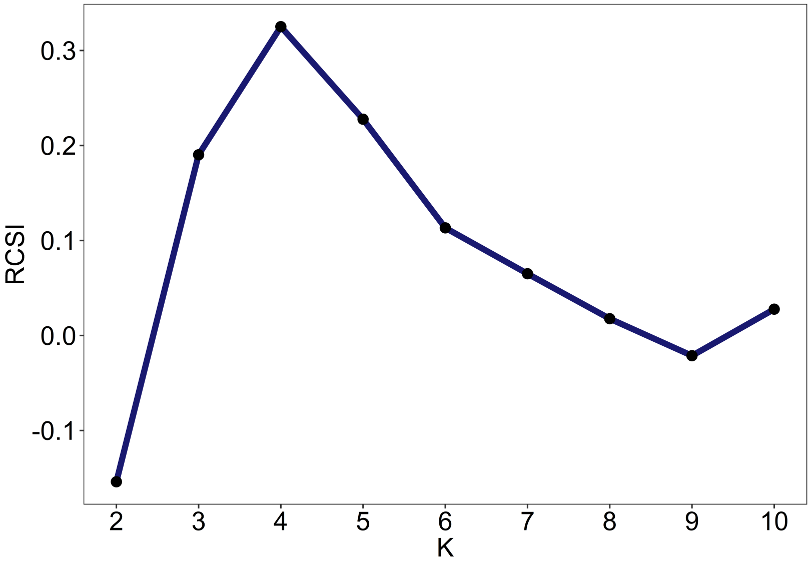

Figure 4: *Figure 4: RCSI*

Finally, we calculate a p value from the distribution, here we display the p values from the beta distribution. If none of the p values reach significance over a reasonable range of K (e.g. 10), then we accept the null hypothesis. In the GBM dataset, we can see K = 4 reaches signfiicance with an alpha of 0.05 (red dotted line), therefore we can reject the null hypothesis K=1 for the GBM dataset.

Now we are pretty convinced there are 4 clusters within this dataset which are not likely simply to have occurred by chance alone.

We can turn to examine the output objects that M3C generates. These allow heatmap generation for publications.

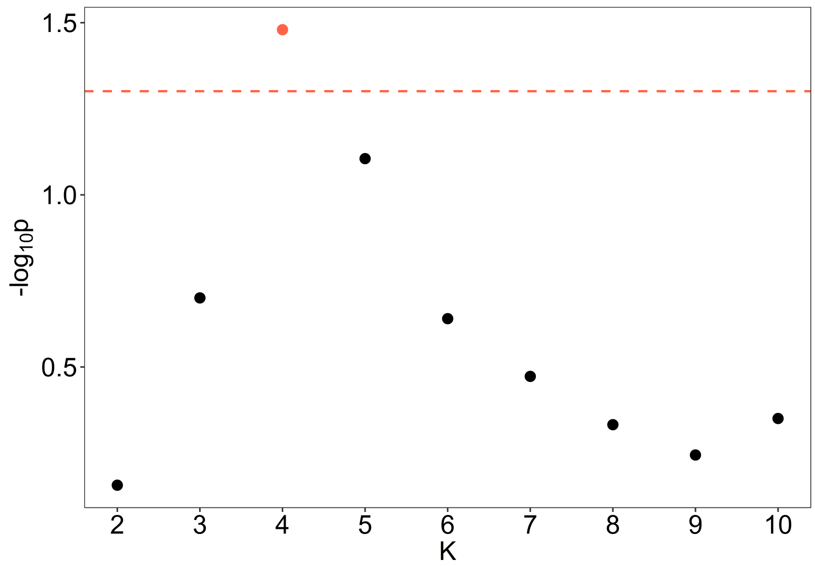

Figure 5: ***Figure 5: Empirical p values***

3.3 Understanding M3C outputs

The first 3 lines below extract the ordered (according in clustering results) expression data and the ordered annotation data from the results object after running M3C for a 4 cluster solution. If we, for example, wanted to extract the data for a 5 cluster solution from the M3C results list, we would simply replace 4 in the below lines to 5. We then take a brief glance at the annotation object M3C outputs, a consensus cluster column has been added by M3C.

```
# get the data out of the results list (by using $ - dollar sign) for K=4
data <- res$realdataresults[[4]]$ordered_data # this is the data
annon <- res$realdataresults[[4]]$ordered_annotation # this is the annotation
ccmatrix <- res$realdataresults[[4]]$consensus_matrix # this is the consensus matrix
head(annon)
##                     consensuscluster     class
## TCGA.02.0048.01A.01                1 Proneural
## TCGA.08.0517.01A.01                1 Proneural
## TCGA.08.0350.01A.01                1 Proneural
## TCGA.08.0524.01A.01                1 Proneural
## TCGA.06.0648.01A.01                1 Proneural
## TCGA.02.0432.01A.02                1 Proneural
```

Next, we scale the data here row wise according to z-score, prior to some light data compression for visualisation purposes in the heatmap. Remember to set Colv = NA for heatmap plotting because the data columns (samples) are already ordered.

Note, we are going to use the NMF aheatmap function to draw the heatmap because it is simple, but for more complex heatmap designs for publication, ComplexHeatmap is definately preferable (https://bioconductor.org/packages/release/bioc/html/ComplexHeatmap.html).

```
data <- t(scale(t(data))) # z-score normalise each row
data <- apply(data, 2, function(x) ifelse(x > 4, 4, x)) # compress data
data <- apply(data, 2, function(x) ifelse(x < -4, -4, x)) # compress data

ann_colors <- ggsci::pal_startrek("uniform")(4)
ann_colors2 <- ggsci::pal_futurama()(4) #
pal <- rev(colorRampPalette(RColorBrewer::brewer.pal(10, "RdBu"))(256))
NMF::aheatmap(data, annCol = annon, Colv = NA, distfun = 'pearson',
         color = gplots::bluered(256),
         annColors = list(class=ann_colors, consensuscluster=ann_colors2))
```

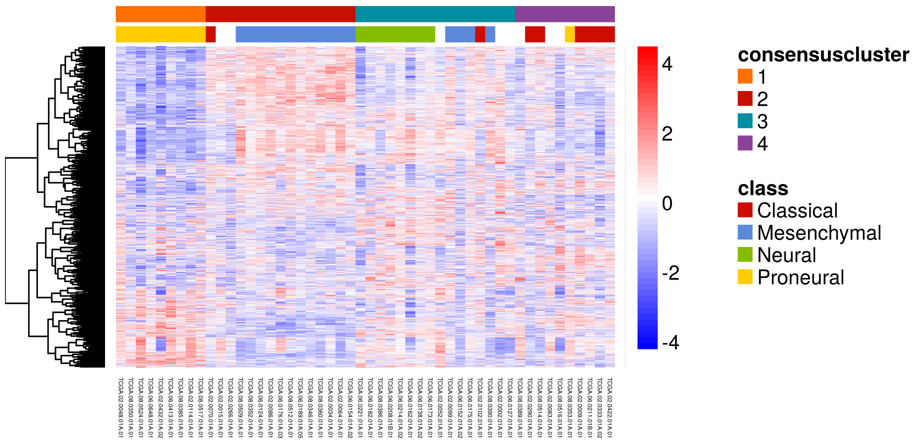

Figure 6: ***Figure 7: aheatmap of GBM consensus clusters with tumour classification***

Another plot we may want to do for publications is print the consensus matrix for our optimal clustering solution (in this case, K = 4). This should be quite crisp reflecting the significant stability of the results. We can see in this heatmap below of the consensus matrix the clusters do indeed look quite clear supporting our view that there is 4 clusters.

Viewing consensus matrices manually should not be used to decide K, better to use the RCSI or p values that M3C provides to quantify the mixing proportions in the consensus matrix versus the K=1 null model. We also caution against the use of cluster validity metrics on the consensus matrices without first testing their behaviour in details on simulated positive control datasets. Our data indicates that these metrics may have substantial bias in them as well due to the underlying method.

```
# set up colours
n <- 10
seq = rev(seq(0,255,by=255/(n)))
palRGB = cbind(seq,seq,255)
mypal <-rgb(palRGB,maxColorValue=255)
# plot consensus matrix heatmap, do not cluster rows and columns
NMF::aheatmap(ccmatrix,
              annCol = annon[,1,drop=FALSE],
              color = mypal, scale = 'none', cexRow = 0, cexCol = 0,
              Colv=NA,Rowv=NA,annColors = list(consensuscluster=ann_colors2))
```

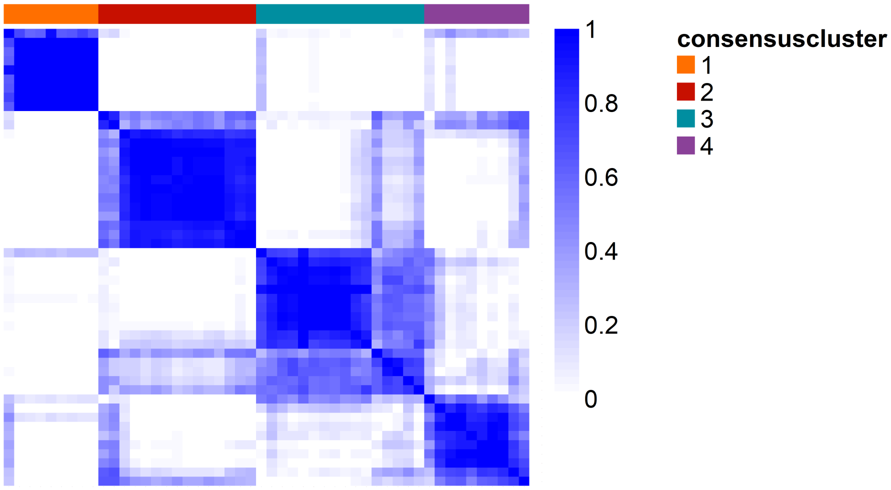

Figure 7: *Figure 8: aheatmap of GBM consensus matrix*

3.4 Final check of consensus cluster structure

A last analysis we recommend to do is to examine how the clusters are arranged in principal component and t-SNE latent variable space. To do this the pca and t-SNE functions can be run directly on the M3C output object. We may need to experiment with the perplexity parameter of t-SNE in the range of 5-20 to better suit the data. Both functions can also be used to print their results to the working directory. It is also possible to run UMAP on the data in a streamlined manner, see ?umap.

```
PCA2 <- pca(res, K=4)
TSNE <- tsne(res, K=4, perplex=15)
```

We can see in the PCA and t-SNE plots that clusters 3 and 4 are closer together. Doing a visual check like this allows the user to ensure there are no obvious artifacts that have occured.

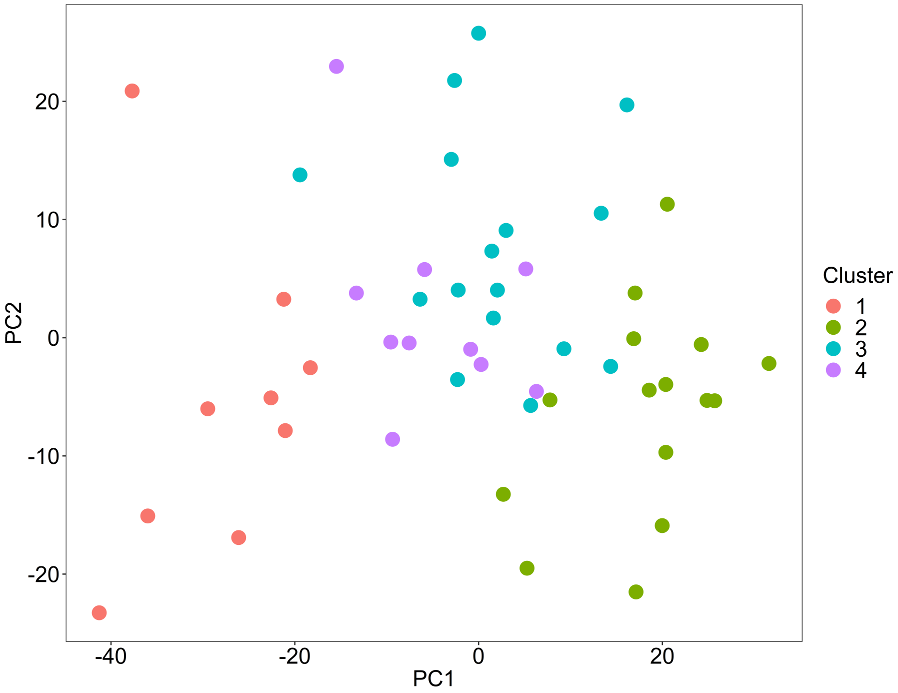

Figure 8: *Figure 9: PCA of a 4 cluster test dataset*

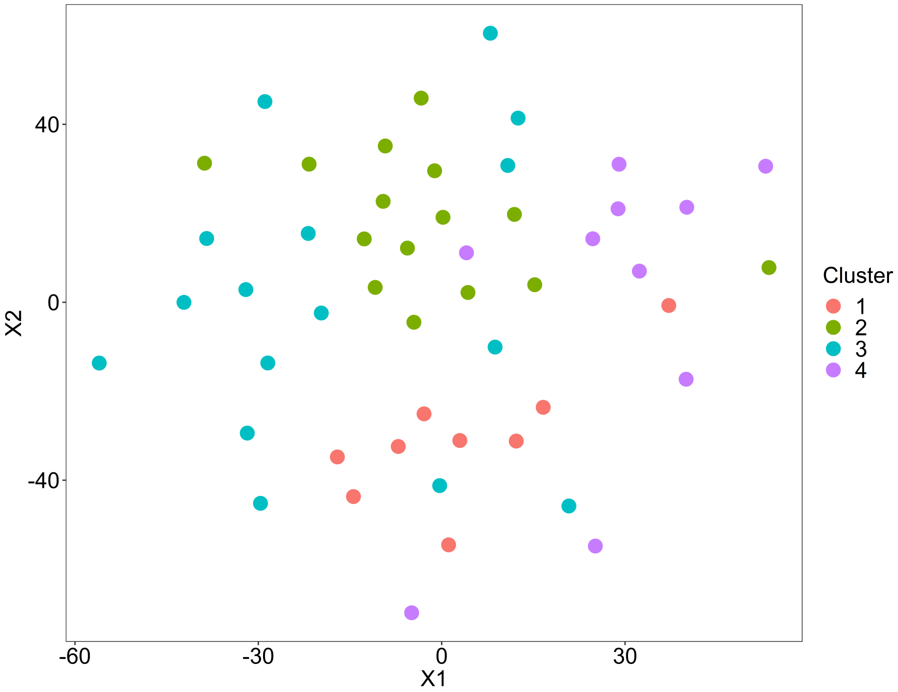

Figure 9: *Figure 10: t-SNE of a 4 cluster simulated dataset*

4 Example workflow II: Fast penalty method

4.1 Running M3C in fast mode

The second method minimises log(PAC) subject to lambda*K to find the optimal K. This metric is called the Penalised Cluster Stability Index (PCSI).

Lambda is a small constant that controls the regularisation strength, it has a preset value defined from experimental observations.

We recommend using this method if the user has a massive dataset or little time. It will correct for the overestimation of K observed with the PAC score.

```
res <- M3C(mydata, method = 2)
```

Here we can see the PCSI is minimised at K=4, this is the optimal K using this method.

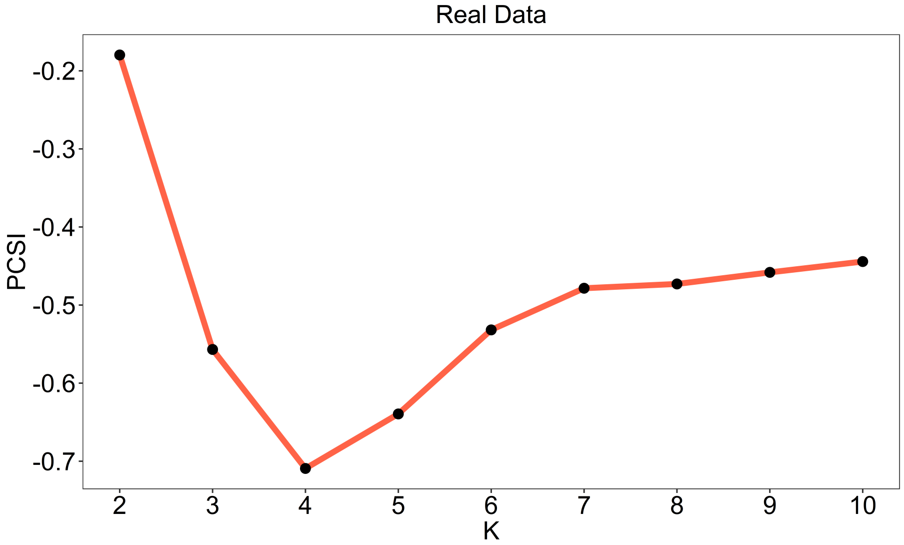

Figure 10: ***Figure 11: PCSI results***

For this method, it is a lot faster than running a Monte Carlo simulation so it is suggested to increase the ‘repsreal’ parameter to 250-1000 for increased result stability.

5 Additional functions

5.1 Generating simulated data

We have included a function for generating simulated data to test various clustering algorithms. This cluster simulator is simple to use. Using the code below, clustersim generates a dataset with 225 samples, 900 features, a radius cut-off for the initial square of 8, a cluster number of

4, a seperation of clusters of 0.75, and a degree of noise added to each co-ordinate of 0.025. After running, a PCA will print of the data so we can visualise the 4 clusters in principle component space.

Note, clustersim's method has recently been replaced with the CRAN package, clusterlab. We recommend using this for more sophisticated benchmarking. Clusterlab can generate Gaussian clusters with controlled variance and spacing, it can also create more complex higher order structures.

```
res <- clustersim(225, 900, 8, 4, 0.75, 0.025, print = FALSE, seed=123)
```

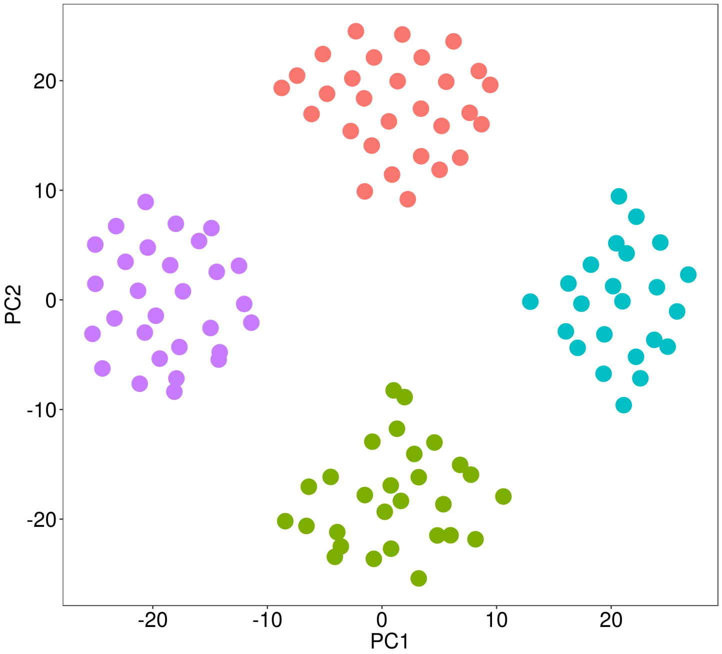

Figure 11: *Figure 12: PCA of a 4 cluster simulated dataset*

5.2 Filtering features by variance

A simple feature filter function has been included in M3C that uses one of variance, median absolute deviation (MAD), Pearson's coefficient of variation (A), or its second order derivative (A2) to filter the data. MAD will yield similar results to variance, however, because it uses the median instead of the mean it will be more robust to outliers. This will be the preferable metric in many cases.

The statistics for each feature (and the filtered data itself) are saved in a convienient data frame in the results list for plotting and further analysis. In the first case, the user could set percentile to 100 which collects statistics for all features. We then suggest to do some manual examination and plotting using the results of this function, for example, with the qplot function from ggplot2 of the mean-variance trend. If the data is appropiately transformed there should be little increase of variance with the mean, so we can use MAD or variance to filter the data. If the variance increases linearly with the mean, A or A2 may be more appropiate than variance or MAD.

Below, we set the percentile to filter by and input the data. The code below extracts the 10% most variable features using MAD as the filtering metric. In this case the example data has been already been filtered for variance then scaled additionally, so the results are not intended to be interpreteted as it is just an example. The statistics for the topN features will be printed to the console. Type ?featurefilter for more information.

```
  filtered_results <- featurefilter(mydata, percentile=10, method='MAD', topN=5)
## ***feature filter function***
## extracting the most variable: 10 percent
## features to start with: 1740
## performing calculations for median absolute deviation
## printing topN most variable features with statistics...
##        feature       mean      var       sd      MAD
## POSTN    POSTN  0.2967294 7.301561 2.702140 3.205915
## LTF        LTF  0.0117520 4.580893 2.140302 3.046446
## DCX        DCX -0.0115316 3.583629 1.893047 2.706227
## TMSL8    TMSL8  0.0846228 4.988073 2.233399 2.652757
## NNMT      NNMT  0.2344958 3.510178 1.873547 2.527811
## features remaining: 174
```

6 Closing comments

In this tutorial, we have seen that M3C provides a rigourous approach for selecting the number of clusters in the data. We have found in our analyses this approach results in increased performance relative to other methods and the removal of systematic bias inherant in the results. M3C's methodological and software developments are primarily built on the work of Tibshirani et al. (2001), Senbabaoglu et al. (2014), and Monti et al. (2003). Notably, it is best to use M3C after careful consideration of the format of the data, its normalisation and transformation, correction for any batch effects, and in conjunction with dimensionality reduction tools such as PCA and t-SNE to confirm the structure.

During the development of M3C, we also devised a flexible method and associated CRAN package for generating Gaussian clusters called, clusterlab (https://cran.r-project.org/web/packages/clusterlab/index.html). This tool should prove useful in testing the performance of class discovery algorithms.

For fast clustering of multi-omic data, we recommend Spectrum (https://cran.r-project.org/web/packages/Spectrum/index.html) (John et al., 2019). Spectrum is a self-tuning spectral clustering algorithm that can integrate heterogenous data sources and reduce noise. It uses a tensor product graph data integration and diffusion procedure, a density-aware kernel, and contains an optional data compression method to cluster 1000s of samples quickly. The method is also effective at single view cluster analysis.

7 References

John, Christopher Robert, et al. "M3C: A Monte Carlo reference-based consensus clustering algorithm." bioRxiv (2018): 377002.

John, Christopher R., et al. “Spectrum: Fast density-aware spectral clustering for single and multi-omic data.” BioRxiv (2019): 636639.

Monti, Stefano, et al. “Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data.” Machine learning 52.1 (2003): 91-118.

Kvålseth, Tarald O. “Coefficient of variation: the second-order alternative.” Journal of Applied Statistics 44.3 (2017): 402-415.

Senbabaoglu, Yasin, George Michailidis, and Jun Z. Li. “Critical limitations of consensus clustering in class discovery.” Scientific reports 4 (2014): 6207.

Tibshirani, Robert, Guenther Walther, and Trevor Hastie. “Estimating the number of clusters in a data set via the gap statistic.” Journal of the Royal Statistical Society: Series B (Statistical Methodology) 63.2 (2001): 411-423.