INCREASE PRODUCTIVITY WITH MINITAB
MACROS

PATRICIA A CYR
Master Black Belt / Staff Statistician/ CQE

THIS INFORMATION IS NOT EXPORT CONTROLLED
THIS INFORMATION IS APPROVED FOR RELEASE WITHOUT EXPORT
RESTRICTIONS IN ACCORDANCE WITH A REVIEW OF THE INTERNATIONAL
TRAFFIC IN ARMS REGULATIONS (ITAR), 22CFR 120-130, AND THE EXPORT
ADMINISTRATION REGULATIONS (EAR) 15 CFR 730-774.
Topics Covered in the Presentation

• Description of our journey

• Review the creation of simple global macros

• Provide the details for using Task Scheduler to call Minitab macros

• Code examples from our current state

• Review tricks learned to make the macros modular, self-correcting, and more efficient
Background on the Data

- **We make assured communication devices**
  - We populate the printed circuit boards
  - We operationally test the boards
    - We can then combined several boards to make modules
      - We operationally test the modules
  - We combine boards, modules, flexible wiring boards, etc. to make final units we sell to customers

- At the unit level we thoroughly test each unit
  - Leak testing to be sure units are water tight
  - Initial testing and programming
  - Vibrational and thermal stress testing
  - Final testing

By the time a unit leaves our factory, it had to pass up to 32,000 tests between board, module, and unit level testing
We have LOTS of Data – Over 1 Terabyte

- The test data is kept in 2 main databases
- Within each of these databases, the data of interest is kept in 2 different tables
  - Sessions Table
  - Data Table

- The Sessions Table
  - Holds information about the test session
    - Date
    - Station
    - Operation Code (Process Step)
    - Part Number
    - Serial Number
    - Overall test session pass/fail results
    - Key
    - Etc.
The Data Table

• **The Data Table**
  - Holds the detailed test results for all the tests performed during a session
    - Test name
    - Reference number
    - Limits
    - Data values
    - Individual test pass/fail results
    - Key
    - Etc.

• **In general we can combine the tables based on the key**
Disclaimer about Your Data

• Obviously, your data is arranged differently

• The important thing to remember as we go through this case study is the method –

  ➢ Define a standard way to present the data to Minitab
    ➢ Our data is already organized in a standard format from the database

  ➢ Get Minitab to provide the analysis you want
    ➢ We agreed on p-charts for yield and Pareto charts for failures

  ➢ Capture that analysis for future execution by a macro
    ➢ Use the Minitab menus to do the analysis, then copy the code into a macro
Our Journey

• In 2007 our data was obtained one session at a time through a web interface that provided either a paper hard copy or an EXCEL file

• Looking for trends over time was not done

• We got permission to use Microsoft ACCESS to link to the database tables where the test data and results were kept

• We used Microsoft Access to link the sessions and data tables and to set the criteria to obtain the information of interest, then used copy and paste to pull the data into Minitab

• We had to keep our requests to less than about 65,000 lines to use the clipboard for copy and paste
How Did We Turn Our Data into Information?

• The quality engineers wanted to track performance
  • How are we doing?
  • What is failing?
    – How can we prevent those failures from happening again?
How Are We Doing? - The Business Metric was Yield

- We decided to present yield as a p-chart
  - Gave perspective that small shifts from day to day were normal variation
  - Allowed us to identify real shifts in performance

Tests performed with unequal sample sizes
To make p-charts with day as the subgroup, we needed to use the information contained in the Sessions Table

- Identify which day a test was run
- Identify the outcome of the first test at each operation for each item
- Summarize the data to have a table with day, number of passes that day, and number of total tests that day
The Details

• Using the Minitab **calculator** and **date()** function we extracted just the day part from the test time and date variable

• Using the Minitab **calculator** we assigned a value of 1 when the first 4 characters in the test result were PASS, 0 otherwise

• Using **store descriptive statistics** we were able to create a table that counted the number of passes (sum) and the number tested (number non missing)

<table>
<thead>
<tr>
<th>ByVar1</th>
<th>C11</th>
<th>C12</th>
</tr>
</thead>
<tbody>
<tr>
<td>4/9/2016</td>
<td>36</td>
<td>62</td>
</tr>
<tr>
<td>4/10/2016</td>
<td>68</td>
<td>94</td>
</tr>
<tr>
<td>4/11/2016</td>
<td>69</td>
<td>99</td>
</tr>
<tr>
<td>4/12/2016</td>
<td>77</td>
<td>98</td>
</tr>
<tr>
<td>4/13/2016</td>
<td>81</td>
<td>108</td>
</tr>
<tr>
<td>4/14/2016</td>
<td>61</td>
<td>83</td>
</tr>
<tr>
<td>4/15/2016</td>
<td>29</td>
<td>44</td>
</tr>
<tr>
<td>4/16/2016</td>
<td>23</td>
<td>36</td>
</tr>
<tr>
<td>4/17/2016</td>
<td>11</td>
<td>18</td>
</tr>
<tr>
<td>4/18/2016</td>
<td>18</td>
<td>24</td>
</tr>
<tr>
<td>4/19/2016</td>
<td>50</td>
<td>82</td>
</tr>
<tr>
<td>4/20/2016</td>
<td>52</td>
<td>75</td>
</tr>
<tr>
<td>4/21/2016</td>
<td>28</td>
<td>63</td>
</tr>
</tbody>
</table>

• Made a p-chart from the table
All of the Minitab code used to execute the choices made through the menus are stored in the history tab.
Creating a Macro

You can make a global Minitab macro for this code. The next time you pull data *in the same format*, you can just run the macro to produce the p-chart.

Creating a simple global macro:

1. Open notepad
2. Type the word `gmacro` on the first line
3. Type the `macroname` on the next line (must start with a letter and have no spaces or some special characters)
4. Copy the code from the history window
5. Type the word `endmacro` on the last line
6. Save as `macroname.mac` in the location from which you will run macros

You can add extra lines for readability. Notes can be added by starting the line with `#`
Executing a Macro

Have the worksheet with the Session Table data active. In the session window, at the command prompt, enter

MTB > %macroname

The steps saved to the macro will now be executed on the current worksheet

Since our macro references specific columns and variable names, it is important that the data be presented to Minitab in the same format as it was when the macro was created
Which Tests Are Failing Most?

We used a 28 day Pareto type chart to find the tests failing most

• We added color coding to track if problems were getting better or worse
  – Failures occurring yesterday were shown in red
  – Failures between 2 and 7 days ago were shown in orange
  – Failures between 8 and 14 days ago were shown in yellow
  – Failures between 15 and 28 days ago were shown in green

• The color scheme was already in use by one quality engineer using EXCEL to process his data

• People were already used to the notion that the green covered the same amount of time as the other three colors combined
Time Trending Pareto Chart – the Data Table

- We had to create a variable that calculated the age of a result – we called it delta.

- We then created another variable into which we could code the age into the 4 buckets of (1 day, 2-7 days, 8-14 days, 15-28 days) – we called that ‘variable’.

- We used the stacked bar chart in the Graph menu to create a Pareto type chart.
  - The outermost variable was a test identification variable and the next variable was the age group indicator which is called ‘variable’.
  - Under chart options we selected Decreasing Y.
  - Under scale we selected Transpose value and category scales.
Evidence these failures may be getting worse
The Code

Name c24 "timevalue"
Numeric 'MinOFTEST_DATE_TIME' 'timevalue'.
Let k1 = today()
Name C25 'delta'
Let 'delta' = 'timevalue'-K1
Name c26 "variable"
Code (-1:0) "d 1 day" (-7:-1) "c 2 - 7 days" (-14:-7) "b 7 - 14" (-28 & :-14) "a 14 - 28" (0:1) "today" 'delta' 'variable'

Chart ‘TEST_NAME’;
  Group 'variable';
  Stack;
  Decreasing;
  Transpose;
  Exclude;
  Where "C25 >0";

The letters in front of the date ranges are needed to order 1 day to 14-28 days in the legend from top to bottom

Minitab 15 defaulted to the desired color scheme. With newer versions of Minitab, we need to change the first 4 color definitions in TOOLS > OPTIONS

This code was used to create a second macro to make time trending Pareto Charts
Productivity Update

Now the quality engineers could create the p-charts and the trending Pareto charts in minutes by running the two macros, once they collected and processed the initial data

They went from reviewing the data every 2 weeks to a weekly review

The trending Pareto was used to prioritize improvement actions. We could see that effective improvements resulted in bars on the pareto that were all green since there were no longer any failures of that type occurring in the past 2 weeks!
Where Are We in the Journey?

At this point, the user still had to extract the data for each product by editing the criteria in an ACCESS database, then copy and paste the resulting data into Minitab keeping the number of rows to about 65k or less.

The next iteration for the macro design used the ODBC functionality of Minitab to import the data directly from ACCESS.

A separate ACCESS database was set up for each product so the criteria did not have to be changed. The time criteria were indexed, such as > (now() - 90) to allow for a moving window of information.

Separate tables were set up to contain the first pass session results for 3 months and the individual test failures for the past 28 days.

A separate Minitab macro was created for each product that pulled in both types of data and made both types of charts.
Productivity Update

• Now that the data conditioning was accomplished through the macro as well, the data was gathered and analyzed daily!

• Impatience ensued – “In addition to the 28 day charts, can we also have 7 day charts? We want to know if improvements are helping sooner”

• “Can we also have charts for 1 day and 3 day performance to monitor daily?”
• The 3 day was needed for Monday mornings so that the entire weekend performance could be combined
The Initial Macro Layout

At this point, the macros evolved to extract the session information first and process this data for all process steps.

The worksheet was split on operations.

P-charts were created for each operation.

Next, the failure information was extracted for all operations.

The worksheet for failures was split on operations.

Time trending Pareto charts were created for each operation.

Pitfall – All operations were hard coded in the macro. If there was a temporary OP code in the data for troubleshooting purposes or in error, the macro would fail because the worksheet number for the next extract no longer matched the worksheet number hard coded into the macro.
Improvement in Macro Design

The first major operational improvement involved moving both data extracts to the beginning of the macros.

There is always session data and some failure data, so worksheets 1 and 2 were always the correct worksheet numbers to be filled.

Form there, splitting these worksheets could easily be accomplished into the next available worksheets.

The first major content addition involved expanded Pareto information. The 28 day charts were helpful, but more specific Pareto charts for the past 7, 3, and 1 day were requested. The color scheme in these versions was by serial number to see if multiple units were having the same failure patterns.

Pitfalls – If there were many failed unique test numbers, the y axis was unreadable.

Needed another macro to limit the trending Pareto charts to the top 20.
Macro Storage

• In the beginning, macros were developed for specific people. The macros were saved to the macro folder in Minitab on individual C drives.

• Every time there was a change to a macro, we had to remember to send the updated version to all the right people.

• Due to an IT policy change, it became impossible to save the files locally.

• We decided to keep all macros in a central, shared location.

• We used TOOLS > Options to map all users to the shared macro location.

• We maintain the latest macros in that shared location where anyone can use them.
Working on the biggest product problems was successful!!!

Now we had 2 different problems with the macros

1. Sometimes there were **no failures** for the past 1, or 3, or 7, or even 28 days for some process steps
   - Worksheet names and Pareto chart codes were hard written in the macros. Macros fail when a specified worksheet is not found
   - Failing macros stop execution. No code past that point is processed
     - Solution - move the ‘offending’ code to the end so the error occurs after all processing of interest is completed

2. The macros were so popular that every morning between 6 and 8 am there were so many calls to the database that it locked up, and no one got any data!!

The next step in the journey was to use Microsoft Task Scheduler to run the macros for the major products between 2 and 5 am
Getting Minitab Macros to Run with Task Scheduler

- Task Scheduler executes `.bat` files
  - We create `macroname.bat` to keep things organized
- You need to create a `.mtb` Minitab batch file to be called by the `.bat`
  - We use the `macroname.mtb` again to keep things organized
- The `.mtb` file calls the `macroname.mac`
- We keep the `.bat`, `.mtb`, and `.mac` files in the same location

**.bat file content** is where to find the `mtb.exe` file and the location of the Minitab batch file:

```
"C:\Program Files (x86)\Minitab\Minitab 17\mtb.exe" "file location for .mtb file\macroname.mtb"
```

<table>
<thead>
<tr>
<th>.mtb file content:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brief 2</td>
</tr>
<tr>
<td>echo</td>
</tr>
<tr>
<td>%macroname</td>
</tr>
<tr>
<td>noprompt</td>
</tr>
<tr>
<td>stop</td>
</tr>
</tbody>
</table>

The output of the macros is stored in a shared location. The project name is `macroname.mpj` and it is replaced every time the macro is executed.
Major Enhancements for Current State

- The p-charts and trending Pareto charts were good, but customers asked:
  - Can we also get information about how many were tested each day with the p-chart?
  - Can we also know how many were tested within each age group?
  - Can you limit the number of failures shown to the top 20 so we can always read the names?
  - Can you show us both first pass data and any pass data for everything?

- Form a maintenance point of view,
  - Can the macros be set up to know which operations actually have data for plotting?
  - Can there be one universal macro so that changes and additions can be made only once and not have to be added to all the individual product macros?

- IT requested that the ACCESS intermediate step be eliminated and the SQL query draw directly from the database
New Macro Structure

1) Only one macro for each top level product
   • Includes all board, unit, and OP Code combinations

2) P-charts and tested quantities for past 3 months, first pass and any pass for all combinations

3) 28 Day trending Pareto chart of failures across all combinations to understand biggest issues product wide, first pass and any pass

4) 28 Day trending Pareto charts of failures for each combination, first pass and any pass. Include number tested and number failed for each time period

5) 7, 3, 1 Day Pareto charts of failures by serial number for each combination only when data exist, first pass and any pass. Include number tested and number failed

6) Nest Check for Test Station performance for each process step
A Unified Approach

- To make the enhancements listed on the previous page, and others, it was obvious that a new macro structure was needed.

- A modular approach was adopted
  1) Separate macros are created for each top level product
     - These macros define the SQL query with the product specific information to pull the data directly from the database in the defined format.
     - Data prep creates 3 worksheets that are needed by the core macro.
     - They call the core macro.
     - Upon completion of the core macro, they define the storage location, file name, and save the data.

- The core macro has all the code. Improvements are only needed in one file and they are applied to all products.

- Creating a macro for a new product takes less than 5 minutes. Only the part numbers, operation steps, and file storage name need to be changed.
Minitab Helps Write the SQL

- First you need to create a data source name, DSN
- Use **FILE > Query Database (ODBC). . .> New**
  - Associate the type of data and physical location with the DSN

- Use the DSN, get to the data source. A dialog box appears with available tables, queries, etc.

- Select the fields you need. Minitab adds them to the current worksheet

- Minitab stores all the connection information and SQL code in the history window

```odbc
ODBC;
Connect "DBQ=C:\Users\cyr\Desktop\MINITAB CONFERENCE EXAMPLE.accdb;DefaultD" & "r=C:\Users\cyr\Desktop;Driver={Microsoft Access Driver (*.mdb, *.accdb)};" & "DriverId=25;FIL=MS Access;FILEDSN=C:\Users\cyr\Documents\desktop.dsn;MaxBu" & "fferSize=2048;MaxScanRows=8;PageTimeout=5;SafeTransactions=0;Threads=3;UID" & "=admin;UserCommitSync=Yes;";
SQLString "SELECT `KEY` ,`NEST_NUMBER` ,`OPERATION_CODE` ,`PART_NUMBER` ,`SERIAL_NUMBER` ,`TEST_DATE_TIME` ,`TEST_RESULT` ,`TEST_STATION` FROM `SESSIONS TABLE` .
```
2) Details for P-charts

- Concatenate part number and OP code to make a new ID variable

- Use sort and lag() to identify first tests for each serial at each OP code
  - Subset the worksheet to get just the first pass results

- Use store descriptive statistics to find number tested and number failed from test sessions table
  - Needed for each combination of ID to make p-charts
  - Needed to pass information to worksheets from the sessions table

- Create separate macros to handle p-charts for first pass and any pass results. Within those macros
  - Open layout
  - Make p-charts from session table data worksheets and place in top half of layout frame
  - Make bar chart of the number of units tested by day and place in the bottom half of the layout frame
The Macro Called Within the Core Macro

- K1 finds the number of ID combinations
- The do loop executes for each unique ID
- K2 extracts the unique ID for each iteration
- K3 makes the title to be used in the iteration
- The include statement subsets the data to only include the ID for the current iteration

```
gmacro
firstpcharts
let k1=count(c18)
do k10 = 1:k1
let k2 = c18[k10]
let k3 = concatenate(“p-chart for “,k2,” “,“first pass”)
layout;
wtitle k3.
PChart 'PASSED' 'TESTED';
Scale 1;
   Angle 90;
   Stamp 'DAY';
   Title k3;
   Include;
   Where "c11=k2'';
   figure 0 1 0.5 1.
Chart ( 'Tested' ) * 'day';
   Summarized;
title "Quantities Tested'';
   Bar;
mgrid 2;
   Include;
   Where "c11=k2'';
   figure 0.03 .93 0 0.5.
Endlayout.
Enddo
endmacro
```
3) 28 Day Overall Trending Pareto

- Failure details contained in Data Table
- Combine Part number/OP Code/Ref_num/Test_name to identify unique failure modes called idtype
- Limit the chart to the top 20
- Same analysis for first pass data and any pass data
Code to Pick the Top 20 Failures

Name C36 "Tally1" C37 "Tally2".
Tally 'idtype';
   Counts;
   Store 'Tally1' - 'Tally2'.
Sort 'Tally1' 'Tally2' 'Tally1' 'Tally2';
   By 'Tally2';
   Descending 'Tally2'.
N 'Tally2' k1.
Set c38
   1( 1 : k1 / 1 )1
   End.
name c39 'top 20'
Copy 'Tally1' c39;
   Include;
   Where "C38<21".
let k56 = if(k1>20,20,k1)
set c40
   1( 1 : k56 / 1 )1
   End.
Convert c39 C40 c27 c41.

- Idtype is the unique identifier
- Store the idtype names and Counts. Sort by the counts from largest to smallest
- Place the total number of rows in k1
- Make a column from 1 to k1
- Copy the idtype information only for the top 20
- K56 is error proofing in case there are less than 20 idtypes
- Make a column with the numbers 1 to k56
- Use a Conversion table to create a new column (c41)
- When the idtype in C27 matches one of the Top 20 in C39, the count in the top 20 (c39) is saved in C41. An * is saved if the record is not in the top 20
Code for the Charts

let k4 = left(c4[1],9)

Chart 'idtype';
   Group 'variable';
   Stack;
   Decreasing;
   Transpose;
Excluding;
   Where "C41 >20";
wtitle "First Pass top 20";
scale 1;
   PSize 6;
grid 2;
   Title "First Pass Top 20 Failure Types Across Operations";
subtitle k4;
   Bar.

This is handled in the core macro since it is done across all combinations
4) 28 Day Pareto for Each Combination

To have counts of the number tested and the number failed in each time group, the information must be passed from the session table based worksheet to be included in the data table based worksheet.

Our choice was to find the counts of tests and fails, create a table with ID information in the first column in a ‘temporary’ worksheet. These columns in the ‘temporary’ worksheet were then merged with the failure data worksheet.

A separate stand alone macro can be used to cycle through all the ID combinations.

```
gmacro
first28day
let k1=count(c28)
do k10 = 1:k1
let k2 = c28[k10]
let k3 = concatenate("28 Day First Pass Failures for ",k2)
let k101=c33[k10]
let k102=c32[k10]
let k107=c31[k10]
let k114=c30[k10]
let k201=c52[k10]
let k202=c51[k10]
let k207=c50[k10]
let k214=c49[k10]
Copy c1 - c26 c101 - c126;
Include;
Where "c11=k2";
Varnames.

let k5 = concatenate("Radio Counts 14-28 days ",k114," 7-14 days ",k107," 2-7 days ",k102," 1 day ",k101)
let k6 = concatenate("Radio Fails 14-28 days ",k214," 7-14 days ",k207," 2-7 days ",k202," 1 day ",k201)
nname c114 "test name" c126 "timeframe".
*****Pick top 20 code construct *****
Chart c114;
 Group c126;
Stack;
Decreasing;
Transpose;
Exclude;
  Where "C141 =miss()";
wttitle k3;
scale 1;
PSize 6;
grid 2;
Title k3;
subtitle k5;
subtitle k6;
psize 10;
Bar.
erase c100-c141
Enddo
endmacro
```
Example of Current State

28 Day First Pass Failures for XXXXX-XXXX XXX
Radio Counts 14-28 days 2065 7-14 days 1541 2-7 days 1109 1 day 284
Radio Fails 14-28 days 194 7-14 days 189 2-7 days 83 1 day 8

- test name
  - 15.1.2
  - ERROR
  - 15.3.1
  - 15.4.2
  - 15.2.1
  - 15.1.1
  - 50.1.1
  - 15.4.1
  - 15.2.2
  - 15.3.2
  - 16.1.1
  - 40.1.2
- FAIL BY USER
- FAILED

Count

timeframe
- d 1 day
- c 2 - 7 days
- b 7 - 14
- a 14 - 28
5) 7, 3, and 1 day Failure Pareto Charts

Failure data does not always exist

Start the routine by finding the maximum age of failure data

Use **If** statement to transfer control past the code if no failures are present for the timeframe

```
maximum 'delta' k78.

if k78>=-7

*** Code including do loops to identify and work through combinations that actually have data ***

endif
```
Example of Current State – First Pass

7 Day First Pass Failures for XXXXX-XXXX_XXX
7 day Radio count 1393  7 day Radio fails 91

test name
ERROR
15.1.2
50.1.1
15.1.1
15.4.2
15.3.1
15.2.1
16.1.1
15.4.1
40.1.2
15.3.2
15.2.2
22.1.2
22.2.2
22.3.2
FAIL BY USER

Count
6) Nest Checks for all Test Stations

- Testing electronics uses cables and connectors
- Cables and connectors wear out
- Deterioration in cables and connectors is not always visible
- Stations testing general product should exhibit the same yield
- P-charts of yield by station test position should exhibit control around a common value
- Positions with significantly different yield values should be examined for wear

This is a by product of having the data available
Summary of ‘Tricks’ for Productivity in Macros

- **Do loops** can be used in Minitab macros to perform the same analysis for multiple similar conditions. Setting the index based only on combinations that actually have data is an error control strategy.
- **Constants** can be defined in Minitab macros and reused.
- Macros can call other macros.
- Let \( k1 = c1[n] \) will **assign** the value in column 1 row \( n \) to a constant \( k1 \).
- You can open and close layout in a macro to create structured output.
- You can **name graphs** in the graph manager with `wtitle` for easy identification.
- **If statements** allow you to control program flow if no data available.
- **Brief** controls the amount of printed output in the session window.
- **Tally** can be used to identify the top 20 defects along with conversion tables.

**Note:** Some command language changes in Minitab 17 are not compatible with previous versions.
‘Helpful’ Hints

- **HELP > Help** can be used to get to the reference section for Session Commands
- The Alphabetical list shows all the Minitab commands
- Within a command you can find the available subcommands that can be used to control options
What Now?

• You have seen the learning curve develop over 9 years

• You have the information to create simple global macros

• You have the details to use Task Scheduler for executing Minitab macros

• You have specific code examples in case you wish to use any of the methods seen here

• Using the lessons I’ve learned, you can make your macros modular, self correcting, and more efficient from the very start!

What repetitive tasks do you perform that can be done quicker with a macro?