Getting help

One of the main skills you are going to be called upon for as a data scientist is your ability to solve problems. And sometimes to do that, you need help. The ability to solve problems is at the root of data science; so the importance of being able to do so is paramount. In this lesson, we are going to equip you with some strategies to help you when you get stuck with a problem and need some help! Much of this information has been compiled from Roger Peng's video on "Getting Help" and Eric Raymond's "How to ask questions the smart way" - so definitely check out those resources!

Why is knowing how to get help important?

First off, this course is not like a standard class you have taken before where there may be 30 to 100 people and you have access to your professor for immediate help. In this class, at any one time there can be thousands of students taking the class; no one person could provide help to all of these people, all of the time! So we'll introduce you to some strategies to deal with getting help in this course.

Also, as we said earlier, being able to solve problems is often one of the core skills of a data scientist. Data science is new; you may be the first person to come across a specific problem and you need to be equipped with skills that allow you to tackle problems that are both new to you and to the community!

Finally, troubleshooting and figuring out solutions to problems is a great, transferable skill! It will serve you well as a data scientist, but so much of what any job often entails is problem solving. Being able to think about problems and get help effectively is of benefit to you in whatever career path you find yourself in!

Before you ask for help

Before you begin asking others for help on your problem, there are a few steps you can take on your own. Oftentimes, the fastest answer is one you find for yourself.

One of your first stops for data analysis problems should be reading the manuals or help files (for R problems, try typing ?command) – if you post a question on a forum that is easily answered by the manual, you will often get a reply of "Read the manual" … which is not the easiest way to get at the answer you were going for!

Next steps are searching on Google and searching relevant forums. Common forums for data science problems include StackOverflow and CrossValidated. Additionally, for you in this class, there is a course forum that is a great resource and super helpful! Before posting a question to any forum, try and double check that it hasn't been asked before, using the forums' search functions.

While you are Googling, things to pay attention to and look for are: tutorials, FAQs, or vignettes of whatever command or program is giving you trouble. These are great resources to get you started – either in telling you the language/words to use in your next searches, or outright showing you how to do something.

First steps for solving coding problems

As you get further into this course and using R, you may run into coding problems and errors and there are a few strategies you should have ready to deal with these. In my experience, coding problems generally fall into two categories: your command produces no data and spits out an error message OR your command produces an output, but it is not at all what you wanted. These two problems have different strategies for dealing with them.

If it's a problem producing an error message:

- Check for typos!
- **Read the error message and make sure you understand it**
- Google the error message, exactly

I've been there – you type out a command and all you get are lines and lines of angry red text telling you that you did something wrong. And this can be overwhelming. But taking a second to check over your command for typos and then **carefully** reading the error message solves the problem in nearly all of the cases. The error messages are there to help you – it is the computer telling you what went wrong. And when all else fails, you can be pretty assured that somebody out there got the same error message, panicked and posted to a forum – the answer is out there.

On the other hand, if you get an output, but it isn't what you expected:

- Consider how the output was different from what you expected
- Think about what it looks like the command actually did, why it would do that, and not what you wanted

Most problems like this are because the command you provided told the program to do one thing and it did that thing exactly… it just turns out what you told it to do wasn't actually what you wanted! These problems are often the most frustrating – you are so close but so far! The quickest way to figuring out what went wrong is looking at the output you did get, comparing it to what you wanted, and thinking about how the program may have produced that output instead of what you wanted. These sorts of problems give you plenty of practice thinking like a computer program!

Next steps

Alright, you've done everything you are supposed to do to solve the problem on your own – you need to bring in the big guns now: other people!

Easiest is to find a peer with some experience with what you are working on and ask them for help/direction. This is often great because the person explaining gets to solidify their understanding while teaching it to you, and you get a hands on experience seeing how they would solve the problem. In this class, your peers can be your classmates and you can interact with them through the course forum (double check your question hasn't been asked already!).

But, outside of this course, you may not have too many data science savvy peers – what then?

"Rubber duck debugging" is a long held tradition of solitary programmers everywhere. In the book "The Pragmatic Programmer," there is a story of how stumped programmers would explain their problem to a rubber duck, and in the process of explaining the problem, identify the solution.

Wikipedia explains it well:

> Many programmers have had the experience of explaining a programming problem to someone else, possibly even to someone who knows nothing about programming, and then hitting upon the solution in the process of explaining the problem. In describing what the code is supposed to do and observing what it actually does, any incongruity between these two becomes apparent.

So next time you are stumped, bring out the bath toys!

When all else fails: posting to forums

You've done your best. You've searched and searched. You've talked with peers. You've done everything possible to figure it out on your own. And you are still stuck. It's time. Time to post your question to a relevant forum.

Before you go ahead and just post your question, you need to consider how you can best ask your question to garner (helpful) answers.

How to effectively ask questions on forums

Details to include:

- The question you are trying to answer
- How you approached the problem, what steps you have already taken to answer the question
- What steps will reproduce the problem (including sample data for troubleshooters to work from!)
- What was the expected output
- What you saw instead (including any error messages you received!)
- What troubleshooting steps you have already tried
- Details about your set-up, eg: what operating system you are using, what version of the product you have installed (eg: R, Rpackages)
- Be specific in the title of your questions!

How to title forum posts

Most of these details are self-explanatory, but there can be an art to titling your posting. Without being specific, you don't give your potential helpers a lot to go off of – they don't really know what the problem is and if they are able to help you.

Bad:

- HELP! Can't fit linear model!
- HELP! Don't understand PCA!

These titles don't give your potential helpers a lot to go off of – they don't really know what the problem is and if they are able to help you. Instead, you need to provide some details about what you are having problems with. Answering what you were doing and what the problem is are two key pieces of information that you need to provide. This way somebody who is on the forum will know exactly what is happening and that they might be able to help!

Better:

- R 3.4.3 lm() function produces seg fault with large data frame (Windows 10)
- Applied PCA to a matrix - what are U, D, and Vt?

Even better:

- R 3.4.3 lm() function on Windows 10 – seg fault on large dataframe
- Using principle components to discover common variation in rows of a matrix, should I use, U, D or Vt?

Use titles that focus on the very specific core problem that you are trying to get help with. It signals to people that you are looking for a very specific answer; the more specific the question, often, the faster the answer.

Forum etiquette

Following a lot of the tips above will serve you well in posting on forums and observing forum etiquette. You are asking for help, you are hoping somebody else will take time out of their day to help you – you need to be courteous. Often this takes the form of asking specific questions, doing some troubleshooting of your own, and giving potential problem solvers easy access to all the information they need to help you. Formalizing some of these do's and don't's, you get the following lists:

Do's

- Read the forum posting guidelines
- Make sure you are asking your question on an appropriate forum!
- Describe the goal
- Be explicit and detailed in your explanation
- Provide the minimum information required to describe (and replicate) the problem
- Be courteous! (Please and thank you!)
- Follow up on the post OR **post the solution**

Let's take a few seconds to talk a bit about this last point, as we have touched on the others already. First, what do we mean by "follow up on the post"? You've asked your question and you've received several answers and lo and behold one of them works! You are all set, get back to work! No! Go back to your posting, reply to the solution that worked for you, explaining that they fixed your problem and thanking them for their solution! Not only do the people helping you deserve thanks, but this is helpful to anybody else who has the same problem as you, later on. They are going to do their due diligence, search the forum and find your post – it is so helpful for you to have flagged the answer that solved your problem.

Conversely, while you are waiting for a reply, perhaps you stumble upon the solution (go you!) – don't just close the posting or never check back on it. One, people who are trying to help you may be replying and you are functionally ignoring them, or two, if you close it with no solution, somebody with the same problem won't ever learn what your solution was! Make sure to post the solution and thank everybody for their help!

Don't's:

- Immediately assume you have found a bug

- Post homework questions
- Cross post on multiple forums
- Repost if you don't immediately get a response

These are all pretty clear guidelines. Nobody wants to help somebody who assumes that the root cause of the problem isn't because they have made a mistake, but that there is something wrong with a program. Spoiler alert, it's (almost) always because you made a mistake. Similarly, nobody wants to do your homework for you, they want to help somebody who is genuinely trying to learn – not find a short cut.

Additionally, for people who are active on multiple forums, it is always aggravating when the same person posts the same question on five different forums…. Or when the same question is posted on the same forum repeatedly. Be patient – pick the most relevant forum for your purposes, post once, and wait.

Summary

In this lesson, we look at how to effectively get help when you run into a problem. This is important for this course, but also for your future as a data scientist!

We first looked at strategies to use before asking for help, including reading the manual, checking the help files, and searching Google and appropriate forums. We also covered some common coding problems you may face and some preliminary steps you can take on your own, including paying special attention to error messages and examining how your code behaved compared to your goal.

Once you've exhausted these options, we turn to other people for help. We can ask peers for help or explain our problems to our trusty rubber ducks (be it an actual rubber duck or an unsuspecting coworker!). Our course forum is also a great resource for you all to talk with many of your peers! Go introduce yourself!

And if all else fails, we can post on forums (be it in this class or at another forum, like StackOverflow), with very specific, reproducible questions. Before doing so, be sure to brush up on your forum etiquette - it never hurt anybody to be polite! Be a good citizen of our forums!

There is an art to problem solving, and the only way to get practice is to get out there and start solving problems! Get to it!