Biclustering Tutorial: Gene-expression data

Trying to find structure within a M-x-N Gene-expression data matrix

Aaditya V Rangan, NYU

In this tutorial we'll slowly walk through a biclustering analysis of a particular gene-expression data set. The biclustering method we will use is based on the simple 'loop-counting' algorithm proposed in (Rangan, *A simple filter for detecting low-rank submatrices.* Journal of Computational Physics. Volume 231 Issue 7, April, 2012 Pages 2682-2690).

The data-set under consideration will be a subset of the GSE17536 data taken from the gene-expression-omnibus (uploaded in 2009). The full data-set can be accessed at http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE17536. The subset of data that we use comprises 17942 gene-expression measurements (referred to later as simply 'genes' for simplicity) collected across 175 patients, each diagnosed with colorectal cancer. We chose to use this data-set for the tutorial because of its accessibility and moderate number of patients. In addition, the data-set is annotated with several covariates; allowing us to demonstrate the covariate-corrected version of our algorithm (although this covariate-corrected algorithm is not required to find high-quality biclusters).

To actually run this tutorial, download the file '`tutorial_archive_lvl0.tar.gz`' or '`tutorial_archive_lvl1.tar.gz`' from and then, from the unzipped/untarred directory, run the following command in Matlab:

```
>> tutorial_create_GSE17536; % reads data from GSE17536.GPL570.pcl and GSE17536.gsms.txt ;
```

Once the data is ready for analysis choose path names, normalization convention and search paradigm:

```
>> path_base=sprintf('%s/dir_GSE17536',pwd); % choose paths ;
>> gen_fname='GSE17536_n2x'; % choose normalization+paradigm, such as _n0_, _n0x, _n1_, _n1x, etc.;
```

The following command takes roughly 10 minutes on a standard laptop to extract+delineate `n_to_find==2` biclusters from the original data :

```
>> n_to_find=2; tutorial_w1(path_base,gen_fname,1,0,n_to_find,'frwd'); % bicluster the data ;
```

The following command takes about 30-seconds to bicluster each of 2048 label-shuffled permutations; you may perform fewer if you wish:

```
>> for np=1:2048; tutorial_w1(path_base,gen_fname,0,np,1,'frwd'); end; % bicluster shuffled data ;
```

Now you can use these label-shuffled trials to assign a p-value to the `n_to_find` biclusters from the original data:

```
>> tutorial_collate(path_base,gen_fname,'frwd'); % use shuffled data to determine p-values ;
```

Finally, if you have also installed seek.GeneEnrichTest (see seek.princeton.edu), then you may run:

```
>> tutorial_genri(path_base,gen_fname,'frwd'); % determine gene-enrichment of each bicluster ;
>> tutorial_summarize(path_base,gen_fname,'frwd'); % summarize enrichment data ;
>> tutorial_plot(path_base,gen_fname,'frwd'); % plot summary figures ;
```

Biclustering Tutorial: Gene-expression data

Trying to find structure within a M-x-N Gene-expression data matrix

Aaditya V Rangan, NYU

In this tutorial we'll slowly walk through a biclustering analysis of a particular gene-expression data set. The biclustering method we will use is based on the simple 'loop-counting' algorithm proposed in (Rangan, *A simple filter for detecting low-rank submatrices.* Journal of Computational Physics. Volume 231 Issue 7, April, 2012 Pages 2682-2690).

The data-set under consideration will be a subset of the GSE17536 data taken from the gene-expression-omnibus (uploaded in 2009). The full data-set can be accessed at http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE17536. The subset of data that we use comprises 17942 gene-expression measurements (referred to later as simply 'genes' for simplicity) collected across 175 patients, each diagnosed with colorectal cancer. We chose to use this data-set for the tutorial because of its accessibility and moderate number of patients. In addition, the data-set is annotated with several covariates; allowing us to demonstrate the covariate-corrected version of our algorithm (although this covariate-corrected algorithm is not required to find high-quality biclusters).

We remark that the output-files generated by the previous commands are all contained in the '`tutorial_archive_lvl2.tar.gz`' file, so if you actually downloaded the entire lvl2 archive, you don't need to run the biclustering algorithm in order to look at the results (in other words, if you downloaded everything you should be able to run `tutorial_collate` or `tutorial_plot` immediately).

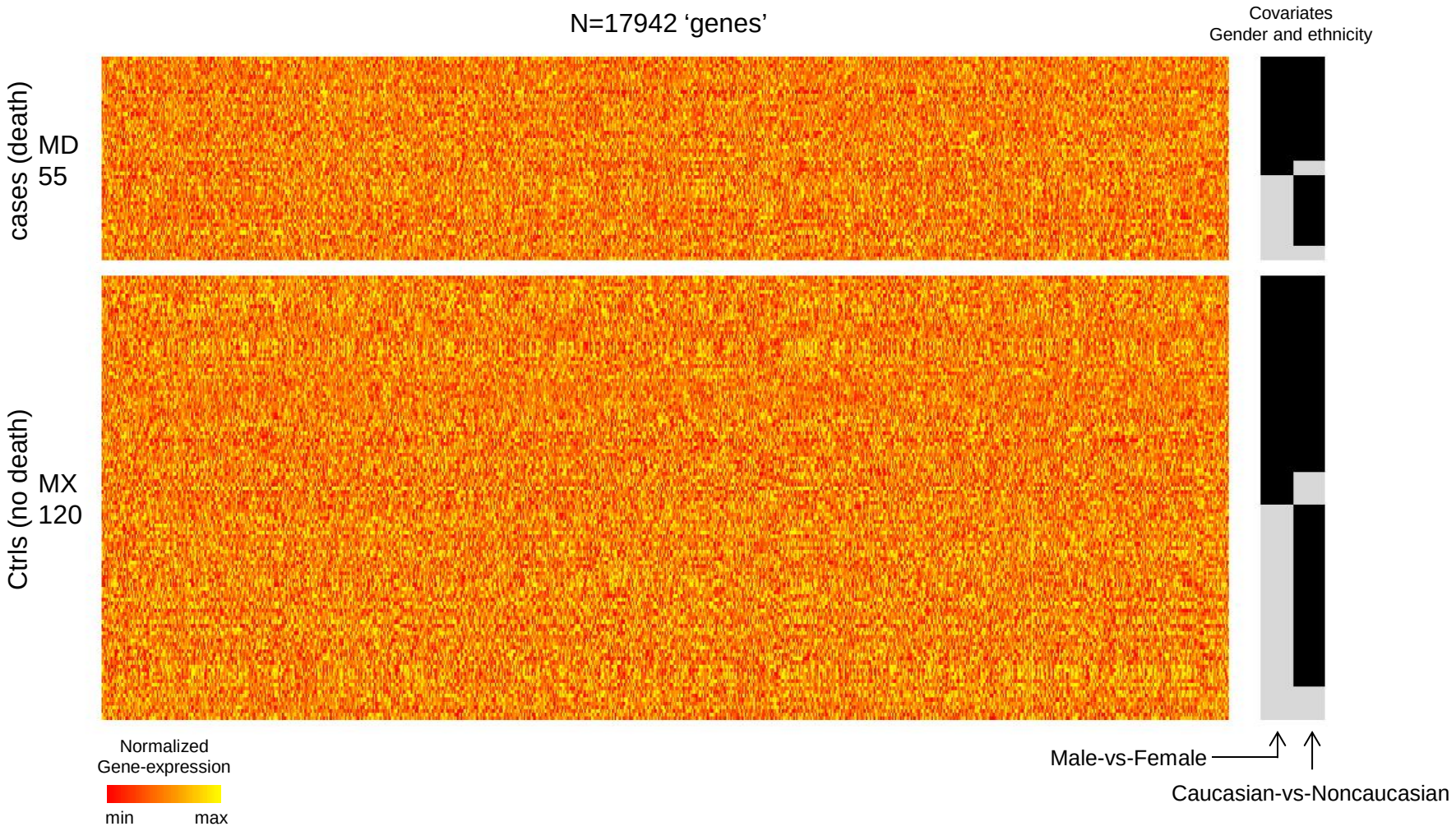
As a preview of our results, we illustrate the data-set itself on the next slide, then show three of the biclusters we found in the slides after that. We will use the following notation:

- MD = the number of patients deemed to be 'cases'
- MX = the number of patients used as 'controls'.
- N = the number of genes measured for each case and control.

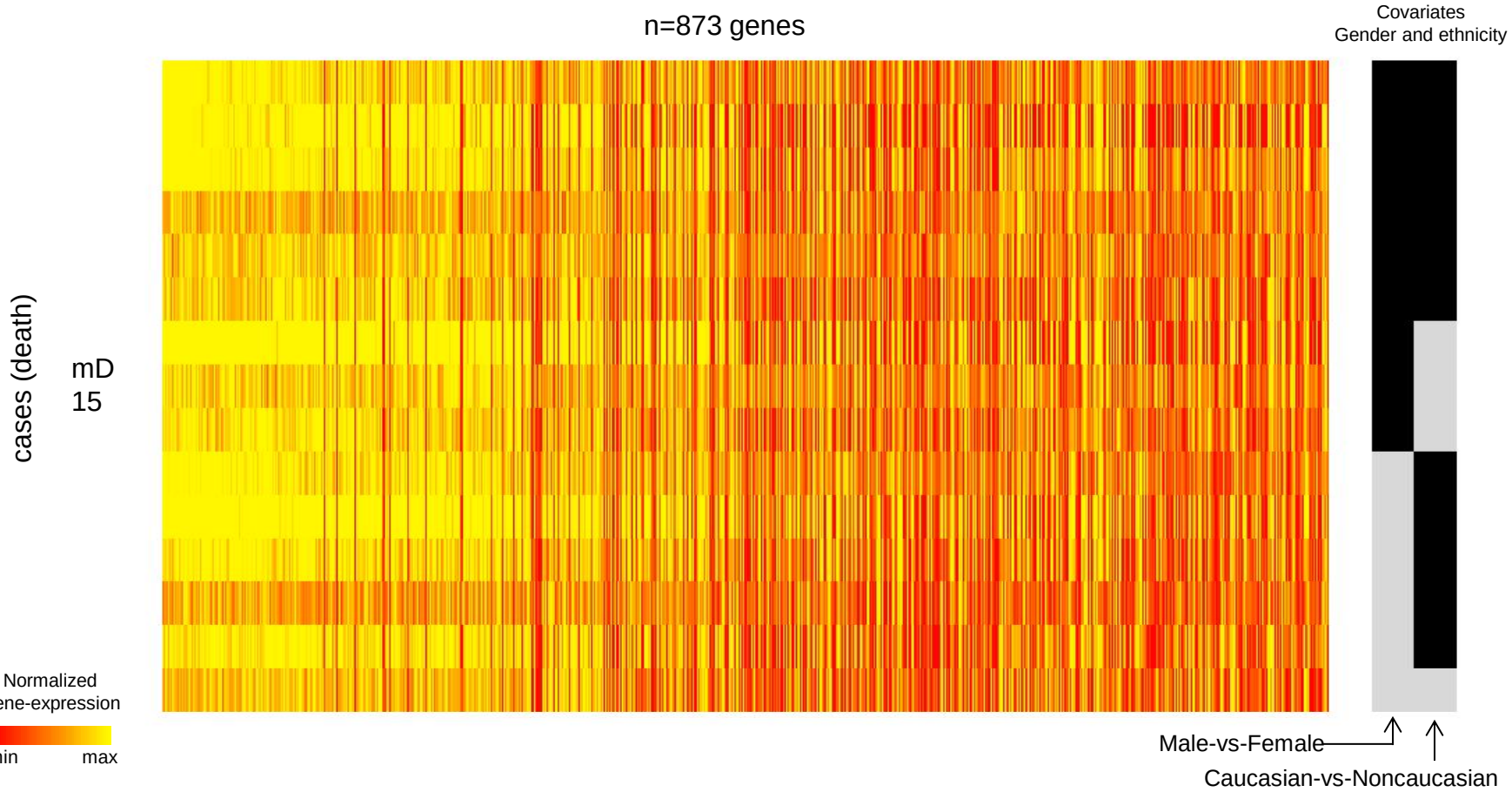
Sometimes we will refer to a bicluster of patients as a submatrix of the original gene-expression array:

- mD = the number of patients within the bicluster
- n = the number of genes involved in the bicluster

Here we illustrate the GSE17536 data set, comprising N=17942 gene-expression measurements collected across M=175 patients.
Each row corresponds to a patient, and each column to a 'gene' (i.e., gene-expression measurement): the color of each pixel codes for the intensity of a particular measurement of a particular patient (see colorbar to the bottom).
Some of these patients (MD=55) passed away, with cancer determined to be a significant contributor to their cause of death.
The remaining patients (MX=120) either have not yet died, or have died of other causes.
We will use these categories to divide our patient population into cases and controls respectively.
We will then search this data-set for 'biclusters' – namely subsets of genes that are structured in some way across a significantly large subset of the case-patients, while not being similarly structured across the control population.
Aside from their gene-expression data, each patient is also endowed with a variety of other characteristics, including their gender and ethnicity. These covariates are shown to the far right, (i.e., gray vs black).

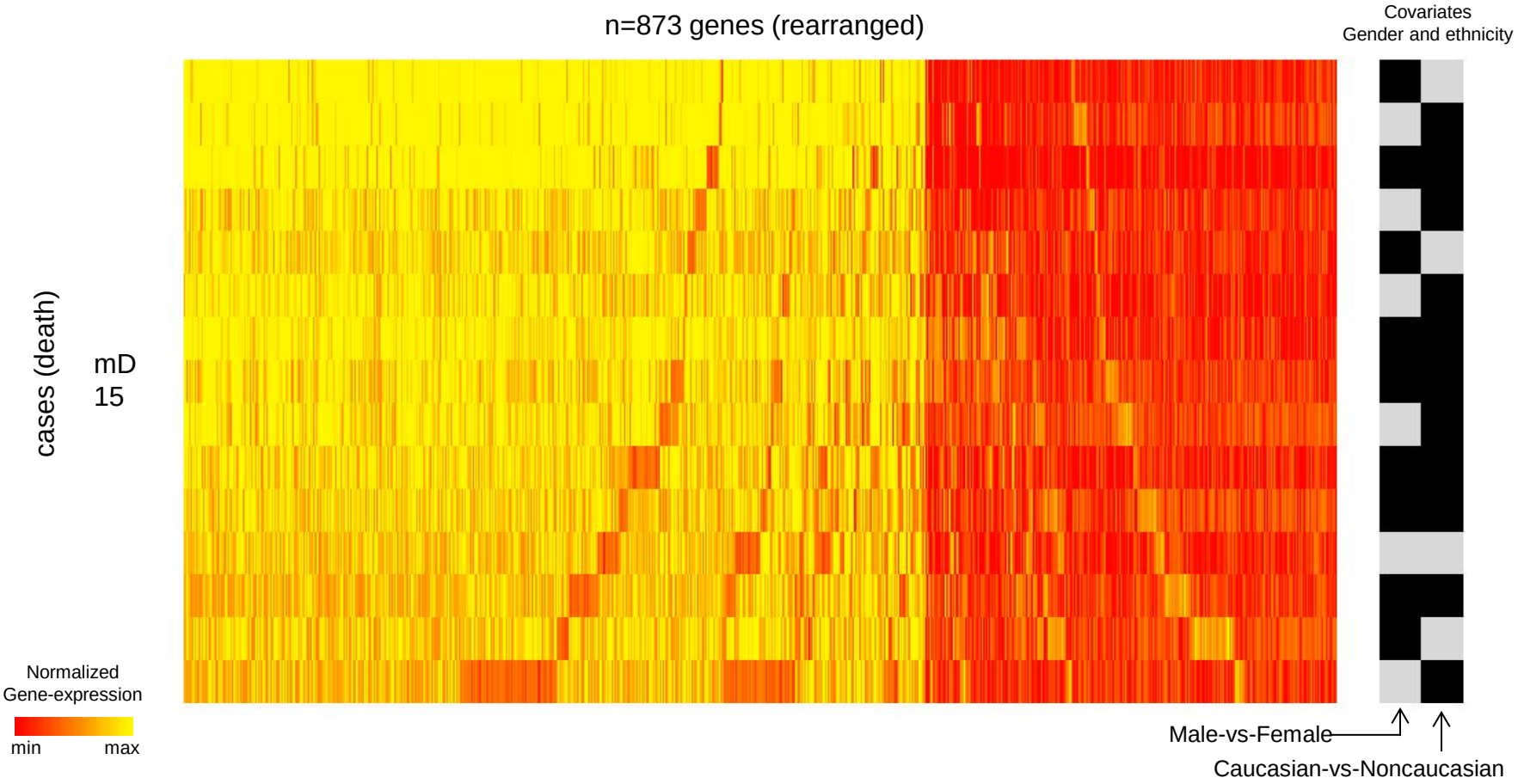


Here we illustrate our first example: a large bicluster – comprising mD=15 of the MD=55 cases and n=873 of the N=17942 genes – that was embedded in the previous matrix. This bicluster consists of genes that, taken individually, are either (i) significantly over-expressed or (ii) significantly under-expressed – across these 15 cases relative to the control population. Note that this bicluster is also balanced across covariate-categories; i.e., the patients comprising the bicluster are not all male or all caucasian.

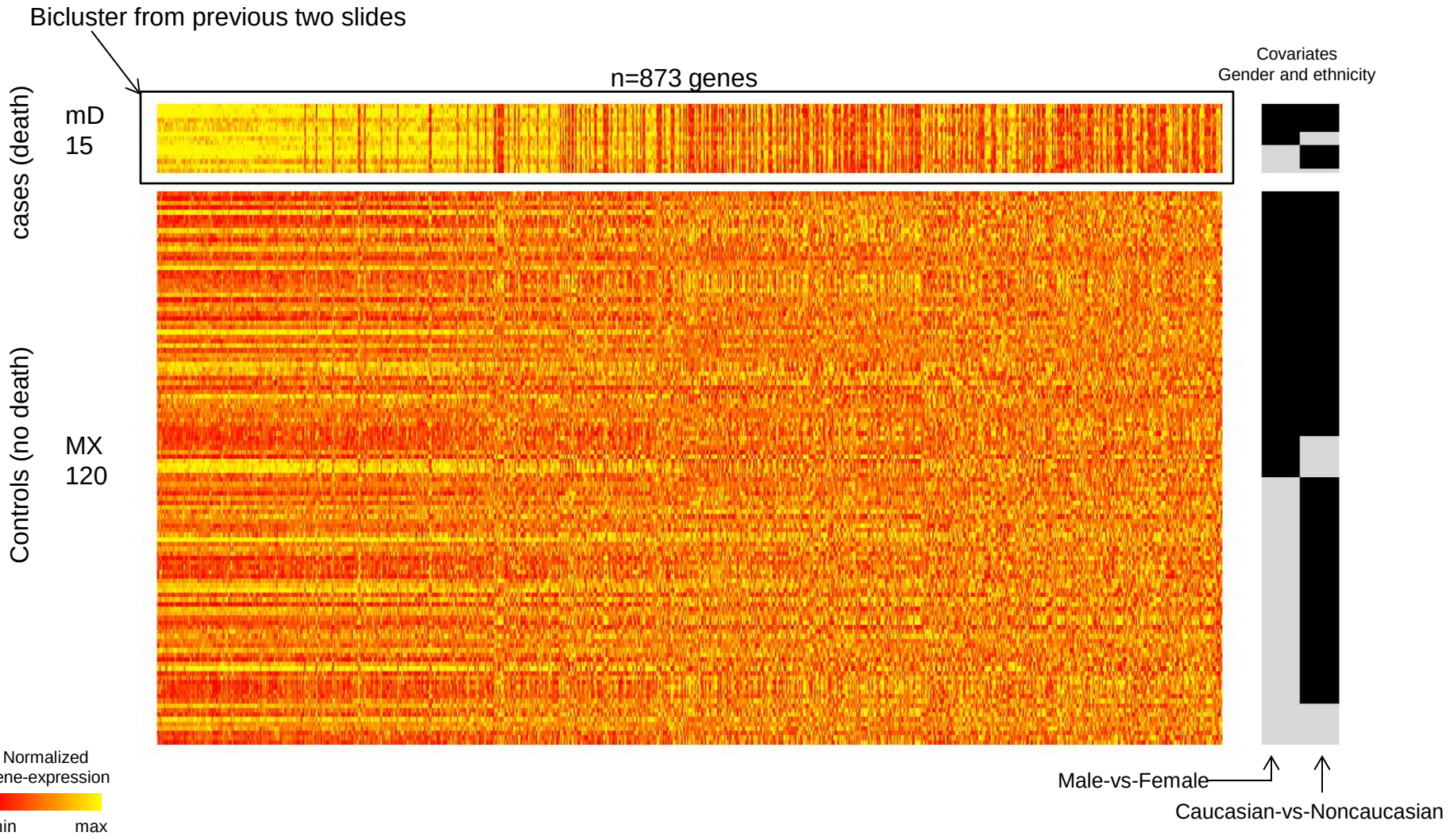


Here we illustrate our first example: a large bicluster – comprising mD=15 of the MD=55 cases and n=873 of the N=17942 genes – that was embedded in the previous matrix. This bicluster consists of genes that, taken individually, are either (i) significantly over-expressed or (ii) significantly under-expressed – across these 15 cases relative to the control population. Note that this bicluster is also balanced across covariate-categories; i.e., the patients comprising the bicluster are not all male or all caucasian.

The depiction in the previous slide was constructed using the output of our algorithm – which orders the rows and columns in a manner that is explained in Appendix-1, but which does not make the differential-expression pattern particularly clear. Thus, to show this differential-expression more clearly, we present the bicluster again, except this time with the rows and columns rearranged based on the magnitude of the gene-expression-data. As can be seen, each gene is either mostly high or mostly low across these 15 cases.



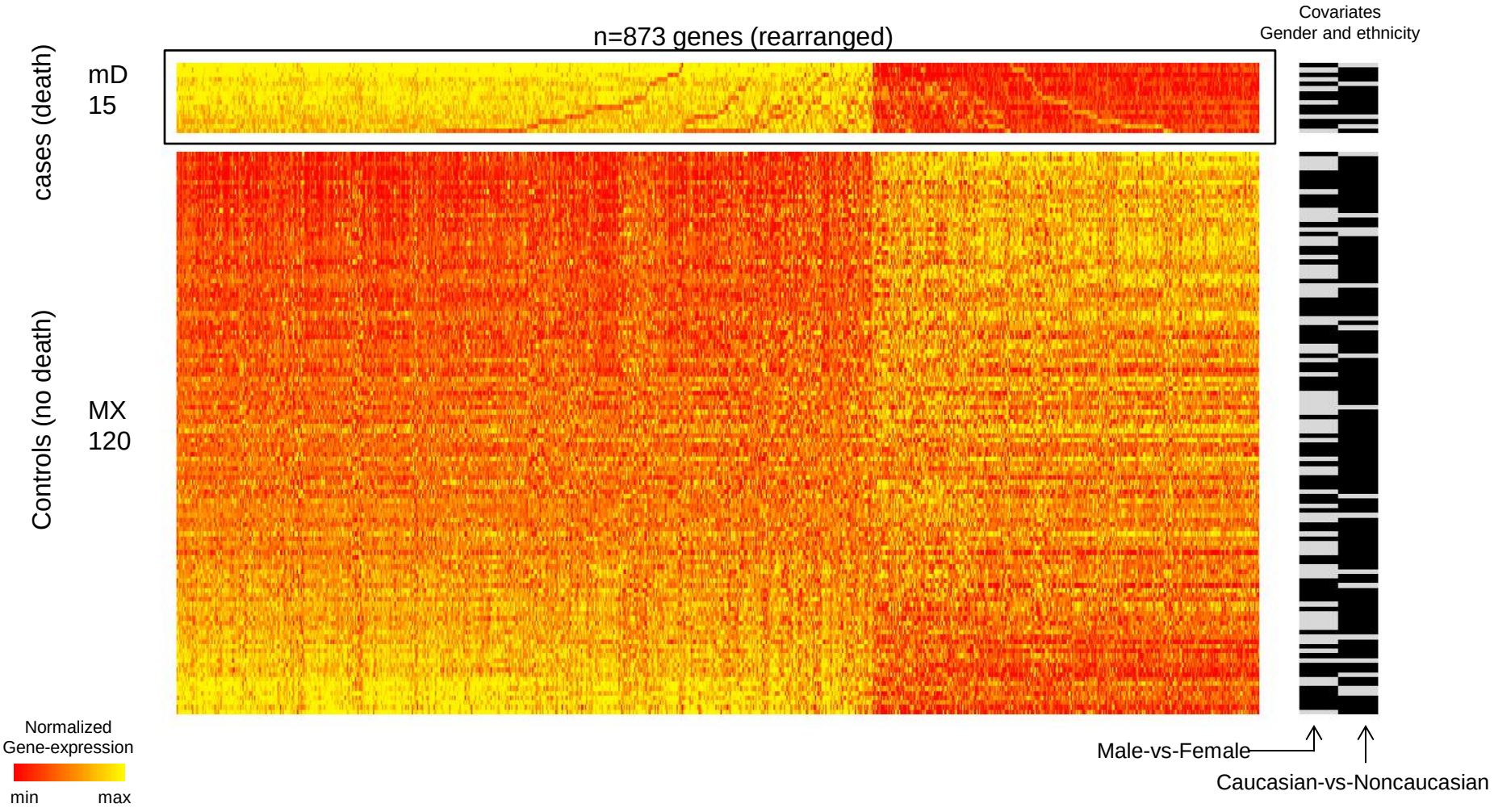
To illustrate that this differential-expression pattern is indeed observed in relation to the controls, we plot that same bicluster again (top) against the rest of the controls (bottom).



When we rearrange the genes to emphasize the differential-expression, we see that the bicluster is indeed evident; even though some of the controls do show the same pattern of differential-expression they are a minority and exhibit that pattern to a lesser degree.

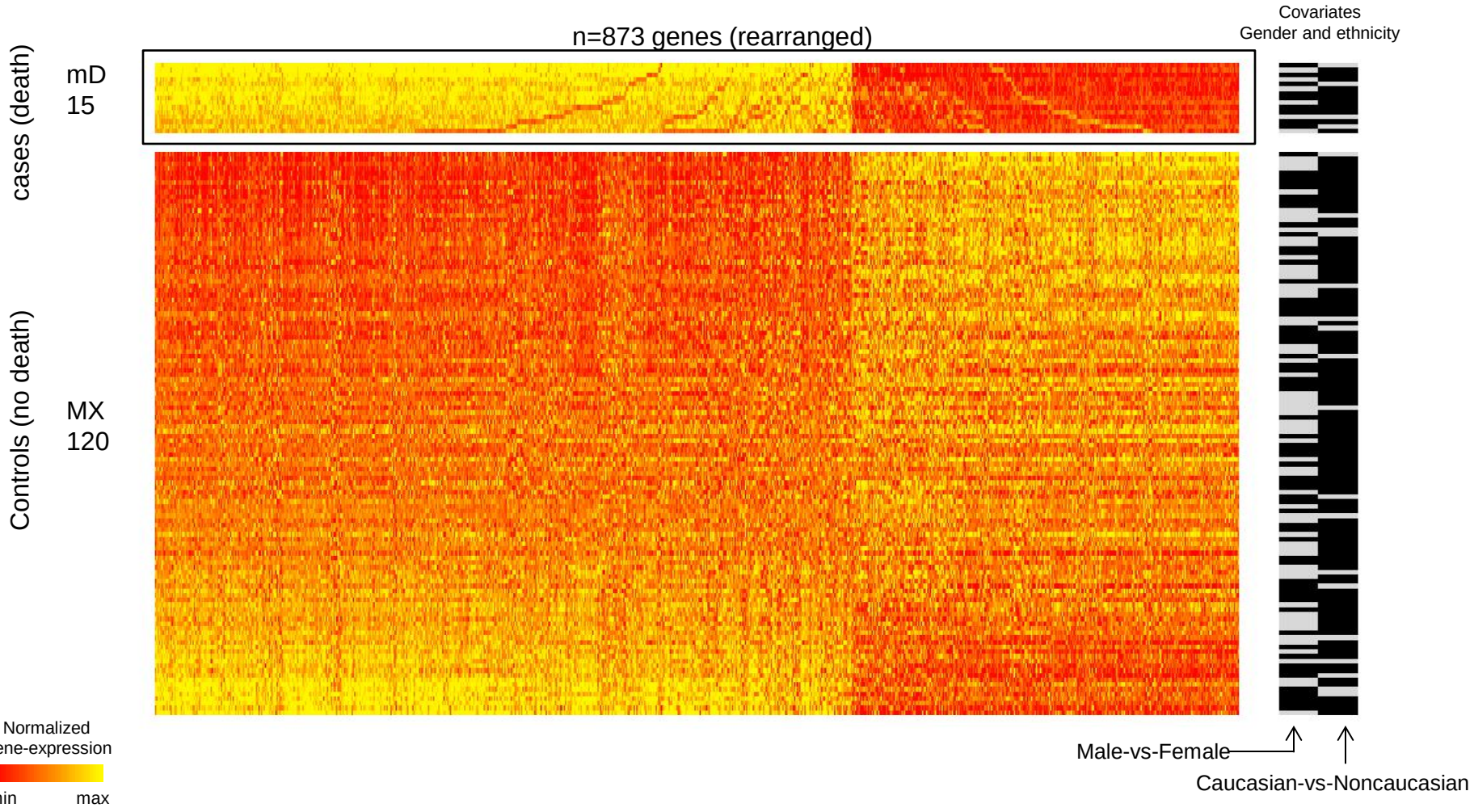
This observation is a consequence of the fact that this bicluster is 'significant'. That is, the bicluster that we found is large enough (and the pattern strong enough) that it is unlikely to have occurred by chance. Quantitatively, this bicluster has a P-value<0.040. This P-value is determined by our algorithm using a 'label-shuffled null hypothesis' (i.e., by comparing this bicluster to those found after randomly shuffling the case-control labels across the patients).

This significance implies that many of the genes implicated in this bicluster may affect cancer-related pathways: Indeed, a gene-enrichment analysis performed on these n=873 genes reveals a significant enrichment (p=2e-22) for extracellular matrix organization, as well as epithelial cell proliferation (p=7e-4), angiogenesis (p=1e-3), endothelial cell migration (p=3e-3) and many more pathways that influence colon cancer.

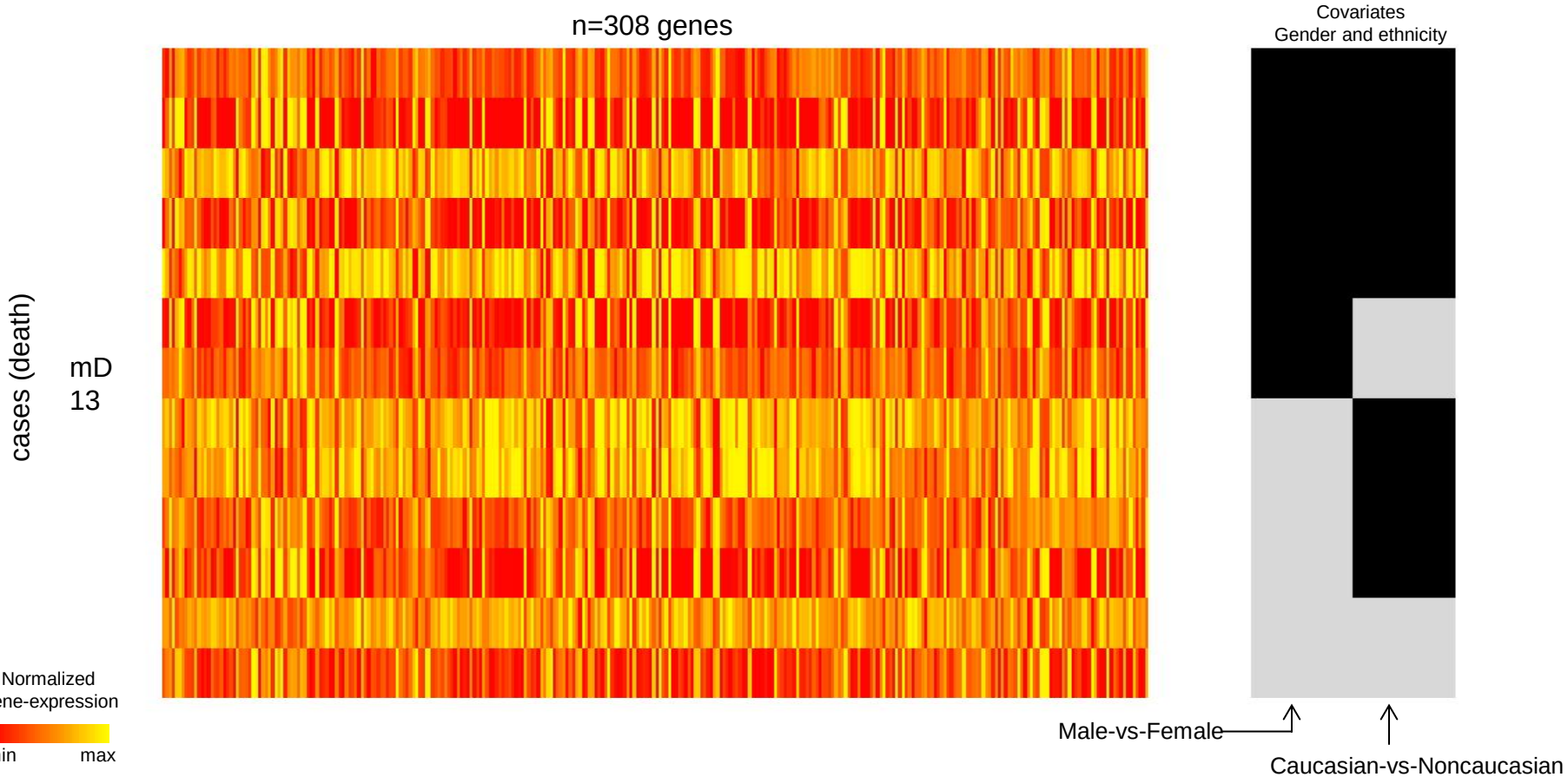


Taking a step back from our own methodology, one can independently confirm that this bicluster highlights a strong pattern of differential-expression. One way to quantify this strength is to measure, for each row, the dot-product with the row-wise average ‘u’ of the bicluster (which is very close to the dominant principal component of the bicluster).

This u-dot-product is uniformly high across the 15 cases in the bicluster, and uniformly lower across the control-rows. Moreover, the AUC of the bicluster u-dot-products vs the control u-dot-products is greater than 0.95. This by itself is not surprising, since the ‘u’ used is a function of the bicluster. Nevertheless, even if we calculate the u-dot-products across all 55 cases (and compare them to the 120 control u-dot-products) we still get an AUC of more than 0.70, which is significantly greater than the 0.50 we would see if the cases and controls were similarly distributed with respect to the u-pattern.

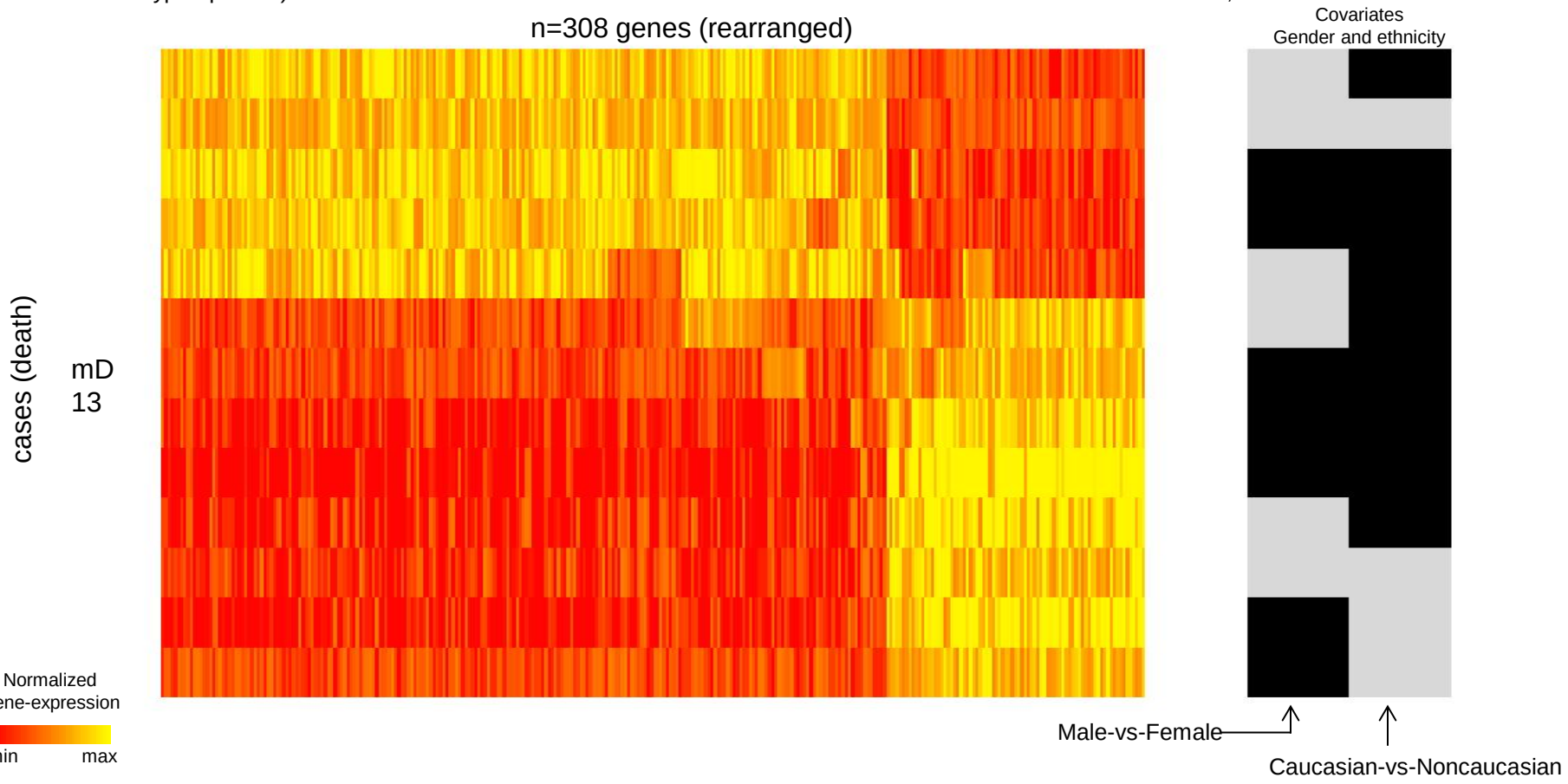


Here we illustrate our second example: another large bicluster – comprising mD=13 of the MD=55 cases and n=308 of the N=17942 genes – that was embedded in the previous matrix.
This bicluster consists of genes that, taken individually, are neither significantly over-expressed nor under-expressed – relative to the control population. Instead, these 308 genes are significantly co-expressed (i.e., either strongly correlated or anti-correlated) across a significant fraction of the case population (in this case 13/55 ~ 25% of the cases), without being as significantly co-expressed across a comparable fraction of the control population. I.e., the bicluster is 'low-rank'. In addition, this bicluster is relatively well balanced across the covariate categories. That is to say, the 13 patients that exhibit this co-expression are not all male, nor all female, nor all of one ethnicity.

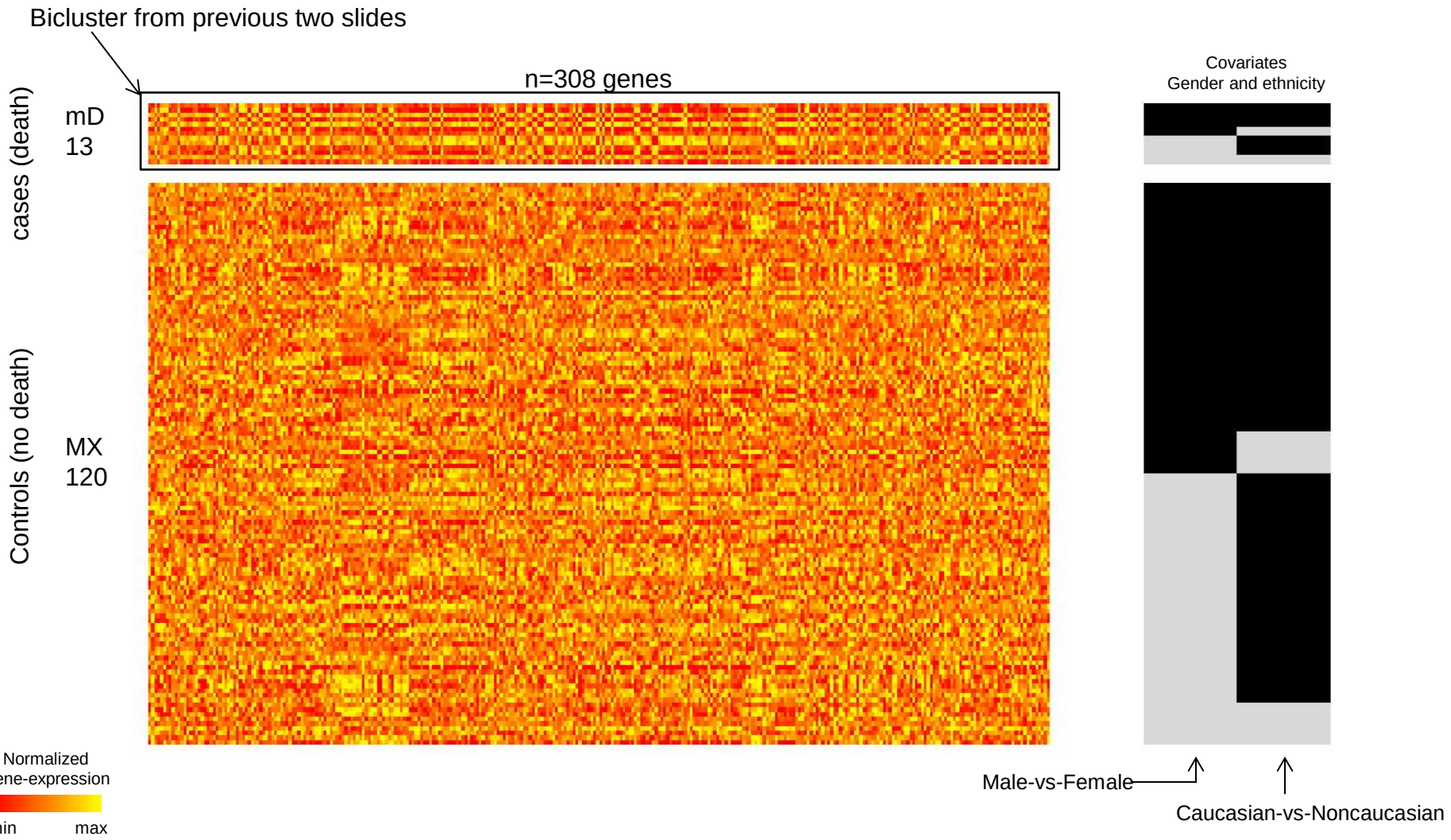


Here we illustrate our second example: another large bicluster – comprising mD=13 of the MD=55 cases and n=308 of the N=17942 genes – that was embedded in the previous matrix.
This bicluster consists of genes that, taken individually, are neither significantly over-expressed nor under-expressed – relative to the control population. Instead, these 308 genes are significantly co-expressed (i.e., either strongly correlated or anti-correlated) across a significant fraction of the case population (in this case 13/55 ~ 25% of the cases), without being as significantly co-expressed across a comparable fraction of the control population. I.e., the bicluster is 'low-rank'. In addition, this bicluster is relatively well balanced across the covariate categories. That is to say, the 13 patients that exhibit this co-expression are not all male, nor all female, nor all of one ethnicity.

While we've just stated that these genes are correlated, the depiction in the previous slide did not make that point particularly clear.
The reason being that the ordering we chose for the rows and columns was based on the output of our algorithm which, while structured in a particular way (see Appendix 2), does not directly illustrate the co-expression pattern within the discovered bicluster.
So, to show these correlations, we plot again the same bicluster, except with the rows and columns reorganized based on magnitude of the gene-expression data. As one can see, the submatrix indeed contains a significant co-expression pattern, with each patient either strongly correlated or anti-correlated with this stereotyped pattern. In fact, if we use the absolute value of the correlation as a measure of 'alignment', almost all the rows are aligned (with the stereotyped pattern) at a value of 70% or more. We will later refer to this kind of bicluster as a 'low rank' bicluster; in this case rank-1.

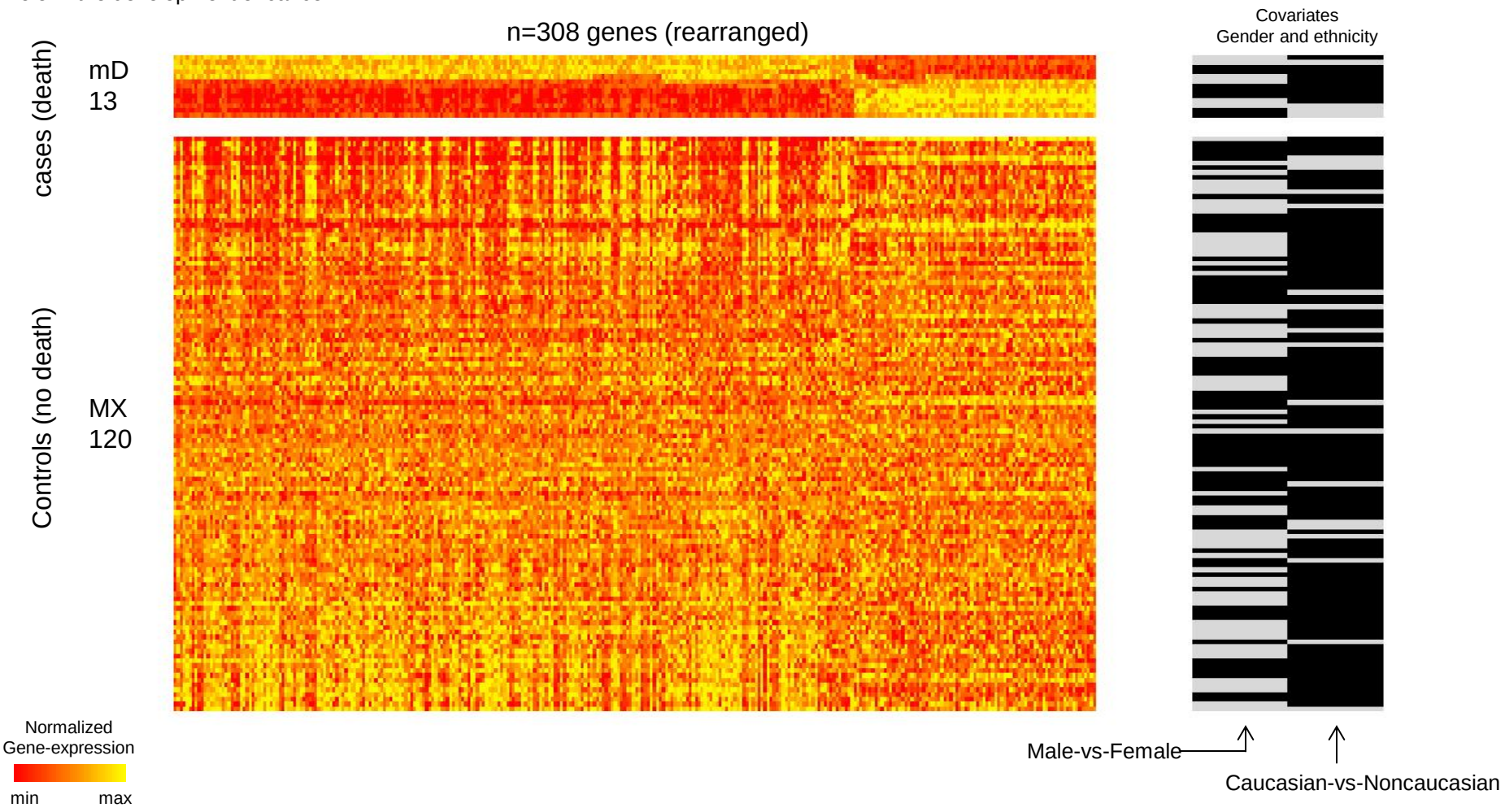


To illustrate that this stereotyped co-expression pattern is not comparably shared across the controls, we replot the bicluster again (vertically rescaled; top) and below we plot the control data...

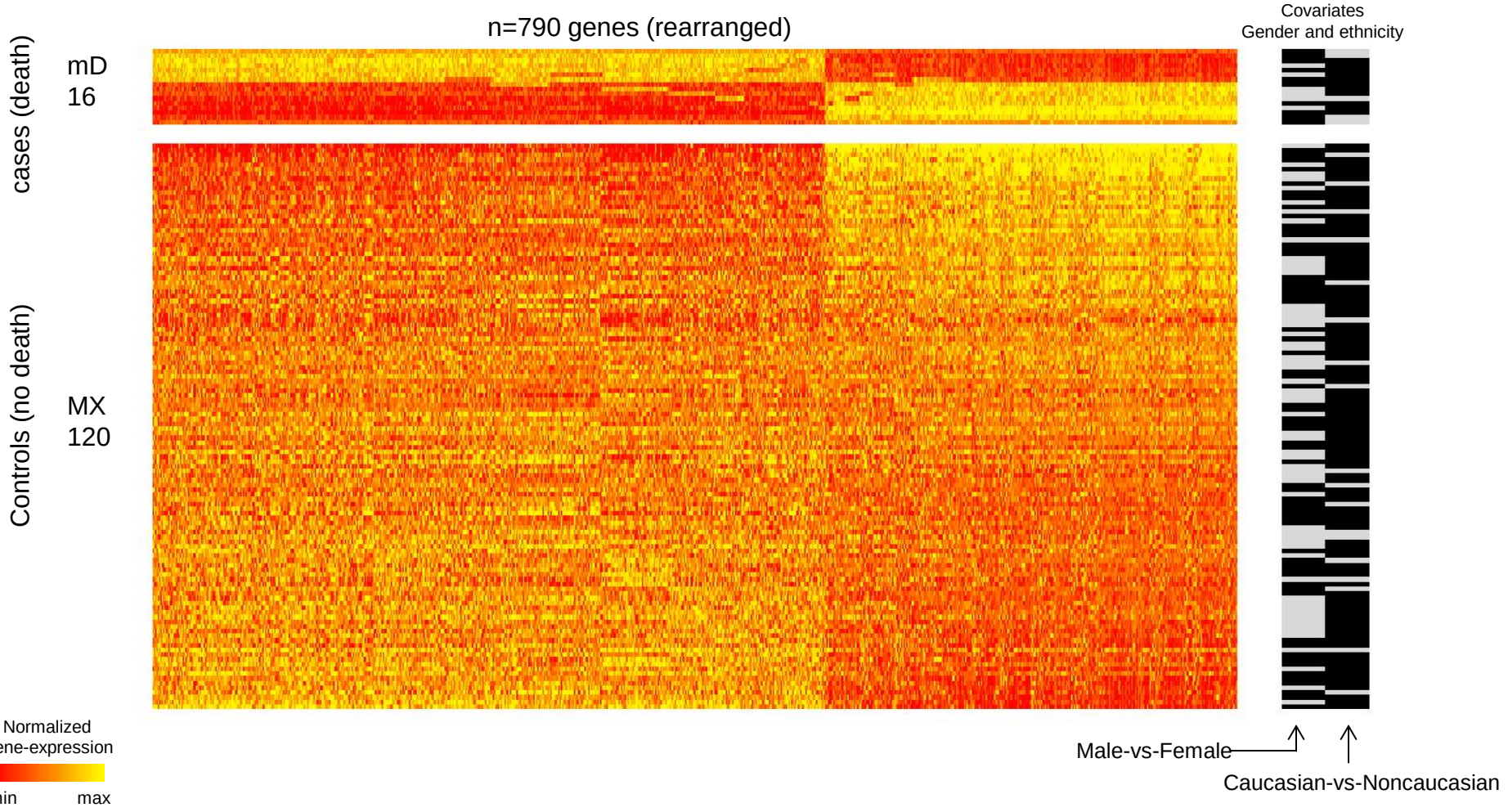


To illustrate that this stereotyped co-expression pattern is not comparably shared across the controls, we replot the bicluster again (vertically rescaled; top) and below we plot the control data – again reorganized in an attempt to reveal co-expression patterns. As one can see, while there are certainly some control patients that exhibit strong correlation or anti-correlation with the stereotyped gene-expression pattern of the bicluster, the majority are not so strongly aligned. In fact, for this example, only two of the controls (i.e., significantly less than 13/55) show a 70% alignment with the stereotyped pattern; most only exhibit an alignment around 20%.

How significant is this bicluster? We obtain a P-value of <0.035 for this bicluster by comparing it against the distribution of biclusters obtained under the 'label-shuffled' null-hypothesis (i.e., again formed from shuffling the case-vs-control labels). This level of statistical significance implies that many of the genes implicated in this bicluster may serve similar functions or affect the same pathway. This is indeed the case: a gene-enrichment analysis performed on these n=308 genes reveals a significant enrichment (p=5e-8) for Mitosis, as well as spindle-organization (p=6e-4), chromosome-condensation (p=2e-3), microtubule-anchoring (p=5e-3), response-to-ionizing-radiation (p=2e-2), and DNA-dependent DNA-replication (p=2e-2); all pathways that may play a role in the development of cancer.



Here we show our third example, another low-rank bicluster, plotted using the same conventions as before. This bicluster involves 16 patients, 13 of which are aligned with the bicluster-pattern at a level of >70%. Of the 120 controls, only 5 are aligned this strongly. The significance level of this bicluster (against a label-shuffled null hypothesis) is p<0.001. The 790 genes within this bicluster are enriched for: DNA dependent DNA replication (1e-4), spindle organization (2e-4), mitosis (3e-3), spindle checkpoint (3e-3), rRNA processing (4e-3), microtubule anchoring (4e-3), mitotic recombination (7e-3), chromatin assembly and disassembly (6e-3), and centromere complex assembly (2e-2).



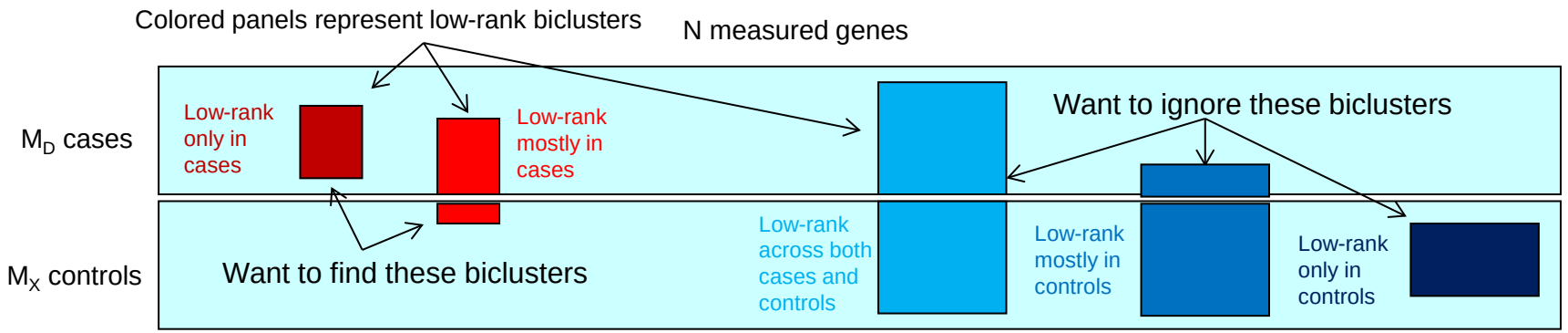
We defer a detailed description of our biclustering algorithm to the appendices at the end. However, in order to set the stage for this tutorial, we discuss some of the goals and considerations that will influence our methodology later on.

First, we need to decide – broadly speaking – what we are looking for. The search for biclusters is usually cast as the search for certain kinds of structure within the data; but exactly what that structure is can often be unsatisfyingly vague. In our case we will look for two very specific kinds of structure within our data – namely submatrices within our data that exhibit either (i) differential-expression, like the first example shown, or (ii) low-rank structure, like the second and third examples shown. While there are certainly many other structures that our algorithm will *not* find, we believe that the combination of (i) and (ii) above are sufficiently broad to capture many interesting phenomena. Our exposition will focus on the discovery (and validation) of biclusters exhibiting the second kind of structure – namely low-rank structure as described in Appendix-2, but almost all our methodology also applies to differential-expression as well (described more thoroughly in Appendix 1).

What does it mean for a bicluster to exhibit low-rank structure?

Roughly speaking, our basic algorithm (described in Appendix-2) reveals submatrices within the original matrix – defined in terms of subsets of rows and columns – which are low-rank. This means that these submatrices are dominated by their first few principal components. Put another way, most of the rows and columns within these submatrices are very similar to a small number (usually 1 or 2) of representative 'stereotypical' rows and columns. Put a third way, the rows and columns within a low-rank bicluster will be strongly correlated with one another.

Now, in addition to being low-rank, the biclusters that we are most interested in should also be *uncorrelated* with the controls. More specifically, if we imagine a bicluster consisting of mD of the MD case-patients, and n of the N genes, then the rows of the mD-by-n submatrix defining the bicluster should be strongly correlated with one another, but *not* strongly correlated with a significant subset of the MX controls taken across the same n genes. In terms of gene-expression data, this means that we would like to find n genes that are coexpressed across a large fraction (i.e., mD/MD) of the cases, but not across a similarly large fraction of the controls; such a bicluster would highlight genes that are involved in a case-specific pattern, in contrast to genes that are typically coexpressed across the entire population and have nothing to do with the disease.



That being said, we still need to specify what we mean by 'correlated'. For our purposes we define the correlation of two vectors x and y to be the cosine of the angle between x and y (i.e., the dot-product of x/|x| and y/|y|). Note that this definition of correlation does not involve centering each vector about its mean, and as a result depends on the 'origin' (i.e., zero-point) used to define x and y. One way of interpreting this correlation is as a score (ranging from -1 to +1) which is positive when the vectors x and y often share the same sign, and which is negative when the vectors x and y often differ in sign. The choice of origin is what determines the signs of the various entries of x and y.

With this in mind, we investigate a variety of normalization conventions for our data, corresponding to different choices for the origin, and different methods for selecting the genes to investigate:

- *Normalization-0*: First, we simply normalize each gene (i.e., each column) across all patients, including both cases and controls. Specifically, for each gene, we choose an origin (i.e., zero-point) which is the median gene-expression value for that column. We elect to use the median, rather than the mean, so that the origin is insensitive to outliers. Moreover, this use of the median ensures that the sign of each entry of the gene-vector (relative to the origin) is preserved under any monotonic change of variables (such as a log-transformation of the gene-expression data).
- *Normalization-1*: After performing the above normalization we often find that there are many genes which, individually, are strongly differentially-expressed; e.g., high across the majority of cases and low across the majority of controls. These genes carry a strong disease-specific signal, and are usually the focus of many other studies. While normalization-0 retains all these genes by default, we may want to focus on the other genes – those without such a strong signal. For this purpose we devise normalization-1, where we remove these strongly differentially-expressed genes, retaining only the genes that carry no significant signal by themselves. Our biclustering analysis later on will demonstrate that – indeed – many of these genes participate in disease-specific co-expression patterns (i.e,. low-rank biclusters), illustrating the richness of the mechanisms underlying gene-regulation.
- *Normalization-2*: Finally, we normalize each gene across each 'subgroup' of patients, where the subgroups are determined by the case-control status and covariate categories. In the case of this particular data-set, we have 8 subgroups (i.e., case+male+caucasian all the way to control+female+noncaucasian). For each gene and each subgroup we choose an origin which is the median gene-expression value within that column across that subset of rows. After employing this normalization convention no gene will be strongly differentially-expressed over the cases relative to the controls; any such 'first-order' signal will be eliminated, leaving only higher-order information. When we bicluster this data we will focus on this kind of higher-order information, revealing only co-expression-patterns which manifest relative to the origins associated with each subgroup of patients.

The examples shown at the beginning of this tutorial were found using these various normalization conventions:

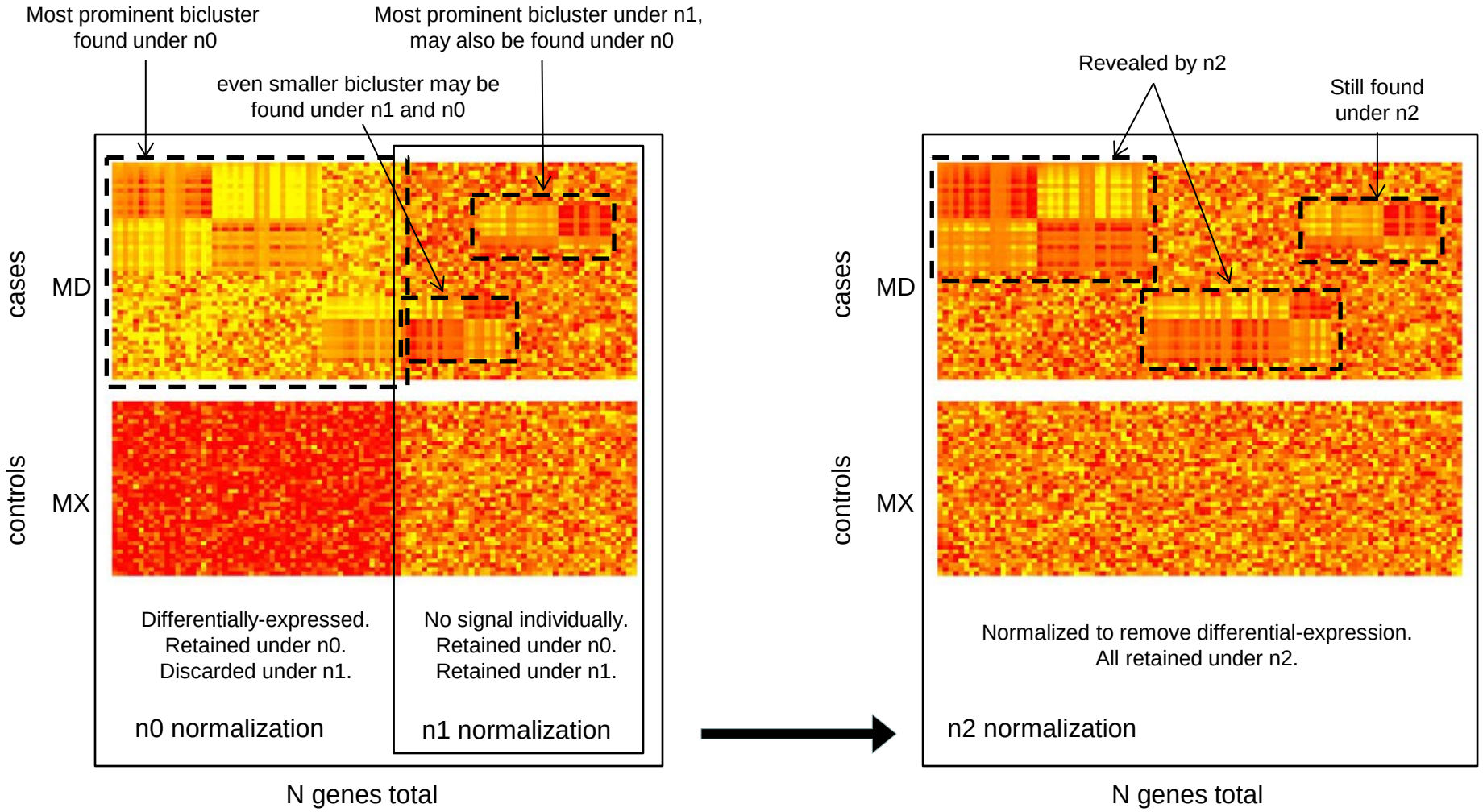
- Normalization-0: the 15-by-873 bicluster†. As we'll describe later, this bicluster was found during the first sweep under the 'nA_' paradigm.
- Normalization-1: the 13-by-308 bicluster. This bicluster was found during the first sweep under the 'n1x' paradigm.
- Normalization-2: the 16-by-790 bicluster. This bicluster was found during the first sweep under the 'n2_' paradigm.

†: This bicluster was found under the Normalization-0 convention, but not by using the algorithm described in Appendix-2.
Rather, this bicluster was found using the algorithm described in Appendix-1, which searches for biclusters exhibiting differential-expression.

Here we provide idealized illustrations of the kinds of biclusters that could be revealed by these different normalization conventions.

On the left we show a gene-expression array after normalization-0 (i.e., n0). The median of each gene across all cases and controls is 0, but many of the genes (left) are differentially expressed. These genes form a giant bicluster spanning all the patients (sometimes termed a 'gene-cluster'). This giant bicluster is the most prominent bicluster under n0. A secondary smaller bicluster exists within the other genes; this secondary bicluster is the most prominent bicluster under normalization-1 (i.e., n1).

On the right we show the same data under normalization-2 (i.e., n2). The median of each gene is set to 0 across the cases, and then again to 0 across the controls. This normalization reveals smaller biclusters which intersect the larger n0-bicluster found on the left.



We set up these normalization conventions (referred to as n0, n1 and n2, respectively) using the following routine:

Name: **`tutorial_create_GSE17536.m`**
Call from Matlab with:
```
>> tutorial_create_GSE17536;
```
This file sets up these normalization conventions (referred to as n0, n1 and n2, respectively).

Inputs:
This file requires the files: **`GSE17536.GPL570.pcl`** and **`GSE17536.gsms.txt`**, which hold the data and annotations, respectively.

Output:
This file generates the matlab save files **`GSE17536_n0D.mat, GSE17536_n1D.mat, GSE17536_n2D.mat`**, one for each normalization.
These files will be used as a basis to bicluster the data, both with and without taking into account the covariates.
Each of these output files contains within it the following variables:

- npats: the number of patients (i.e., rows) in the gene-expression array.
- ngenes: the number of genes (i.e., columns) in the gene-expression array.
- Pnames: the list of patient-ids associated with each row.
- Gnames: the list of gene-ids (entrez labels) associated with each column.
- B: the original gene-expression array, of dimension ngenes by npats.
- D: the transposed version of the normalized gene-expression array, of dimension npats by ngenes.
- g_ij: the list of column indices to use in the biclustering algorithm (e.g., the n1 convention excludes some columns).
- d_ij: the list of row indices which correspond to case-patients.
- x_ij: the list of row indices which correspond to control-patients.
- cov_mat: an array storing the categorical covariate information for each patient. Given T categorical covariates (e.g., gender and ethnicity) this matrix will be size npats-by-T. The entry cov_mat(nr,nc) is a {0,1} binary number indicating the classification of the nr^{th} patient with respect to the nc^{th} covariate-category.
- cov_cat: a npats-by-1 vector coding the covariate-categories of each patient. cov_cat is typically constructed by forming:
  `cov_cat = 1 + cov_mat * transpose(2.^[0:T-1]);`
  With T categorical covariates the vector cov_cat will include entries ranging from 1 to 2^T.

Later on we will use a 3-letter code to denote which normalization convention we adopt, and whether or not we will use covariates to guide our search:
'`n0_`': search for low-rank biclusters under the n0-normalization convention. No covariates used during the search. Runs `tutorial_lakcluster_2.m`.
'`n0x`': search for low-rank biclusters under the n0-normalization convention. Uses covariates to guide search. Runs `tutorial_lakcluster_2.m`.
'`n1_`': search for low-rank biclusters under the n1-normalization convention. No covariates used during the search. Runs `tutorial_lakcluster_2.m`.
'`n1x`': search for low-rank biclusters under the n1-normalization convention. Uses covariates to guide search. Runs `tutorial_lakcluster_2.m`.
'`n2_`': search for low-rank biclusters under the n2-normalization convention. No covariates used during the search. Runs `tutorial_lakcluster_2.m`.
'`n2x`': search for low-rank biclusters under the n2-normalization convention. Uses covariates to guide search. Runs `tutorial_lakcluster_2.m`.
'`nA_`': search for differentially-expressed biclusters under n0-normalization. No covariates used during the search. Runs `tutorial_dexcluster_1.m`.
'`nAx`': search for differentially-expressed biclusters under n0-normalization. Uses covariates to guide search. Runs `tutorial_dexcluster_2.m`.

Note that the first two letters of the 3-letter code determine the normalization-convention used
(e.g., Both the '`n0_`' and '`n0x`' conventions make use of the '`GSE17536_n0D.mat`' file).

Once we have chosen a normalization convention, say the n2x convention, we need to run the actual biclustering algorithm. The basic version of this algorithm can be called via the following routine:

Name: **`tutorial_w1.m`**
Call from Matlab (within the directory containing the m-files you downloaded) with something like:

```
>> path_base=sprintf('%s/dir_GSE17536',pwd); gen_fname='GSE17536_n2x'; gen_flag=1; prm_flag=0; n_to_find=2;
>> tutorial_w1(path_base,gen_fname,gen_flag,prm_flag,n_to_find,'frwd');
```

This function reads from the files created with **`tutorial_create_GSE17536.m`**, and calls a driver such as '**`tutorial_lackluster_1.m`**'

Inputs:

- path_base [string]: the directory in which the files sit. For example: '`/home/myname/dir_biclustering/dir_GSE17536/`'
- gen_fname [string]: a concatenation of the data-set prefix and a code determining which kind of bicluster to search for. The data-set prefix in this case is '`GSE17536`'. The second part of gen_fname – the code determining the type of bicluster to search for – is determined by the normalization convention (e.g., 'n2') followed by either an underscore (i.e., '_') if covariates are to be ignored, or a letter 'x' if covariates are to be used during the searching process. For example, gen_fname could take the form: '`GSE17536_n2_`' or '`GSE17536_n2x`'.
- gen_flag [an integer either 0 or 1]: whether or not to generate temporary files. 1 = generate, 0 = do not generate. This is typically set to 1 initially, and then later set to 0 when we perform multiple runs after the first for the sake of determining the bicluster's p-value.
- prm_flag [an integer >= 0]: whether or not to randomly permute case-control labels. prm_flag=0 implies no permutation (i.e., the original data), whereas any number > 0 implies a permutation (with different numbers corresponding to different random permutations). This option will be exercised to generate samples from the label-shuffled null hypothesis; ultimately allowing us to determine the p-value of any bicluster which we found under the prm_flag=0 setting.
- n_to_find [integer >= 1]: how many biclusters to search for. When this number is set to 1, the algorithm attempts to find the most dominant bicluster. If this number is set to 2, the algorithm finds the most dominant bicluster (exactly as in the n_to_find = 1 case) and then tries to search for the next most dominant bicluster. If n_to_find = 3, then the algorithm finds and removes the first two dominant biclusters before searching for the third, and so on.
- frwd_vs_back [string]: This variable, either 'frwd' or 'back' determines whether we search for biclusters among the cases, relative to the controls ('forward') or within the controls, relative to the cases ('backwards'). For the purposes of this tutorial we will use the 'frwd' setting.

Outputs:

- `*_out_xdrop.txt`: a file which stores the row- and column-indices in the order they were eliminated. This file is named [gen_fname]_[frwd_vs_back]_[n]_out_xdrop.txt, with n being the bicluster number (suppressed when n=0 for first bicluster), for example: '`GSE17536_n2x_frwd_out_xdrop.txt`', or '`GSE17536_n2x_frwd_1_out_xdrop.txt`'.
- `*_out_trace.txt`: a file which stores the average row- and column-scores generated by the biclustering algorithm during its iterations. This file is named similarly to the out_xdrop file. For example: '`GSE17536_n2x_frwd_out_trace.txt`' or '`GSE17536_n2x_frwd_1_out_trace.txt`'. These output-traces will be used to determine the p-value of the bicluster later on.
- `*_out_loopS.mat`: a file which stores the mean-squared correlations (dubbed the 'loop-scores') associated with the bicluster. This file is named similarly to the above. For example: '`GSE17536_n2x_frwd_out_loopS.mat`', or '`GSE17536_n2x_frwd_1_out_loopS.mat`'.
- `*_bc.txt`: a file which stores the subset of extracted row- and column-indices which delineate the bicluster (based on thresholding the loop-scores). This file is named in a manner similar to the above. For example, '`GSE17536_n2x_frwd_bc.txt`'.
- `*_plist.txt`: a file which stores the patient list associated with `*_bc.txt`. For example, '`GSE17536_n2x_frwd_plist.txt`'.
- `*_glist.txt`: a file which stores the gene indices associated with `*_bc.txt`. For example, '`GSE17536_n2x_frwd_glist.txt`'.
- `*_gname.txt`: a file which stores the gene names associated with `*_bc.txt`. For example, '`GSE17536_n2x_frwd_gname.txt`'. The gene indices and names stored in the `*_glist.txt` and `*_gname.txt` files are intended to be used in reference to the `gene_entrez_symbol.txt` database (for determining gene-enrichment later on).

Of course, running the biclustering algorithm on our original data (i.e., with `prm_flag==0`) will only show us the biclusters within that data. In order to determine whether or not these biclusters are statistically significant we need to compare them to the distribution of biclusters we would obtain under a suitable null-hypothesis. In our case we choose what corresponds to a 'label-shuffled' null-hypothesis: i.e., that the case-ctrl labels are actually arranged randomly, and have no disease-related structure. Under this null-hypothesis the biclusters which we found in the original data should be similar to the biclusters we would find if we were to shuffle the case-ctrl labels (of the various patients) randomly.

By setting `prm_flag>0`, we draw samples from our label-shuffled null-hypothesis in one of two ways (determined automatically by `gen_fname`):

- Null Hypothesis 'H_': If we are not considering covariates in our search (e.g., the 'n0_', 'n1_', 'n2_' and 'nA_' paradigms), then we shuffle the case-ctrl-labels indiscriminately while retaining the same number of case-patients and ctrl-patients.
- Null Hypothesis 'Hx': On the other hand, if we are considering covariates in our search (e.g., the 'n0x', 'n1x', 'n2x' and 'nAx' paradigms), then we shuffle the case-ctrl-labels as described above, except that we also respect covariate-categories (e.g., shuffle case-ctrl-labels within caucasian-males, then shuffle case-control-labels within caucasian-females, then within noncaucasian-males, then finally within noncaucasian-females).

For each sample we retain the `*_out_trace.txt` file, building a library of such output-traces stored, e .g., in the subdirectory `dir_GSE17536/GSE17536_n2x_prm/dir_out`.

This library will later allow us to obtain a p-value for each of the biclusters within the original data by comparing the output-trace of each original bicluster with the distribution of output-traces obtained under the appropriate label-shuffled null hypothesis (e.g., hypothesis H_ for the 'n1_' paradigm, hypothesis Hx for the 'n2x' paradigm, and so forth).

In terms of procedure, we implement the above via the following:

1. First run `tutorial_w1` on the original data (setting `gen_flag` to 1, `prm_flag` to 0, and `n_to_find` to some small finite number, say 2-6). This initial run will sweep through the original data (e.g., `GSE17536_n2D.mat`) `n_to_find` times, generating `n_to_find` index-orderings (e.g., `GSE17536_n2x_frwd_out_xdrop.txt, GSE17536_n2x_frwd_1_out_xdrop.txt,` etc.), as well as `n_to_find` output-traces (e.g., `GSE17536_n2x_frwd_out_trace.txt, GSE17536_n2x_frwd_1_out_trace.txt,` etc.). Because each successive sweep through the data depends on the results of the previous sweeps, these sweeps cannot be run in parallel, but instead must be run in series (i.e., one after another).
2. Independently from step 1, run tutorial_w1 again, but this time on various permutations of the original data (setting `gen_flag` to 0 and prm_flag to values 1,2,…,etc.). These permutations do not depend on one another or on the original biclusters, and so can most certainly be run in parallel either before or after biclustering the original data. For each permutation we only consider the most dominant bicluster, and so `n_to_find` will be automatically set to 1 during these runs. Each permutation will generate a trace of its own (e.g., `dir_GSE17536/GSE17536_n2x_prm/dir_out/GSE17536_n2x_prm001_frwd_out_trace.txt`).
3. After collecting several such traces, we can compare the distribution of permuted traces to the traces generated by the original data. This comparison is performed by '`tutorial_collate.m`', described on the next slide.

Note that, as described above, we actually impose different label-shuffled null-hypotheses (i.e., 'H_' vs 'Hx'), depending on whether or not we choose to use covariates to guide our search. One important feature is that Hx is 'nested within' (and thus more restrictive than) the hypothesis H_; i.e., the collection of case-control label-permutations associated with Hx is a subset of the collection of permutations associated with H_. As we will see later, the difference between these two null-hypotheses will be reflected in their derived p-values. Generally speaking, because Hx is nested within H_, the p-values associated with Hx will be larger (i.e., less significant) than those associated with H_.

Put another way, the hypothesis Hx allows only 'legitimate' label-shuffled permutations which respect covariate-categories, whereas the hypothesis H_ allows for many kinds of 'illegitimate' permutations which do not respect covariate-categories. These illegitimate permutations are less likely than the legitimate permutations to manifest the same kinds of biclusters as the original data. As a result, a smaller subset of samples from H_ are likely to 'look like the real data', and so the p-value associated with H_ will be smaller (i.e., more significant) than the p-value associated with Hx. We will mention this point once again after we discuss how to use our library of output-traces to generate a p-value.

Once we have run `tutorial_w1` multiple times (both on the original data and for several permutations of the original data), we are ready to use the library of output-traces within, e.g., `dir_GSE17536_n2x_prm/dir_out` to determine p-values for the original biclusters. This is done by calling:

Name: **`tutorial_collate.m`**
Call from Matlab (within the directory containing the m-files you downloaded) with something like:

```
>> tutorial_collate(path_base,gen_fname,'frwd');
```

This function reads from the various output-traces and calculates p-values for each bicluster.

The Inputs are a subset of those passed to `tutorial_w1.m`.

- path_base [string]: the directory in which the files sit.
- gen_fname [string]: a concatenation of the data-set prefix and a code determining which kind of bicluster to search for.
- frwd_vs_back [string]: For the purposes of this tutorial we will use the 'frwd' setting.

Outputs:

- `*_collate.mat`: This file is named [gen_fname]_[frwd_vs_back]_collate.mat, for example: '`GSE17536_n2x_frwd_collate.mat`'. This file stores, among other things, the arrays `tr_base_ra, tr_ra_ and tr_cmb_pval`.

- `tr_base_ra` is an array of size n_iterations-by-NO, where NO is the number of successive times the biclustering algorithm was run on the original data (i.e., barring any missing data, NO should equal the `n_to_find` passed to `tutorial_w1`). The entry `tr_base_ra(m,n)` stores the average row-scores observed during the mth iteration of the nth successive sweep through the original data. In the typical situation where the biclustering algorithm eliminates one case-patient per iteration, there will be (MD-m) case-patients remaining after the mth iteration of the biclustering algorithm. If we are using covariates to guide our search, then for sufficiently large m there will be insufficiently many case-patients remaining for all covariate-categories to be adequately represented. We take this into account below; for each n we restrict our calculations to involve only iterations m for which all covariate-categories are adequately represented.

- `tr_ra_` is an array of size n_iterations-by-NP, where NP is the number of times the biclustering algorithm was run on different permutations of the original data. The entry `tr_ra_(m,n)` stores the average row-scores observed during the mth iteration of the nth permutation of the original data. Similarly to `tr_base_ra`, we restrict our attention for each n to iterations m for which all the covariate-categories are represented.

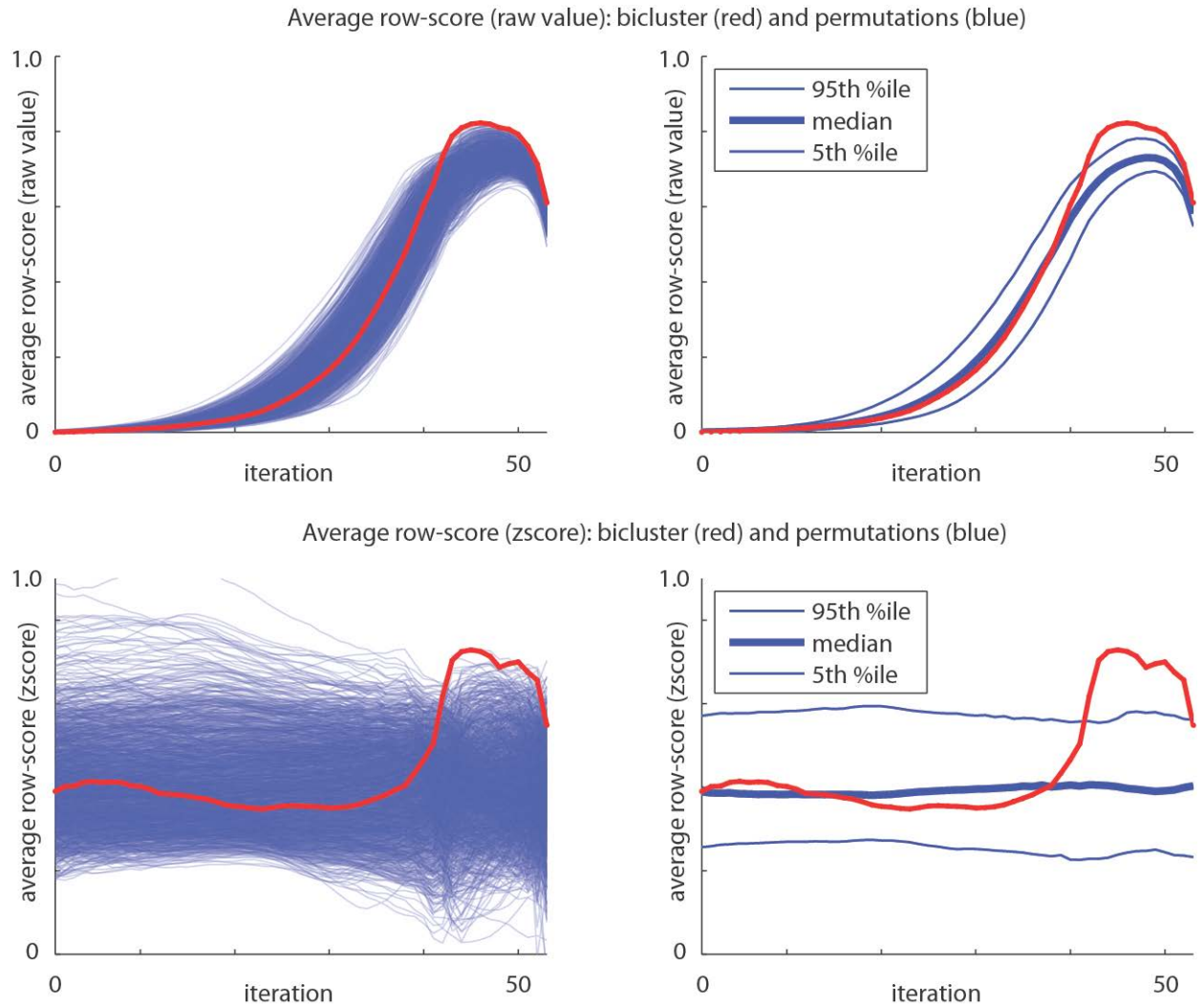
- `tr_cmb_pval` is an array of size NO. Each entry `tr_cmb_pval(n)` stores the p-value (calculated from the label-shuffled distribution) which is associated with the performance of the nth successive sweep through the original data. This p-value is obtained by considering a combination of:
  1. the maximum raw-value of `tr_base_ra(tmp_ij,n)`,
  2. the maximum z-score of `tr_base_ra(tmp_ij,n)`, calculated relative to the distribution `tr_ra_`,
  3. the average value of `tr_base_ra(tmp_ij,n)`,

  where the index array `tmp_ij` includes only those iterations for which all covariate-categories are represented. We compare each statistic above against the corresponding distribution of that same statistic across the label-shuffled distribution, obtaining `tr_cmb_pval(n)` by using standard techniques for multiple-hypothesis testing. Note that this p-value only depends on the `*_out_trace.txt` files, and not on the precise manner in which the bicluster was delineated. Thus, the p-value `tr_cmb_pval(n)` can be thought of as the p-value of the output-ordering stored in the nth `*_out_xdrop.txt` file, and not merely the p-value of the associated `*_bc.txt` file.

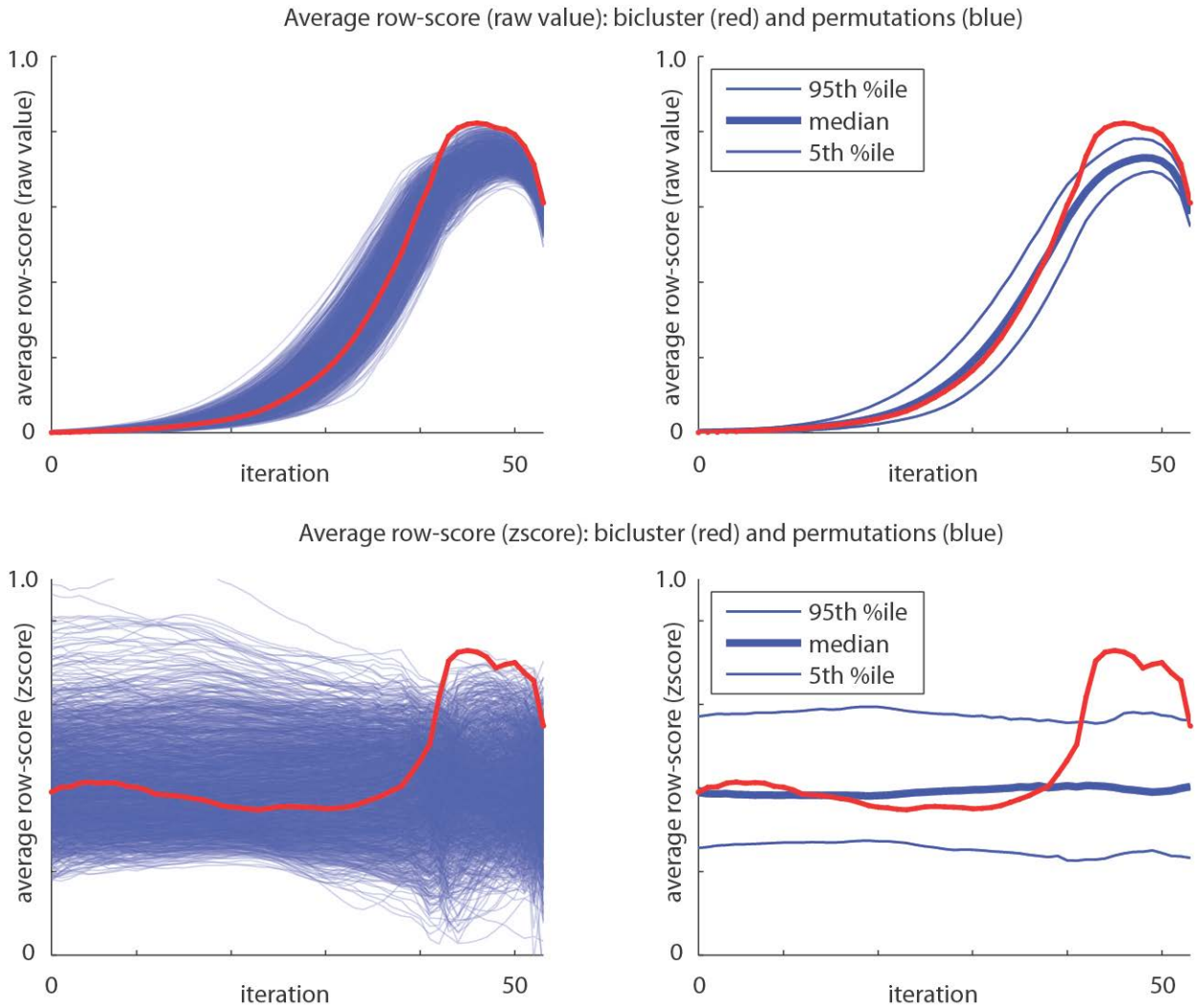
To illustrate how we use our library of label-shuffled output-traces to determine p-values, we use the third bicluster presented at the beginning as a case-study.

To recap: this bicluster was found during the initial sweep through the 'n2' normalization under the 'n2_' paradigm (i.e., without using covariates to guide the search). The output trace associated with this bicluster is found within '`GSE17536_n2__frwd_out_trace.txt`', and contains the average-row-scores for iterations m = 1,…,53 (only 53 iterations, rather than 55, since the algorithm terminates when only 2 patients remain). This same data is contained within the '`tr_base_ra(m,1)`' variable stored within '`GSE17536_n2__frwd_collate.mat`'.

In the figure below we compare these average-row-scores with the n=1,..,2048 trials from the label-shuffled null-hypothesis (stored in '`dir_GSE17536_n2__prm`'). This data is stored within the '`tr_ra_(m,n)`' variable within the same .mat file.

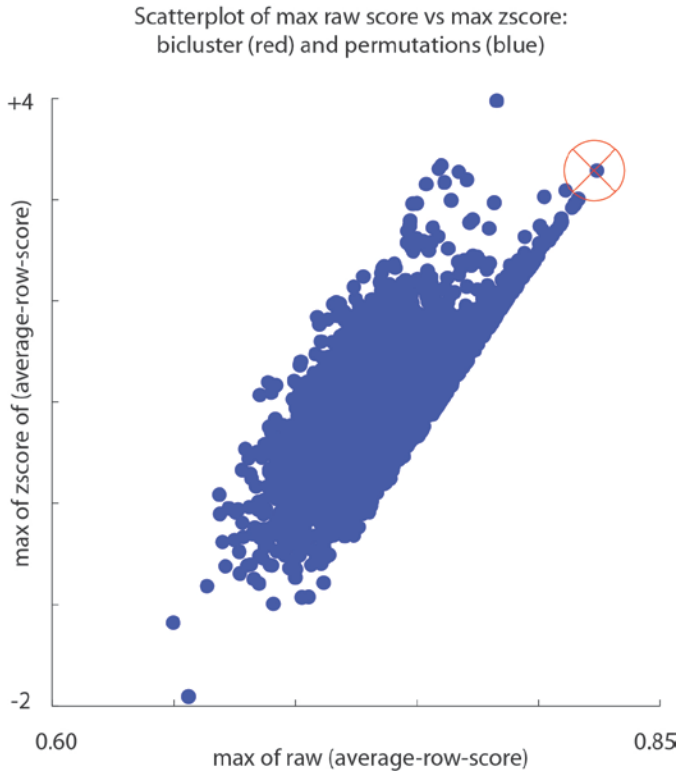


On the top-left we show the raw-values of the original-trace `tr_base_ra(:,1)` in red, and the label-shuffled traces `tr_ra_(:,:)` in blue. On the top-right we show the percentiles of the label-shuffled distribution (calculated independently per iteration: e.g., '`prctile(tr_ra_,95,2)`').
On the bottom-left we show the z-scores of the original-trace in red, calculated as:
`zr_base_ra(:,1) = (tr_base_ra(:,1) - mean(tr_ra_,2))./std(tr_ra_,[],2);`
and the z-scores of each label-shuffled trace in blue, calculated as:
`zr_ra_(:,n) = (tr_ra_(:,n) - mean(tr_ra_,2))./std(tr_ra_,[],2);.`
On the bottom-right we show the percentiles of this z-score distribution (e.g., '`prctile(zr_ra_,95,2)`').



By examining these plots it is clear that there are certain features of `tr_base_ra(:,1)` which are different from the majority of the label-shuffled distribution. In this case, the original trace reaches an absolute and relative maximum – at around 45 iterations – which is much higher than that reached by the label-shuffled distribution. This implies that, when biclustering the original data, the submatrix remaining after 45 iterations was far more sharply correlated than the corresponding submatrices observed after 45 iterations under the label-shuffled null-hypothesis.
To quantify this difference we can measure either:
(i) the maximum raw-score of the trace, calculated as '`tr_base_max_ra(1) = max(tr_base_ra(:,1))`', or
(ii) the maximum z-score calculated as '`tr_base_zmx_ra(1)= max(zr_base_ra(27:53,1))`'.
The max-raw-score will highlight traces that are high in the absolute sense, and so we measure this maximum across all iterations. The max-z-score, on the other hand, highlights traces that are relatively high, and so we measure this maximum over only the last half of the iterations (so as to avoid the initial iterations where the standard-deviation of the label-shuffled distribution is close to 0 and the z-score is poorly defined).

To the right we show a scatterplot of these maximum values, with the values of the original trace indicated by the red-⊗, and the analogous values for each label-shuffled permutation:
(i) '`tr_max_(n) = max(tr_ra_(:,n))`', and
(ii) '`tr_zmx_ = max(zr_ra_(:,n)`'
indicated by blue dots.



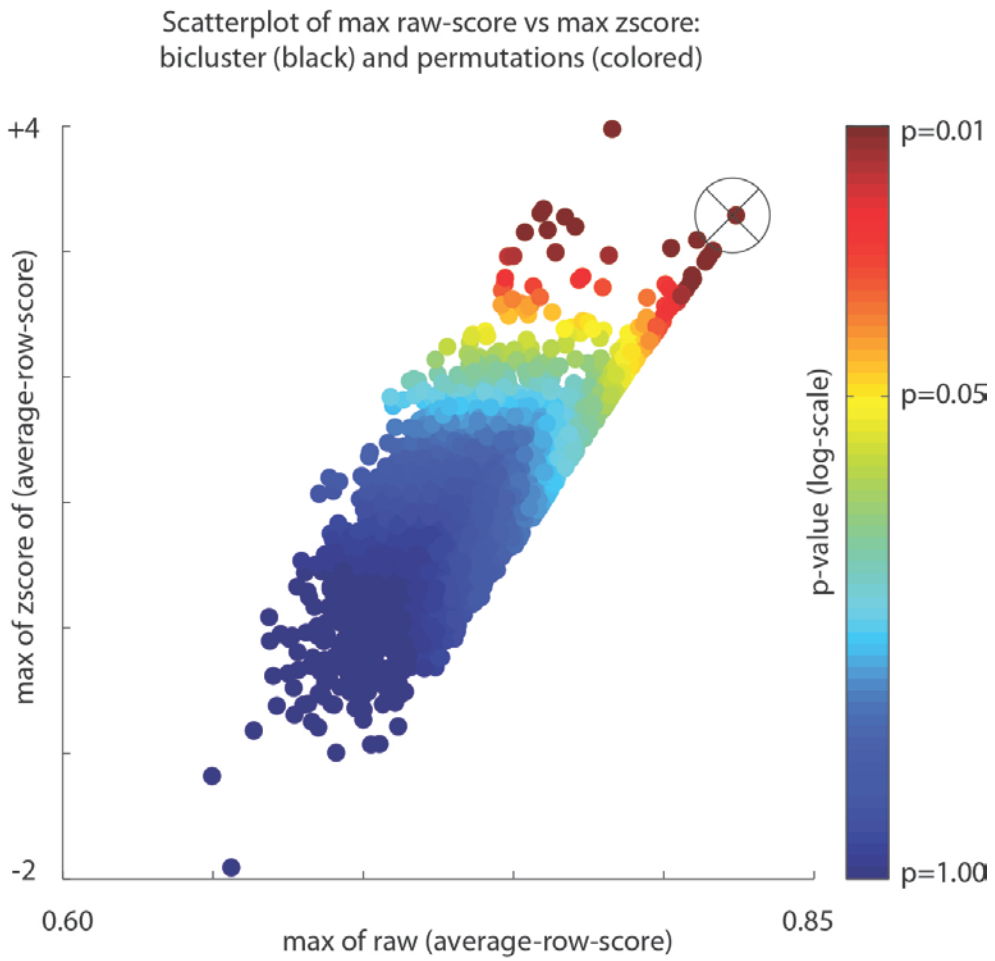
By considering the placement of the original trace with respect to the label-shuffled distribution we can determine a p-value for the original trace.
In general, the p-value of any location w (in the xy-plane) is equal to the fraction of blue-dots that have an x-position larger than x(w) or a y-position larger than y(w), where x(w) and y(w) are the x- and y-percentiles associated with the most extreme coordinate of w. Specifically:

```
tr_max_pval = length(find(tr_max_>tr_base_max_ra(1))/2048;
tr_zmx_pval = length(find(tr_zmx_>tr_base_zmx_ra(1))/2048;
fraction_best = min(tr_max_pval,tr_zmx_pval);
ls_max = find(tr_max_>=prctile(tr_max_,100*(1-fraction_best)));
ls_zmx = find(tr_zmx_>=prctile(tr_zmx_,100*(1-fraction_best)));
tr_cmb_pval = length(union(ls_max,ls_zmx))/2048;
```

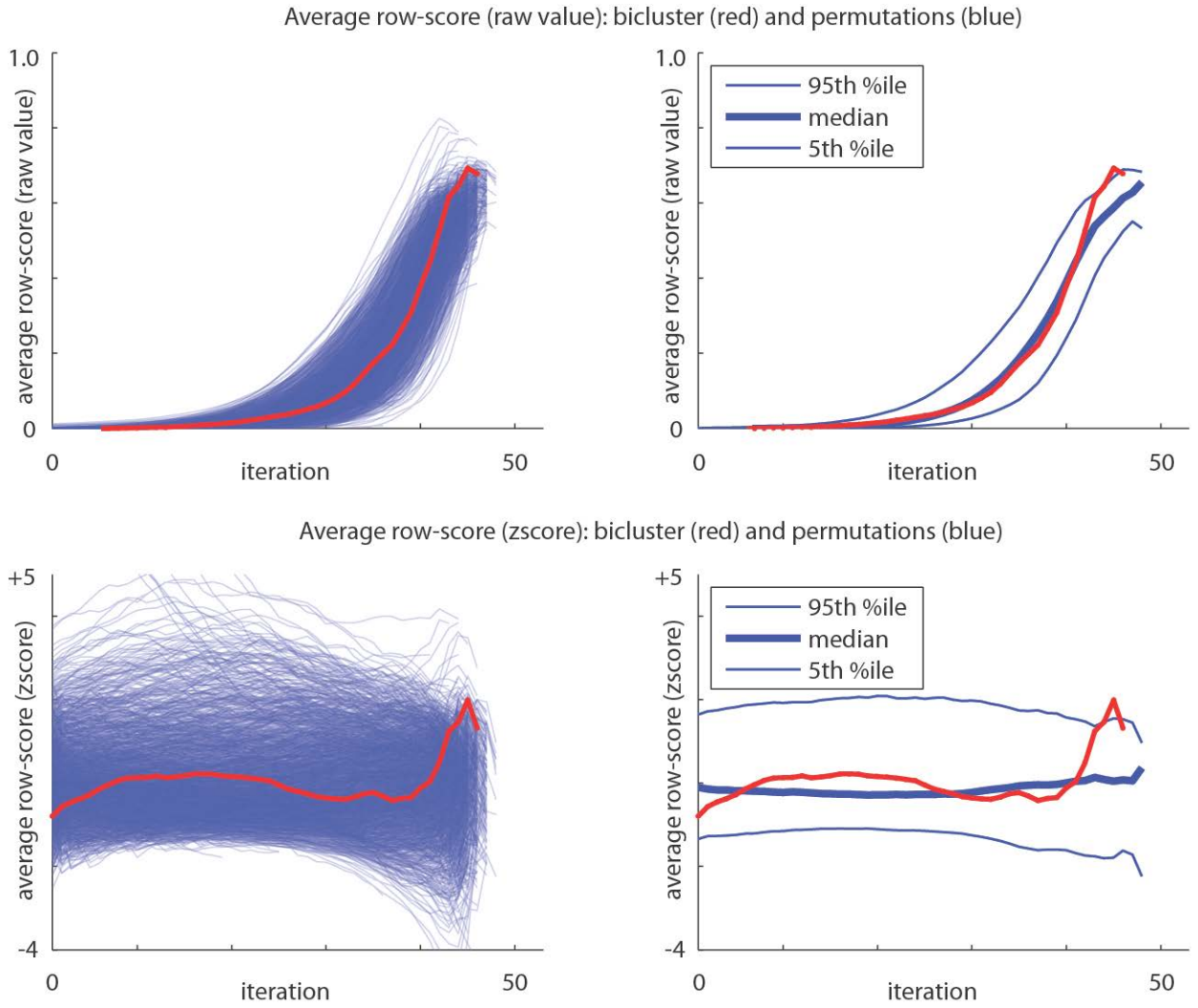
This allows us to calculate a 'combined-pvalue' `tr_cmb_pval` which accounts for multiple hypothesis testing while considering the correlations in the various measurements†.

†: We actually also measure one additional feature of the trace, namely:
(iii) the average raw-score taken across all iterations, calculated as: '`tr_base_avg_ra(1) = mean(tr_base_ra(:,1))`'.
This is an indicator of how broad – rather than how sharp – the correlations are. While the average raw-score is not remarkable for this bicluster, there are other biclusters where this feature is of note. We include this third measure in our p-value calculations, modifying the formula for multiple-hypothesis testing accordingly.

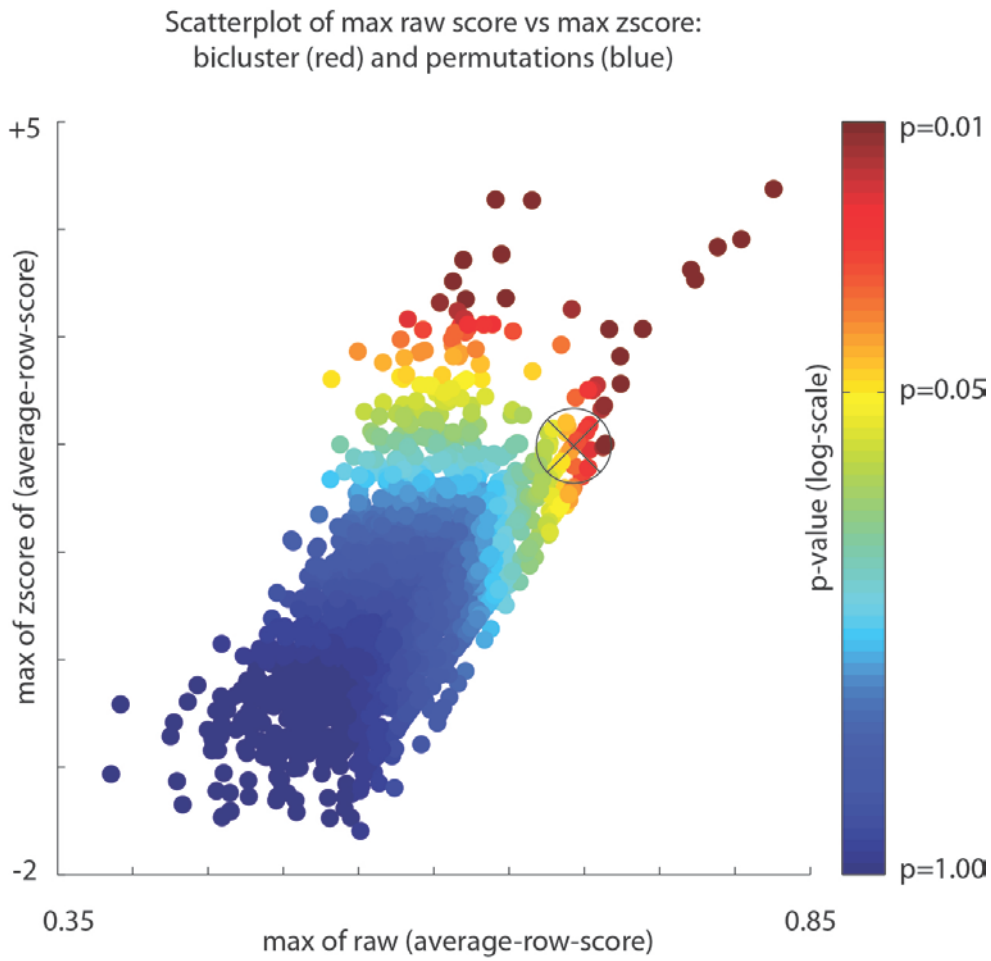
The p-values associated with each of the label-shuffled trials is shown in the graph below (see colorbar to the right). The point corresponding to the original trace is now colored black, and has a combined p-value of <0.001.



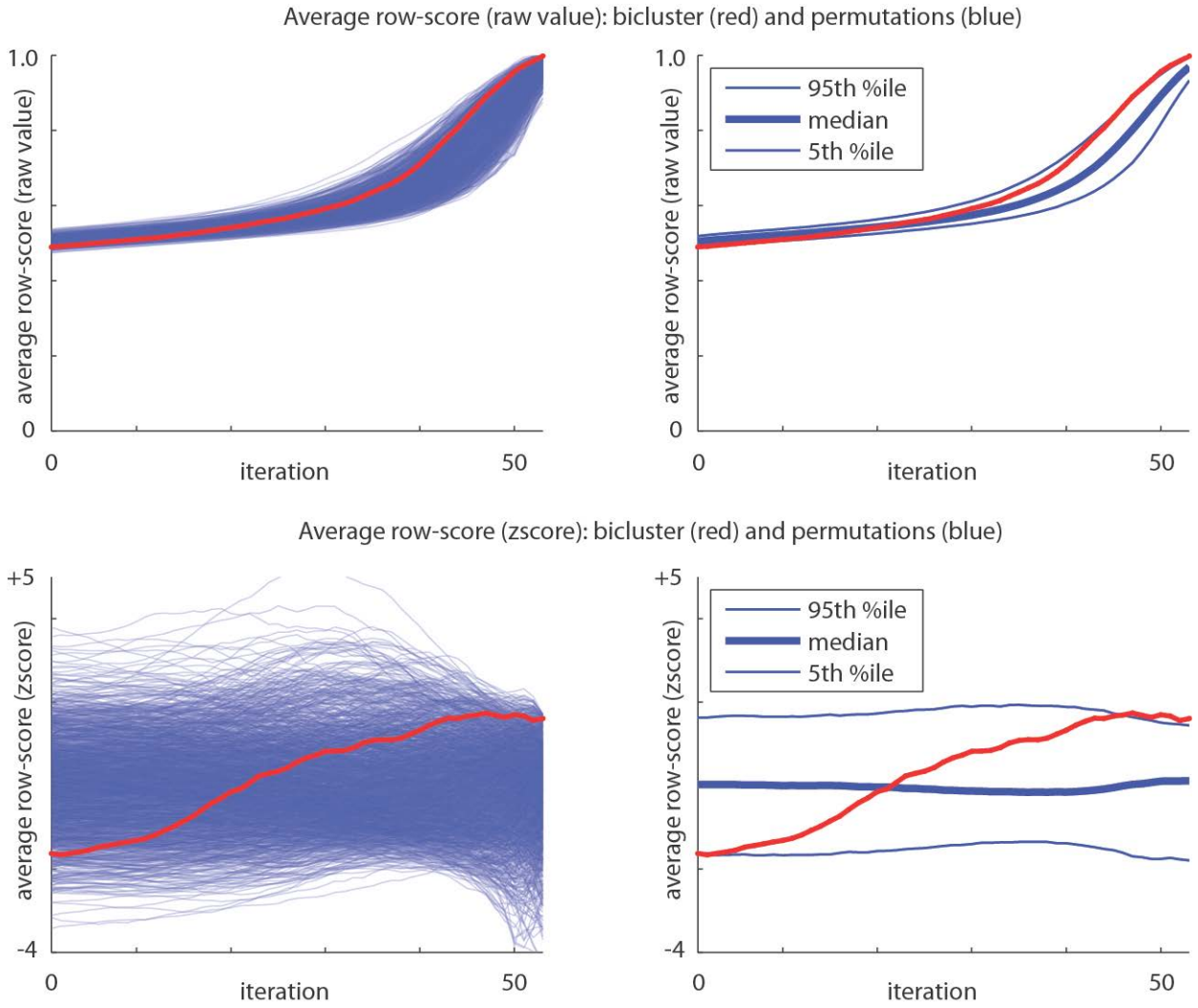
Here we show an analogous figure for the second bicluster presented at the beginning. This bicluster was found during the initial sweep through the ‘n1’ normalization under the ‘n1x’ paradigm (i.e., this time using covariates to guide the search). Because the covariates were used to guide the search, we only consider iterates for which all covariate-categories were represented. Other than this modification, the calculation of the combined p-value is identical, and in this case yields a p-value of 0.0347.



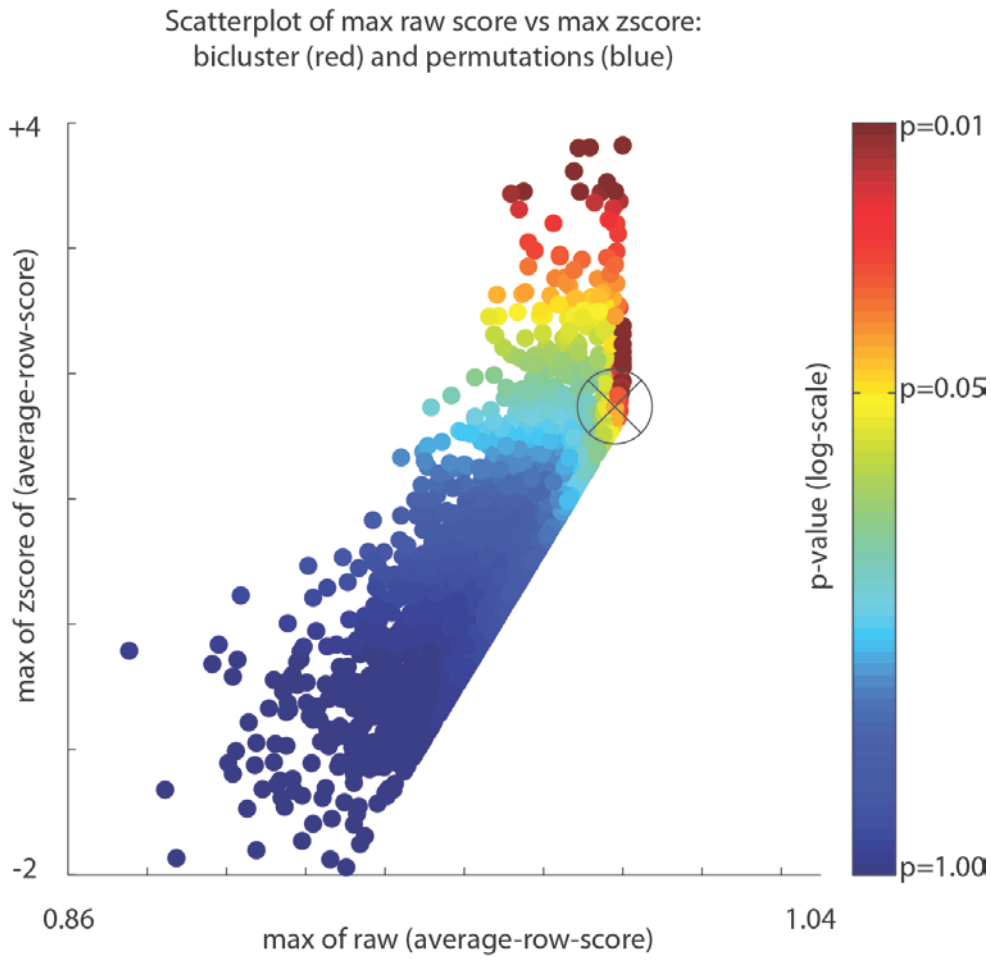
Here we show an analogous figure for the second bicluster presented at the beginning. This bicluster was found during the initial sweep through the 'n1' normalization under the 'n1x' paradigm (i.e., this time using covariates to guide the search). Because the covariates were used to guide the search, we only consider iterates for which all covariate-categories were represented. Other than this modification, the calculation of the combined p-value is identical, and in this case yields a p-value of 0.0347.



Finally, we show an analogous figure for the first bicluster presented at the beginning. This bicluster was found during the initial sweep through the 'n0' normalization under the 'nA_' paradigm (i.e., without using covariates to guide the search). As in the 'n2_' paradigm we use all the iterations. The p-value in this case is 0.0397.



Finally, we show an analogous figure for the first bicluster presented at the beginning. This bicluster was found during the initial sweep through the ‘n0’ normalization under the ‘nA_’ paradigm (i.e., without using covariates to guide the search). As in the ‘n2_’ paradigm we use all the iterations. The p-value in this case is 0.0397.



While it is certainly important to determine the statistical-significance of each bicluster, it is also important to determine whether or not each bicluster is enriched for any biological processes. To perform this enrichment we use the software `seek.GeneEnrichTest` (seek.princeton.edu), called via:

Name: **`tutorial_genri.m`**
Call from Matlab (within the directory containing the m-files you downloaded) with something like:

```
>> tutorial_genri(path_base,gen_fname,'frwd');
```

This function is designed to organize a gene-enrichment analysis of the gene-sets stored in files such as `GSE17536_n2x_genes_frwd_glist.txt`. The actual program we use to perform the gene-enrichment analysis is seek.GeneEnrichTest, which is written in java (see seek.princeton.edu for code and documentation).

The Inputs are a subset of those passed to `tutorial_w1.m`.

- path_base [string]: the directory in which the files sit.
- gen_fname [string]: a concatenation of the data-set prefix and a code determining which kind of bicluster to search for.
- frwd_vs_back [string]: For the purposes of this tutorial we will use the 'frwd' setting.

Outputs:

- `*_genri.txt`: This file is named [gen_fname]_[frwd_vs_back]_[n]_genri.txt, where n is the bicluster number (n ranges from 0 to n_to_find-1, and the suffix is suppressed when n==0 for the first bicluster). For example: '`GSE17536_n2x_frwd_genri.txt`' or '`GSE17536_n2x_frwd_1_genri.txt`'. Each such text file lists the significant pathways associated with the gene-sets demarcated by the respective biclusters (contained within the various `*_glist.txt` files). The `*_genri.txt` files are generated using the default gene-ontology database provided by seek. The format of each of these files is explained in more detail within the actual file `tutorial_genri.m`. (try '`>> help tutorial_genri`' from the matlab prompt).
- `*_gslim.txt`: These files are similar to the `*_genri.txt` files except that, instead of using the default gene-ontology database, we use the '`experimental_bp_slim`' database (which is curated more strictly).

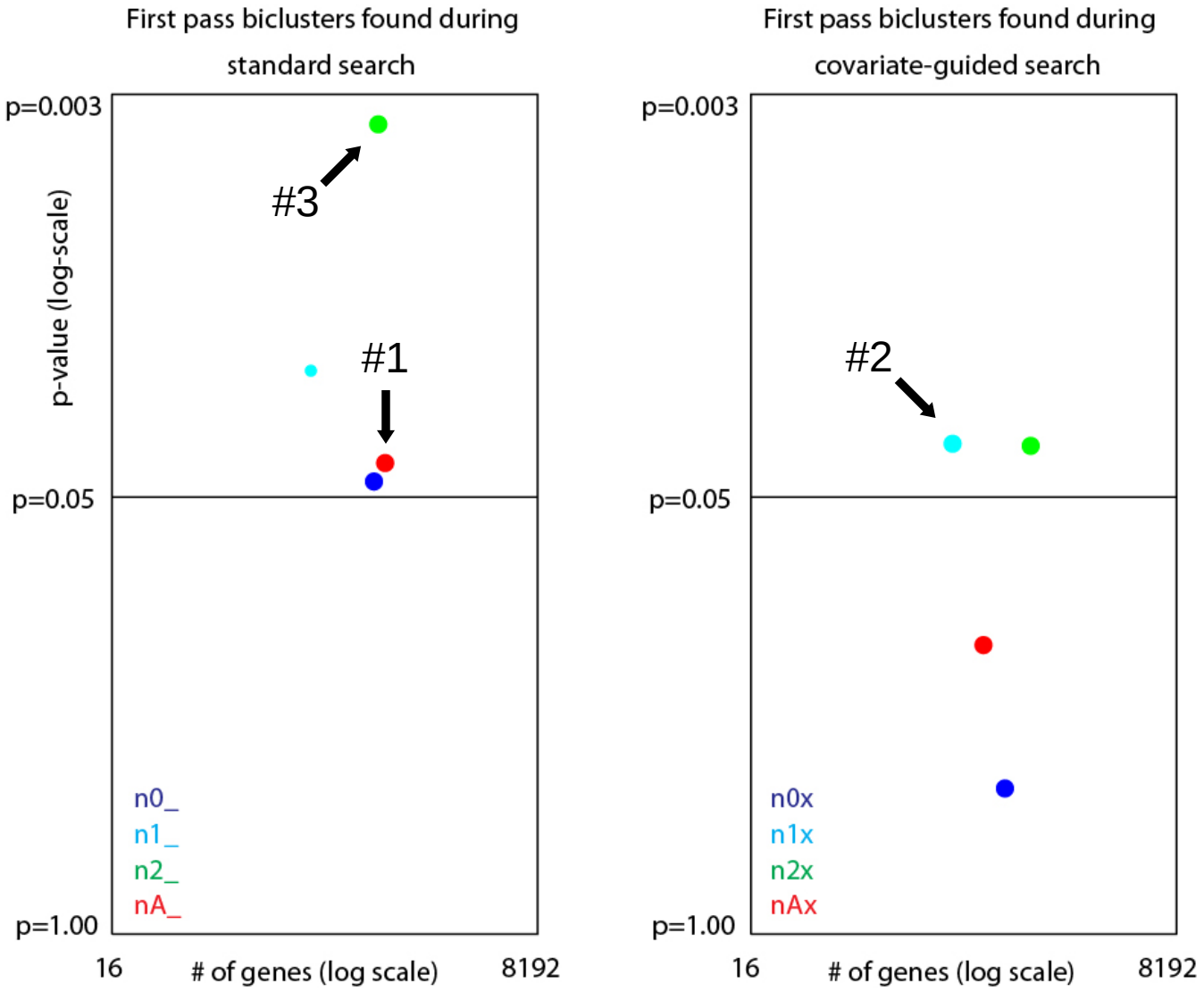
Obviously, seek is not necessary; one can certainly use other programs to determine the gene-enrichment of each bicluster. One caveat is that that, for this data-set, the gene IDs (numbers) associated with each gene align with the entrez database (and the entrez symbols), and not necessarily with the HUGO symbols. This is because this data is curated by the gene-expression-omnibus, which uses the entrez database. Conversion from entrez to HUGO can be performed if necessary (see, e.g., `www.broadinstitute.org/cancer/software/gsea/wiki/index.php/FAQ`).

Once gene-enrichment analysis is finished, we use the functions `tutorial_summarize.m` and `tutorial_plot.m` to generate plots summarizing certain aspects of the biclusters we found. We don't discuss these functions here, since they depend strongly on the structure of the gene-enrichment files `*_genri.txt` and `*_gslim.txt`, which in turn depend on the vagaries of seek. Nevertheless, documentation for these functions is provided within the actual files themselves (e.g., try '`>> help tutorial_summarize`' from the matlab prompt).

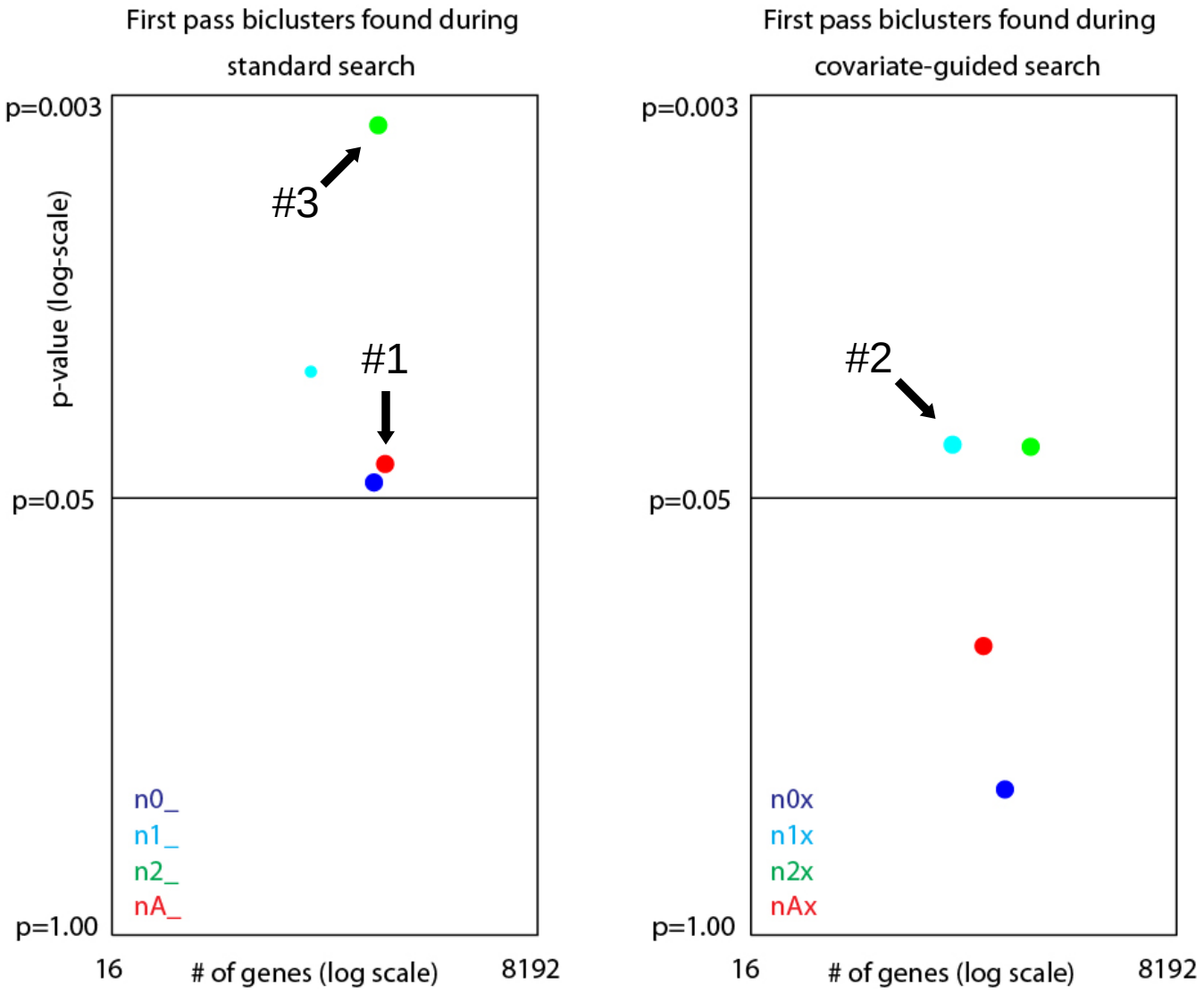
These summary plots are shown in the next few slides.

To begin with we consider, for each of our search paradigms, the first bicluster found (i.e., the bicluster found with `n_to_find==1`). These biclusters are shown on the scatterplots below, with those found during a covariate-guided search (i.e,. 'n0x', 'n1x', 'n2x' and 'nAx') on the subplot to the right, and the others on the left.

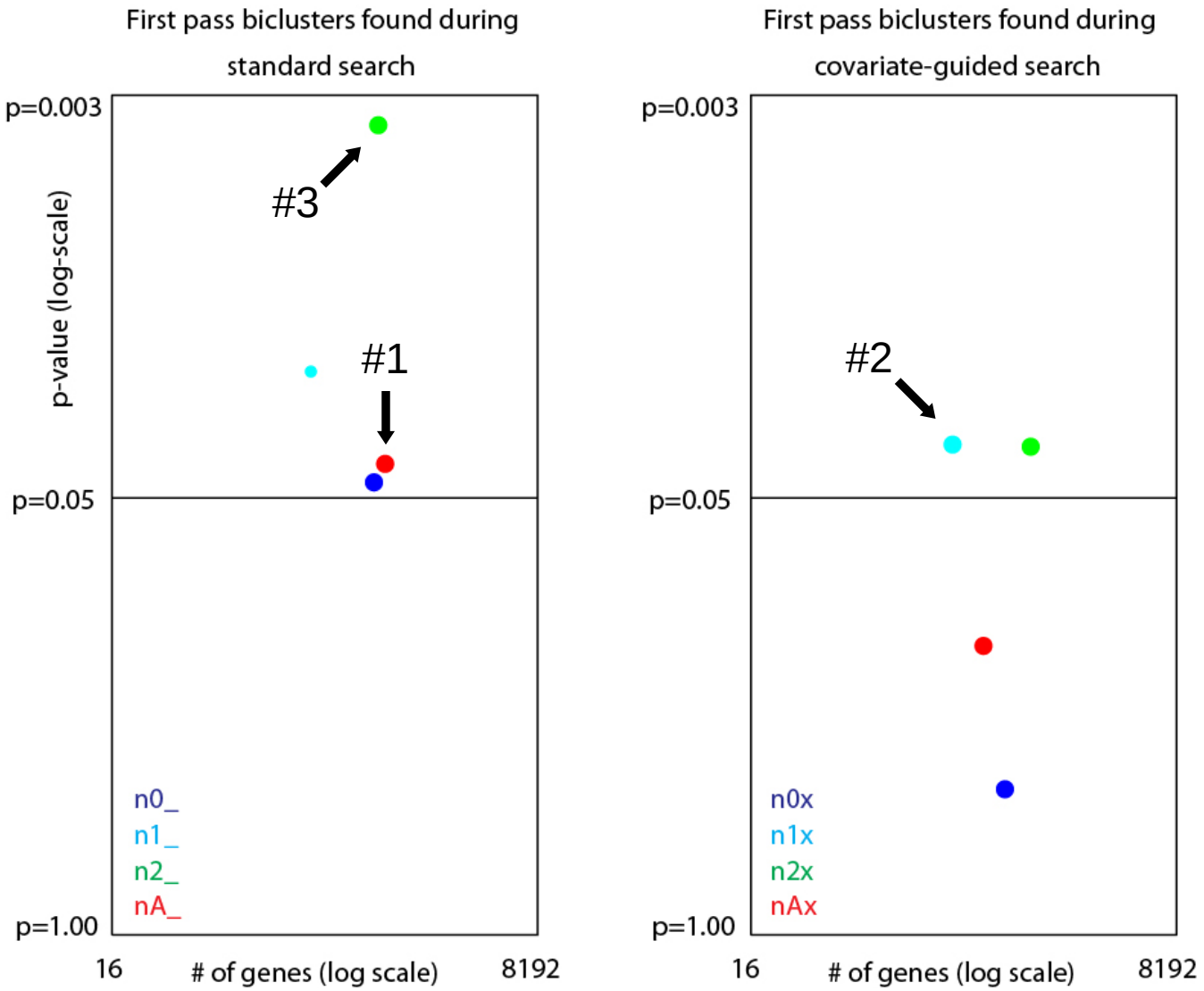
We plot each bicluster as a dot. The horizontal coordinate indicates the size of the bicluster (i.e., number of genes) and the vertical coordinate indicates the p-value that we calculated by using the label-shuffled null-hypothesis (capped at a minimum of 0.003). The p=0.05 level of significance indicated by a horizontal line; several of these biclusters are significant. The various colors correspond to the various search paradigms (see legend). The three examples we showed at the beginning are indicated via the arrowheads.



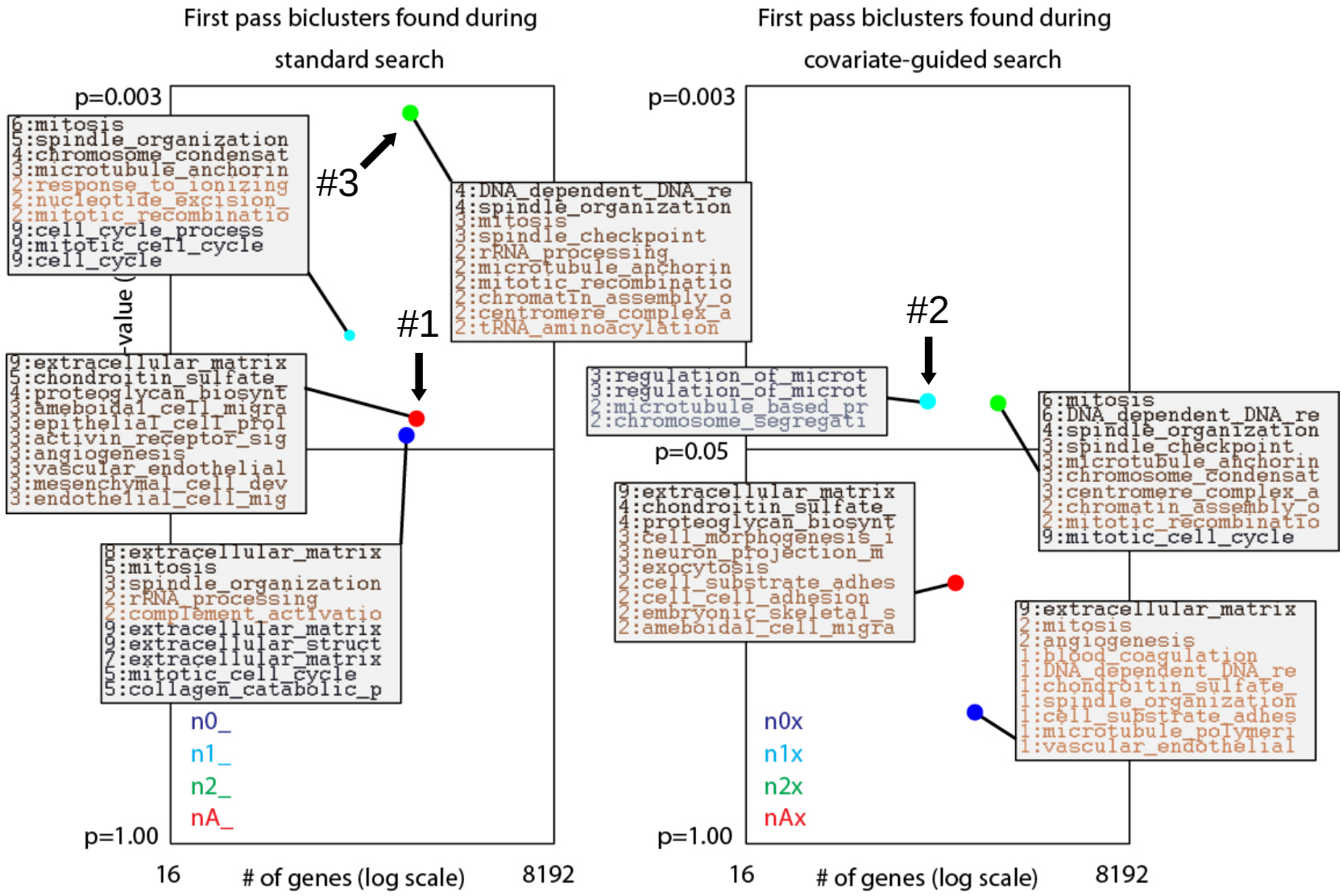
We remark that – a priori – one should not expect these biclusters to be exactly equal to one another. For example, the first bicluster found under the ‘nAx’ paradigm need not involve the same patients or genes as the first bicluster found under the ‘nA_’ paradigm. As one might guess, however, these bicluster are indeed related, as they both involve searching through the same data-set (with the same normalization-convention). We will discuss this point later on, when we illustrate how the various biclusters overlap one another.



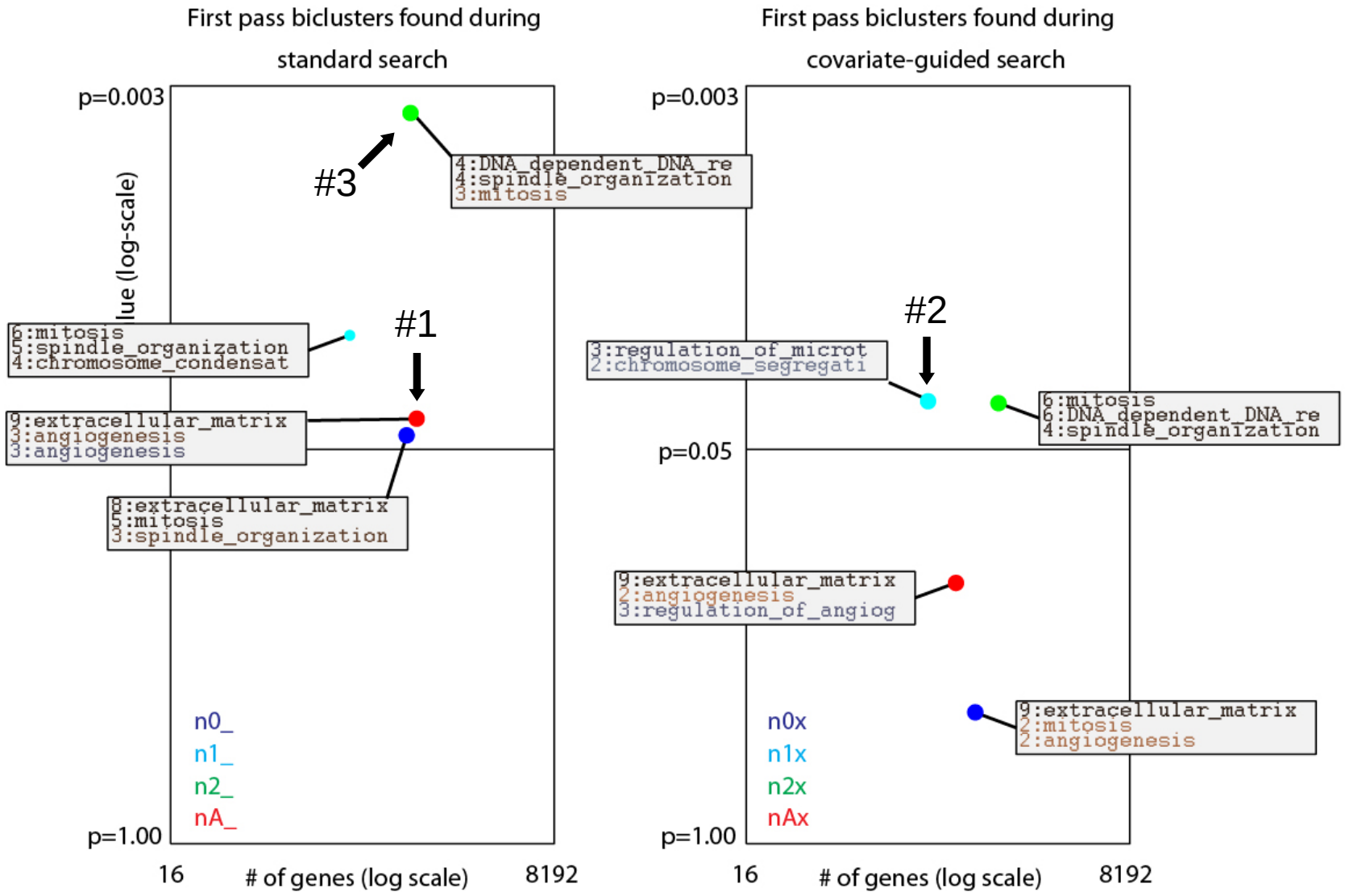
Before we continue, we make one further remark: Generally speaking, the p-values associated with a covariate-guided search are lower than the p-values associated with a standard search. The reason for this is that, as discussed earlier, the label-shuffled null-hypothesis 'H_' for the standard search is less restrictive than the label-shuffled null-hypothesis 'Hx' for the covariate-guided search. Consequently, the hypothesis Hx is more likely to generate samples that look like the original data, and thus will usually yield a larger (i.e., less significant) p-value.



Moving on; after finding biclusters within the data, we run a gene-enrichment analysis. Each bicluster (viewed as a set of genes) produces its own list of annotations, shown in a box nearby. Each line within each annotation box lists a header describing that particular pathway, prefaced by a number which is the negative-log10 of the p-value associated with that particular enrichment (calculated by seek, and capped at a maximum of 9). Blue-shaded terms have been found using the default seek database, whereas red-shaded terms use the 'bp_slim' database. For example, the line '3: mitosis' in the annotation-box of our 3rd example means that the gene-set corresponding to that bicluster is enriched for a pathway involving 'mitosis', and the p-value of that enrichment is less than 10^(-3) (these p-values have been corrected for multiple-hypothesis testing across multiple possible pathways). Observing the annotations below, one can plainly see that the gene-sets corresponding to our biclusters are far from random – and are often significantly enriched for many pathways.

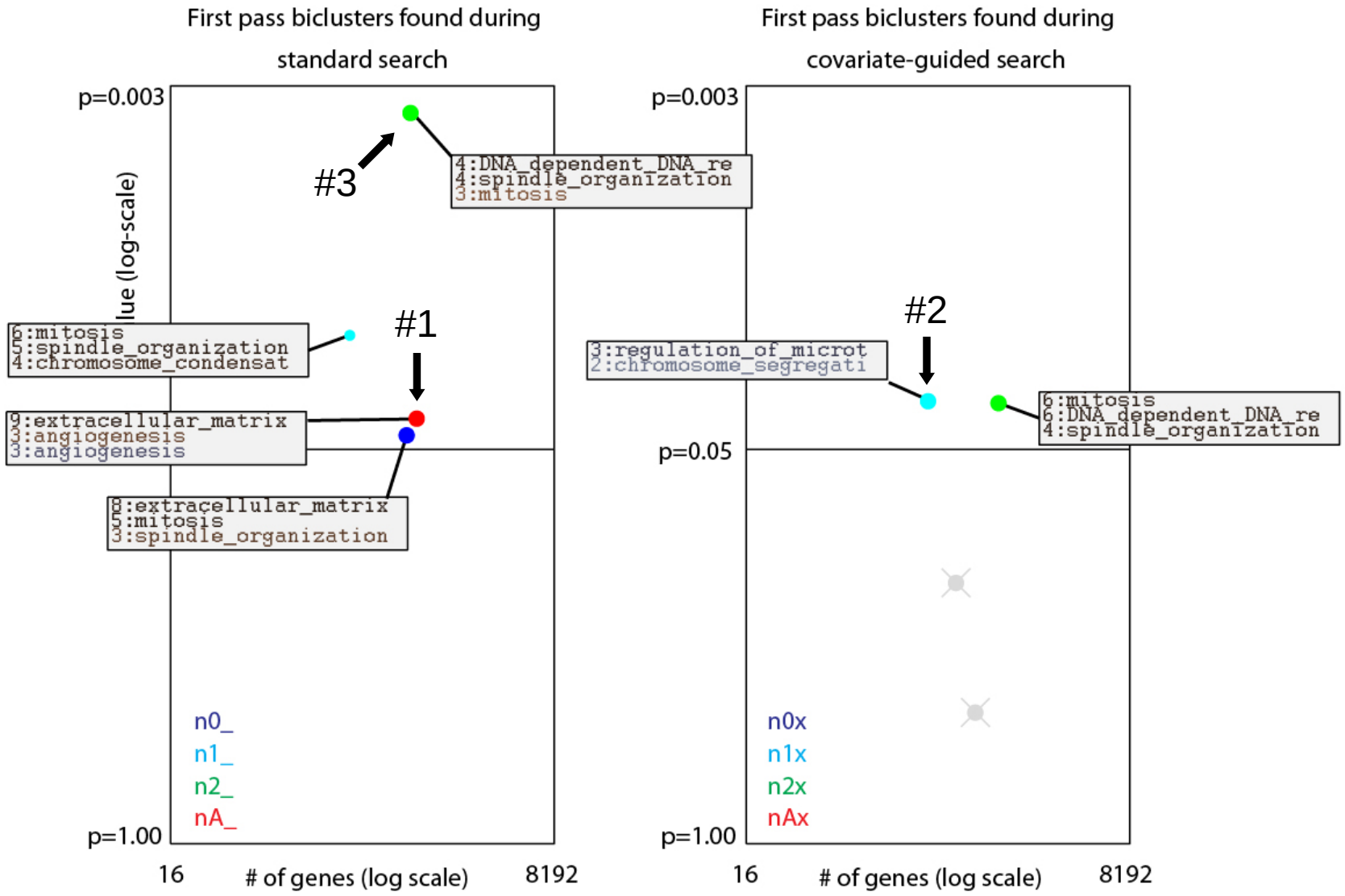


To clarify the picture somewhat we can illuminate pathways that might be most relevant for colorectal cancer. To this end we build a list of cancer-specific 'keywords': {'extracellular matrix', 'epithelial cells', 'chromosome', 'chromosomal', 'DNA', 'RNA', 'spindle', 'chromatin', 'centromere', 'mitotic', 'mitosis', 'angiogenesis'}. Shown below is the same data as before, except that we have limited our display to include only the top 3 pathways that intersect our list of keywords. As can be seen, not only do the significant biclusters display significant enrichment, but they are also significantly enriched for pathways associated with the formation and development of cancer.



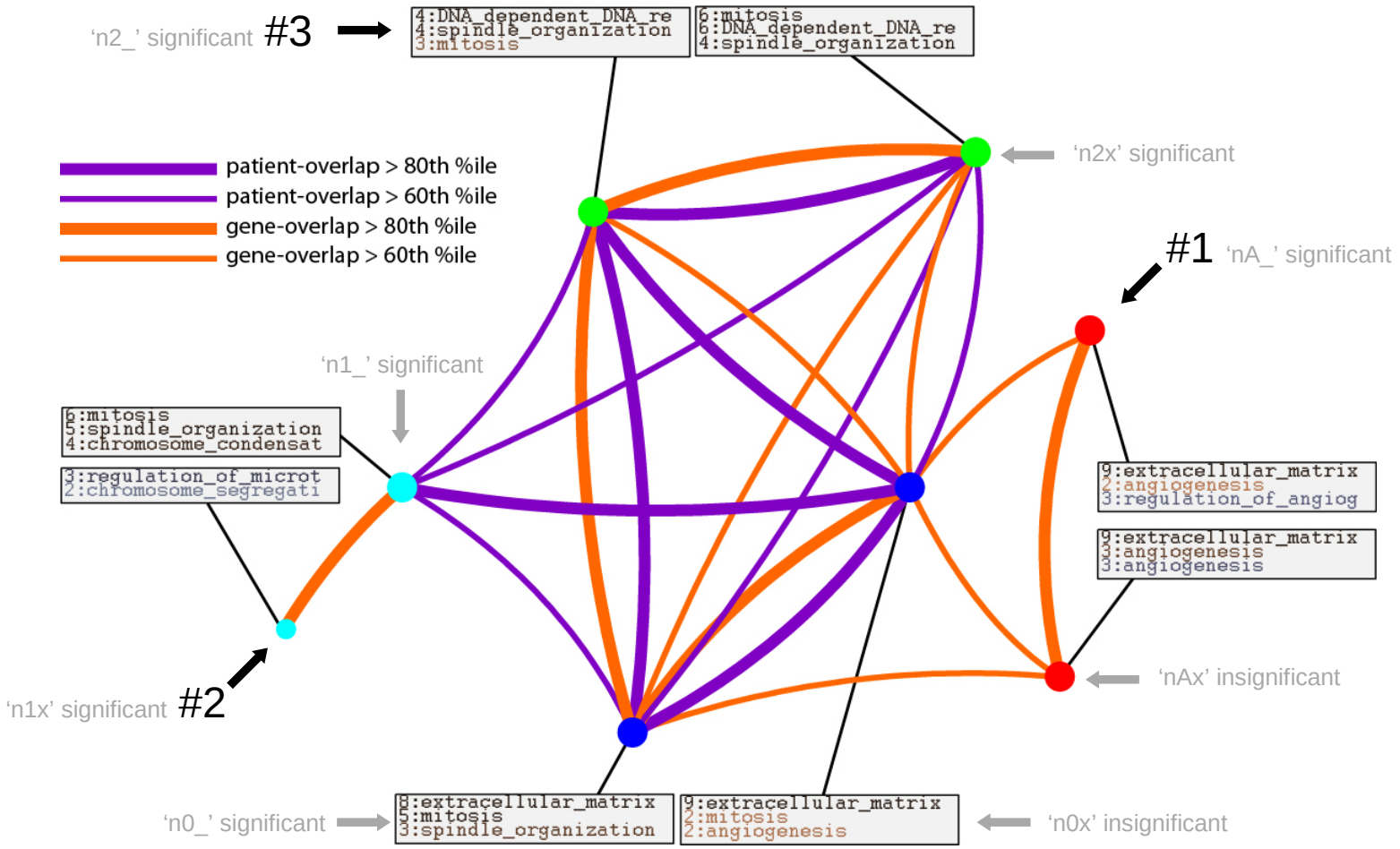
The two biclusters found under the ‘n0x’ and ‘nAx’ paradigms (crossed out in right subplot) were not very significant with respect to our label-shuffled null-hypothesis. However after further inspection we found that, while not significant by themselves, these biclusters did involve patient- and gene-subsets which overlapped quite strongly with the other biclusters found in our searches (specifically, the ‘n0_’ and ‘nA_’ biclusters). This is not uncommon, and many of the significant biclusters that we have found also overlap strongly with one another.

We discuss this issue in slightly more detail within the next slide.



To demonstrate the dependencies across the patient- and gene-subsets of our various significant biclusters, we rearrange our picture into a network, as shown below (see the code '`tutorial_net.m`' and further documentation within that file). Each dot represents one of the biclusters from the previous slide; dots are joined with a purple edge if they share many rows in common (i.e., if their patient-subsets overlap), and dots are joined with an orange edge if they share many columns (i.e., if their gene-subsets overlap). The thicker lines correspond to a greater fraction of overlap (with thresholds determined by percentiles taken from the collection of all overlap-fractions across the biclusters).

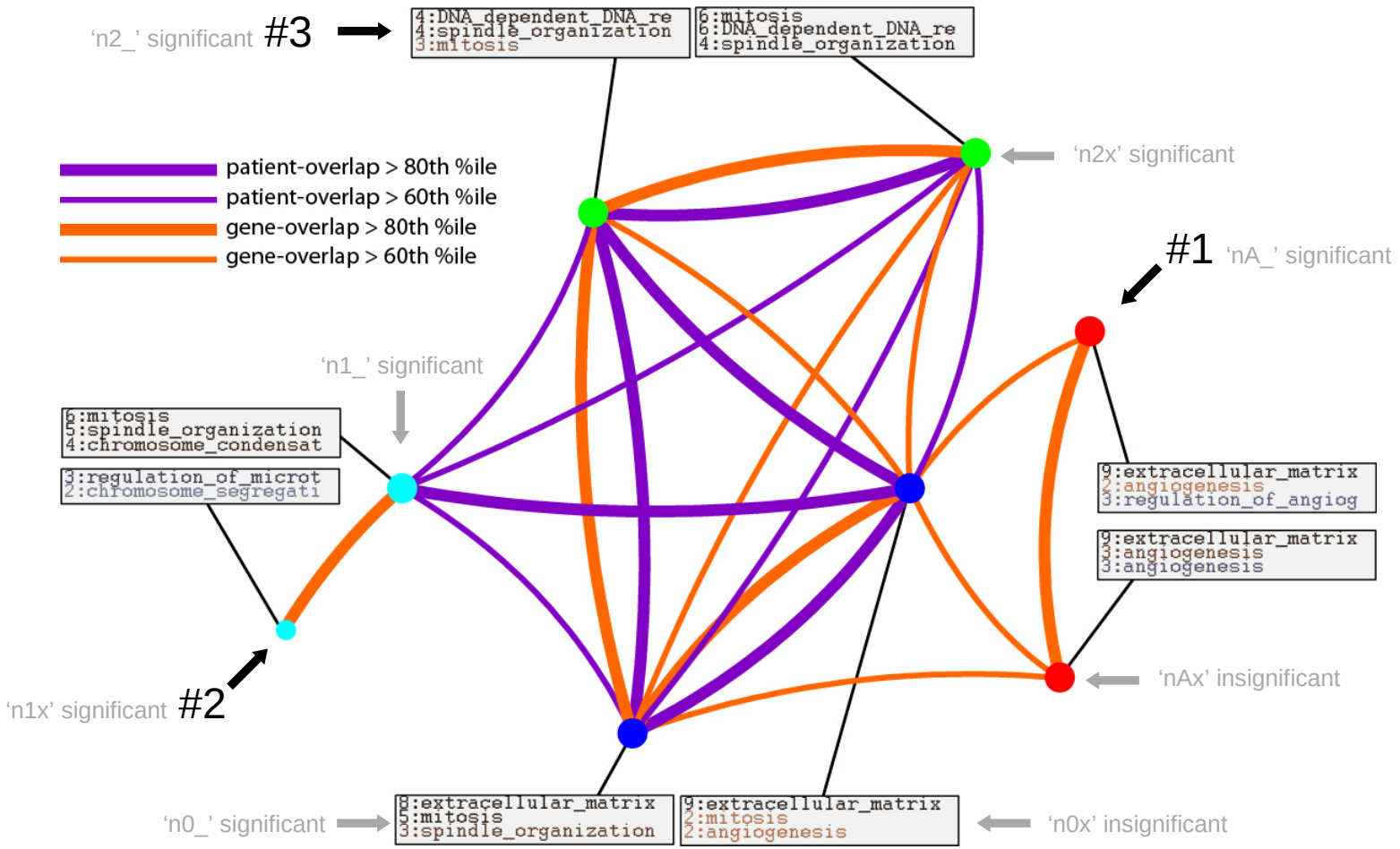
As can be seen, many of these biclusters involve similar genes, as well as similar patients. Notably, the 'n1_' and 'n1x' biclusters overlap strongly, as do the 'n2_' and 'n2x' biclusters. This is not wholly unexpected, as each of these pairs is derived from the same normalization-convention. In fact, the same holds for the 'nA_' and 'nAx' biclusters, as well as the 'n0_' and 'n0x' biclusters (the latter of each pair indicated as insignificant below).



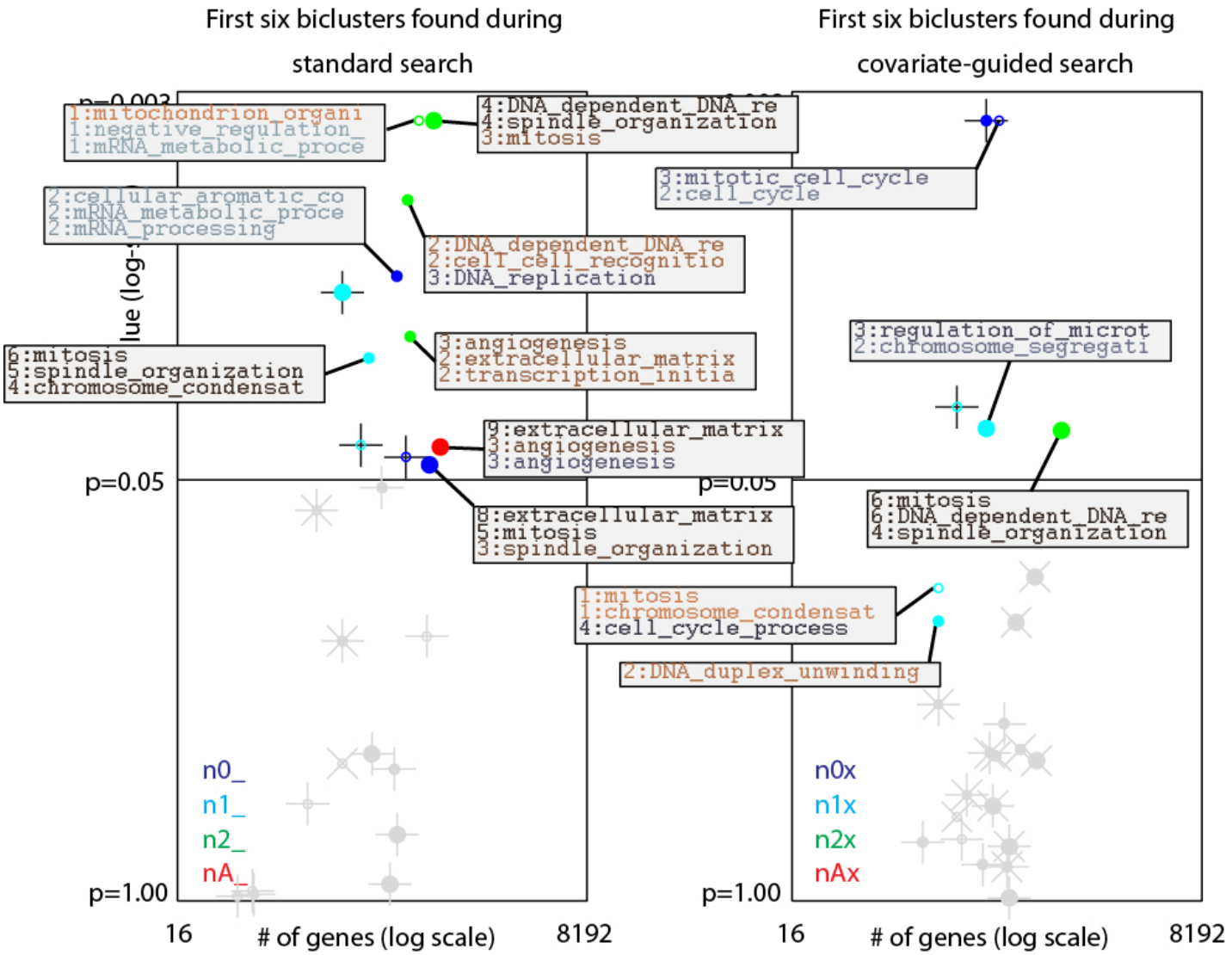
Looking a little more closely at this network, one can conclude that the ‘n2_’ and ‘n2x’ paradigms are finding largely the same signal: a collection of patients and genes which manifest under the ‘n2’ normalization-convention. The same seems to hold for other normalization-conventions (i.e., ‘n1’, ‘n0’).

This observation may allow one to draw conclusions about both biclusters derived from the same normalization-convention. For example, even though the ‘n0_’ bicluster seems significant under the H_ null-hypothesis, the lower Hx-significance level of its counterpart ‘n0x’ implies that that perhaps the ‘n0_’ bicluster should be treated with skepticism. Specifically, even though the ‘n0_’ bicluster may highlight an important structural element of the data, this structural element may not be sufficiently independent of the underlying covariates (i.e., gender and ethnicity) to be considered a true manifestation of heterogeneity due to case-status alone. Similar statements may apply to the ‘nA_’ bicluster, which had a loosely connected counterpart ‘nAx’ that was also not significant under Hx.

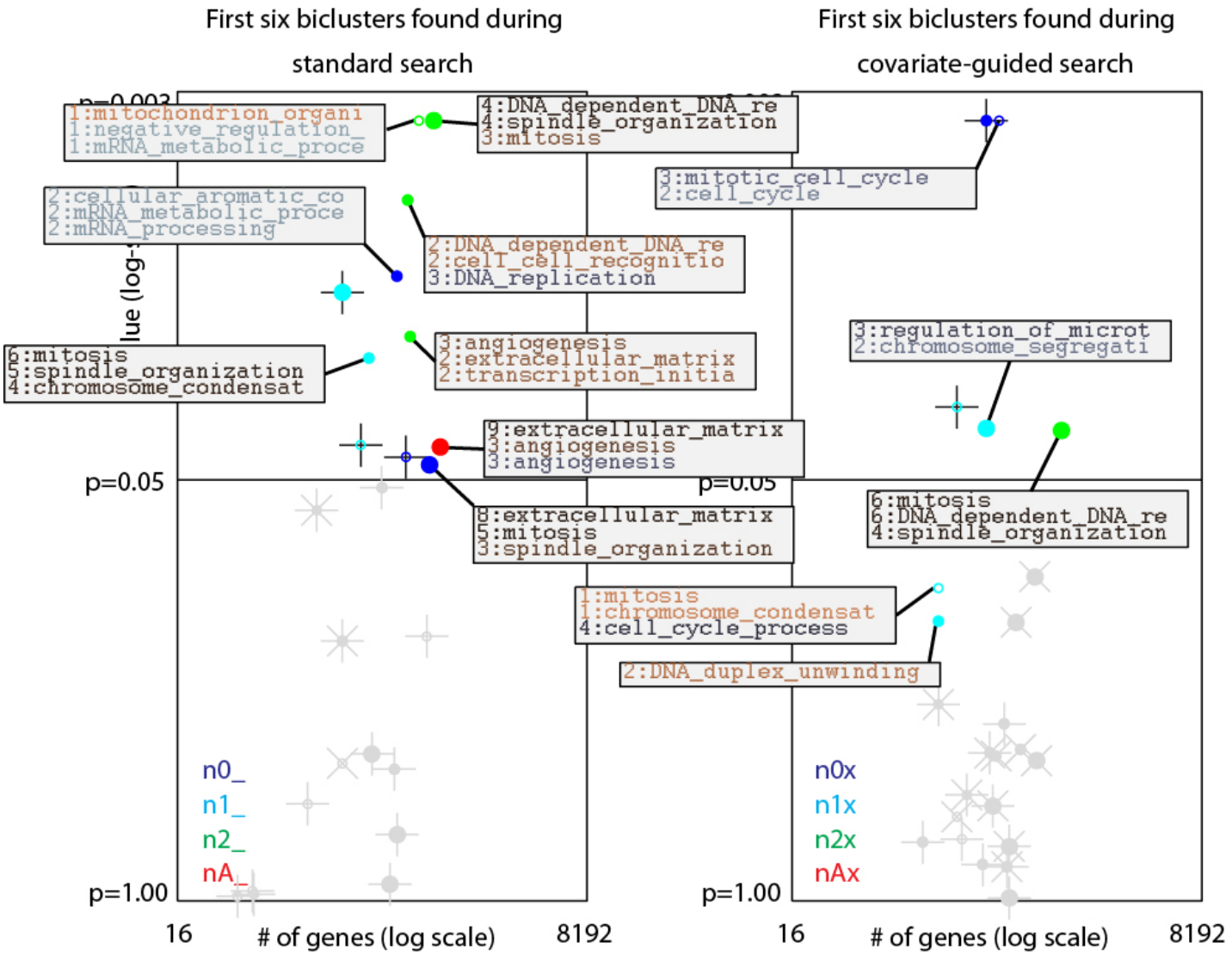
On the other hand, The figure below might serve to support the significance of the ‘n1_’ and ‘n2_’ biclusters (whose counterparts were indeed both significant). The patient- and gene-subsets highlighted by these biclusters may be worth investigating more deeply.



Our previous summary slides included only the first bicluster found via each search paradigm. In this plot we summarize the biclusters found using '`n_to_find==6`', rather than `1`. We use the same conventions as before, with the following additions: The size of each dot is determined by the overall enrichment of that bicluster; smaller (hollow) dots exhibit less significant enrichment. We also put a '+' through biclusters that do not enrich for any of our cancer-specific keywords (even large dots may be '+ed out' if they are not significantly enriched for our keyword-pathways). Finally, we put an '×' through insignificant biclusters that significantly overlap with significant biclusters (e.g., that are largely contained within biclusters that we have already found).



As can be seen below, with few exceptions, most of the biclusters that we deem significant (using our label-shuffled null-hypothesis) are in fact specifically enriched for cancer-associated pathways. Recall that our biclustering algorithms and label-shuffled null-hypothesis depend only on the data-matrix, and do not take into account any gene-enrichment information. With this in mind, we can view the correspondence below as a validation of sorts, and as a promising sign for future applications of our methodology.



We close by presenting the biclusters found after searching for structure exhibited within the control-patients instead of the case-patients (i.e., by using the 'back' flag rather than the 'frwd' flag). The control population is likely more heterogeneous than the case-population, and so many of these biclusters my not have anything to do with cancer. Consequently, we list more than merely the keyword-specific enrichment terms below. Nevertheless, it is possible that some of these biclusters represent subsets of genes which offer a 'protective effect' to subsets of control-patients; perhaps affecting their morbidity.