4 Computational and Analytical Tools for Diagnostic Measurements

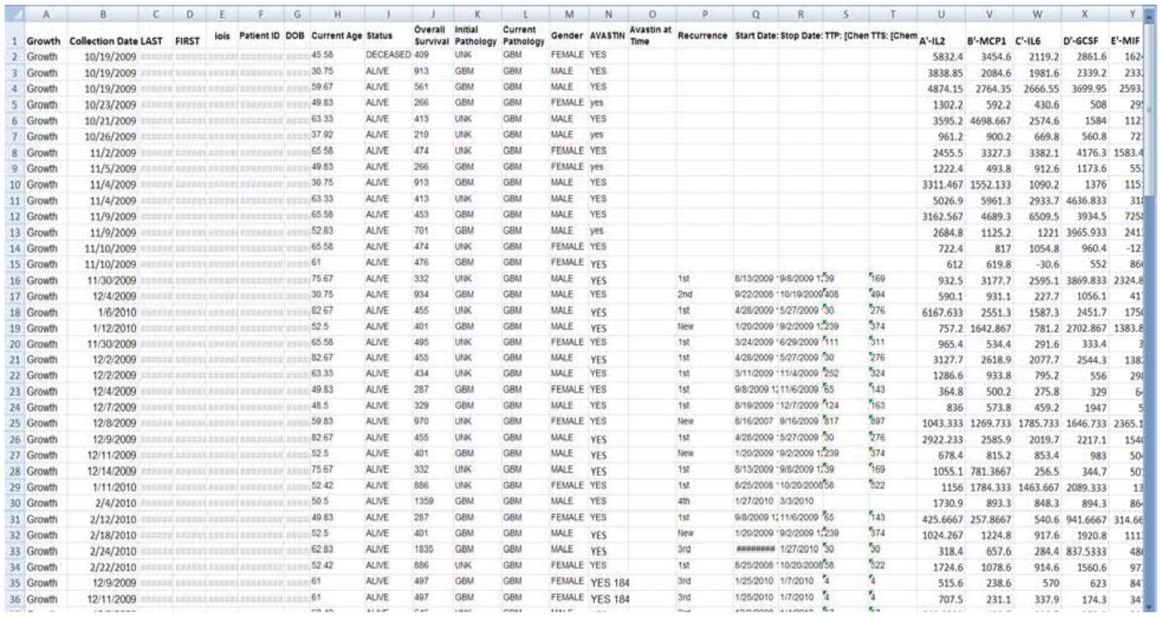
4.1 Automation of Data Processing and Analysis

Analyzing highly-multiplexed protein assays from large numbers of patients requires an efficient means of processing large datasets. Automating the computational steps from data acquisition to statistical analysis can save a considerable amount of time and effort. In fact, without automation, scaling clinical trials to assays of hundreds or thousands of proteins and patient samples would render analyses of the resulting datasets intractable. A straightforward approach for creating algorithms to manipulate data in Microsoft Excel is to write macro procedures in Visual Basic for Applications (VBA).

In our clinical trial examining patients with glioblastoma, plasma samples were assayed for 35 proteins (and a spiked reference oligo) within ELISA-like wells (12 per slide), each containing six repeating 6x6 spot arrays. These wells were fashioned by bonding a PDMS slab with 12 square holes to a DNA-spotted, polylysine-coated glass substrate. The output file from the GenePix scanner software gives the row, column, and block (or well) number of each spot based on its location in a graphical spot array template whose parameters (number of blocks, rows, and columns, as well as spot sizes and spacings) are defined by the user. Had all 12 square holes in the PDMS slab been cut with uniform dimensions and spacings, and had the PDMS slab been precisely aligned with respect to the spots on the slide, the registry of oligo spots in all wells and among all slides would be identical. In other words, the identity of a spot located within a particular row and column of a well would be the same for all wells. A list in which the row and column positions of each spot within a well (block) are matched with their corresponding identifiers could then be input into the GenePix analysis software, allowing for instant spot assignment.

However, in our study, the square holes in the PDMS slab were cut by hand, resulting in slight variability in the well dimensions and spacings. Furthermore, we did not attempt to align the PDMS slab with the spotted arrays in any way, as this would have greatly extended fabrication time and effort. As a result, the registry of spots could vary considerably across wells on the same slide and between different slides. Consequently, some means of accurately assigning an identifier to all assay spots in a well was needed. To accomplish this, we designated one of the oligos (oligo M) as a reference. To distinguish this spot from all other spots, we incubated all wells (in the final assay step) with a Cy3 (green) dye-conjugated oligo having sequence complementarity with oligo M. By contrast, all remaining assay spots fluoresced red due to development of the protein assays with Stretavidin-Cy5. Since the oligos were spotted in the same order within all arrays of the slide, the oligo identity (and its associated antibody) for any given spot could be determined by counting its row and column distance from the green reference oligo. Alternatively, an Excel macro (or VBA subroutine) could be written, as was done here, that accomplishes the spot assignment task in exactly the same way, but far more quickly.

Macros were also written to perform all subsequent data handling steps (see Appendix, **Section 4.4**). For example, once the spot positions within a well and their fluorescent intensity values were assigned to a specific oligo/antibody, the average intensity and standard deviation of all repeats were calculated for each protein. Experiments showed that at least 4 proteins in each sample assay exhibited intensities close to those in negative controls (which were performed by substituting 3% BSA/PBS for plasma samples). Therefore, a baseline intensity (intensity of a spot in the absence of cognate protein) for each patient sample could be approximated by averaging the intensities of the 4 lowest-intensity proteins within each assay. The (mean) intensities for all proteins and the baseline protein intensity level were then displayed graphically for all patients and transferred to Powerpoint automatically. Finally, the mean protein intensity values for each of the 12 patients on a slide were collated (into 12 rows) onto a single Excel worksheet for subsequent processing. This procedure was repeated in automated fashion for all patient samples on all slides. Datasets containing the baseline-subtracted intensity values and standard deviations (for all patients) were created in a similar fashion. A subroutine was written that could transfer the collated data from all open Excel workbooks (each containing its analysis of a different 12-patient slide) to a new Excel file, such that the data for all patients could be found in a single Excel worksheet. Patient ID numbers and clinical information were then manually transferred and aligned with their corresponding row of data. The final result was a master dataset in which each row – corresponding to a distinct patient sample - contained the patient characteristics and clinical information, mean protein intensities, baseline-subtracted mean protein intensities, and standard deviations. More specifically, the format of the master worksheet was as follows: Column A – Tumor Growth Status (Growth vs. No Growth); Column B – Blood Collection Date; Columns C and D – Patient Last Name and First Name, respectively; Column E – IOIS Number; Column F – Patient ID Number; Column G – Date of Birth; Column H – Current Age; Column I – Alive or Deceased; Column J – Overall Survival; Column K – Initial Pathology; Column L – Current Pathology; Column M – Gender; Column N – Chemotherapy Drug (i.e. Avastin vs. No Avastin); Column O – “Was patient on Avastin at the time of the blood collection date in Column B?"; Column P – Tumor Recurrence Number; Column Q - Chemotherapy Start Date; Column R – Chemotherapy End Date; Columns U through BD – Mean Fluorescent Intensities (Baseline Subtracted) for Proteins 1 through 35 (plus M'-Cy3 reference). Columns BF through CO – Standard Deviations for Proteins 1 through 35 (plus M'-Cy3 reference); Column CQ – Time of Blood Sample Collection; Column CR – Time at which Plasma Sample was Frozen; Column CS – Total Processing Time; Columns CU through ED – Proteins 1 through 35 (plus M'-Cy3 reference) Mean Fluorescent Intensities (Non-Baseline Subtracted). In summary, all the data and relevant clinical information for every single patient in the study was included in the master worksheet (See **Figure 4.1** below).



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1	Growth	Collection Date	LAST	FIRST	iois	Patient ID	DOB	Current Age	Status	Overall Survival	Initial Pathology	Current Pathology	Gender	AVASTIN	Avastin at Time	Recurrence	Start Date:	Stop Date:	TTP: [Chen	TTS: [Chem	A'-IL2	B'-MCP1	C'-IL6	D'-GCSF	E'-MIF
2	Growth	10/19/2009	######	######	######	########	####	45.58	DECEASED	409	UNK	GBM	FEMALE	YES							5832.4	3454.6	2119.2	2861.6	162
3	Growth	10/19/2009	######	######	######	########	####	30.75	ALIVE	913	GBM	GBM	MALE	YES							3838.85	2084.6	1981.6	2339.2	233
4	Growth	10/19/2009	######	######	######	########	####	59.67	ALIVE	561	GBM	GBM	MALE	YES							4874.15	2764.35	2666.55	3699.95	2593.
5	Growth	10/23/2009	######	######	######	########	####	49.83	ALIVE	266	GBM	GBM	FEMALE	yes							1302.2	592.2	430.6	508	29
6	Growth	10/21/2009	######	######	######	########	####	63.33	ALIVE	413	UNK	GBM	MALE	YES							3595.2	4698.667	2574.6	1584	112
7	Growth	10/26/2009	######	######	######	########	####	37.92	ALIVE	210	UNK	GBM	MALE	yes							961.2	900.2	669.8	560.8	72
8	Growth	11/2/2009	######	######	######	########	####	65.58	ALIVE	474	UNK	GBM	FEMALE	YES							2455.5	3327.3	3382.1	4176.3	1583.4
9	Growth	11/5/2009	######	######	######	########	####	49.83	ALIVE	266	GBM	GBM	FEMALE	yes							1222.4	493.8	912.6	1173.6	55
10	Growth	11/4/2009	######	######	######	########	####	30.75	ALIVE	913	GBM	GBM	MALE	YES							3311.467	1552.133	1090.2	1376	115
11	Growth	11/4/2009	######	######	######	########	####	63.33	ALIVE	413	UNK	GBM	MALE	YES							5026.9	5961.3	2933.7	4636.833	31
12	Growth	11/9/2009	######	######	######	########	####	65.58	ALIVE	453	GBM	GBM	MALE	YES							3162.567	4689.3	6509.5	3934.5	725
13	Growth	11/9/2009	######	######	######	########	####	52.83	ALIVE	701	GBM	GBM	MALE	yes							2684.8	1125.2	1221	3965.933	241.
14	Growth	11/10/2009	######	######	######	########	####	65.58	ALIVE	474	UNK	GBM	FEMALE	YES							722.4	817	1054.8	960.4	-12
15	Growth	11/10/2009	######	######	######	########	####	61	ALIVE	476	GBM	GBM	FEMALE	YES							612	619.8	-30.6	552	86
16	Growth	11/30/2009	######	######	######	########	####	75.67	ALIVE	332	UNK	GBM	MALE	YES		1st	8/13/2009	9/8/2009 1	39	169	932.5	3177.7	2595.1	3869.833	2324.8
17	Growth	12/4/2009	######	######	######	########	####	30.75	ALIVE	934	GBM	GBM	MALE	YES		2nd	9/22/2008	10/19/2009	408	494	590.1	931.1	227.7	1056.1	41
18	Growth	1/6/2010	######	######	######	########	####	82.67	ALIVE	455	UNK	GBM	MALE	YES		1st	4/28/2009	5/27/2009	30	276	6167.633	2551.3	1587.3	2451.7	175
19	Growth	1/12/2010	######	######	######	########	####	52.5	ALIVE	401	GBM	GBM	MALE	YES		New	1/20/2009	9/2/2009 1	239	374	757.2	1642.867	781.2	2702.867	1383.8
20	Growth	11/30/2009	######	######	######	########	####	65.58	ALIVE	495	UNK	GBM	FEMALE	YES		1st	3/24/2009	6/29/2009	111	311	965.4	534.4	291.6	333.4	3
21	Growth	12/2/2009	######	######	######	########	####	82.67	ALIVE	455	UNK	GBM	MALE	YES		1st	4/28/2009	5/27/2009	30	276	3127.7	2618.9	2077.7	2544.3	138.
22	Growth	12/2/2009	######	######	######	########	####	63.33	ALIVE	434	UNK	GBM	MALE	YES		1st	3/11/2009	11/4/2009	252	324	1286.6	933.8	795.2	556	29
23	Growth	12/4/2009	######	######	######	########	####	49.83	ALIVE	287	GBM	GBM	FEMALE	YES		1st	9/8/2009 1:	11/6/2009	65	143	364.8	500.2	275.8	329	6
24	Growth	12/7/2009	######	######	######	########	####	48.5	ALIVE	329	GBM	GBM	MALE	YES		1st	8/19/2009	12/7/2009	124	163	836	573.8	459.2	1947	5
25	Growth	12/8/2009	######	######	######	########	####	59.83	ALIVE	970	UNK	GBM	FEMALE	YES		New	8/16/2007	9/16/2009	817	897	1043.333	1269.733	1785.733	1646.733	2365.1
26	Growth	12/9/2009	######	######	######	########	####	82.67	ALIVE	455	UNK	GBM	MALE	YES		1st	4/28/2009	5/27/2009	30	276	2922.233	2585.9	2019.7	2217.1	154
27	Growth	12/11/2009	######	######	######	########	####	52.5	ALIVE	401	GBM	GBM	MALE	YES		New	1/20/2009	9/2/2009 1	239	374	678.4	815.2	853.4	983	50
28	Growth	12/14/2009	######	######	######	########	####	75.67	ALIVE	332	UNK	GBM	MALE	YES		1st	8/13/2009	9/8/2009 1	39	169	1055.1	781.3667	256.5	344.7	50
29	Growth	1/11/2010	######	######	######	########	####	52.42	ALIVE	886	UNK	GBM	FEMALE	YES		1st	8/25/2008	10/20/2008	58	522	1156	1784.333	1463.667	2089.333	13
30	Growth	2/4/2010	######	######	######	########	####	50.5	ALIVE	1359	GBM	GBM	MALE	YES		4th	1/27/2010	3/3/2010			1730.9	893.3	848.3	894.3	86
31	Growth	2/12/2010	######	######	######	########	####	49.83	ALIVE	287	GBM	GBM	FEMALE	YES		1st	9/8/2009 1:	11/6/2009	65	143	425.6667	257.8667	540.6	941.6667	314.66
32	Growth	2/18/2010	######	######	######	########	####	52.5	ALIVE	401	GBM	GBM	MALE	YES		New	1/20/2009	9/2/2009 1	239	374	1024.267	1224.8	917.6	1920.8	111
33	Growth	2/24/2010	######	######	######	########	####	62.83	ALIVE	1835	GBM	GBM	MALE	YES		3rd	########	1/27/2010	30	30	318.4	657.6	284.4	837.5333	48
34	Growth	2/22/2010	######	######	######	########	####	52.42	ALIVE	886	UNK	GBM	FEMALE	YES		1st	8/25/2008	10/20/2008	58	522	1724.6	1078.6	914.6	1560.6	97.
35	Growth	12/9/2009	######	######	######	########	####	61	ALIVE	497	GBM	GBM	FEMALE	YES	184	3rd	1/25/2010	1/7/2010	4	4	515.6	238.6	570	623	84
36	Growth	12/11/2009	######	######	######	########	####	61	ALIVE	497	GBM	GBM	FEMALE	YES	184	3rd	1/25/2010	1/7/2010	4	4	707.5	231.1	337.9	174.3	34

Figure 4.1 Master patient dataset: organization of clinical information. Only a portion of the full dataset is shown. (Patient identifiers have been removed).

Macros were also written to automate graphing of the patient data within the master worksheet. One of these macros graphs the protein data in each row (corresponding to a unique patient sample) in a separate graph, all of which can then be automatically transferred to a

Powerpoint file. Other macros can display the protein data from all of a patient's blood collections in a single graph (once the file has been sorted first by patient name and then by collection date), such that changes in protein levels within the patient's plasma can be traced over time. These macros then repeat the process for all patients in the worksheet.

From the master worksheet, patient cohorts can straightforwardly be created by reorganizing, sorting, and trimming the data with regard to any one of the parameters in the clinical information columns. For example, one could sort the dataset based on current clinical pathology (Column L) to extract a cohort of GBM patients vs. healthy controls. To create a cohort in which tumor growth status is compared among Avastin-treated GBM patients, the dataset is sorted first by Column L (GBM vs. No GBM), then by Column N (Avastin vs. No Avastin), and finally by Column A (Tumor Growth vs. No Growth). Patients who do not have GBM and are not on Avastin are subsequently removed from the set.

Once these cohorts are created, a series of subroutines are required to facilitate statistical and graphical analysis, hierarchical clustering of the data, and the utilization of these hierarchical clusters for patient classification. The "RunClusterPrep" macro accomplishes these tasks as follows. First, the patient data worksheet is reorganized and formatted appropriately for compatibility with the clustering software, *Cluster 3.0*. Second, experimental and control group mean and median fluorescent intensities are calculated for each protein assayed (as well as the differences and root-mean-square distances between experimental and control group means and medians). These values are then displayed graphically. Next, an additional file is created in which the experimental and control data (for each protein) are formatted for facile transfer to and analysis by "AnalyseIt", a statistical software add-in for Excel (For details and additional related macros, see **Section 4.4.5**). The user can then run a number of different statistical tests on the transferred data (now residing in tabulated form within an AnalyseIt template file). In our clinical trial, we most commonly utilized the Student's t-test (sensitive to differences in population means) and Mann Whitney test (sensitive to differences in population medians) to assess the statistical significance (*p*-value) of differential protein expression between experimental and control groups. We also utilized AnalyseIt's box plot function to be able to visually compare (for each protein) the experimental and control population means, standard deviations, and 95% confidence intervals, as well as medians, quartiles, outliers, and general spread of the data.

In addition, the "RunClusterPrep" subroutine facilitates diagnostic testing in the following way. The subroutine randomly assigns a certain number of patients (number specified by the user) within a cohort dataset as "unknown" test samples. The resulting test file, containing both "known" and "unknown" patient samples, is converted to text format, such that it can then be clustered (by Average-Linkage Hierarchical Clustering) using *Cluster 3.0*. The cluster map (or heat map) can subsequently be viewed using Java *TreeView*. In a classification scheme that can most appropriately be described as "guilt-by-association", the unknown patients are classified by the tester as belonging to the experimental or control group based on the majority diagnosis of neighbors within their cluster. The macro "CalculateStatistics" (**Section 4.4.4**) then compares the predicted and actual diagnoses, determines the true positives/negatives and false positives/negatives, and creates a 2x2 contingency table for these values. Measures of diagnostic accuracy, such as the sensitivity, specificity, positive predictive value (PPV), and negative predictive value (NPV), are then calculated by the macro. The RunClusterPrep subroutine creates multiple test files (number specified by the user), each with its own set of randomly assigned unknown samples. Thus, the "guilt-by-association" classification procedure can be repeated multiple times, allowing the diagnostic accuracy of the procedure to be assessed with greater statistical power.

As mentioned before, the number of test files to be created and the number of unknowns to be assigned within each test file are specified by the user. In addition, the user must specify the number of proteins being examined. To facilitate entry of these parameters by the user, a customized user interface has been created. This interface also allows the user to specify the directory into which the new folder, "NewTrialFolder" – containing the files to be created by the "RunClusterPrep" macro - should be saved. The combination of the "RunClusterPrep" macro (with its associated subroutines) and the user interface form a software package we call "ClusterPrep". To initiate or "open" the program, we have created a command button for the Excel Add-Ins Toolbar labeled "RunAnalysis" (see below).

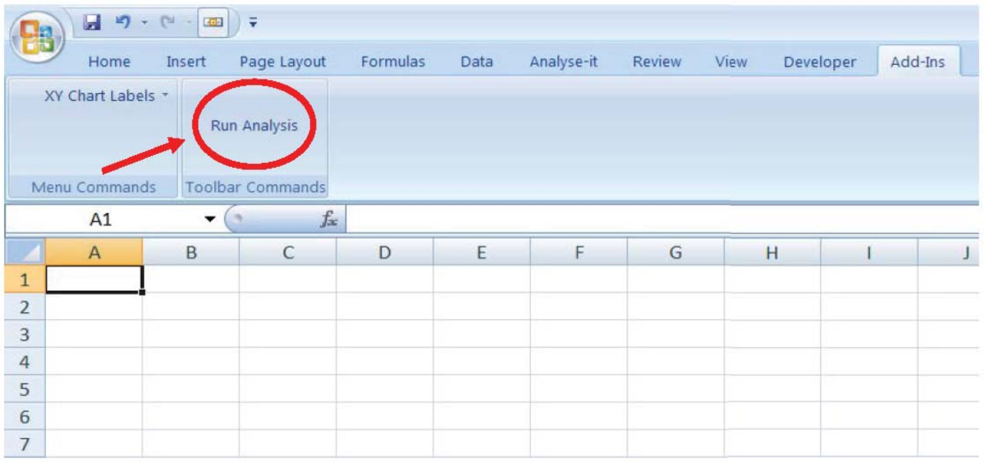


Figure 4.2 The "Run Analysis" command button in the Excel add-ins toolbar.
When this command button is clicked, the "ClusterPrep" software program is initiated.

When this button is clicked, the user interface is first displayed (**Figure 4.3**). Once the user inputs the required parameters and clicks "Okay", the "RunClusterPrep" macro and its associated subroutines are executed. The output files and folders are typically created within about a

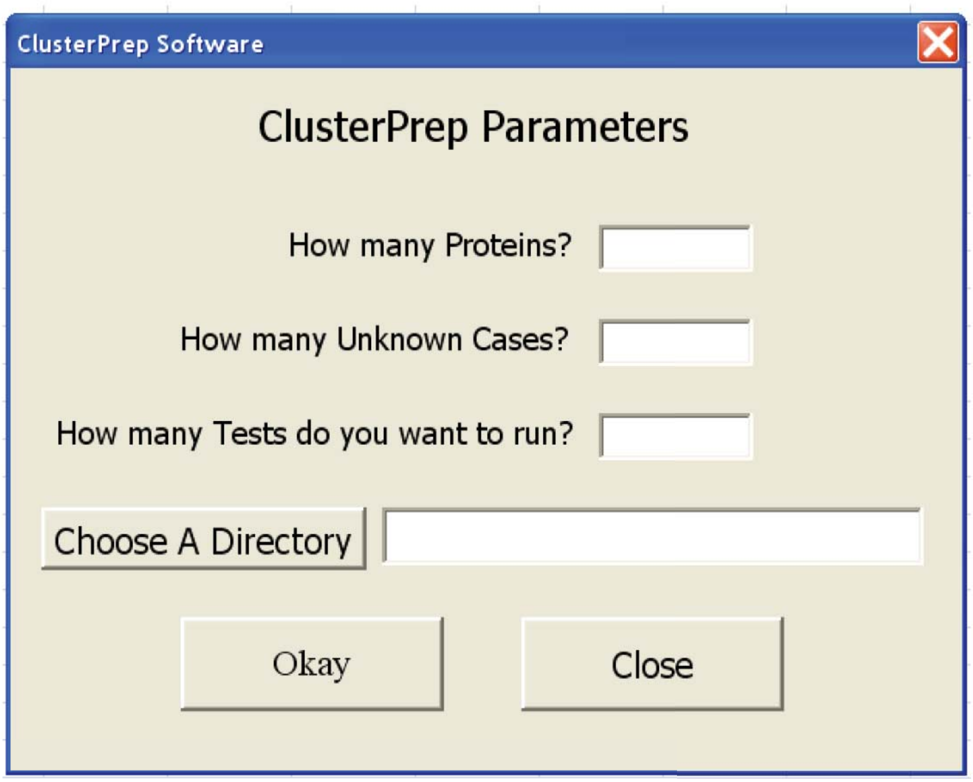


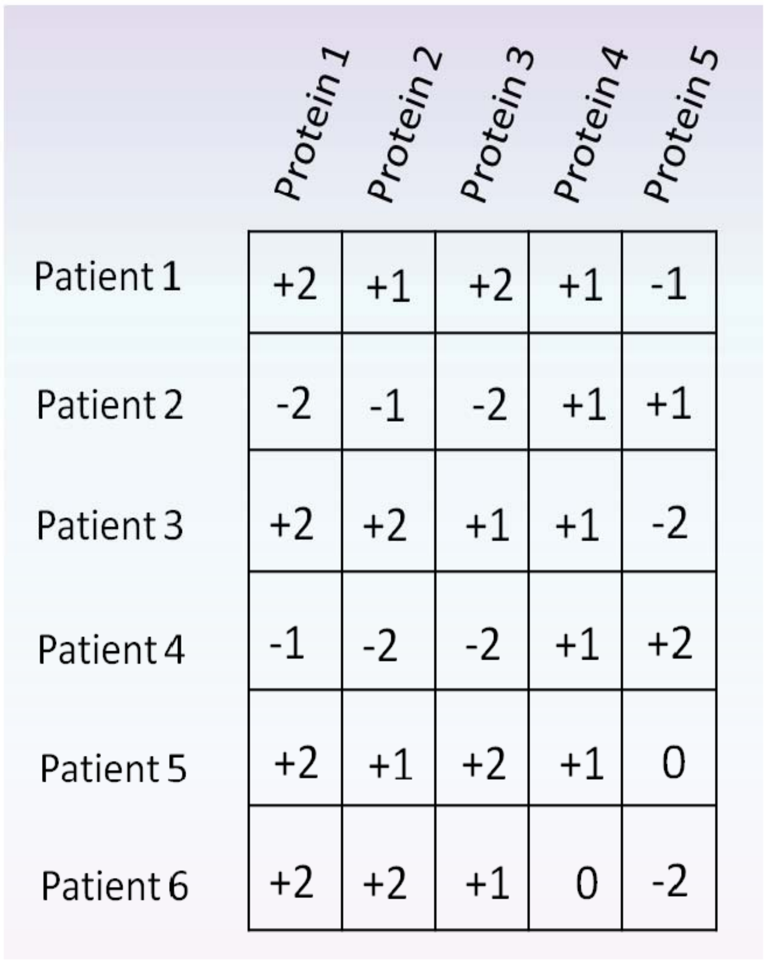
Figure 4.3 The "ClusterPrep" user interface. The user inputs the number of proteins being analyzed, the number of samples to randomly set aside as test samples, and the number of tests to run. The user also designates the directory into which the output files will be saved.

minute; however, much longer times are needed if the number of test files and unknowns specified by the user is great. For our data analysis, we typically chose to have "ClusterPrep" create 10 test files with 10 unknowns in each file.

While the "ClusterPrep Software" package greatly increases the efficiency of statistical analysis and of creating files for cluster analysis, transferring these files into *Cluster 3.0* manually is still a time-consuming task. Therefore, we have created a batch file that executes the cluster analysis on each test file in the "NewTrialFolder" directory directly from the command line. The batch file can be edited to produce multiple Cluster output files (.cdt) for each test file, each with a different distance/similarity measure, normalization, and clustering method. For this clinical trial, we used the Average-Linkage Hierarchical Clustering method with the Pearson correlation as the distance measure. The parameters that were adjusted included indicating whether normalization would be performed on the proteins only, samples only, or both, and whether the Pearson Correlation would be centered or uncentered (See *Cluster 3.0* Manual for more information). A different .cdt file could be created for each of these permutations. Typically, we chose to normalize across both proteins and samples. This means that for both variables, all values in each row (or column) are multiplied by a scalar such that the sum of the squares of the values in each row (or column) is 1 (a separate scalar is computed for each row). The batch file had to be placed in the folder containing the test files (saved as text) created by the "ClusterPrep Software", where it could be executed by double-clicking on its icon. An additional batch file was created that could then open each .cdt file in Java *TreeView*, adjust the contrast of the heat map, and save the heat map as a .png file within the same directory. Finally, a macro was created for Microsoft Powerpoint that would transfer and center each .png file in the given directory onto a separate slide in a Powerpoint Presentation.

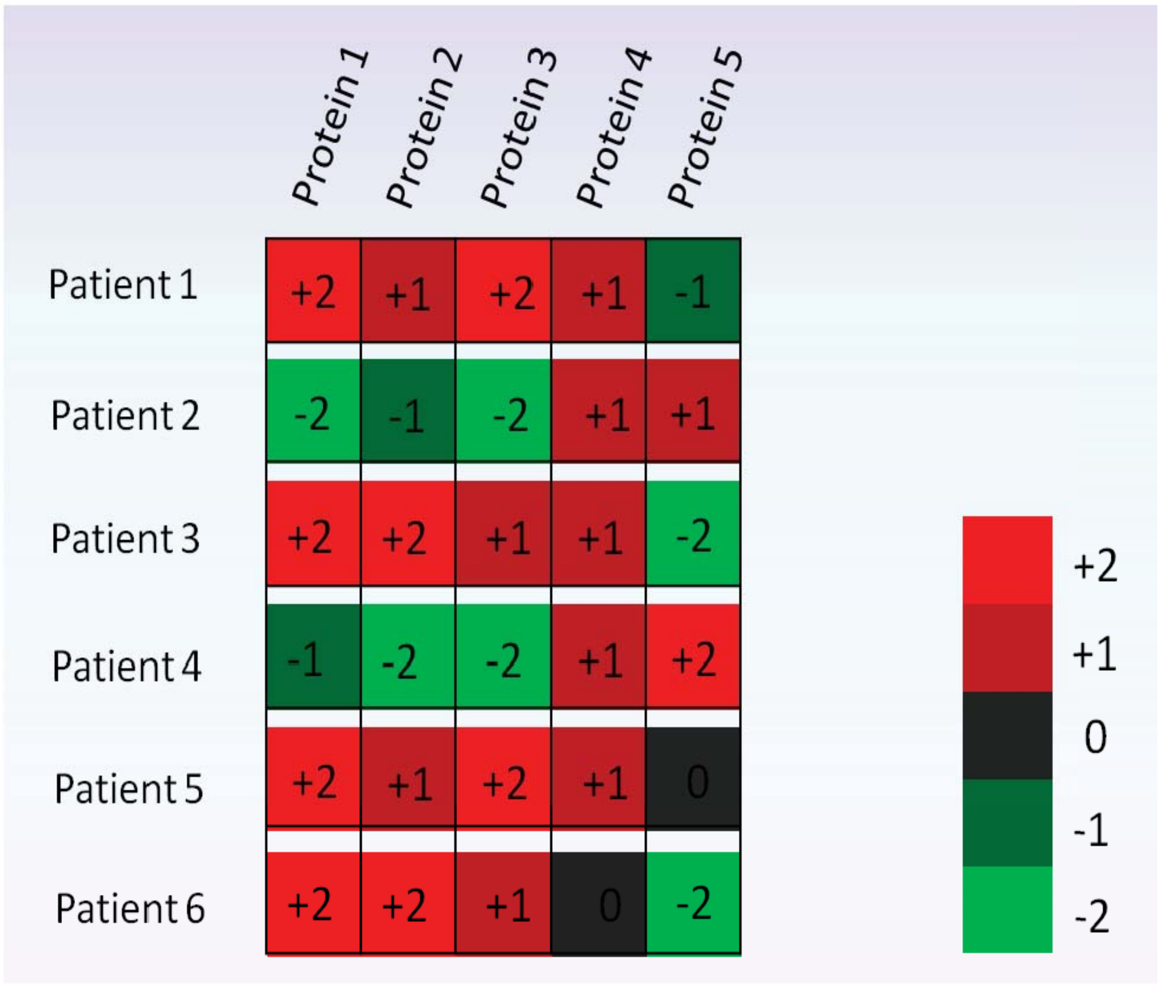
4.2 Average-Linkage Hierarchical Clustering

The master dataset contains all protein intensities for all patient samples. The clustering algorithm groups the samples based on the similarities between their component protein intensities. To illustrate how this is done, let's say we were studying the plasma levels of 5 different proteins in 6 different patients, and we obtained the following intensity scores (where -2 is the lowest intensity and +2 is the highest):



	Protein 1	Protein 2	Protein 3	Protein 4	Protein 5
Patient 1	+2	+1	+2	+1	-1
Patient 2	-2	-1	-2	+1	+1
Patient 3	+2	+2	+1	+1	-2
Patient 4	-1	-2	-2	+1	+2
Patient 5	+2	+1	+2	+1	0
Patient 6	+2	+2	+1	0	-2

To better visualize this table of values, we can convert these intensity scores to colors. For example, higher intensity scores can be assigned as brighter red, lower intensity scores as brighter green, and middle intensities as black. This would lead to the following heat map:



	Protein 1	Protein 2	Protein 3	Protein 4	Protein 5
Patient 1	+2	+1	+2	+1	-1
Patient 2	-2	-1	-2	+1	+1
Patient 3	+2	+2	+1	+1	-2
Patient 4	-1	-2	-2	+1	+2
Patient 5	+2	+1	+2	+1	0
Patient 6	+2	+2	+1	0	-2

By casually glancing at the color-coded rows, one can begin to group these patients according to similarities between their protein profiles. For example, Patient 1's protein profile looks most similar to that of Patient 5 (alternating bright and dark red for Proteins 1-4 followed by a lower intensity in Protein 5). Therefore, these two patients can be grouped together by branches intersecting at a node (**Figure 4.4**). Similarly, Patient 3's profile is almost identical to Patient 6's profile (except for Protein 4), so these two patients can be coupled. Likewise, Patients 2 and 4 can be grouped together. Among these 3 pairs of patients, the first and second pairs most closely resemble each other in that they generally exhibit higher intensities for Proteins 1-4, and lower intensities for Protein 5. Therefore, these two pairs can be linked into a single cluster. Finally, this cluster is linked with the third pair, which is more distantly related as it has low intensities for Proteins 1-3 followed by higher intensities for Proteins 4 and 5. It is noteworthy that the lengths of the branches are set to the distance between the joined items. Therefore, more highly correlated patient samples are joined by shorter branches, whereas more distantly related patient samples are joined by longer branches (see **Figure 4.4** below).

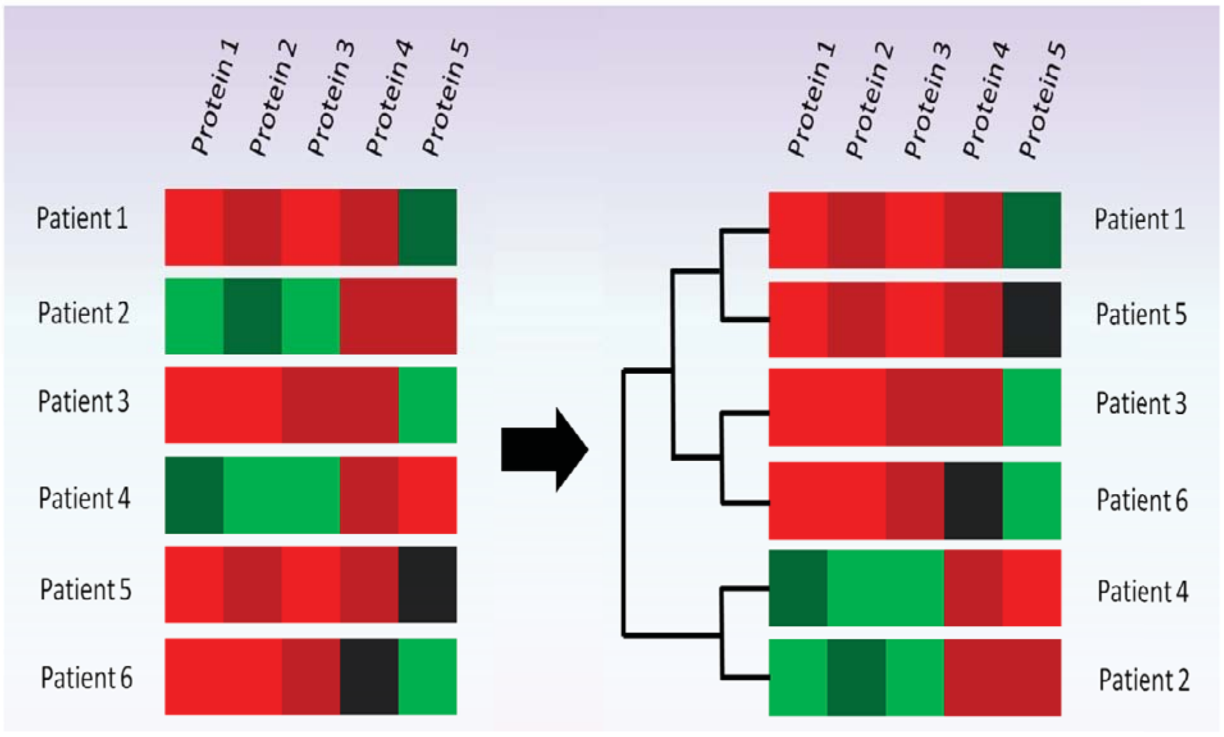


Figure 4.4 Illustration of clustering by visually grouping patient samples based on protein profile similarities.

While clustering of patient samples might be accomplished visually for small sample sizes and few assayed proteins, much larger datasets – like our 120 samples x 35 protein set - requires the clustering analysis to be done computationally. As such, the correlation between patients' protein profiles must be determined mathematically. This is most commonly done using the Pearson correlation between the protein profiles, though other distance measures (Euclidean, city-block, and non-parametric measures) can also be used. The Pearson correlation r_{xy}, between the protein profiles of two patient samples (X and Y) is given by:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{(n-1)s_x s_y}$$

where x_i and y_i are the fluorescent intensities of the *i*th protein, $\bar{x}$ and $\bar{y}$ are the mean protein intensities, and s_x and s_y are the corresponding standard deviations, of samples X and Y, respectively. These correlation coefficients are calculated and the resulting clustering of the data is accomplished using *Cluster 3.0*. In addition, for this study, the clustering algorithm was set to "Average-Linkage Clustering", in which the distance between the two patient samples, X and Y, is the mean of all pairwise distances between their component protein intensities.

4.3 Test Sample Classification: "Guilt by Association"

As mentioned previously, test samples were classified based on the majority diagnosis of their nearest neighbors. To illustrate how this "guilt-by-association" technique works, let's look at a few examples of test samples ("unknowns") within clusters containing varying ratios of experimental (magenta) and control group (green) samples. In **Figure 4.5a**, we have a cluster containing two unknowns, two GBM patients, and two healthy controls. Since this cluster is evenly split between experimental and control samples, no determination can be made about the classification of the two unknowns. As can be seen, in unbiased clusters such as these, the test sample classification is indeterminate. In this study, test samples with indeterminate classifications were excluded from further analysis. In **Figure 4.5b**, the unknown resides within a cluster in which there are 2 samples from patients with tumor growth and 3 samples from patients with no tumor growth (since their last MRI scan). Because this cluster has a slight "No Growth" bias, the unknown is classified as having no tumor growth (control group). However, the confidence level in this assignment is not very high since the number of "No Growth" samples barely exceeds the number of "Growth" samples. In **Figure 4.5c**, the unknown is situated within a cluster in which there are 4 "No Tumor Growth" samples and only one "Tumor Growth" sample. This is an example of a highly biased cluster, in which the unknown can be unambiguously assigned to the "No Tumor Growth" group with a relatively high level of confidence. Finally, in **Figure 4.5d**, the unknown is located within a homogeneous cluster (or "zone") in which all members belong to the "Tumor Growth" group. Therefore, the test sample can be assigned to the "Tumor Growth" group with a very high level of confidence. In this study, the diagnostic accuracy of the "guilt-by-association" classification technique was assessed: i. for all unknowns (excluding indeterminates); ii. for the set of unknowns within highly biased and homogeneous clusters; and iii. for unknowns within homogeneous clusters only. The diagnostic accuracy of classifying test samples using "guilt-by-association" within each of these groups is discussed in **Sections 3.3** and **3.4**.

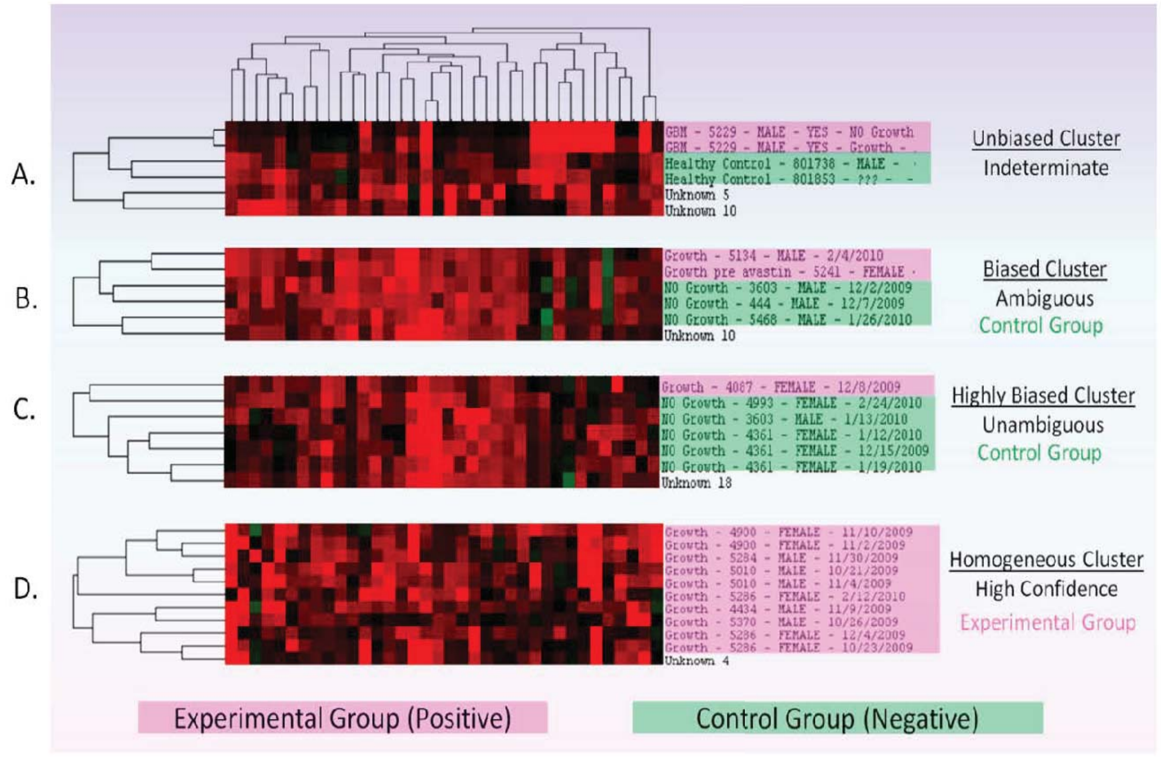


Figure 4.5 Classifying test samples via "Guilt by Association": illustrative examples. In **(a)**, the cluster is unbiased so the classification of the test samples ("unknowns") is not possible. In **(b)**, the cluster is biased, but only slightly, so the test sample is assigned with low confidence to the control group based on the majority diagnosis. In **(c)**, the cluster is highly biased, so the test sample can be assigned unambiguously to the control group. In **(d)**, the cluster is homogeneously comprised of patients from the experimental group, so the test sample is assigned to this group with very high confidence.

4.4 Appendix: Excel Macros for Data Analysis

4.4.1 Processing GenePix-Scanned Array Data to Create a Master Dataset

```
'The following subroutine ("RunProgramFor6x6Arrays") takes GenePix data
'(text format) that has been transferred to an excel file and formats it
'for statistical and graphical analysis. In particular, for each of the 12
'blocks (patient wells) in each file (slide), a new sheet is created. The
'oligo names are then tabulated on each sheet in exactly the order in
'which they appear on the slide. The spots with the highest intensity in
'the green channel (Cy3) are then assigned as oligo M (reference oligo),
'and the tabulated oligo order is then used to assign all other spots. The
'six repeats of each oligo/antibody spot (red channel - Cy5) are then
'organized into a list beneath each oligo name, and these columns are then
'sorted in alphabetical order by oligo name. The mean and standard
'deviation of six repeats are calculated for each oligo/antibody. Outliers
'are removed and the mean and standard deviation are then re-calculated.
'The mean values are then graphed (for each sheet), with error bars
'corresponding to the standard deviations. The mean intensity values for
'all proteins for each of the 12 sheets are then collated into one
'(additional) sheet. Furthermore, a baseline (or background) intensity is
'calculated for each graph (patient well) based on the average intensity
'of the 4 proteins with the lowest intensities. This baseline is added to
'each patient graph, and the baseline-subtracted protein intensity values
'are calculated. Each of the 12 patient graphs is then transferred to a
'separate slide within a Powerpoint file.


Sub RunProgramFor6x6Arrays()

   FormatSheetFor6x6Arrays

   'Formats the GenePix data in excel such that only the "Block", "Row",
   '"Column","Cy5 Mean","Cy5 SD","Cy3 Mean", and "Cy3 SD" Columns are
   'shown (minus the headings).Due to variation in the GenePix output
   'file, this step must sometimes be performed manually.

   NewSheetForEachBlock
   'Creates a new sheet for each block/well of patient data (for a total of
   '12 sheets)

   WritesOligoOrderOnSheetFor6x6Array
   'Tabulates the order of the oligos exactly as they appear on the slide
   'The following macros are run on all 12
   'sheets(patient samples) in the excel file.

   PlaceOligoOrderOnEachSheet
   'Copies this table to all sheets of the excel file

   PlaceMOnEachSheetFor6x6Arrays
   'Finds the highest intensity green (Cy3) spots and assigns them
   'as oligo M
```

```
OligoIDFor6x6ArrayForEachSheet
'Fills in the oligo ID for each spot using M as a reference and the
tabulated oligo order.

OligoAndIntensityOnlyForEachSheet
'Result displays only the oligo ID and associated mean Cy5 (protein)
intensity

CollatesIntensityValues4EachOligo4EachSheet
'Displays intensity values of all 6 spot repeats under each oligo ID.

AlphabeticalOrderForEachSheet
'Lists the columns in alphabetical order by oligo ID: i.e.
"A,B,C...Z,AA,BB,CC...

MeanAndStandardDeviationForEachSheet
'Displays the mean intensity and standard deviation for the 6 spot
repeats of each oligo/protein

EliminatesLowValuesForEachSheet
'Eliminates intensity values less than a set threshold, typically ~100
for background.

FindsConsistencyAndThrowsOutSingleOutlierForEachSheet
'Throws out 3 of the 6 repeats for a given oligo/protein if the spots in
the first round of array spotting are significantly brighter than those
in the second round.
'Otherwise, throws out a single outlier (that minimizes the SD of the
remaining repeats).

InsertGraphForEachSheet
'Inserts graph of the mean intensity values of each oligo/protein
 for each patient sample (sheet).

FormatChartForEachSheet
'Formats each graph to a set max x- and y-scale (typically 37 and
15000)

ErrorBarsForEachSheet
'Inserts up and down error bars with magnitude equal to the standard
deviation.

CollateData
'Collates the mean intensities of proteins from all 12 patient samples
onto a single sheet.

Baseline
'Uses average of 4 lowest protein intensity values as baseline, then
subtracts all values by the baseline value.
'It then collates the background-subtracted data from all sheets on a
single sheet.

TransferAllGraphsOnSheetsToPowerpoint
```

```
   'Creates a new Powerpoint file and transfers all graphs on each sheet
   to a separate slide

End Sub
```

Procedures Called by the “RunProgramFor6x6Arrays” Macro

```
Sub FormatSheetFor6x6Arrays()

'This subroutine trims the GenePix data file in excel so that it
'contains only the "Block", "Row", "Column","Cy5 (635 nm wavelength)
'Mean","Cy5 SD","Cy3 (594 nm wavelength) Mean", and "Cy3 SD" Columns
'are shown. These row containing the headings is subsequently deleted.
'This subroutine runs properly if the "Block" heading appears in the
'first column when the file is transferred from GenePix to Excel.
'Otherwise, the file should be formatted manually.

    Rows("1:32").Select
    Selection.Delete Shift:=xlUp
    Columns("D:I").Select
    Selection.Delete Shift:=xlToLeft
    Columns("F:M").Select
    Selection.Delete Shift:=xlToLeft
    Columns("H:H").Select
    Columns("F:F").ColumnWidth = 8.89
    Columns("H:AQ").Select
    Selection.Delete Shift:=xlToLeft
    Range("J4").Select
    Columns("D:D").ColumnWidth = 9.33
    Rows("1:1").Select
    Selection.Delete Shift:=xlUp

End Sub
```

```
Sub NewSheetForEachBlock()

'This program creates 9 additional worksheets and fills each
'of the resulting 12 worksheets with data from one of the 12
'wells (corresponding to blocks on "Sheet1") on the slide

'The data must reside on "Sheet1" and the Workbook
'must start out with exactly 3 worksheets for this
'program to work properly

   ActiveWorkbook.Worksheets("Sheet1").Range("A1").Select

   For i = 1 To 9

      Sheets.Add After:=Sheets(Sheets.Count)

   Next i
   For i = 1 To 11

      ActiveWorkbook.Worksheets("Sheet1").Select
      Range(Range("A1").Offset((217 * i) - i, 0), _
      Range("A1").Offset(i + (215 * (i + 1)), 6)).Select
      Selection.Cut
      ActiveWorkbook.Worksheets(i + 1).Select
      Range("A1").Select
      ActiveSheet.Paste

    Next i

End Sub
```

```
Sub WritesOligoOrderOnSheetFor6x6Array()

'This program creates a 6x6 table of the 36 oligo names
'(at 'Sheet1, L7') in the row/column order in which they
'appear on the slide.

   ActiveWorkbook.Worksheets(1).Select

   Range("L7") = "U"
   Range("L8") = "II"
   Range("L9") = "QQ"
   Range("L10") = "WW"
   Range("L11") = "F"
   Range("L12") = "L"

   Range("M7") = "S"
   Range("M8") = "HH"
   Range("M9") = "PP"
   Range("M10") = "VV"
   Range("M11") = "E"
   Range("M12") = "K"

   Range("N7") = "P"
   Range("N8") = "CC"
   Range("N9") = "NN"
   Range("N10") = "UU"
   Range("N11") = "D"
   Range("N12") = "J"

   Range("O7") = "O"
   Range("O8") = "BB"
   Range("O9") = "MM"
   Range("O10") = "TT"
   Range("O11") = "C"
   Range("O12") = "I"

   Range("P7") = "N"
   Range("P8") = "AA"
   Range("P9") = "KK"
   Range("P10") = "SS"
   Range("P11") = "B"
   Range("P12") = "H"

   Range("Q7") = "M"
   Range("Q8") = "Z"
   Range("Q9") = "JJ"
   Range("Q10") = "RR"
   Range("Q11") = "A"
   Range("Q12") = "G"

End Sub
```

```
Sub PlaceOligoOrderOnEachSheet()

'This subroutine copies the table of ordered oligo names (created in
'"WritesOligoOrderOnSheetFor6x6Array") and pastes it at "L7"
'on each of the 12 sheets

   For i = 1 To 11

      ActiveWorkbook.Worksheets("Sheet1").Select
      Range("K7:V12").Select
      Selection.Copy
      ActiveWorkbook.Worksheets(i + 1).Select
      Range("K7").Select
      ActiveSheet.Paste

   Next i

End Sub
```

```
Sub PlaceMOnEachSheetFor6x6Arrays()

'This subroutine finds the reference oligos M in all 12 sheets
'(by running "FindMFor6x6Arrays" in each sheet)

   For i = 1 To 12

      ActiveWorkbook.Worksheets(i).Select
      FindMFor6x6Arrays

   Next i

End Sub
```

```
Sub OligoIDFor6x6ArrayForEachSheet()

'This subroutine runs the "OligoIDFor6x6Array" program on each sheet/
'(block) to fill in the oligo name assignments for all spots in
'all blocks/sheets

   For i = 1 To 12

      ActiveWorkbook.Worksheets(i).Select
      OligoIDFor6x6Array

   Next i

End Sub
```

```
Sub OligoAndIntensityOnlyForEachSheet()

'This subroutine trims the data set to just the column of oligo names
'and their associated red-channel (Cy5) mean intensities for all
'12 sheets (blocks)

   For i = 1 To 12

      ActiveWorkbook.Worksheets(i).Select
      Range("A2").Select
      OligoAndIntensityOnly

   Next i

End Sub
```

```
Sub CollatesIntensityValues4EachOligo4EachSheet()

'This subroutine lists the intensity values for the six spot repeats
'of each oligo/antibody under the name of that oligo (for all 36
'oligos), and repeats this for all 12 sheets/blocks.

Dim i As Integer

   For i = 1 To 12

      ActiveWorkbook.Worksheets(i).Select
      CollatesIntensityValues4EachOligo

   Next i

End Sub
```

```
Sub AlphabeticalOrderForEachSheet()

'This subroutine places the columns of oligo intensity values in
'alphabetical order according to their oligo names:
'Importantly, it ensures that ordering is from A->Z, followed
'by AA->WW, as opposed to AA coming directly after A, and so forth.
'It does this by adding an extra worksheet, placing double-letter oligo
'names in that sheet, alphabetically ordering them, and then appending
'them with the ordered single-letter names in the previous
'sheet. The extra sheet is then deleted. This is repeated for all
'12 worksheets. A command prompt asks the user whether they want to
'delete the extra sheet (12 times). Click "Okay" all 12 times.

Dim i, j, StringLength As Integer
Dim myString As String
Dim ws As Worksheet

i = 0
j = 1

   For j = 1 To ActiveWorkbook.Sheets.Count

      i = 0
      ActiveWorkbook.Worksheets(j).Select
      Sheets.Add After:=ActiveSheet
      ActiveSheet.Name = "TwoLetterOligos"
      ActiveWorkbook.Worksheets(j).Select
      Range("C1").Select

      Do
         myString = Range("C1").Offset(0, i).Text
         StringLength = Len(myString)

         If StringLength > 1 Then

            ActiveCell.EntireColumn.Select
            Selection.Cut
            ActiveWorkbook.Worksheets("TwoLetterOligos").Select
            Range("C1").Offset(0, i).Select
            ActiveSheet.Paste

         End If

            ActiveWorkbook.Worksheets(j).Select
            i = i + 1
            Range("C1").Offset(0, i).Select

      Loop Until i = 36

      ActiveWorkbook.Worksheets(j).Select
      DeleteEmptyColumns
      AlphabeticalOrder
      ActiveWorkbook.Worksheets("TwoLetterOligos").Select
      DeleteEmptyColumns
      AlphabeticalOrder

      Range("A1:Z7").Select
      Selection.Cut
      ActiveWorkbook.Worksheets(j).Select
      Range("A1").Select

      Do
         Range("A1").Offset(0, k).Select
         k = k + 1
      Loop Until IsEmpty(ActiveCell)

      ActiveSheet.Paste
      ActiveWorkbook.Worksheets("TwoLetterOligos").Delete
      k = 0
   Next j

End Sub
```

```
Sub MeanAndStandardDeviationForEachSheet()

'This subroutine outputs the mean and standard deviation of the intensity
'values for the six spot repeats for each oligo/antibody (beneath
'each list of intensity values). This is repeated for all 12 sheets.

Dim i As Integer

   For i = 1 To 12

      ActiveWorkbook.Worksheets(i).Select
      MeanAndStandardDeviation

   Next i

End Sub
```

```
Sub EliminatesLowValuesForEachSheet()

'This subroutine deletes all data values on a sheet that are less than
'100 Intenisty Units. Typically, such low intensity values correspond
'to background, and suggest a defect in the spot loading or assay
'in that region. However, it could also suggest that the area was
'covered by PDMS and therefore unavailable for the assay.

   For i = 1 To 12

      ActiveWorkbook.Worksheets(i).Select
      EliminatesLowValues

   Next i

End Sub
```

```
Sub FindsConsistencyAndThrowsOutSingleOutlierForEachSheet()

'This subroutine carries out the two-mode outlier elimination of
'the "FindsConsistencyOrThrowsOutASingleOutlier" code, and repeats it
'for all 12 sheets/blocks

Dim ws As Worksheet

   For i = 1 To 12

      ActiveWorkbook.Worksheets(i).Select
      FindsConsistencyOrThrowsOutASingleOutlier

   Next i
```

```
End Sub
```

```
Sub InsertGraphForEachSheet()

'This subroutine graphs the mean intensity values for each column of
'oligo/protein intensities (on the same graph). As a result, the
'mean intensities for all proteins in a patient sample can quickly
'be evaluated visually. This is repeated for all 12 patient samples
'(worksheets) assayed on the slide.

   For i = 1 To 12

      ActiveWorkbook.Worksheets(i).Select
      InsertGraph

   Next i

End Sub
```

```
Sub FormatChartForEachSheet()

'This subroutine formats each chart to maximum scales on the x-
'and y- axes of 37 and 15000, respectively. Of course these
'values can be re-set to values of one's choosing. This is
'repeated for all 12 sheets/blocks.



   For i = 1 To 12

      ActiveWorkbook.Worksheets(i).Select
      FormatChart

   Next

End Sub
```

```
Sub ErrorBarsForEachSheet()

'This subroutine adds two-sided error bars (up- and down- magnitudes
'corresponding to standard deviations) to the graph of mean
'protein intensity values. This is repeated for all 12 sheets
'(all 12 patient graphs).

   For i = 1 To 12
      ActiveWorkbook.Worksheets(i).Select
      ErrorBars
   Next

End Sub
```

```
Sub InsertBaselineForEachSheet()
```

```
'This subroutine sorts the mean protein intensities from smallest to
'largest and places them in row 17. It then takes the average of
'the first 4 and 9 smallest values and places them in rows 18
'19, respectively, as well as adding them as baselines to the
'patient graph. This is based on the observation that the 4
'lowest values in a patient graph typically exhibit intensities
'equivalent to a negative control (non-specific IgG). This is
'repeated for all 12 sheets.

   For i = 1 To 12
      ActiveWorkbook.Worksheets(i).Select
      InsertBaseline
   Next

End Sub
```

```
Sub SubtractBaselineForEachSheet()

'This subroutine subtracts the baseline value from each of the 35
'mean protein intensities to yield a baseline-subtracted
'net mean protein intensity. It places these values in Row 24.
'This is repeated for all 12 sheets.

   For i = 1 To 12
      ActiveWorkbook.Worksheets(i).Select
      SubtractBaseline
   Next

End Sub
```

```
Sub CollateBackgroundSubtractedData()

'This subroutine collates all background subtracted mean protein
'intensity values from all 12 worksheets onto a single
'worksheet.

Sheets.Add After:=Sheets(Sheets.Count)
Range("A1") = "Collated Background Subtracted Data"

   For i = 1 To 12

      ActiveWorkbook.Worksheets(i).Select
      Range("24:24").Select
      Selection.Copy
      ActiveWorkbook.Worksheets(Sheets.Count).Select
      Range("A2").Offset(i, 0).Select
      ActiveSheet.Paste

   Next i

End Sub
```

```
Sub Baseline()

'This subroutine runs the "InsertBaselineForEachSheet",
'"SubtractBaselineForEachSheet", and
```

```
'"CollateBackgroundSubtractedData" subroutines

   InsertBaselineForEachSheet
   SubtractBaselineForEachSheet
   CollateBackgroundSubtractedData

End Sub
```

```
Sub CollateData()

'This subroutine collates the mean protein intensity values for all
'12 patients on a single sheet.

   Sheets.Add After:=Sheets(Sheets.Count)
   Range("A1") = "Collated Data"
   For i = 1 To 12
      ActiveWorkbook.Worksheets(i).Select
      Range("11:11").Select
      Selection.Copy
      ActiveWorkbook.Worksheets(Sheets.Count).Select
      Range("A2").Offset(i, 0).Select
      ActiveSheet.Paste
   Next i

End Sub
```

```
Sub CollateStandardDeviations()

'This subroutine collates the standard deviations for all 36
'proteins from all 12 worksheets (into a table of values)
'onto a single sheet. Each patient's values are listed in
'a separate row.

   Sheets.Add After:=Sheets(Sheets.Count)
   Range("A1") = "Collated Standard Deviations"

   For i = 1 To 12

      ActiveWorkbook.Worksheets(i).Select
      Range("12:12").Select
      Selection.Copy
      ActiveWorkbook.Worksheets(Sheets.Count).Select
      Range("A2").Offset(i, 0).Select
      ActiveSheet.Paste

   Next i

End Sub
```

```
'"TransferAllGraphsOnSheetsToPowerpoint" Procedure - See Appendix 4.8
```

Subroutines Called by the Above Procedures

```
Sub FindMFor6x6Arrays()

'This subroutine searches for intensity values in the green (Cy3) channel
'that exceed 20000 AU, and labels them as the reference oligo M

   Range("F1").Select

   Do
      If ActiveCell.Value > 20000 Then
         ActiveCell.Offset(0, 2).Value = "M"
      End If

      ActiveCell.Offset(1, 0).Select

   Loop Until ActiveCell.Offset(-1, 2).Value = "M"

      If ActiveCell.Offset(5, 0).Value > 20000 Then
         ActiveCell.Offset(5, 2).Value = "M"
      End If

End Sub
```

```
Sub OligoIDFor6x6Array()

'This subroutine searches for the three sets of oligo M pairs in a
'worksheet, placed by the "FindMFor6x6Arrays" or the
'"PlaceMOnEachSheetFor6x6Arrays" programs, and uses them as a reference to
'guide the correct assignment of oligo names (using the table of ordered
'oligo names created in "WritesOligoOrderOnSheetFor6x6Array" and/or
'"PlaceOligoOrderOnEachSheet" to all other spots listed (by rows and
'columns) in the block (on the sheet)

i, j, k, m = 0

Range("H1").Select

   Do

      If ActiveCell.Value = "M" Then

      'Once this condition is satisfied, the index i gets the value of the
       column previous to M; this is useful because that's how many cells
       we need to count back to get to and select the first column within
       the row in which M resides; the index i's value does not change
       from this point until the the entire block is sequenced

         m = ActiveCell.Offset(0, -5).Value
         'the index M gets the row number at which oligo M resides

         For j = 0 To 5
         'The index j allows us to select each cell in M's row, starting i
```

```
'cells above (or i cells to the left of M in the array sequence)

   If Not IsEmpty(Range("Q7").Offset(0, -i + j).Cells) Then

      ActiveCell.Offset(-i + j, 0).Value = _
      Range("Q7").Offset(0, -i + j).Value

      For k = 0 To 17

         ActiveCell.Offset((-i + j) - 12 * (m - 1) + 6 *(2*k), _
          0) = Range("Q7").Offset((6 - (m - k - 1)) Mod 6, _
          -i + j).Value
         ActiveCell.Offset((-i + j) - 12 * (m - 1) + _
          6 * ((2 * k) + 1),  0)
          Range("Q7").Offset((6 - (m - k - 1)) Mod 6, -i+j).Value

      Next k

'Since the oligo M resides in the mth row of the block (say 4th
'row), we need to offset by 3 rows to get us to the first row of
'the block. To get to the first row, we therefore need to
'multiply 3 (in this example) by the number of columns (12) in
'the array sequence (3x12=36). In other words, if we subtract 36
'from the i+j offset(from M's location in the oligo assignment
'column), we will hit the oligo in the array sequence at the same
'column offset position but in the first row of the block. The
'oligo at the same position in the next row down is assigned to
'the cell 12 cells below in the oligo assignment column and so
'forth (in multiples of 12) until that oligo position in all 18
'rows are accounted for. Notice that if oligo M is in the 4th row
'of of the block, the value of the oligo in the first row of the
'block (3 rows up)is the same as if you go (6-3) rows down in the
'array sequence table, hence the 6-(M-1). By taking the Mod 6 of
'this value, we ensure that we always stay within the confines of
'the 6-row array sequence table. The index k then allows us to
'scan through values in each row at the same column offset
'position.

   Else

      ActiveCell.Offset(-i+j, 0).Value = Range("Q7").Offset(0, _
       -i + j - 6).Value

      For k = 0 To 17

         ActiveCell.Offset((-i + j) - 12 * (m - 1) + 6 * (2*k), _
          0) = Range("Q7").Offset((6 - (m - k - 1)) Mod 6, _
          -i + j - 6).Value
         ActiveCell.Offset((-i + j) - 12 * (m - 1) + 6 * ((2*k) _
          + 1), 0) = Range("Q7").Offset((6 - (m - k - 1)) Mod 6, _
          -i + j - 6).Value

      Next k
```

```
            End If

         Next j

      End If

      ActiveCell.Offset(1, 0).Select
      'This will continue to offset the selected cell until a cell
      'containing M is reached

      i = (i + 1) Mod 6
      'The index i is the same as the block column value of the previous
      'cell in the oligo assignment column

   Loop Until ActiveCell.Offset(-1, 0).Value = "M"

End Sub
```

```
Sub OligoAndIntensityOnly()

'This subroutine trims the data set to just the column of oligo names
'and their associated red-channel (Cy5) mean intensities. The names
'are moved from column "H" to column "A". The intensities are
'moved from column "D" to column "B". All other data is deleted.

   Columns("H:H").Select
   Selection.Cut
   Columns("A:A").Select
   ActiveSheet.Paste

   Columns("D:D").Select
   Selection.Cut
   Columns("B:B").Select
   ActiveSheet.Paste

   Columns("C:Z").Select
   Selection.Delete

   Rows("1:1").Select
   Selection.Insert Shift:=xlDown, CopyOrigin:=xlFormatFromLeftOrAbove

End Sub
```

```
Sub CollatesIntensityValues4EachOligo()

'This subroutine lists the intensity values for the six spot repeats
'of each oligo/antibody under the name of that oligo (for all 36
'oligos).

Dim i, j, k As Integer
Dim myRange As Object

   ActiveWorkbook.ActiveSheet.Select
   Range("A2").Select
   i, k = 1
   j = 0

   Do
      CurrentCell = ActiveCell.Value
      Range("A2").Select
      Range("A2").Offset(-1, i + 1).Select

      Set myRange = Range("C1")
      ActiveCell.Value = CurrentCell
      Range("A2").Select

      Do
         If ActiveCell.Value = CurrentCell Then

            myRange.Offset(k, j).Value = ActiveCell.Offset(0, 1)
            k = k + 1
            Range(ActiveCell, ActiveCell.Offset(0, 1)).Delete Shift:=xlUp
            ActiveCell.Offset(-1, 0).Select

         End If

         ActiveCell.Offset(1, 0).Select

      Loop Until IsEmpty(ActiveCell)

      Range("A2").Select
      i = i + 1
      j = j + 1
      k = 1

   Loop Until IsEmpty(ActiveCell)

End Sub
```

```
Sub AlphabeticalOrder()

'This subroutine sorts a list or table in alphabetical order by
'headings in the first row.

   Range("A1:AZ10").Select
   ActiveSheet.Sort.SortFields.Clear
   ActiveSheet.Sort.SortFields.Add Key:=Range("A1:AZ1"), _
    SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal

   With ActiveSheet.Sort

      .SetRange Range("A1:AZ10")
      .Header = xlGuess
      .MatchCase = False
      .Orientation = xlLeftToRight
      .SortMethod = xlPinYin
      .Apply
   End With

End Sub
```

```
Sub DeleteEmptyColumns()

'This subroutine deletes any empty columns from a worksheet
'containing the list of protein intensities arranged in
'alphabetical order by heading (protein name).

Dim i, j As Integer

i, j = 0

   Range("A1").Select

   Do
      If Not IsEmpty(ActiveCell) Then
         Range("A1").Offset(0, i).Select
         i = i + 1
      Else
         ActiveCell.EntireColumn.Delete
         j = j + 1
      End If

   Loop Until i + j = 50

End Sub
```

```
Sub MeanAndStandardDeviation()

'This subroutine outputs the mean and standard deviation of the intensity
'values for the six spot repeats for each oligo/antibody (beneath
'each list of intensity values).

   Range("A9").Select
   ActiveCell.FormulaR1C1 = "=AVERAGE(R[-7]C:R[-2]C)"
   Range("A9").Select
   Selection.Copy
   Range("A9:AJ9").Select
   ActiveSheet.Paste

   Range("A10").Select
   Application.CutCopyMode = False
   ActiveCell.FormulaR1C1 = "=STDEV(R[-8]C:R[-3]C)"
   Range("A10").Select
   Selection.Copy
   Range("A10:AJ10").Select
   ActiveSheet.Paste

End Sub
```

```
Sub EliminatesLowValues()

'This subroutine deletes all data values on a sheet that are less than
'100 Intenisty Units. Typically, such low intensity values correspond
'to background, and suggest a defect in the spot loading or assay
'in that region. However, it could also suggest that the area was
'covered by PDMS and therefore unavailable for the assay.

Dim i, j As Integer

i , j = 0

   Range("A2").Select

   For j = 0 To 35

      Range("A2").Offset(0, j).Select

      For i = 0 To 5

         If ActiveCell.Value < 100 Then

            ActiveCell.ClearContents

         End If

         ActiveCell.Offset(1, 0).Select

      Next i

   Next j

End Sub
```

```
Sub FindsConsistencyOrThrowsOutASingleOutlier()

'This subroutine eliminates outliers (from the list of intensity values
'of the six repeats for each oligo/column)from calculations of mean and
'standard deviation by one of two modes: 1) eliminating 3 outliers
'if the difference between intensity values in odd and even numbered
'rows is greater than 25%, or 2) if this is not the case, throwing
'out a single value that minimizes the standard deviation of the
'remaining values. The purpose of 1) is to circumvent an issue
'arising from array-spotting oligos in two separate runs: with half
'the repeats of each oligo spotted in the first run, and the other
'half spotted in the second run. By the time the second half are
'spotted, the humidity has caused the slides to become too resistant
'to oligo binding. As a result, the first set of (3) repeats yields
'significantly greater intensity values compared with the second set.
'In those cases, the second set of (3) repeats are eliminated from
'calculations of mean and standard deviation (and only the first set
'of (3) repeats is counted.
```

```
Dim i, j, k, x, y, z As Integer

i , j, k, x, y, z = 0

   Range("A2").Select

   For j = 0 To 35

      Range("A2").Offset(0, j).Select
      z = 0

      For i = 0 To 5

         If Not IsEmpty(ActiveCell) Then
            z = z + 1
         End If

         ActiveCell.Offset(1, 0).Select

      Next i

      If z > 3 Then

         Range("A2").Offset(0, j).Select
         i, x, y = 0

         Do While Not IsEmpty(ActiveCell)

            If (ActiveCell.Value - ActiveCell.Offset(1, _
             0).Value)/ActiveCell.Value > 0.25 Then
               x = x + 1
            End If

            y = y + 1
            ActiveCell.Offset(2, 0).Select

         Loop

         If IsEmpty(ActiveCell) And y < 3 Then

            x = 0
            y = 0
            Range("A2").Offset(1, j).Select


            For y = 0 To 2

               If Not IsEmpty(ActiveCell) Then

                  If (ActiveCell.Value - ActiveCell.Offset(-1, _
                     0).Value) / ActiveCell.Value > 0.25 Then
                     x = x + 1
                  End If
```

```
    End If

    ActiveCell.Offset(2, 0).Select

  Next y

End If

If x <> 3 Then

  x = 0
  y = 0
  Range("A2").Offset(0, j).Select

  Do While Not IsEmpty(ActiveCell)

    If (ActiveCell.Value - ActiveCell.Offset(1, _
     0).Value) / ActiveCell.Value < -0.25 Then

      x = x + 1

    End If

    y = y + 1
    ActiveCell.Offset(2, 0).Select

  Loop

  If IsEmpty(ActiveCell) And y < 3 Then

    x = 0
    y = 0

    Range("A2").Offset(1, j).Select

    For y = 0 To 2

      If Not IsEmpty(ActiveCell) Then

        If (ActiveCell.Value - ActiveCell.Offset(-1, _
         0).Value) / ActiveCell.Value < -0.25 Then
          x = x + 1
        End If

      End If

      ActiveCell.Offset(2, 0).Select

    Next y

  End If

End If
```

```
Range("A2").Offset(6, j).Select

If x = 3 Then

   ActiveCell.Offset(10, 0).FormulaR1C1 = "=Average(R[-16]C, _
    R[-14]C,R[-12]C)"
   ActiveCell.Offset(10, 0).Select
   Selection.Copy
   Selection.PasteSpecial Paste:=xlPasteValues, _
    Operation:=xlNone, SkipBlanks:=False, Transpose:=False
   ActiveCell.Offset(1, 0).Select
   ActiveCell.FormulaR1C1 = "=Average(R[-16]C,R[-14]C,R[-12]C)"
   ActiveCell.Select
   Selection.Copy
   Selection.PasteSpecial Paste:=xlPasteValues, _
    Operation:=xlNone, SkipBlanks:=False, Transpose:=False
   ActiveCell.Offset(-8, 0).FormulaR1C1 = "=Max(R[7]C, R[8]C)"
   ActiveCell.Offset(-8, 0).Select
   Selection.Copy
   Selection.PasteSpecial Paste:=xlPasteValues, _
    Operation:=xlNone, SkipBlanks:=False, Transpose:=False

   With ActiveCell.Font
      .Color = -16776961
      .TintAndShade = 0
   End With

   If ActiveCell.Value = ActiveCell.Offset(8, 0).Value Then
      ActiveCell.Offset(1, 0).FormulaR1C1 = "=stdev(R[-9]C, _
       R[-7]C,R[-5]C)"
      ActiveCell.Offset(1, 0).Select
      Selection.Copy
      Selection.PasteSpecial Paste:=xlPasteValues, _
      Operation:=xlNone, SkipBlanks:=False, Transpose:=False

      With ActiveCell.Font
         .Color = -16776961
         .TintAndShade = 0
      End With


      With Range("A2,A4,A6").Offset(1, j).Font
         .Color = -16776961
         .TintAndShade = 0
      End With

   End If

   If ActiveCell.Value = ActiveCell.Offset(7, 0).Value Then

      ActiveCell.Offset(1, 0).FormulaR1C1 = _
      "=stdev(R[10]C,R[-8]C,R[-6]C)"
      ActiveCell.Offset(1, 0).Select
      Selection.Copy
```

```
         Selection.PasteSpecial Paste:=xlPasteValues, _
         Operation:=xlNone, SkipBlanks:=False,Transpose:=False

         With ActiveCell.Font
            .Color = -16776961
            .TintAndShade = 0
         End With

         With Range("A2,A4,A6").Offset(0, j).Font
            .Color = -16776961
            .TintAndShade = 0
         End With

      End If

   Else

      Range("A2").Offset(i, j).Select
      i = 0

      For i = 0 To 5

         Range("A2").Offset(i, j).Select
         Selection.Cut
         Range("A2").Offset(30, j).Select
         ActiveSheet.Paste
         Range("A2").Offset(i, j).Select
         ActiveCell.Offset(12, 0).FormulaR1C1 = "=stdev(R" & _
          2 & "C:R" & 7 & "C)"
         ActiveCell.Offset(12, 0).Select
         Selection.Copy
         Selection.PasteSpecial Paste:=xlPasteValues, _
          Operation:=xlNone, SkipBlanks:=False,Transpose:=False
         ActiveCell.Offset(10, 0).FormulaR1C1 = "=average(R"& _
          2 & "C:R" & 7 & "C)"
         ActiveCell.Offset(10, 0).Select
         Selection.Copy
         Selection.PasteSpecial Paste:=xlPasteValues, _
         Operation:=xlNone, SkipBlanks:=False,Transpose:=False
         Range("A2").Offset(30, j).Select
         Selection.Cut
         Range("A2").Offset(i, j).Select
         ActiveSheet.Paste

      Next i

      Range("A2").Offset(19, j).FormulaR1C1 = "=min(R" & _
       14 & "C:R" & 19 & "C)"
      Range("A2").Offset(19, j).Select
      Selection.Copy
      Selection.PasteSpecial Paste:=xlPasteValues, _
      Operation:=xlNone, SkipBlanks:=False, Transpose:=False
      Range("A14").Offset(0, j).Select
```

```
            Do While ActiveCell.Value <> Range("A21").Offset(0, j).Value
               ActiveCell.Offset(1, 0).Select
            Loop

            Range("A12").Offset(0, j).Value = ActiveCell.Value

            With Range("A12").Offset(0, j).Font
               .Color = -16776961
               .TintAndShade = 0
            End With

            Range("A11").Offset(0, j).Value = ActiveCell.Offset(10, _
             0).Value

            With Range("A11").Offset(0, j).Font
               .Color = -16776961
               .TintAndShade = 0
            End With

            ActiveCell.Offset(-12, 0).Select

            With Selection.Font
               .Color = -16776961
               .TintAndShade = 0
            End With

            Range("A14:A30").Offset(0, j).Delete

         End If

         Range("A13:A20").Offset(0, j).Select
         Selection.ClearContents

      Else

         Range("A2").Offset(9, j).FormulaR1C1 = "=average(R" & _
          2 & "C:R" & 7 & "C)"


         If z > 1 Then
            Range("A2").Offset(10, j).FormulaR1C1 = "=stdev(R" & 2 _
             & "C:R" & 7 & "C)"
         End If

         Range("11:12").Select
         Selection.Copy
         Selection.PasteSpecial Paste:=xlPasteValues, _
         Operation:=xlNone, SkipBlanks:=False, Transpose:=False

         With Selection.Font
            .Color = -16776961
            .TintAndShade = 0
         End With
      End If

   Next j

End Sub
```

```
Sub InsertGraph()

'This subroutine graphs the mean intensity values for each column of
'oligo/protein intensities (on the same graph). As a result, the
'mean intensities for all proteins in a patient sample can quickly
'be evaluated visually.

   Rows("11:11").Select
   ActiveSheet.Shapes.AddChart.Select
   ActiveChart.SetSourceData Source:=ActiveSheet.Range("$11:$11")
   ActiveChart.ChartType = xlXYScatter
   ActiveChart.Axes(xlValue).Select
   ActiveChart.Axes(xlValue).MaximumScale = 60000

End Sub
```

```
Sub FormatChart()

'This subroutine formats each chart to maximum scales on the x-
'and y- axes of 37 and 15000, respectively. Of course these
'values can be re-set to values of one's choosing.


   ActiveSheet.ChartObjects(1).Activate

   If ActiveChart.HasLegend = True Then
      ActiveChart.Legend.Select
      Selection.Delete
   End If

   ActiveSheet.ChartObjects(1).Activate
   ActiveChart.Axes(xlCategory).Select
   ActiveChart.Axes(xlCategory).MinorUnit = 1
   ActiveChart.Axes(xlCategory).MajorUnit = 37
   Selection.MinorTickMark = xlInside
   ActiveChart.Axes(xlCategory).MaximumScale = 37
   ActiveChart.Axes(xlValue).Select
   ActiveChart.Axes(xlValue).MaximumScale = 15000

End Sub
```

```
Sub ErrorBars()

'This subroutine adds two-sided error bars (up- and down- magnitudes
'corresponding to standard deviations) to the graph of mean
'protein intensity values.

   ActiveWorkbook.ActiveSheet.Select
   ActiveSheet.ChartObjects(1).Activate

   With ActiveChart.SeriesCollection(1)
      .ErrorBar Direction:=xlY, Include:=xlBoth, _
      Type:=xlCustom, Amount:=ActiveSheet.Range("12:12"), _
      MinusValues:=ActiveSheet.Range("12:12")
   End With

End Sub
```

```
Sub InsertBaseline()

'This subroutine sorts the mean protein intensities from smallest to
'largest and places them in row 17. It then takes the average of
'the first 4 and 9 smallest values and places them in rows 18
'19, respectively, as well as adding them as baselines to the
'patient graph. This is based on the observation that the 4
'lowest values in a patient graph typically exhbiti intensities
'equivalent to a negative control (non-specific IgG).


   Range("A11:AJ11").Select
   Selection.Copy
   Range("A17:AJ17").Select
   ActiveSheet.Paste
   Application.CutCopyMode = False
   ActiveSheet.Sort.SortFields.Clear
   ActiveSheet.Sort.SortFields.Add Key:=Range("A17:AJ17"), _
    SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal

   With ActiveSheet.Sort
      .SetRange Range("A17:AJ17")
      .Header = xlGuess
      .MatchCase = False
      .Orientation = xlLeftToRight
      .SortMethod = xlPinYin
      .Apply
   End With

   Range("A18").Select
   ActiveCell.FormulaR1C1 = "=AVERAGE(R[-1]C,R[-1]C[4])"
   Range("A19").Select
   ActiveCell.FormulaR1C1 = "=AVERAGE(R[-2]C,R[-2]C[9])"
   Range("A18:A19").Select
   Selection.Copy
   Range("A18:AJ19").Select
   Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, _
    SkipBlanks:=False, Transpose:=False
   ActiveSheet.ChartObjects(1).Select
   ActiveChart.SeriesCollection.NewSeries
   ActiveChart.SeriesCollection(2).Name = "=""Series2"""
   ActiveChart.SeriesCollection(2).XValues = Range("$A$1:$AJ$1")
   ActiveChart.SeriesCollection(2).Values = Range("$A$18:$AJ$18")
   ActiveChart.SeriesCollection.NewSeries
   ActiveChart.SeriesCollection(3).Name = "=""Series3"""
   ActiveChart.SeriesCollection(3).XValues = Range("$A$1:$AJ$1")
   ActiveChart.SeriesCollection(3).Values = Range("$A$19:$AJ$19")
   ActiveChart.SeriesCollection(3).Select

   With Selection
      .MarkerStyle = 3
      .MarkerSize = 2
   End With

   ActiveChart.SeriesCollection(2).Select

   With Selection
      .MarkerStyle = 1
      .MarkerSize = 2
   End With

End Sub
```

```
Sub SubtractBaseline()

'This subroutine subtracts the baseline value from each of the 35
'mean protein intensities to yield a baseline-subtracted
'net mean protein intensity. It places these values in Row 24.

   Range("A24").Select
   ActiveCell.FormulaR1C1 = "=R[-13]C-R[-6]C"
   Range("A24").Select
   Selection.Copy
   Range("A24:AJ24").Select
   ActiveSheet.Paste
   Range("A24:AJ24").Select
   Selection.Copy
   Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, _
    SkipBlanks:=False, Transpose:=False

End Sub
```

Collating Patient Data from all Patient Files

```
Sub CollateAllCollatedStandardDeviations()

'This subroutine collates the tables of standard deviations
'from each open workbook into a single sheet within a
'new workbook. For this code to run properly, the tables
'of standard deviations must be located on the last
'worksheet of all workbooks (typically "Sheet26").

Dim i As Integer
Dim NewWorkbook As Excel.Workbook

Set NewWorkbook = Application.Workbooks.Add
```

```
   For i = 1 To Workbooks.Count

      Workbooks(i).Activate
      ActiveWorkbook.Worksheets(Sheets.Count).Activate
      ActiveSheet.Range("A3:AJ14").Select
      Selection.Copy
      NewWorkbook.Worksheets(1).Activate
      ActiveSheet.Range("A2").Offset(13 * (i - 1), 0).Select
      ActiveSheet.Paste

   Next i

End Sub
```

```
Sub CollateAllCollatedNonBaselineSubMeans()

'This subroutine collates the tables of non-baseline subtracted mean
'protein intensities from each open workbook into a single sheet
'within a new workbook. For this code to run properly, the
'tables of mean values must be located on "Sheet25" within
'each workbook.

Dim i As Integer
Dim NewWorkbook As Excel.Workbook

Set NewWorkbook = Application.Workbooks.Add

   For i = 1 To Workbooks.Count

      Workbooks(i).Activate
      ActiveWorkbook.Worksheets("Sheet25").Activate
      ActiveSheet.Range("A3:AJ14").Select
      Selection.Copy
      NewWorkbook.Worksheets(1).Activate
      ActiveSheet.Range("A2").Offset(13 * (i - 1), 0).Select
      ActiveSheet.Paste

   Next i

End Sub
```

4.4.2 Graphing Patient Data from the Master Dataset

```
'The following macro sorts the master dataset by patient name followed by
'blood collection date (such that all samples corresponding to eachpatient
'are listed in chronological order by collection date). Depending on which
'procedure is then used (see below for options), a variety of different
'graphical analyses are enabled. For example, if one chooses the
'"GraphEachSelection" procedure, the protein data for each row/sample will
'be graphed separately (Fluorescent Intensity vs. Protein Identity).
'Alternatively, if one chooses "GraphTimeCourseData3", each patient's time
```

```
'course data (Protein Intensity vs. Blood Collection Date) for eachprotein
'will be displayed on a single chart. (See below for description of other
'alternatives).

Sub RunGraphAllSelections()

    SortbyDateForEachName
    GraphEachSelection
      'Note: This line can be interchanged with GraphTimeCourseData,
      'GraphTimeCourseData2, or GraphTimeCourseData
    FormatAllChartsOnSheet
    TransferAllGraphsOnSheetsToPowerpoint

End Sub
```

```
Sub SortbyDateForEachName()

'This subroutine uses as input an excel file in which the patient
'last names have been sorted alphabetically in Column C (with
'first names in column D)and in which the blood collection
'dates are listed in column B. It then sorts the data set
'by date for each last name.

Dim i, j, k As Integer
Dim str As String

i, j = 0
k = 1

   ExtractFirstWord
   ActiveSheet.Cells(2, 3).Select

   Do While Not IsEmpty(ActiveCell)

      i = 0

      Do While InStr(1, Trim(ActiveSheet.Cells(2 + i + j, 3).Value),
      Trim(ActiveSheet.Cells(2 + j + (i + 1), 3).Value), vbTextCompare) <>
      0 And InStr(1, Trim(ActiveSheet.Cells(2 + i + j, 57).Value),
      Trim(ActiveSheet.Cells(2 + j + (i + 1), 57).Value), vbTextCompare) <>
      0

         i = i + 1

      Loop

      i = i + 1
      ActiveWorkbook.ActiveSheet.Sort.SortFields.Clear
      ActiveWorkbook.ActiveSheet.Sort.SortFields.Add Key:=Range("B:B"),
      SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal

      With ActiveWorkbook.ActiveSheet.Sort
         .SetRange Range(Cells((2 + j), 3), Cells((i + j + 1), _
            3)).EntireRow
              .Header = xlGuess
              .MatchCase = False
              .Orientation = xlTopToBottom
              .SortMethod = xlPinYin
              .Apply
        End With

        j = j + i
        ActiveSheet.Cells(2, 3).Offset(j, 0).Select

   Loop

End Sub
```

```
Sub GraphEachSelection()

'This subroutine creates a graph of the mean intensities for all 36
'proteins in each row of patient data (each row of data corresponds
'to a different patient sample). For this code to run properly, the
'36 proteins must be located in columns U:BD. The graphs are labeled
'with the patient name, diagnosis, growth status, chemotherapy drug,
'and blood collection date. Unlike earlier version of this code, in
'this version, the range of cells to be graphed is selected before the
'chart is created, which speeds up the computing time considerably. In
'addition, the marker size is more compact, and the chart title acquires
'the same color as the excel row from which it's derived. As in past
'versions, the chart background color alternates from blue to
'gray between different patients.

Dim i, j, k, m As Integer

i , j = 0
k , m = 1

   ActiveSheet.Cells(2, 3).Select

   Do While Not IsEmpty(ActiveCell)

      i = 0

      Do

         Union(Range(Cells(1, 21), Cells(1, 56)), Range(Cells(i + j + 2, _
         21), Cells(i + j + 2, 56))).Select
         ActiveSheet.Shapes.AddChart.Select

         'Baseline-Subtracted Data
         ActiveChart.SetSourceData Source:=Union(Range(Cells(1, 21),
         Cells(1,56)), Range(Cells(i + j + 2, 21), Cells(i + j + 2, 56))),
         PlotBy:=xlRows

         'Non-Baseline-Subtracted Data (Option)…
```

```
'ActiveChart.SetSourceData Source:=Union(Range(Cells(1, 99),
Cells(1, 134)), Range(Cells(i + j + 2, 99), Cells(i + j + 2,
134))), PlotBy:=xlRows

With ActiveChart

   .ChartType = xlXYScatter
   .SetElement (msoElementChartTitleAboveChart)

   With .ChartTitle
      .Text = StringConcat(" ", ActiveSheet.Cells(2 + i + j, _
       4).Value, ActiveSheet.Cells(2 + i + j, 3).Value, "-", _
       ActiveSheet.Cells(2 + i + j, 12).Value, "-", _
       ActiveSheet.Cells(2 + i + j, 1).Value, Chr(10), _
       "Avastin", ActiveSheet.Cells(2 + i + j, 14).Value, _
       Chr(10), CStr (ActiveSheet.Cells(2 + i + j, 2).Value))
         .Font.Size = 10
         .Font.Name = "Calibri (Body)"
         .Font.Color = ActiveSheet.Cells(2 + i + j, _
          3).Font.Color
   End With

   With .Axes(xlCategory)
      .MinorUnit = 1
      .MajorUnit = 37
      .MaximumScale = 37
      .MinorTickMark = xlTickMarkInside
      .TickLabels.Delete
   End With

    With .Axes(xlValue)
      .MinorUnit = 1000
      .MajorUnit = 5000
      .MinimumScale = -5000
      .MaximumScale = 30000
    End With

   .HasLegend = False
   .SeriesCollection(1).ErrorBar Direction:=xlY, _
    Include:=xlBoth, Type:=xlCustom, _
    Amount:=ActiveSheet.Range(Cells(i + j + 2, 58), _
    Cells(i + j + 2, 93)), _
    MinusValues:=ActiveSheet.Range(Cells(i + j + 2, 58), _
    Cells(i + j + 2, 93))
    .SeriesCollection(1).ErrorBars.Border.ColorIndex = 5

End With

If k > 0 Then
   With ActiveChart.ChartArea.Fill
      .Visible = True
      .ForeColor.SchemeColor = 15
      .BackColor.SchemeColor = 17
      .TwoColorGradient msoGradientHorizontal, 1
```

```
            End With

         End If

         If k < 0 Then
            With ActiveChart.ChartArea.Fill
               .Visible = True
               .ForeColor.SchemeColor = 41
               .BackColor.SchemeColor = 17
               .TwoColorGradient msoGradientHorizontal, 1
            End With
         End If

         For m = 1 To ActiveChart.SeriesCollection.Count
            ActiveChart.SeriesCollection(m).MarkerSize = 4
          Next m

         i = i + 1
         ActiveSheet.ChartObjects(i + j).Visible = False

      Loop Until InStr(1, ActiveSheet.Cells(1 + i + j, 3).Value, _
         ActiveSheet.Cells(1 + j + (i + 1), 3).Value, vbTextCompare) = 0 _
         And InStr(1, ActiveSheet.Cells(1 + i + j, 52).Value, _
          ActiveSheet.Cells(1 + j + (i + 1), 52).Value, vbTextCompare) = 0

      k = -1 * k
      j = j + i
      ActiveSheet.Cells(2, 3).Offset(j, 0).Select

    Loop

    For i = 1 To ActiveSheet.ChartObjects.Count
        ActiveSheet.ChartObjects(i).Visible = True
    Next i

End Sub
```

```
Sub GraphTimeCourseData()

'This subroutine graphs all 36 proteins/spot mean intensity values
'at every collection time points for each patient. The chart
'title consist of the patient number and diagnosis. Each
'patient graph plots intensity as a function of protein ID.
'The color-coding for the time points is given in the legend.

Dim i, j, k As Integer
Dim str As String
Dim x As Object

i, j = 0
k = 1

   Application.ScreenUpdating = False
```

```
ExtractFirstWord
ActiveSheet.Cells(2, 3).Select

Do While Not IsEmpty(ActiveCell)

   i = 0

   Do While InStr(1, ActiveSheet.Cells(2 + i + j, 3).Value, _
   ActiveSheet.Cells(2 + j + (i + 1), 3).Value, vbTextCompare) <> 0 And
   InStr(1, ActiveSheet.Cells(2 + i + j, 57).Value, ActiveSheet.Cells(2
   + j + (i + 1), 57).Value, vbTextCompare) <> 0

      i = i + 1

   Loop

   i = i + 1
   Union(Range(Cells(1, 21), Cells(1, 56)), Range(Cells(j + 2, 21), _
   Cells(j + (i + 1), 56))).Select
   ActiveSheet.Shapes.AddChart.Select
   ActiveChart.SetSourceData Source:=Union(Range(Cells(1, 21), _
   Cells(1, 56)), Range(Cells(j + 2, 21), Cells(j + (i + 1), 56))), _
   PlotBy:=xlRows 'PlotBy:=xlColumns
   ActiveChart.ChartType = xlXYScatter
   ActiveChart.SetElement (msoElementChartTitleAboveChart)
      ActiveChart.ChartTitle.Text = StringConcat(" ", "Patient#", _
   ActiveSheet.Cells(2 + j, 5).Value, "-", ActiveSheet.Cells(2 + j, _
   12).Value)
   ActiveChart.ChartTitle.Font.Size = 10
   ActiveChart.ChartTitle.Font.Name = "Calibri (Body)"

   With ActiveChart.PlotArea
      .Width = 300
      .Height = 175
   End With

   With ActiveChart.Legend
      .Left = 300
      .Width = 50
      .Height = 300
       .Top = 35
       .Font.Size = 6
   End With

   k = 1

   For Each x In ActiveChart.SeriesCollection
      ActiveChart.SeriesCollection(k).Name = StringConcat(" - ", _
      CStr (ActiveSheet.Cells(1 + k + j, 2).Value), _
      ActiveSheet.Cells(1 + k + j, 14).Value, _
      ActiveSheet.Cells(1 + k + j, 1))
         x.MarkerSize = 4

      With ActiveChart.SeriesCollection(k)
```

```
            .MarkerForegroundColorIndex = 2 + k
            .MarkerBackgroundColorIndex = 2 + k
            .ErrorBar Direction:=xlY, Include:=xlBoth, _
             Type:=xlCustom, Amount:=ActiveSheet.Range(Cells(1 + k + j, _
             58), Cells(1 + k + j, 93)), _
             MinusValues:=ActiveSheet.Range(Cells(1 + k + j, 58), _
             Cells(1 + k + j, 93))
            .ErrorBars.Border.ColorIndex = 2 + k
         End With

         k = k + 1

      Next x


      With ActiveChart

         With .Axes(xlCategory)
            .MinorUnit = 1
            .MajorUnit = 37
            .MaximumScale = 37
            .MinorTickMark = xlTickMarkInside
         End With

         With .Axes(xlValue)
            .MinorUnit = 100
            .MajorUnit = 1000
            .MinimumScale = -1000
            .MaximumScale = 10000
         End With

      End With

      j = j + i
      ActiveSheet.Cells(2, 3).Offset(j, 0).Select

    Loop

    Application.ScreenUpdating = True

End Sub
```

```
Sub GraphTimeCourseData2()

'This subroutine creates 6 graphs showing mean protein intensity vs.
'collection date for each set of 6 (out of the 36) distinct
'proteins/spots for each patient. The chart title consist of
'the patient number and diagnosis. The color-coding for the
'proteins is given in the legend.


Dim i, j, k, ProteinGroup, intIndex As Integer
Dim vntLabels As Variant
Dim str As String
Dim x As Object

i , j = 0
k = 1

   Application.ScreenUpdating = False

   ExtractFirstWord
   ActiveSheet.Cells(2, 3).Select

   Do While Not IsEmpty(ActiveCell)

      i = 0

      Do While InStr(1, ActiveSheet.Cells(2 + i + j, 3).Value, _
       ActiveSheet.Cells(2 + j + (i + 1), 3).Value, vbTextCompare) <> 0
       And InStr(1, ActiveSheet.Cells(2 + i + j, 57).Value, _
       ActiveSheet.Cells(2 + j + (i + 1), 57).Value, vbTextCompare) <> 0

         i = i + 1

      Loop

      i = i + 1

      For ProteinGroup = 0 To 5

         Union(Range(Cells(1, 21 + (6 * ProteinGroup)), Cells(1, _
          26 + (6 * ProteinGroup))), Range("B1"), Range(Cells(j + 2, 2), _
          Cells(j + (i + 1), 2)), Range(Cells(j + 2, _
          21 + (6 * ProteinGroup)), Cells(j + (i + 1), _
          26 + (6 * ProteinGroup)))).Select
         ActiveSheet.Shapes.AddChart.Select
         ActiveChart.SetSourceData Source:=Union(Range(Cells(1, _
          21 + (6 * ProteinGroup)), Cells(1, 26 + (6 * ProteinGroup))), _
          Range("B1"), Range(Cells(j + 2, 2), Cells(j + (i + 1), 2)), _
          Range(Cells(j + 2, 21 + (6 * ProteinGroup)), _
          Cells(j + (i + 1), 26 + (6 * ProteinGroup)))), PlotBy:=xlColumns
         ActiveChart.ChartType = xlLineMarkers
         ActiveChart.SetElement (msoElementChartTitleAboveChart)
         ActiveChart.ChartTitle.Text = StringConcat(" ", "Patient#", _
          ActiveSheet.Cells(2 + j, 5).Value, "-", _
          ActiveSheet.Cells(2 + j, 12).Value)

         ActiveChart.ChartTitle.Font.Size = 10
         ActiveChart.ChartTitle.Font.Name = "Calibri (Body)"

         With ActiveChart.PlotArea
            .Width = 300
            .Height = 175
            .Top = 20
         End With

          With ActiveChart.Legend
   .Left = 320
   .Width = 50
   .Height = 70
   .Top = 50
   .Font.Size = 6
End With

k = 1

For Each x In ActiveChart.SeriesCollection

   x.MarkerSize = 4

   With ActiveChart.SeriesCollection(k)
      .MarkerForegroundColorIndex = 2 + k
      .MarkerBackgroundColorIndex = 2 + k
      .ErrorBar Direction:=xlY, Include:=xlBoth, _
       Type:=xlCustom, Amount:=ActiveSheet.Range(Cells(j + 2, _
       58 + (6 * ProteinGroup) + k - 1), Cells(j + i + 1, _
       58 + (6 * ProteinGroup) + k - 1)), _
       MinusValues:=ActiveSheet.Range(Cells(j + 2, _
       58 + (6 * ProteinGroup) + k - 1), Cells(j + i + 1, _
       58 + (6 * ProteinGroup) + k - 1))
      .ErrorBars.Border.ColorIndex = 2 + k

      With .Border
         .ColorIndex = 2 + k
         .Weight = 2.5
         .LineStyle = xlContinuous
      End With

   End With
   k = k + 1

Next x


With ActiveChart

   With .Axes(xlCategory)
      With .TickLabels
         .Alignment = xlCenter
         .Offset = 100
         .Orientation = -40
      End With
    End With

   With .Axes(xlValue)
      .MinimumScale = -1000
      .MaximumScale = 5000
   End With

   With .Parent
         .Left = 100
```

```
                 .Width = 500
                 .Top = 75
                 .Height = 440
            End With

         End With

      Next ProteinGroup

      j = j + i
      ActiveSheet.Cells(2, 3).Offset(j, 0).Select

   Loop

   Application.ScreenUpdating = True

End Sub
```

```
Sub GraphTimeCourseData3()

'This subroutine creates graphs showing mean protein intensity vs.
'collection date for the full set of 36 distinct proteins/
'spots for each patient. The chart title consist of the
'patient number and diagnosis. The color-coding for all
'proteins is given in the legend.

Dim i, j, k, ProteinGroup, intIndex As Integer
Dim vntLabels As Variant
Dim str As String
Dim x As Object

i = 0
j = 0
k = 1

   Application.ScreenUpdating = False

   ExtractFirstWord
   ActiveSheet.Cells(2, 3).Select

   Do While Not IsEmpty(ActiveCell)

        i = 0

      Do While InStr(1, ActiveSheet.Cells(2 + i + j, 3).Value, _
      ActiveSheet.Cells(2 + j + (i + 1), 3).Value, vbTextCompare) <> 0
      And InStr(1, ActiveSheet.Cells(2 + i + j, 57).Value, _
      ActiveSheet.Cells(2 + j + (i + 1), 57).Value, vbTextCompare) <> 0

         i = i + 1

      Loop
```

```
i = i + 1

Union(Range(Cells(1, 21), Cells(1, 56)), Range("B1"), _
Range(Cells(j + 2, 2), Cells(j + (i + 1), 2)), _
Range(Cells(j + 2, 21), Cells(j + (i + 1), 56))).Select
ActiveSheet.Shapes.AddChart.Select
ActiveChart.SetSourceData Source:=Union(Range(Cells(1, 21), _
Cells(1, 56)), Range("B1"), Range(Cells(j + 2, 2), _
Cells(j + (i + 1), 2)), _
Range(Cells(j + 2, 21), Cells(j + (i + 1), 56))), PlotBy:=xlColumns
'PlotBy:=xlRows
ActiveChart.ChartType = xlLineMarkers
ActiveChart.SetElement (msoElementChartTitleAboveChart)
ActiveChart.ChartTitle.Text = StringConcat(" ", "Patient#", _
ActiveSheet.Cells(2 + j, 5).Value, "-", ActiveSheet.Cells(2 + j, _
12).Value)
ActiveChart.ChartTitle.Font.Size = 10
ActiveChart.ChartTitle.Font.Name = "Calibri (Body)"

With ActiveChart.PlotArea
   .Width = 270
   .Height = 175
   .Top = 20
End With

With ActiveChart.Legend
   .Left = 350
   .Width = 50
   .Height = 330
   .Top = 10
   .Font.Size = 5
End With

k = 1

For Each x In ActiveChart.SeriesCollection

   x.MarkerSize = 4

   With ActiveChart.SeriesCollection(k)

      .MarkerForegroundColorIndex = 2 + k
      .MarkerBackgroundColorIndex = 2 + k
      .ErrorBar Direction:=xlY, Include:=xlBoth, Type:=xlCustom, _
       Amount:=ActiveSheet.Range(Cells(j + 2, _
       58 + (6 * ProteinGroup) + k - 1), Cells(j + i + 1, _
       58 + (6 * ProteinGroup) + k - 1)), _
       MinusValues:=ActiveSheet.Range(Cells(j + 2, _
       58 + (6 * ProteinGroup) + k - 1), Cells(j + i + 1, _
       58 + (6 * ProteinGroup) + k - 1))
      .ErrorBars.Border.ColorIndex = 2 + k

      With .Border
         .ColorIndex = 2 + k
```

```
            .Weight = 2.5
            .LineStyle = xlContinuous
         End With

      End With

      k = k + 1

   Next x


   With ActiveChart

      With .Axes(xlCategory)
         With .TickLabels
            .Alignment = xlCenter
            .Offset = 100
            .Orientation = -40
         End With
      End With

      With .Axes(xlValue)
         .MinimumScale = -5000
         .MaximumScale = 20000
      End With

      With .Parent
         .Left = 100
         .Width = 500
         .Top = 75
         .Height = 440
      End With

   End With

   j = j + i
   ActiveSheet.Cells(2, 3).Offset(j, 0).Select

Loop

Application.ScreenUpdating = True

End Sub
```

```
Sub FormatAllChartsOnSheet()

'This subroutine formats each chart to maximum scales on the x-
'and y- axes of 37 and 10000, respectively. Of course these
'values can be re-set to values of one's choosing. Tick
'marks are placed on the inside of the x-axis. The legend is
'deleted. This is repeated for all 12 sheets/blocks.

   For i = 1 To ActiveSheet.ChartObjects.Count
```

```
        ActiveSheet.ChartObjects(i).Activate

        If ActiveChart.HasLegend = True Then
           ActiveChart.Legend.Select
           Selection.Delete
        End If

           ActiveSheet.ChartObjects(i).Activate
           ActiveChart.Axes(xlCategory).Select
           ActiveChart.Axes(xlCategory).MinorUnit = 1
           ActiveChart.Axes(xlCategory).MajorUnit = 37
           Selection.MinorTickMark = xlInside
           ActiveChart.Axes(xlCategory).MaximumScale = 37
           ActiveChart.Axes(xlValue).Select
           ActiveChart.Axes(xlValue).MaximumScale = 10000

   Next i

End Sub
```

4.4.3 File Preparation for Cluster Analysis and Diagnostic Testing

```
'This section describes the “RunClusterPrep” macro, which formats and
'prepares cohort datasets for statistical analysis (by Excel and AnalyseIt)
'and for later cluster analysis (by Cluster 3.0). It also creates a
'worksheet for assessing the accuracy of classifying patients within
'hierarchical clusters based on “guilt-by-association”.

'This macro begins by creating a new directory, "NewTrialFolder", which
'contains a number of excel files: "Format4Cluster","Format4AnalyseIt",
'and "Diagnostic Performance" as well as a number of "Case" subfolders
'and an "All Text Files" folder.

'Sheet1 of "Format4Cluster" contains all the patient data (experimental
'and control) in a format that, once saved as a text document, can be used
'by the software Cluster 3.0. In particular, all the pertinent clinical
'information for each patient sample is listed in the first column. The
'first row contains only headers (i.e. protien names). The intersection
'between each row and column contains the intensity value of a single
'protein for a single patient sample.

'Sheet2 of "Format4Cluster" separates the experimental and control
'data and displays the calculated mean and median intensities for each
'protein in each group (both on the sheet and graphically). It also
'displays the differences (and root-mean-square distances) between
'experimental and control means and medians.

'"Format4AnalyseIt" contains the experimental and control group data for
'each protein in a format that can easily be transferred into and analyzed
'by "AnalyseIt", a statistical analysis add-in for Excel. Specifically,
'for each protein, the column of intensity values for experimental (red)
'and control(green) groups are situated adjacent to each other in table
'format. When the command button "Activate AnalyseIt-Dataset Defined" is
'clicked, an AnalyseIt excel file,"AnalyseIt-Dataset Defined" opens up
'into which the table of experimental and control columns for each protein
'can be transferred, one at a time, for a whole host of statistical tests
'available in the AnalyseIt toolbar. The macros "TransferNext2AnalyseIt"
'and "TransferPrevious2AnalyseIt" were written to allow one to toggle to
'the next or previous protein's data within the "Format4AnalyseIt"
'worksheet and instantly transfer that data table to the "AnalyseIt-
'Dataset Defined" worksheet by clicking on left or right arrows within the
'latter sheet.

'In addition, this subroutine facilitates diagnostic testing. It randomly
'assigns a certain number of patients (number specified by the user) to be
'"unknown" test samples. This can be repeated multiple times (i.e.
'multiple cases/tests), as specified by the user. Each of these case/test
'files (containing both data from known samples and randomly assigned
'unknowns) is saved into its own case subfolder within the
'"NewTrialFolder" directory. The resulting data sets are then saved as
'text documents (that are compatible with Cluster 3.0) in the "Text Files"
'folder within the case subfolder. Separately, all text files from all
'cases/tests are also saved in the "All Text Files" folder within the
'"NewTrialFolder" directory.

'The "Diagnostic Performance" file contains the actual diagnoses for all
'randomly assigned unknowns in all tests. However, these are hidden from
'view until the user has entered all their diagnostic predictions in the
'"Prediction" column and clicked on the "Diagnostic Performance!" command
'button. At that point, the predictions are scored and 2x2 contingency
'tables are created containing the numbers of true- and false- positives,
'and true- and false- negatives for each test. In addition, the
'specificity, sensitivity, and positive and negative predictive values 'for
each test are indicated. Most importantly, also created is a table 'that
contains the overall values (over all tests run) for all of these
'diagnostic parameters.


Public strNewFolderPathAndName As String
Public strFolderPathAndName As String
Public NumUnknowns As Integer   'Number of Unknowns for Test Set
Public NumProteins As Integer   'Number of Proteins to examine
Public TestNumber As Integer 'Number of Tests to Perform
Public CurrentTest As Integer
Public RangeA, RangeB As Range

'Note: NumUnknowns, NumProteins, and TestNumber are User-Defined


Sub RunClusterPrep()

Dim CurrentCasePathName As String

Application.ScreenUpdating = False
```

```
   Format4Cluster
   CreateNewDirectoryAndSaveAs 'Creates "NewTrialFolder" and Saves Excel
   Files as "Format4Cluster" and "Format4AnalyseIt Files"
   Workbooks("Format4AnalyseIt").Activate
   Format4AnalyseIt 'Formats the excel file for use with the AnalyseIt
   add-in in Excel"
   ActiveWorkbook.Save
   ActiveWorkbook.Close

   'Create Case Folders for each File of Unknowns

   For CurrentTest = 1 To TestNumber  'Test number is set on the user form

      CurrentCasePathName = strNewFolderPathAndName & "Case" & _
       CurrentTest & "\"
      MkDir CurrentCasePathName
      Workbooks.Open FileName:=strNewFolderPathAndName & _
      "Format4Cluster.xlsx"
      Workbooks("Format4Cluster").Activate
      SelectRandomCases (CurrentCasePathName)
      'Selects NumUnknowns random cases as unknowns (where NumUnknowns is
      'defined by the user), creates new sheet for each unknown with the
      'set of knowns, and saves as notepad file in Case\Text Files folder

      ActiveWorkbook.SaveAs FileName:=CurrentCasePathName & "Case" & _
      CurrentTest & ".xlsx", FileFormat:=xlOpenXMLWorkbook, _
      CreateBackup:=False
      'Saves Excel File containing 20 unknowns, one in each sheet, in the
       appropriate Case Folder
      ActiveWorkbook.Close

   Next CurrentTest

   Workbooks.Open FileName:=strNewFolderPathAndName & _
   "Format4Cluster.xlsx"
   ActiveWorkbook.Sheets(1).Cells.Copy
   Sheets(2).Select
   ActiveSheet.Paste
   TwoCategoriesGraphMeansMedians2
   'Outputs the mean and median intensity values for each protein within
    experimental and control groups (and graphs them).
   ActiveWorkbook.Save
   ActiveWorkbook.Close
   PrepareNewSheetForStatistics
   'Creates and formats a sheet for diagnostic testing
   ActiveWorkbook.Save
   ActiveWorkbook.Close

Application.ScreenUpdating = True

End Sub
```

Procedures Called by the “RunClusterPrep” Macro

```
'The following subroutines are used directly by the "RunClusterPrep"
'subroutine: Format4Cluster,CreateNewDirectoryAndSaveAs,Format4AnalyseIt,
'SelectRandomCases, TwoCategoriesGraphMeansMedians2, and
'PrepareNewSheetForStatistics.

Sub Format4Cluster()

'To be compatible with Cluster 3.0, header/label information can be placed
'only in the first row and column, with all remaining rows and columns
'containing the mean fluorescent intensity values for each protein
'(columns) within each patient sample (rows). (See Cluster 3.0 Manual).


'This subroutine formats a patient data file such that all the relevant
'clinical parameters (namely, tumor growth status, IOIS#, gender, blood
'collection date, current diagnosis, and chemo drug treatment,are
'concatenated in a single cell (in the left-most column). All other
'patient information columns except the protein data values are deleted,
'such that the data set begins in the 2nd column of the worksheet. The
'first row of headers (protein/conjugate names) is maintained.

Dim i As Integer

i = 0

   ActiveWorkbook.ActiveSheet.Activate
   Union(Range("P:T"), Range("F:K"), Range("C:D"), Range("BE:EF")).Select
   Selection.Delete

   'Column A = Growth Status
   'Column C = IOIS
   'Column D = Current Pathology
   'Column F = Avastin Status
   'Column E = Gender
   'Column B = Collection Date

   Range("C2").Select

   Do While Not IsEmpty(ActiveCell)
      Range("G2").Offset(i, 0).Value = StringConcat(" - ",
      Range("A2").Offset(i, 0).Value, Range("C2").Offset(i, 0).Value, _
      Range("D2").Offset(i, 0).Value, Range("F2").Offset(i, 0).Value, _
      Range("E2").Offset(i, 0).Value, _
      Range("B2").Offset(i, 0).Value)
      Range("C2").Offset(i, 0).Select
      i = i + 1
   Loop

   Range("G2").Offset(i - 1, 0).ClearContents
   Range("A:F").Delete
   Range("A1").ClearContents
```

```
End Sub
```

```
Sub CreateNewDirectoryAndSaveAs()

'This subroutine creates a new directory, "NewTrialFolder1", on the
'Desktop. If a folder named "NewTrialFolder1" already exists, the
'name of the new folder will be "NewTrialFolder2" and so forth.
'It then creates a subdirectory within this folder called
'"All Text Files". Finally, it saves the active excel workbook
'as "Format4Cluster" and "Format4AnalyseIt".

Dim strFolderPath As String
Dim n As Integer

n = 1

   strFolderPathAndName = "C:\Documents and Settings\Heath Group\Desktop"
   ActiveWorkbook.ActiveSheet.Cells.Copy
   Workbooks.Add
   ActiveSheet.Paste

   strNewFolderPathAndName = strFolderPathAndName & "\NewTrialFolder\"
   strFolderPathAndName = strFolderPathAndName & "\NewTrialFolder"

   Do While Dir(strNewFolderPathAndName, vbDirectory) <> ""
      strNewFolderPathAndName = strFolderPathAndName & n & "\"
      n = n + 1
   Loop

   MkDir strNewFolderPathAndName
   'This is now the NewTrialFolder\
   MkDir strNewFolderPathAndName & "All Text Files"
   'This Folder Goes into the NewTrialFolder

   ActiveWorkbook.SaveAs FileName:=strNewFolderPathAndName &
   "Format4Cluster.xlsx", _
   FileFormat:=xlOpenXMLWorkbook, CreateBackup:=False

   ActiveWorkbook.SaveAs FileName:=strNewFolderPathAndName &
   "Format4AnalyseIt.xlsx", _
   FileFormat:=xlOpenXMLWorkbook, CreateBackup:=False

End Sub
```

```
Sub Format4AnalyseIt()

'This procedure formats the experimental and control group data within a
cohort dataset so that it can easily be transferred into and analyzed
'by "AnalyseIt", a statistical analysis add-in for Excel. Specifically,
'for each protein, the column of intensity values for experimental (red)
'and control (green) groups are situated adjacent to each other.


   InsertColumns
   ExtractFirstWord4AnalyseIt
   ChangeCellFontColorAndPlaceColumnsAdjacently2
   PaintColumnFontBlack
   Range("A2").Select

   Call GenButtons("Activate AnalyseIt-Dataset Defined", _
    "OpenAnalyseItDataSetDefined")

End Sub
```

```
Sub SelectRandomCases(ByVal FilePathName As String)

'This subroutine selects a number of cases randomly to serve as
'unknowns in a test set. The number of random cases (NumUnknowns)
'is assigned by the user in the user form. After a case is assigned
'as an unknown, it is moved to the bottom of the patient sample
'list. The next unknown is randomly assigned from the list of
'remaining samples (excluding the previously assigned unknowns).

'The subroutine then calls two functions: the first creates a
'separate worksheet for the set of patient samples with each
'unknown, as well as with all the unknowns combined. The second
'function saves each of these as a notepad file.

'The subroutine receives the file path name as an argument which it
'relays to the "SaveToNotepad" function.

Dim RandomIndex, i, m, n, UpperBound As Integer

i = 0

   ActiveWorkbook.ActiveSheet.Activate

   Range("A2").Offset(i, 0).Select
   Do While Not IsEmpty(ActiveCell)
      Range("A2").Offset(i, 0).Select
      i = i + 1
   Loop

   UpperBound = i - 2

   For m = 1 To NumUnknowns

      RandomIndex = Int((UpperBound - 1 + 1) * Rnd + 1)
      Range("A2").Offset(RandomIndex, 0).EntireRow.Select
      Selection.Copy
      Range("A2").Offset(i, 0).Select
      ActiveSheet.Paste
      Range("A2").Offset(RandomIndex, 0).EntireRow.Select
      Selection.Delete
      UpperBound = UpperBound - 1

   Next m
   Range(Range("A2").Offset(i - m + 1, 0), Range("A2").Offset(i - 1, _
    NumProteins)).Select
   Selection.Font.ColorIndex = 3
   Range("A2").Offset(i - m).EntireRow.Select
   Selection.Delete

   Range("B:B").Select
   Selection.Insert
   Range("A:A").Select
   Selection.Copy
   Range("B:B").Select
   ActiveSheet.Paste

   For n = 0 To NumUnknowns - 1
      Range("B2").Offset(i - m + n).Value = StringConcat(" ", "Unknown", _
      n + 1)
   Next n

   Call NewSheetForEachUnknown(i, m)
   Call SaveToNotepad(i, FilePathName)

End Sub
```

```
Sub TwoCategoriesGraphMeansMedians2()

'This subroutine splits category 1 samples (typically
'experimental) and category 2 samples (typically control)
'by 8 empty rows. It then calculates the mean, median
'and standard deviation for all protein intensities in
'each category and lists them in blue under the last row
'of that category. Two graphs are created: one of the
'mean and the other of the median protein intensity values
'for the two categories (category 1 - red; cateogry 2 -
'green). The difference between the category means and
'medians are also calculated for each protein. The
'absolute value is taken for each of these, and sorted
'from smallest to largest. In addition the root-mean-square
'is calculated for the set of means and the set of medians.

Dim i, j, k, m, n As Integer
Dim str1, str2 As String

i = 0
j = 0

   ActiveWorkbook.ActiveSheet.Select

   'Insert Column
   ActiveSheet.Range("B2").Select
   Selection.EntireColumn.Select
   Selection.Insert Shift:=xlRight

   'Extracts Words Before First Dash in label and places it in Column B
   ExtractFirstWord4AnalyseIt
```

```
'ExtractWordsBeforeDash
Range("B2").Select
Selection.EntireColumn.Select
Selection.Copy
Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, _
SkipBlanks:=False, Transpose:=False
Range("B2").Select

'Find First Row after Category 1 (by counting number of rows - i - in
'category 1)
Do While StrComp(ActiveSheet.Range("B2").Offset(i, 0).Value, _
   ActiveSheet.Range("B2").Offset(i + 1, 0).Value, vbTextCompare) = 0
   i = i + 1
Loop

Range("B2").Offset(i + 1, 0).EntireRow.Select

'Place m = 8 empty rows between Category 1 and Category 2
For m = 1 To 8
   Selection.Insert Shift:=xlDown
Next m

'Find First Row after Category 2 (by counting number of rows - j - in
'category 2)
Do While StrComp(ActiveSheet.Range("B2").Offset((i + 1) + m + j, _
   0).Value, ActiveSheet.Range("B2").Offset((i + 1) + m + (j + 1), _
   0).Value, vbTextCompare) = 0
   j = j + 1
Loop

ActiveSheet.Range("B2").Offset((i + 1) + m + j, 0).Select

'Get String Values (such as "Growth" vs. "No Growth")
str1 = Range("B2").Value
str2 = Range("B2").Offset(i + 1 + m).Value

'Delete Column Containing Extracted First Word
ActiveSheet.Range("B2").Select
Selection.EntireColumn.Select
Selection.Delete

'Average, Median, and Standard Deviation of All Values For Each Protein
'in Category 1
Range("B2").Offset(i + 1, 0).Select
ActiveCell.FormulaR1C1 = "=AVERAGE(R[" & (-i - 1) & "]C:R[-1]C)"
Range("A2").Offset(i + 1, 0).Value = StringConcat(" ", str1, _
"- Average")

Range("B2").Offset(i + 2, 0).Select
ActiveCell.FormulaR1C1 = "=MEDIAN(R[" & (-i - 2) & "]C:R[-2]C)"
Range("A2").Offset(i + 2, 0).Value = StringConcat(" ", str1, _
"- Median")
```

```
Range("B2").Offset(i + 3, 0).Select
ActiveCell.FormulaR1C1 = "=STDEV(R[" & (-i - 3) & "]C:R[-3]C)"
Range("A2").Offset(i + 3, 0).Value = StringConcat(" ", str1, _
"- Standard Deviation")

Range(Range("B2").Offset(i + 1, 0), Range("B2").Offset(i + 3, _
0)).Select
Selection.Copy
Range(Range("B2").Offset(i + 1, 0), Range("B2").Offset(i + 3, _
35)).Select
ActiveSheet.Paste
Selection.Font.ColorIndex = 33

'Average, Median, and Standard Deviation of All Values For Each Protein
'in Category 2
Range("B2").Offset((i + 1) + m + (j + 1), 0).Select
ActiveCell.FormulaR1C1 = "=AVERAGE(R[" & (-j - 2) & "]C:R[-1]C)"
Range("A2").Offset((i + 1) + m + (j + 1), 0).Value = _
StringConcat(" ", _  str2, "- Average")

Range("B2").Offset((i + 1) + m + (j + 2), 0).Select
ActiveCell.FormulaR1C1 = "=MEDIAN(R[" & (-j - 3) & "]C:R[-2]C)"
Range("A2").Offset((i + 1) + m + (j + 2), 0).Value = _
StringConcat(" ", str2, "- Median")

Range("B2").Offset((i + 1) + m + (j + 3), 0).Select
ActiveCell.FormulaR1C1 = "=STDEV(R[" & (-j - 4) & "]C:R[-3]C)"
Range("A2").Offset((i + 1) + m + (j + 3), 0).Value = _
StringConcat(" ", str2, "- Standard Deviation")

Range(Range("B2").Offset((i + 1) + m + (j + 1), 0),
Range("B2").Offset((i + 1) + m + (j + 3), 0)).Select
Selection.Copy
Range(Range("B2").Offset((i + 1) + m + (j + 1), 0), _
Range("B2").Offset((i + 1) + m + (j + 3), 35)).Select
ActiveSheet.Paste
Selection.Font.ColorIndex = 33

'Graph Average Protein Values for Both Categories
Union(Range(Cells(1, 1), Cells(1, 37)), Range(Cells(i + 3, 1), _
Cells(i + 3, 37)), Range(Cells((i + 1) + m + (j + 3), 1), _
Cells((i + 1) + m + (j + 3), 37))).Select
ActiveSheet.Shapes.AddChart.Select
ActiveChart.SetSourceData Source:=Union(Range(Cells(1, 1), _
Cells(1, 37)), Range(Cells(i + 3, 1), Cells(i + 3, 37)), _
Range(Cells((i + 1) + m + (j + 3), 1), Cells((i + 1) + m + (j + 3), _
37))), PlotBy:=xlRows

With ActiveChart
   .SeriesCollection(1).ErrorBar Direction:=xlY, Include:=xlBoth, _
    Type:=xlCustom, Amount:=ActiveSheet.Range(Cells(i + 5, 2), _
    Cells(i + 5, 37)), MinusValues:=ActiveSheet.Range(Cells(i + 5, _
    2), Cells(i + 5, 37))
   .SeriesCollection(2).ErrorBar Direction:=xlY, Include:=xlBoth, _
```

```
    Type:=xlCustom, Amount:=ActiveSheet.Range(Cells((i + 5) + m + _
    (j + 1), 2), Cells((i + 5) + m + (j + 1), 37)), _
    MinusValues:=ActiveSheet.Range(Cells((i + 5) + m + (j + 1), 2), _
    Cells((i + 5) + m + (j + 1), 37))
End With

FormatActiveChart
ActiveChart.ChartTitle.Text = StringConcat(" ", Range("A1").Value, _
str1, "vs.", str2, "- Means")

'Graph Median Protein Values for Both Categories
Union(Range(Cells(1, 1), Cells(1, 37)), Range(Cells(i + 4, 1), _
Cells(i + 4, 37)), Range(Cells((i + 4) + m + (j + 1), 1), _
Cells((i + 4) + m + (j + 1), 37))).Select
ActiveSheet.Shapes.AddChart.Select
ActiveChart.SetSourceData Source:=Union(Range(Cells(1, 1), Cells(1, _
 37)), Range(Cells(i + 4, 1), Cells(i + 4, 37)), _
Range(Cells((i + 4) + m + (j + 1), 1), Cells((i + 4) + m + (j + 1), _
 37))), PlotBy:=xlRows

With ActiveChart
   .SeriesCollection(1).ErrorBar Direction:=xlY, Include:=xlBoth, _
    Type:=xlCustom, Amount:=ActiveSheet.Range(Cells(i + 5, 2), _
    Cells(i + 5, 37)), MinusValues:=ActiveSheet.Range(Cells(i + 5, _
    2), Cells(i + 5, 37))
   .SeriesCollection(2).ErrorBar Direction:=xlY, Include:=xlBoth, _
    Type:=xlCustom, Amount:=ActiveSheet.Range(Cells((i + 5) + m + _
    (j + 1), 2), Cells((i + 5) + m + (j + 1), 37)), _
    MinusValues:=ActiveSheet.Range(Cells((i + 5) + m + (j + 1), _
    2), Cells((i + 5) + m + (j + 1), 37))
End With

FormatActiveChart
ActiveChart.ChartTitle.Text = StringConcat(" ", Range("A1").Value, _
str1, "vs.", str2, "- Medians")
n = (i + 1) + m + (j + 5) 'The A2 Offset index for "Mean Difference"

'Mean Difference
Range("A2").Offset(n, 0).Value = "Mean Difference"
Range(Range("B2").Offset(n, 0), Range("B2").Offset(n, _
 35)).FormulaR1C1 = "=SUM(R[" & -5 - j - m & "]C, -R[-4]C)"
Range("A2").Offset(n + 3, 0).Value = "RMS of Means"
Range("B2").Offset(n + 3, 0).FormulaR1C1 = _
"=SQRT(SUMSQ(R[-3]C:R[-3]C[35])/COUNTA(R[-3]C:R[-3]C[35]))"

'Median Difference
Range("A2").Offset(n + 1, 0).Value = "Median Difference"
Range(Range("B2").Offset(n + 1, 0), Range("B2").Offset(n + 1, _
 35)).FormulaR1C1 = "=SUM(R[" & -5 - j - m & "]C, -R[-4]C)"
Range("A2").Offset(n + 4, 0).Value = "RMS of Medians"
Range("B2").Offset(n + 4, 0).FormulaR1C1 = _
"=SQRT(SUMSQ(R[-3]C:R[-3]C[35])/COUNTA(R[-3]C:R[-3]C[35]))"

'Mean Difference Absolute Value
```

```
Range("A2").Offset(n + 7, 0).Value = "Abs(Mean Difference)"
Range(Range("B2").Offset(n + 7, 0), Range("B2").Offset(n + 7, _
 35)).FormulaR1C1 = "=ABS(R[-7]C)"

'Median Difference Absolute Value
Range("A2").Offset(n + 8, 0).Value = "Abs(Median Difference)"
Range(Range("B2").Offset(n + 8, 0), Range("B2").Offset(n + 8, _
 35)).FormulaR1C1 = "=ABS(R[-7]C)"

'Sort Mean Difference Abs
Range(Range("B2").Offset(n + 7, 0), Range("B2").Offset(n + 7, _
 35)).Select
Selection.Copy

Range("B2").Offset(n + 10, 0).Select
Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, _
 SkipBlanks:=False, Transpose:=False

Range(Range("B2").Offset(n + 10, 0), Range("B2").Offset(n + 10, _
 35)).Select
ActiveSheet.Sort.SortFields.Clear
ActiveSheet.Sort.SortFields.Add Key:=Range(Range("B2").Offset(n + 10, _
 0), Range("B2").Offset(n + 10, 35)), SortOn:=xlSortOnValues, _
 Order:=xlAscending, DataOption:=xlSortNormal

With ActiveSheet.Sort
   .SetRange Range(Range("B2").Offset(n + 10, 0), _
    Range("B2").Offset(n + 10, 35))
   .Header = xlGuess
   .MatchCase = False
   .Orientation = xlLeftToRight
   .SortMethod = xlPinYin
   .Apply
End With

Range("A2").Offset(n + 10, 0).Value = "Sorted-Abs(Mean Difference)"

'Sort Median Difference Abs
Range(Range("B2").Offset(n + 8, 0), Range("B2").Offset(n + 8, _
 35)).Select


Selection.Copy
Range("B2").Offset(n + 11, 0).Select
Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, _
 SkipBlanks:=False, Transpose:=False

Range(Range("B2").Offset(n + 11, 0), Range("B2").Offset(n + 11, _
 35)).Select
ActiveSheet.Sort.SortFields.Clear
ActiveSheet.Sort.SortFields.Add Key:=Range(Range("B2").Offset(n + 11, _
 0), Range("B2").Offset(n + 11, 35)), SortOn:=xlSortOnValues, _
 Order:=xlAscending, DataOption:=xlSortNormal
With ActiveSheet.Sort
```

```
        .SetRange Range(Range("B2").Offset(n + 11, 0), _
         Range("B2").Offset(n + 11, 35))
        .Header = xlGuess
        .MatchCase = False
        .Orientation = xlLeftToRight
        .SortMethod = xlPinYin
        .Apply
    End With

    Range("A2").Offset(n + 11, 0).Value = "Sorted-Abs(Median Difference)"

End Sub
```

```
Sub PrepareNewSheetForStatistics()

'This subroutine creates a new excel workbook, saves it as
'"Diagnostic Performance" and formats it for running diagnostic
'tests. Columns of Actual and Predicted diagnoses are located in
'columns A and C respectively. A heading is created for each test
'number (the number of tests/runs was specificied earlier by the
'user). For each test, the excel case file in the corresponding
'case folder is automatically opened, and the "unknowns" (patient
'samples randomly assigned to be unknown test samples) are copied
'and pasted beneath the appropriate test heading in the "Diagnostic
'Performance" file. For each unknown, the first word (corresponding
'to the diagnosis) is extracted and placed in the adjacent cell in
'Column B. These cells are then hidden so that the tester cannot
'reference them (cheat) when assigning predicted diagnoses. A
'command button called "Diagnostic Performance!" is created,
'which runs the "CalculateStatistics" subroutine when clicked.

Dim i, m, Test As Integer
Dim RangeA, RangeB As Range
'RangeA and RangeB are First and Last Cell (respectively) of Each Test

   strNewFolderPathAndName = "C:\Documents and Settings\Heath _
    Group\Desktop\"
   'ActiveWorkbook.ActiveSheet.Activate
   Workbooks.Add
   FileName4Paste = strNewFolderPathAndName & _
   "Diagnostic Performance.xlsx"
   ActiveWorkbook.SaveAs FileName:=FileName4Paste, _
   FileFormat:=xlOpenXMLWorkbook, CreateBackup:=False

   Range("A1").Value = Category1Name & " vs."
   Range("A1").Font.Bold = True

   If StrComp(Category2Name, "no", vbTextCompare) = 0 Then
      Range("B1").Value = Category2Name & " " & Category1Name
   Else
   Range("B1").Value = Category2Name
   'Should have global variable for Cat2 Name
   End If
```

```
    Range("B1").Font.Bold = True

    Test = 1
    Set RangeA = Range("A1").Offset((NumUnknowns + 2) * (Test - 1) + 12, 0)

    With RangeA.Offset(-3, 0)
       .Value = "Actual"
       .Font.Bold = True
    End With

    With RangeA.Offset(-3, 2)
       .Value = "Predicted"
       .Font.Bold = True
    End With

    For Test = 1 To TestNumber

       Set RangeA = Range("A1").Offset((NumUnknowns + 2) * (Test - 1) + _
        12, 0)
       Set RangeB = RangeA.Offset((NumUnknowns - 1), 0)
       CurrentCasePathName = strNewFolderPathAndName & "Case" & Test & "\"
       FileName4Copy = CurrentCasePathName & "Case" & Test & ".xlsx"
       Workbooks.Open FileName:=FileName4Copy
       ActiveWorkbook.Sheets(1).Activate
       Range(Range("A2").Offset((NumRows - 1) - NumUnknowns, 0), _
       Range("A2").Offset(NumRows - 2, 0)).Select
       Selection.Copy
       ActiveWorkbook.Close
       Workbooks("Diagnostic Performance").Activate
       RangeA.Offset(-1, 0).Value = "Test" & Test
       RangeA.Offset(-1, 0).Font.Underline = True
       RangeA.Offset(-1, 2).Value = "Test" & Test
       RangeA.Offset(-1, 2).Font.Underline = True
       RangeA.Select
       ActiveSheet.Paste
       RangeA.Offset(0, 1).Select
       ActiveCell.FormulaR1C1 = "=LEFT(RC[-1],FIND("" - "",RC[-1])-1)"
       Selection.Copy
       Range(RangeA.Offset(0, 1), RangeB.Offset(0, 1)).Select
       ActiveSheet.Paste
       Range(RangeA, RangeB.Offset(0, 1)).NumberFormat = ";;;"

    Next Test

    Range("B:B").Copy
    Range("B:B").PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, _
     SkipBlanks:=False, Transpose:=False
    Workbooks("Diagnostic Performance").Activate
    Range("A3").Value = TestNumber & " Tests"
    Range("A4").Value = NumUnknowns & " Unknowns Each"

    Call GenButtons("Diagnostic Performance!", "CalculateStatistics")

End Sub
```

Subroutines Called by the Above Procedures

```
'The following subroutines are used directly by the Format4AnalyseIt
'subroutine: InsertColumns, ExtractFirstWord4AnalyseIt,
'ChangeCellFontColorAndPlaceColumnsAdjacently2, PaintColumnFontBlack
'GenButtons.

Sub InsertColumns()

'This subroutine inserts two blank columns between columns
'of protein intensity values.

   ActiveSheet.Range("B2").Select

   Do While Not IsEmpty(ActiveCell)
      Selection.EntireColumn.Select
      Selection.Insert Shift:=xlRight
      Selection.Insert Shift:=xlRight
      ActiveCell.Offset(0, 3).Select
   Loop

   ActiveSheet.Range("C2").Select

   Do While Not IsEmpty(ActiveCell.Offset(0, 1))
      Selection.EntireColumn.Select
      Selection.Font.ColorIndex = 0
      ActiveCell.Offset(0, 3).Select
   Loop

End Sub
```

```
Sub ExtractFirstWord4AnalyseIt()

'This subroutine extracts the first word of each cell
'in column A and places it in the adjacent cell in
'column B.

NumRows = 1

   Range("A2").Select


   Do While Not IsEmpty(ActiveCell)
      ActiveCell.Offset(1, 0).Select
      NumRows = NumRows + 1
   Loop

   Range("B2").Select
   ActiveCell.FormulaR1C1 = "=LEFT(RC[-1],FIND("" "",RC[-1])-1)"
   Range("B2").Select
   Selection.Copy
   Range(Range("B2"), Range("B2").Offset(NumRows - 2, 0)).Select
   ActiveSheet.Paste
```

```
End Sub
```

```
Sub ChangeCellFontColorAndPlaceColumnsAdjacently2()

'This subroutine color-codes all rows corresponding to one
'diagnosis (typically, experimental group) red, and color-codes
'all rows corresponding to the other diagnosis (typically,
'control group) green. It then calls a function that places
'the column of protein values for the control group (green)
'adjacent to the columns of protein values for the
'experimental group (red). This allows the values from both
'groups to be tabulated in the correct format to be copied into
'an AnalyseIt Add-in file in Excel for statistical analysis.

Dim Cat1, Cat2 As Variant
Dim i, j As Integer

i, j = 0

   Range("B2").Select
   Selection.EntireColumn.Select
   Selection.Copy
   Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, _
    SkipBlanks:=False, Transpose:=False
   Range("B2").Select

   Do While StrComp(ActiveSheet.Range("B2").Offset(i, 0).Value, _
      ActiveSheet.Range("B2").Offset(i + 1, 0).Value, vbTextCompare) = 0
      i = i + 1
   Loop

   Range(Range("B2"), Range("B2").Offset(i, 0)).EntireRow.Select
   Selection.Font.ColorIndex = 3
   Cat1 = Split(Range("B2").Offset(i, 0).Value, " ")
   Category1Name = Cat1(0)

   Do While StrComp(ActiveSheet.Range("B2").Offset(i + 1 + j, 0).Value, _
      ActiveSheet.Range("B2").Offset((i + 1) + (j + 1), 0).Value, _
      vbTextCompare) = 0
      j = j + 1
   Loop

   Range(Range("B2").Offset(i + 1, 0), Range("B2").Offset(j + i + 1, _
    0)).EntireRow.Select
   Selection.Font.ColorIndex = 4
   Cat2 = Split(Range("B2").Offset(j + i + 1, 0).Value, " ")
   Category2Name = Range("B2").Offset(j + i, 0).Value

   Call PlaceColumnsAdjacently(i, j)
   Range("D2").Select

End Sub
```

```
Sub PaintColumnFontBlack()

'This subroutine adjusts the color of empty columns
'(between green and red columns) to black.

   ActiveSheet.Range("C2").Select

   Do While Not IsEmpty(ActiveCell.Offset(0, 1))
      Selection.EntireColumn.Select
      Selection.Font.ColorIndex = 0
      ActiveCell.Offset(0, 3).Select
   Loop

End Sub
```

```
Sub GenButtons(ByVal strCaption As String, ByVal strAction As String)

'This function generates a command button by first
'receiving two string arguments. The first is the text
'that will appear on the command button. The second
'string argument is the subroutine the command button
'will run when clicked. The coordinates on the excel
'worksheet at which the command button is to be placed
'are also set.

Dim cBtn As Button

Set cBtn = ActiveSheet.Buttons.Add(0, 0, 175, 25)

cBtn.OnAction = strAction
cBtn.Caption = strCaption

End Sub
```

Subroutines Called by the Above Subroutines

```
'The following subroutine is called by the subroutine
'"ChangeCellFontColorAndPlaceColumnsAdjacently2".


Sub PlaceColumnsAdjacently(ByVal i As Integer, ByVal j As Integer)

'This subroutine places the column of protein intensity values for
'the control group (green)adjacent to the columns of protein values
'for the experimental group (red). This allows the values from both
'groups to be tabulated in the correct format to be copied into
'an AnalyseIt Add-in file in Excel for statistical analysis.

k = 0

   Range("D2").Select
   Do While Not IsEmpty(Range("D2").Offset(0, 3 * k))
      Range(Range("D2").Offset(i + 1, 3 * k), Range("D2").Offset(j+i+1, _
       3 * k)).Select
      Selection.Cut
      Range("D2").Offset(0, 3 * k + 1).Select
      ActiveSheet.Paste
      Range("D1").Offset(0, 3 * k).Select
      Selection.Copy
      ActiveCell.Offset(0, 1).Select
      ActiveSheet.Paste
      k = k + 1
   Loop

End Sub
```

```
'The following functions are used directly by the
'"SelectRandomCases" subroutine: NewSheetForEachUnknown
'and SaveToNotepad.

Sub NewSheetForEachUnknown(ByVal i As Integer, ByVal m As Integer)

'This function creates a separate worksheet containing
'the set of 'known' patient samples with each unknown,
'as well as one with all the unknowns combined (and labels
'each sheet as such). The arguments i and m are integers
'passed by the "SelectRandomCases" function. The integer
'i refers to the row number of the last patient sample
'on the worksheet. The integer m is a number one unit
'greater than the number of unknowns.

Dim j As Integer

   ActiveWorkbook.Worksheets(1).Select
   Range(Range("B1"), Range("B1").Offset(i - 1, NumProteins)).Select
   Selection.Copy
   Sheets.Add After:=Sheets(Sheets.Count)
   ActiveSheet.Paste
   ActiveSheet.Name = "AllUnknowns"

   For j = 1 To NumUnknowns

      ActiveWorkbook.Worksheets(1).Select
      Union(Range(Range("B1"), Range("B1").Offset(i - m, NumProteins)), _
       Range(Range("B1").Offset(i - m + j, 0), _
       Range("B1").Offset(i - m + j, NumProteins))).Select

      Selection.Copy
      Sheets.Add After:=Sheets(Sheets.Count)
      ActiveSheet.Paste
      ActiveSheet.Name = Range("A1").Offset(i - m + 1, 0).Value

   Next j

   Application.DisplayAlerts = False
   Sheets("Sheet2").Delete
   Sheets("Sheet3").Delete
   Application.DisplayAlerts = True

End Sub
```

```
Sub SaveToNotepad(ByVal i As Integer, ByVal FilePathName As String)

'This subroutine gets the path and file name of an excel workbook
'in which the first worksheet contains a set of 'known' patient
'samples with the full set of randomly assigned unknowns. Each
'subsequent worksheet contains the set of 'known' patient samples
'with each unknown individually. Each of the worksheets is
'saved as a text file (for use directly with Cluster 3.0) in
'both the "All Text Files" subfolder within the "NewTrialFolder"
'directory, and in the "Text Files" folder within a "Case"
'subfolder (also in the "NewTrialFolder" directory).

Dim strPath, strFileName, strPathAndFilename As String
Dim n As Integer

'Note: FilePathName = CurrentCasePathName
'CurrentCasePathName = strNewFolderPathAndName & "Case" & Test & "\"

   MkDir FilePathName & "Text Files"
   'This folder goes into the Test/Case Folder

   ActiveWorkbook.Worksheets(2).Select
   ActiveWorkbook.SaveAs FileName:=FilePathName & "Text Files\Test" & _
    CurrentTest & "_AllUnknowns.txt", FileFormat:=xlText, _
    CreateBackup:=False
   ActiveWorkbook.SaveAs FileName:=strNewFolderPathAndName & _
   "All Text Files\Test" & CurrentTest & "_AllUnknowns.txt", _
    FileFormat:=xlText, CreateBackup:=False

   For n = 1 To NumUnknowns

      ActiveWorkbook.Worksheets(n + 2).Select
      ActiveWorkbook.SaveAs FileName:=FilePathName & "Text Files\Test" & _
       CurrentTest & "_Unknown" & n & ".txt", _
      FileFormat:=xlText, CreateBackup:=False
      ActiveWorkbook.SaveAs FileName:=strNewFolderPathAndName & _
      "All Text Files\Test" & CurrentTest & "_Unknown" & n & ".txt", _
      FileFormat:=xlText, CreateBackup:=False

   Next n

End Sub
```

4.4.4 Assessing the Diagnostic Performance of "Guilt-by-Association" Classification of Test Samples within Hierarchical Clusters

```
Sub CalculateStatistics()

'This subroutine is activated when the "Diagnostic Performance!"
'command button is clicked in the "Diagnostic Performance" excel
'file. The subroutine compares the actual diagnoses within each
'test (column B) with the predicted diagnoses entered in by the
'tester (column C). If the predicted diagnosis is correct, a
'check mark is shown in the cell adjacent to the prediction
'(column D). Otherwise, a red x is shown, and an indication of
'whether the prediction was a false negative (FN) or false
'positive (FP) is given in column G. The samples within each
'test are numbered in column E.

'In addition, 2x2 contingency tables are drawn (with appropriate
'labels) for each test, indicating the numbers of true positives,
'true negatives, false positives, and false negatives. The sensitivity
'and specificity are given 2 columns to the right of the table,
'and the positive and negative predictive values are given
'two rows beneath the table. A contingency table indicating
'overall values (for all tests combined) is shown at the top.

Dim i, m, Test As Integer
Dim TruePositive, TrueNegative, FalsePositive, FalseNegative As Integer
Dim PPV, NPV, Sensitivity, Specificity As Double
Dim OverallTP, OverallTN, OverallFP, OverallFN As Integer
Dim OverallPPV, OverallNPV, OverallSensitivity, _
    OverallSpecificity As Double
Dim RangeAConst, RangeTable, OverallTable As Range
Dim strSplitCat1, strSplitCat2, strSplitActual, strSplitPredicted, tNum, _
    nUnk As Variant

OverallTP , OverallTN, OverallFP, OverallFN = 0
TruePositive , TrueNegative, FalsePositive, FalseNegative = 0
Sensitivity , Specificity, PPV, NPV = 0

   ActiveWorkbook.ActiveSheet.Activate

   Range("A:A").NumberFormat = "General"
   Range("D:D").Font.Name = "Wingdings"
   strSplitCat1 = Split(Range("A1").Value, " ")
   strSplitCat2 = Split(Range("B1").Value, " ")
   nUnk = Split(Range("A4").Value, " ")
   NumUnknowns = CInt(nUnk(0))
   tNum = Split(Range("A3"), " ")
   TestNumber = CInt(tNum(0))
   Test = 1

   Set RangeAConst = Range("A1").Offset((NumUnknowns + 2) * (Test - 1) _
    + 12, 0)
```

```
For Test = 1 To TestNumber

   i = 0
   TruePositive, TrueNegative, FalsePositive, FalseNegative = 0
   Sensitivity, Specificity, PPV, NPV = 0

   Do
      Set RangeA = Range("A1").Offset((NumUnknowns + 2) * (Test - 1) _
       + 12, 0)
      Set RangeB = RangeA.Offset((NumUnknowns - 1), 0)

      RangeA.Offset(i, 0).Select
      strSplitActual = Split(RangeA.Offset(i, 1).Value, " ")
      strSplitPredicted = Split(RangeA.Offset(i, 2).Value, " ")

      If RangeA.Offset(i, 1).Value <> "" And RangeA.Offset(i, _
       2).Value <> "" Then

         If StrComp(strSplitActual(0), strSplitPredicted(0), _
          vbTextCompare) = 0 Then
          RangeA.Offset(i, 3).Value = "ü"

            If StrComp(strSplitActual(0), strSplitCat1(0), _
             vbTextCompare) = 0 Then
               'RangeA.Offset(i, 6).Value = "TP"
                TruePositive = TruePositive + 1
                OverallTP = OverallTP + 1
            End If

            If StrComp(strSplitActual(0), strSplitCat2(0), _
             vbTextCompare) = 0 Then
               'RangeA.Offset(i, 6).Value = "TN"
               TrueNegative = TrueNegative + 1
               OverallTN = OverallTN + 1
            End If

         Else
            RangeA.Offset(i, 3).Value = "û"
            RangeA.Offset(i, 3).Font.ColorIndex = 3

            If StrComp(strSplitActual(0), strSplitCat1(0), _
               vbTextCompare) = 0 Then
               RangeA.Offset(i, 6).Value = "FN"
               FalseNegative = FalseNegative + 1
               OverallFN = OverallFN + 1
            End If

            If StrComp(strSplitActual(0), strSplitCat2(0), _
               vbTextCompare) = 0 Then
               RangeA.Offset(i, 6).Value = "FP"
               FalsePositive = FalsePositive + 1
               OverallFP = OverallFP + 1
            End If
      End If

   End If

   RangeA.Offset(i, 4).Value = i + 1
   i = i + 1

Loop Until IsEmpty(ActiveCell.Offset(1, 0))

If TruePositive + FalsePositive <> 0 Then
   PPV = Round((TruePositive / (TruePositive + _
   FalsePositive)) * 100, 1)
End If

If TrueNegative + FalseNegative <> 0 Then
   NPV = Round((TrueNegative / (TrueNegative + _
   FalseNegative)) * 100, 1)
End If

If TruePositive + FalseNegative <> 0 Then
   Sensitivity = Round((TruePositive / (TruePositive + _
   FalseNegative)) * 100, 1)
End If

If TrueNegative + FalsePositive <> 0 Then
   Specificity = Round((TrueNegative / (TrueNegative + _
   FalsePositive)) * 100, 1)
End If

Set RangeTable = RangeA.Offset(0, 7)

With RangeTable

   .Offset(0, 1) = "Positive"
   .Offset(0, 2) = "Negative"
   .Offset(1, 0) = Range("A1").Value
   .Offset(2, 0) = Range("B1").Value
   .Offset(1, 1) = "TP = " & TruePositive
   .Offset(2, 2) = "TN = " & TrueNegative
   .Offset(2, 1) = "FP = " & FalsePositive
   .Offset(1, 2) = "FN = " & FalseNegative
   .Offset(4, 1) = "PPV = " & PPV & "%"
   .Offset(4, 2) = "NPV = " & NPV & "%"
   .Offset(1, 4) = "Sensitivity = " & Sensitivity & "%"
   .Offset(2, 4) = "Specificity = " & Specificity & "%"
   .Offset(0, 7) = Sensitivity
   .Offset(0, 8) = Specificity
   .Offset(0, 9) = PPV
   .Offset(0, 10) = NPV

End With

RangeTable.Offset(1, 1).BorderAround ColorIndex:=0, Weight:=xlThin
```

```
      RangeTable.Offset(1, 2).BorderAround ColorIndex:=0, Weight:=xlThin
      RangeTable.Offset(2, 1).BorderAround ColorIndex:=0, Weight:=xlThin
      RangeTable.Offset(2, 2).BorderAround ColorIndex:=0, Weight:=xlThin

   Next Test

   If OverallTP + OverallFP <> 0 Then
      OverallPPV = Round((OverallTP / (OverallTP + OverallFP)) * 100, 1)
   End If

   If OverallTN + OverallFN <> 0 Then
      OverallNPV = Round((OverallTN / (OverallTN + OverallFN)) * 100, 1)
   End If

   If OverallTP + OverallFN <> 0 Then
      OverallSensitivity = Round((OverallTP / (OverallTP + _
      OverallFN)) * 100, 1)
   End If

   If OverallTN + OverallFP <> 0 Then
      OverallSpecificity = Round((OverallTN / (OverallTN + _
      OverallFP)) * 100, 1)
   End If

   Set OverallTable = RangeAConst.Offset(-10, 7)

   OverallTable.Select

   With OverallTable
      .Offset(-1, 1) = "Overall"
      .Offset(-1, 2) = "Overall"
      .Offset(0, 1) = "Positive"
      .Offset(0, 2) = "Negative"
      .Offset(1, 0) = Range("A1").Value
      .Offset(2, 0) = Range("B1").Value
      .Offset(1, 1) = "TP = " & OverallTP
      .Offset(2, 2) = "TN = " & OverallTN
      .Offset(2, 1) = "FP = " & OverallFP
      .Offset(1, 2) = "FN = " & OverallFN
      .Offset(4, 1) = "PPV = " & OverallPPV & "%"
      .Offset(4, 2) = "NPV = " & OverallNPV & "%"
      .Offset(1, 4) = "Sensitivity = " & OverallSensitivity & "%"
      .Offset(2, 4) = "Specificity = " & OverallSpecificity & "%"
   End With

   Range(OverallTable.Offset(-1, 0), OverallTable.Offset(4, _
    4)).Font.ColorIndex = 3
   OverallTable.Offset(1, 1).BorderAround ColorIndex:=3, Weight:=xlThick
   OverallTable.Offset(1, 2).BorderAround ColorIndex:=3, Weight:=xlThick
   OverallTable.Offset(2, 1).BorderAround ColorIndex:=3, Weight:=xlThick
   OverallTable.Offset(2, 2).BorderAround ColorIndex:=3, Weight:=xlThick

End Sub
```

4.4.5 Macros for Working with *AnalyseIt*

```
'The following set of macros facilitates straightforward transfer of cohort
'data (experimental and control data for each protein) into a pre-defined
'table format within an AnalyseIt template. They also facilitate the
'transfer of AnalyseIt graphs into Powerpoint.

Public Category1Name, Category2Name As String

'These are provided by the ChangeCellFontColorAndPlaceColumnsAdjacently
subroutine

Public NumRows As Integer
```

```
Sub OpenAnalyseItDataSetDefined()

'This subroutine opens an Excel "AnalyseIt" workbook
'in which a table has been created containing two
'column headers: category 1 (experimental group) and
'category 2 (control group). The number and types
'of variables (i.e. categorical), and the type of
'dataset (list, one-way, or two-way table) have all been
'pre-defined to facilitate ease of use with AnalyseIt.

Dim PathName, FileName, FilePathAndName As String
Dim wBook As Workbook

'Set wBook = Workbooks("AnalyseIt-DatasetDefined")

   ActiveWorkbook.ActiveSheet.Activate
   PathName = ActiveWorkbook.Path
   FileName = ActiveWorkbook.Name
   FilePathAndName = PathName & "\" & FileName
   'If wBook Is Nothing Then
   Workbooks.Open FileName:="C:\Documents and Settings\Heath
   Group\Desktop\AnalyseIt-DatasetDefined.xlsm"
    'End If
   Workbooks("AnalyseIt-DatasetDefined").Sheets("Dataset").Activate
   Range("E4").Value = FilePathAndName

End Sub
```

```
Sub TransferNext2AnalyseIt()

Dim varFileName As Variant
Dim strFileName As String
Dim myString As String

On Error Resume Next

   Workbooks("AnalyseIt-DatasetDefined").Activate
   varFileName = Split(Range("E4").Value, "\")
   strFileName = varFileName(UBound(varFileName))
   Workbooks(strFileName).Activate


   Do While ActiveCell.Font.ColorIndex <> 3
      ActiveCell.Offset(0, 1).Select
   Loop

   Do While ActiveCell.Font.ColorIndex = 3
      ActiveCell.Offset(-1, 0).Select
   Loop

   ActiveCell.Offset(1, 0).Select
   ActiveCell.Offset(0, 3).Select
   ExtractStringAfterDash (strFileName)
   myString = ActiveCell.Offset(-1, 2).Value
   Range(ActiveCell, ActiveCell.Offset(146, 1)).Select
   Selection.Copy
   Workbooks("AnalyseIt-DatasetDefined").Activate
   ActiveSheet.Range("B6").Select
   ActiveSheet.Paste
   ActiveSheet.Range("B3").Value = StringConcat(" ", myString, "Levels
    for", Range("B5").Value, "vs.", Range("C5").Value)

End Sub
```

```
Sub TransferPrevious2AnalyseIt()

Dim varFileName As Variant
Dim strFileName As String
Dim myString As String

On Error Resume Next

   Workbooks("AnalyseIt-DatasetDefined").Activate
   varFileName = Split(Range("E4").Value, "\")
   strFileName = varFileName(UBound(varFileName))
   Workbooks(strFileName).Activate

   Do While ActiveCell.Font.ColorIndex <> 3
      ActiveCell.Offset(0, 1).Select
   Loop

   Do While ActiveCell.Font.ColorIndex = 3
      ActiveCell.Offset(-1, 0).Select
   Loop

   ActiveCell.Offset(1, 0).Select
   ActiveCell.Offset(0, -3).Select
   ExtractStringAfterDash (strFileName)
   myString = ActiveCell.Offset(-1, 2).Value
   Range(ActiveCell, ActiveCell.Offset(146, 1)).Select
   Selection.Copy
   Workbooks("AnalyseIt-DatasetDefined").Activate
   ActiveSheet.Range("B6").Select
   ActiveSheet.Paste
   ActiveSheet.Range("B3").Value = StringConcat(" ", myString, "Levels
    for", Range("B5").Value, "vs.", Range("C5").Value)

End Sub
```

```
Sub TransferAnalyseItGraphsToPowerpoint()

Dim i As Integer
Dim ppt, pres, NewSlide As Object
Dim s As PowerPoint.Slide
Dim shp As PowerPoint.Shape
Dim ws As Worksheet

Set ppt = CreateObject("powerpoint.application")
Set pres = ppt.Presentations.Add

i = 1

   For Each ws In ActiveWorkbook.Worksheets
      ws.Select
      PrintTheScreen
      Set NewSlide = pres.Slides.Add(i, ppLayoutBlank)
      NewSlide.Shapes.Paste
      i = i + 1
   Next ws

   ppt.Visible = True

End Sub
```

4.4.6 User Interface Macros

```
Private Sub OkayButton_Click()

'This subroutine allows the user to input values into the
'"ClusterPrep" user form for the number of proteins to be
'analyzed, the number of samples to be randomly assigned
'as unknowns (test samples), and the number of runs desired.
'These integer values are then assigned to the global
'variables "NumProteins","NumUnknowns, and "TestNumber" for
'use in the "RunClusterPrep" subroutine. The user also
'specifies the directory into which the statistical analysis
'files generated will be placed. If any of the fields in the
'user form remain unfilled, a message box prompts the user
'to fill in that field. Upon clicking "Okay", the
'"RunClusterPrep" subroutine gets underway.

Dim iRow As Long
Dim ws As Worksheet
Dim str As String
```

```
   If Trim(Me.ProteinTextBox.Value) = "" Then
      Me.ProteinTextBox.SetFocus
      MsgBox "Enter number of proteins"
      Exit Sub
   End If

   If Trim(Me.UnknownCasesTextBox.Value) = "" Then
      Me.UnknownCasesTextBox.SetFocus
      MsgBox "Enter the number of unknowns"
      Exit Sub
   End If

   If Trim(Me.TestsTextBox.Value) = "" Then
      Me.TestsTextBox.SetFocus
      MsgBox "Enter number of tests"
      Exit Sub
   End If

   If Trim(Me.DirectoryTextBox.Value) = "" Then
      Me.DirectoryTextBox.SetFocus
      MsgBox "Please choose a directory"
      Exit Sub
   End If

   'copy the data to the database
   NumProteins = Me.ProteinTextBox.Value
   NumUnknowns = Me.UnknownCasesTextBox.Value
   TestNumber = Me.TestsTextBox.Value
   'strDirectoryPathName = Me.DirectoryTextBox.Value

   RunClusterPrep

End Sub
```

```
Sub FolderSelection()

'This subroutine assigns the folder path and name chosen
'by the user via the SelectFolder function to a string.
'It then displays a message box containing that string.
'If no folder was chosen, it displays the message
'"Cancel was pressed".

   strFolderPathAndName = SelectFolder("Select Folder", "")

   If Len(strFolderPathAndName) Then
      MsgBox strFolderPathAndName
   Else
      MsgBox "Cancel was pressed"
   End If

End Sub
```

```
Function SelectFolder(Optional Title As String, Optional TopFolder _
                         As String) As String

'This function opens up a hierarchical menu of directories
'such that the user can choose a folder (in which to save files,
'for example). The function uses two optional arguments. The first
'is the dialog caption and the second is is to specify the top-most
'visible folder in the hierarchy. The default is "My Computer."

Dim objShell As New Shell32.Shell
Dim objFolder As Shell32.Folder

'If you use 16384 instead of 1 on the next line,
'files are also displayed

Set objFolder = objShell.BrowseForFolder(0, Title, 1, TopFolder)
   If Not objFolder Is Nothing Then
      SelectFolder = objFolder.Items.Item.Path
   End If

End Function
```

```
Private Sub ChooseDirectory_Click()

'Upon clicking the "Choose Directory" button on the user
'form, this subroutine runs the FolderSelection subroutine,
'which allows the user to select the directory into which
'their files are to be saved. A string containing the file
'path and name then fills the directory field in the user
'form.

   FolderSelection
   Me.DirectoryTextBox.Value = strFolderPathAndName

End Sub
```

```
Private Sub CloseButton_Click()

'Upon clicking the "Close" button, this subroutine deletes
'all values from the user form.

   Unload Me

End Sub
```

```
Private Sub ClusterPrep_QueryClose(Cancel As Integer, _
  CloseMode As Integer)

   If CloseMode = vbFormControlMenu Then
      Cancel = True
      MsgBox "Please use the button!"
   End If

End Sub
```

4.4.7 String Manipulations

```
Sub ExtractFirstWord()

   Range("BE2").Select
   ActiveCell.FormulaR1C1 = _
    "=IF(LEN(RC[-49])=0,"""",IF(ISERR(FIND("" "",RC[-53])),RC[53],_
      LEFT(RC[-53],FIND("" "",RC[-53])-1)))"
   Range("BE2").Select
   Selection.Copy
   Range("BE2:BE500").Select
   ActiveSheet.Paste

End Sub
```

```
Sub ExtractStringAfterDash(ByVal strAfterDash As String)

    Workbooks(strAfterDash).ActiveSheet.Activate
    ActiveCell.Offset(-1, 2).FormulaR1C1 = "=Mid(RC[-2],FIND(""-"", _
    RC[-2])+1,20)"

End Sub
```

```
Sub ExtractWordsBeforeDash()

Dim m As Integer

m = 1

    Range("A2").Select

    Do While Not IsEmpty(ActiveCell)

        ActiveCell.Offset(1, 0).Select
        m = m + 1

    Loop

    Range("B2").Select
    ActiveCell.FormulaR1C1 = "=LEFT(RC[-1],FIND("" - "",RC[-1])-1)"

    'Or Extract Words Before Space
    'ActiveCell.FormulaR1C1 = "=LEFT(RC[-1],FIND("" "",RC[-1])-1)"
     Range("B2").Select

    Selection.Copy
    Range(Range("B2"), Range("B2").Offset(m - 2, 0)).Select
    ActiveSheet.Paste

End Sub
```

```
Sub ConvertDateToString()
Dim i As Integer

   ActiveWorkbook.ActiveSheet.Activate

   Do While Not IsEmpty(ActiveCell)
      ActiveCell.Value = "'" & CStr (ActiveCell.Value)
      ActiveCell.Offset(1, 0).Select
   Loop

End Sub
```

```
Function StringConcat(Sep As String, ParamArray Args()) As String
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' StringConcat
' This function concatenates all the elements in the Args array,
' delimited by the Sep character, into a single string. This function
' can be used in an array formula.
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Dim s As String
Dim n,m,NumDims,LB,RN,CN As Long
Dim R As Range
Dim IsArrayAlloc As Boolean

   '''''''''''''''''''''''''''''''''''''''''''''
   ' If no parameters were passed in, return
   ' vbNullString.
   '''''''''''''''''''''''''''''''''''''''''''''
   If UBound(Args) - LBound(Args) + 1 = 0 Then
      StringConcat = vbNullString
      Exit Function
   End If


   For n = LBound(Args) To UBound(Args)
      ''''''''''''''''''''''''''''''''''''''''''''''''
      ' Loop through the Args
      ''''''''''''''''''''''''''''''''''''''''''''''''
      If IsObject(Args(n)) = True Then
         ''''''''''''''''''''''''''''''''''''''
         ' OBJECT
         ' If we have an object, ensure it
         ' it a Range. The Range object
         ' is the only type of object we'll
         ' work with. Anything else causes
         ' a #VALUE error.
         ''''''''''''''''''''''''''''''''''''''
         If TypeOf Args(n) Is Excel.Range Then
            ''''''''''''''''''''''''''''''''''''''''''
            ' If it is a Range, loop through the
            ' cells and create append the elements
      ' to the string S.
      '''''''''''''''''''''''''''''''''''''''''
      For Each R In Args(n).Cells
         s = s & R.Text & Sep
      Next R

   Else
      ''''''''''''''''''''''''''''''''''
      ' Unsupported object type. Return
      ' a #VALUE error.
      ''''''''''''''''''''''''''''''''''
      StringConcat = CVErr(xlErrValue)
      Exit Function
   End If

Else If IsArray(Args(n)) = True Then

   On Error Resume Next
   '''''''''''''''''''''''''''''''''''''
   ' ARRAY
   ' If Args(N) is an array, ensure it
   ' is an allocated array.
   '''''''''''''''''''''''''''''''''''''
   IsArrayAlloc = (Not IsError(LBound(Args(n))) And _
   (LBound(Args(n)) <= UBound(Args(n))))

   On Error GoTo 0

   If IsArrayAlloc = True Then
      '''''''''''''''''''''''''''''''''''''
      ' The array is allocated. Determine
      ' the number of dimensions of the
      ' array.
      ''''''''''''''''''''''''''''''''''''''
      NumDims = 1

      On Error Resume Next

      Err.Clear
      NumDims = 1

      Do Until Err.Number <> 0

         LB = LBound(Args(n), NumDims)

         If Err.Number = 0 Then
            NumDims = NumDims + 1
         Else
            NumDims = NumDims - 1
         End If

      Loop
      ''''''''''''''''''''''''''''''''''
      ' The array must have either
```

```
            ' one or two dimensions. Greater
            ' that two caues a #VALUE error.
            ''''''''''''''''''''''''''''''''''''
            If NumDims > 2 Then
               StringConcat = CVErr(xlErrValue)
               Exit Function
             End If

            If NumDims = 1 Then

               For m = LBound(Args(n)) To UBound(Args(n))
                  If Args(n)(m) <> vbNullString Then
                     s = s & Args(n)(m) & Sep
                  End If
               Next m

            Else

               For RN = LBound(Args(n), 1) To UBound(Args(n), 1)
                  For CN = LBound(Args(n), 2) To UBound(Args(n), 2)
                     s = s & Args(n)(RN, CN) & Sep
                  Next CN
               Next RN
            End If

         Else
            s = s & Args(n) & Sep
         End If

      Else
         s = s & Args(n) & Sep
      End If

   Next n

   '''''''''''''''''''''''''''''''''''''
   ' Remove the trailing Sep character
   '''''''''''''''''''''''''''''''''''''
   If Len(Sep) > 0 Then
      s = Left(s, Len(s) - Len(Sep))
   End If

   StringConcat = s

End Function
```

4.4.8 Other Useful Macros

```
Sub AddRunAnalysisCommandBarButton()

'This subroutine adds a button called "Run Analysis" to the
'Excel Add-Ins command bar which, when clicked, opens the
```

```
'user form and runs the ClusterPrep analysis on the active
'Excel Worksheet.

Dim AddBttn As CommandBarButton

Set AddBttn = CommandBars("Standard").Controls.Add

   With AddBttn
      .Caption = "Run Analysis"
      .OnAction = "ClusterPrep.Show"
      .Style = msoButtonCaption
   End With

End Sub
```

```
Sub TransferAllGraphsOnSheetsToPowerpoint()

'This subroutine transfers all graphs on a worksheet to a
'powerpoint file. It repeats this for all sheets in the workbook.

Dim ppt, pres, NewSlide As Object
Dim i As Integer

i = 1
Set ppt = CreateObject("powerpoint.application")
ppt.Visible = True
Set pres = ppt.Presentations.Add

   For Each ws In Worksheets

      ws.Activate

      For j = 1 To ActiveSheet.ChartObjects.Count

         ActiveSheet.ChartObjects(j).Select
         ActiveSheet.ChartObjects(j).Copy
         Set NewSlide = pres.Slides.Add(i, ppLayoutBlank)
         ActiveChart.CopyPicture Appearance:=xlScreen, Size:=xlScreen, _
         Format:=xlPicture
         NewSlide.Select
         NewSlide.Shapes.Paste.Select
         ppt.ActiveWindow.Selection.ShapeRange.ScaleWidth 1.1, msoFalse, _
         msoScaleFromBottomRight
         ppt.ActiveWindow.Selection.ShapeRange.ScaleHeight 1.1, _
         msoFalse, msoScaleFromBottomRight
         ppt.ActiveWindow.Selection.ShapeRange.Align msoAlignCenters, True
         ppt.ActiveWindow.Selection.ShapeRange.Align msoAlignMiddles, True
         ppt.ActiveWindow.Selection.SlideRange.Shapes(1).Select
         i = i + 1

      Next j

   Next ws
```

```
End Sub
```

```
Sub ImportABunch()

'This subroutine copies and pastes each .png file within the
'directory specified (in this case, the Desktop) into
'a separate slide in powerpoint.

Dim strTemp, strPath,strFileSpec As String
Dim Sld As Slide
Dim Pic, TextShape As Shape
Dim ShpRange As ShapeRange

   ' Edit these to suit:
   strPath = "C:\Documents and Settings\Heath Group\Desktop\"
   strFileSpec = "*.png"

   strTemp = Dir(strPath & strFileSpec)

   Do While strTemp <> ""
      ActiveWorkbook.ActiveSheet.Activate
      'Range("A1").Value = strTemp
      'Range("A2").Value = InStr(1, strTemp, "corr", vbTextCompare)
      strTemp = Dir
   Loop

End Sub
```

```
Sub FormatActiveChart()

Dim k As Integer
Dim x As Object

   With ActiveChart
      .ChartType = xlXYScatter
      .SetElement (msoElementChartTitleAboveChart)
      .PlotArea.Width = 330
      .HasLegend = True
      .Legend.Position = xlLegendPositionTop
      .Legend.Left = 90
      .Legend.Top = 20

      With .ChartTitle
         .Font.Size = 10
         .Font.Name = "Calibri (Body)"
      End With

      With .Axes(xlCategory)
         .MinorUnit = 1
         .MajorUnit = 37
         .MaximumScale = 37
         .MinorTickMark = xlTickMarkInside
      End With
      With .Axes(xlValue)
         .MinorUnit = 100
         .MajorUnit = 500
         .MinimumScale = -500
         .MaximumScale = 1500
      End With

   End With

   k = 1

   For Each x In ActiveChart.SeriesCollection

      x.MarkerSize = 4

      With ActiveChart.SeriesCollection(k)
         .MarkerForegroundColorIndex = 2 + k
         .MarkerBackgroundColorIndex = 2 + k
         .ErrorBars.Border.ColorIndex = 2 + k
      End With

         k = k + 1
   Next x

End Sub
```

```
Sub DeleteAllChartsOnEachSheet()

'This subroutine deletes all charts on all sheets
'of the active workbook

   For i = 1 To ActiveWorkbook.Worksheets.Count

      ActiveWorkbook.Worksheets(i).Select
      DeleteAllChartsOnSheet

   Next

End Sub
```

```
Sub DeleteAllChartsOnSheet()

'This subroutine deletes all charts on the active worksheet

Dim myshape As Shape

   For Each myshape In ActiveSheet.Shapes
      myshape.Delete
   Next myshape

End Sub
```

```
Sub PrintTheScreen()

   Application.SendKeys "(%{1068})"
   'Application.SendKeys "{1068}"
   DoEvents

End Sub
```

4.4.9 Batch Files for Running Cluster 3.0 and Java Treeview

Recall that among the "ClusterPrep" output files are a set of text files (typically 10) that contain data from all patients in a cohort as well, including a set of randomly assigned test samples. Manually opening and processing each of these files in Cluster 3.0 can be a very time-consuming process. Moreover, one may want to create .cdt files (Cluster files) with a number of different normalization, centering, and clustering permutations. To automate this process, we wrote a batch file "clusterstuff.bat" to allow all the text files created by the "ClusterPrep" software to be automatically processed by Cluster 3.0. This batch file instructs cluster to produce 8 (2 sets of 4) different .cdt files. The first set utilizes a centered Pearson correlation, whereas the second set utilizes an uncentered correlation. Within each set, the following normalization methods are employed: 1. No normalization, 2. Proteins normalized for each patient sample, 3. Patient samples normalized for each protein, and 4. Both proteins and patient samples normalized. Typically, only the .cdt files produced using method 4 were used. Note that Cluster 3.0 gives the option of normalizing by genes and arrays rather by proteins and patient samples. This is because the program is typically used for cluster analysis of gene expression microarrays. However, these designations (i.e. genes vs. proteins) are interchangeable. The batch file, stored in the Cluster 3.0 folder (C:\Program Files\Stanford University\Cluster 3.0), is shown below:

```
@echo off

set filename=%1

set namer=%filename:~0,-4%
set namer=%namer%_CorrUncentered.txt
type %1 > %namer%
cluster -f %namer% -g 1 -e 1 -m a
del %namer%

set namer=%filename:~0,-4%
set namer=%namer%_CorrCentered.txt
type %1 > %namer%
cluster -f %namer% -g 2 -e 2 -m a
del %namer%

set namer=%filename:~0,-4%
set namer=%namer%_NormalizedGenes_CorrUncentered.txt
type %1 > %namer%
```

```
cluster -f %namer% -ng -g 1 -e 1 -m a
del %namer%

set namer=%filename:~0,-4%
set namer=%namer%_NormalizedGenes_CorrCentered.txt
type %1 > %namer%
cluster -f %namer% -ng -g 2 -e 2 -m a
del %namer%

set namer=%filename:~0,-4%
set namer=%namer%_NormalizedArray_CorrUncentered.txt
type %1 > %namer%
cluster -f %namer% -na -g 1 -e 1 -m a
del %namer%

set namer=%filename:~0,-4%
set namer=%namer%_NormalizedArrays_CorrCentered.txt
type %1 > %namer%
cluster -f %namer%  -na -g 2 -e 2 -m a
del %namer%

set namer=%filename:~0,-4%
set namer=%namer%_NormalizedGeneArrayCorrUncentered.txt
type %1 > %namer%
cluster -f %namer% -ng -na -g 1 -e 1 -m a
del %namer%

set namer=%filename:~0,-4%
set namer=%namer%_NormalizedGeneArray_CorrCentered.txt
type %1 > %namer%
cluster -f %namer% -na -ng -g 2 -e 2 -m a
del %namer%
```

This batch file is executed when the file “analysis.bat” is clicked. The latter file is placed in the directory containing the text files that are to be analyzed by Cluster 3.0. The “analysis.bat” file contains the following set of instructions:

```
@echo off

set path=%path%;C:\Program Files\Stanford University\Cluster 3.0;
for /f %%a in ('dir /b *.txt') do clusterstuff.bat %%a
```