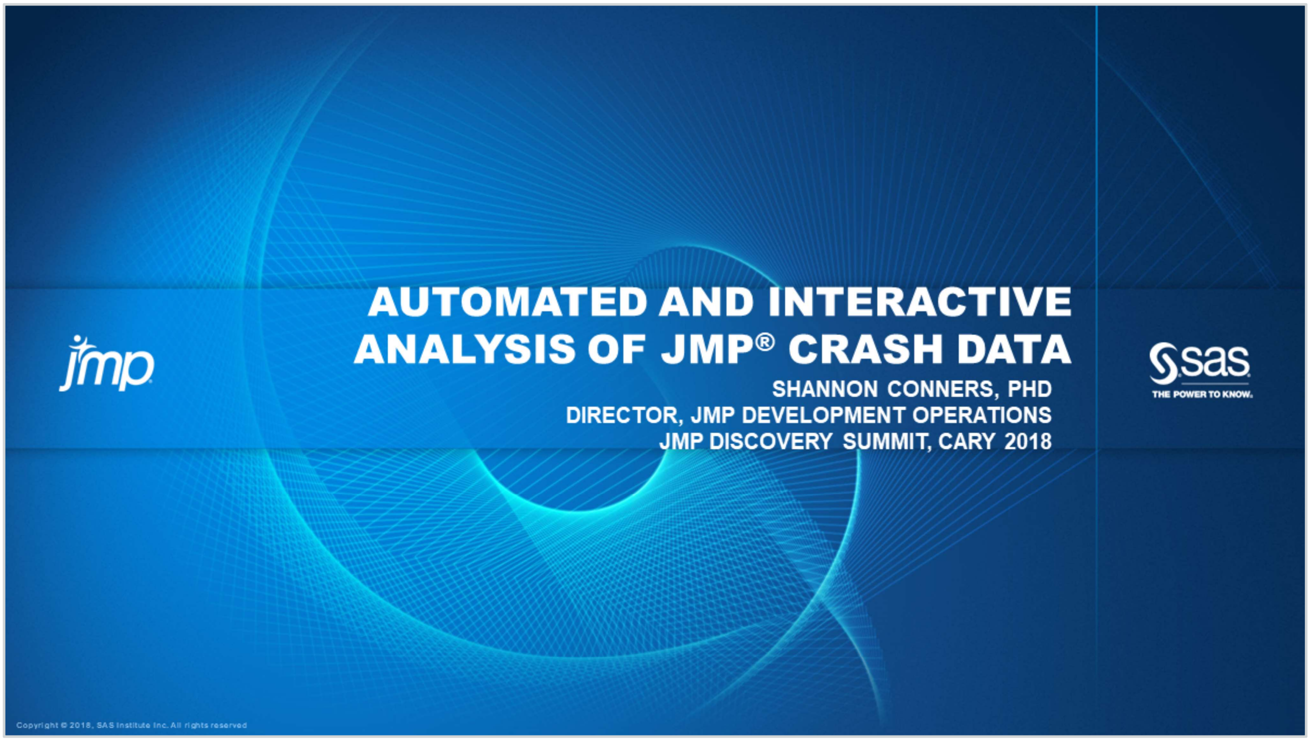


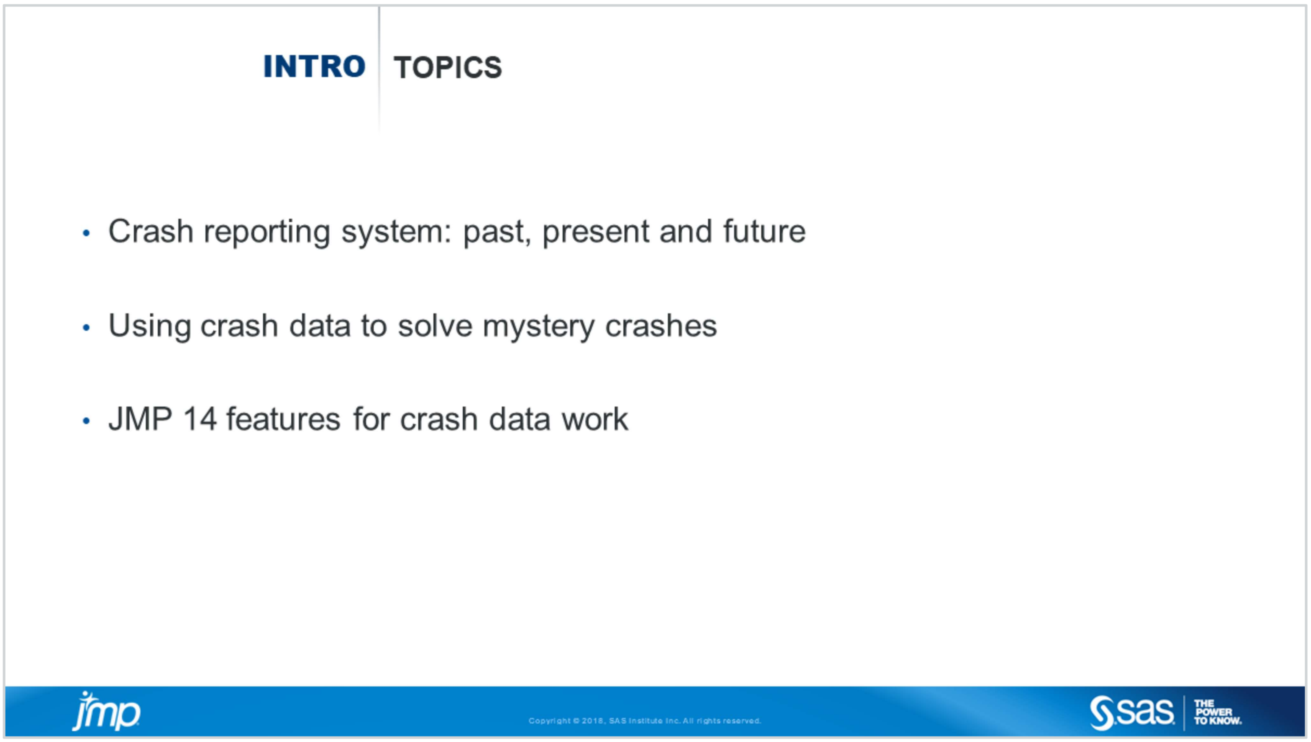
AUTOMATED AND INTERACTIVE ANALYSIS OF JMP® CRASH DATA

SHANNON CONNERS, PHD
DIRECTOR, JMP DEVELOPMENT OPERATIONS
JMP DISCOVERY SUMMIT, CARY 2018

Hi everyone, thanks so much for coming to my talk! If you've seen talks I've given at past Discovery conferences, I've shared personal data projects relating to visualization of my diet and fitness data. More recently, I've begun to share about some of the data projects I've been involved in here at JMP.

Last year in St Louis I talked about how I use one of my favorite hidden features in JMP- the custom regex editor in text explorer-to parse details out of build and test logs in an effort to improve system performance and troubleshoot issues as we build JMP throughout the day. And I presented on our crash system in poster form in Frankfurt this spring. But quite a lot has happened since then, so I'm glad to get the chance to talk to you today about the system that we use to collect data on software crashes and what we are doing with the data.

If you stopped by my poster in Frankfurt or happen to work for JMP, then you already know that I've become a bit obsessed with our crash data over the past few releases. Crashes are data artifacts of the software development process and we continue to collect information about them even once the software has been released to the field. Customer bug and crash reports provide precious insights into our customers' experience of JMP product quality, and we have learned a lot from closely examining them.

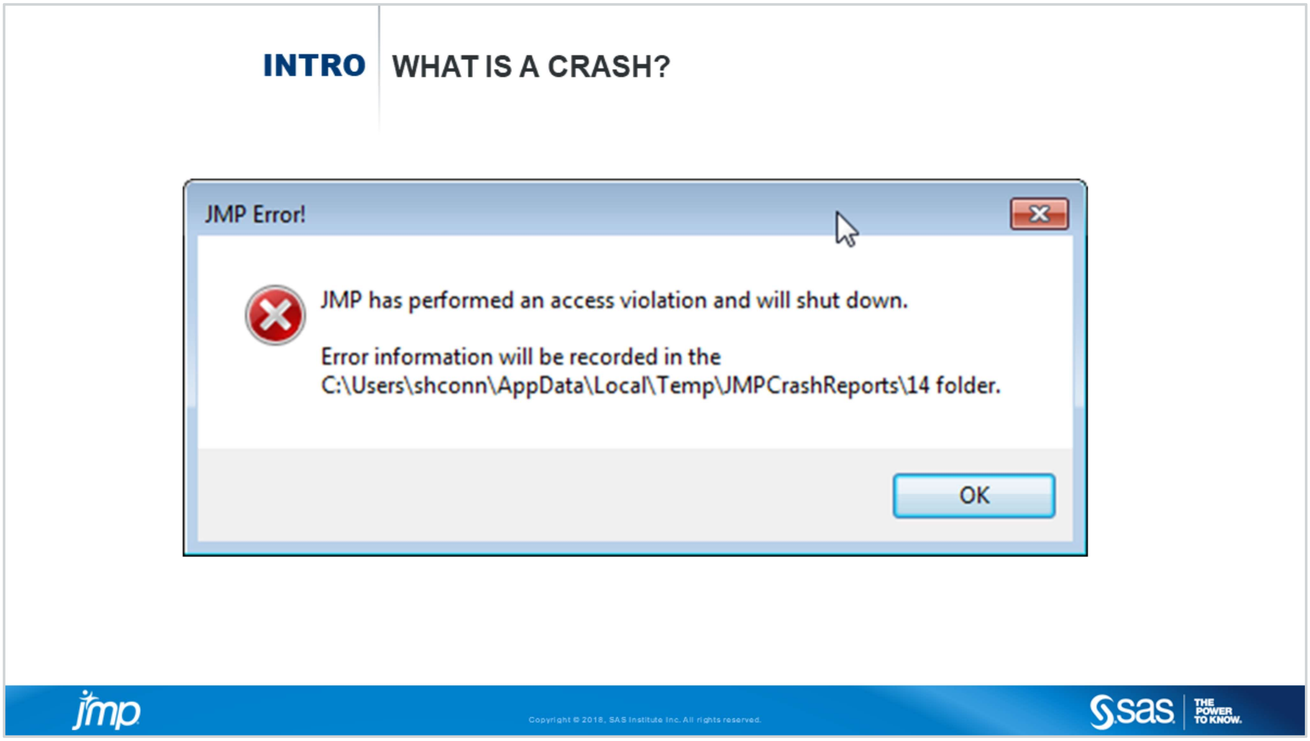


INTRO TOPICS

- Crash reporting system: past, present and future
- Using crash data to solve mystery crashes
- JMP 14 features for crash data work

Today we will talk about the past, present and future of JMP's crash data collection efforts. We will cover some examples of how we are using this data to solve customer mystery crashes. I also would like to take a few minutes to highlight several JMP 14 features that I use when working with our crash data.

Like many of JMP customers, our development team is simultaneously developing new products, new releases, maintenance releases, and receiving problem reports from releases already out in the field. Unfortunately, we do not catch everything in testing prior to release, and so we aim to identify those problems and offer fixes in the next available release. Crashes are just one of these issues but they're what I will focus on today.


INTRO WHAT IS A CRASH?

JMP Error!

JMP has performed an access violation and will shut down.

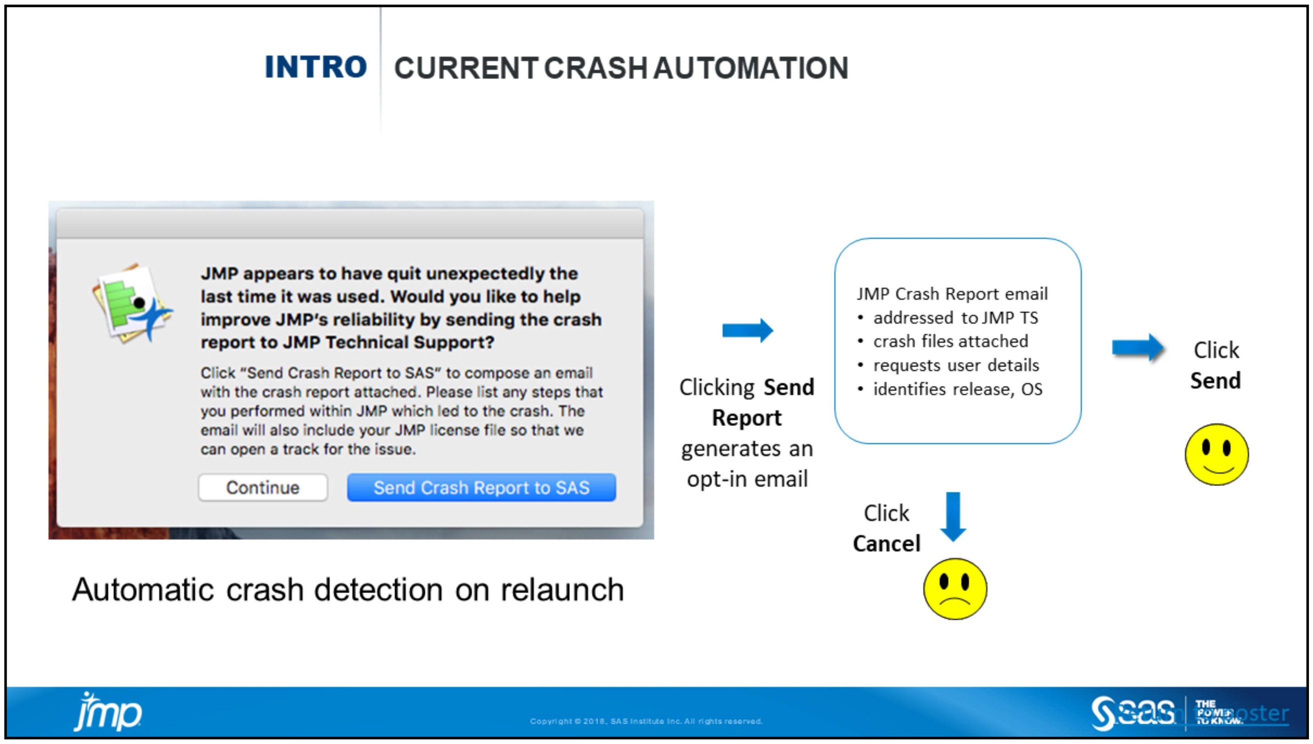
Error information will be recorded in the C:\Users\shconn\AppData\Local\Temp\JMPCrashReports\14 folder.

Just in case you are sitting there wondering what I am talking about… When I say “JMP crash,” I am referring to when the software suddenly stops working and most of the time, you get a message like this. Occasionally, the software crashes very hard and shuts down too rapidly to even display this message.

You may see this for a variety of reasons:
JMP or your machine may run out of memory
There could be a flaw in JMP software
There could be a problem in the OS software or a third party software package that interacts with JMP

That interaction could be as simple as an extension being loaded when JMP launches or an external program being triggered to scan JMP files during install or usage

From a user perspective, it is difficult if not impossible to know which case you have encountered, and this is why we prompt you to report all crashes so development can investigate them. Once we understand the nature of the crash, hopefully JMP TS can guide you to understand what happened and provide a work-around if one is available.



INTRO | CURRENT CRASH AUTOMATION

JMP appears to have quit unexpectedly the last time it was used. Would you like to help improve JMP's reliability by sending the crash report to JMP Technical Support?

Click "Send Crash Report to SAS" to compose an email with the crash report attached. Please list any steps that you performed within JMP which led to the crash. The email will also include your JMP license file so that we can open a track for the issue.

Automatic crash detection on relaunch

Clicking **Send Report** generates an opt-in email

JMP Crash Report email
- addressed to JMP TS
- crash files attached
- requests user details
- identifies release, OS

Click **Send**

Click **Cancel**

Copyright © 2018, SAS Institute Inc. All rights reserved.

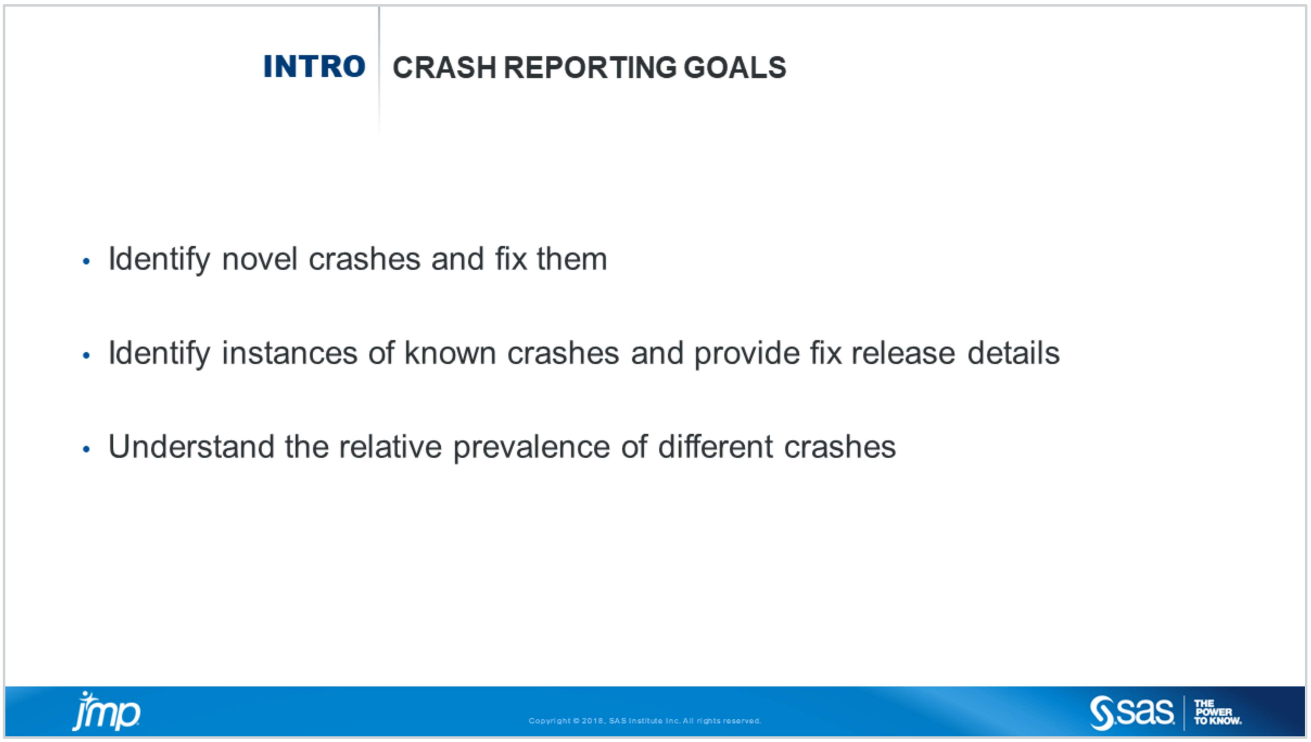
JMP introduced an email mechanism for Mac crash reporting in JMP 12 thanks to work by Michael Hecht, and JMP 13 added emailing for crash reports for Windows thanks to John Schroedl.

The way it works is this: If JMP shuts down after a crash, it saves a set of crash files. Upon the next relaunch, the software SHOULD detect the presence of the crash files, and prompt you to email them. I say "SHOULD" because this does not always happen. Usually, these files will contain traces of what JMP functions were running when the crash occurred. Of course, you don't have to opt in to sending these files, but we hope that you will!

We also prompt for any details you can share about what you were doing before JMP crashed.

And in case you are wondering why we ask for details when you are already sending us your files. Sometimes, the files alone aren't enough to help us replicate the issue. It's ideal to have detailed instructions for replication, but even vaguer clues like "I was closing a set of virtually joined tables" or "dragging variables into dialog for platform X" can be helpful.

Every instance of a related crash has the potential to reveal a little more about the conditions that led to the crash. But the quickest route to a fix is undoubtedly a clear test case.



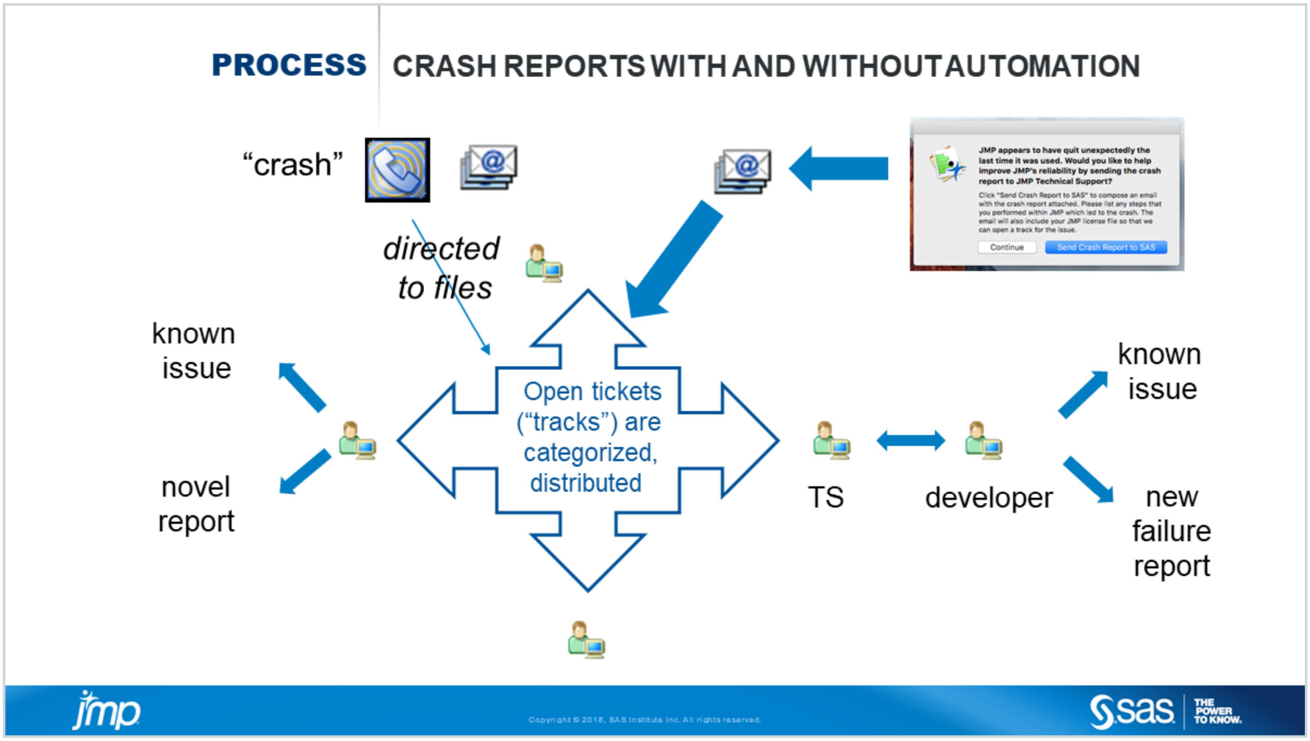
INTRO CRASH REPORTING GOALS

- Identify novel crashes and fix them
- Identify instances of known crashes and provide fix release details
- Understand the relative prevalence of different crashes

The goals of our crash reporting system are simple-first, we want to be able to detect and solve novel crashes as soon as possible. We use the system internally in the hopes of receiving reports early in development. If we characterize a crash early, we may be able to fix it before a customer ever encounters it. Regrettably, novel crashes do get reported post-release, and we aim to fix them in the next available release.

So finding new crashes is one goal. In the process of comparing a new report to known crashes, we may determine that we already know about an issue and have already fixed it in a later release. If so, we want to provide those details to a customer so that they can update JMP to get the fix. For example, if you are working in JMP 13.0 and encounter a known crash was fixed in 13.2 or 14.0, JMP technical support can share that information after you report it.

We also appreciate the data we receive about the relative prevalence of different crashes. We expect to see crashes in the commonly used areas of the product reported more often. These may be the easiest to solve because the more people who report them, the more likely we are to eventually get that key detail that leads to a replication case. It is harder to solve a rare crash, but we still strive to do so if possible. We know that even rarely reported crashes are disrupting some customers' work.


PROCESS
CRASH REPORTS WITH AND WITHOUT AUTOMATION
"crash"
directed to files
known issue
novel report
Open tickets ("tracks") are categorized, distributed
TS
developer
known issue
new failure report
Continue
Send Crash Report to SAS
Copyright © 2016, SAS Institute Inc. All rights reserved.

JMP appears to have quit unexpectedly the last time it was used. Would you like to help improve JMP's reliability by sending the crash report to JMP Technical Support?

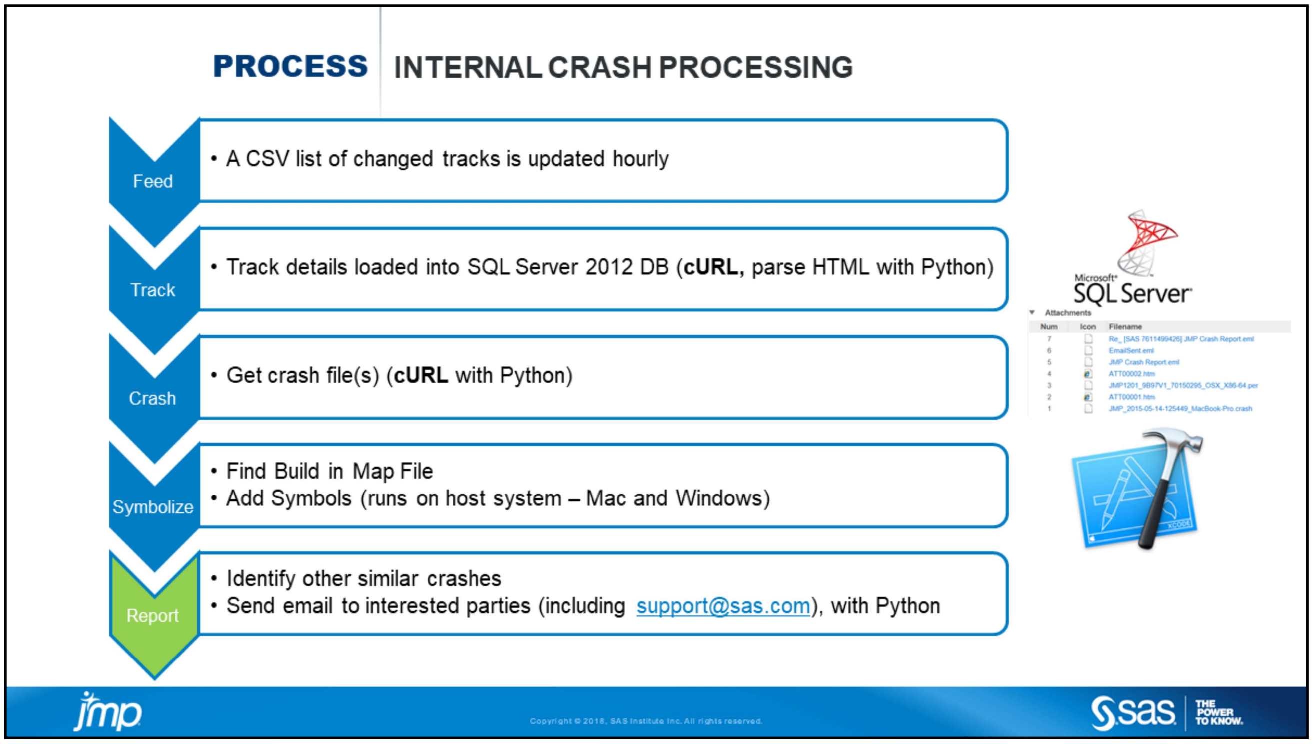
Click "Send Crash Report to SAS" to compose an email with the crash report attached. Please list any steps that you performed within JMP which led to the crash. The email will also include your JMP license file so that we can open a track for the issue.

I wanted to give you a little detail about how we handle crash reports.

We're focusing on crashes here, but of course they are only one of the problems our customers experience. Like many companies, we use a centralized system for handling all customer issues. A JMP team within SAS technical support works with customers who call or email, and tries to identify the problem.

1) Known issue, → the TS team member gives fix release or a workaround.
2) Novel issue → can replicate the problem → failure report for development.
3) Novel issue → can't replicate the problem → provide details to development, hoping they can replicate the problem and enter a new failure report.

Prior to adding automatic crash detection features, software crash reports flowed through the same workflow. A customer indicated they had a crash, the TS team member told them where to find the files on their system, and directed them to email the files back to the track. Hopefully the customer/team member/developer could replicate the issue, determine the root cause, offer a work-around, and enter a new failure report to be addressed in the next appropriate release. But we didn't get a lot of reports, maybe only a handful a week. Once the release with crash automation launched, we still received some reports the old way from direct interactions with a TS team member, but more reports began to come in via email.



PROCESS | INTERNAL CRASH PROCESSING

- **Feed:** A CSV list of changed tracks is updated hourly
- **Track:** Track details loaded into SQL Server 2012 DB (**cURL**, parse HTML with Python)
- **Crash:** Get crash file(s) (**cURL** with Python)
- **Symbolize:**
  - Find Build in Map File
  - Add Symbols (runs on host system – Mac and Windows)
- **Report:**
  - Identify other similar crashes
  - Send email to interested parties (including support@sas.com), with Python

Attachments

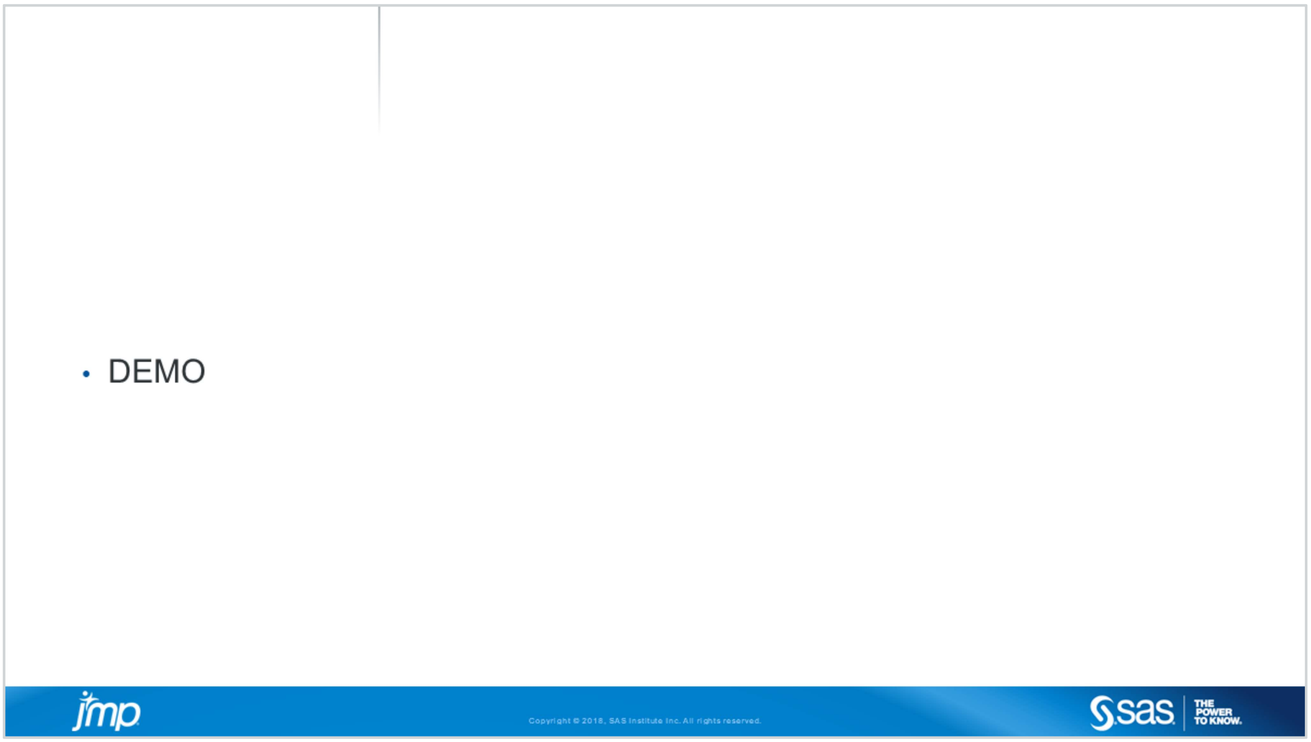
Num	Icon	Filename
7		Re_ [SAS 7611499426] JMP Crash Report.eml
6		EmailSent.eml
5		JMP Crash Report.eml
4		ATT00002.htm
3		JMP1201_9B97V1_70150295_OSX_X86-64.per
2		ATT00001.htm
1		JMP_2015-05-14-125449_MacBook-Pro.crash

Over time, via collaboration between devops and development, we developed the following pipeline to handle crash data received via this email-based mechanism.

In order to process the crashes quickly, we needed to know that they had arrived. We don't own the mission critical database underlying our technical support system so querying it directly was out of the question. We worked with IT to get an hourly feed of all JMP product tracks that had changed within the last 24 hours, and added any new or changed tracks to a list to process.

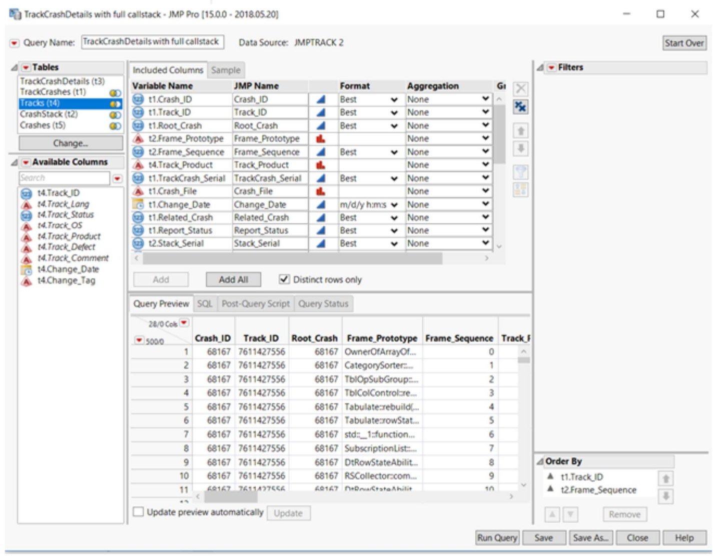
The system curled the text of the tracks to check for crash file attachments, and if so, copied the file to a location where they could be processed in hourly batches.

We added basic reporting into the process-an email back to the track with the symbolized stack in a text file that provided some details about the crash, and some info about the crash.



- DEMO

FEATURES | RETRIEVING CRASH DATA WITH QUERY BUILDER



TrackCrashDetails with full callstack - JMP Pro [15.0.0 - 2018.05.20]

Query Name: TrackCrashDetails with full callstack   Data Source: JMPTRACK 2

Tables: TrackCrashDetails (t3), TrackCrashes (t1), Tracks (t4), CrashStack (t2), Crashes (t5)

Available Columns: t4.Track_ID, t4.Track_Lang, t4.Track_Status, t4.Track_OS, t4.Track_Product, t4.Track_Defect, t4.Track_Comment, t4.Change_Date, t4.Change_Tag

Included Columns:

Variable Name	JMP Name	Format	Aggregation
t1.Crash_ID	Crash_ID	Best	None
t1.Track_ID	Track_ID	Best	None
t1.Root_Crash	Root_Crash	Best	None
t2.Frame_Prototype	Frame_Prototype		None
t2.Frame_Sequence	Frame_Sequence	Best	None
t4.Track_Product	Track_Product		None
t1.TrackCrash_Serial	TrackCrash_Serial	Best	None
t1.Crash_File	Crash_File		None
t1.Change_Date	Change_Date	m/d/y h:m:s	None
t1.Related_Crash	Related_Crash	Best	None
t1.Report_Status	Report_Status	Best	None
t2.Stack_Serial	Stack_Serial	Best	None

✓ Distinct rows only

Query Preview (28/0 Cols, 500/0):

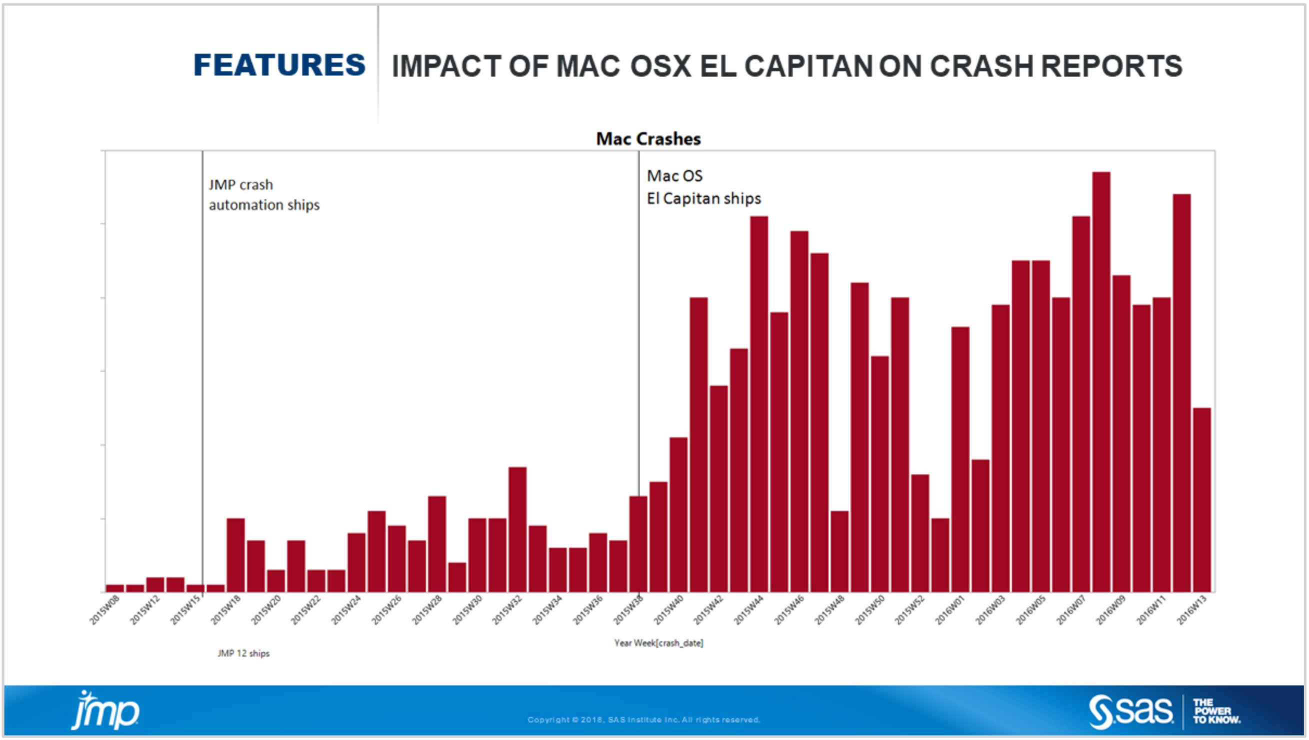
	Crash_ID	Track_ID	Root_Crash	Frame_Prototype	Frame_Sequence
1	68167	7611427556	68167	OwnerOfArrayOf...	0
2	68167	7611427556	68167	CategorySorter::...	1
3	68167	7611427556	68167	TblOpSubGroup::...	2
4	68167	7611427556	68167	TblColControl::re...	3
5	68167	7611427556	68167	Tabulate::rebuild(...	4
6	68167	7611427556	68167	Tabulate::rowStat...	5
7	68167	7611427556	68167	std::__1::function...	6
8	68167	7611427556	68167	SubscriptionList::...	7
9	68167	7611427556	68167	DtRowStateAbilit...	8
10	68167	7611427556	68167	RSCollector::com...	9
11	68167	7611427556	68167	DtRowStateAbilit	10

Order By: t1.Track_ID, t2.Frame_Sequence

I would like to share some of the features in JMP that have been especially helpful to me when working with our crash data over the past several versions.

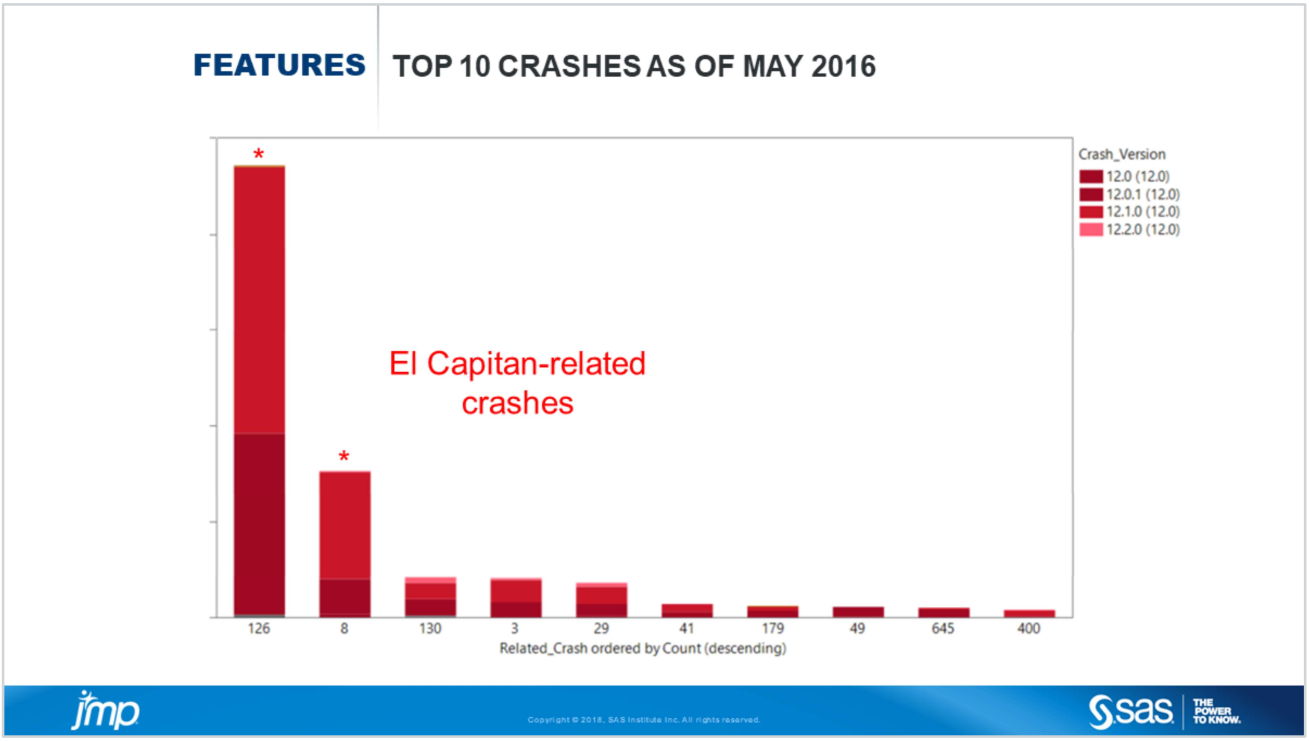
Initially when I got interested in exploring our crash data, I needed to figure out how to get at it, as it was stored in a SQL Server database. I don't write SQL, but I wanted the control to create my own tables without needing to go to the database admin every time I wanted something different. Fortunately, I knew that JMP had a database Query Builder so I learned how to use it and figured out how to add post-query scripts to further manipulate my tables.

I used Query Builder interactively to explore the content of the various tables in the schema and figure out which ones I needed to retrieve and set up the correct joins. Later, when I wanted to automate the data pull, I copied the Query Builder script into my master script.



FEATURES IMPACT OF MAC OSX EL CAPITAN ON CRASH REPORTS

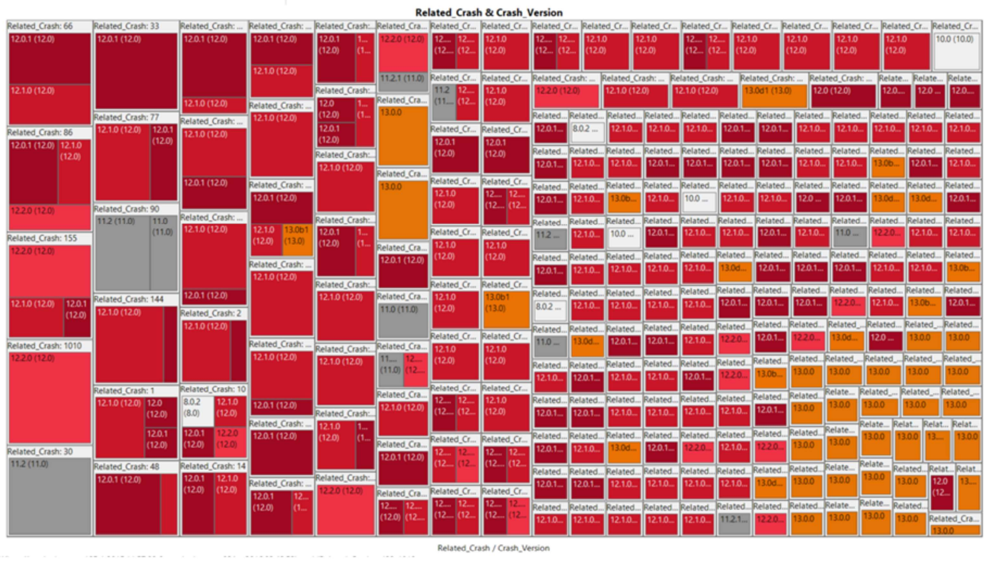
Of course, when I initially began to explore this data, I used Graph Builder to look at trends over time. Here is one view of our early crash data, showing how launching JMP 12 with Mac crash automation increased the number of crashes reported by customers-and how we were impacted by one OS-specific issue when El Capitan launched. We expected to see customer crash reports rise when we introduced automation-but we didn't expect this other big jump in reports. Although we had tested with El Capitan beta releases, we and other applications were surprised by a last minute windowing feature in the production release. In JMP, a very common action-maximizing windows using the green button-caused JMP to crash immediately. We released a fix in 12.2.1, but it's been our most prevalent crash in the database.


FEATURES TOP 10 CRASHES AS OF MAY 2016

Not surprisingly, one of my earliest graphs of our Mac crash data by prevalence showed that the most common crash signatures were the El Capitan crashes-the super prevalent one I mentioned earlier and a related issue. Fortunately, we had a clear direction for customers to update to the release where these common crashes were fixed-12.2.

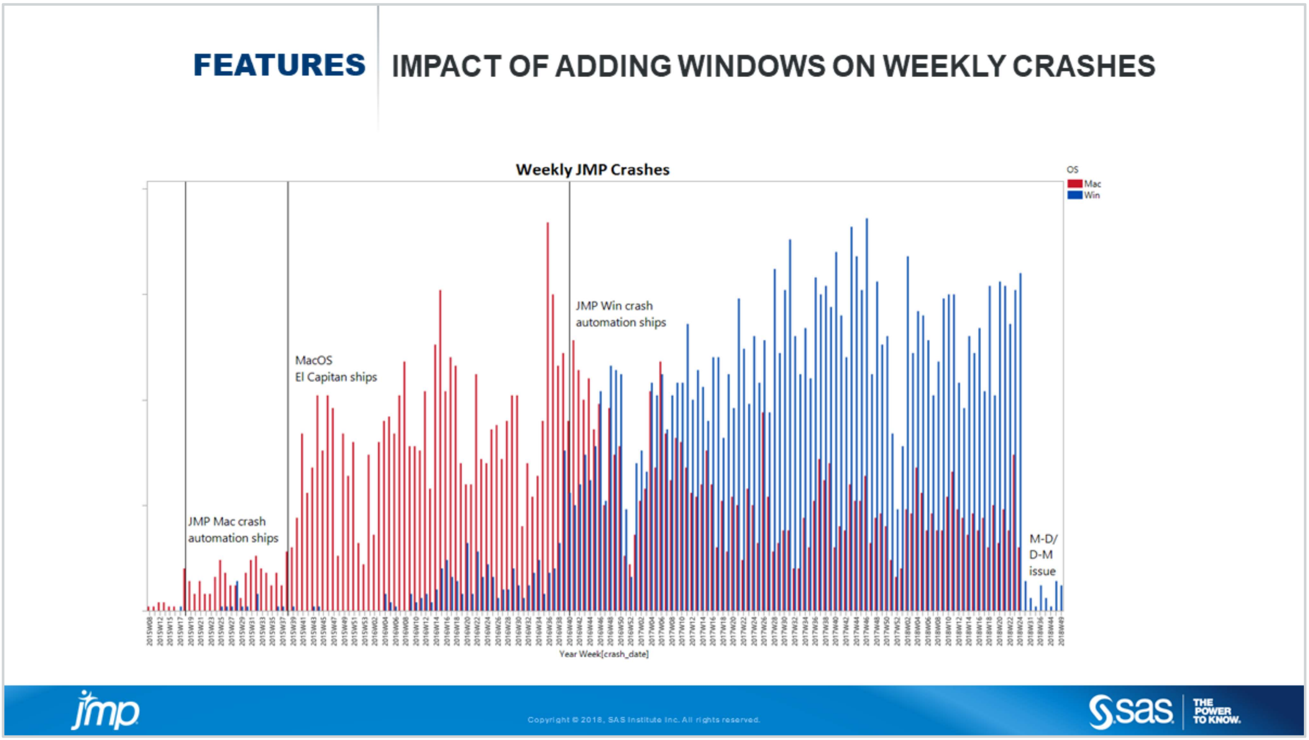
Here I used the stacked bars by release in Graph Builder and a data filter to restrict the timeframe and OS.

FEATURES | TREEMAP IN GRAPH BUILDER



I also like treemaps for visualizing the whole set of crashes by prevalence. This treemap shows what was left after removing the top 10 most prevalent crashes. After a year or so of data collection, nearly 70% of Mac customer crash reports were connected to known software defect reports. But removing the top 10 prevalent crashes left a sea (over 200) of smaller-N cases to investigate. With less data on each one, we had less of a chance that a customer would have given a helpful tip that would let us replicate the problem. But we knew customers were encountering these issues and they needed to be fixed.

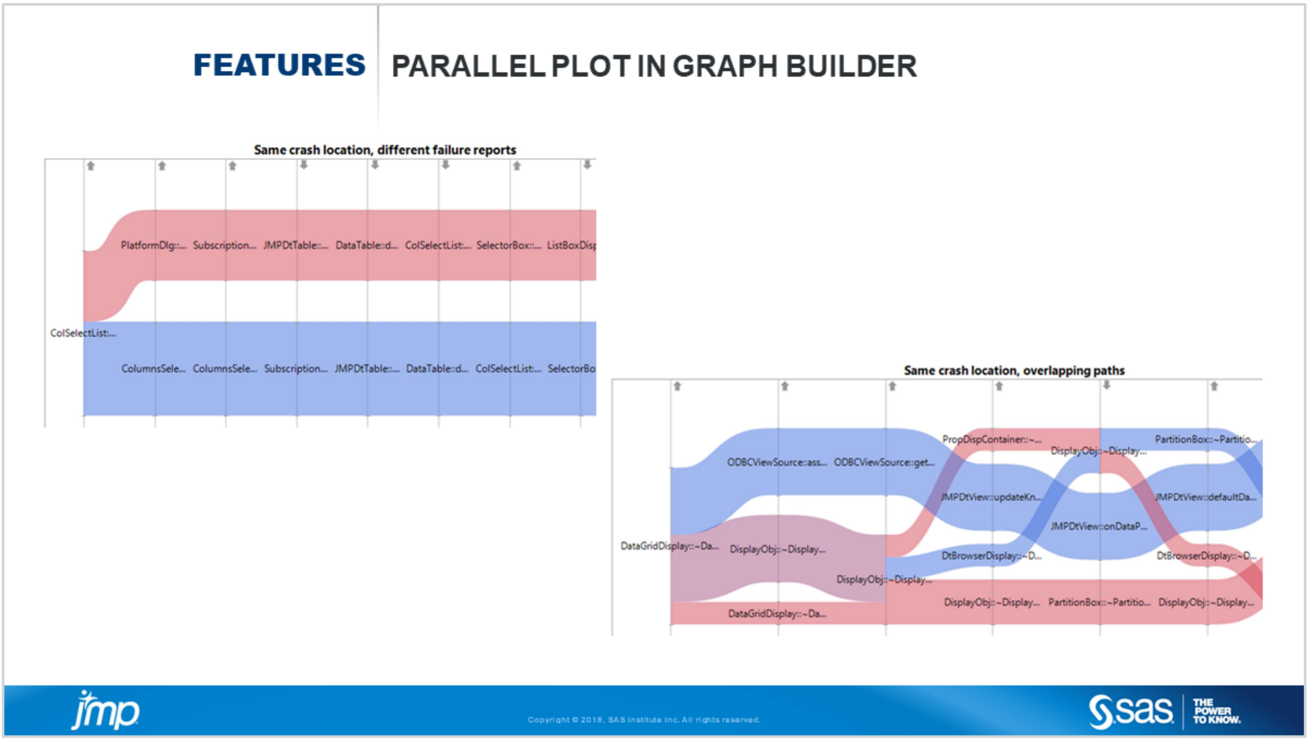
I began to use prevalence information to prioritize sets of cold cases to send to developers to investigate, in the hopes that we could find a reproducible test case or match the problem up with a known issue we had found in testing. I worked with a student to run crashes found during development through the system, in the hopes of matching up known crashes with customer mystery crashes.



FEATURES IMPACT OF ADDING WINDOWS ON WEEKLY CRASHES

During this timeframe, development added emails for Windows crash reporting, and this definitely increased the number of reports we received as we have more JMP customers running on Windows than the Mac. This enabled us to connect more defects to crash signatures.

You may be interested to know that JMP crashes don't always manifest the same-or manifest at all-on both hosts. Some crashes that occur on both hosts, and some are host specific. This is why we try crash cases and their fixes on both hosts.



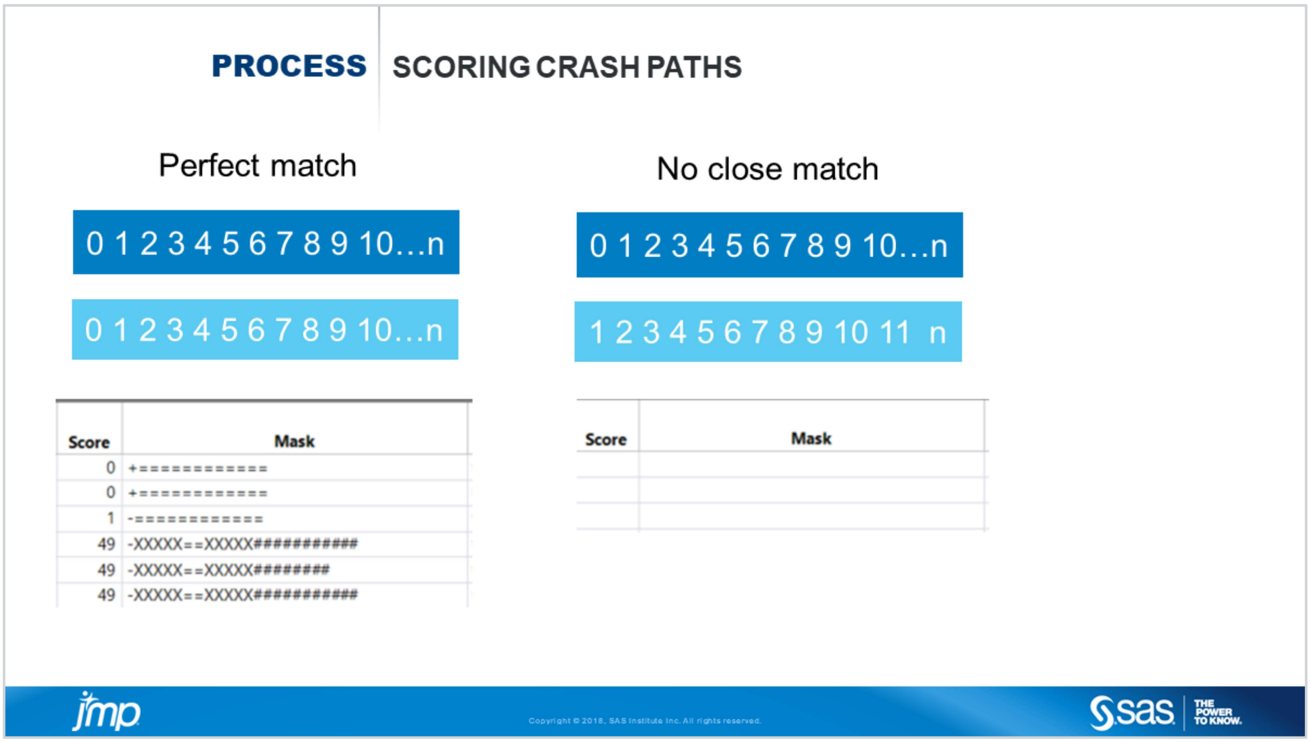
FEATURES PARALLEL PLOT IN GRAPH BUILDER

I used Graph Builder not only to look at aggregated crash data, but to look at and understand the path crashes take through JMP code using parallel plots. I believe these were actually introduced in JMP 13, but a fix I got in JMP 14-truncated labels-made them much nicer for visualizing crash paths.

Our basic crash matching looks at the very last function called by JMP before it crashed. However, there are cases where that simple approach doesn't work as well.

For example, on the left, JMP ends at the same function before it crashes, but takes two completely different paths to get there and different fixes might be required. On the right is a case that is even more confusing. Not only are there multiple paths diverging from a single top frame, but they overlap to some extent and are associated with different software problem reports.

As our database grew, we needed a better way to compare crash paths and automate those comparisons.



PROCESS SCORING CRASH PATHS

Perfect match

0 1 2 3 4 5 6 7 8 9 10…n

0 1 2 3 4 5 6 7 8 9 10…n

Score	Mask
0	+=============
0	+=============
1	-=============
49	-XXXXX==XXXXX############
49	-XXXXX==XXXXX#########
49	-XXXXX==XXXXX############

No close match

0 1 2 3 4 5 6 7 8 9 10…n

1 2 3 4 5 6 7 8 9 10 11 n

Score	Mask

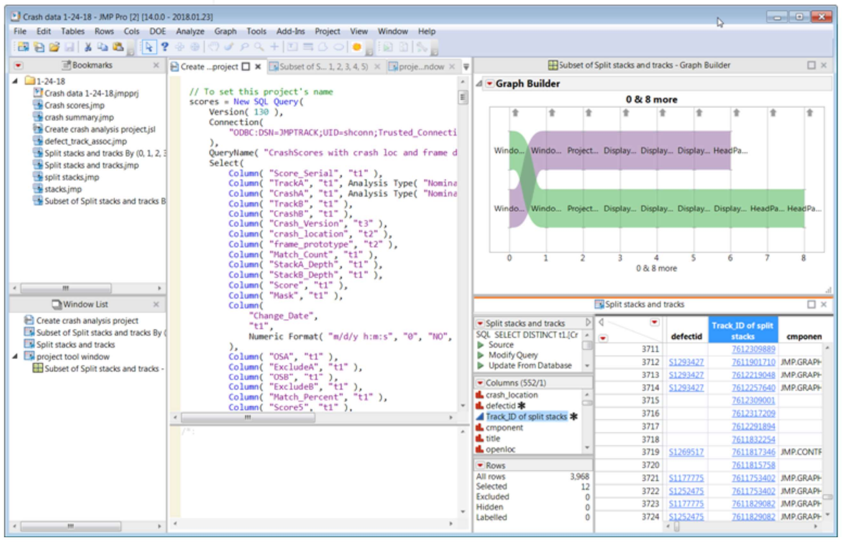
Looking at a path diagrams on a case by case basis was helpful, but we needed a quicker way to screen new crashes against other crashes in the database so I worked with developers to get recommendations for a crash scoring function.

We began scoring crashes against one another by comparing the functions at matching positions in the crash stack and penalizing non-matches. All results that meet a specific threshold are saved in a database table. Scores get recalculated hourly as new crashes come in.

If a crash signature is identical to a known signature attached to a failure report, and is in the right release range, we can conclude with pretty high confidence that they are describing the same issue.

However, if a crash has no low-scoring matches, we start investigating other nearby frames in the stack, searching for other crash signatures that match those, and sending the details to a developer to investigate, and possibly to open a new bug report.

FEATURES CRASH DATA PROJECT


Crash data 1-24-18 - JMP Pro [2] [14.0.0 - 2018.01.23]
File Edit Tables Rows Cols DOE Analyze Graph Tools Add-Ins Project View Window Help
Bookmarks
1-24-18
Crash data 1-24-18.jmpprj
Crash scores.jmp
crash summary.jmp
Create crash analysis project.jsl
defect_track_assoc.jmp
Split stacks and tracks By (0, 1, 2, 3
Split stacks and tracks.jmp
split stacks.jmp
stacks.jmp
Subset of Split stacks and tracks B
Window List
Create crash analysis project
Subset of Split stacks and tracks By (
Split stacks and tracks
project tool window
Subset of Split stacks and tracks -
Create ...project
Subset of S... 1, 2, 3, 4, 5)
proje...ndow
Subset of Split stacks and tracks - Graph Builder
Graph Builder
0 & 8 more
Windo... Windo... Project... Display... Display... Display... HeadPa...
Windo... Windo... Project... Display... Display... Display... Display... HeadPa... HeadPa...
0 1 2 3 4 5 6 7 8
0 & 8 more
Split stacks and tracks

```
// To set this project's name
scores = New SQL Query(
    Version( 130 ),
    Connection(
        "ODBC:DSN=JMPTRACK;UID=shconn;Trusted_Connecti
    ),
    QueryName( "CrashScores with crash loc and frame d
    Select(
        Column( "Score_Serial", "t1" ),
        Column( "TrackA", "t1", Analysis Type( "Nomina
        Column( "CrashA", "t1", Analysis Type( "Nomina
        Column( "TrackB", "t1" ),
        Column( "CrashB", "t1" ),
        Column( "Crash_Version", "t3" ),
        Column( "crash_location", "t2" ),
        Column( "frame_prototype", "t2" ),
        Column( "Match_Count", "t1" ),
        Column( "StackA_Depth", "t1" ),
        Column( "StackB_Depth", "t1" ),
        Column( "Score", "t1" ),
        Column( "Mask", "t1" ),
        Column(
            "Change_Date",
            "t1",
            Numeric Format( "m/d/y h:m:s", "0", "NO",
        ),
        Column( "OSA", "t1" ),
        Column( "ExcludeA", "t1" ),
        Column( "OSB", "t1" ),
        Column( "ExcludeB", "t1" ),
        Column( "Match_Percent", "t1" ),
        Column( "Score5", "t1" ),
```

Split stacks and tracks
SQL SELECT DISTINCT t1.[Cr
Source
Modify Query
Update From Database

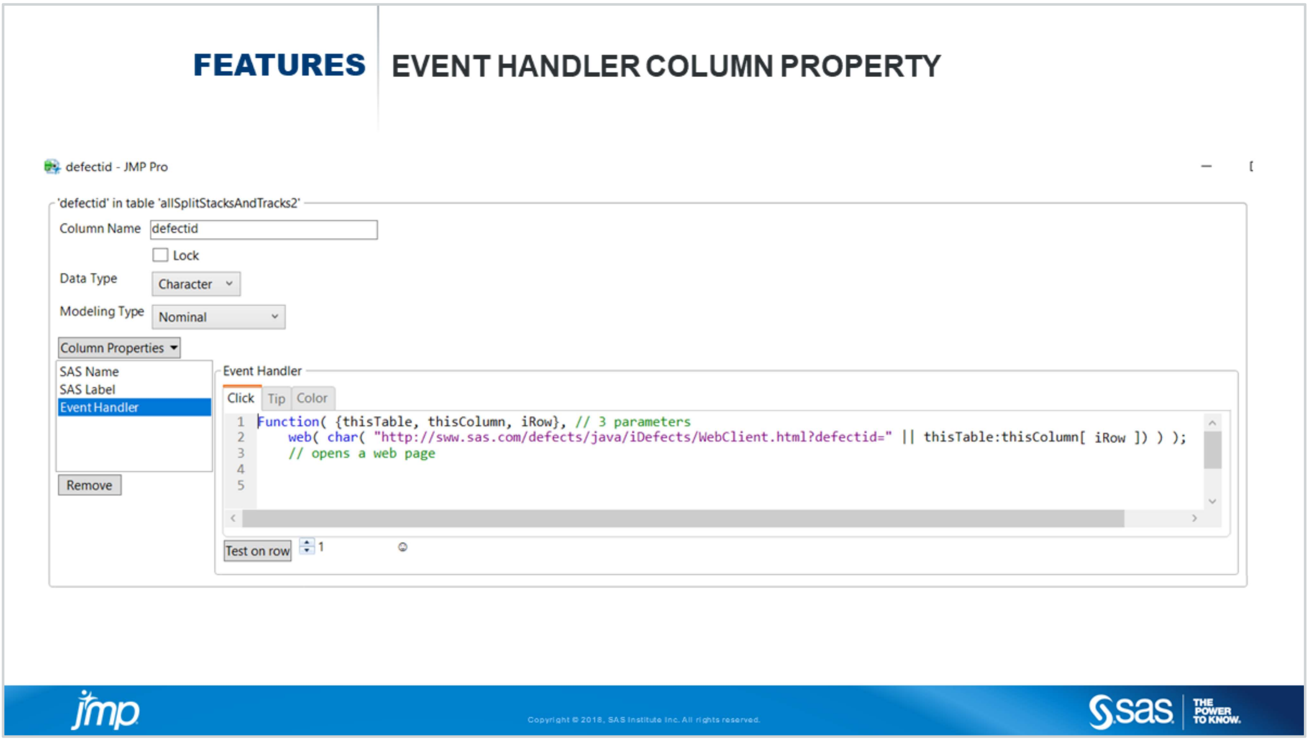
Columns (552/1)
crash_location
defectid ✱
Track_ID of split stacks ✱
cmponent
title
openloc

Rows

All rows	3,968
Selected	12
Excluded	0
Hidden	0
Labelled	0

	defectid	Track_ID of split stacks	cmponen
3711		7612309889	
3712	S1293427	7611901710	JMP.GRAPH
3713	S1293427	7612219048	JMP.GRAPH
3714	S1293427	7612257640	JMP.GRAPH
3715		7612309001	
3716		7612317209	
3717		7612291894	
3718		7611832254	
3719	S1269517	7611817346	JMP.CONTR
3720		7611815758	
3721	S1177775	7611753402	JMP.GRAPH
3722	S1252475	7611753402	JMP.GRAPH
3723	S1177775	7611829082	JMP.GRAPH
3724	S1252475	7611829082	JMP.GRAPH

As I built up a library of queries, tables and graphs during the JMP 14 development cycle, it made a lot of sense to organize them within a JMP 14 project. I could bookmark the files in the folder and open or close them as needed. I could drag and drop tables and graphs to create the layout I needed. And I had a script that drove the creation of the contents of the project-including query builder data pulls from our crash system and our defects database, table manipulation steps, and inserting JMP scripts into tables. My goal was to be able to run the script to get a new cut of data whenever I needed it. We have batches of crashes processing hourly, so at any given moment, I might want to refresh the data to get the latest set.



FEATURES | EVENT HANDLER COLUMN PROPERTY

defectid - JMP Pro

'defectid' in table 'allSplitStacksAndTracks2'

Column Name: defectid

Lock

Data Type: Character

Modeling Type: Nominal

Column Properties

- SAS Name
- SAS Label
- Event Handler

Remove

Event Handler

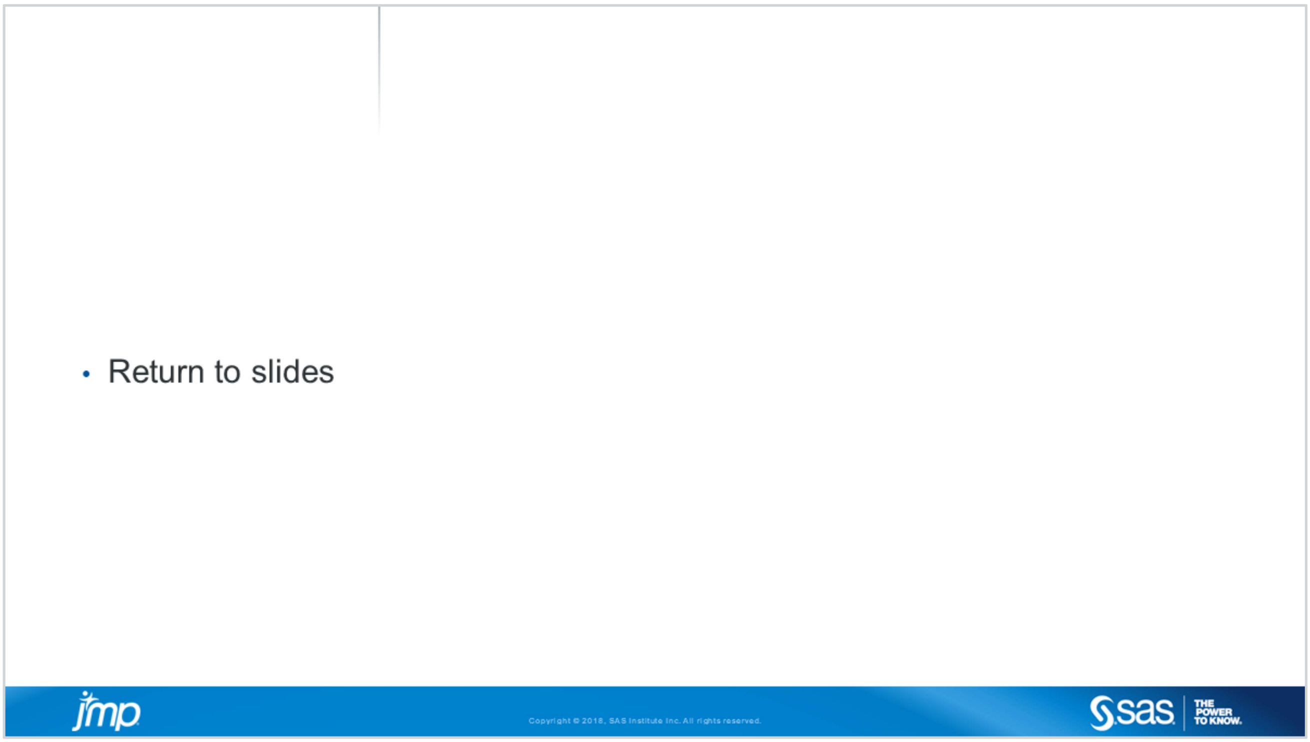
Click | Tip | Color

```
Function( {thisTable, thisColumn, iRow}, // 3 parameters
    web( char( "http://sww.sas.com/defects/java/iDefects/WebClient.html?defectid=" || thisTable:thisColumn[ iRow ]) ) );
    // opens a web page
```

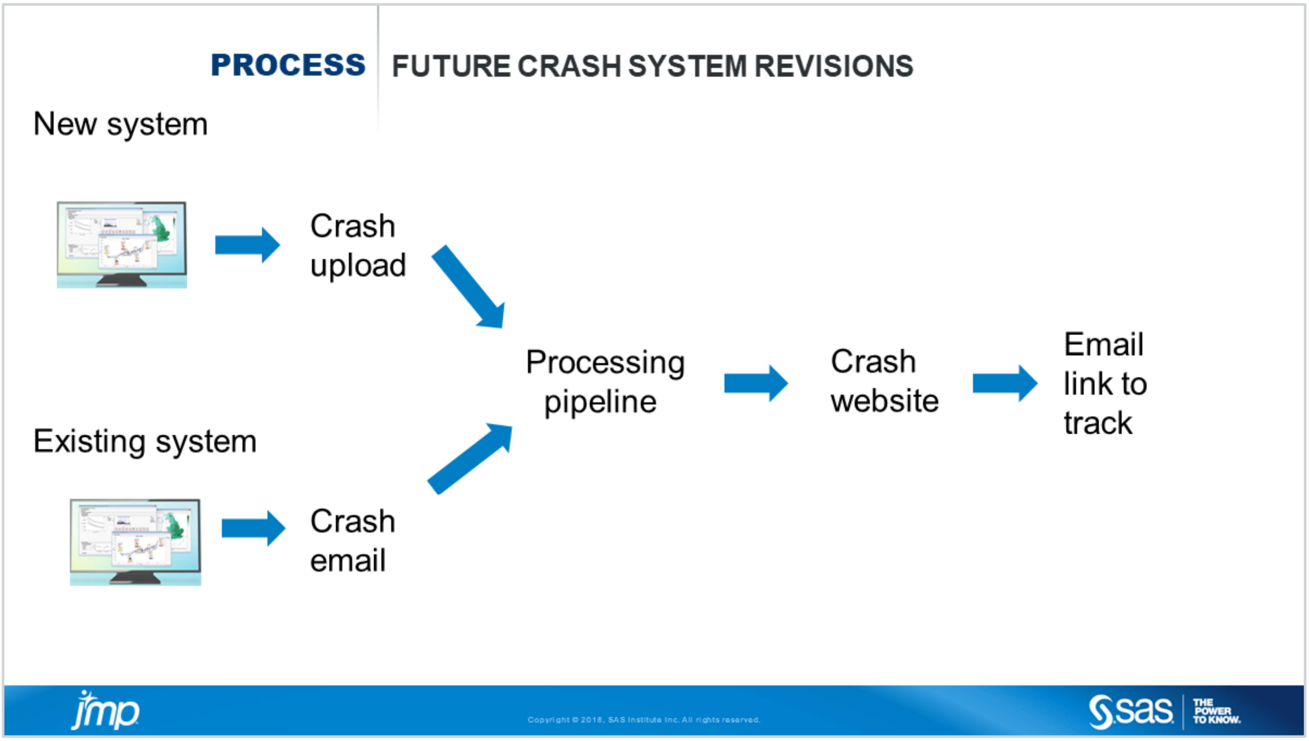
Test on row 1

One of the steps in my script inserted live links into the data table using the new Event handler column property added in JMP 14. Using this property, you can define actions to happen when clicking on a cell, such as running a script, or opening a webpage via a customized link.

With this column property, I could use a formula to create the link and display whatever text I wanted in the cell. I used it to create links out to technical support tracks and our software defects reporting system, which made it easier to move from my JMP tables into other systems that held details that I needed to investigate crashes.



- Return to slides



PROCESS FUTURE CRASH SYSTEM REVISIONS

New system

Crash upload

Processing pipeline

Crash website

Email link to track

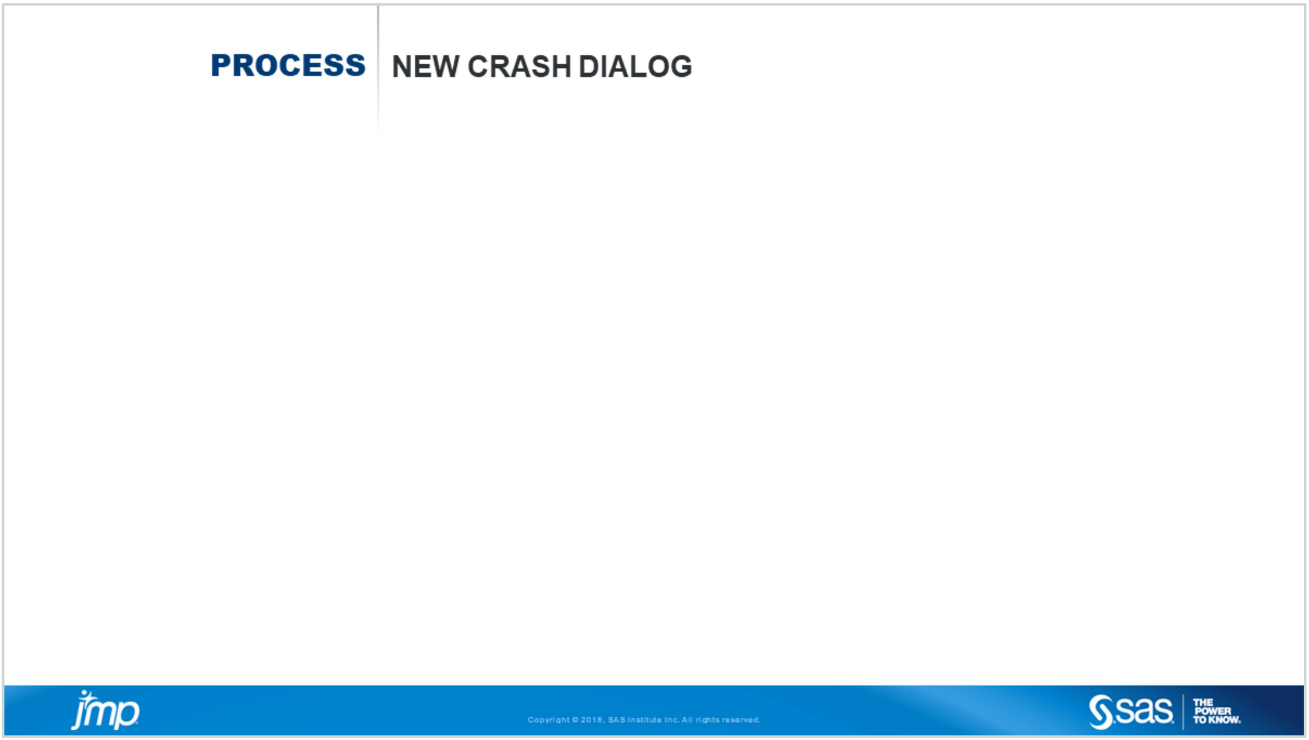
Existing system

Crash email

We will still need to support the existing email based system for older versions, but our new release will allow customers to provide details and upload the crash by communicating with a web API on a listening server. The customer will be able to opt into working with a support team member by providing their email, but they will have the option to submit their reports anonymously. We find that many customers are willing to share information about their crashes, but don't really want the in-depth follow up of working with JMP TS. We'd rather get the crash report than not, and are hoping the anonymous report will generate more responses.

We are also changing the way we report results. If we're allowing anonymous reports, for those cases there will be no JMP TS track to report back to. We will now be processing the reports and delivering results not back to the support system but into a reporting system development maintains. If a support track is requested, then we will email a link to our system back to the track.

I have been working with the JMP developers who created the server based system and a developer on my team (Myron) who has been writing our crash tracking web system to create an interface that can walk team members through the most common tasks that we are doing when investigating a new track.

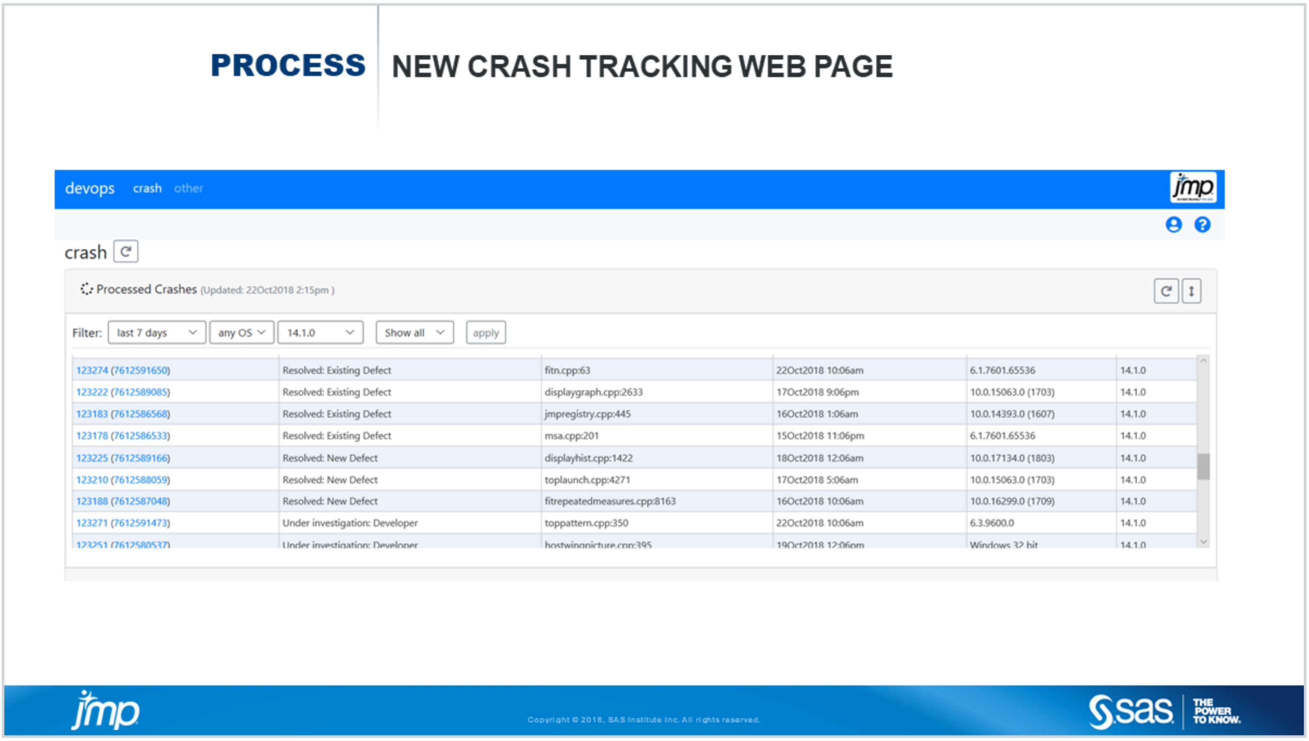


PROCESS NEW CRASH DIALOG

For our next release, development has revised the mechanism by which customers send crash details. While email was a good way to get started, it had disadvantages, primarily email client compatibility issues-e.g., Outlook on the Mac, Windows 10. If you did have a Windows configuration that worked, sending a file would tie up JMP and Outlook till the email was sent.
The most visible change to you will be that for JMP 15, you will no longer be prompted to send us an email, but instead if you crash, you should see a dialog like this.

As you can see, in JMP 15 you will have the option in JMP 15 to submit a crash report anonymously. If you don't specify your email address, your report will not go to JMP technical support or open a track. Please, please send in your crashes whenever you can! We really do use the data to try to improve the stability of the product and appreciate everything you can share about what you were doing when JMP crashed. If you can share anything about what you might have been doing when JMP crashed, that will be very helpful to us, even if you don't have a list of all the steps you took on the way to the crash.

We hope this will encourage customers who don't want to follow up with a TS team member to send their crash files anyway. But it does have repercussions for us internally and the way that we investigate and track crashes. We will no longer have the technical support system to store details of ongoing crashes and their investigations.



PROCESS | NEW CRASH TRACKING WEB PAGE

devops crash other

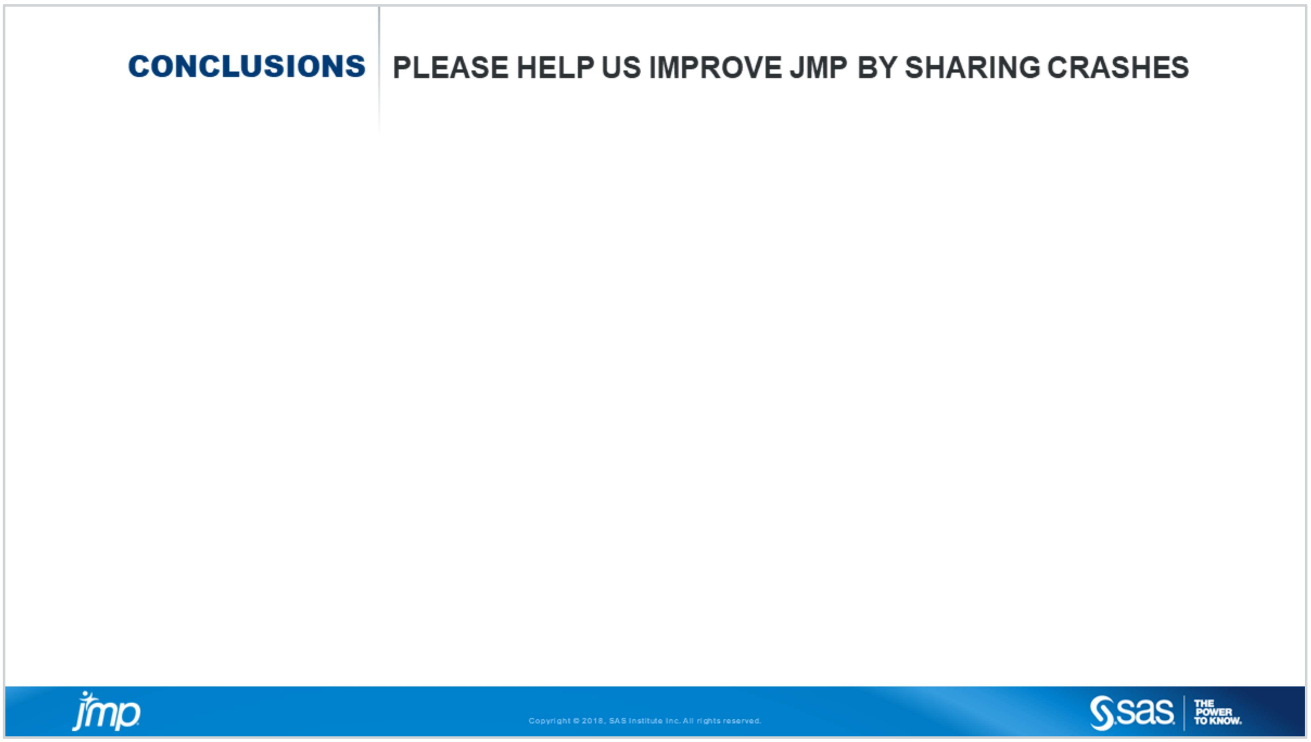
crash

Processed Crashes (Updated: 22Oct2018 2:15pm)

Filter: last 7 days | any OS | 14.1.0 | Show all | apply

123274 (7612591650)	Resolved: Existing Defect	fitn.cpp:63	22Oct2018 10:06am	6.1.7601.65536	14.1.0
123222 (7612589085)	Resolved: Existing Defect	displaygraph.cpp:2633	17Oct2018 9:06pm	10.0.15063.0 (1703)	14.1.0
123183 (7612586568)	Resolved: Existing Defect	jmpregistry.cpp:445	16Oct2018 1:06am	10.0.14393.0 (1607)	14.1.0
123178 (7612586533)	Resolved: Existing Defect	msa.cpp:201	15Oct2018 11:06pm	6.1.7601.65536	14.1.0
123225 (7612589166)	Resolved: New Defect	displayhist.cpp:1422	18Oct2018 12:06am	10.0.17134.0 (1803)	14.1.0
123210 (7612588059)	Resolved: New Defect	toplaunch.cpp:4271	17Oct2018 5:06am	10.0.15063.0 (1703)	14.1.0
123188 (7612587048)	Resolved: New Defect	fitrepeatedmeasures.cpp:8163	16Oct2018 10:06am	10.0.16299.0 (1709)	14.1.0
123271 (7612591473)	Under investigation: Developer	toppattern.cpp:350	22Oct2018 10:06am	6.3.9600.0	14.1.0
123251 (7612580537)	Under investigation: Developer	hostwingpicture.cpp:395	19Oct2018 12:06pm	Windows 32 bit	14.1.0

To support this new approach, a developer on my team, Myron Chandler, has created an internal web system for monitoring, triaging and investigating new customer crashes. It makes use of all that we have learned while looking at our crash data in JMP for the past several years and provides us an interface for tracking data on internal and external crashes.



CONCLUSIONS

PLEASE HELP US IMPROVE JMP BY SHARING CRASHES

I hope I've accomplished two main things in this talk today: 1) given you an idea of what we do with crash data we ask you to share and 2) shown you that JMP has been an integral part of what we do with the data.