# Table of Contents for Introduction to ROC5 (Version 5.07)

[Table of Contents for Introduction to ROC5 (Version 5.07) 1](#_Toc29083416)

[1. Introduction to ROC5 2](#_Toc29083417)

[1.1 Files included with download 2](#_Toc29083418)

[1.2 What is this Program Good for? 3](#_Toc29083419)

[1.2.1 Producing a “Decision Tree” 4](#_Toc29083420)

[1.2.2 Weighing the Importance of False Positives versus False Negatives 5](#_Toc29083421)

[1.3 Who Owns this Program? 5](#_Toc29083422)

[1.4 Where Does the Theory Behind the Program Come From? 5](#_Toc29083423)

[2. Overview of Programming Strategy 6](#_Toc29083424)

[3. Data Preparation 7](#_Toc29083425)

[3.1 The Gold Standard versus Predictors 7](#_Toc29083426)

[3.2 Details of Data Preparation 7](#_Toc29083427)

[3.2.1 Missing data 7](#_Toc29083428)

[3.2.2 ID Numbers, Character Variables 7](#_Toc29083429)

[3.2.3 Note on Data Recoding 7](#_Toc29083430)

[3.2.4 Note on Variable Names 7](#_Toc29083430)

[3.2.5 Note on Number of Decimal Places 7](#_Toc29083430)

[4. Running the ROC Program 8](#_Toc29083431)

[4.1 How do you Run the Program? 8](#_Toc29083432)

[4.1.1 Batch Files Basics 8](#_Toc29083433)

[4.1.2 Batch Files Quirks 9](#_Toc29083434)

[4.2 What does the ROC Output Mean? (and how to read it) 9](#_Toc29083435)

[4.3 How to Change Emphasis on Sensitivity versus Specificity 10](#_Toc29083436)

[4.4 How to Get Results for Plots, i.e. ROC Curves 10](#_Toc29083437)

[4.4.1 How to Actually Get a ROC Plot out of the Data 10](#_Toc29083438)

[4.4.2 ROC Plots in Excel and SAS 11](#_Toc29083439)

[5. Run it Again Sam? More on Decision Trees 11](#_Toc29083440)

[6. FAQ (Frequently Asked Questions) 11](#_Toc29083441)

[Appendix 1: Note on Memory Allocation and Run Time 13](#_Toc29083442)

[Appendix 2: Note on Data Recoding 14](#_Toc29083443)

[IF 14](#_Toc29083444)

[Even more important is the operator AND in Excel: 16](#_Toc29083445)

[Even more important is the operator = in Excel: 16](#_Toc29083446)

[Appendix 3: Formulae 17](#_Toc29083447)

[Appendix 4: Example SAS Program for Graphics 18](#_Toc29083448)

# 1. Introduction to ROC5

This READ\_ME is designed to cover all aspects of our program, which is designed to perform a number of signal detection functions.

## Files included with download

## (located at http://web.stanford.edu/~yesavage/ROC.html)

Download the file ROC\_507.ZIP. It can be unzipped by programs such as WinZip. The .ZIP file contains the following:

File Descriptions:

READ\_ME.yymmdd.doc An explanation of all this (what you are reading)

ROC5.07\_xxxxxxx.docx

A word file of the actual C++ code. Change the .docx (with a .docx MS Word extension) to .c for a C compiler

Demo.txt Demo dataset

ROC\_5.07\_wnn.exe The current version of the ROC program

nn=32 for 32-bit PCs and nn=64 for 64-bit PCs

To determine whether your computer is 32-bit or 64-bit: Select the Windows Button 🡪 Right-click ”Computer” 🡪 Left-click “Properties”. Under “System 🡪 System type” will tell whether your computer is 32 bit or 64 bit

rDemoData.wnn.ppp.bat The batch file that does all the housekeeping and runs the program on the right dataset with the right settings.

nn=32 for 32-bit PCs and nn=64 for 64-bit PCs

ppp=05 uses p<.05 criteria. p<.01 and p<.001 also available

rDemoData.wnn.ppp.doc The text file (with a .docx MS Word extension) that contains the ROC output

nn=32 for 32-bit PCs and nn=64 for 64-bit PCs

ppp=05 uses p<.05 criteria. p<.01 and p<.001 also available

ROC\_Graph\_Excel.xlsx MS Excel file with sample ROC graph

## 1.2 What is this Program Good for?

This program is designed to help a clinician/researcher with a PC to evaluate clinical databases and discover the characteristics of subjects that best predict a binary outcome. That outcome may be any binary outcome such as:

* Whether or not the patient has a certain disorder (medical test evaluation)
* Whether or not the patient is likely to develop a certain disorder (risk factor evaluation)
* Whether or not the patient is likely to respond to a certain treatment (evaluation of treatment moderators)

When the predictors considered are themselves all binary (e.g., male/female; inpatient/outpatient; symptoms present/absent), the program identifies the optimal predictor. When one or more of the predictors are ordinal or continuous (e.g., age, severity of symptoms) it identifies the optimal cut-point for each of the ordinal or continuous predictors. It also determines the overall “best” predictor and cut-point.

### 1.2.1 Producing a “Decision Tree”

The program runs on different subsets of the same dataset, thus producing a "decision tree", which combines various predictors with "and/or" rules to best predict the binary outcome. The “bottom line” of the output is a “Decision Tree”. This is a schematized example from a hypothetical study predicting conversion to Alzheimer’s Disease using age and the Mini-Mental State Exam (MMSE) as potential predictors:

All Subjects (20% Convert)

#### Age at 75

Age < 75 Age >= 75

30% Convert

10% Convert

#### MMSE at 27

MMSE < 27 MMSE >= 27

40% Convert

20% Convert

In this example, subjects who are less than 75 years old have a 10% conversion rate. Those who are at least 75 AND have an MMSE score less than 27 have a 20% conversion rate. Finally, subjects older than 75 AND have an MMSE score of at least 27 have a 40% conversion rate. These cut-points are significant at the p=.05, .01 or .001 level, depending on which batch file is used.

### 1.2.2 Weighing the Importance of False Positives versus False Negatives

This program (a type of recursive partitioning) differs from other programs which creates trees (such as CART) in that the criterion for splitting is based on a CLINICAL judgment of the relative clinical or policy importance of false positive versus false negative identifications via weights called r. The program automatically considers three possibilities:

* Optimal Sensitivity: Here r=1, and the total emphasis is placed on avoiding false negatives. This would be appropriate, for example, for self-examination for breast or testicular lumps.
* Optimal Efficiency: Here r=1/2, and equal emphasis is placed on both types of errors. This would be appropriate, for example, for mammography.
* Optimal Specificity: Here r=0, and total emphasis is placed on avoiding false positives. This would be appropriate, for example, for frozen tissue biopsy done during breast surgery to decide on whether or not a mastectomy should be done.

When the user does not have reason to favor either false positives or false negatives, use of r=1/2 is advised, and is the default setting of this program..

It is also possible that a user might want to choose a weight of, say, 0.70 to put more emphasis on avoiding false negatives, but not total emphasis. The program has an option for the user to input the value of r (between 0 and 1) to obtain the optimal predictor for that cut-point. How you do this is described below in Section 4.3: How to Change Emphasis on Sensitivity versus Specificity.

## 1.3 Who Owns this Program?

It is in the public domain. The work that went into this was mostly paid for by the Department of Veterans Affairs and the National Institute of Aging of the United States of America.

## 1.4 Where Does the Theory Behind the Program Come From?

From HC Kraemer, Evaluating Medical Tests. Sage Publications, Newbury Park, CA 1992. The formulae for the calculations are taken from page “X” from the book and are presented in Appendix 3.

# 2. Overview of Programming Strategy

The ROC5 program is designed to perform basic signal detection computations in a Windows environment. The program is written in C++ Microsoft version 6.0. Original “Mark 4” version was written circa October 2001. Likely it can be recompiled on other platforms that use C++ or C, such as Sun, SGI or other UNIX workstations, and maybe the Mac. For details on capacity of the program see Appendix 1, but basically it has been successfully run on datasets of up to 50 variables and up to 8000 cases on standard PCs. It will also run successfully on much larger datasets, albeit a lot slower.

To get the full benefit from this program it would probably be easiest to use Excel. ROC curves can be generated using Excel or SAS. It is a waste of time to recreate the editing and statistical capabilities of Excel and SAS, especially the latter for plotting ROC curves and the former for creating a clean dataset.

So, the basic idea is that however you prepare your data, move it to Excel and output the data as a text tab-delimited (separated) text file (.txt extension in Excel). Then, after running ROC5, you also get a text tab-delimited dataset, which is readable by Excel or SAS (SAS Institute Inc., Cary NC) for plots. Details on plots are in Section 4.4.2. However, you may just be satisfied with results that come out of ROC.

The basic idea is:

|  |
| --- |
| Data Prep (**Excel**) 🡪 Signal Detection Calcs (**ROC5)** 🡪 Graphics (**Excel or SAS**) |

# 3. Data Preparation

## 3.1 The Gold Standard versus Predictors

The ROC program reads in data via a text tab-delimited format. The last column is a set of 0’s and 1’s representing the “gold standard”. This is the criterion for “success”. The other columns are the “predictors”. This can all be arranged in an Excel file and then output to a tab-delimited .txt file.

## 3.2 Details of Data Preparation

### 3.2.1 Missing data

Represent missing data only with a **–9999**. If you have blanks, edit it in Word first and do a global replace of ^t^t (two tabs) with ^t-9999^t.

**IMPORTANT NOTE: In ROC4 the missing value code was “-9999.99”. Note that this has been changed.**

### 3.2.2 ID Numbers, Character variables

Remove any columns of data that will not be analyzed (e.g. ID numbers and character variables).

### 3.2.3 Note on Data Recoding

This should be done in Excel before submitting the data to ROC5. See Appendix 2 for information on recoding. A Demonstration dataset (Demo.txt) is also enclosed as part of the Zip package.

### 3.2.4 Note on Variable Names

While ROC5 accepts variable names up to 24 characters in length, it is recommended to keep your variable name length 10 characters or fewer to minimize possible confusion. On the summary page, there was only room to print the first 10 characters of each variable.

### 3.2.5 Note on Number of Decimal Places

While raw data are not rounded when performing ROC analyses, numbers are rounded to 3 decimal places when printing results. If your data has 4 or more significant decimal places, multiply by the appropriate factor of 10 to remove the decimal places. For example, if you have a variable with values like 0.1234, multiply this variable by 10,000 to remove the decimal places in your raw data set. When this value appears in your ROC output (like 1234), divide by 10,000 to get the original value back.

# 4. Running the ROC Program

## 4.1 How do you Run the Program?

### 4.1.1 Batch Files Basics

It is easiest to run the program as a batch file (.bat), i.e., you just double-click the file name or icon. This basically is a place that keeps all your files and commands straight. For example, rDemoData.w64.05.bat consists of a single line that can be edited in Notepad or MS Word:

**ROC\_5.07\_w64 Demo.txt 50 NO\_PLOT PRINT NO\_DE\_BUG 05 20 > runDemoData\_w64.05.docx**

This tells **ROC\_5.07\_w64** (the 64-bit version of ROC\_5.07) to use **Demo.txt** as the data file and **output** (the “>”) the results to **runDemoData\_w64.docx** as a MS Word (.docx) file.

The other command line arguments are now required and are defined as follows:

* “ROC\_5.07\_w64” runs the 64 bit version of ROC 5.07; “ROC\_5.07\_w32” runs the 32 bit version
* “Demo.txt” is the name of the .txt data file to be read in. It is the name of the supplied demonstration dataset. Replace “Demo.txt” with the name of your data file
* “50” is the percentage weight emphasizing sensitivity vs. specificity. r=50 places equal weight on both. A 70 would place 70% emphasis on sensitivity vs 30% specificity. Any multiple of 10, from 0 to 100 can be used. We often use “50”. Further explanation is in Section 4.3
* “NO\_PLOT”: Do not output data for an ROC Curve. For now, please leave “NO\_PLOT” as is. We are currently working on a “PLOT” option
* “PRINT”: Print all intermediate output. If your output ROC file is too large to easily handle, replace with the “NO\_PRINT” option, which will considerably shorten the output
* “NO\_DE\_BUG”: Please leave “NO\_DE\_BUG” as is unless you want to see debugging output
* “05” is the Chi-Square p-value criteria (p<.05) for displaying a cut-point on the ROC tree. Other options are “01” (p<.01) and “001” (p<.001). “05” is the least stringent criteria and may result in a bigger ROC tree; “001” is the most stringent criteria and may result in a smaller ROC tree. We often use “01”.
* “20” is the number of subjects needed for the marginal counts. “30” is the most stringent criteria. Other options are “25”, “15” and “10”. “10” is the least stringent criteria and may result in a bigger ROC tree. We often use “20”. Please note this is not the number in each of the 2x2 Chi-Square cells but the of sum two cells and is not readily apparent from the short output. You can see how this works if you follow the longer output and see how results are eliminated. The relevant C++ code is at the top of the next page (this is not obvious or simple):
* aa=True\_Positives[k][j]; /\*\*predicted 1 actual 1\*\*/
* bb=False\_Positives[k][j]; /\*\*predicted 1 actual 0\*\*/
* cc=False\_Negatives[k][j]; /\*\*predicted 0 actual 1\*\*/
* dd=True\_Negatives[k][j]; /\*\*predicted 0 actual 0\*\*/
* ac=aa+cc; /\*\* actual 1 actual 0 marginal counts \*\*/
* ab=aa+bb; /\*\* predicted 1 aa bb ab \*\*/
* bd=bb+dd; /\*\* predicted 0 cc dd cd \*\*/
* cd=cc+dd; /\*\* marginal counts ac bd abcd \*\*/
* “runDemoData\_w64.05.docx” is where the ROC output will be directed. Substitute “DemoData” with the name of the dataset you are using. By default, the output is sent to a MS Word “.docx” file. However, if you would like the output directed to a .txt file instead, replace the “.docx” with a “.txt”. As the ROC output is text, any program that can handle .txt files should be able to read it in.

### 4.1.2 Batch Files Quirks

Batch (.bat) files seem a bit quirky in Windows. We have found that it is easiest to modify one that already works (such as those supplied) and save it as a text file with a different name (and keeping the .bat extension). This can be easily done in Notepad or MS Word. After that you can just double-click the new filename.

**Note well**: Please make sure your data file (.txt) or output file (.docx) are closed. The batch file will not run if either is open.

How do I know it is running? When you double-click the .bat file, a black (DOS) screen will show up, with the contents of your .bat file listed. If your dataset is small, this black screen may literally flash on the screen, as the ROC program might take less than a second to run.

If the black screen persists for more than a couple of minutes, look in the folder where your output file (.docx) is directed. Right-click your mouse, select “Refresh”, note the file size, and wait a few more minutes (or longer if your file is huge). Right-click your mouse and select “Refresh”, again. If the file size is larger, take heart that the ROC program is working and go get some coffee (or a good night’s sleep if you have a slow processor or huge dataset). To get a rough idea of how long it may take to run your ROC program please see Appendix 1.

## 4.2 What does the ROC Output Mean? (and how to read it)

The output is readable in MS Word, after some formatting adjustments. It is designed to be read in and printed using the following format:

1. Select the “.docx” ROC output. Microsoft Word will automatically open.
2. Select Page Layout 🡪 Orientation 🡪 Landscape.
3. Select Margins 🡪 Narrow
4. Select All (Control-A), then go to Home and change the font to Courier New, 6 point.
5. Go to the bottom of the document, then scroll up a bit. Insert a page break (Insert 🡪 Page Break) between the lines “Computation 14 & 15 over” and “\*\*\* Summary”.
6. SAVE (as a new file name if you wish).

There are six segments to the output:

1. The output starts with descriptive statistics for the predictor variables and gold standard; i.e. this is here for a data check. Make sure n’s are ok and missing values are handled properly.
2. You then get a listing of the signal detection results for each variable and for each cut-point (value) of each variable in your dataset.
3. Next a summary of the results for the highest weighted kappa values for each variable is printed. If you chose have the default r=50, then this is the value “k0\_50”. In general the best cut point to separate successes from failures will be the value of the variable with this highest kappa over all the variables.
4. The program will do a series of “iterations”, basically taking the best cut-point identified in (3) above and rerun the data that are above and below that cut-point. This step is repeated until all cut-points (up to three-way interactions) are identified. If you would like to identify interactions beyond three-way, see Section 5.
5. A summary of the results
6. The Decision Tree (a simplified version was presented in Section 1.2.1)

## 4.3 How to Change Emphasis on Sensitivity versus Specificity

The program has an option for the user to input the value of r (between 0 and 1) to obtain the optimal predictor for that cut-point. Why you might want to do this is described in Section 1.2.2: Weighing the Importance of False Positives versus False Negatives. Note how the script is changed to accomplish the change in emphasis:

**ROC\_5.07\_w64 Demo.txt 70 NO\_PLOT PRINT NO\_DE\_BUG 05 20 > runDemoData\_w64.05.docx**

This version of the script has a 70 added. This will calculate a 70/30 split to kappa emphasizing sensitivity (70%) versus specificity (30%). Default is 50/50. You can use any proportion as long as it is a multiple of 10; e.g. 0, 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100 are acceptable. Note that optimal sensitivity and specificity (100 and 0, respectively) are automatically calculated by the program, regardless of what is chosen here.

## 4.4 How to Get Results for Plots, i.e. ROC Curves

### 4.4.1 How to Actually Get a ROC Plot out of the Data

Several programs such as Excel or SAS can simply read in the ROC output.docx file. The output from ROC4 has some means at the top of the file and headers at the top of each variable, which needs to be stripped off (easily done in Excel and saved as a tab-delimited text file) before creating graphics.

Although there are many programs that do graphics, programs such as Excel may only allow relatively simple plots. The SAS program supplied in Appendix 4 will read in the data and create classic ROC Plots, after a couple of lines in the supplied code are modified.

### 4.4.2 ROC Plots in Excel and SAS

In Excel: The file ROC\_Graph\_Excel.xlsx is supplied in the .zip file. This file contains: 1) in the “ROC\_Output” tab, results copied and pasted from Segment 3 as described in Section 4.2 above. (Note these results are not from the supplied demo.txt): 2) in the “ROC\_Graph” tab, the ROC graph and the actual data used for the graph.

“ROC\_Output” tab: In this analyses, we are interested in 20 different predictor variables (numbered 5 to 24). The data needed for the graph are highlighted in yellow, and the optimal cutpoint with the highest weighted kappa (using r=50) is in red.

“ROC Graph” tab: The yellow highlighted data are pasted in rows 5-24. The data are then reformatted for the graph as follows:

* A column (Column D) is created for 1-Specificity
* Every predictor variable and cut-point is listed separately in Row 4.
* The sensitivity for each predictor is pasted in the appropriate column
* Select all the columns created above and select “Scatterplot”. Rows/columns may need to be switched
* All lines, arrows, etc. were drawn (hard-wired)
* This graph replicates the graph on p. 73 of Dr. Kraemer’s book, except that 1-specificity is on the x-axis rather than specificity.

In SAS: SAS (Statistical Analysis System) is the most widely used professional statistical language in the world. It used to run only on mainframes and UNIX workstations but now runs on PCs. Most Universities have contacts to use it. Pharmaceutical company clinical trial data are submitted to the FDA in SAS datasets. Template for SAS code is provided in Appendix 4.

To facilitate SAS direct input, unnecessary lines of the output file are preceded by a “**\*\***”. This allows these lines to be ignored by SAS.

# 5. Run it Again Sam? More on Decision Trees

Once the first optimal cut-point is found, yielding the first positive and negative test groups, the program runs again on those with a positive first test, and those with a negative first test, to find the optimal cuts within those subpopulations. Thus if the optimal predictor at the first stage is gender, males and females are analyzed separately in the next two runs of the program. When the second level of cut-points are determined (yielding 4 subpopulations), the program could run again on each of the four subpopulations. Doing this produces a "decision tree", a series of "and/or" rules that identify subgroups at different risks of the binary outcome.

Note: Next to the Chi-Square values, \*, \*\*, and \*\*\*, represent p < .05, .01 and .001 respectively. This "significance" is the result of multiple testing and should not be regarded as a true significance level, but is useful as a stopping rule. This in fact is how the program works. It will result in a maximum of three levels of cut-points and a final maximum of eight (2x2x2) subgroups. The reason it stops at this point is two-fold: 1) programmer exhaustion after having created 2,422 lines of C++ code and 2) three-way interactions of predictors are being generated.

If you would like a four-way (or more interaction), divide the initial dataset on the cut-points identified by the three-way interaction and rerun the program. We have successfully identified up to a ten-way interaction using this method.

# 6. FAQ (Frequently Asked Questions)

Q: How is this program different from CART, SPSS Answer Tree, etc:

A: This program is more concise as it uses kappa to minimize false positives and false negatives. We believe other programs produce too many ‘branches’ (and thus sometimes requires ‘pruning’ afterwards), and / or use the odds ratio, which we do not favor.

Q: In the SAS graphing program in Appendix 4 is there a specific reason why only the top 25 points (sorted by descending k0\_50) are plotted:

A: We arbitrarily decided to plot the top 25 points because we felt plotting more would clutter the graph. Of course it is easy to change to program to plot more or less points if you desire.

Q: Does the ROC program use the empirical ROC curve rather than a fitted curve?

A: Yes, that is correct. The fitted curve assumptions are usually not true.

Q: I am curious how the weighting is done when you select differing values for the sensitivity/specificity emphasis. I have been taught that the bias parameter often represents the ratio of the ordinate of the S distribution over the ordinate of the SN distribution (I may have these backwards). Does ‘emphasis’ in the ROC program map onto that concept in some way?

A: No. The ROC program makes no distribution assumptions. It is non parametric.

# Appendix 1: Note on Memory Allocation and Run Time

**(Note: ROC5 no longer has a strict maximum allocation, and we have not reached the upper limit of the capabilities of this program. We invite you to try running the ROC program on your data, no matter how large, and see what happens. What is written below applied to the previous version of this program, ROC4, and is now obsolete. ROC5 runs much faster than indicated below).**

The program is designed to make maximum use of the memory capacity of your machine. This is industrial-design programming, rather than convenience-designed programming. As written it uses about 12MB of memory to run a matrix of 25 variables for 35000 cases (largest test so far). In this situation, it may take the program over an hour to run data on a 650mHz Intel processor, but at least it runs in the background, though it takes up 90% of my processor. The maximum initialization is for 1000 variables and 50,000 cases. It does not initialize to a larger number because a larger initial data matrix may crash on a machine with small memory capacity or at least make the program use virtual memory (disk instead of RAM). If you have a large dataset, a big machine and want to edit the program, just change the values for NCOL and NROW to the number of variables and cases you need, then recompile, or ask me and I can do it. The standard program, however, allocates memory only as needed and will run much faster on small datasets, such those with as 10 variables and 500 cases (a few seconds). Slowness appears to increase exponentially with N because computations involve not only each case, but every other case too, etc. etc...

To see how all this memory allocation is done, look for the function malloc() in the program. Clever programmers with time on their hands could do this more efficiently and free memory once it is no longer used. The way it is written, you use about 6 bytes/case times the number of variables. Stingier allocation of memory would get it down from 5MB. FYI, on my machine Windows 2000 toys seem to use about 40MB, Microsoft Word 20 MB and Outlook 15MB. Bottom line, forget running this on a machine with 64MB main storage (it will have to use slow virtual or paging memory). I may compile a larger capacity version and leave it on the web site – for those who want to deal with US census data and have the machine to complete the task.

How long might the program take to run?

We have successfully run the ROC program with about 20 variables (half of which were binary) and 35000 observations. This took 2 hours on a Dell M60 laptop with Windows XP, 512MB memory and 1.7 Gz P4m processor, BUT, only by using the new NO\_PRINT option. Adding this to the command line will supress the initial intemediate values. The sorting of these intermediate values could take literally DAYS in large datasets. The suggestion is, if it runs slowly on a large dataset, use this option.

# Appendix 2: Note on Data Recoding

I think it is best to do this in Excel for two reasons:

By the time you indicate how you should recode data in command line arguments, they get too big and confusing and prone to error. Same thing for gold standard computations. Better to get the dataset fixed up nice in Excel before running the program. You can just delete any offending columns and you can compute any gold standard you want using conditional statements i.e. ifs. Here is an example. Say you want to consider an infant feeding program success if children are over 1800g birthweight, well just put:

**=IF(C2<1800,0,1)**

In the last column first cell – assuming C2 is where the birth weight is located. Then highlight the whole column down to the bottom and hit Control-d, or copy down the column. Voila, data recoding with 0s and 1s in that column. Now, people might say I am just lazy, but Microsoft has put a whole lot of options in those conditional statements and they work for text. Let the following help page for the one conditional “if” from Microsoft Excel be a testimonial:

## IF

Returns one value if a condition you specify evaluates to TRUE and another value if it evaluates to FALSE.

Use IF to conduct conditional tests on values and formulas.

**Syntax**

**IF**(**logical\_test**,**value\_if\_true**,value\_if\_false)

**Logical\_test**   is any value or expression that can be evaluated to TRUE or FALSE. For example, A10=100 is a logical expression; if the value in cell A10 is equal to 100, the expression evaluates to TRUE. Otherwise, the expression evaluates to FALSE. This argument can use any [comparison calculation operator](file:///\\vhapalgeminia\..\Documents%20and%20Settings\Administrator\Local%20Settings\Temporary%20Internet%20Files\011105%20download\xlrefUnderstandinghowOperatorswork.htm).

**Value\_if\_true**   is the value that is returned if logical\_test is TRUE. For example, if this argument is the text string "Within budget" and the logical\_test argument evaluates to TRUE, then the IF function displays the text "Within budget". If logical\_test is TRUE and value\_if\_true is blank, this argument returns 0 (zero). To display the word TRUE, use the logical value TRUE for this argument. Value\_if\_true can be another formula.

**Value\_if\_false**   is the value that is returned if logical\_test is FALSE. For example, if this argument is the text string "Over budget" and the logical\_test argument evaluates to FALSE, then the IF function displays the text "Over budget". If logical\_test is FALSE and value\_if\_false is omitted, (that is, after value\_if\_true, there is no comma), then the logical value FALSE is returned. If logical\_test is FALSE and value\_if\_false is blank (that is, after value\_if\_true, there is a comma followed by the closing parenthesis), then the value 0 (zero) is returned. Value\_if\_false can be another formula.

**Remarks**

* Up to seven IF functions can be nested as value\_if\_true and value\_if\_false arguments to construct more elaborate tests. See the following last example.
* When the value\_if\_true and value\_if\_false arguments are evaluated, IF returns the value returned by those statements.
* If any of the arguments to IF are arrays, every element of the array is evaluated when the IF statement is carried out.
* Microsoft Excel provides additional functions that can be used to analyze your data based on a condition. For example, to count the number of occurrences of a string of text or a number within a range of cells, use the [COUNTIF](file:///\\vhapalgeminia\..\Documents%20and%20Settings\Administrator\Local%20Settings\Temporary%20Internet%20Files\011105%20download\xlfctCOUNTIF.htm) worksheet function. To calculate a sum based on a string of text or a number within a range, use the [SUMIF](file:///\\vhapalgeminia\..\Documents%20and%20Settings\Administrator\Local%20Settings\Temporary%20Internet%20Files\011105%20download\xlfctSUMIF.htm) worksheet function. Learn [about calculating a value based on a condition](file:///\\vhapalgeminia\..\Documents%20and%20Settings\Administrator\Local%20Settings\Temporary%20Internet%20Files\011105%20download\xlconCondFormulas.htm).

**Examples**

On a budget sheet, cell A10 contains a formula to calculate the current budget. If the result of the formula in A10 is less than or equal to 100, then the following function displays "Within budget". Otherwise, the function displays "Over budget".

IF(A10<=100,"Within budget","Over budget")

In the following example, if the value in cell A10 is 100, then logical\_test is TRUE, and the total value for the range B5:B15 is calculated. Otherwise, logical\_test is FALSE, and empty text ("") is returned that blanks the cell that contains the IF function.

IF(A10=100,SUM(B5:B15),"")

Suppose an expense worksheet contains in B2:B4 the following data for "Actual Expenses" for January, February, and March: 1500, 500, 500. C2:C4 contains the following data for "Predicted Expenses" for the same periods: 900, 900, 925.

You can write a formula to check whether you are over budget for a particular month, generating text for a message with the following formulas:

IF(B2>C2,"Over Budget","OK") equals "Over Budget"

IF(B3>C3,"Over Budget","OK") equals "OK"

Suppose you want to assign letter grades to numbers referenced by the name AverageScore. See the following table:

|  |  |
| --- | --- |
| **If AverageScore is** | **Then return** |
| Greater than 89 | A |
| From 80 to 89 | B |
| From 70 to 79 | C |
| From 60 to 69 | D |
| Less than 60 | F |

You can use the following nested IF function:

IF(AverageScore>89,"A",IF(AverageScore>79,"B",

IF(AverageScore>69,"C",IF(AverageScore>59,"D","F"))))

In the preceding example, the second IF statement is also the value\_if\_false argument to the first IF statement. Similarly, the third IF statement is the value\_if\_false argument to the second IF statement. For example, if the first logical\_test (Average>89) is TRUE, "A" is returned. If the first logical\_test is FALSE, the second IF statement is evaluated, and so on.

## Even more important is the operator AND in Excel:

IF(AND(1<B4, B4<100), B4, "The value is out of range.") equals "The value is out of range."

This allows you to say if the baby has a big birth weight AND not sick, it is a success.

There is simply no way to match Bill Gates on this flexibility.

## Even more important is the operator = in Excel:

Finally, note that the IF is usually preceded by an = in Excel so that the value is assigned to the cell.

# Appendix 3: Formulae

Taken from Kraemer (1992) page ‘X’:

##### TP=true positive=% positive test and diagnosis

TN=true negative=% negative test, negative diagnosis

FP=false positive=% positive test, negative diagnosis

##### FN=false negative=% negative test, positive diagnosis

P=prevalence=TP+FN (P’=1-P)

Q=level of test=TP+FP (Q’=1-Q)

SENS=sensitivity=TP/P

SPEC=specificity=TN/P’

PVP=predictive value of positive test=TP/Q

PVN=predictive value of negative test=TN/Q’

EFFIC=efficiency=TP+TN

k0\_0, k\_50, k1\_0 are weighted kappa coefficients where:

k0\_0=(SPEC-Q')/Q

k1\_0=(SENS-Q)/Q'

k0\_50=(EFFIC-PQ-P'Q')/(1-PQ-P'Q')

# Appendix 4: Example SAS Program for Graphics

**(Note: This program has not been popular, so it has not been well tested and should be considered preliminary)**

You will need to have a site license for SAS/Graph in order to run this program successfully. Your SAS Site Representative should know whether you have this.

In order to run this program, two lines of code may need to be modified:

1. In the **infile** statement about half a page down below

infile 'C:\art\QROC\people\Jerry\011101 Helena Data\run\_controlANDtreatment\_501.txt'

Change the directory structure and filename to that corresponding on your PC

1. Towards the end of the program there is a goptions statement :

(goptions device= /\* WIN \***/ LJIV600).**

If you would like to view the results on your screen, remove the “/\*” and”\*/” surrounding “WIN” and place them around “LJIV600”, which refers to a particular HP printer. If you have a different printer, please consult the SAS documentation for the code for your particular printer.

Once these steps are complete, the program can be run and a listing showing the top 25 points (sorted by descending k\_r) will either be plotted or displayed on your screen and these top 25 will also be shown in the SAS output window, where they can be printed.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* QROCPLOT.011030.SAS \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

10/30/01

Read in output from Jerry's QROC program and create two plots:

1) K1\_0 vs K0\_0 (ROC Curve)

2) Sensitivity vs Specificity (QROC Curve)

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* SET UP DEFAULT SAS OPTIONS \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

options ls=**110** ps=**50** nocenter pageno=**1** errors=**2** mprint;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* READ IN OUTPUT FROM JERRY's PROGRAM \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**data** qroc;

infile 'C:\art\QROC\people\Jerry\011101 Helena Data\run\_controlANDtreatment\_501.txt'

dlm='09'x lrecl=**500** missover;

input varnum n ltge $ critraw $ p q sens spec effic k0\_0 k0\_r k1\_0

pvp pvpnum pvpden pvn pvnnum pvnden

chi\_square sig $;

/\* Set up new variable CRITERION which concatenates LTGE and CRITRAW \*/

criterion=compress(ltge||critraw);

**run**;

**proc** **print** data=qroc uniform;

var varnum n criterion p q sens spec effic k0\_0 k0\_r k1\_0

pvp pvpnum pvpden pvn pvnnum pvnden

chi\_square sig;

title "Output from Jerry's QROC Program";

**run**;

/\* Keep top 25 obs, sorted by descending k0\_r (r=0.5 by default) \*/

**proc** **sort** data=qroc;

by descending k0\_r;

**run**;

**data** qroc;

set qroc;

if \_n\_ le **25**;

**run**;

**proc** **print** data=qroc uniform;

var varnum n criterion p q sens spec effic k0\_0 k0\_r k1\_0

pvp pvpnum pvpden pvn pvnnum pvnden

chi\_square sig;

title "Top 25 records, sorted by descending k0\_r";

**run**;

/\* Set up reference lines for plots as follows: \*/

/\* 1) for k1\_0 vs k0\_0 plot, add k1\_0=k0\_0 line \*/

/\* 2) for sens vs spec plot, add \*/

/\* (a) sens=1-spec line and \*/

/\* (b) line from (P Prime,P) to (1,1) \*/

**data** qroc;

set qroc;

if \_n\_=**1**

then

do;

k1\_0ref=**0**;

k0\_0ref=**0**;

sensref=**1**;

specref=**0**;

pref=p;

pprimeref=**1**-p;

end;

else

do;

k1\_0ref=**1**;

k0\_0ref=**1**;

sensref=**0**;

specref=**1**;

pref=**1**;

pprimeref=**1**;

end;

**run**;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* SET UP PLOTTING OPTIONS \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*-------------------------------------------------------------------\*

| Summary: |

| Creating a simple plot using the table |

| SASUSER.SENSSPEC and plotting |

| sens against spec. |

| Generated: 29OCT2001 1:05:01 |

\*------------------------------------------------------------------\*/

/\*------------------------------------------------------------------\*

| The GOPTIONS statement allows you to have more control over the |

| final appearance of your output such as fonts, colors, text |

| height and so on. The output device and destination is also |

| specified in the goptions statement. |

\*------------------------------------------------------------------\*/

/\* HSIZE and VSIZE are different, but give a square plot \*/

goptions reset=(axis, legend, pattern, symbol, title, footnote) norotate

hpos=**0** vpos=**0** htext= ftext= ctext= target= gaccess= gsfmode=

hsize=**6**in vsize=**7**in;

/\* Use device = WIN to print to monitor, LJIV600 to print directly to HP Printer \*/

goptions device= /\* WIN \*/ LJIV600 ctext=black

graphrc interpol=none;

/\*------------------------------------------------------------------\*

| SYMBOL statements allow you to supply information such as plot |

| character, plot lines, color and interpolation. |

\*------------------------------------------------------------------\*/

/\* POINTLABEL is the variable number as specified in Jerry's output \*/

symbol1

color=black

value=star

pointlabel = ("#varnum")

;

symbol2

interpol=join

width=**3.0**

color=black

;

symbol3

interpol=join

width=**3.0**

color=black

;

/\*------------------------------------------------------------------\*

| AXIS statements allow you to supply information on how your |

| vertical and horizontal axes will appear on the graph. |

\*------------------------------------------------------------------\*/

axis1

order=**0.0** to **1.0** by **0.1**

color=black

width=**3.0**

style=**1**

;

axis2

order=**0.0** to **1.0** by **0.1**

color=black

width=**3.0**

style=**1**

;

axis3

color=black

width=**3.0**

;

/\*-----------------------------------------------------------------\*

| This section produces the actual plot and any options that |

| directly relate to the data and the axis area. |

\*------------------------------------------------------------------\*/

**proc** **gplot** data=qroc;

where (**0** le k1\_0 le **1**) and (**0** le k0\_0 le **1**);

plot k1\_0 \* k0\_0 k1\_0ref \* k0\_0ref /

haxis=axis1

vaxis=axis2

frame

overlay

;

title1 'QROC Plot';

title2 'K1\_0 vs. K0\_0';

**run**;

**proc** **gplot** data=qroc;

where (**0** le sens le **1**) and (**0** le spec le **1**);

plot sens \* spec sensref\*specref pref\*pprimeref /

haxis=axis1

vaxis=axis2

frame

overlay

;

title1 'ROC Plot';

title2 'Sensitivity vs. Specificity';

**run**;

quit;