# Title Slide

Good afternoon, and thank you for joining me for EDirect Office Hours for April, 2018. This is our first EDirect Office Hours, and we’re very excited to be here today to answer your questions, and maybe teach you a few new EDirect tricks.

# Today’s Agenda

As I said, most of today is going to be about answering your questions, but we’re going to start off with introducing you to an EDirect technique that’s very useful, but that you might not be familiar with, which is using variables in xtract.

Then, we’re going to a have a few brief announcements about some recent and upcoming changes to EDirect.

And the rest of the time is for you!

# Variables: Overview

Before we get to your questions, we wanted to introduce you to what is probably a new topic for many of you, since we do not cover it in EDirect for PubMed, and that’s the use of variables.

It’s a technique that can be very useful for helping you filter your output to only specific information. In fact, those of you who have taken EDirect for PubMed might remember some questions that we didn’t answer, or examples that we didn’t address, saying that they were technically feasible using EDirect, but that they involved some advanced techniques that were out of scope for the class. This is one of those advanced techniques.

The reason we don’t cover this in EDirect for PubMed is that it’s kind of tricky to wrap your head around how variables work, and when you’d want to use them. Even if you’re an experienced programmer, who has used variables with other programming languages, you might find they don’t work exactly the way you expect.

The top-level summary of variables is that they’re a method of storing information in one part of an xtract statement, and retrieving it in another part. However, to really get a handle on what’s going on here, I’m going to start with an example.

# Example

In fact, I’m going to start with an example that you may have seen in EDirect for PubMed, especially if you’ve taken the class more recently.

Say we want to do a search for an author, and analyze the different affiliations associated with that author in PubMed. We can use this to track an author’s career over time, seeing the different places they’ve worked, but it might be even more useful for disambiguation.

As you probably know, PubMed does not do author disambiguation, so it’s up to the user to make sure the results they’re looking at are for the correct author. This is especially true for authors with more common names.

One way you can try to disambiguate authors with common names is by looking at affiliation data: if you know that an author exclusively works in China, for example, then you can probably assume any records that have an author of the same name working in Germany are attributable to a different author.

# One approach…

This is actually a block of code that we showed in EDirect for PubMed, which we used to accomplish a similar task:

esearch –db pubmed –query "smith bh[Author]" \

-datetype PDAT –mindate 2014 –maxdate 2017 | \

efetch –format xml | \

xtract –pattern PubmedArticle –element MedlineCitation/PMID \

–block Author –if LastName –equals Smith \

–and Initials –equals BH \

–element LastName Initials Affiliation

Let’s walk through this code block and see what it’s doing.

First, we search PubMed for “smith bh” tagged with [author]. This is the recommended formulation for PubMed author searching.

We limit our search by date, to articles published between 2014 and 2017.

We use efetch to retrieve XML for all of these records, and send that into xtract.

For this project, we want to look at each record that has an author of BH Smith, and see what affiliation is listed for BH Smith on that record. To start, we’ll have xtract create a new row for each record, and put the PMID in the first column. Then we need to find the affiliation data for the BH Smith attached to each record. We’ll do this using -block.

If you remember the way the Author data is structured in PubMed, for each author on a record we have an Author element, which has child elements of LastName, Initials, and, optionally Affiliation. So we’ll use -block to look at each of these Author elements for the record, one at a time, but only display the author information for the Author element for BH Smith.

As xtract goes through each Author -block, it checks to see if LastName equals Smith and Initials equals BH. If both of these things are true, we’ll output data for that author. This will output LastName and Initials (which we don’t really need, since we know they’ll be Smith BH, but I’ve left them in to demonstrate that the condition is working), followed by Affiliation.

When we run this, we get our list of PMIDs in the first column, Smith and BH in the next to columns, and then the Affiliation data for BH Smith in the last column, if it’s present.

This is good, as far as it goes, but we’ve got a bunch of rows here that don’t have any Affiliation data.

As you may know, Affiliation is not a required field. Up until 1988, there was no Affiliation information in PubMed at all. Up until about five years ago, only the first author’s affiliation was listed.

If we only want to look at BH Smith’s affiliation data, and don’t care about the records where BH Smith has no affiliation data, can we exclude these other records?

Probably the most obvious way would be to get this output into another program (like Excel), either by saving it to a file or copying and pasting. We could then use Excel’s filtering tools to filter out any row with a blank fourth column. However, we’re here to learn about EDirect, so let’s see if we can accomplish this without using Excel.

# Second Try

If we only want to see records where BH Smith has an affiliation, we could AND in another condition…

esearch –db pubmed –query "smith bh[Author]" \

-datetype PDAT –mindate 2014 –maxdate 2017 | \

efetch –format xml | \

xtract –pattern PubmedArticle –element MedlineCitation/PMID \

–block Author –if LastName –equals Smith \

–and Initials –equals BH -and Affiliation \

–element LastName Initials Affiliation

We’d like this to make it so we only see output for an author when the LastName is Smith, Initials are BH, and there’s Affiliation data for that author. But, when we execute it, we’re still seeing rows with a PMID and nothing else. Putting that condition in the inside the -block prevented us from seeing the LastName and Initials if there wasn’t any Affiliation data, but didn’t prevent us seeing the PMID for the records where that was true.

# Third Try

We could try putting the condition on the pattern, rather than in the block. But, as you’re about to see, that’s also not going to cut it.

esearch –db pubmed –query "smith bh[Author]" \

-datetype PDAT –mindate 2014 –maxdate 2017 | \

efetch –format xml | \

xtract –pattern PubmedArticle –if Affiliation \

–element MedlineCitation/PMID \

–block Author –if LastName –equals Smith \

–and Initials –equals BH \

–element LastName Initials Affiliation

This problem with this condition is it only excludes records if there are no Affiliation elements at all. So all of these unwanted rows still show up, because those records have Affiliation elements, just not for BH Smith.

What we really want to do is put the PMID as part of that last -element argument, inside our -block.

Unfortunately, once we’re in the -block, we can only access data that is part of that block. PMID is not a child or descendant of author, so we can’t pull just pull the PMID here.

# This should do it!

But there is a way we can accomplish this. As you may have suspected, we can do it using variables.

esearch –db pubmed –query "smith bh[Author]" \

-datetype PDAT –mindate 2013 –maxdate 2017 | \

efetch –format xml | \

xtract –pattern PubmedArticle –VAR1 MedlineCitation/PMID \

–block Author –if LastName –equals Smith \

–and Initials –equals BH -and Affiliation \

–element "&VAR1" LastName Initials Affiliation

This time, in our xtract command, rather than using the -element argument to output the PMID for each row, we’re actually saving that PMID into a variable, named “VAR1”.

We do that by making up a new argument with the name of the variable we want to use: in this case, VAR in all caps, and the number 1. This puts the PMID for this -pattern into a variable named VAR1.

It’s important to note that we could call this variable almost anything. I’m calling it VAR1 to avoid confusion, but we could call it VARIABLE in all caps, UID, we could actually call the variable PMID, we could even call the variable BOB if we wanted to. It just has to be all capital letters and digits.

Once we have the PMID in the variable, we start our -block. For each record, we go through each author until we find one that has a LastName of “Smith”, has the Initials “BH”, and has affiliation data. When we find an author that meets all of our conditions, we will use the -element argument to output data.

The first thing we’re going to output is the contents of our variable VAR1. We do this by using quotes and ampersand: “&VAR1”. Again, we ordinarily couldn’t just pull the PMID in here, since it’s not a child of Author, but we saved it into a variable outside the block, as part of our -pattern PubmedArticle. Then, we’ll output LastName, Initials and Affiliation.

When we look at our output, we see that every row has a PMID, LastName, Initials and Affiliation. We’ve filtered out any rows that don’t have Affiliation data, thanks to our conditions. For records where BH Smith didn’t have any affiliation data, we never found a -block that met our conditions of LastName Smith, Initials BH, and having Affiliation Data. Because our only -element argument is in our -block, when a record didn’t have a -block that met our conditions, we didn’t output anything at all. The result is that only the rows where BH Smith has affiliation data are output.

# Variables: Nuts and Bolts

So we’ve already seen the syntax a little bit, but let’s go over the nuts and bolts of how you actually use variables.

Variable names can be any combination of digits and capital letters. You declare them in place of an -element argument. Use a dash, then the name of the variable.

Pay attention to where you’re declaring variables. Like we said before, if you’re in a -block Author, xtract can only access information from within that Author. If you declare a variable inside a -block or -pattern, xtract can only access information from within that -block or -pattern.

To output the data in a variable, you use an -element argument, just like you normally do. The variable name should be preceded by an ampersand, and the whole thing put in quotes.

# Variables: Tips and Tricks

There’s a couple of handy shortcuts I’ve found when working with variables.

First, you can put multiple values in the same variable. Just like you can put multiple values in the same column, by separating them with a comma instead of a space. This can help you organize your xtract command a little better, by putting, for example, year and month together in a variable called “-DATE”.

Second, you can output multiple variables in the same -element argument. Just like you can output multiple elements or attributes in the same -element argument by separating them with space. You can do the same thing with variables. You can even mix and match variables, elements and attributes.

Finally, you can output the same variable multiple times, if you want. This is one reason why putting multiple values into one variable is convenient, as it can condense things a bit if you need to output those values multiple times.

**[PAUSE FOR QUESTIONS]**

# What’s New with EDirect?

Until recently, when people asked about doing project on really large sets of PubMed data (like tens or hundreds of thousands of records), we used to mention the NLM Data Distribution program, as an alternative to folks who wanted to work with the entirety of PubMed, but we knew it wasn’t practical for everyone.

Many of our Data Distribution customers are companies or researchers who have the infrastructure to take these bulk downloads, pull out the XML, and file the records into their own homegrown databases so they can work with them. And many of them have teams of developers to make sure all of that works correctly. Most EDirect users don’t have these resources.

However, starting with EDirect version 8.00, there’s a new feature that can help you get the best of both worlds. It takes the flexibility of EDirect, and puts it to work on the entire 28 million records of PubMed.

# Create a local copy of PubMed

This new feature lets you build a local copy of PubMed on your own drive, so you don’t have to wait for E-utilities to fetch the records every time. You can still use esearch to search through the live PubMed database, but rather than fetching the XML with E-utilities, you fetch it from your local copy. This can speed up your retrieval exponentially, especially when you’re trying fetch tens or hundreds of thousands (or even millions) of records.

Now, there are a few caveats with this process. First, in order to actually see any speed benefit, you’ll need to have an external solid state drive, and you’ll want it to be at least 500 GB. Second, building the local copy takes some time. Depending on your system, it could take anywhere from six to over thirty hours the first time you build it.

As you can see, this might be a useful option for some projects where you’re analyzing huge amounts of data, but, for many projects where you’re retrieving less than ten thousand records, you’re probably better off using EDirect the old fashioned way, given the startup costs. If you’re interested in pursuing this, we have a documentation page on the Insider’s Guide website. You can get there via this link, and we’ve also added a link to the homepage.

Also, coming on May 9, the developers of EDirect will be presenting an NCBI Minute webinar that talks about this and some of the other new advanced features of EDirect, so be sure to check that out.

# Affiliation Searching with Variables

I have another variables example to show you, which is a technique you can use to help with affiliation searching. As many of you may know, affiliation data in PubMed is supplied by the publisher of the article, and is free text, not structured data. Different authors report their affiliation differently, making it difficult to search for effectively.

For example, let’s say you’re looking for papers written by researchers at the Center for Translational Medicine at Thomas Jefferson University in Philadelphia, PA.

# Affiliation Searching is Hard, Part 1

Here are just a few of the different ways that papers written by authors from that institution represent their affiliation. This makes it very difficult to search with good precision and recall.

If you just search “translational medicine” or “thomas jefferson” you’ll get many institution with translational medicine centers, or many different components of Thomas Jefferson University.

A different strategy would be to search for each of these different variations in their entirety, plus all of the other variations you find, ORed together, but that seems pretty impractical.

# Affiliation Searching is Hard, Part 2

Another common approach is searching for two portions of the affiliation, ANDed together: “translational medicine[ad] AND thomas jefferson[ad]”. This should return all records that have authors with “translational medicine” and “thomas jefferson” in their affiliation, but it will also get some irrelevant records.

For example, it will get records like this PMID, which has multiple authors:One of the listed authors works at the Research Center for Translational Medicine at Tongji University School of Medicine in Shanghai, and a different author is from the Department of Microbiology at Thomas Jefferson University in Philadelphia. This meets our search criteria, but isn’t what we want.

What we want to do is find only records where both of these search strings are present in the same affiliation field for the same author.

Let’s start with a search.

esearch -db pubmed -query "translational medicine[ad] AND thomas jefferson[ad]"

Like we said before, this search should find all the records from authors with “translational medicine” and “thomas jefferson” in their affiliation, but it will also get us records where “translational medicine” and “thomas jefferson” are in the affiliation for two different authors. That’s okay, we’ll filter it down.

I’m going to run the search by itself first, just to see how many records we’ll get.

Then we use efetch as normal to retrieve the XML of all of these records.

Then we’ll start building our xtract. We want to get a list of PMIDs that have at least one author with both of our affiliation strings in their affiliation. Because we want a list of PMIDs, we’ll use -pattern PubmedArticle. As we already talked about, xtract needs to be able to find the PMID in order to save it to a variable. Before we start looking at authors or affiliations, we need to save the PMID to a variable at the PubmedArticle level. Once we have the PMID in the variable, we’ll be able to display it later on, when we need it.

Next, we’ll use -block to check each Affiliation element, one at a time. I could use -block Author instead, to check each Author one at a time, but this way, if an author has multiple affiliations listed, we’ll make sure that both of our strings are in the same affiliation, not just on the same author.

If an Affiliation element contains both of our search strings, “translational medicine” and “thomas jefferson”, we’ll output the contents of the variable, using “-element “&VAR1”. Thinking about what this means, we can see that, if no Affiliation elements in our record have the desired strings, we will never output the contents of the variable, and we’ll just skip on down to the next record.

esearch -db pubmed -query "translational medicine[ad] AND thomas jefferson[ad]" | \

efetch -format xml | \

xtract -pattern PubmedArticle -VAR1 MedlineCitation/PMID \

-block Affiliation -if Affiliation -contains "translational medicine" -and Affiliation -contains "thomas jefferson" \

-element "&VAR1"

 [EXECUTE]

However, if we have multiple authors whose affiliations meet the criteria (for example, if several colleagues at an institution co-author a paper together), we’ll be printing the PMID multiple times on the same line, one for each Affiliation that matched our criteria. This does makes our PMID list a little less usable.

We can fix this by adding in a special -tab argument. By making the -tab “\n”, we’re telling xtract to separate the columns not with tabs, but with newlines. That way, each PMID we output will be on its own line.

esearch -db pubmed -query "translational medicine[ad] AND thomas jefferson[ad]" | \

efetch -format xml | \

xtract -pattern PubmedArticle -VAR1 MedlineCitation/PMID \

-block Affiliation -if Affiliation -contains "translational medicine" -and Affiliation -contains "thomas jefferson" \

-tab "\n" -element "&VAR1"

[EXECUTE]

Of course, for those records where we have multiple affiliations that meet our criteria, we’ll be printing the same PMID for several lines in a row. But that’s okay, because of our next step.

Once we have this list of PMIDs for records that meet our criteria, we can feed that list back into the epost command. If you don’t remember epost, it’s the command that lets you upload a list of PMIDs to the history server, so you can retrieve them with efetch.

Once I specify my database, epost will upload the list of PMIDs that I’m piping in to the history server, but it doesn’t care about the duplicates; they’ll only appear in the list once. I would then pipe this list of PMIDs into an efetch, and another xtract, to actually get the data that I want from this set of records with authors that have the correct affiliation.

Before I do that, though, I’m just going to run this command up through the epost. This will show us how many PMIDs we uploaded to the history server.

esearch -db pubmed -query "translational medicine[ad] AND thomas jefferson[ad]" | \

efetch -format xml | \

xtract -pattern PubmedArticle -VAR1 MedlineCitation/PMID \

-block Affiliation -if Affiliation -contains "translational medicine" -and Affiliation -contains "thomas jefferson" \

-tab "\n" -element "&VAR1" | \

epost -db pubmed

[EXECUTE]

We’ve weeded out over fifty records that were returned by our first search, but that don’t meet our criteria.

As I said, I can now continue this script by pulling the data that I want, like the most frequent granting agencies or MeSh Headings attached to articles published by authors from this institution.