Introduction to BLAST

Install BLAST

- If you are on Ubuntu/Linux/Unix (NOT MacOS), at the command prompt (from anywhere) type the following:

    sudo apt-get install ncbi-blast+

  It will then ask you for your password and then go ahead and directly install BLAST. It will also setup blast such that it can be executed from anywhere in the system.

- If you are on MacOS, bring up a terminal and if HomeBrew (http://brew.sh) is not installed, install homebrew. To do this, at the command prompt copy and paste the following:

```
/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

  This will install HomeBrew (think of this like sudo for Mac). Next, type the following to install blast:

    brew install blast

  This will install blast.

- For the purposes of this exercise, please download the files "My_sequences.ffn" and "My_sequences.faa".

BLAST basics

In general, you will always be blasting a fasta file against a database. Both of these are files that will be found locally on your computer. With the new BLAST+ implementation, BLAST is now configured to run from individual programs rather than blastall, as was previously implemented. As such, there are significant differences in the program usage as described below.

Database creation with makeblastdb

Any file in fasta format can be converted into a database. Typically, most people download a complete genome and use this to create a database. This can be either in the format of nucleotides or proteins. For example, I could download all of the genes in the *E. coli* K12 genome in fasta nucleotide format from NCBI (see: A note on accessing genome data from NCBI at the end of this file). In this instance, the file will have an extension, .ffn. In order to create a database for this file, I would open up a terminal window, and browse to where the file is located (if you have not setup BLAST to use anywhere, you will have to make sure that this file is in the same directory that the blast program is located). If this file is known as Ec_K12.ffn, I would type the following at the command prompt:

    makeblastdb –in Ec_K12.ffn –dbtype nucl

Upon hitting enter, this will format the Ec_K12.ffn file into a usable database file for standalone blast. You will notice in the directory where the Ec_K12.ffn file is located that 4 new files will be created with the same name but different extensions. OK, so let's take a look at the anatomy of the command we just ran. The first word, "makeblastdb" is the command. The second piece "-in" is known as a flag. Flags are part of unix-style

programs that inform the program of various options. In this case, the "in" stands for input, meaning we are specifying the input file that makeblastdb should use. This is followed by the input file we wish to format, "Ec_K12.ffn". The last flag, "-dbtype" indicates to the program if Ec_K12.ffn is in protein or nucleotide format. The "-dbtype" database type, and the following "nucl" stands for nucleotide. If we had the same file in protein format (i.e. Ec_K12.faa), the command would look like this:

```
makeblastdb –in Ec_K12.faa –dbtype prot
```

Note that before you can begin blasting, you must ALWAYS have a formatted database to blast against.

Running BLAST

So, now that we have the database, we can begin to use the BLAST programs. The following table describes the appropriate program to use based on what type of comparison you wish to make:

Program	Query Seq. Type	Database Seq. Type	Alignment Level
blastn	nucleotide	nucleotide	nucleotide
blastp	protein	protein	protein
blastx	nucleotide	protein	protein
tblastn	protein	nucleotide	protein
tblastx	nucleotide	nucleotide	protein

In general, the two most widely used programs are "blastn" or "blastp" for nucleotide-nucleotide and protein-protein comparison, respectively. To run a blast, let's say we have a file containing a nucleotide sequence called "My_sequences.ffn". We wish to compare it against the genome of *E. coli* that we just formatted earlier. In this case, it is a nucleotide-nucleotide comparison, and so we will be using the "blastn" program. To run this we need to notice a few things. First, our sequence file "My_sequences.ffn" and our database files (all of them!) need to be in the SAME directory. If this is not the case, the blastn program will spit out an error telling you that it can not find either the input file, or the database file. To run the comparison, type the following:

```
blastn –db Ec_K12.ffn –query My_sequences.ffn –out My_sequences.blastn –evalue 1e-05
```

This will blast our file My_sequences.ffn against the formatted Ec_K12.ffn database. So, let's deconstruct the program. So blastn is the program that we're running. The "-db" flag stands for database, and indicates the database we are using, in this case, our newly formatted "Ec_K12.ffn" database. Next, the "-query", which stands for query, indicates our input file (the sequence file we want to compare against the database), "My_sequences.ffn". The "-out" flag, which stands for output, specifies the name of the file the output of the program should save the results in: My_sequences.blast. Finally, for the purpose of best practices, we will set an e-value cutoff of 1e-05, which will instruct blastn to only return hits with an e-value lower than 1e-05. If run correctly, a new file called "My_sequences.blast" will be created in the directory you are in, with the results from the comparison.

The flags used in this command are standard regardless of what program (blastn, blastp, blastx, etc.) you are using. The first three flags (db, query, out) used in this example are the minimum number of flags that must be used to run the blastn command. If you do not include these flags, the program will spit out an error message telling you that it cannot run the program. For best practices, I always include the fifth –evalue flag.

For comparison purposes, let's say we have the same sequence in protein format: My_sequences.faa, and we wish to compare this against the protein database we generated earlier: Ec_K12.faa. To do this, we would run the following:

blastp –db Ec_K12.faa –query My_sequences.faa –out My_sequences.blastp –evalue 1e-05

Notice that we have changed the program (to blastp, because we are comparing protein to protein), the database to Ec_K12.faa, the input file (My_sequences.faa), and the output file (My_sequences.blastp).

Using the other blast programs is the same as blastn and blastp, except that the inputs and database would be different depending on what you are doing. For example, you could query My_sequences.ffn against a formatted Ec_K112.faa using blastx as follows:

blastx –db Ec_K12.faa –query My_sequences.ffn –out My_sequences.blastx –evalue 1e-05

Other iterations using the other programs are as follows:

tblastn –db Ec_K12.ffn –query My_sequences.faa –out My_sequences.tblastn –evalue 1e-05

tblastx –db Ec_K12.ffn –query My_sequences.ffn –out My_sequences.tblastx –evalue 1e-05

Additional flags that are useful to know (full list here: https://www.ncbi.nlm.nih.gov/books/NBK279675/)

-evalue Expectation value. This tells the program to include only those hits that are LESS than the provided value. For example:

blastp –db Ec_K12.faa –query My_sequences.faa –out My_sequences.blast –evalue 1e-05

will return all hits that have an e-value LESS than 1e-05 in the output.

-outfmt This flag indicates how the output should be displayed. The "8" option indicates that the output should be in tabular format. In other words, it just shows the hits and no alignments. For example:

blastp –db Ec_K12.faa –query My_sequences.faa –out My_sequences.blast –evalue 1e-05 –outfmt 8

outputs the result in tabular format and returns only those hits that have an e-value less than 1e-05.

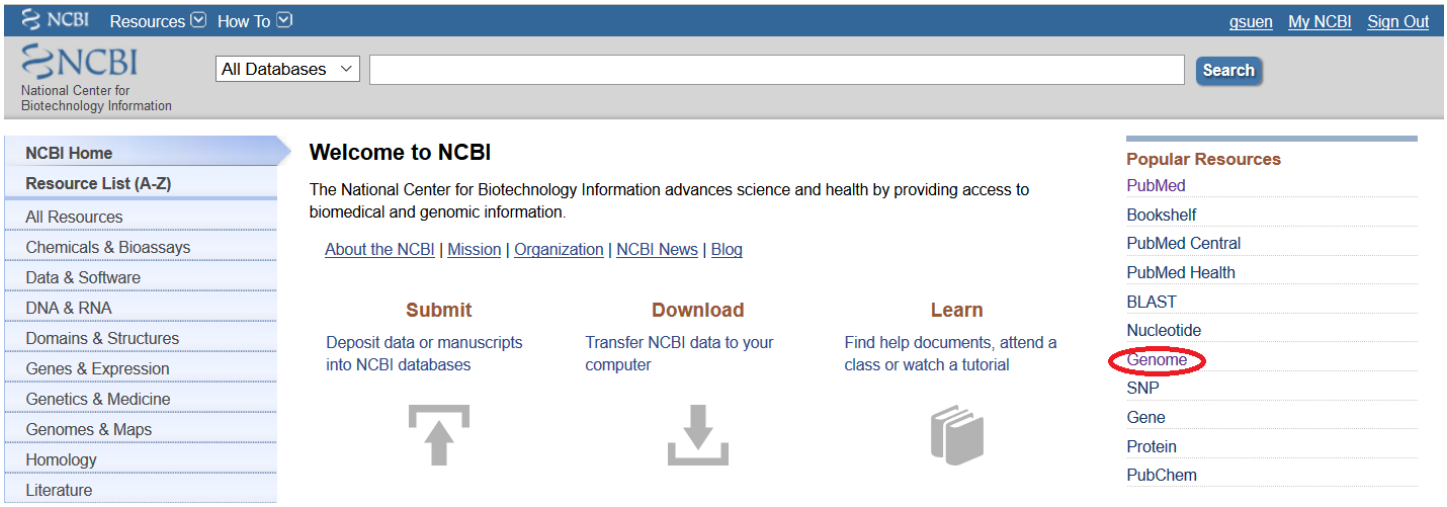
-word_size Modify the word_size to use for your blast. Remember that the default for nucleotide is 11 and the default for protein is 3. Modifying this value can improve your ability to find matches with a shorter sequences.

-num_alignments This flag tells blast how many hits to display in the output. If there are a lot of matches, you may wish to limit the number of hits displayed to a smaller number for both readability and size of the output file.

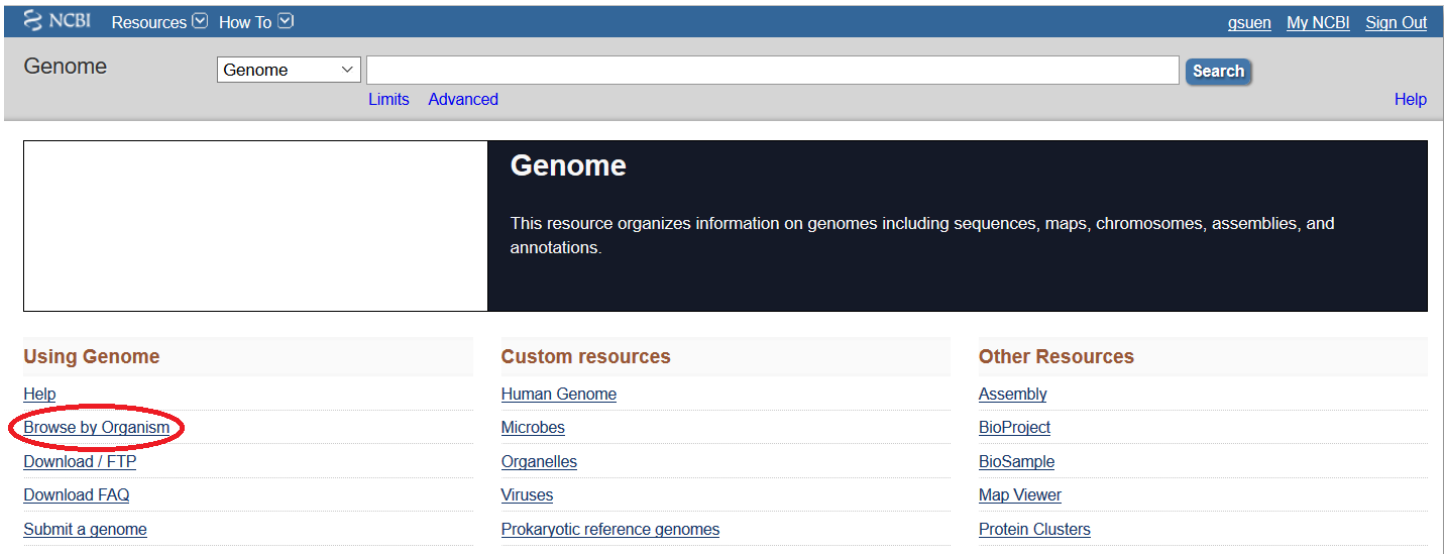
-num_threads This tells blast how many CPUs you wish to use for your blast. This may be useful if you have a particularly large blast job to do and you have multiple CPU cores that you can throw at the problem.

A note on accessing genome data from NCBI

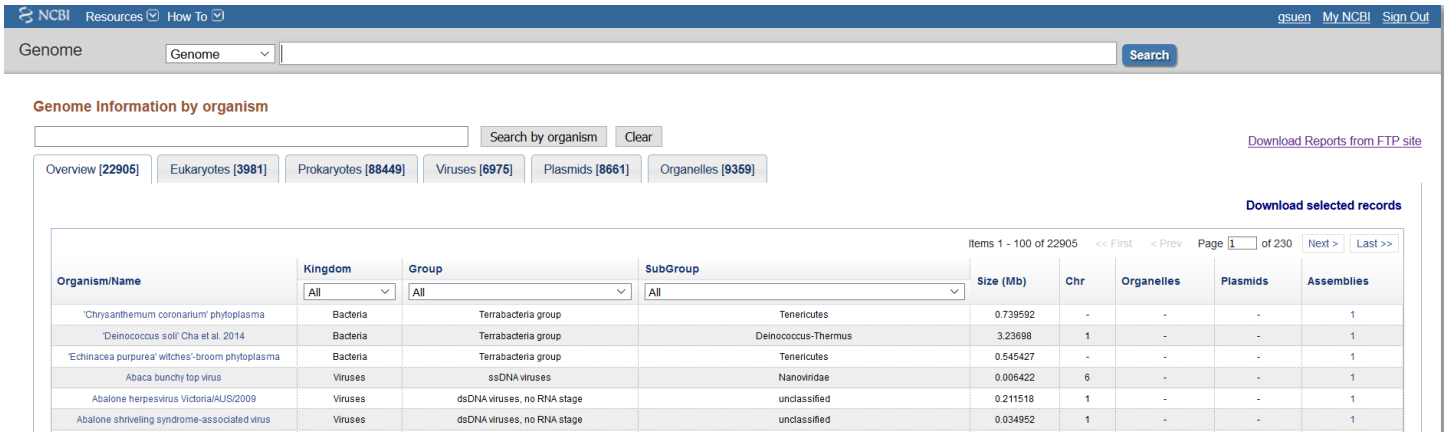
Accessing genome data from the NCBI can be accomplished using a number of approaches. The most common, is to directly search through the database and download the data you need through the NCBI's GUI-based search table. Accessing the search table is fairly easy – you go to the NCBI home page (http://www.ncbi.nlm.nih.gov), click on "Genomes" under the "Popular Resources" Tab:



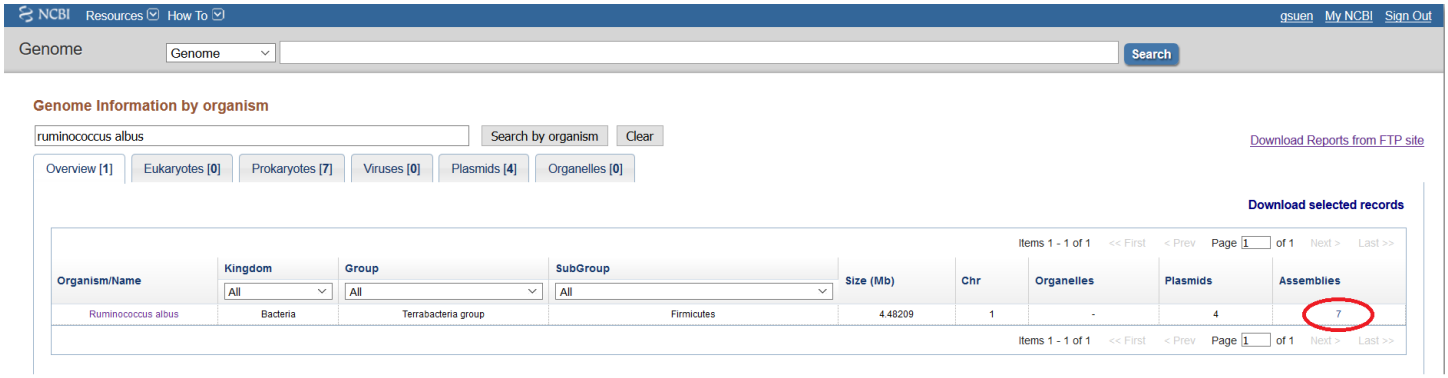
then click on "Browse By Organism":



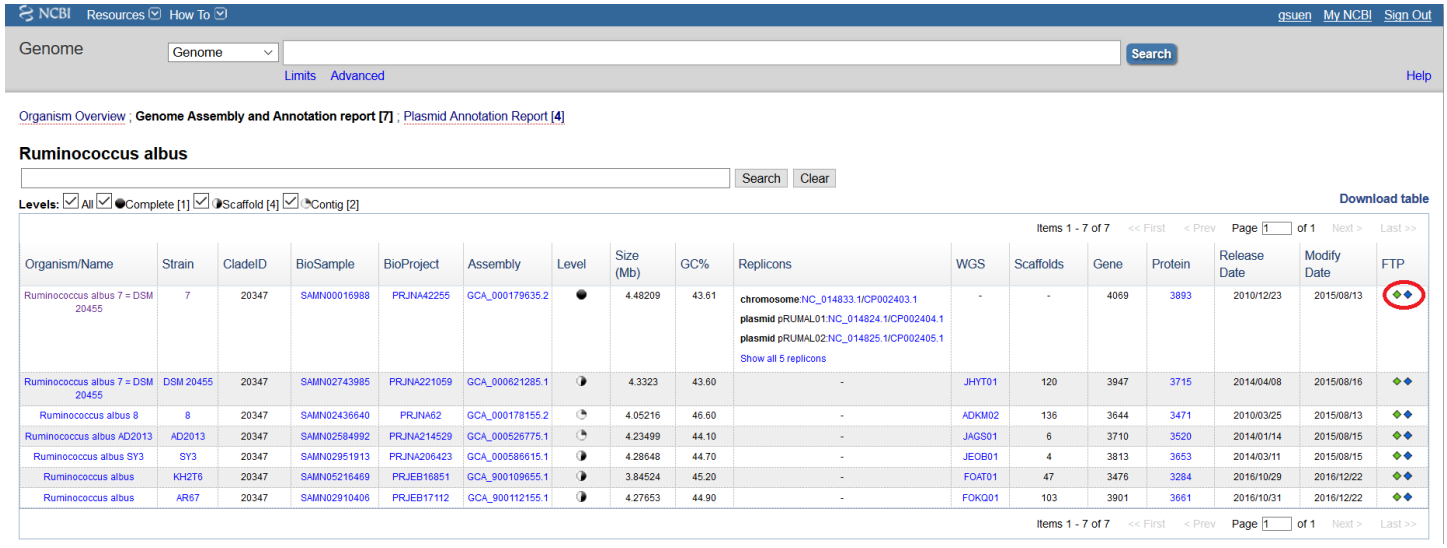
Which then takes you to the Organism Search Table:



From here, you can then search for your favorite organism and access their genomic information. For example, I could do a search for *Ruminococcus albus*, which narrows the list down to this:



The important thing to note here is the number of assemblies. This tells you a bit about how many genomes exist for this organism (think strains), and different types of assemblies (there may be multiple assemblies for the same genome that NCBI keeps track of, esp. if they were done at different sequencing centers). Clicking on the Assemblies link gives a description of these Assemblies:



To download the genome information, you select the genome of interest (Here I have highlighted the strain *Ruminococcus albus* 7, which is the only complete genome (note the filled in circle under "Level" indicating

that it is a complete genome) for the entire group, and there are links out to the FTP site containing the information. Two links exist – one for refseq and the other for GenBank. For the most part, these are identical to each other, and the major difference is that GenBank includes all user-submitted genomes, whereas regseq is a smaller subset that is heavily curated. In other words for each assembly, there will always be a blue diamond FTP link, whereas there may or may not be a green diamond FTP link depending on if the genome is of "good enough quality" to be considered as part of the reference sequence (refseq) database. Clicking on either of these links connects you to the FTP, which looks like this:

Index of ftp://ftp.ncbi.nlm.nih.gov/genomes/refseq/bacteria/Ruminococcus_albus/all_assembly_versions/GCF_000178155.2_ASM17815v2/

Up to higher level directory

Name	Size	Last Modified
GCF_000178155.2_ASM17815v2_assembly_report.txt	15 KB	10/13/2016 3:37:00 PM
GCF_000178155.2_ASM17815v2_assembly_stats.txt	4 KB	10/13/2016 3:37:00 PM
GCF_000178155.2_ASM17815v2_cds_from_genomic.fna.gz	1152 KB	5/22/2016 12:00:00 AM
GCF_000178155.2_ASM17815v2_feature_table.txt.gz	146 KB	5/22/2016 12:00:00 AM
GCF_000178155.2_ASM17815v2_genomic.fna.gz	1172 KB	5/22/2016 12:00:00 AM
GCF_000178155.2_ASM17815v2_genomic.gbff.gz	2790 KB	5/22/2016 12:00:00 AM
GCF_000178155.2_ASM17815v2_genomic.gff.gz	202 KB	5/22/2016 12:00:00 AM
GCF_000178155.2_ASM17815v2_protein.faa.gz	704 KB	5/22/2016 12:00:00 AM
GCF_000178155.2_ASM17815v2_protein.gpff.gz	966 KB	5/22/2016 12:00:00 AM
GCF_000178155.2_ASM17815v2_rna_from_genomic.fna.gz	5 KB	5/22/2016 12:00:00 AM
GCF_000178155.2_ASM17815v2_wgsmaster.gbff.gz	2 KB	5/22/2016 12:00:00 AM
README.txt		9/20/2016 2:03:00 PM
annotation_hashes.txt	1 KB	5/22/2016 12:00:00 AM
assembly_status.txt	1 KB	1/31/2017 5:21:00 AM
md5checksums.txt	1 KB	5/22/2016 12:00:00 AM

To save space, NCBI has zipped up all of the files using gzip (extension .gz). From this list, probably the most useful files for our purposes are the _cds_from_genomic.fna, feature_table.txt, genomic.fna, genomic.gbff and the protein.faa. Each of these corresponds to predicted coding sequences from the genome, a description of each feature in the genome (i.e each gene) in a tab-delimited form that can be opened by Excel, raw genomic data, raw genomic data + predicted coding sequences in GenBank format, and the protein sequences of the predicted coding sequences, respectively. One of the really nice aspects of this update to the FTP system is that each file you download now contains every element from every chromosome/plasmid that make up the organisms DNA. For example, *R. albus* has 4 plasmids and 1 chromosome. If you download the _cds_from_genomic.fna file, it will contain each predicted ORF from all 4 plasmids and the chromosome in one file, which is handy for creating BLAST database (contrast this with the old system, described below). Using this approach, you can quickly and efficiently get to the genome data of your interest.

BUT, what if you'd like to browse all of the data, and download more than just one organism at a time? Most bioinformaticsts will write a script to do this via FTP (one of the major strengths of using an FTP server) so they don't have to point and click Downloading entire groups of genomes from NCBI for the purposes of BLAST *used* to be non-problematic and routine via FTP. However, in September of 2016, NCBI switched their FTP structure such that it is now less intuitive. Their reasoning and rationale is that they require a more systematic way to store their data in the face of the deluge of data that is being submitted daily. Their decided switch seemingly caused a tremendous amount of problems and bioinformaticists are still coming to grips with

how to properly access genomes. Moreover, NCBI decided to do away with Gen Info numbers, which was not met with enthusiasm by the community. The NCBI FTP system continues to evolve, and as such, it is a moving target. However, NCBI recognized that many people and programs still rely on the previous FTP structure, so they have archived the entire "old" FTP, which they stopped updating in December of 2015. Below is an explanation of the old vs. new FTP system.

The OLD FTP system is found here: ftp://ftp.ncbi.nlm.nih.gov/genomes/archive/old_genbank/. In this folder are numerous subfolders corresponding to different kingdoms for which genomes are available. For example, there is a subfolder called "Bacteria" which contains all of the bacterial genomes (ftp://ftp.ncbi.nlm.nih.gov/genomes/archive/old_genbank/bacteria). This folder contains further subfolders, one for each draft and complete genome in NCBI. Within each subfolder is the data associated with the genome itself. A screencapture of the structure is shown below for the example genome *Ruminococcus albus* 7.

Index of ftp://ftp.ncbi.nlm.nih.gov/genomes/archive/old_genbank/Bacteria/Ruminococcus_albus_7_uid42255/

Up to higher level directory

Name	Size	Last Modified
CP002403.asn	9311 KB	1/13/2014 12:00:00 AM
CP002403.faa	1293 KB	1/13/2014 12:00:00 AM
CP002403.ffn	3350 KB	1/13/2014 12:00:00 AM
CP002403.fna	3651 KB	1/13/2014 12:00:00 AM
CP002403.frn	29 KB	1/13/2014 12:00:00 AM
CP002403.gbk	8583 KB	1/13/2014 12:00:00 AM
CP002403.gff	1395 KB	1/13/2014 12:00:00 AM
CP002403.ptt	244 KB	1/13/2014 12:00:00 AM
CP002403.rnt	5 KB	1/13/2014 12:00:00 AM
CP002403.rpt	1 KB	1/13/2014 12:00:00 AM
CP002403.val	4509 KB	1/13/2014 12:00:00 AM
CP002404.asn	1056 KB	1/13/2014 12:00:00 AM
CP002404.faa	139 KB	1/13/2014 12:00:00 AM
CP002404.ffn	360 KB	1/13/2014 12:00:00 AM

Each file in this bacterial genome folder corresponds to different types of files that can be downloaded and used. First, note that each file is named for an Accession number (e.g. CP002403). This Accession number corresponds to the unique accession given to this genome, and is broken down into genomic elements. As such, there are numerous accessions for *R. albus*, one for each genomic element (1 chromosome, 4 plasmids). The types of files provided correspond to each of these elements. Perhaps the most important files are those listed with extensions: ".ffn", ".faa", ".gbk", and ".ptt". The first two correspond to the nucleotide and protein sequences of the predicted ORFs from that element (chromosome or plasmid). The .gbk file is the GenBank file corresponding to that element. The last one, the .ptt file is a flat file that contains an Excel-spreadsheet readable tab-delimited annotation that lists each predicted gene/protein, their location on the element, predicted function, and other information. This last file is very useful for quickly obtaining information about each element.

Note that since this example contains multiple elements, one must collate the .ffn for all elements into one file in order to collect the entirety of the predicted ORFs (same for the .faa). However, the nice thing is that once saved to a folder, a simple concat of all of the files into a single file can be done at command line as follows:

```
cat *.ffn > Rumal7.ffn
```

This will in essence read all .ffn files and concatenate them and redirect them into a new file called "Rumal7.ffn". In this way, the entire genome sequence for a given bacteria can be easily downloaded and made available for local BLAST purposes.

The NEW FTP system for bacteria can be found here: ftp://ftp.ncbi.nlm.nih.gov/genomes/refseq/bacteria/. This folder is HUGE since it contains ALL complete and draft genomes for every bacteria in NCBI (we're talking thousands). As such, even loading the FTP folder takes time. Here, NCBI has grouped like strains into single species folder in an attempt to reduce the number of folders. For example, there are now 7 assemblies of *R. albus* available, and this includes 2 annotated assemblies from their respective sequencing centers (c.f. with the 7 listed assemblies shown via the Search Table interface). Instead of listing the files as in the old system, the new system now categorizes genomes based on latest assemblies. Much like the old system, each bacterium is given its own unique Accession assembly ID. For example, the listing below are subfolders for the latest assemblies for the 9 *R. albus* strains.

Index of ftp://ftp.ncbi.nlm.nih.gov/genomes/refseq/bacteria/Ruminococcus_albus/all_assembly_versions/

Up to higher level directory

Name	Size	Last Modified
GCF_000178155.1_ASM17815v1		1/31/2017 5:19:00 AM
GCF_000178155.2_ASM17815v2		1/31/2017 5:21:00 AM
GCF_000179635.1_ASM17963v1		1/31/2017 5:19:00 AM
GCF_000179635.2_ASM17963v2		1/31/2017 5:20:00 AM
GCF_000526775.1_ASM52677v1		1/31/2017 6:19:00 AM
GCF_000586615.1_RaSY3		1/31/2017 6:29:00 AM
GCF_000621285.1_ASM62128v1		1/31/2017 6:35:00 AM
GCF_900109655.1_IMG-taxon_2651870358_annotated_assembly		1/31/2017 10:20:00 AM
GCF_900112155.1_IMG-taxon_2593339152_annotated_assembly		1/31/2017 10:21:00 AM

So, if I'm interested in *R. albus* 7, which one is it? (If you guessed the second one, good going!). You can pull this information from a file found higher in the directory tree, but it's not easy in practice as these Accessions aren't so conducive to short term memory. Once you drop into one of these folders, you'll notice that it's the same as accessing the information from the Search Table Interface FTP link.

Index of ftp://ftp.ncbi.nlm.nih.gov/genomes/refseq/bacteria/Ruminococcus_albus/all_assembly_versions/GCF_000178155.2_ASM17815v2/

Up to higher level directory

Name	Size	Last Modified
GCF_000178155.2_ASM17815v2_assembly_report.txt	15 KB	10/13/2016 3:37:00 PM
GCF_000178155.2_ASM17815v2_assembly_stats.txt	4 KB	10/13/2016 3:37:00 PM
GCF_000178155.2_ASM17815v2_cds_from_genomic.fna.gz	1152 KB	5/22/2016 12:00:00 AM
GCF_000178155.2_ASM17815v2_feature_table.txt.gz	146 KB	5/22/2016 12:00:00 AM
GCF_000178155.2_ASM17815v2_genomic.fna.gz	1172 KB	5/22/2016 12:00:00 AM
GCF_000178155.2_ASM17815v2_genomic.gbff.gz	2790 KB	5/22/2016 12:00:00 AM
GCF_000178155.2_ASM17815v2_genomic.gff.gz	202 KB	5/22/2016 12:00:00 AM
GCF_000178155.2_ASM17815v2_protein.faa.gz	704 KB	5/22/2016 12:00:00 AM
GCF_000178155.2_ASM17815v2_protein.gpff.gz	966 KB	5/22/2016 12:00:00 AM
GCF_000178155.2_ASM17815v2_rna_from_genomic.fna.gz	5 KB	5/22/2016 12:00:00 AM
GCF_000178155.2_ASM17815v2_wgsmaster.gbff.gz	2 KB	5/22/2016 12:00:00 AM
README.txt		9/20/2016 2:03:00 PM
annotation_hashes.txt	1 KB	5/22/2016 12:00:00 AM
assembly_status.txt	1 KB	1/31/2017 5:21:00 AM
md5checksums.txt	1 KB	5/22/2016 12:00:00 AM

So, why would anyone go through the rigmarole of doing this to get access to genomes? As noted above, if you were interested in all bacterial species belonging to say 15 different genera, doing this by hand through the Search Table Interface is NOT efficient. You could write a script in python/perl to directly download everything you need via FTP without having to go through the web interface. The challenge is linking each file to its proper name, which requires additional file handling. However, the strength of this approach is that each assembly/annotation is given its own unique ID and gets around the issue of naming conflicts.