Instructions on/documentation of getting set up for/results of “Projection of Uncertainties in Extracted from Anticipated SoLID PVDIS Proton Data” project

Contents

[Getting X11 to work on Windows machines 1](#_Toc110869923)

[To access ifarm in a command-line environment (tested with Windows Command Prompt and with PuTTY) 2](#_Toc110869924)

[Accessing the git repo through ifarm 4](#_Toc110869925)

[Prepare for creating a Q2 vs x grid 4](#_Toc110869926)

[To access ROOT 5](#_Toc110869927)

[Obtain the Q2 vs x grid 5](#_Toc110869928)

[Obtaining dV/uV analytical predictions and uncertainties 6](#_Toc110869929)

[Obtaining the PDF Uncertainties using quark\_PDF\_vals.py 8](#_Toc110869930)

[Obtaining Primary Uncertainty Comparison Plot using du\_projection\_plots.py 9](#_Toc110869931)

[Obtaining Other Plots using Uncertainty\_plots.py 9](#_Toc110869932)

[To update your forked GitHub repository from ifarm 10](#_Toc110869933)

[Updating your ifarm directory to match the GitHub repo 11](#_Toc110869934)

[How to download a file from ifarm to your local computer using sftp: 11](#_Toc110869935)

By George Evans, SULI Intern, last updated 8/8/2022

## Getting X11 to work on Windows machines

* (You can probably skip this section if you aren’t using Windows)
* (Steps based on the information found here: [https://cc.jlab.org/windows/X11onWindows](https://cc.jlab.org/windows/X11onWindows%20) (link obtained 6/16/2022). Some of the steps there are out of date, so I’ve updated them and noted when I made a change. IT is planning on doing something to make sure updated information is available, but I don’t know if/when they would do that. Background on what X11 is and why we use Xming and PuTTY can be found at the website)

1. Download and install Xming from <https://sourceforge.net/projects/xming/> (link obtained 6/16/2022).
2. Download and install PuTTY from <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html> (link obtained 6/16/2022).
   1. You probably want one of the package files. I downloaded the 64-bit x86 installer
3. ‘PuTTY should be preconfigured to work in conjunction with Xming. The only configuration item is the flag that tells PuTTY to forward X connections. This is set in the PuTTY config panel under connection -> ssh -> X11. "Enable X11 Forwarding" should be checked, and the X Display Location box should be set to "localhost:0".’ (Changed from “-> Tunnels” to “-> X11”)
4. ‘Start the X Server on your PC -- Xming, that is -- either each time you reboot your PC or whenever you want to use X Windows:
   1. ‘Click the Start button
   2. ‘Select All Programs >> Xming >> Xming.
   3. ‘Xming will appear in your system tray (lower right with clock) as a black X with an orange circle around the middle’
   4. THIS NEEDS TO BE DONE EVERY TIME YOUR COMPUTER RESTART. If I forget, I start getting the error
      * 1. \_tkinter.TclError: couldn't connect to display "localhost:34.0"
      1. which can be frustrating since the error just appears even though nothing was changed.
5. In the PuTTY config panel, select “Session”. In the space for “Host Name (or IP address), type [username@login.jlab.org](mailto:username@login.jlab.org), where username is your email username.
6. Save this session for later
   1. Type a name, such as JLab, into the space for “Saved Sessions”
   2. Click “Save”
   3. In the future, after opening the PuTTY config panel, you will only need to select the saved session and click the button “Load” instead of doing the things in the X11 tab and typing in your host name
   4. Note, after I pinned PuTTY to my taskbar, right clicking its icon lets me open a saved session without needing to do bullet point c
7. Click “Open” and skip to step three of “To access ifarm in a command-line environment”

Note that commands are put in quotation marks, but quotation marks should not be included in the command.

## To access ifarm in a command-line environment (tested with Windows Command Prompt and with PuTTY)

* ifarm is where many useful files are stored. It also has access to ROOT, C++, and python, all of which will be needed.

1. Open up your command-line environment
   1. For Windows, this is Command Prompt, although I would recommend going through the steps for “Getting X11 to work on Windows machines” instead. Doing so will mean your command-line environment is through Xming and PuTTY.
2. Command: “ssh -X [username@login.jlab.org](mailto:username@login.jlab.org)”
   1. Insert your username (as found on your JLab email) in place of “username”
   2. -X isn’t always required. It is used to access X11
3. Type in your JLab password
   1. Result: Text

      Description automatically generated
4. Command: “ssh -X ifarm”
   1. You need to be specially authorized to use ifarm
   2. -X isn’t always required. It is used to access X11
5. Type your JLab password
   1. Result Text

      Description automatically generated
6. First time:
   1. Command: “cd /w/eic-scshelf2104/users”
   2. Command: “mkdir username”
      1. Use your desired username. This will be the name of your personal directory.
   3. Command: “cd username”
7. After first time:
   1. Command: “cd /w/eic-scshelf2104/users/username”
   2. (It’s possible to create a keyword that allows you to skip typing the above)
8. Some useful commands:
   1. “ls” to see files in the directory; “pwd” to see directory path; “cd ..” to go up a directory level
   2. Tab autocompletes things, especially file/folder names
   3. “more” lets you see inside files (“q” to quit and spacebar to see more)
   4. Look up “cp” to copy files from one place to another.

## Accessing the git repo through ifarm

* All of the files required for this project can be found in this git repo, so setting it up will allow you to quickly start work
* It is assumed that you already have a GitHub account and have created your own directory on ifarm

1. cd into your directory on the ifarm
2. On the GitHub page (<https://github.com/mnycz/Summer2022_solid_proton>), click “Fork”
   1. This allows you to create your own personal version of the repository which you can edit without fear of breaking the original
   2. Skip this step if you’ve already created your own fork
3. Once you have created your fork, on your fork’s main page, there is a green box that says “Code.” Click on it
4. Copy the link it gives you for HTTPS
5. In your directory on ifarm, command: “git clone HTTPS\_Link”
   1. where “HTTPS\_Link” is the link you copied

* This will create a new directory in your ifarm directory. You can cd into it and should have the same files there as on the website

## Prepare for creating a Q2 vs x grid

1. Navigate to Summer2022\_solid\_proton (the GitHub directory that you should now have in your ifarm directory after performing the last step)
2. To look inside and edit the file for creating the grid, command: “emacs -nw Q2\_vs\_x\_grid.C”
   * For me, I can copy the above command (leaving out the quotation marks), then paste it in the command-line by right clicking
   * If desired, “emacs Q2\_vs\_x\_grid.C” opens the file in a new window.
3. Near the top of the file, update “const double runtime” to equal the time over which the experiment runs in seconds (currently in the form of (# of days) \* (seconds per day))
   * Default is 90 days
4. Near the top of the file, update “const double pb” to equal the experiment’s beam polarization
   * Default is 0.85
5. Near the top of the file, there is a line that says “const char\* kKeyList[]={“
   * Currently, the program assumes the directory of the simulation file is “/lustre19/expphy/volatile/halla/triton/mnycz/SOLID/” and that the file begins with “nt\_PVDIS\_”. If this is the case, edit the first entry to kKeyList to whatever text comes after “nt\_PVDIS” in the filename.
   * If this is not the case, scroll down to “void analysis\_PVDIS\_FOM\_sim(int, double, double)” and edit the text in sprintf.
6. Change the directory to which processed files are saved by editing the definition of pFile. Specifically, replace “11GeV\_files” with your desired directory inside of the directory Files (making sure to leave “/%s\_%s.txt” intact)
   * You can find pFile using the search command ctrl-s (to find next occurrence of the term)
     + To find the next occurrence of the term, do ctrl-s again
     + To exit the search, ctrl-g
   * To undo unintentional edits, ctrl-x then u (for me, ctrl-/ also works, but that isn’t always the case I hear)
     + To redo, just press Ctrl+g first then undo. Further undo will be redo. Press Ctrl+g again to reverse direction
   * Note, some of the files will be saved to the directory you input here. Others appear to be saved to the same directory as fine\_Q2\_vs\_x\_grid.C, although I’ve commented out the section of the code that generated other files.
7. Somewhere in the middle of the file, update nBinQ2, Q2Max, and nBinx to match your desired purpose
   * For 11 GeV, nBinQ2=Q2Max=14 works best. For 22 GeV, nBinQ2=Q2Max=30 works best.
   * If creating a grid to be analyzed directly, use nBinx=10. If you are creating a grid to be used in obtaining PDF uncertainty, use nBinx=100. For the heat map that has fine Q2 and x, use the normal Q2 values but multiplied by 10 and use 100 for x.
8. To save the file, ctrl-x then ctrl-s
9. To return to command-line environment, ctrl-x then ctrl-c

## To access ROOT

* “ROOT is a software framework for data analysis and I/O: a powerful tool to cope with the demanding tasks typical of state of the art scientific data analysis. Among its prominent features are an advanced graphical user interface, ideal for interactive analysis, an interpreter for the C++ programming language, for rapid and efficient prototyping and a persistence mechanism for C++ objects, used also to write every year petabytes of data recorded by the Large Hadron Collider experiments.” <https://root.cern/primer/>

1. Optional command: “setup cernlib”
   1. This is sometimes needed, but not to run ROOT
2. Command: “root -l”
   1. The “-l” gets rid of the startup logo

## Obtain the Q2 vs x grid

* Q2\_vs\_x\_grid.C takes a root file (simulation file) as the input and then selects a series of Branches (like x, Q2, W, rate, etc...) from the root Tree. It also has a header file which is used to estimate the trigger efficiency. It uses these, along with a physics cut on W (so only DIS events) to determine the rate (it also may calculate uncertainty) for the fine bins. It produces figures (for larger bins). It does this for the case where everything is taken into account (baffles to block background particles and trigger efficiency) and also if they are not (which was meant to be turned off to suppress some of the output, but this may not have happened).
* The most important part of the script is that it generates the Q2 vs x grid. This grid is later used to calculate the Apvand the PDF uncertainties
  + Above description paraphrases Michael Nycz’s description of the file
* Note: PVDIS\_tri\_eff\_Rakitha1.h needs to be in the same directory as Q2\_vs\_x\_grid.C for the program to work (I think)

1. Go through the steps described for preparing to create the fine Q2 vs x grid
2. Navigate to the GitHub directory (if you haven’t already)
3. Command: “source setroot\_CUE”
   1. This uses the file “setroot\_CUE” in the GitHub directory
4. Access ROOT
5. Command: “.x fine\_Q2\_vs\_x\_grid.C”
   1. This would normally create a bunch of files, but I’ve commented it out so that only the desired file is produced. Specifically, it would create some TXT files in a directory chosen within the code of Q2\_vs\_x\_grid.C and some other files in the same directory as Q2\_vs\_x\_grid.C
6. To exit ROOT, command: “.q”
7. Navigate your way to the directory where the newly created file is
   1. E.g., command: “cd Files/22GeV\_files/”
8. Rename the freshly created file via command: “mv filename.txt Q2x\_#x#\_rateGrid\_#GeV\_#days\_#Polarization\_ProtonTarget.txt”
   1. where
      1. “filename” is the name of the freshly created txt file (e.g., rate\_Q2x\_acc\_finebin\_50uA\_solid\_PVDIS\_LH2\_moved\_full\_eAll\_filenum100\_22GeV\_Z10cm\_1e6),
      2. “#x#” is the size of the grid (e.g., 30x10, where 30 is the number of Q2 rows and 10 is the number of x columns),
      3. “#days” is the number of days of runtime (e.g., 90days)
      4. “#Polarization” is the percent polarized (e.g., 85Polarization for 85% Polarized)
      5. “Proton” can be switched for something else if it’s a different target, but that won’t be addressed in this overview.
   2. I’ll use the example Q2x\_30x10\_rateGrid\_22GeV\_90days\_85Polarization\_ProtonTarget.txt for the next steps.
   3. Renaming allows it to be easily used by later steps and ensures that only the correct file is used for analysis (as opposed to one accidently generated while testing)
9. In the directory where the TXT file was generated, command: “awk '{print $1 "," $2 "," $3}' filename.txt > filename.csv”
   1. where “filename” is the filename you created in the previous step
   2. This converts the TXT file to a CSV file (which is easier to process)
   3. For example, command: “awk '{print $1 "," $2 "," $3}' Q2x\_30x10\_rateGrid\_22GeV\_90days\_85Polarization\_ProtonTarget.txt > Q2x\_30x10\_rateGrid\_22GeV\_90days\_85Polarization\_ProtonTarget.csv”

## Obtaining dV/uV analytical predictions and uncertainties

* This takes a Q2 vs x grid of rates as inputs and outputs several CSV files. Specifically, it outputs a CSV file with information on each bin (such as the asymmetry pseudo-data, d/u projections, and uncertainties), a CSV file with information on each x value (the d/u projections and their uncertainties), and a CSV file with d/u and d/u uncertainties for each x after the Q2 bins for each x were fitted for the best d/u fit.
* Note: fits\_20211017.sh is required to be in the same directory as SoLID\_d\_u\_analysis.cpp for SoLID\_d\_u\_analysis.cpp to run, even though it doesn’t do anything for the program anymore. If I have time, I’ll make it not needed.
* Note: Makefile needs to be in the same directory as SoLID\_d\_u\_analysis.cpp for SoLID\_d\_u\_analysis.cpp to be compiled when typing “make” (I think)
  + I don’t know what it does, but near the top it has the name of SoLID\_d\_u\_analysis.cpp and if that name gets changed, “make” doesn’t work.

1. Obtain a Q2 vs x grid for analysis
2. Navigate to the GitHub directory
3. Command: “emacs -nw SoLID\_d\_u\_analysis.cpp”
4. Near the top of the file, update RUNTIME\_DAYS, BEAM\_POLARIZ, BEAM\_E, NQ2BINS, and NXBINS to match the file you are hoping to use as input
5. Near the top of the file, update SIN2\_TH and PDF\_NAME to match your desired analysis
6. Check INFILE\_DIR and INFILE\_NAME to ensure they match your input file, noting that the program assumes the input file is at INFILE\_DIR + INFILE\_NAME
7. Check OUTFILE\_ALLBINS, OUTFILE\_XBINS, and OUTFILE\_FITS to make sure they are named correctly.
   1. They should automatically generate good names based on previous inputs
   2. These CSV files will be created in the same directory as SoLID\_d\_u\_analysis.cpp.
8. If doing the fine-Q2 vs fine-x grid, change MAXBINS from 500 to 5000; otherwise, ensure it is 500
9. Save (ctrl-x then ctrl-s) and quit (ctrl-x then xtrl-c)
10. Command: “source use\_djangoh\_4.6.15.csh”
    1. Only needed once per session
    2. This gets LHAPDF set up (which is needed now)
    3. This uses the file source use\_djangoh\_4.6.15.csh, which is in the GitHub directory
11. Command: “make”
    1. This will attempt to compile an executable. If there are errors, it will inform you of them and not compile the code. If there are only warnings, it will inform you of them but still compile the code. Fix any errors by using emacs to edit SoLID\_d\_u\_analysis.cpp.
12. Command: “./fits\_20211017.sh ./eAll.Linux.x86\_64.exe”
    1. This should run the executable and produce the CSV files in the same directory as SoLID\_d\_u\_analysis.cpp
13. Move the files to the correct file folder. Command: “mv \*.csv Files/foldername”
    1. where “foldername” is the folder to store the files in
    2. “\*.csv” means “all files that end in .csv in this folder (excluding files that are inside of subfolders)
    3. Note: a plot, du\_plot.png, is also produced, but it isn’t used in a deliverable, so I’ll ignore that plot

## Obtaining the PDF Uncertainties using quark\_PDF\_vals.py

* quark\_PDF\_vals.py reads in a Q2 vs x grid and generates a CSV file with the PDF values for each of those bins. Specifically, the CSV file will have d, u, s, c, dbar, ubar, sbar, cbar, and the PDF uncertainties of each of those. It will also have d+, u+, s+, c+, dV, uV, sV, and cV. Finally, it has dV/uV and the PDF uncertainty in dV/uV.

1. Obtain a grid from Q2\_vs\_x\_grid.C which has 100 x columns and an appropriate number of Q2 rows (14 for 11 GeV or 30 for 22 GeV)
2. Submit that grid to SoLID\_d\_u\_analysis.cpp.
   1. The resultant file for each x bin has the best Q2 values, which we will be using.
   2. Move the resultant files to the correct directory
3. Navigate to the GitHub directory
4. Command: “source use\_djangoh\_4.6.15.csh”
   * Only needed once per session
   * This gets LHAPDF set up
   * This uses the file use\_djangoh\_4.6.15.csh, which is in the GitHub directory
5. Command: “emacs -nw quark\_PDF\_vals.py”
6. In the middle of the file, after the last function, update beamE, NQ2bins, Nxbins, runtime, and bp to match your input file
7. Ensure that Bins is reading data from the analytic values for each x CSV that was produced by SoLID\_d\_u\_analysis.cpp
8. Ensure savefile\_dir and savefile\_append save the file as you wish it to be saved, noting that the file’s name will be the PDF’s name with savefile\_append appended to the end.
   * As it is currently, the names are designed to be easily used by the next parts of the process
   * If you are using a PDF other than CT18NLO as your base, this will need to be changed slightly
9. Ensure Grid, N\_PDF, and base match the PDFs you want to analyze
   * Grid defaults to ['NNPDF40\_nlo\_as\_01180', 'PDF4LHC21\_40', 'CT18NLO'], which is the list of PDF sets to use ('NNPDF31\_nlo\_as\_0118' is commented out due to large uncertainty)
   * N\_PDF defaults to [101, 41, 59], which is the number of set members in the respective PDF set. (‘101,’ the number of members in ‘NNPDF31\_nlo\_as\_0118,’ is commented out)
   * Base defaults to ["replica", "Hessian", "Hessian"], which is whether the PDF is replica-based or Hessian-based (used in determining uncertainty) (NNPDF31\_nlo\_as\_0118 is replica based)
10. Update find\_unc, make\_plots, and save\_data according to your wishes
11. Save (ctrl-x then ctrl-s) and quit (ctrl-x then ctrl-c)
12. Ensure you can run this program a long time (~1 hour)
13. Command: “python quark\_PDF\_vals”
14. When the files are generated, navigate to them and rename them slightly to have the “q” next to the “PDF.” For example: “mv CT18NLO\_22GeV\_q\_PDF\_vals\_1x100grid.csv CT18NLO\_22GeV\_qPDF\_vals\_1x100grid.csv”
    * This ensures that accidentally forgetting to change a name won’t overwrite this data with other data without you knowing
    * Do this for each PDF file that was generated.

## Obtaining Primary Uncertainty Comparison Plot using du\_projection\_plots.py

* du\_projection\_plots.py brings together files generated elsewhere into a plot that compares the various uncertainties

1. Obtain the CSV file for analytic values for each x from SoLID\_d\_u\_analysis.cpp when x=10 and an appropriate number of Q2 bins were used (14 for 11 GeV; 30 for 22 GeV)
2. Obtain the CSV file for quark PDF values from quark\_PDF\_vals.py when x=100 and an appropriate number of Q2 bins were used (same as before)
3. Navigate to the GitHub directory
4. Command: “emacs -nw du\_projection\_plots”
5. At the top of the file, ensure that BEAM\_E, NQ2bins, Nxbins\_analytical, and Nxbins\_PDF all have good values
   1. Ensure that PDF\_names is correct. These correspond to the analytic uncertainties Note that PDF\_names should be able to support multiple PDFs (assuming you have the files ready), but it hasn’t been tested
6. Ensure that footer\_vals is correct for your file (footer\_vals is used to avoid plotting the last data point if there is excessive uncertainty on it)
7. Pick whether you want the errors stacked on top of each other (true) or separated (false) using stacked\_errors
8. Ensure that dir\_path\_a, file\_suffix\_a, dir\_path\_f, and file\_suffix\_f are all pointing to real files, noting that the following files are loaded:
   1. dir\_path\_a + PDF\_name[i] + file\_suffix\_a
   2. dir\_path\_f + PDF\_name[i] + file\_suffix\_f
9. Ensure that PDF\_names2 is correct. These correspond to the PDF uncertainties
10. Ensure that dir\_path\_PDF and file\_suffix\_PDF helps things point to the right place, noting that the following will be loaded
    1. dir\_path\_PDF + PDF\_names2[i] + file\_suffix\_PDF
11. Ensure that dir\_path\_pic and pic\_suffix point to where you want to save the resulting plot and have the correct file type (.pdf and .png both work, for example)
12. Save (ctrl-x then ctrl-s) and quit (ctrl-x then ctrl-c)
13. Command: “source use\_djangoh\_4.6.15.csh”
    1. Only needed once per session
    2. This gets LHAPDF set up (which is needed now)
    3. This uses the file source use\_djangoh\_4.6.15.csh, which is in the GitHub directory
14. Command: “python du\_projection\_plots”

## Obtaining Other Plots using Uncertainty\_plots.py

* Uncertainty\_plots.py brings together files generated elsewhere into a several that compares the various uncertainties or plot the asymmetry values on a Q2 vs x grid

1. Obtain the CSV files for analytic values for each x and for all bins from SoLID\_d\_u\_analysis.cpp when x=10 and an appropriate number of Q2 bins were used (14 for 11 GeV; 30 for 22 GeV)
2. Obtain the CSV file for quark PDF values from quark\_PDF\_vals.py when x=100 and an appropriate number of Q2 bins were used (same as before)
3. Navigate to the GitHub directory
4. Command: “emacs -nw Uncertainty\_plots”
5. At the top of the file, ensure that BEAM\_E, NQ2bins, Nxbins\_analytical, and Nxbins\_PDF all have good values
   1. Ensure that PDF\_names is correct. These correspond to the analytic uncertainties Note that PDF\_names should be able to support multiple PDFs (assuming you have the files ready), but it hasn’t been tested
6. Ensure that footer\_vals is correct for your file (footer\_vals is used to avoid plotting the last data point if there is excessive uncertainty on it)
7. Ensure that dir\_path\_al, file\_suffix\_al, dir\_path\_ea, and file\_suffix\_ea are all pointing to real files, noting that the following files are loaded:
   1. dir\_path\_a + PDF\_name[i] + file\_suffix\_a
   2. dir\_path\_f + PDF\_name[i] + file\_suffix\_f
8. Ensure that dir\_path\_PDF and file\_suffix\_PDF helps things point to the right place, noting that the following will be loaded
   1. dir\_path\_PDF + PDF\_names[i] + file\_suffix\_PDF
9. Ensure that dir\_path\_ff and file\_suffix\_ff helps things point to the right place, noting that the following will be loaded
   1. dir\_path\_ff + PDF\_names[i] + file\_suffix\_ff
10. Ensure that dir\_path\_pic points to where you want to save the resulting plot and that pic\_type is the correct file type (.pdf and .png both work, for example)
11. Save (ctrl-x then ctrl-s) and quit (ctrl-x then ctrl-c)
12. Command: “source use\_djangoh\_4.6.15.csh”
    1. Only needed once per session
    2. This gets LHAPDF set up (which is needed now)
    3. This uses the file source use\_djangoh\_4.6.15.csh, which is in the GitHub directory
13. Command: “python du\_projection\_plots”

## To update your forked GitHub repository from ifarm

* Once you’ve made (and saved) a change…

1. In the ifarm directory you created earlier, command: “git add filename”
   1. where “filename” is the file that was changed (or created)
   2. You can repeat this until you’ve added all the files you’ve changed
   3. To remove a file, use “git rm filename”, or you can remove the file on GitHub. You can also rename files on GitHub, which avoids creating a new file and needing to delete the old file.
2. Command: “git commit”
   1. This will make you write a description of the changes you made
      1. Sometimes it doesn’t start typing when you start typing
   2. When finished writing, hit ESC then type “:wq” (without the quotation marks, as usual) to quit
3. Command: “git push”
   1. This will update your local fork with each individual commit timestamped.
   2. This required me to type in my GitHub username then, instead of a password, a “Personal access token.” Follow the instructions here to get a personal access token: <https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token>. Note that you can paste the token into the password space (on Windows, this seems to be done by right-clicking in the terminal)

## Updating your ifarm directory to match the GitHub repo

* Within the directory that matches the repository, command: “git pull origin main”

## How to download a file from ifarm to your local computer using sftp:

1. In the terminal, cd to where you would like to have the file on your own computer
   1. On Windows, this can be done easily using the technique “Using CMD command in File Explorer to open Command Prompt Window” described here: <https://www.itechtics.com/open-command-window-folder/>
2. Command: “sftp [username@ftp.jlab.org](mailto:username@ftp.jlab.org)”
   1. where “username” is your username
3. cd to the directory of the file you want to download
   1. e.g., “cd /w/eic-scshelf2104/users/username”, where “username” is your username
4. command: “get filename”
   1. where “filename” is the name of the file you want to download
   2. Now the file should be on your computer. You can exit sftp by typing quit

* A few notes: the same basic commands work still
  + “cd”, “pwd”, and “ls” still work on the remote directory (ifarm)
  + You can also do basic things on your local device by appending an “l” to these commands:
    - “lcd”, “lpwd”, and “lls” give you information about where you are on your local computer and can help you navigate there