The Journey to Solving a Sage Bug
A Story Based on True Events

Bradly Schlenker

June 2, 2014

1 In the beginning

There was nothing... and then I checked out Sage using the following command in the terminal/bash prompt on Sage Math Cloud.

```
git clone git://trac.sagemath.org/sage.git
cd sage
make
```

I waited a few hours and came back to a ready to go Sage Math Cloud terminal command line. What I had done is checked out a steady verison of Sage and compiled the program. (Thus, git clone, changing into the directory where Sage was checked out, and then make-ing/compiling the source) I already knew the bug that I was going to try to solve. It was trac ticket 15178, which should mean nothing to you. To me, it meant that I needed to change my sage checkout to be in the ticket 15178 branch of the Sage git repo. This would allow any changes I make to the Sage source code to be easily uploaded onto trac.sagemath.org. And, not only that but it would also allow for me to get any changes that were already made while editing that ticket and easily pull down changes that others made to the ticket while I was working on it. The nice part of this is that Sage takes care most of the work for you. I simply wrote the following command it prompted me a few questions with yes/no responses.

```
./sage -dev checkout –ticket 15178
```

Once that was complete, I was in the ticket 15178 branch. And, at any time I could easily switch branches, using Sage commands (./sage -dev checkout –branch somebranch), to go back to the master branch or work on another ticket without losing any of my work. What followed next is that I created a patch file to modify the source code. When modifying the source code that is a part of Sage, one can directly edit the source code for most of it. However, Sage also uses a lot of packages that cannot be edited directly. Instead, one has to create patch files that edit the source code of the packages during the build/make process. It's a clever workaround, but it's confusing and the implementation is questionable. It is especially confusing if you don't realize that what you're working on is in a package and not in the Sage src folder. You won't ever find the file you're looking for because it's in a tarball in the upstream folder (Which is not a subfolder of the src folder).

Once I found what I was looking for (a long and arduious journey), I promptly copied the make-d/built version of the file that I was going to be working on and then made my changes on that copy. Once that was done, it was time to create the patch that would describe the changes between the freshly make-d/built version of the file and my edited version. It's important to edit the freshly built file rather than extracting the tarball from the upstream and editing that. It will not work if other patches already exist for this file. And those other patches frequently exist. So, it's important to go to the local/lib/python2.7/site-packages/somePackage directory and edit the file

there. Otherwise, you'll create an improper patch. Anyway, once I had made the changes I made the patch with the following command. It's important to note that the first file is the original built file and the second file is the edited copy of the built file.

```
diff -u pexpect.py pexpect2.py > filename.patch
```

Some people have used diff -ru, but that's for subdirectories (-r being for recursive). If both files are in the same directory then it's unnecessary to have the r in -ru. I created the patch and named it filename.patch. That was one of my many mistakes that I'm glossing over. The way patching works is not clear but it seems to work alphabetically. So, if I put my newest patch in as b.patch but a.patch and c.patch already exist then you're going to have a bad day. If you're lucky, it'll throw an error. If you're unlucky, it'll get past some checks but completely break your code in the long run. The reason for this is that patches need to be applied chronologically in order for them to work as desired. This is because patches can do multiple edits to a file and you don't want to have conflicting edits in your patches. So, I ran into some problems and eventually renamed my filename.patch to zzfilename.patch. That seemed to solve the issue and output the desired result.

Once I had made that patch, I decided to test the package I was working on. In this case, it meant running the first line of the following command and then eventually running the second. (Along with running sage itself to see if it worked)

```
./sage -f pexpect
./sage -b
```

This had disasterious results at first when the patch was called filename.patch but after it was labeled zzfilename.patch life was good. I edited the SPKG.txt for the package I was working on and tried again. It seemed to work and sage ran. I didn't know how to test it for much else and the bug was so simple that I figured I had gotten it correct. Now, it was time for committing the changes I had made to git and then pushing my results onto trac.sagemath.org. Of course, I had already made an account on trac.sagemath.org (have to do this). So I ran the following two commands and answered the questions that followed.

```
./sage -dev commit
./sage -dev push
```

Once I had done that, I went onto trac.sagemath.org and edited the ticket I was working on and labeled it as needing review. That's the beginning and the next step is to get it reviewed and eventually published in the next major release version of Sage.

2 And then there were none

I had published my edits onto trac and they got noticed by a few people. You can view the story yourself here:

http://trac.sagemath.org/ticket/15178

To summarize the story though, it went from me publishing filename.patch online to being shown that it had severe issues. Thus, came in zzfilename.patch and another review. Once another person had seen my edit and some other comments they decided to do a little digging. What we found out is that one of the other patches that existed for this package was causing the bug and had been written by William Stein. Once that was noticed, they declared that the patch that I had worked on be deleted and removed and then that I edit the patch that was causing the issue. I did that and it got committed and pushed onto trac.
In the end, there was another issue which was that the patchbot complained about my fix not passing one the tests. It seemed to be a bug with the patchbot though and not actually my fault. (As one said, "How could the change here be related to this?") Finally, my patch was reviewed again, got a positive review, and the ticket was closed. The problem had been solved.

3 Here I go again

Since I had solved my first bug, I asked William Stein for another one. He forwarded me an email he had just received and told me to create the ticket for the issue. It was a bug that was fresh off the presses! (As opposed to the years old one I had just worked on) Ah, it was an issue with plotting lists of complex numbers on a graph. It couldn't be done with list_plot for some reason.
I created a new ticket with some finicking and got a ticket number of 16378 assigned to the issue at hand. As I did that I wasn't too sure what to do about it all but the person who reported it to William commented on the trac ticket to explain a bit further. I decided to take a dive and try to really get a solid explanation down of what the issue was. I started by replicating the bug in Sage Math Cloud and then looking at what the error said. One line was of imporant detail.

> File "/usr/local/sage/sage-6.2.rc0/local/lib/python2.7/
> site-packages/sage/plot/plot.py", line 1810, in list_plot
> data = [(z.real(), z.imag()) for z in [CC(z[1]) for z in data]]

What that told me is that there was something happening at line 1810 on plot.py that was going severely wrong. I decided to look at the source code and go through the code manually computing the results of what should happen. While working through it, I stumbled upon a function called RDF and I didn't know what that function did. I looked it up and decided to test the part of the code in sage math cloud like it would be used. I ran

```
from sage.rings.all import RDF
data = [1+i,4+i,4+2*i,2+2*i,2+3*i,1+3*i,1+i]
tmp = RDF(data[0])
```

and found it fail miserably. It was giving an error about RDF not liking complex-valued input. This made sense because RDF was about making a special real representation of a given real number. If you gave it a complex-valued number then it wasn't going to be able to make much sense of that since it's not within its means. So, it would throw a TypeError and skip a vital part of the code that needed to be excuted before you arrived at line 1810 (that was the line throwing the original error). Unfortunately, since a important line of code wasn't being excuted then line 1810 would throw an error because the data variable would not be setup properly for usage. There was the problem.
I explained this on the ticket and I got a reply back saying that this functionality was working completely fine on Sage 5.1.2, a different version than what I had been using (Sage 6.2.rc0 on SMC and 6.2). So, something must've changed that broke it from version 5.1.2 to 6.2. The plot thickened! The mystery deepened! Had William Stein striked again!? In order to find this out, I had to download the source of Sage 5.1.2 and build it. This is similar to the process before but now I am using a version that is not recent. After realizing I had ran out of storage space on SMC and deleted a few things, I got the source built and the programmed compiled. I decided to test whether or not it indeed worked in

5.1.2 and it seemed to, but I wasn't sure because the command line interface doesn't return list_plot graphics. I had to take it on faith that it did work. I decided to find the source for the old plot.py and compare it to the new one. Specifically I wanted to compare the list_plot functions and so I did such. I found the source and saw that it was significantly different than the new version. I wrote a comment showing the source code for the function in 5.1.2 and told the person who reported the bug that I did not know how to find out the revision history of the file. Something like that is crucial to find out who caused the bug but more importantly as to why it was caused. It is important to understand why things were changed the way they were so that we can minimize new bugs and issues in the future.

The reporter was able to track down the ticket that the changes were specifically made in and cc'd someone onto the list in order to get the bug fixed. They knew that the someone who had made the change was still an active Sage developer and knew that that person could be asked questions as to why things are the way they are. The someone ended up responding a couple days later and explained the situation. There was some confusion between the original bug reporter and the someone but eventually they came to an agreement about how to do things and what method made the most sense. The eventual fix was tested by the someone and then committed to the branch for that trac ticket for Sage. And, as of writing this paper and this paper being due very soon, there has not yet been a review of the code changes. But, I think there will be a positive review eventually.

Overall, I think what I did was important so that I could help track down the original person who made the changes and have them work through the code because it was of their creation. In most situations, this is a good thing to have because if it's not commented a lot or there's just poor written record of what was going on, it can be best to use their knowledge of what they did and why they did it. If I hadn't done it, this bug could have gone on for a lot longer than it did because the original reporter did not have any Sage 5.1.2 builds up and running. So, they would have had more work to do in order to find out the reason. Simple things like that that can take time but aren't necessarily difficult are big reasons a lot of things don't get done. And so, there we have it. I helped fix two bugs in Sage. I hoped you learned something about the process of working on some open source software. I certainly did.