E-MAP analysis pipeline

An illustrated guide

[Visual inspection of plate photos](#h.u47jv1aep4ju)

[24h vs. 48h photos](#h.6lns1enf86oq)

[Colony Measurer](#h.jaf8jsnsy1c6)

[A note on very sick strains](#h.eo936tizyp09)

[Matlab E-MAP GUI and additional processing](#h.cw31jsttgguc)

[Installation](#h.d7ovrm9nrpj0)

[Preparation](#h.ldd2yt7xj6ko)

[Basic E-MAP GUI workflow](#h.enizlfkiso9f)

[Further processing of E-MAP data in Matlab](#h.a9obbftxk48j)

[Considerations for Filtering and Quality Control](#h.1nn05fdi3h30)

[Quality control for individual screens](#h.i6pozln84azo)

[Notes from Hannes (Krogan lab quality control)](#h.v6uya6wtt5gf)

[Processing data from multiple E-MAP screens](#h.evti5wq0j8zm)

[Quality Control Options for Combined E-MAPs of our own strains](#h.59bwhych880v)

[The “Ubermap”](#h.vwaow33u0ykv)

[Quality Control after joining with the Ubermap](#h.u9zx1noxdqs0)

[General Matlab notes](#h.z8b2s8pwn75n)

[Clustering](#h.ye09etbwqew)

[Visualization of clustering results](#h.1yqvdq8swrj0)

[Follow-up analysis](#h.wsrb6lpujprt)

[Module analysis with gia](#h.fj5yn8n6j0y)

[Curated Clusters](#h.niskf65e2m6n)

[Biological process annotations](#h.utmljel2lu9w)

# 

# Visual inspection of plate photos

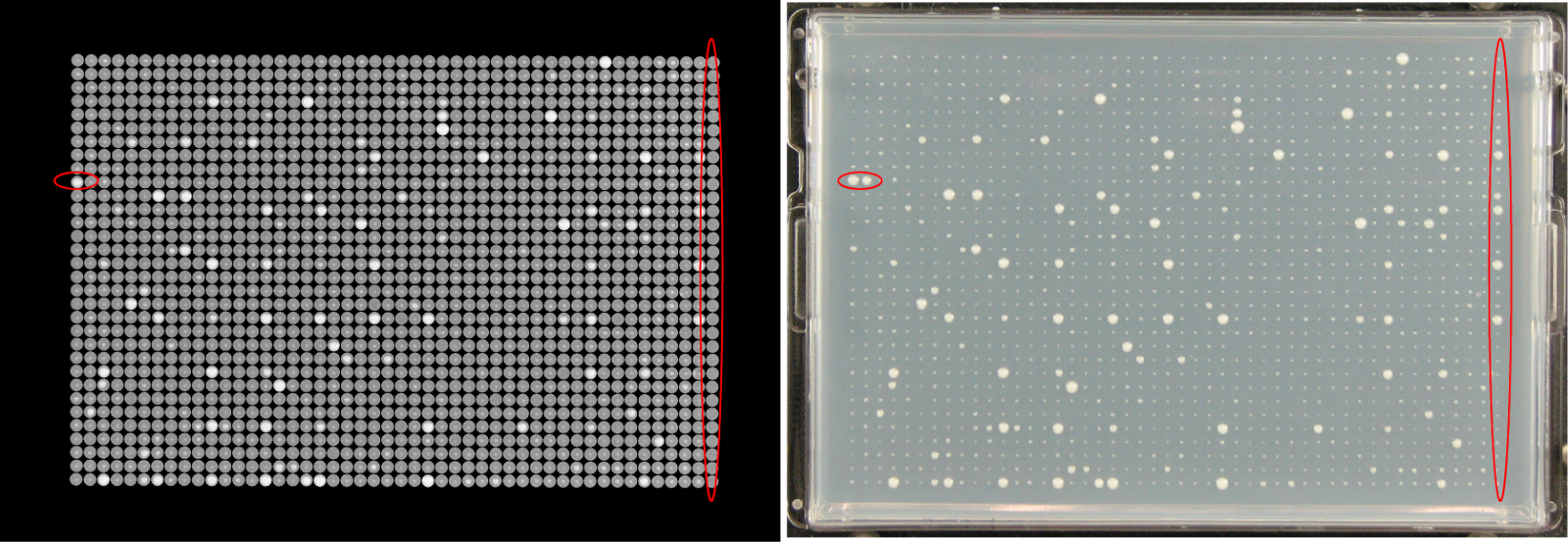
* do the pictures look consistent across triplicates, or are there many colonies that switch between being very small and very large (probably revertants, random mutations that happen to essentially restore WT fitness levels)?
* pay particular attention to corners and edges -- if the plates aren’t exactly level, pinning may not transfer cells evenly across the plate, leading to growth differences that are technical artifacts rather than biologically relevant
* if pictures are not in focus, this may create problems for the Colony Measurer
* if strains are very sick or not consistent across triplicates, they may need to be removed before running the analysis, as some steps compare against the global average, which will become noisy through such data
  + if strains are very sick, repeating the E-MAP screen on 384 plates may help
  + for this project, annotations for all screens can be found in [the deltaSH3 E-MAP evaluation spreadsheet](https://docs.google.com/spreadsheet/ccc?key=0AqpJl4EV2fcudFl5bXVmMFVQc25YS25hMGRveXBEeXc)
  + Technically, to remove strains just put the .dat files into a different folder. However, this must *not* be a subfolder of the one holding all the valid .dat files, as subfolders are processed along with the main selected folder, and so the sick strains wouldn’t be removed from the analysis.
    - It can be useful to put very sick stains into a dedicated folder, separated from those removed for other reasons (blurry picture, plate problems etc.) -- this allows adding the strains back once multiple screens have been performed and thus random revertants can be separated from true positive epistasis (see also discussion below)

## 24h vs. 48h photos

48h give sick strains a better chance to grow, but the risks of accumulating mutations that may affect fitness are also higher. As all plates are scaled to reach an average (?) of 500 pixels per colony, 48h data should be used for sicker strains to minimize scaling up noise. Ideally, the analysis should be done for the same time point for all plates, at least within a screen.

# Colony Measurer

* /kortemmelab/shared/amelie/E\_MAP\_analysis/colony\_measurer
* cd into the directory [of your copy] and run it there
  + results will be written to the folder in which the .JPG files are
* select correct grid size, default is 384, while most of our E-MAPs are on 1536 plates
* select File / batch mode, and choose the folder with all the .JPG files from 24h or 48h, will generate .dat files in which the colony size is quantified, and .dat\_grid\_ovr.jpg files visualizing the calculated grid overlay
  + these should be cross-checked to make sure the grid was placed correctly (especially with blurry photos and other potential problems)
  + pay particular attention to the edges -- if the leftmost or rightmost column is empty, this most likely indicates a failure in grid placement and should be cross-checked against the original image:



* + If the grid offset is not correct, this needs to be fixed, as the position on the plate is used to determine which library strain was used for this double mutant, and thus shifted grids mean that all strains on this plate will be wrongly annotated. To manually set the grid, use File->Manual mode, then
    - Auto-Crop Image
    - Draw a line between the centers of the first and last colony of the topmost row. Yellow circles will appear around each corner colony and four colonies in the middle of each edge; make sure these overlap with the actual colonies.
    - Generate Grid & Measure Image.
    - Finish.
      * This will write a new .dat file, but not overwrite the batch-mode generated grid image.

## A note on very sick strains

* individual large colonies on a plate with mostly tiny colonies are often revertants, i.e., they’re not surviving because of a positive epistatic effect but because some random rescue mutation occurred
* for interpretation of a single screen, it is basically impossible to tell random rescue mutants apart from true positive epistatic interactions
  + if the mutation happened after triplication, there will be lots of noise indicating such random reversions, which should always be taken as a warning sign
  + however, random rescue mutations can also occur before triplication -- this would give consistent patterns across triplicates, yet not necessarily a signal of a true genetic interaction
  + thus, unless the corresponding .JPG files were removed in the previous step, the .dat files from very sick or noisy or other removed strains should be removed before running the Matlab analysis for a single screen
* however, once a very sick strain has been screened multiple times, random revertant should not happen for the same double mutant, and thus those cases will be filtered by “strict averaging” -- in contrast, true positive genetic interactions should show up in each of those screens
  + this means that very sick strains can only be reasonably interpreted after several independent E-MAP screens
  + this may even argue for screening those strains more often, to ensure that random mutations are filtered out
* The colony measurer has a harder time with very sick strains, as many colonies are below its detection limit, so check the .dat\_grid\_ovr.jpg files very carefully.

# Matlab E-MAP GUI and additional processing

## Installation

* /kortemmelab/shared/amelie/E\_MAP\_analysis/EMAP toolbox
* using this requires a Matlab license as well as the Statistics Toolbox
* the E-MAP GUI must be “registered” via File -> Set path -> Add with subdirectories
  + it should be enough to do this once on each computer
  + in the new version of Matlab (R2012b) “Set Path” is a button on the “Home” tab in the main menu, “Environment” section
  + once the GUI is registered, Matlab will complain about a function name that clashes with a built-in function -- it seems we’ll have to live with this for now
  + conveniently, all newly added functions within those folders will be immediately available, there’s no need to repeat the “add path” step

## Preparation

* visual inspection of plates and colony size measuring, as described above
* create a tab-separated file that contains
  + the name of each query strain as used for the pictures (e.g., if the triplicate pictures are called DELAbp1\_1.JPG, DELAbp1\_2.JPG and DELAbp1\_3.JPG, the first column should state DELAbp1)
  + the ORF ID, for linkage filtering
  + a description of the mutation, for later reference -- note that this should not be DELETION as this will trigger a particular type of analysis that is not what we want/need (it would try to generate a square matrix, which in our case would be very sparse)
  + note that the first line of this file will always be ignored - this is intended for a header line (but if one directly starts with labels the first one will be missing)
* alternatively, create a spreadsheet with columns corresponding to the information described above (e.g. [https://docs.google.com/spreadsheet/ccc?key=0AqpJl4EV2fcudFl5bXVmMFVQc25YS25hMGRveXBEeXc&usp=drive\_web#gid=11](https://docs.google.com/spreadsheet/ccc?key=0AqpJl4EV2fcudFl5bXVmMFVQc25YS25hMGRveXBEeXc&usp=drive_web" \l "gid=11)), export that as a tab-separated file and use awk or other tools to bring it into the desired format:

**awk -F "\t" '{ print $5 "\t" $1 "\t" $4; }' full\_Google\_Doc.txt > name\_mapping.txt**

## Basic E-MAP GUI workflow

This section describes how to use the emapGUI from the Krogan lab to initially process the E-MAP data and store it in a format that can be handled by Matlab, as well as by other programs for clustering. Further processing can be applied in Matlab and other tools, and is described in later sections.

* type emapGUI into the Matlab command window
* Options->Preprocessing for position artifacts->2nd order surface for automatic preprocessing
* File -> Load Raw Data. A series of three windows will now pop up.
  + The first window is “Choose KAN strain coordinate map file”. You should load chromatin\_array\_1536.txt
  + “Choose Data Directory”. Choose the directory with all of the .dat files that were produced from the colony measurer. Note that .dat files in subdirectories will be processed as well.
  + “File with translations between file names labels and orf names”. Load the list of filenames, ORFs and mutation descriptions, prepared as described above.
* After the data is done loading 2 check marks will appear in the boxes for “Load Data” and “Normalize Data” (may take a few minutes)
  + any problems with data processing (e.g. mismatches between the mapping table and actual file names) will be reported in the main Matlab window
* skip the next checkbox “Filter Noisy Strains”. Hannes does not use this, instead manipulates the Matlab variables.
* Data Processing -> Score Data.
* Two check marks should appear “Score Data” and “Read Orf Coordinates”
* You might opt to skip the next checkbox “Filter by Linkage”. Hannes mentioned that he not use this, but it may work for this experiment setup (Amelie?)
  + how to visualize the linkage data? Is it necessary to store the matlab file and then read it in again?
    - also, when reading it in again, into the GUI or into Matlab, or both? (it makes a difference, surprisingly... the variables only show up in the workspace when loading them in via Matlab->Open, but not from the GUI)
  + how to apply linkage removal? Do we really need to write a file? (I thought there was an automated way to do this)
  + **this is a separate way of removing strains with “bad linkage” (determining this would require looking through them one-by-one) -- the automatic removal based on proximity in the genome is performed, see comments on scoremat and scorematL below**
* (optionally) Data Processing -> Average Data
  + is this the linkage-removed data or the standard data?
    - yes, and the format of the averageData variable is the same as of scoremat, so it can be used as input for the same functions (e.g. ExportForCluster3\_0)
  + the raw and averaged data are stored in different variables, so the raw values won’t be lost no matter whether averaging is performed or not
* Export->File for Cluster 3.0 (will be a txt file)
  + if one does not average, strains (sets of triplicates) with the same name in column 3 of the name mapping file will still show up separately, whereas they will be joined in averaging
  + File->Save data as .mat file
  + file is quite large
  + needed for further processing or joining as described below

## Further processing of E-MAP data in Matlab

The standard workflow as described above will perform the standard processing. Additional analysis and filtering functions are available, e.g. for correlation coefficients or combination of data from multiple screens. To run these, the .mat file exported from the emapGUI must be loaded into Matlab (not into the GUI). Then a number of functions that come with the GUI can be applied:

* linkageSDscatter
  + apply to scoremat or scorematL - the latter is filtered and should not have any major deviations, while scoremat will have a dip in the center
* join/merge scorematLs
* correlation coefficients
* Hannes has specific averaging functions that remove cases with too much variation between replicas; given the patterns I’ve seen this could be quite appropriate
  + thresholds may need to be tweaked for our dataset

Ask Hannes about

* filtering noisy strains beyond the visual inspection of plates
* other filtering steps he does

## Considerations for Filtering and Quality Control

### Quality control for individual screens

* Each plate is scaled up or down to (average?) 500 pixels per colony. If the colonies are too small, this will mainly scale up noise. Cross-check average sizes for each query strain (in the “raw” variable) to make sure it is high enough for scaling. Plates/strains where the colonies are too small must be removed (can be repeated on 384 plates for better stability, or perhaps try 48h pictures).
  + ReplicateControlCorrWithRowMut(rawN)
  + Hannes says he’d like to get values of 0.6 or higher. Some of our sicker strains have values around 0.3 and thus need to be considered with a grain of salt (or post-processed by hand)
    - the idea behind using them anyway is that, across independent screens, we should not get random revertants of the same combination repeatedly -- and if a combination does repeat, that may be an interesting suppressor
  + the Hof1 full gene deletion can get negative values, which is a clear sign that it should not be trusted
* The typical E-MAP introduction states that fitness (colony size) is compared to the expected fitness from the individual single deletions. We don’t have single deletion data though. The assumption is that true functional interactions are rare, and thus that the peak of the distribution of all colonies observed for [combinations involving] a particular construct represents the expected double mutant size.
  + for each query strain, 1500+ data points are available, but for library strains we only have one for each strain in the screen, which is why at least 30 query strains are required / strongly recommended
  + if all mutations in a particular screen target the same gene (e.g., many point mutations in the same protein) and all of those mutations have a similar effect on a library strain, this will be normalized out
    - probably not a big concern in our study, as the SH3-containing genes are quite diverse, but maybe we should be careful when analyzing endocytosis- or cytoskeleton-related double mutations
    - it helps to include strains from other people interested in other questions, as their strains are very likely to affect other functions, leading to “better normalization”
* Linkage information: In the newer version of the GUI there’s an option to get linkage visualized for each individual strain: Plot -> Linkage -> QLP (query linkage profile)
  + ensure that each strain has a dip (representing low scores) where the dashed red line (representing the location of the marked ORF) is
    - if this is not the case, the marker is elsewhere on the genome, which could be due to strain mislabeling or other problems
    - this usually implies that the strain needs to be regenerated, or, if possible, labeled correctly
* Hannes has a plugin that should be able to visualize standard deviations between plates. We could use this to identify double mutations with high standard deviations (which are not reliable and should not be used).
* There are 10 duplicate strains in the library (can only find 9 though, plus the WT is present 4 times) -- are these in any way used for quality control?
  + [YFL027C](http://www.yeastgenome.org/cgi-bin/locus.fpl?locus=YFL027C) (Gyp8)
  + [YGR214W](http://www.yeastgenome.org/cgi-bin/locus.fpl?locus=YGR214W) (Rps0A)
  + [YIL084C](http://www.yeastgenome.org/cgi-bin/locus.fpl?locus=YIL084C) (Sds3)
  + [YJL208C](http://www.yeastgenome.org/cgi-bin/locus.fpl?locus=YJL208C) (Nuc1)
  + [YJL210W](http://www.yeastgenome.org/cgi-bin/locus.fpl?locus=YJL210W) (Pex2)
  + [YJR040W](http://www.yeastgenome.org/cgi-bin/locus.fpl?locus=YJR040W) (Gef1)
  + [YLR110C](http://www.yeastgenome.org/cgi-bin/locus.fpl?locus=YLR110C) (Ccw12)
  + [YPL138C](http://www.yeastgenome.org/cgi-bin/locus.fpl?locus=YPL138C) (Spp1)
  + [YPL244C](http://www.yeastgenome.org/cgi-bin/locus.fpl?locus=YPL244C) (Hut1)
* scoremat contains all the data, whereas scorematL contains the linkage-filtered version
* When you average, if for example nat marked strains are the ones on the y axis (the query strains) and kan marked are the library strains. If a query strain says abp1 deletion and then one of the library strains is also probably going to be abp1 deletion, then this toolbox thinks that it’s a symmetric emap. Once you average it, the software then makes it symmetric. However, since it’s not meant to be symmetric in this case would have huge amounts of grey space. It would be 1500 by 1500 and mostly grey, so we just put “deletion1” in the name mapping file rather than “deletion” in so that the software thinks its a different strain. When the software was made early on, its was always used with symmetric emaps. Then look at ABP1 vs nat, and try to determine linkage. We would like ultimately for these to have low scores. This is a good way of seeing if the markers are in the right location. If one isn’t in the right location, you would have a pair that should be linked and they will still have high scores. Say you have two markers that are really close to one another and it still has a high score, it shouldn’t (should be dead) This is a way of getting an idea of which strains are good and bad based on the markers. Usually the nat strains are less likely to be wrong because they are made here w.hereas the library strains are bought
* TODO: include an example of desired linkageSDscatter(scoremat) and linkageSDscatter(scorematL) plots
  + other peaks probably indicate that we have some problem
    - how to find out where that problem lies?

### Notes from Hannes (Krogan lab quality control)

%Hannes Braberg, 081112

Outline of how Sean processes and quality controls EMAP data:

-Process as usual with EMAP Toolbox.

-Run ReplicateControlCorr.m to compute the correlation coefficients between the colony sizes on all plates with the median colony size (controlsize) in each corresponding position. The important thing here is that all 3 replicates should give similar correlation coefficients. Also, if they are lower than say 0.2, something may be off. If corr coeff is e.g. 0.3 but all three similar, this is probably just a strain with a lot of interactions.

-Compute correlation between median array colony sizes and growth rate data for the corresponding mutations (Breslow, Giaever and Warr...). You have the Breslow, Giaever and Warr sets in fullset 05-28-08.mat that you got from Sean. In order to compute the correlations, you first have to modify this variable. This is done using the following commands (see diary file in case i made a mistake):

%%

EDU>> fullset.colMut = cell(5924,1);

EDU>> fullset.colMut(1:5924,1) = cellstr('DELETION');

EDU>> fullset.data(1:5924,1) = fullset.bres;

EDU>> fullset.data(1:5924,2) = fullset.giaev;

EDU>> fullset.data(1:5924,3) = fullset.warr;

EDU>> fullset.data = fullset.data';

EDU>> fullset.rowlabels = cell(3,1);

EDU>> fullset.rowlabels(1,1) = cellstr('bres');

EDU>> fullset.rowlabels(2,1) = cellstr('giaev');

EDU>> fullset.rowlabels(3,1) = cellstr('warr');

EDU>> fullset.rowMut(1:3,1) = cellstr('DELETION');

EDU>> fullset.geneToOrf = readGeneNames(fullset);

%%

I have saved this modified version of fullset as fullset\_modified.mat in this directory, so you should not actually have to do this again.

Now, use computeRowScatterUsingNames.m to compute the correlation coeffs between median sizes and growth rate data.

%%

EDU>> computeRowScatterUsingNames(fullset,1,rawN,1:55); %This command will give corr coeffs with the Breslow set.

%%

A common and reasonable value for the correlation of a good quality dataset with the deletion growth rates is e.g 0.4. In the diary file and growth\_rate\_corr.fig we get 0.56, which is very high, and hence very good. Ignore the cosine corr. Only care about the Pearson and Spearman. So, first check that the total correlation is okay, then check out the outliers. In particular, we are interested in strains that should be sick but are not in our data, i.e. upper left part of the graph (x axis is growth rate, y axis median # of pixels in the EMAP). A median size of 400 pixels and below counts as sick. 500 pixels and above for a strain with growth rate lower than ~.95 is suspicious. Click on these points to get gene names, and scrutinize those strains. They are likely wrong, but not necessarily. Watch these carefully; check linkage, check if they correlate with old data etc (guess biological replicates should be checked as well). if much seems wrong for those strains, trash them. Note that if you input a vector to computeRowScatterUsingNames(), it automatically uses the median of the elements, hence "...,rawN,1:55);".

-Also run computeRowScatterUsingNames.m for the new set compared with an old EMAP data set instead of growth rates. Obviously, this can only be done for sets that have partial overlap in the arrays. Correlations here should be very high. If some strain deviates, that particular strain in the array could have been contaminated in one of the screens. If two screens were processed separately and have high correlation but not on the exact x=y line, then they should not be scored together. In this case, they should be scored separately and merged after scoring. If perfect correlation on x=y diagonal, then you can go back and score them together. Never score screens that were made on different media (and/or different markers) together. E.g. do not score double and triple mutants together.

-Next thing is looking at linkage. Both Sean and Assen have functions for this. For Assen's functions, look in the assen\_fcns/ directory. I have a diary file there that shows some tricks. Sean's functions include plotScoresByChromCoordinate.m to see if some query strains have good/bad linkage in different parts of the array, and listLinkageScores.m which prints lists of all linkage scores for a strain, starting with closest neighbor going further and further away.

-Periodically, run scatterBiologicalReplicates on your fully pruned matrix as well.

%%

EDU>> scatterBiologicalReplicates(scorematL);

%%

Example, see BiologicalReplicates.fig. For the ESP dataset, R ~0.5. Then prune out bad strains based on this plot and make new variable and rerun scatterBiologicalReplicates.m on the new var. R should now be significantly better.

-Periodically, go through each row and column in your data and compare with profiles in an old dataset to find what it correlates best with. Should correlate with the same. Use a master averaged dataset for the comparison. That would be the ubermap in the cerevisiae case. Also, have a master unaveraged dataset including everything we've done, to which you add new data when you get it. I think that to the unaveraged dataset you add everything you have, whereas to the averaged, you only add the subsets that you can average (ie that exist in both marker-swapped variants).

exportMostCorrelatedvsReferenceSet generates an excel file for marker swap replicates. works as scatterBiologicalReplicates but without scatters. It will show what it correlates best to, and this should be to its replicate. Guess you can maybe use this fcn to get correlations with the master sets mentioned above as well?

-Finally, of course, see if the clustering makes sense.

Sean does not use the noisy strains function much these days. Seems that the other quality control steps will find such strains anyway.

## 

## 

## Processing data from multiple E-MAP screens

* Usually, averaging should only be done once all the screens have been done.
  + However, averaging also removes the duplicated strains (see above), so it can be useful for interpretation of single screens.
* joinScoremat and mergeScoremat functions can be used to combine data from multiple screens, details to be figured out..
  + in order to join multiple scoremat(L)s, one needs to load them in and make copies (as loading the next file will overwrite the previous scoremat), then join

### Quality Control Options for Combined E-MAPs of our own strains

## The “Ubermap”

(todo)

### Quality Control after joining with the Ubermap

## General Matlab notes

* filenames (e.g. for manually exporting data via exportForCluster3\_0) and strings must be enclosed in single quotes
* Matlab doesn’t do lazy (short-circuit) evaluation of multiple conditions in an if statement, which means that statements must be nested unless all components of the statement are always well defined
  + e.g.,   
    if (size(l\_details) == 3 && strcmpi(l\_details{3}, 'damp'))  
    will fail with a not-very-helpful error message if there are less than 3 elements in l\_details, or if strcmpi() isn’t well-defined for that element
  + this means nested IF statements are required
* one needs to use curly braces to access a string in a matrix/cell array, otherwise it isn’t recognized as a string...
* size(x) can return multidimensional values (e.g. for matrices), but does not warn about this -- beware of using size() in IF or FOR statements

## 

# Clustering

<http://bonsai.hgc.jp/~mdehoon/software/cluster/software.htm>

* Open Cluster 3.0
* File->open the file generated for cluster 3.0
* Hierachical clustering
  + cluster both “genes” (query strains) and “arrays” (library strains)
    - i.e., check “cluster” in both fields
  + Hannes uses “Correlation (centered)” as the similarity metric for both, and average linkage
* A .cdt file will be output in the same directory as the emap file

# Visualization of clustering results

<http://jtreeview.sourceforge.net/>

* Open the cdt file in JavaTreeView
* consider adjusting the coloring threshold so that only strong positives/negatives show up (depending on signal strength)
  + 5 works well for our current E-MAPs
  + when looking at correlation coefficients, all values will be between 1 and -1, so the threshold will need to be adjusted

# Follow-up analysis

[mapping of the clustered results to actual biological functions, and selection of candidates for further analysis -- todo]

## Module analysis with gia

* Input
  + Dataset: processed E-MAPs, I use the exportForGia function which removes spaces from names (gia chokes on those, Cluster doesn’t)
  + Annotations:
    - left: labels to group the rows in the E-MAP output file into modules
      * for now, we’re pretending that each case is its own module -- however, this will give lower statistical significance
    - right: curated clusters or biological annotations (GO-term-like), see details below
      * must be formatted so that the first column contains the ORF ID, and the second column contains the description
* Good to know:
  + Annotations that don’t match any of the input data will be ignored (they won’t even be shown in the gui) -- if no annotations are listed after loading a file, it’s probably a problem with the formatting
  + It’s not necessary to set thresholds on p-values when running gia, as they will all be recorded in the output file and can then be used when filtering the resulting graph in Cytoscape
  + Minimum measured interactions: default is 3, which means that binary complexes will be ignored
    - not sure if this is the most appropriate/helpful for SH3-related data though...

### Curated Clusters

### Biological process annotations

### Visualization of gia modules in Cytoscape

* import the network:
  + Cytoscape 3.0: File -> Import -> Network -> File...
  + “Show Text File Import Options”
  + make sure that Tab is selected, but Space is not
  + check “Transfer first line as column names”
* Source <-> column 1
* Target <-> column 2
* check at least the P\_bonf < 0.05, FDF < 0.1, uncorrected P < 0.001 columns, as these will be used for filtering