# Title Slide

Hello and thank you for joining us for EDirect for PubMed, part of the “Insider’s Guide to Accessing NLM Data” series of classes.

My name is Peter Seibert and I’m here with Sara Helson who will be presenting this Thursday’s session, as well as Mike Davidson, who will be helping me answer questions in the chatbox.

Over the course of these five sessions, we will introduce you to using EDirect to search for, retrieve, and manipulate PubMed data in a Unix environment. There’s a lot to cover but we’ll go over the basics, and give you enough to get you started with the resources to continue exploring and learning on your own.

# EDirect for PubMed Agenda

As I said, this course is made up of five sessions.

Today, we’re going to talk about how we use EDirect to get data from PubMed, including how to download full XML PubMed records.

Next time, we’ll start talking about how to extract specific data elements from XML records and arrange them into a custom output format.

Next week at this time, we’ll get into some more specific ways to format and customize our results, and look at some Unix tools that you can use when building your solutions.

In Part Four, we’ll explore the xtract Conditional arguments, which let you filter your output.

And finally, in two weeks, we’ll put all the pieces together and look at some strategies for developing and building solutions to real world problems.

We know most of you already have experience searching in PubMed, so it’s important to understand that many of the tasks and concepts we’ll be talking about in today’s session aren’t necessarily completely new to you. For example, we know you already know how to search PubMed, retrieve and review records. But today we’re going to teach you new techniques and systems to accomplish those tasks. This is going to lay the foundation for the rest of the course, where we’ll build on these new techniques, and show you how to extract the specific data you’re looking for in custom output formats, which is something you can’t do in PubMed.

# Today’s Agenda

We expect most of you have taken our Welcome to E-utilities for PubMed class, or are at least familiar with the content discussed in that class, but we’ll start with a brief recap so that everyone is on the same page

We’re going to briefly talk about Unix, which is the environment we’ll be working in.

I’ll show you how to construct a PubMed search using the esearch command.

We’ll talk about downloading citation data from PubMed records with efetch.

And I’ll finish up by talking about scripts, and how to combine a series of commands into a data pipeline.

# Keep this theme in mind…

Before I recap the content from Welcome to E-utilities for PubMed, I want to remind you of a general theme in that class, that will be overarching these next few weeks of classes: E-utilities in general, and EDirect more specifically are best utilized to get you the data you need, and only the data you need, in exactly the format you need it in.

We’ll come back to that a little later on, but first, let me take you back to the beginning of the Welcome to E-utilities class and provide a few definitions, starting with…

# What is an API?

API stands for Application Programming Interface. It is a set of tools, routines, and protocols for building software applications.

# The E-utilities API

So building on that general definition, the NLM API suite, E-utilities, is a set of tools, routines, and protocols that allow you to interact directly with the data in over two dozen NCBI databases, including PubMed, the MeSH database, and PubMed Central (or PMC).

# The E-utilities API is just a series of rules for querying a database.

Simply put, the E-utilities API is just a series of rules for querying the backend of NLM databases.

# URLs as Database Queries

Those queries are constructed in the form structured URLs. The response you get to the query depends on how you built your URL. You choose one of nine utilities to specify what type of query you’re asking and you select parameters to define the details of the query.

Ideally, you don’t want to create these URLs by hand. With the time spent you might be better off just using the Web version of PubMed

# E-utilities in a Programming Environment

What you really want to do is use E-utilities in a programming environment, to combine multiple queries in sequence, using the results from one query to create the next query.

A programming environment also gives you more options for manipulating the output, rather than just taking XML or text.

# EDirect

That’s why NCBI developed EDirect. It is a set of tools with the E-utilities URL creation rules built in. It also includes a powerful tool for extracting specific data elements from XML and was written to work in a Unix environment

How many of you have used Unix before?

# What is Unix?

Unix is an operating system that allows you to interact directly with your computer. You interact with your computer via a command-line interface, which is basically a text prompt. You type instructions to the computer, tell the computer to execute those instructions, and see the results.

You may also see a command-line interface referred to as a shell or terminal. These terms are not exactly synonymous, but for our purposes today, the distinctions aren’t really important, so I may use those terms interchangeably.

Depending on what kind of computer you’re using, you may have different options for which terminal you want to use. I’m using a Windows 7 PC, so I have downloaded a free terminal emulator called Cygwin. If you’re using a Mac or Linux machine, the terminal you are using may be different.

Any terminal you use will basically amount to the same thing: A prompt where you can type in instructions to the computer, execute them, and see results.

I’m going to give you a quick sneak peak of Cygwin right now, just so you can see what a terminal looks like.

[SHOW CYGWIN PROMPT.]

As you can see, it’s just a prompt where you can enter text, and a blank screen.

[BACK TO SLIDES]

It’s been with us since around the 1970’s so it’s well documented and there are tons of online resources to help learn.

# Some Unix Philosophy

Like most operating systems, Unix is built to work with files: you can create, rename, move, delete, open, edit. Many of the tools and functions of Unix won’t be needed today, but it can you help you understand why things did or did not work. We will be using a few of these “file manipulation” tools in some of the later sessions, and we’ll point them out when we get to them.

Unix is also built on the idea of modular design. Each program is supposed to do one thing but do it very well. Developers and programmers have their own personal tastes and have developed different tools to do almost the same thing, but slightly differently, based on their own idiosyncrasies. This means that there are often multiple different ways to do the same thing.

Because each program generally does only one thing, you will often want to combine a series of programs together in sequence, using the output of one as the input for the next. Unix is really built around this idea, and facilitates the use of “scripts”, which are basically mini-programs that can combine multiple programs together.

# Some Unix terms (ANIMATED)

I’m going to define a few terms we’ll be using throughout the class as work we work in the Unix environment.

Commands are instructions given by a user telling a computer to do something, like, for example, run a program.

Arguments can provide data to be used as the command’s input, or can modify the behavior of the command.

In our first example here, **[CLICK]** “esearch” is our command. It’s one of the EDirect commands we’ll be using a lot.

**[CLICK]** This “-db pubmed” is the argument. It’s providing input to esearch, telling the command that we’ll be using “pubmed” as our database.

In our second example, **[CLICK]** “einfo” is our command. **[CLICK]** With this “-dbs” argument, rather than supplying data, we’re modifying the behavior of “einfo”, telling it that we want to use the “-dbs” option when we execute the command.

As you can see here, different commands have different arguments. Some commands have required arguments, some have optional arguments, and some commands don’t accept any arguments

# Combining commands together

Like we said, we often want to take the output from one command and use it as the input for another command. Rather than running a command that saves its output to a file, then running a second command using that file as an argument, we can combine or pipe these together using a Unix function.

The “|” character (pronounced pipe) is just over your enter key (Shift-backslash). It “Pipes” or channels the output of one command into the next. We’ll look at this in action toward the end of today’s class.

# Why didn’t it work?

When working in any programming language, one thing you should be prepared for is failure and the need for repetition and troubleshooting. Something you’re trying to do is not going to work, and you might have trouble figuring out why it failed.

With Unix, the details matter: it is case-sensitive, space-sensitive, and just-plain-sensitive. If you don’t do everything in the correct syntax, or you drop or add a space, your carefully thought out script won’t work, and a lot of times it won’t tell you what you did wrong. Even if it does work, it might not tell you!

The best advice is to have patience and be willing to experiment.

# Some Unix/EDirect tips

Test early, often and with all incremental changes. When you’re running a ten-line script, and it fails, it can be hard to figure out exactly which line is the root cause.

Try each command separately to identify the failure points. If you can, use small sets of dummy data -often just a small set of PMIDs. You can add in a search strategy that is artificially limited by date so shrink your retrieval set. This will allow you test faster and more frequently.

Know when you need help. The Internet is a great place to start – there are great established Unix user communities out there with not only how-to documentation, but with users who actively answers others questions. Search for the word “Unix” and what you’re trying to do. One site we get a lot of responses from is StackOverflow, which is a sort of crowdsourced Q&A site covering all sorts of programming and technical topics.

# Tips for Cygwin users

For those of you used to using keyboard shortcuts, some don’t work the way you expect. If you’re using Cygwin, you’ll find that Ctrl + C doesn’t copy. By default, Cygwin uses “Ctrl + Insert” as a keyboard shortcut for copy.

Likewise, Ctrl + V does not usually paste. Again, Shift + Insert is default for Cygwin.

These keyboard shortcuts are adjustable in the Cygwin options, so you can tweak them to make them more useful for you.

# Tips for all users

Regardless of what system you’re using, you’ll want to know that Ctrl + C is the keyboard shortcut for “Cancel”. This gives you a quick way out of a mistake, and can be a good way to bail out if your program just seems to be running and running and running forever.

Another handy trick is using the up and down arrow keys to cycle through your history of recent commands. We’ll see this in action a little later on.

And finally, the “clear” command clears your screen and gets you back to a blank command prompt. All you need to do is type the word “clear” and hit enter. We’ll be seeing this a lot later on!

We’re about to get into the first of our EDirect commands, but before we do, does anyone have any questions?

**[PAUSE FOR QUESTIONS]**

# esearch (ANIMATED)

Does anyone remember from the previous class, or previous experience with E-utilities, what “esearch” does?

**[PAUSE FOR RESPONSES]**

That’s right **[CLICK].** The esearch command searches a database and returns the unique identifiers of every record that meets your search criteria. For PubMed, that would be the PMIDs of every PubMed record that matches our query.

# Basic esearch

We’ll start with a basic command that allows you to search a database.

(DEMO IN CYGWIN: esearch –db pubmed –query “seasonal affective disorder”)

The command is “esearch”, and we have two arguments:

‘-db pubmed’ defines which database we’re going to be searching.

‘-query “seasonal affective disorder”’ defines our search string.

(EXECUTE COMMAND)

Now let’s look at the results:

We retrieve an XML snippet that includes a couple of items we’ll discuss later. One item I want to address is the “Count” element. This is the number of records returned.

(SWITCH TO PUBMED)

And if we hop into [www.pubmed.gov](http://www.pubmed.gov) and run the same search, we see that we get the same number of results. This is because esearch is running the same search against the same data that the PubMed interface is.

Let’s look at the Search Details in the right hand column. This will show us how PubMed interpreted our search terms, and then executed that search. This is really helpful in refining your search.

(SWITCH TO CYGWIN)

Going back to our terminal, we can display the Search Details when using EDirect by simply amending the “-log” argument to your existing esearch script.

(DEMO IN CYGWIN: esearch –db pubmed –query “seasonal affective disorder” -log)

One other item you’ll see in your initial retrieval is WebEnv/QueryKey – we’ll get into these later. Right now, I just want you to note that this item will be what allows you to string multiple commands together, and use the results of one to create other commands and processes.

You will note there aren’t any PMIDs; there’s a reason for this. If your results set is huge, you could have hundreds of thousands of PMIDs scrolling down the page. This shows you the size of your results set, and lets you decide what you want to do with it. Which directly ties back into what you’ll see is a recurring theme: Build slowly, one step at a time.

And remember, we can do all the same searches we do in the web version of PubMed in esearch.

(SWITCH TO PUBMED)

So, in PubMed, we can run a search using Boolean operators if we want, or we can tag terms to restrict our search to certain fields.

(Search: malaria AND jama[journal])

We can use those same techniques in EDirect using esearch.

(DEMO IN CYGWIN: esearch -db pubmed -query "malaria AND jama[journal]")

Also just like in PubMed, one way we can manage the size of our results set is by using a date restriction. Let’s say I want to modify my previous search and get only articles published between 2015 and 2017.

I could do this by just writing the date restrictions into my –query argument using the publication date tag as I would in PubMed.

However, esearch offers arguments that let you limit by date range. This can be useful, especially if you have a carefully constructed search string, but want to put some artificial date restrictions on a search for testing.

As I mentioned before, rather than retyping my previous command, I’m going to use the up arrow-key to pull the last command I executed back into my prompt. Now I can edit it and re-run it. I can even use the arrow-keys to page up and down through my history, so I can re-use and edit earlier commands.

(DEMO IN CYGWIN: esearch -db pubmed -query "malaria AND jama[journal]" -datetype PDAT -mindate 2015 -maxdate 2017)

So I’ll re-enter my original esearch: -db pubmed -query "malaria AND jama[journal]" and add my -datetype argument.

We use the –datetype argument to say which type of date we’re restricting by in this case, the publication date.

We then use the –mindate and –maxdate to define our range.

(EXECUTE COMMAND)

Now, this works perfectly well, but if you’re like me, this long string of text is getting hard to read. It’s easier to read and make sense of if we can break it up into different lines at logical break points, to kind of segment the different parts of the command. We can do this using the backslash.

(DEMO IN CYGWIN: esearch -db pubmed -query "malaria AND jama[journal]" \)

Backslash tells Unix that the command isn’t done yet. When I hit enter, Unix is expecting the rest of the command to come. I just type in the rest of the command and hit Enter.

(DEMO IN CYGWIN: -datetype PDAT -mindate 2014 -maxdate 2016)

(EXECUTE COMMAND)

This can be especially useful when you are copying and pasting blocks of code. You’ll see it in a lot of my examples. When you see a line of code end with \, and then see the command continue on the next line, you can always put the whole command on the same line if you want. Just make sure to remove the additional back slashes!

# Search like you do in PubMed

[SKIP SLIDE: Content is covered in demo, slide included for reference only.]

# Restricting by Date

[SKIP SLIDE: Content is covered in demo, slide included for reference only.]

# Be Careful with Quotes (ANIMATED)

We’re about to do our first exercise, but before we do, there’s one other more item we need to discuss – the use of quotes within a search string. If you’ve taken any of our PubMed classes, you know that we generally suggest you avoid using quotation marks in your search since quotation marks turn off Automatic Term Mapping and generally function differently than what you are used to in a standard web search engine.

However, there are times when you need to do a literal search of a particular term, and want to enclose it in quotes. Let’s say we wanted to search for articles about cancer in the journal Science.

(DEMO IN PUBMED: cancer AND science[journal])

You can see in the search details that while cancer gets translated correctly, science[journal] is actually pulling in results for a number of journals with “science” as an alternative title. To avoid this, we can put the word science in quotes:

(DEMO IN PUBMED: cancer AND “science”[journal])

This can cause us some trouble in EDirect.

(SWITCH TO SLIDES)

We’ll want to construct that same search using EDirect. However, our –query is already enclosed in quotes. **[CLICK]**

This is going to cause problems, because Unix is going to get confused about where the search string ends. **[CLICK]** Unix is going to interpret the opening quotes around “Science” as our close quotes for our –query.

**[CLICK]** If you do have to use quotes within your search string, put a backslash before each quotation mark. **[CLICK]** That tells Unix to treat the quotation marks around science as just a normal character that’s part of the string, and not the quotation marks that end our “-query” string.

Before we get into our first exercises, does anyone have any questions?

**[PAUSE FOR QUESTIONS]**

# Exercise 1: esearch

Exercise 1: How many Spanish-language articles about diabetes are in PubMed?

If you need a hint, think about using the “[lang]” tag.

# Exercise 1 Solution (ANIMATED)

(DEMO IN CYGWIN)

esearch –db pubmed –query “diabetes AND Spanish[lang]”

(SWITCH TO SLIDES)

# Exercise 2: more esearch

Exercise 2: How many articles were written by BH Smith, and published between 2012 and 2017, inclusive? There are a few possible ways to do this.

# Exercise 2 Solutions (ANIMATED)

(DEMO IN CYGWIN: esearch -db pubmed -query "smith bh[Author]" -datetype PDAT -mindate 2012 -maxdate 2017)

(SWITCH TO SLIDES)

You could also include the date restrictions in your search query.

* **[CLICK]**
* **[CLICK]**

If you’re still working on the exercises, I’m going to ask you to set them aside for now, since we’re about to move on. The answers to these and all of the exercises are at the bottom of the course materials text file, and we encourage you to go back and look at them after the class, but we’re about to change gears and I don’t want to lose anyone.

# efetch

I feel like my esearch definition question was a softball so anyone want to take a swing at what “efetch” does?

**[PAUSE FOR RESPONSES]**

Given a PMID or a list of PMIDs, it retrieves full records in a bunch of different formats.

# efetch Example

Let me show you an example of efetch.

(DEMO IN CYGWIN: efetch –db pubmed –id 25359968 –format abstract)

Our command is efetch. Our –db argument is pubmed. Our –id argument specifies which record or records we want to retrieve. And our –format argument specifies how we want our results to appear. In this case, we’re asking for the text abstract format.

(EXECUTE COMMAND)

# efetch Formats

(STAY IN CYGWIN)

That got us our one PubMed record in abstract format, but we can change our format argument to view the same record in different formats.

For example, we can change the –format to “medline” to get the field-by-field MEDLINE view:

(DEMO IN CYGWIN: efetch –db pubmed –id 25359968 –format medline)

We can change the –format to “xml” to get the full PubMed XML:

(DEMO IN CYGWIN: efetch –db pubmed –id 25359968 –format xml)

We’ll be doing that a lot in the second and third session, when we talk about extracting specific elements from XML records.

We can also use format “uid” to fetch a list of PMIDs:

(DEMO IN CYGWIN: efetch -db pubmed -id 25359968 -format uid)

This last format may not seem that useful, especially since we already know what PMIDs we’re fetching (since we just supplied them in the –id argument). However, it will be really useful later on when we talk about combining multiple commands together.

# efetch Multiple Records

We can also fetch multiple PubMed records with the same efetch, by providing multiple PMIDS in the –id argument, separated by commas:

(DEMO IN CYGWIN: efetch –db pubmed –id 24102982,21171099,17150207 –format abstract)

You do want to be careful when fetching large numbers of records in more verbose formats like MEDLINE or XML. If you don’t control your output carefully, things can get out of hand quick.

Let me show you another quick efetch example to demonstrate what I mean:

(DEMO IN CYGWIN: efetch –db pubmed –id 26024162 –format abstract)

I’m going to copy and paste this command in. Remember what we said earlier about keyboard shortcuts not working the way you expect. Shift + Insert is the default paste in Cygwin.

(EXECUTE, SCROLL UP TO SHOW RESULTS)

This PubMed record has over 5000 authors on it. Now, this is only one record, and it’s in abstract view. Imagine we were looking at this in XML format (which takes up even more space), and we were looking at more than a single record. Our results would be simply unmanageable.

We’ll talk a little bit later on about some ways to help control, limit, or redirect your output. But the moral of the story is: be careful with your testing, and remember to use Ctrl + C to cancel out of a runaway command.

# Exercise 3: efetch

(SWITCH TO CYGWIN)

There are multiple ways to do this. Obviously, the fastest would be to use PubMed, but that wouldn’t help us learn EDirect! I’m going to use this efetch command:

(DEMO IN CYGWIN: efetch –db pubmed –id 26287646 –format abstract)

I’m using the –format abstract, because I find it easiest to read, and because I know it has the author information, but we could also use MEDLINE or XML instead.

(EXECUTE COMMAND)

And our first author is PF Brennan. That’s Patti Brennan, the current director of the National Library of Medicine.

Any questions about that exercise, or any of the material we’ve covered so far?

**[PAUSE FOR QUESTIONS]**

Okay, if you’re still working on the exercise, I’m going to ask you to put it on hold for now, as we’re about to move on to a different topic. Again, the answer to the exercise is at the bottom of your handout, and we will be posting the slides and recording.

# Creating a data pipeline

(SWITCH TO SLIDES)

I said at the beginning of this class that what makes Unix and EDirect powerful and efficient tools is the ability to combine commands together, using the results of one as the input for another. What we’re talking about is creating a pipeline of data through a series of commands. And we do that by using “|”

Right now, we’ve only talked about two commands that fit together like this, esearch and efetch. If we pipe the search results from an esearch into an efetch, we can get all of the results from our search as a list of PMIDs, in text abstract format, in XML, or in whatever other format we want.

(DEMO IN CYGWIN)

Using “|” is really easy: Just type the first command, type space, type “|”, type space, then type the second command. For example:

DEMO IN CYGWIN: esearch –db pubmed –query “asthenopia[mh] AND nursing[sh]” | efetch –format uid)

This takes the PMIDs identified by the esearch, and uses them as the –id argument for efetch.

It also sends the –db pubmed, to make it clear that these are PMIDs, not UIDs from another database.

(EXECUTE COMMAND)

This is how we get the results of an esearch as a list of PMIDs, but we could always change the -format argument to get our results in a different format.

And we can combine the | with the “\” to put our piped commands on multiple lines.

# Tips for Scripting

We’re about to do an exercise where we create our first multi-command script, but I want to give you a few tips first for things to think about when building scripts.

Things can get unwieldy pretty quickly. You’ll want to keep your initial query manageable and check the search to make sure the results set is not to big

Combine your commands slowly, using a smaller set of test data to make sure your testing is manageable. And again, work in small steps, building up your script slowly and testing frequently.

Any questions about what we’ve talked about so far?

**[PAUSE FOR QUESTIONS]**

# Exercise 4: Combining Commands (ANIMATED)

**[CLICK]**Exercise 4: How do we get a list of PMIDs for all of the articles written by BH Smith between 2012 and 2017?

**[CLICK]** Hint: use Up Arrow to scroll through previous commands

**[CLICK]** Another Hint: Remember “-format uid”

# Exercise 4 Solution

(DEMO IN CYGWIN: esearch -db pubmed -query "smith bh[author]" -datetype PDAT -mindate 2012 -maxdate 2017 | efetch -format uid)

This gives us a big long list of PMIDs that is a little tricky to copy/paste.

Not next class, but the class after, we’re going to talk about ways to save the results of a series of commands to a file. So make sure you stick around!

# Coming next class

That will just about do it for today.

Make sure you come back next class; we are going to talk about how to get specific data elements out of XML and tabulate them using the xtract command.

# In the meantime…

If you’re curious about some basics of working with files in Unix, take a look at this NCBI Now video. It’s also linked from the class page on our website, which is the next thing I wanted to tell you about!

If you want more information about EDirect check out the new EDirect documentation section of our website.This includes the course materials for today’s class, along with the slides, script, sample code, and more. We will also post the video up there, too, which we’ll try to get up as quickly as possible, and hopefully before next week, so you can review it if you like.

If you are playing around with EDirect between now and next week (perhaps working on your homework) and have questions, submit them via this Contact Us button. We’ll try and get back to you ASAP, but we’ll also take some time at the beginning of next class to answer any questions that we think might be instructive to the whole class.

And while you’re on the Contact Us page, consider signing up for the utilities-announce mailing list. Any E-utilities related announcements or alerts are distributed through that list, plus future Insider’s Guide classes will be posted there as well.

# Homework

The homework questions are at the bottom of the handout for today’s class. It’s on the honor system that you complete it before next class but we intend these simple homework questions to help reinforce some of today’s teaching and help get you ready for the next class.

# Questions?

And that will wrap it up for today, but if you have questions right now, I’d be happy to answer them!