Knowledge_Progression_handling_RNA-seq_datasets

Aims

I want to see if I can replicate results published on the basis of a bulk RNA-seq dataset in terms of:
- differential gene expression (DGE) analysis
- GO term analysis on differentially expressed genes
- Functional enrichment if possible
- enrichment in disease-related genes or SNPs
- gene network analysis

Finding a suitable study

I searched for clinically relevant, loosely cerebral cortex development-related RNA-seq datasets with attached publication. I selected only datasets uploaded last year and whose publications don't do highly in-depth analyses, as I believe that me working on this data will bring the biggest contribution to expanding our knowledge horizon. As to why I selected datasets that were associated with a publication: this was for me to be able to cross-check the results of my analyses and make sure I am performing my analysis properly.

I found this study: GSE167193 on GEO at NCBI. This study analyzed a potential connection between a pregnancy disorder known as pre-eclampsia and autism spectrum disorders. Pre-eclampsia is a common and dangerous increase in maternal blood pressure that can occur after the 20th week of pregnancy. One in 20 pregnant people are affected by pre-eclampsia, which has a severe negative impact on both the pregnant person and the fetus. There is no treatment known to date, except for the delivery of the placenta, and very little is known about the long-term consequences on the development of the child's brain.

The authors used a pre-eclampsia mouse model in which the disorder was induced in pregnant mice by treating them with a compound called L-NAME. The mouse offspring from mothers that were exposed to L-NAME behaved in ways reminiscent of ASD and scored accordingly on several behavioral tests that are thought to assess metrics related to the condition.

To better understand the relationship between the two conditions, the authors sequenced RNAs from the cortices of mouse embryos at day E 17.5, two days before birth, from mother animals that had or did not have elevated blood pressure, and similarly from the hippocampi of adult offspring. The authors generated cDNA libraries from the cortical RNAs of control and experimental condition embryonic cortices using a kit for stranded mRNA detection. This means that the kit enriches for poly-A-tailed mRNAs and also captures information regarding which genomic DNA strand the mRNAs were transcribed from. This is an excellent fit for my purposes, as I am, for the time being, interested in coding transcripts. Then, the cDNA libraries were fragmented, and the fragments were sequenced on Illumina Hiseq X machines, a process that generated millions of paired-end 150 bp reads (21 million, to be a little more precise [link to post where I open the FastQC files]).

I focused my investigation on the embryonic cortex data, as this is a system I am more familiar with from my own research work.

```
iweber@Lenovo-Ioana:~/Datasets/Pre_eclampsia_mice$ vdb-validate SRR13761520
2024-03-11T18:10:23 vdb-validate.3.1.0 info: Database 'SRR13761520' metadata: md5 ok
2024-03-11T18:10:23 vdb-validate.3.1.0 info: Table 'SEQUENCE' metadata: md5 ok
2024-03-11T18:10:23 vdb-validate.3.1.0 info: Column 'ALTREAD': checksums ok
2024-03-11T18:10:24 vdb-validate.3.1.0 info: Column 'QUALITY': checksums ok
2024-03-11T18:10:25 vdb-validate.3.1.0 info: Column 'READ': checksums ok
2024-03-11T18:10:25 vdb-validate.3.1.0 info: Database 'SRR13761520' contains only unaligned reads
2024-03-11T18:10:25 vdb-validate.3.1.0 info: Database 'SRR13761520' is consistent
iweber@Lenovo-Ioana:~/Datasets/Pre_eclampsia_mice$
```

I downloaded the datasets directly from the entry of this study on GEO in SRA format. As I found out, SRA is an NCBI-specific way to compress FastQ files. These are the ones with the raw reads from the sequencers and I learned that these are precisely what I will need for performing differential sequence analysis. The processing of the files to any other format, such as SAM or BAM, or anything else except for basic quality control, may introduce biases that affect subsequent data analyses.

So what is RNA-seq anyway? And what's the deal with alternative splicing analysis?

To read:

General info on RNA-Seq analysis https://academic.oup.com/bib/article/23/2/bbab563/6514404?login=false

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4712774/ tximport solves issues with artificially inflated gene counts derived from transcript isoforms

Downloading genomes

I knew the next step would be to map the reads to the source genome, and for that I, of course, needed the mouse genome.

I found here that NCBI has two command line tools for Linux that ease the download of datasets. From within the NGS environment and in a new folder "Genomes", I used `curl` to get the installers as shown on the NCBI page, and then made the binaries executable as instructed.

I then proceeded to download the genomes that are relevant to me (mouse and human) using the option to find them by accession number. I got the accessions for the latest assemblies here and here. To be on the safe side, I decided to use the more elaborately curated RefSeq assemblies, so the commands looked as follows:

```
(NGS) iweber@iweber-VirtualBox:~/Documents/Genomes$ datasets download genome accession GCF_000001405.40 --filename genome_H_sapiens_GRCh38.zip
```

...and got a big, fat load of nothing xD, because the shell said it can't find the command "datasets". Instead of fumbling to find where the binaries associated with these two tools are, I decided to simply use conda to install them directly into this environment.

`conda create -n ncbi_datasets` conda activate ncbi_datasets conda install -c conda-forge ncbi-datasets-cli`

```
(NGS) iweber@iweber-VirtualBox:~/Documents/Genomes$ conda activate ncbi_datasets
(ncbi_datasets) iweber@iweber-VirtualBox:~/Documents/Genomes$ conda install -c conda-forge ncbi-datasets-cli
Channels:
 - conda-forge
 - bioconda
 - defaults
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /home/iweber/anaconda3/envs/ncbi_datasets

  added / updated specs:
    - ncbi-datasets-cli


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    ncbi-datasets-cli-16.14.0  |       ha770c72_2        15.8 MB  conda-forge
    ------------------------------------------------------------
                                           Total:        15.8 MB

The following NEW packages will be INSTALLED:

  ca-certificates    conda-forge/linux-64::ca-certificates-2024.2.2-hbcca054_0
  ncbi-datasets-cli  conda-forge/linux-64::ncbi-datasets-cli-16.14.0-ha770c72_2


Proceed ([y]/n)? y


Downloading and Extracting Packages:

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
(ncbi_datasets) iweber@iweber-VirtualBox:~/Documents/Genomes$
```

In the Windows Subsystem for Linux (WSL), getting the SRA Toolkit and SRR run files from the preeclampsia study

Using the documentation at NCBI, I found that one needs a suite of programs called SRA Toolkit in order to losslessly convert the SRA files into FastQ files. I installed the toolkit using the instructions on the GitHub page of the NCBI, https://github.com/ncbi/sra-tools/wiki/ . I decided to do so under Ubuntu running on the Windows Subsystem for Linux (WSL). I have read that, for optimal memory usage, it is preferable to create a virtual machine running a Linux distribution rather than the WSL. However, given that this was my first shot at analyzing a relatively small SRA dataset (~2.8 GB), I concluded that this is good enough for starters.

I downloaded the SRA toolkit as a .tar.gz archive for Ubuntu from one of the FTP servers of the NCBI:

```
wget --output-document sratoolkit.tar.gz https://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/current/sratoolkit.current-ubuntu64.tar.gz
```

then extracted it in one of my Ubuntu folders with the tar utility (included in Ubuntu),

```
tar -vxzf sratoolkit.tar.gz
```

and could afterwards access its tools over the command line.

As per the instructions of the GitHub page, I modified the PATH variable in my system. This essentially tells Ubuntu that there is a shortcut to the folders where SRA Toolkit has its binaries (bin directory) so that I can then easily access the individual tools and commands from the toolkit via the command line without needing to constantly change directories to that in which I extracted the SRA toolkit.

```
export PATH=$PATH:$PWD/sratoolkit.3.1.0-ubuntu64/bin
```

This should tell the shell that, when looking for binaries (tools, programs, etc that are executable), it should also look in the respective folder of the SRA Toolkit.

To verify whether the binaries can be found by the shell, I used

```
fastq-dump --stdout -X 2 SRR390728
```

...which worked as expected with returning the path to the directory of binaries precisely until I restarted the terminal :) I then found out that "export" only does a temporary modification of the PATH environment variable, which isn't kept in the global namespace. So, after advice from Christoph, I found out I need to only add the exact same line, minus "export", into the .profiles file, which should save any customized variants of the PATH variable. And now I have the SRA Toolkit set up to work from any directory, and even after restarting the shell.

Additionally, I created a handy alias for changing the directory to where my datasets are stored by adding the following under "Some more aliases" in the .bashrc file:

```
alias mydatasets='cd /home/iweber/Datasets'
```

So now I can go straight to that folder with only typing "myd" and hitting tab.

I also undertook the Toolkit Configuration as directed by the GitHub page of SRA toolkit using `vdb-dump -i` .double check command One noteworthy difference to how the configuration window looks like for me vs the website is that I do not have, under Main, the frame with options regarding to Workspace name and Workspace location.
I wonder if this is a matter of working on WSL instead of a full Linux distro?

However, in the tab Cache, I did set the location of the user-repository and process-local to be my Datasets folder, so that I may always find my files easily. I also created a subdirectory for my current study so that my data stays organized.

WSL - Extracting FastQ files from the SRR archives

And now that the tools are set up, I went on to try and extract the FastQ files from my dataset in SRA format. In my Datasets directory, I used the fasterq-dump command together with my dataset as a command-line argument:

```
fasterq-dump SRR13761520
```

And waited. I was afraid nothing is happening and I just went into some infinite loop, but then I navigated in Windows to the Datasets folder in WSL and saw that a temporary directory had been created. Since the terminal also didn't show me a fresh command prompt, I assumed it was just working intensely in the background to convert the data. Also, my ventilators were going crazy. s, when I checked the Task Manager, I saw that nearly 60% of my memory was in use by a process called "Vmmem". Sure enough, a brief web search uncovered that this happens precisely when using WSL, and is the Windows process responsible for the virtual memory managementDouble check!. All of this was happening before switching my RAM chips from a total of 32 GB to a total of 128 GB :) Yes, I am aware that I will need to set up a few more things, such as hyper-threading, for the new RAM to work as effectively as possible, but I was reluctant to do this at first, as my first experience after getting my laptop had been it crashing and refusing to start Windows upon installing a VirtualBox virtual machine and tinkering with the virtualization settings).

I had read on the GitHub page of the SRA Toolkit that the extraction of FastQ files requires a RAM size up to as much as 10 times the size of the original SRA file, at least during the extraction process (scratch space - had run into issues with this with Photoshop before, which kept complaining that the scratch disks were full and it therefore couldn't save whatever new artsy monster file I had created). So I was expecting Task Manager to tell me that around 25 of my 32 GB of RAM (~78% of it) were in use by the Vmmem process. The process started out at around 50%. Half an hour of observation later, I realized that it kept increasing over time. I did see it go up to a max of 75.6% with a fairly steady increase over time, so I assume the Toolkit kept adding info to the scratch disks as the process progressed. Looking in the Resource Monitor of Windows, vmmem said it was using a Commit of 10,470,000 and a Working set of about the same size. I suspect Task Manager is not as accurate as the Resource Monitor ?

To check the progress of the unpacking, I peeked at the size of the temporary folder in the SRA folder of this study in Datasets. When I first looked at it, some 3-4 minutes after starting the extraction tool, it was at around 3.8 GB. Some 10 minutes after starting the extraction, it was at 6.5 GB.

At this point, I started wondering where exactly WSL was storing its files in a physical sense (I know, I know, should've done so sooner, etc). It being a subsystem of Windows, I strongly suspected it would also be on my Windows drive, which, unfortunately, is the smaller of my two SSDs, at 250 GB. And indeed, when checking the properties of my Windows drive, I realized that the free space on this drive was decreasing. Being fairly close to the

size limit of the drive (~10 GB free), I was concerned that this might completely kill Windows if working on such a full drive or result in a corrupted, incomplete extraction of the FastQ files. So I started uninstalling any programs I saw were using a lot of space with their files and freed up some 10 more GB of RAM.

I also noticed that the Toolkit had also created another folder under Datasets called "sra", with a .sra.cache file of a only slightly larger size as the original SRA dataset (2,829,033 KB for the dataset and 2,829,054 KB for the .sra.cache file). This did not change in time.

It took around one hour for the extraction to finish, and I then got this:

```
iweber@Lenovo-Ioana:~/Datasets/Pre_eclampsia_mice$ fasterq-dump SRR13761520
spots read      : 21,192,945
reads read      : 42,385,890
reads written   : 42,385,890
iweber@Lenovo-Ioana:~/Datasets/Pre_eclampsia_mice$
```

And bam! I had two FastQ files in my study directory, each at exactly 7,954,345 Kb, so 7.9 GB and a total of around 16 GB. All in all, that's around 5.7x larger than the original SRA file.

But why were there two files with the same accession number but with _1_ or **2** appended to the file name? The answer was fairly easy to find: these were reads from paired-end sequencing runs, as indicated on the NCBI website for this dataset and the paper's methods. Sequencing both ends of the cDNA fragments resulting from the cortex RNAs allows for more precise alignment [source] and is thus more useful especially in detecting alternative splicing isoforms, which may be of interest to me later.

I opened the two behemoth files by ~~torturing~~ convincing Notepad++ to do so, and, at first glance, they looked identical.

After this step, I also realized I had severely underestimated exactly how many files I'd have to download and extract, and to what space requirements that would amount to :teary-smile:

So, the first thing I did was to...get a larger SSD for my OS. I initially had a 256 GB NVMe and switched to a 4 TB instead.

Temporary fix for space issues: change location where WSL stores its files

But, for the time being: how could I change where Windows saves the files of the WSL? Thank goodness that I'm not yet at a level at which my questions haven't yet been asked by anybody on the Internet. I found this set of instructions by NotTheDr01ds on SuperUser: https://superuser.com/questions/1736576/change-the-storage-location-of-a-wsl2 I decided to execute them on the basis that they also create a backup image of Ubuntu on the WSL as I had it set up before, that the code was clearly commented and easy to understand in terms of what environment variables were being modified how, and that the user seemed to have taken extra time to spell out precautions and build in safety checks. This is what I ran:

```
PS C:\Users\Ioana> wsl -l -v
  NAME      STATE           VERSION
* Ubuntu    Stopped         2
PS C:\Users\Ioana> $WSL_ROOT="D:\WSL"
PS C:\Users\Ioana> $WSL_IMAGE_PATH="${WSL_ROOT}\images"
PS C:\Users\Ioana> $WSL_OLD_DISTRO_NAME="Ubuntu"
PS C:\Users\Ioana> Get-ChildItem HKCU:\Software\Microsoft\Windows\CurrentVersion\Lxss\ |
>>     ForEach-Object {
>>         (Get-ItemProperty $_.PSPATH) | Select-Object DistributionName,BasePath
>>     }

DistributionName BasePath
---------------- --------
Ubuntu           C:\Users\Ioana\AppData\Local\Packages\CanonicalGroupLimited.Ubuntu_79rhkp1fndgsc\LocalState


PS C:\Users\Ioana> $WSL_NEW_DISTRO_NAME="Ubuntu_WSL2"
PS C:\Users\Ioana> $WSL_INSTANCE_PATH="${WSL_ROOT}\instances\${WSL_NEW_DISTRO_NAME}"
PS C:\Users\Ioana> mkdir $WSL_IMAGE_PATH


    Directory: D:\WSL


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         3/14/2024   9:35 PM                images


PS C:\Users\Ioana> mkdir $WSL_INSTANCE_PATH


    Directory: D:\WSL\instances


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         3/14/2024   9:35 PM                Ubuntu_WSL2


PS C:\Users\Ioana> wsl --shutdown
PS C:\Users\Ioana> wsl --export $WSL_OLD_DISTRO_NAME "${WSL_IMAGE_PATH}\${WSL_OLD_DISTRO_NAME}.backup.tar"
Export in progress, this may take a few minutes.
The operation completed successfully.
PS C:\Users\Ioana> wsl --import $WSL_NEW_DISTRO_NAME $WSL_INSTANCE_PATH "${WSL_IMAGE_PATH}\${WSL_OLD_DISTRO_NAME}.backup.tar" --version 2
Import in progress, this may take a few minutes.
The operation completed successfully.
PS C:\Users\Ioana> wsl ~ -d $WSL_NEW_DISTRO_NAME
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.146.1-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

This message is shown once a day. To disable it please create the
/root/.hushlogin file.
root@Lenovo-Ioana:~# sudo -e /etc/wsl.conf
root@Lenovo-Ioana:~# wsl --shutdown
Command 'wsl' not found, but can be installed with:
apt install wsl
root@Lenovo-Ioana:~# exit
logout
PS C:\Users\Ioana> wsl --terminate $WSL_NEW_DISTRO_NAME
The operation completed successfully.
PS C:\Users\Ioana> wsl --set-default $WSL_NEW_DISTRO_NAME
The operation completed successfully.
```

And it seems to have worked. Also, this was my first time seeing both Windows ***and*** Linux command lines mixed in the same window and, as the beginner that I am, it blew my mind. I realized that, basically, all I'm doing when starting the Ubuntu terminal I can also do from PowerShell window, provided that I started the WSL within it.

With WSL open in the PowerShell terminal, I did several checks as to whether the new distro at the new location was working properly:

```
PS C:\Users\Ioana> wsl ~
iweber@Lenovo-Ioana:~$ wsl.exe -l -v
  NAME            STATE           VERSION
* Ubuntu_WSL2     Running         2
  Ubuntu          Stopped         2
iweber@Lenovo-Ioana:~$ myfiles_exercises
iweber@Lenovo-Ioana:~/ABI/Exercises_Ioana$ ls -lat
total 48
drwxr-xr-x  4 iweber iweber 4096 Mar  4 10:47 Linux
drwxr-xr-x 12 iweber iweber 4096 Mar  4 09:25 .
drwxr-xr-x  2 iweber iweber 4096 Mar  1 12:32 Bash_scripting_Ioana
drwxr-xr-x  3 iweber iweber 4096 Feb  6 09:56 Databases
drwxr-xr-x  2 iweber iweber 4096 Feb  6 09:55 Biopython
drwxr-xr-x  7 iweber iweber 4096 Feb  6 09:46 ..
drwxr-xr-x  4 iweber iweber 4096 Feb  6 09:45 Alignments
drwxr-xr-x  2 iweber iweber 4096 Feb  6 09:45 Analysis_and_Stats
drwxr-xr-x  2 iweber iweber 4096 Feb  6 09:45 NGS
drwxr-xr-x  6 iweber iweber 4096 Feb  6 09:45 Python
drwxr-xr-x  2 iweber iweber 4096 Feb  5 13:01 Structural_Bioinfo
drwxr-xr-x  2 iweber iweber 4096 Feb  5 13:01 Working_w_Strings
iweber@Lenovo-Ioana:~/ABI/Exercises_Ioana$
```

I started WSL, then tested which distro was running using the code provided by NotTheDr01ds. In addition to that, I checked whether I could still use an alias I had set in my .bashrc some time ago, which simply changes the working directory directly to one where I store files for the ABI course. I also listed the contents of the directory, just to be extra sure that they are what I know them to be and that my default user is still the owner and has all of the permissions, as I know this can be difficult to alter should I not be the owner any longer ***source?***

And I then de-registered my old distro from WSL.

A consequence that I hadn't anticipated is that now I cannot use the Ubuntu app from the store directly any longer, at least not without creating a completely new distro...but that's easily fixed by running Ubuntu from the Windows Terminal instead, which I've taken to doing.

Result: successfully changed location, freeing some space on my currently plighted OS drive. I will need to change it back once I exchange the OS drive to a larger one, but at least the datasets I have stored in WSL and all of my course materials are safe for now on the other SSD.

Considering to make regular .tar/VHD backups of the WSL on the second SSD just to be on the safe side

I then wanted to get and extract all of the other SRA archives related to the E 17.5 cortices in this study. To do so, I wrote a Bash script containing the commands for fetching the remaining SRA archives (`prefetch` command of the SRA toolkit) and to unpack them (`fasterq-dump` command), set it to executable, and started it from the shell.

> **Note**
>
> script?

I checked the progress on occasion via TeamViewer, and saw things moving along through the continual increase in the size of the export folder. As a side note, here I also saw that the WSL virtual machine was still using up almost half of my RAM during this operation, and I think this is due to the limitations that WSL has in terms of memory usage, at least when compared to a virtual machine, so I decided I must take the plunge and create one. (see linked post about VM)

Quality control of FastQ files with the RNA-seq reads (still on WSL)

Quality control of individual FastQ files

For the quality control of this small number of files, I used FastQC, which is an open-source tool that checks the reads in FastQ files for various quality parameters, such as sequence quality scores, GC content, sequence duplication levels, or the presence of sequences left behind by the oligonucleotide adapters used for the amplification and actual sequencing of the cDNA fragments. I was pleasantly surprised that opening the application from the Ubuntu command line resulted in a user-friendly GUI, with which I could select the sequences to be analyzed. After analyzing the two FastQ files resulting from the first sequencing run (SRR13761520_1 and SRR13761520_2), I got quite different results. Whereas the file ending in 2 passed the quality checks in all but one department ("Per base sequence content" gave a warning), the file ending in 1 had a few more issues. It failed the "Per base sequence content" testing and gave warnings for "Per sequence GC content" and "Sequence duplication levels". I suppose this difference results from one of the files representing one item from each pair of reads and the other file the sequence complementary to it [source?]. As these two reads stem from two separate sequencing reactions [source], it is normal that one set can have different sequencing quality, based on differences in the reaction setup, base composition, or technically-introduced biases that only happened for one of the two sequencing runs.

Nonetheless, I wanted to understand better what the parameters returned by FastQC are and what they mean for the usability of the data for further analyses. To better understand how this data was pre-processed, I foraged a bit in the Series Matrix file provided on the GEO page of the study, but the authors don't give a lot of details about the handling of the data, aside from `Reads filtering under criteria removing reads with 20% of the base quality lower than 13"`. Considering that FastQC did not indicate the presence of any known adapter sequences in the data and that the files were already clearly sorted by experimental condition and subject, (I hope) it's safe to assume that any necessary demultiplexing and adapter trimming has been performed by the authors of the study/the sequencing facility they worked with, plus applying this quality filter they mention in the Series Matrix file.

To centralize all analysis info for all of the FastQ files generated by FastQC, I settled on MultiQC . I envision myself working with single-cell RNA-seq datasets at a later time point, and MultiQC can, as the name suggests, run the analyses in parallel on many files, making it a lot more efficient in handling large numbers of datasets. MultiQC is a part of the bcbio-nextgen Python package, so I wanted to install it in my Python on the WSL Ubuntu as instructed by the package's GitHub page. I thought this installed not only the package but also all dependencies (other modules) that are needed for it to run...but it didn't. It told me it was missing crucial data, such as the genome assemblies for mouse and human, which prevented it from running at all.

During the lengthy installation process, I kept getting a message for all packages "Solving environment: failed with initial frozen solve. Retrying with flexible solve.". Thanks to another kind internet stranger, I found that this is a problem that stems from having a version of Python that's newer than what many of these packages are optimized to work with. [what version do i have on my WSL? what versions needed? conda search python, then conda install python=desired_version_number, and then restart Ubuntu and Py, and update bcbio-nextgen package, hoping that this will resolve dependency issues]. However, the installation of bcbio-nextgen was still running (also, still running after 3h with the WSL virtual machine using 85 GB RAM sounds unusual and a bit alarming).

I realized that this may be due to WSL being less efficient in its memory usage than a full-fledged virtual machine [source?], so I proceeded to kill this process and install Oracle VirtualBox 7 and make an Ubuntu VM on it.

MultiQC: General quality stats of the pre-eclampsia FastQ files

I saw that multiqc takes one full folder as input. I could call `fastqc` from the command line individually for all of my FastQ files, but...why would I do something rather tedious to do for all of the 16 FastQ files? So I quickly wrote a small Bash script to automate the process:

```
#!/bin/bash

directory="/home/iweber/Documents/ABI_files_NGS/6484_fastq/"

for file in "$directory"/*.fastq; do
        if [ -f "$file" ]; then
                fastqc "$file" --outdir /home/iweber/Documents/ABI_files_NGS/FastQC_results
        fi

done
```

This simply cycles through all of the files with a .fastq ending in the indicated directory, stores the name of the file in the variable called file, and then calls fastqc to operate on the content of the file name variable ($file), while outputting the result into the FastQC_results folder.
I made it executable with `chmod` and then ran it, and:

```
(NGS) iweber@iweber-VirtualBox:~/Documents/Datasets/Pre_eclampsia_mice$ ./fastqc_loop_folder.sh
null
Started analysis of SRR13761520_1.fastq
Approx 5% complete for SRR13761520_1.fastq
Approx 10% complete for SRR13761520_1.fastq
Approx 15% complete for SRR13761520_1.fastq
Approx 20% complete for SRR13761520_1.fastq
Approx 25% complete for SRR13761520_1.fastq
Approx 30% complete for SRR13761520_1.fastq
Approx 35% complete for SRR13761520_1.fastq
Approx 40% complete for SRR13761520_1.fastq
Approx 45% complete for SRR13761520_1.fastq
Approx 50% complete for SRR13761520_1.fastq
Approx 55% complete for SRR13761520_1.fastq
Approx 60% complete for SRR13761520_1.fastq
Approx 65% complete for SRR13761520_1.fastq
Approx 70% complete for SRR13761520_1.fastq
Approx 75% complete for SRR13761520_1.fastq
Approx 80% complete for SRR13761520_1.fastq
Approx 85% complete for SRR13761520_1.fastq
Approx 90% complete for SRR13761520_1.fastq
Approx 95% complete for SRR13761520_1.fastq
Analysis complete for SRR13761520_1.fastq
null
Started analysis of SRR13761520_2.fastq
Approx 5% complete for SRR13761520_2.fastq
```

It worked! Naturally, I then ran MultiQC to summarize all of the reports obtained with FastQC.

```
(NGS) iweber@iweber-VirtualBox:~/Documents/Datasets/Pre_eclampsia_mice$ multiqc FastQC_results/

  /// MultiQC 🔍 | v1.21

|           multiqc | Search path : /home/iweber/Documents/Datasets/Pre_eclampsia_mice/FastQC_results
|         searching | ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 32/32
|            fastqc | Found 16 reports
|           multiqc | Report      : multiqc_report.html
|           multiqc | Data        : multiqc_data
|           multiqc | MultiQC complete
```

Let's have a more detailed look at the results.

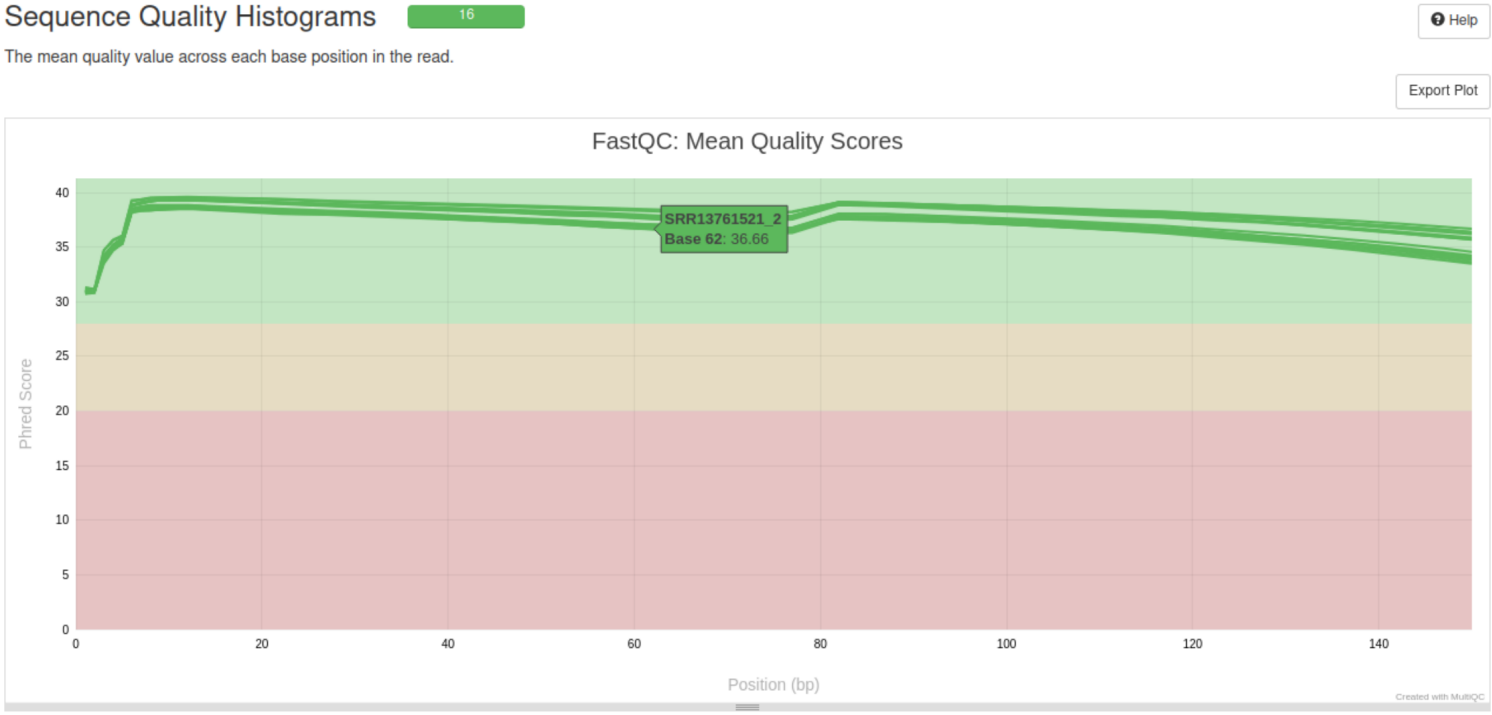
General Statistics

Copy table | Configure columns | Scatter plot | Violin plot | Showing 16/16 rows and 3/6 columns. | Export as CSV

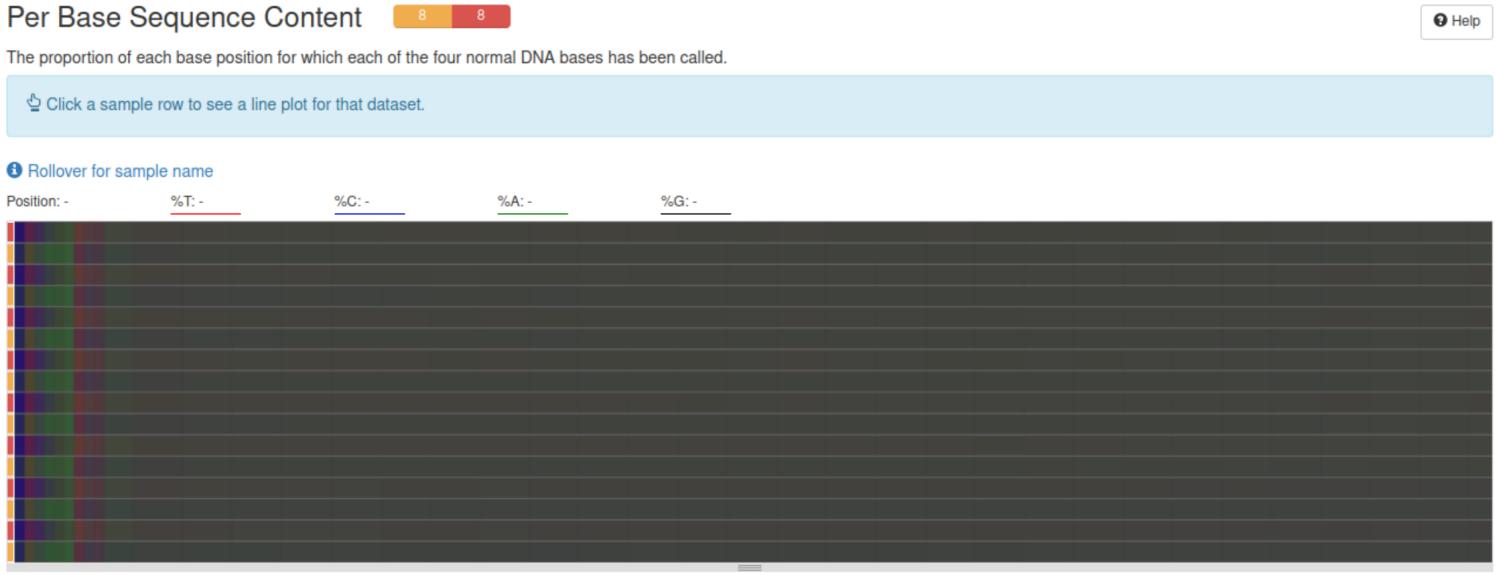
Sample Name	% Dups	% GC	M Seqs
SRR13761520_1	33.6%	48%	21.2M
SRR13761520_2	28.7%	48%	21.2M
SRR13761521_1	33.5%	48%	21.6M
SRR13761521_2	29.6%	49%	21.6M
SRR13761522_1	34.4%	48%	21.1M
SRR13761522_2	29.8%	49%	21.1M
SRR13761523_1	32.8%	48%	19.3M
SRR13761523_2	28.5%	49%	19.3M
SRR13761524_1	33.4%	48%	20.4M
SRR13761524_2	28.3%	49%	20.4M
SRR13761525_1	33.7%	48%	21.4M
SRR13761525_2	28.3%	48%	21.4M
SRR13761526_1	34.4%	48%	20.1M
SRR13761526_2	32.2%	49%	20.1M
SRR13761527_1	35.6%	48%	20.7M
SRR13761527_2	33.6%	49%	20.7M

So far, so good. Each of the files contains around 21 million reads, with a GC content that's normal for the mouse ([source?]). The amount of sequences flagged as duplicates is around 30%, which is to be expected due to the amplification steps during the library preparation ([quote from Ioana Lemnian]).

The Phred scores are, on average, also good across the entire length of the reads., even though, as usual, the quality at the beginning, in the first 10 nucleotides or so from the 5' end, is a bit lower than in the rest of the sequence.



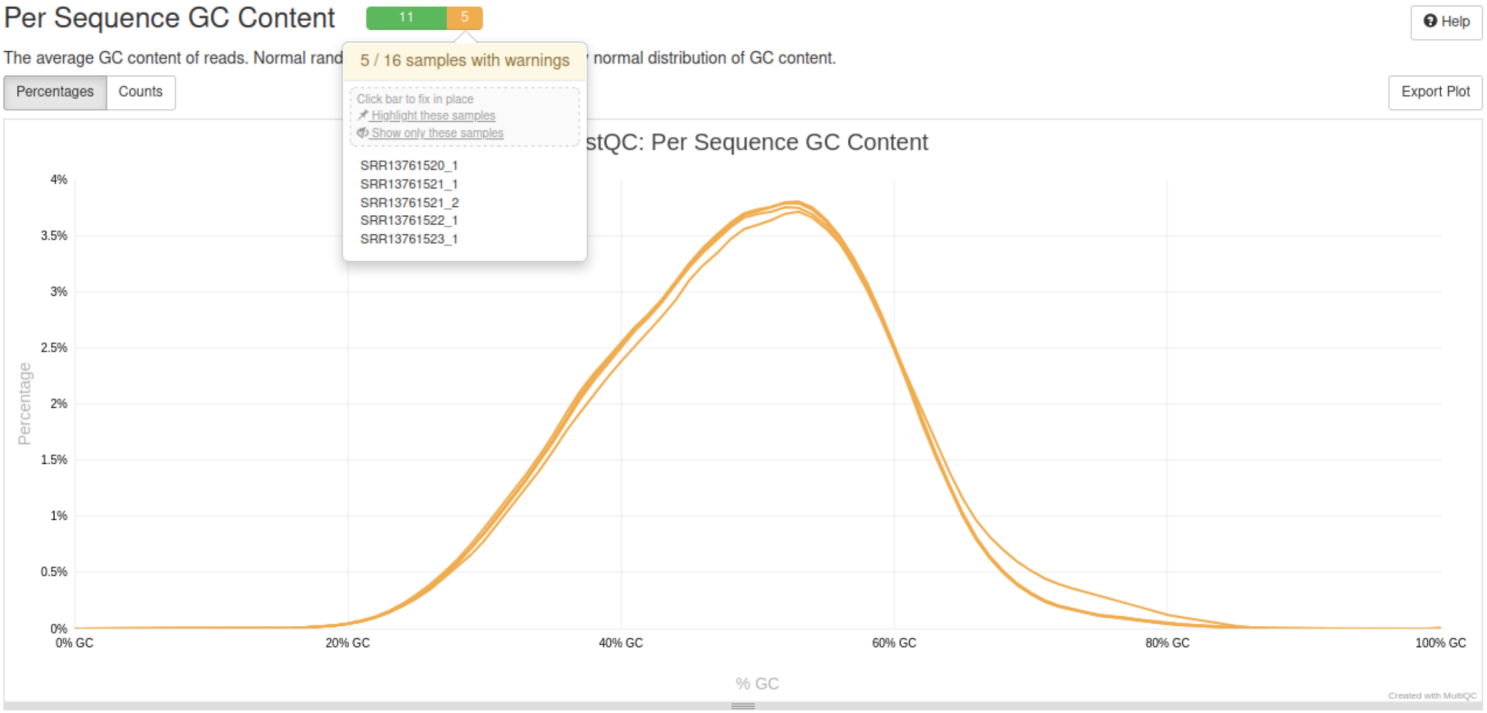
What looked a bit less rosy was the per base sequence content quantification. For all of the `_2` ending samples, FastQC gave a warning for the nucleotide composition, which should ideally be uniform across the read, with each base keeping its proportion percentage and giving a plot with four parallel lines. It's normal that the beginning of the read is a bit more noisy, up to 10 bases, and that's what we see here. Even for the samples ending in `_1` , that were flagged as "failed", the irregularities are restricted to this area. This looks like a good case for trimming these fragment start regions off in subsequent steps.



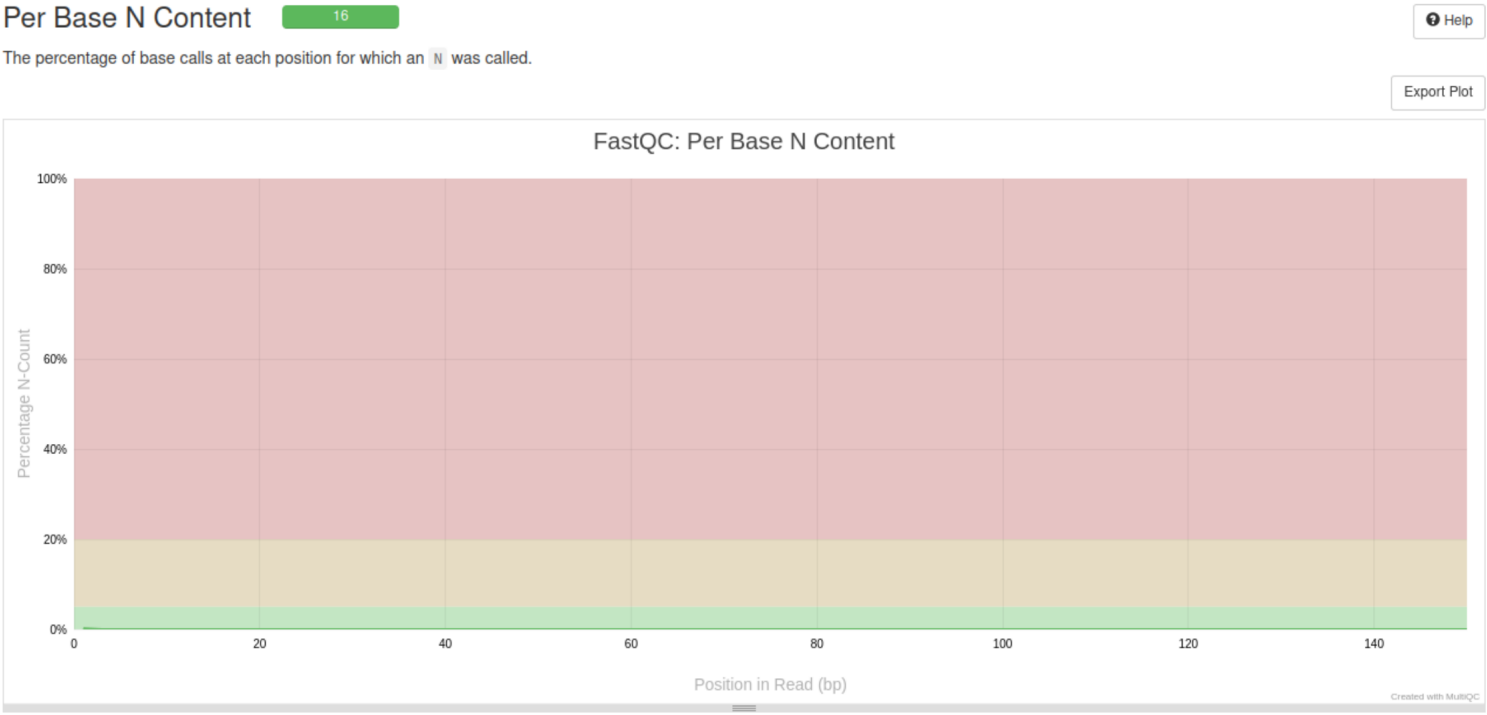
The per sequence GC content looked mostly alright (the reads in 11 of the 16 FastQ files passed with no warning), but there were some where FastQC gave a warning, so I know to keep an eye out in case these files cause some strange, systematic error down the line.

SRR13761520_1
SRR13761521_1
SRR13761521_2
SRR13761522_1

SRR13761523_1



The content of bases that could not be unequivocally identified during the sequencing was, fortunately, negligible.



Many of the FastQ files were found to contain reads from duplications. These usually result from the PCRs from the library preparation or even during sequencing itself, but the concentration of the second peak at 10x duplication suggests that these all stem from some 10-cycle-PCR. FastQC seems to flag any FastQ files where more than 8% of the reads are duplicates at the >10x mark but only those where the dupes surpass 11% of the read count for the 2x duplication mark [why?].

There were no warnings for overrepresented sequences, and FastQC did not find any of the usual sequences of adapters used in NGS kits in the FastQ files. This tells me that the files were processed before their further use, perhaps directly by the sequencing facility after demultiplexing [maybe ***bcl2fastq*** or ***BCL convert*** can also do this? Or maybe the authors just uploaded the already trimmed reads to GEO.]

Overrepresented sequences by sample 16 Help

The total amount of overrepresented sequences found in each library.

16 samples had less than 1% of reads made up of overrepresented sequences

Top overrepresented sequences

Top overrepresented sequences across all samples. The table shows 20 most overrepresented sequences across all samples, ranked by the number of samples they occur in.

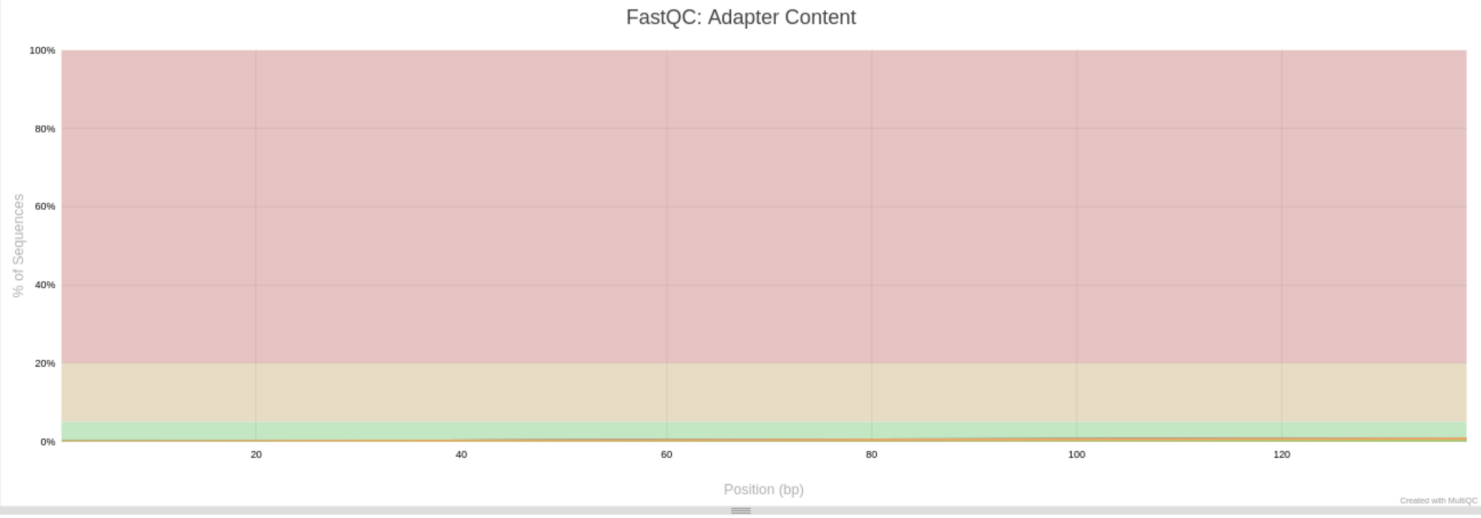
Copy table | Violin plot | Showing % rows. | Export as CSV

Overrepresented sequence

Adapter Content 16 Help

The cumulative percentage count of the proportion of your library which has seen each of the adapter sequences at each position.

Export Plot



Summary of initial quality warnings for the reads

Warnings:

Datasets with warnings for GC content:
SRR13761520_1
SRR13761521_1
SRR13761521_2
SRR13761522_1
SRR13761523_1

Dupe warnings:
SRR13761520_1
SRR13761521_1
SRR13761522_1
SRR13761523_1
SRR13761524_1
SRR13761525_1
SRR13761526_1
SRR13761526_2
SRR13761527_1
SRR13761527_2

Real solution for space issues: Making a VirtualBox virtual machine to run Ubuntu

Before taking the plunge to make a virtual machine, I made some very thorough backups of my system. I made a restore point, then also used File History and Backup and Restore to make images of my WinOS. Finally, I made a recovery USB in case I wouldn't be able to start the OS after making changes to the BIOS to enable virtualization. Maybe I wouldn't be this paranoid if tinkering with the virtualization options hadn't BSOD-ed my back-then three days old laptop right after setting it up out of the box for the very first time...so better safe than sorry.

Got an Ubuntu 22.04 image from the Canonical website, and installed it as VM with the name Ubuntu 22x64 using the Oracle VirtualBox. I allotted it 96 GB RAM, and 6 CPUs, to make sure it has all of the resources it might need to analyze large data sets.

I thought this worked at first, and then the issues started popping up like mushrooms after the rain: the GNOME terminal that comes with the distro didn't work, I couldn't update the Snap Store even though the update was recommended. I installed another terminal emulator called Guake and attempted to install the bcbio-nextgen tool bundle...and it got stuck with conda not being able to solve the environment (this issue seems to be known and caused due to R package dependencies that conda can't solve, or at least not quickly, see link). After stopping that process, I tried updating Python, which almost completely broke the entirety of the Ubuntu installation: the VM stopped showing me the Ubuntu desktop at all and only did so for the TTY terminal, with which I was not able to refresh the installation of the GUI desktop.

(Side note: I also tried the terminal emulator Terminator, whose options I also enjoyed).

I removed the VM completely and reinstalled everything from scratch...only to find out that, suddenly, my user, created during the VM creation process, had no `sudo` privileges and I had no way of accessing the root to give it said privileges. Removed the entire VM again.

Reinstalled Ubuntu 22.04 on a VM with the name ubuntu22x64. Gave it my username iweber and my usual password for the purpose, and selected guest additions from the suggested iso. Let's see if I now have sudo rights for my user... Nope.

Found a forum entry here saying that the issue can be that this unattended installation option that the VM provides creates an user without `sudo` rights. Skipped this option during the creation of the new VM, leaving me to install Ubuntu manually after the creation of the VM.

Did so, selected the keyboard and then to erase disk and install Ubuntu.

After the installation and restart, noticed that the first DOS-like screen had some errors talking about a graphics device and issues with unsupported hyper-visor, but Ubuntu started nonetheless and I could log in with user loWe (my user, with actual username iweber). ***LE: this keeps happening when the virtual machine powers up, but it still boots, and I am currently not seeing any other issues.***

And magic! I could even open the GNOME Terminal now!

I then let Software Updater update everything it said needed updating. It ran and I noticed it again giving some errors regarding Snaps...but then it finalized the update and said it needed to restart to apply all of the updates, so I let it do so.

After the restart, the terminal still works. I could also `whoami` myself AND `sudo whoami`, which, after providing my password, also let me set the user to root, so I now know I can install any software I may be needing down the line.

I went to the Anaconda website to find out how to download packages that I previously found out need to be installed for the Anaconda Navigator GUI (anaconda3) to work. The command I used in the terminal was:

```
sudo apt-get install libgl1-mesa-glx libegl1-mesa libxrandr2 libxrandr2 libxss1 libxcursor1 libxcomposite1 libasound2 libxi6
libxtst6
```

and it executed without me seeing any errors.

Tried to download the Anaconda Navigator installer Anaconda3-2024.02-1-Linux-x86_64.sh as recommended by the Anaconda Navigator page by using

```
curl -O https://repo.anaconda.com/archive/Anaconda3-2024.02-1-Linux-x86_64.sh`
```

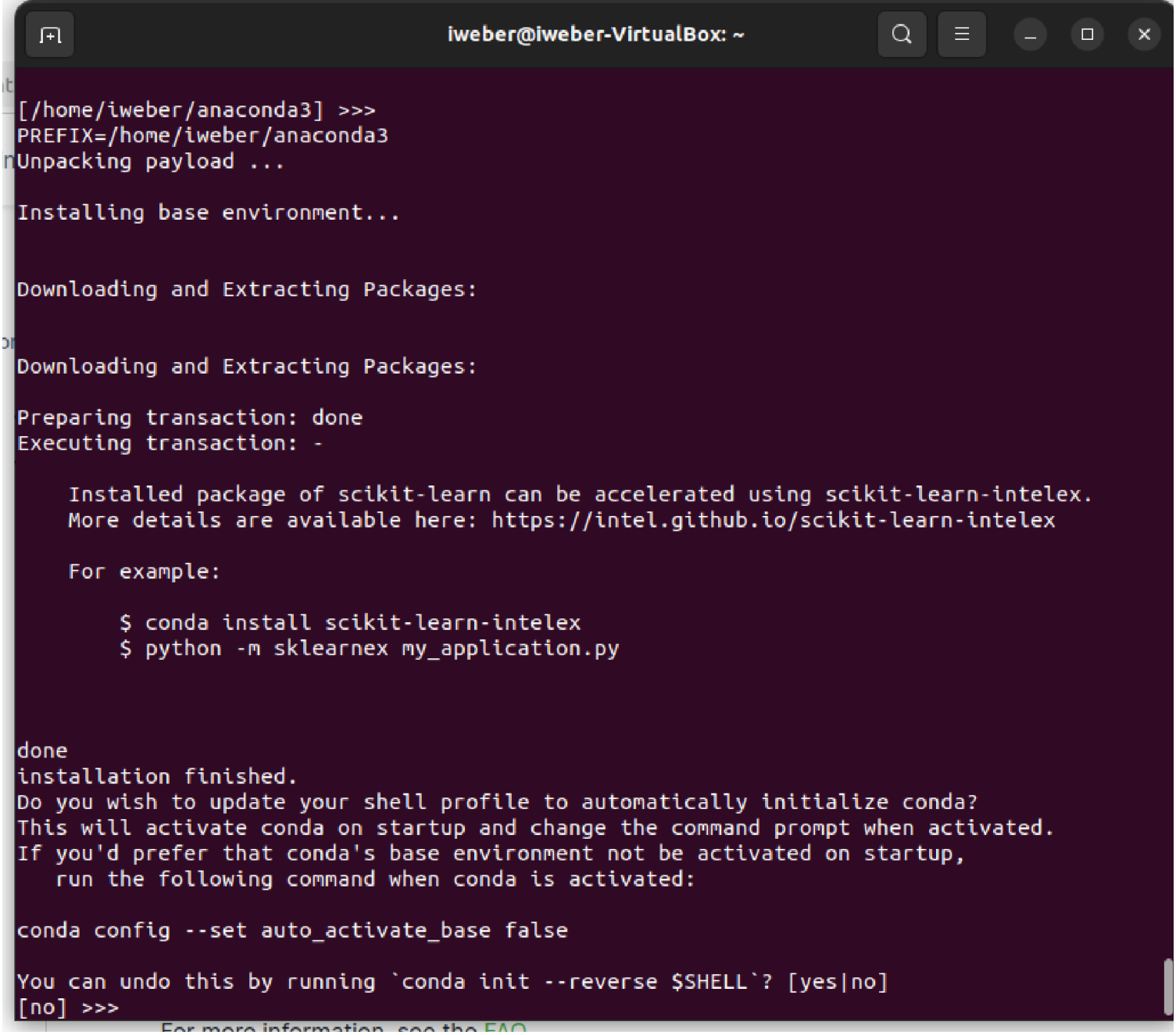
...and found out I first had to install the curl package (*cry-laugh*), which I quickly did with

```
sudo apt install curl
```

I could then finally run the curl command above to get the Anaconda Navigator. What was displayed during the installation was this:

```
iweber@iweber-VirtualBox:~$ curl -O https://repo.anaconda.com/archive/Anaconda3-2024.02-1
-Linux-x86_64.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  8  997M    8 82.9M    0     0  6819k       0  0:02:29  0:00:12  0:02:17 6633k
```

The percentage reached 100 and I had a new command line available, so I knew it was installed. I checked, as per the rec of the Anaconda Navigator install page, whether the checksum is correct using `shasum -a 256 Anaconda3-2024.02-1-Linux-x86_64.sh` (curl always downloads in the working directory). This check returned the expected SHA-256 hash for this version, so I knew I was good to go actually using the installer. Used `bash Anaconda3-2024.02-1-Linux-x86_64.sh` to run the installer, and what I saw was:


iweber@iweber-VirtualBox: ~
[/home/iweber/anaconda3] >>>
PREFIX=/home/iweber/anaconda3
Unpacking payload ...
Installing base environment...
Downloading and Extracting Packages:
Downloading and Extracting Packages:
Preparing transaction: done
Executing transaction: -
Installed package of scikit-learn can be accelerated using scikit-learn-intelex.
More details are available here: https://intel.github.io/scikit-learn-intelex
For example:
$ conda install scikit-learn-intelex
$ python -m sklearnex my_application.py
done
installation finished.
Do you wish to update your shell profile to automatically initialize conda?
This will activate conda on startup and change the command prompt when activated.
If you'd prefer that conda's base environment not be activated on startup,
run the following command when conda is activated:
conda config --set auto_activate_base false
You can undo this by running `conda init --reverse $SHELL`? [yes|no]
[no] >>>
For more information, see the FAQ

I confirmed that I wanted to let conda configure my shell profile so that it can be automatically initialized by entering `yes`.

```
done
installation finished.
Do you wish to update your shell profile to automatically initialize conda?
This will activate conda on startup and change the command prompt when activated.
If you'd prefer that conda's base environment not be activated on startup,
   run the following command when conda is activated:

conda config --set auto_activate_base false

You can undo this by running `conda init --reverse $SHELL`? [yes|no]
[no] >>> yes
no change     /home/iweber/anaconda3/condabin/conda
no change     /home/iweber/anaconda3/bin/conda
no change     /home/iweber/anaconda3/bin/conda-env
no change     /home/iweber/anaconda3/bin/activate
no change     /home/iweber/anaconda3/bin/deactivate
no change     /home/iweber/anaconda3/etc/profile.d/conda.sh
no change     /home/iweber/anaconda3/etc/fish/conf.d/conda.fish
no change     /home/iweber/anaconda3/shell/condabin/Conda.psm1
no change     /home/iweber/anaconda3/shell/condabin/conda-hook.ps1
no change     /home/iweber/anaconda3/lib/python3.11/site-packages/xontrib/conda.xsh
no change     /home/iweber/anaconda3/etc/profile.d/conda.csh
modified      /home/iweber/.bashrc

==> For changes to take effect, close and re-open your current shell. <==

Thank you for installing Anaconda3!
iweber@iweber-VirtualBox:~$
```

This seems to have worked. I closed and reopened my shell and checked if I can use the `conda` command to open the manager.

Success!

```
(base) iweber@iweber-VirtualBox:~$ conda
usage: conda [-h] [-v] [--no-plugins] [-V] COMMAND ...

conda is a tool for managing and deploying applications, environments and packages.

options:
  -h, --help            Show this help message and exit.
  -v, --verbose         Can be used multiple times. Once for detailed output,
                        twice for INFO logging, thrice for DEBUG logging, four
                        times for TRACE logging.
  --no-plugins          Disable all plugins that are not built into conda.
  -V, --version         Show the conda version number and exit.

commands:
  The following built-in and plugins subcommands are available.

  COMMAND
    activate            Activate a conda environment.
    build               Build conda packages from a conda recipe.
    clean               Remove unused packages and caches.
    compare             Compare packages between conda environments.
    config              Modify configuration values in .condarc.
    content-trust       Signing and verification tools for Conda
    convert             Convert pure Python packages to other platforms (a.k.a.,
                        subdirs).
    create              Create a new conda environment from a list of specified
                        packages.
    deactivate          Deactivate the current active conda environment.
    debug               Debug the build or test phases of conda recipes.
    develop             Install a Python package in 'development mode'. Similar
                        to `pip install --editable`.
    doctor              Display a health report for your environment.
    index               Update package index metadata files.
    info                Display information about current conda install.
Applications            Initialize conda for shell interaction.
```

And now I finally had hope of being able to install the bcbio-nextgen package.

...but then I realized I still couldn't see the Anaconda Navigator as a GUI version at this point. I saved the state of my VM and restarted it...and quickly realized that this isn't a restore point like in Windows, but a genuinely frozen point in time with all open apps, so more like a hibernation in Windows. So I then truly restarted it from the VirtualBox menu.

I then researched and found that Anaconda Navigator does not show up as an app in the Ubuntu Apps, but can still be opened by running the following very simple command in the terminal:

```
anaconda-navigator
```

And yes, this opened the software! (after some warnings relating to GNOME and rescaling) I chose to update as recommended. However, I did not find the bcbio-nextgen package listed among the packages that could be installed, in spite of creating environments

I discovered that these issues may also be fixed if using mamba as a dependency solver instead of conda. I read the docs of how to install Mamba here. It seems I need to install Miniforge, which seems to be the most recent version, as opposed to using Mambaforge. I read that I must install Miniforge (Mamba) in the so-called base environment and that only it and conda, and no other packages whatsoever, may be installed in the base, because other packages could break both environment managers.

I downloaded the Miniforge .sh binary and ran it in the terminal with `bash Miniforge3-Linux-x86_64.sh`.

It worked:

```
(base) iweber@iweber-VirtualBox:~$ mamba
usage: mamba [-h] [-v] [--no-plugins] [-V] COMMAND ...

conda is a tool for managing and deploying applications, environments and packages.

options:
  -h, --help            Show this help message and exit.
  -v, --verbose         Can be used multiple times. Once for detailed output,
                        twice for INFO logging, thrice for DEBUG logging, four
                        times for TRACE logging.
  --no-plugins          Disable all plugins that are not built into conda.
  -V, --version         Show the conda version number and exit.

commands:
  The following built-in and plugins subcommands are available.

  COMMAND
    activate            Activate a conda environment.
    clean               Remove unused packages and caches.
    compare             Compare packages between conda environments.
    config              Modify configuration values in .condarc.
    create              Create a new conda environment from a list of specified
                        packages.
    deactivate          Deactivate the current active conda environment.
    doctor              Display a health report for your environment.
    info                Display information about current conda install.
    init                Initialize conda for shell interaction.
    install             Install a list of packages into a specified conda
                        environment.
    list                List installed packages in a conda environment.
    notices             Retrieve latest channel notifications.
    package             Create low-level conda packages. (EXPERIMENTAL)
    remove (uninstall)
                        Remove a list of packages from a specified conda
                        environment.
    rename              Rename an existing environment.
```

Followed installation instructions from the bcbio-nextgen page: https://bcbio-nextgen.readthedocs.io/en/latest/contents/installation.html#automated

First, installed Git from git-scm using `sudo apt-get install git`. Checked if I have tar installed by simply typing `tar` in my terminal, and yes, I do.

I then started the installation by calling python3 and, using it, the Py script that is supposed to install bcbio-nextgen. I used the options --nodata to do a minimal installation, without genomes etc, and --mamba to use mamba as a package manager. The command looked like:

```
python3 bcbio-nextgen-install.py /home/iweber/bcbio --tooldir=/home/iweber/bcbio/tools --nodata --mamba
```

I noticed it said it was installing mamba, and am not sure why - maybe because I am not installing bcbio within the mamba directory? But I thought that that amounts to installing in the base environment and is not recommended? Either way, the installation did not succeed. It also told me I was using a deprecated version of conda, so I updated it to the last version (24.3 at the time of writing this).

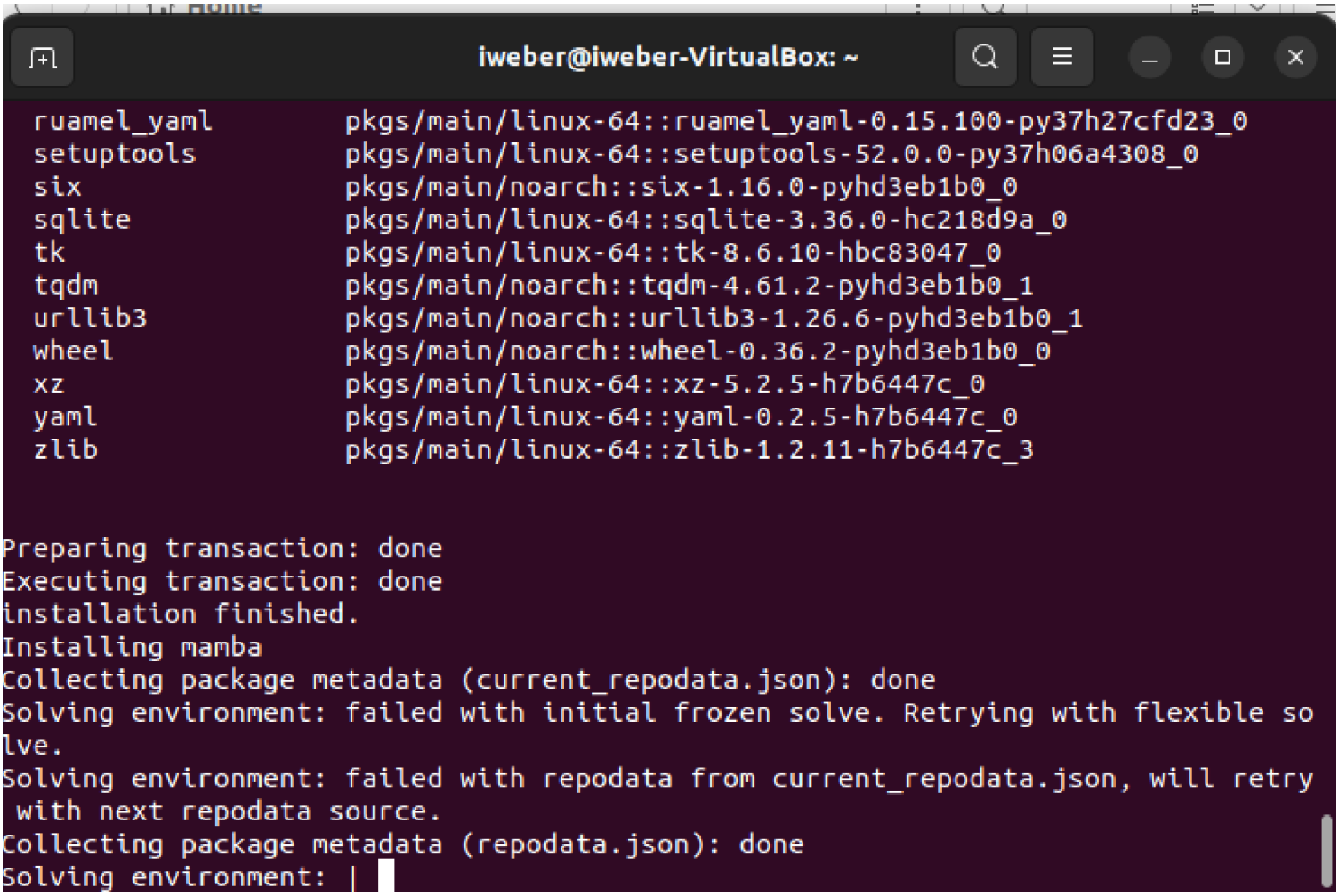
LE: I think the Py installer script for bcbio-nextgen installs Miniconda and uses that to solve the environment/package dependencies, and that's why it pops up as deprecated, even though I just installed the most recent version of conda

Can I somehow force bcbio installer to use the properly installed anaconda3 instead of Miniconda as a solver? - maybe not a good idea. If installer calls for a particular version of conda, it may have a good reason in terms of package dependencies.

After this, whenever I tried running the installation script again, it told me it cannot create the data directory required for the installation, a positional argument that needs to be given upon running the script. So I decided to run it with root privileges using

```
sudo python3 bcbio-nextgen-install.py /bcbio-data --nodata --mamba
```

in order to create a fully new data directory called bcbio-data, and, as before, to not download any genome data and to use mamba as an environment solver instead of conda. And now it seems like I'm stuck in the same solving environment step as has happened every time before when I tried it in the WSL:



I killed the process with `Ctrl+C`. I decided to create a separate conda environment called `bcbio-env` and, within it, to install one of the latest stable versions of Python to see if that helps matters. I had Python 3.10.12 and updated to 3.12.2.

Then, attempted the installation with root privileges in this environment with the same sudo command as before. Same problem with "`failed with intial frozen solve. Retrying with flexible solve.`".

After that, "`failed with repodata from current_repodata.json, will retry with next repodata source`"

Downgraded Python in my bcbio-env to python 3.9.0 and tried installing again with the sudo command. Also "`failed with intial frozen solve. Retrying with flexible solve.`". Killed process.

Upgraded Py to 3.12.2. Tried installing bcbio using pip3 ([pip is the Python package manager]). It said it worked, but, when calling `bcbio_nextgen.py --version`, it gave me an error message saying that the "six" package was not installed. I installed it manually with pip3 and tried calling the version of the package...and then got an error saying another package, toolz, was not installed. Did that as well, only to end up with another package error, this time for "yaml". However, when trying to install yaml via pip3, I got an error mesage that "no version could be found that satisfies requirements/no matching distribution found for yaml."

I ran `mamba clean -a` to remove all of these packages.

As the last strategy, I tried installing bcbio from bioconda as described on the anaconda page:
`conda install bioconda::bcbio-nextgen`

And, suddenly, I could not only use `which bcbio-nextgen` to see where it is installed (/home/iweber/miniforge3/envs/bcbio-env/bin/bcbio_nextgen.py) but ***even got a version number*** when looking for it with `bcbio_nextgen.py --version`, namely version `1.2.9`...which, indeed, is the latest version. At this point, I could've cried of happiness.

As life often goes, I realized only moments later that I can't actually download a new genome because the installation probably didn't create the correct folders for the data that bcbio-nextgen needs to run....

(Side note #2: another thing to remember is to start writing a logfile of everything I do in the terminal. For this, every time I start terminal, should run

`script logfile`

This automatically closes and creates that logfile when exiting terminal with Ctrl+D.

To get all packages currently installed in my conda environment:

`conda list --export > my_conda_packages.txt` [what was the point of this?]

I closed the VM at this point to come back to it another day...only to come back to not being able to open the Anaconda Navigator anymore from bash...

I added the path to the anaconda binaries folder to my PATH variable in the .profile file so that the bash knows to look here for executables as well, because I'd like to type less whenever I want to start the Anaconda Navigator: `export PATH=$PATH:/home/iweber/anaconda3/bin`. This, however, did not solve the issue.

I went ahead and updated the installation with `bash Anaconda3-2024.02-1-Linux-x86_64.sh -u
and got yet another weird error message.

And...the same errors over and over again. I looked further into this pipeline and decided its maintenance is lagging too far behind, as, for many other bioinformatic projects, the main contributors have, in the meanwhile, moved on to other projects.

I have decided to, instead, use Nextflow and established NGS data analysis pipelines that work on it.

Fixing Win-Linux clipboard

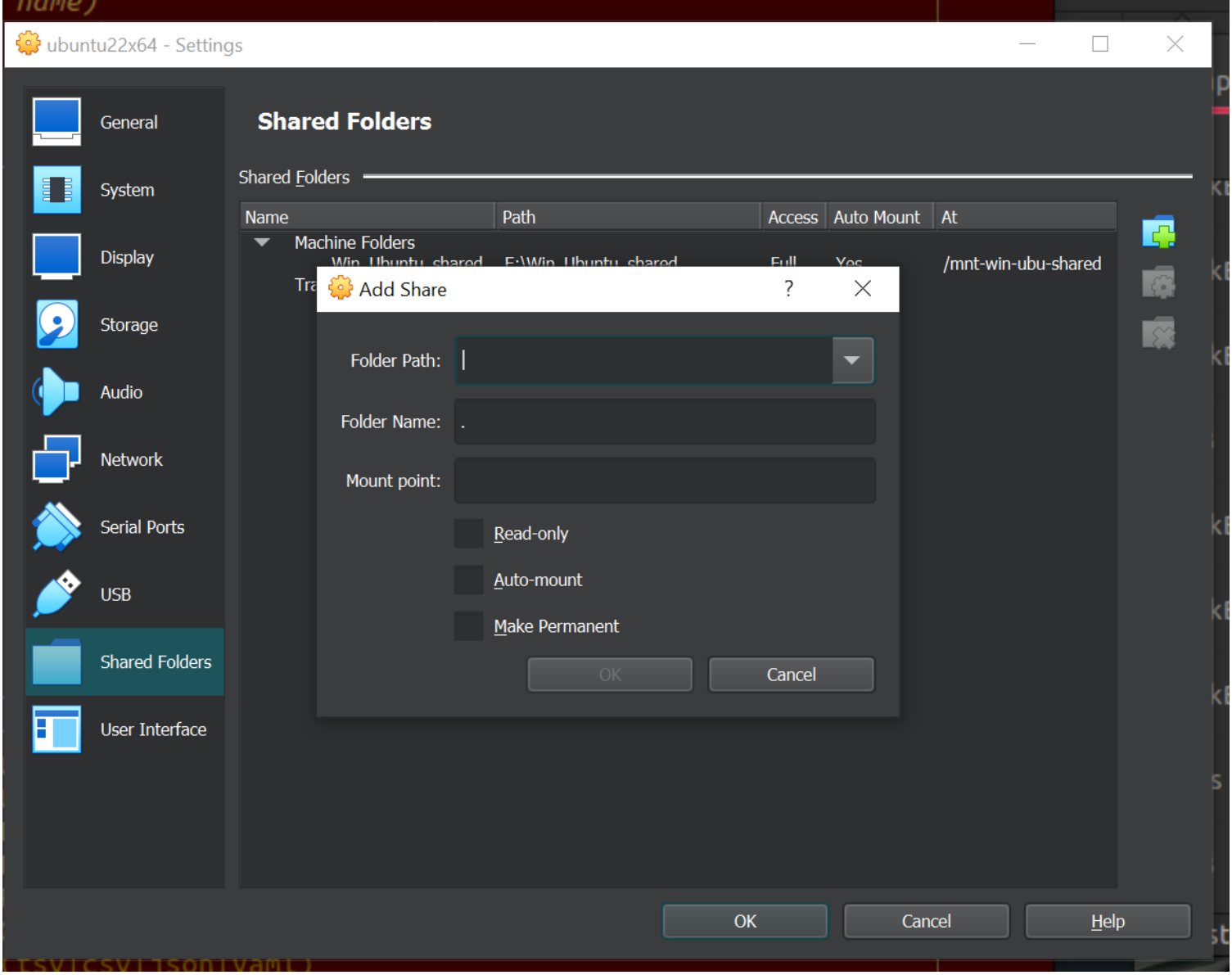
In the meanwhile, I tried fixing my highly annoying copy-paste and drag-and-drop issues between my virtual machine and my Windows host. I first installed:

`sudo apt-install build-essential dkms linux-headers-generic`,

following this answer and the page it refers to . I "mounted" the VA Guest Box addition disc image and ran the "autorun.sh" bash script it contains as a program (right click -> "Run as program"). After it asked for credentials, the terminal opened and I saw the expected screen. Restarted Ubuntu et voila! Copying and pasting now finally works.

Creating an inter-OS shared folder between Win and Linux

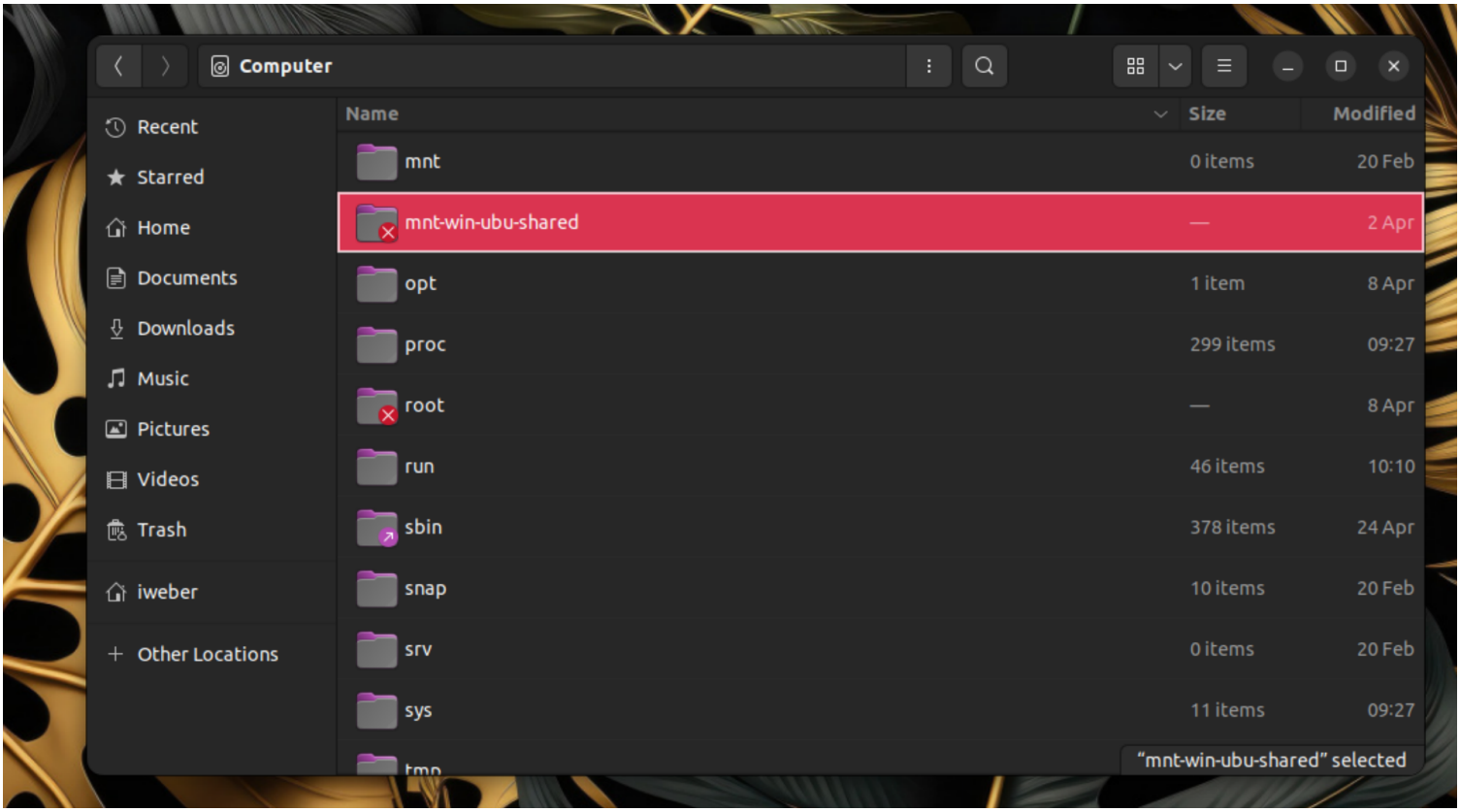
I had previously created a shared folder between my Windows host system and the Ubuntu virtual machine using the VM menu "Devices" -> "Shared folders" -> "Shared folders settings". Clicking the blue folder icon with a plus sign on it, I got such a window:



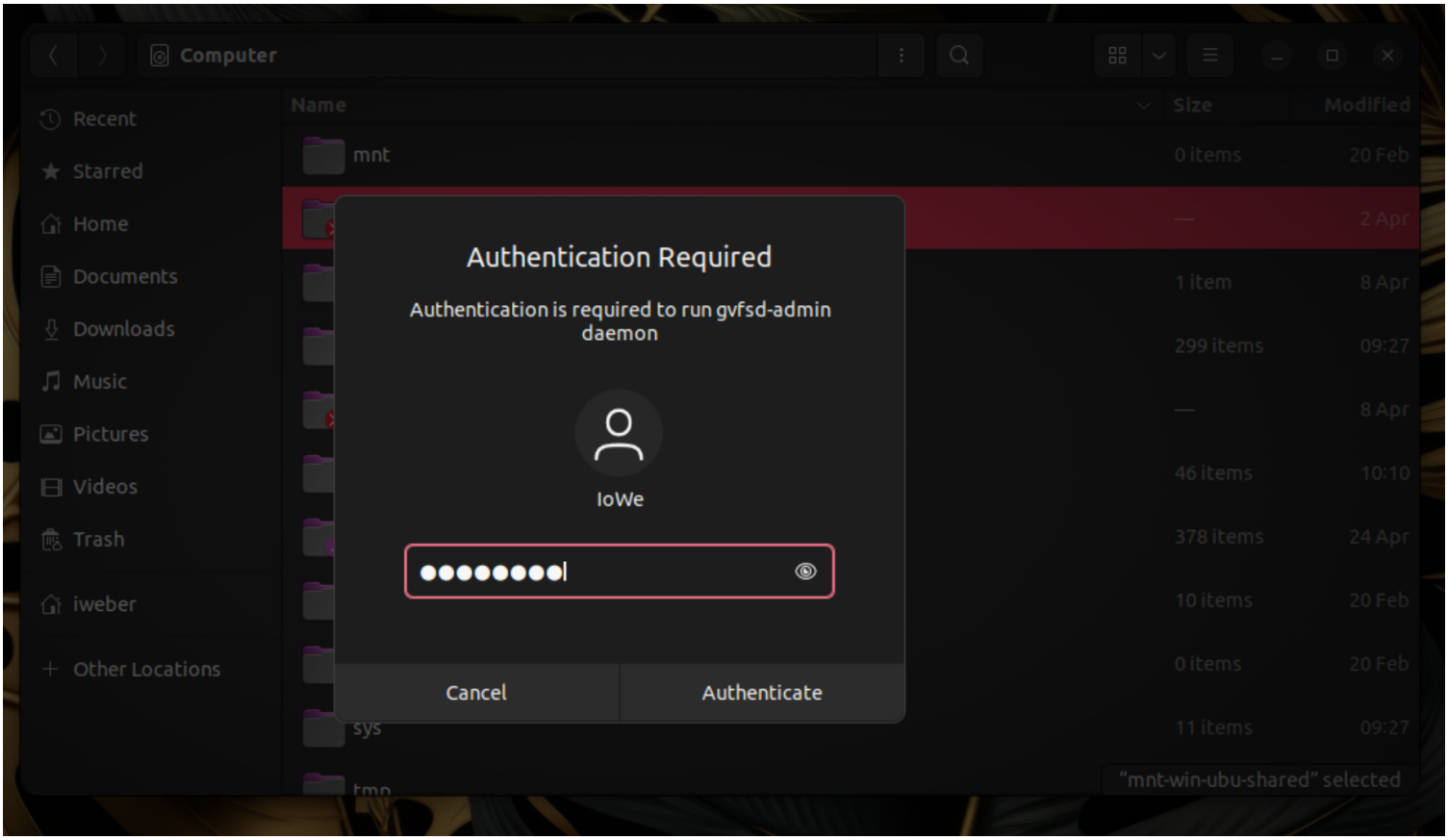
The "Folder Path" field allowed me to select a folder in my host to use as the shared folder. For this purpose, I went to Windows and, on the partition of the SDD that I had allotted to the virtual machine, I created a folder called "Win_Ubuntu_shared". I could then access this from the VM menu displayed above, and selected to auto-mount it and make it permanent in order to have it automatically accessible whenever I run the VM. I also gave it a mount point name in order to find it more easily under Linux, and chose "/mnt-win-ubu-shared" for this name.

But I didn't see this folder in my Linux file explorer, not even after restarting the virtual machine. I followed the instructions given here and ran `sudo usermod -aG vboxsf $USER` in terminal, and then tried accessing folder via Other locations -> Computer, where I finally saw the "mnt-win-ubu-shared"

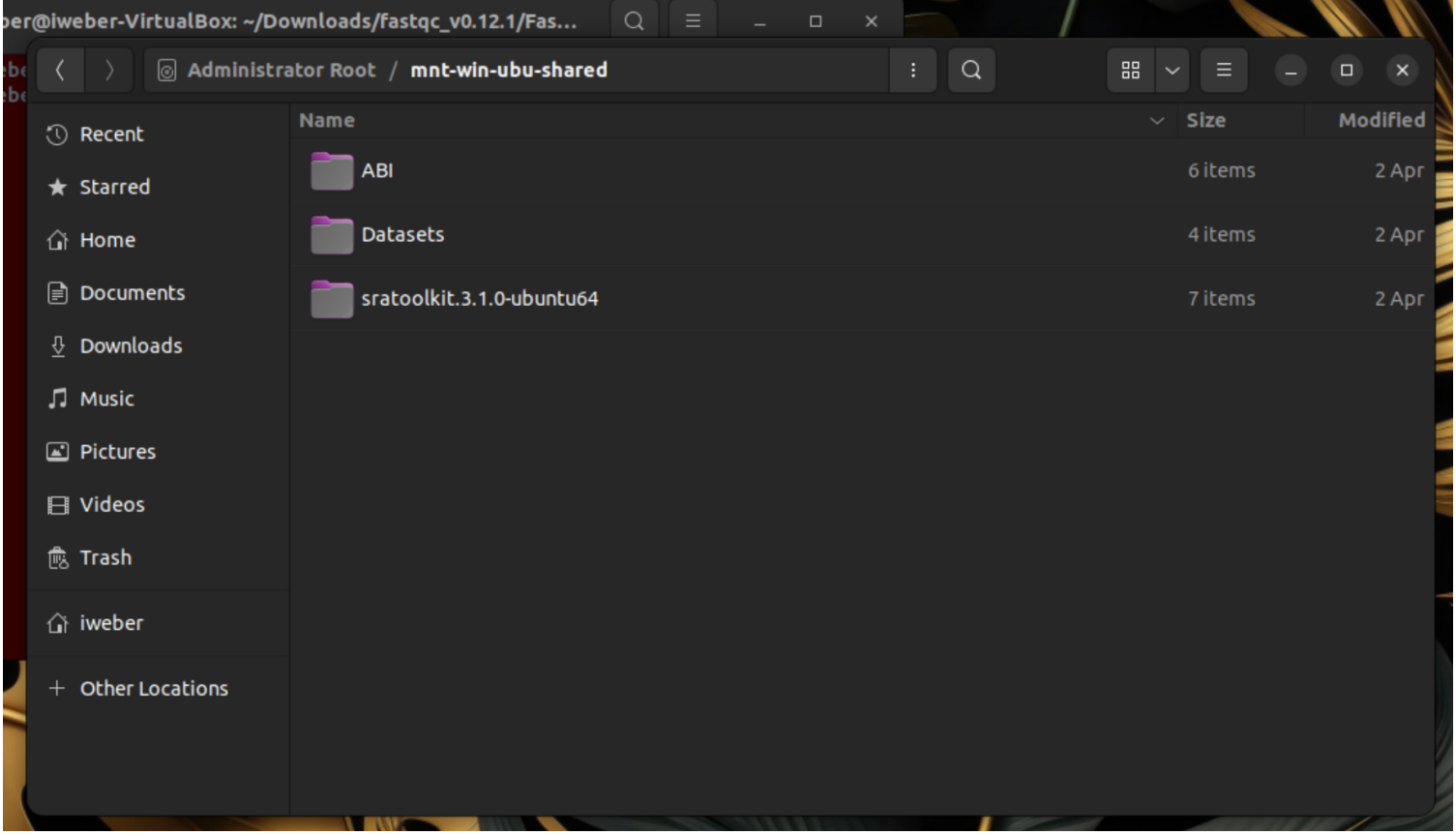
folder (remember, this is the name I gave the folder when setting it up in the virtual machine in the VirtualBox menu "Devices"> "Shared folders").



Got a message window asking me to also log in with my user credentials.



I did so, and I could finally see the contents of the shared folder! I immediately bookmarked this mnt-win-ubu-shared folder from the address bar of the explorer (three-dot menu) for easy access later.



I could now *finally* access the FastQ files I had previously generated on the WSL for further analyses!

Making the inter-OS shared folder auto-mount without asking for my password every time

I had previously created a folder for the internship data that's shared between my Windows host OS and Linux , and I realized I wasn't using it as the sole folder for my data in Linux because, every time I wanted to access it, it asked me for my user password. That made me reluctant to use it especially in the context of an automated analysis pipeline, because I feared that Linux suddenly forgetting the password might interrupt the pipeline at inconvenient times, e.g. when I'm not at my laptop. Additionally, this complicated setting up a proper GitHub repo. So, to fix that, I asked ChatGPT.

I used `mount | grep mnt-win-ubu-shared` to check where the folder is located. It returned `Win_Ubuntu_shared on /mnt-win-ubu-shared type vboxsf (rw,nodev,relatime,iocharset=utf8,uid=0,gid=999,dmode=0770,fmode=0770,tag=VBoxAutomounter)`, so I now knew the address of my mounted folder.

I added my user to the permissions group for the virtual box folder (`vboxsf`) with `sudo usermod -aG vboxsf $USER`, then ran a `chown` command to change ownership of the folder:
`sudo chown -R $USER:vboxsf /mnt-win-ubu-shared/

Now, when checking the permissions, I saw:

```
(base) iweber@iweber-VirtualBox:~$ mount | grep mnt-win-ubu-shared
Win_Ubuntu_shared on /mnt-win-ubu-shared type vboxsf (rw,nodev,relatime,iocharse
t=utf8,uid=0,gid=999,dmode=0770,fmode=0770,tag=VBoxAutomounter)
(base) iweber@iweber-VirtualBox:~$ ^C
(base) iweber@iweber-VirtualBox:~$ sudo usermod -aG vboxsf $USER
[sudo] password for iweber:
(base) iweber@iweber-VirtualBox:~$ sudo chown -R $USER:vboxsf /mnt-win-ubu-share
d/
(base) iweber@iweber-VirtualBox:~$ ls -l /mnt-win-ubu-shared/
total 8
drwxrwx--- 1 root vboxsf 4096 Mai 22 13:40 ABI
drwxrwx--- 1 root vboxsf    0 Apr  2 15:22 Datasets
drwxrwx--- 1 root vboxsf 4096 Apr  2 15:23 sratoolkit.3.1.0-ubuntu64
(base) iweber@iweber-VirtualBox:~$
```

so I knew I now have read and write privileges for the directory for both myself and the virtual box group I am in. I copied and pasted some files into this folder from both Windows and Linux to make sure it works bidirectionally, as it should, and it does.

Giving the inter-OS shared folder the correct permissions upon startup

However, upon restarting the virtual machine, Ubuntu still asked me for my password (specifically, "Authentication is required to run gvfsd-admin daemon") when trying to open the inter-OS shared folder...ChatGPT said this is likely due to startup permissions not being set properly and suggested I edit a file called "fstab" with admin privileges and add the path to the inter-OS shared folder. "fstab" is a "file systems table" with information about files, folders, and mount points [other source to confirm?]. I added `Win_Ubuntu_shared /mnt-win-ubu-shared vboxsf uid=1000,gid=1000,dmode=0770,fmode=0770 0 0` at the end of the file. I tried it, restarted the system, and I still got asked for the password...

ChatGPT's next idea was to create a script that will be automatically run at system startup and will mount the inter-OS shared folder with the appropriate permissions from the get-go. I removed the line I previously added to the fstab file and proceeded to create the new startup file. I created the file directly with nano, `sudo nano /usr/local/bin/mount_shared_folder.sh`, and added to it

```
#!/bin/bash
sudo mount -t vboxsf -o uid=1000,gid=1000,dmode=0770,fmode=0770 Win_Ubuntu_shared /mnt-win-ubu-shared
```

I made it executable with `sudo nano /etc/systemd/system/mount_shared_folder.service`, then created a service to run it at startup: `sudo nano /etc/systemd/system/mount_shared_folder.service`, with the contents

```
[Unit]
Description=Mount VirtualBox Shared Folder
After=vboxadd.service
Requires=vboxadd.service

[Service]
Type=oneshot
ExecStart=/usr/local/bin/mount_shared_folder.sh
RemainAfterExit=yes

[Install]
WantedBy=multi-user.target
```

.
Then, reloaded the Systemd and enabled the newly created service:

```
sudo systemctl daemon-reload
sudo systemctl enable mount_shared_folder.service
```

, which returned

```
Created symlink /etc/systemd/system/multi-user.target.wants/mount_shared_folder.service →
/etc/systemd/system/mount_shared_folder.service.
```

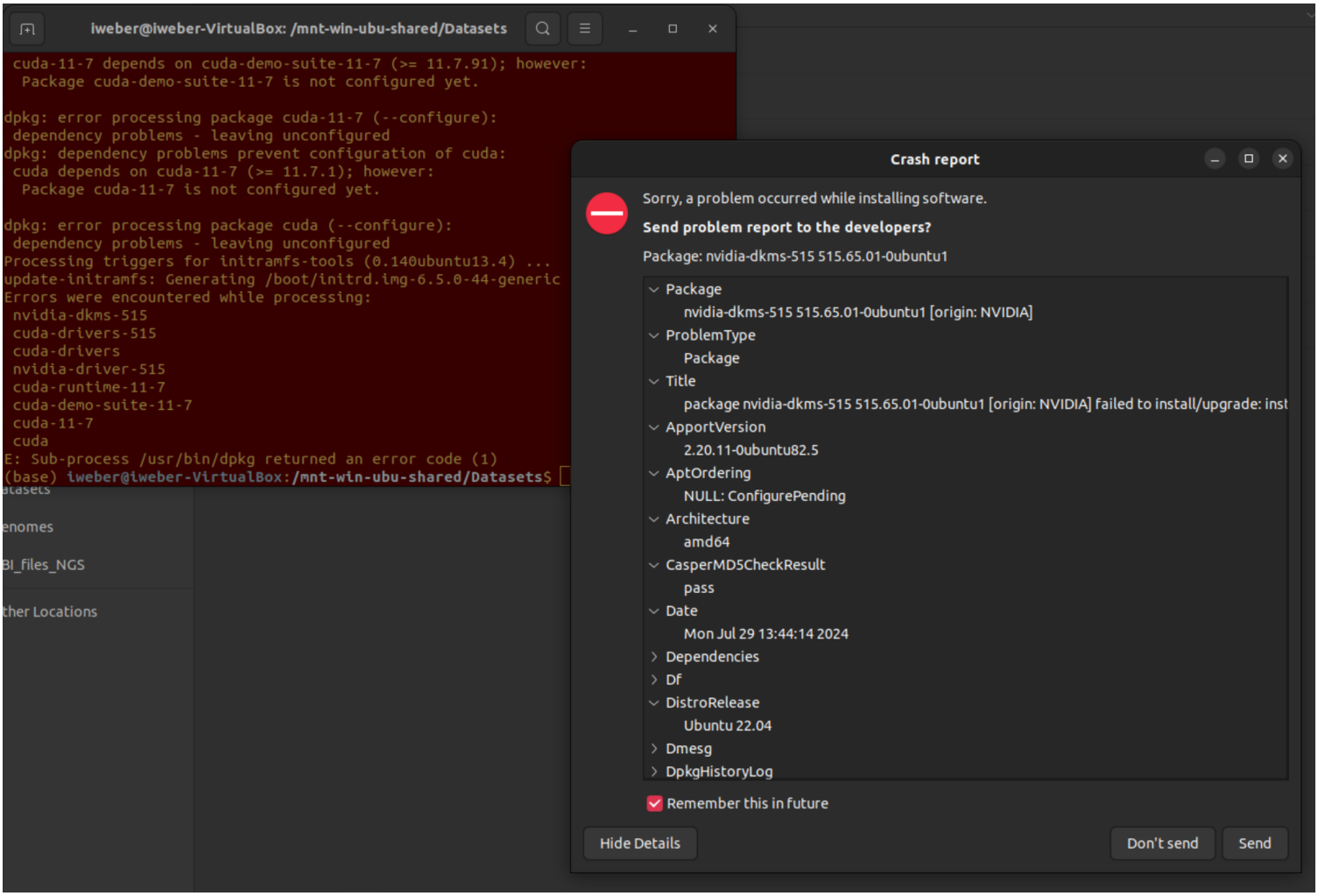
And I restarted the service with `sudo systemctl start mount_shared_folder.service`, and then the virtual machine itself. And it still doesn't work without punching in my password. So I decided that, for the time being, if I only have to input the password once, at system startup, that's fine and unlikely enough to affect the pipeline that I'm willing to drop it and move onto something more productive.

I added aliases to my .bashrc file to more easily access the folder, if nothing else. I just pasted into it:

```
alias inter-OS='cd /mnt-win-ubu-shared'
alias inter-OS_data='cd /mnt-win-ubu-shared/Datasets/'
```

Additionally, I installed conda environment name autocompletion. I first tried installing the bash autocompletion just to be sure it works properly with `sudo apt-get install bash-completion

, but that ran into an error:



The error seems to be related to the Nvidia and Cuda drivers...and I don't have the time to look further into this right now.

I simply proceeded with `conda install bash-completion` in the base environment. It seems to have completed error-free, and the autocompletion now works without errors.

Updating installation for the SRA toolkit

I had initially worked on the WSL and created the VirtualBox virtual machine later, so of course the SRA toolkit was not working any longer. I created a dedicated conda environment for it called `ncbi_SRA`, and installed the package from Bioconda using `conda install sra-tools` and updated it with `conda update sra-tools`

New environment, new life: creating dedicated bioinformatics environments using conda

Since I had all of the aforementioned issues with bcbio, I decided to abandon that approach, and, consequently, deleted the environment with conda, just to be on the safe side that I will have no issues later:

```
conda env remove --name bcbio-env
```

I then created a brand new, fresh environment called "NGS" in which to install the QC packages:
`conda env --create NGS`. Made a separate conda environment called "NGS" with `conda create --name NGS`

Using the shared folder between my Windows host and my Ubuntu virtual machine, I copied the FastQ files I had previously generated with WSL into the virtual machine.

Installing FastQC on my NGS conda environment within the Linux virtual machine

Perl and Java were already installed on my system (I suspect I installed them when installing conda and it creating its base environment), which is important because FastQC needs to run a small Perl script to find the binaries (executable files). The rest was as easy as 123: downloaded the archive, unzipped it, made the file called "fastqc" executable (chmod u+x) and ran it with ./fastqc.

Added path to the FastQC folder to my PATH environment variable so that I can now only type `fastqc` from any folder and Linux still finds the binaries of fastqc and can open it:

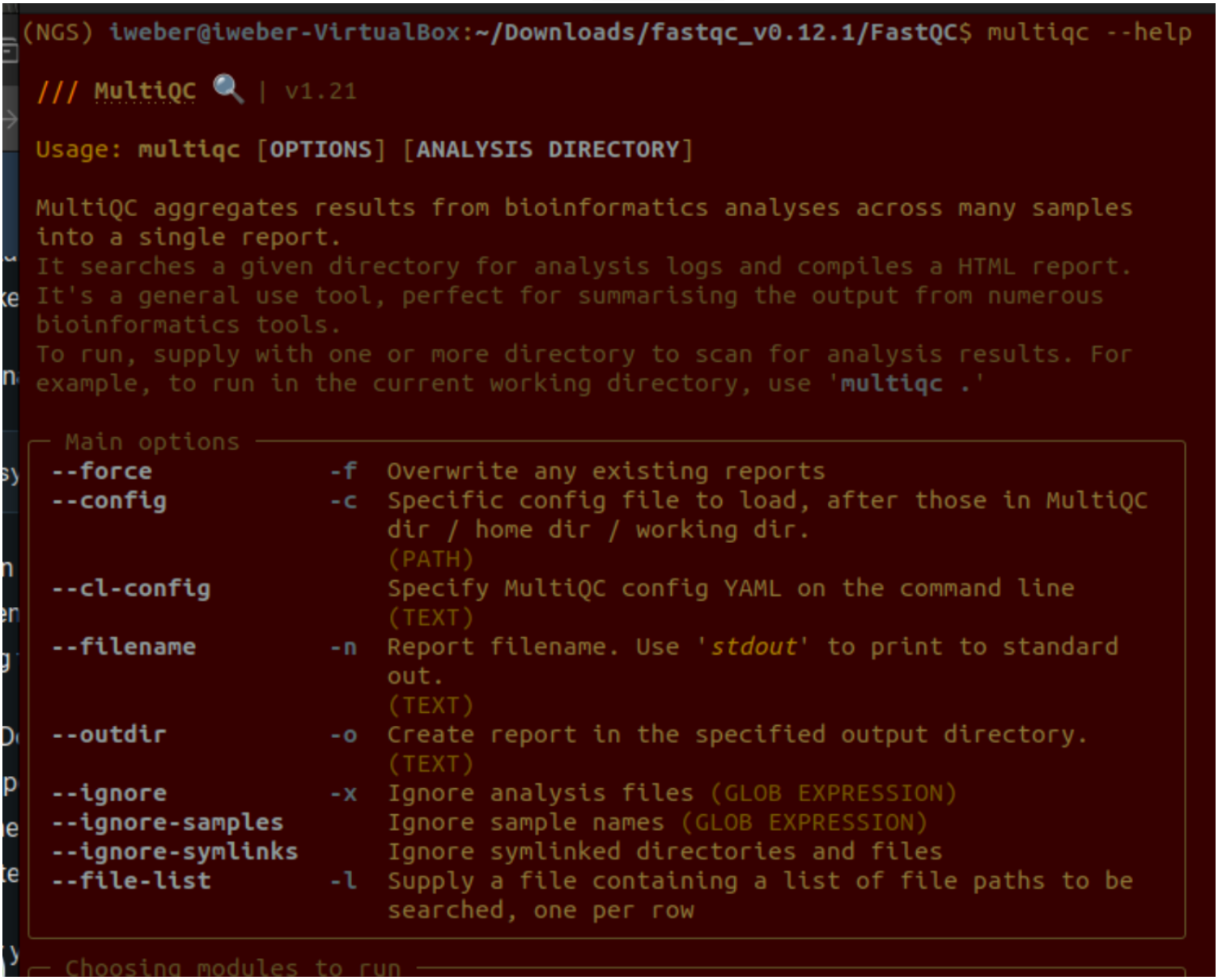
```
sudo ln -s /home/iweber/Downloads/fastqc_v0.12.1/FastQC/fastqc /usr/local/bin/fastqc
```

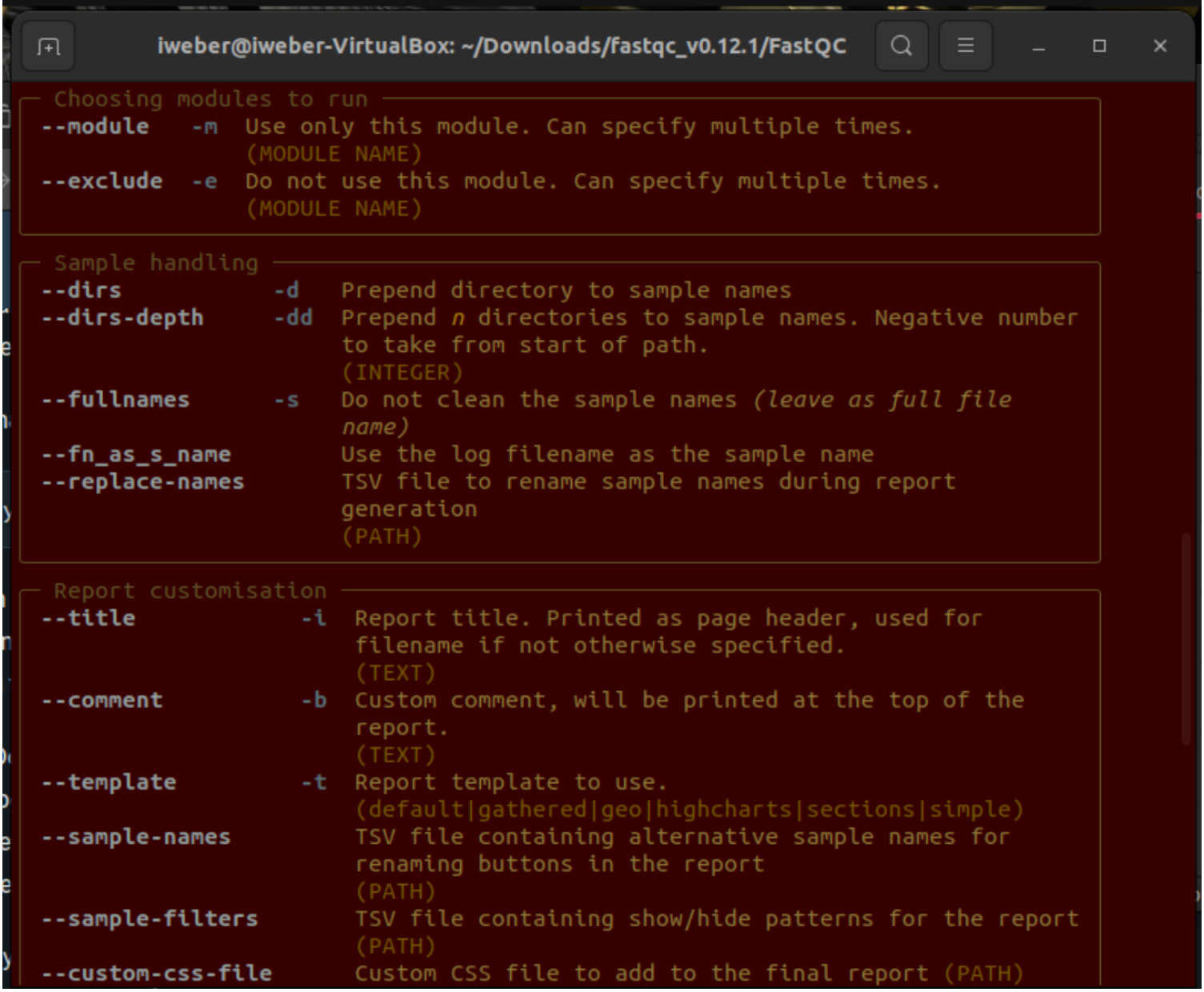
This worked! I can now call `fastqc` from the command line.

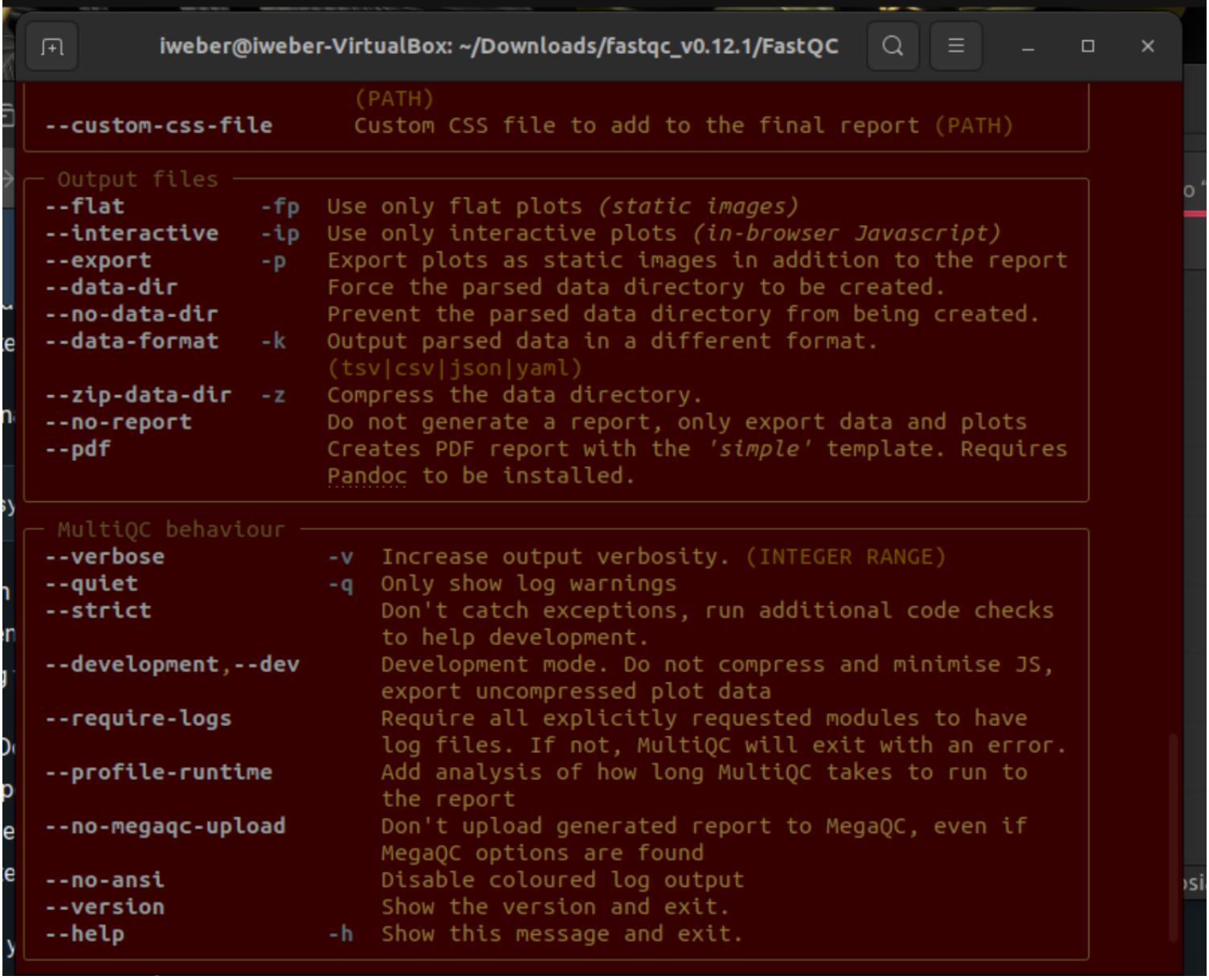
Installing MultiQC on my NGS conda environment within the Linux virtual machine

Once more, this was extra simple from the command line:
`conda install bioconda::multiqc` , after which I could call it from the command line.

After that, I wanted to see what options are available for multiqc, so I ran `multiqc --help`. I got quite a number of options:




iweber@iweber-VirtualBox: ~/Downloads/fastqc_v0.12.1/FastQC
Choosing modules to run
--module -m Use only this module. Can specify multiple times. (MODULE NAME)
--exclude -e Do not use this module. Can specify multiple times. (MODULE NAME)
Sample handling
--dirs -d Prepend directory to sample names
--dirs-depth -dd Prepend n directories to sample names. Negative number to take from start of path. (INTEGER)
--fullnames -s Do not clean the sample names (leave as full file name)
--fn_as_s_name Use the log filename as the sample name
--replace-names TSV file to rename sample names during report generation (PATH)
Report customisation
--title -i Report title. Printed as page header, used for filename if not otherwise specified. (TEXT)
--comment -b Custom comment, will be printed at the top of the report. (TEXT)
--template -t Report template to use. (default|gathered|geo|highcharts|sections|simple)
--sample-names TSV file containing alternative sample names for renaming buttons in the report (PATH)
--sample-filters TSV file containing show/hide patterns for the report (PATH)
--custom-css-file Custom CSS file to add to the final report (PATH)



Creating a GitHub repository for the project

In order to be able to more easily version my work on the project and to allow my internship supervisor to gain insight into what I was doing, I created a GitHub repo for the newly minted "official project folder" (the inter-OS shared folder I created before). Naturally, I should have started a repository much longer ago, but, as the saying goes, "The best time to plant a tree was 20 years ago. The next-best time is now". I started working on this project while my bioinformatics course was still ongoing and we hadn't talked about git yet - now's that next-best time to catch up on this.

Making the repo under Linux

I first created an empty repo on GitHub and also a fine-grained access token for myself, giving myself all of the privileges necessary for using the repo. Then, from Ubuntu, I initialized a new git repo in the inter-OS shared folder with `git init` and pointed its head to the main branch: `git symbolic-ref HEAD refs/heads/main`.

I added the online repo with
`git remote add origin https://github.com/i-weber/Internship_project_RNAseq`.

I added all of the files in the folder to the current staging area with `git add .`, and that might have been a mistake, given that the FastQ files are pretty large in their uncompressed state - the whole folder is around 170 GB *insert clenched teeth smiley*. On the side, I then read one can use the `top` command ***in a new terminal window*** to check resource usage in Linux, similar to Windows's task manager, and I saw that Git is using 70-80% of all resources of the virtual machine, so I gave it time. Next time, I will use `git add . --verbose` to get a better feel of the progress it is making through the files and folders. Also, can use `iostat -x 1` to investigate disk usage (install beforehand with `sudo apt-get install sysstat`) or `htop`, which apparently is `top` on steroids (also needs prior installation with `sudo apt-get install htop`).

So far, clocking at half an hour run time for the `add` command...let's see how long it takes in total.

At the one hour mark, I decided it was time to stop messing around, so I killed the process and proceeded to add the FastQ files and genomic files to the .gitignore. I created it with nano and added to it:

```
Adapters/
Genomes/
Nextflow/
Software/
Datasets/sra
Datasets/Pre_eclampsia_mice/Pre_eclampsia_mice_fastq/
```

to avoid some of the largest and recoverable files from the push and commit.

> **For the future:**
>
> Note to self: add them without the slashes, I think git was looking for "empty" name items in these folders and not finding them, without selecting everything *inside* of the folders.

I then tried to push the .gitignore only, and, after completing the command line sign in, immediately got an error because I had previously set GitHub to block commits that might publicize my email address. So I found out that, in order to avoid this issue, I can use a no-reply address that GitHub itself creates for all of its users as my default email. I ran `git config --global user.email "163516184+i-weber@users.noreply.github.com"` to do so, and deactivated the checkbox in my GitHub email setttings that supposedly protects one from publicizing their email in the commit metadata (I now know that this no-reply address will be used). And I could finally push the .gitignore file and see it on the website!

I then proceeded to add all other files with `git add . --verbose` at 16.48. It crashed around 17.05 because - and I should've thought of this! - I deleted a file from the folder at some point ***facepalm***. One more thing learned...and restarted the same command at 17.06. Annnd it worked!

I proceeded to commit the changes with `git commit -m "All files up to date"`, which returned

```
[main 3a13867] All files up to date
 131 files changed, 98102 insertions(+), 1 deletion(-)
```

It failed, with an error message saying: "Enumerating objects: 129, done. Counting objects: 100% (129/129), done. Delta compression using up to 6 threads Compressing objects: 100% (123/123), done. error: unable to rewind rpc post data - try increasing http.postBuffer error: unable to rewind rpc post data - try increasing http.postBuffer error: RPC failed; HTTP 400 curl 92 Recv failure: Connection reset by peer send-pack: unexpected disconnect while reading sideband packet Writing objects: 100% (127/127), 2.04 GiB | 2.85 MiB/s, done. Total 127 (delta 18), reused 1 (delta 0), pack-reused 0 (from 0) fatal: the remote end hung up unexpectedly Everything up-to-date"

I set the postBuffer to 200 MB using `git config --global http.postBuffer 209715200` , and re-started the commit command at 23:20.

Aaand...it failed again. The message read just as above (I think - I lost the clipboard because I shut the virtual machine off too soon, apparently...).

My next approach was to try the Git Large File System extension (https://git-lfs.com). I used the command
``

```
curl -s https://packagecloud.io/install/repositories/github/git-lfs/script.deb.sh sudo bash
```

to install the package, as directed by the website that git-lfs directed me towards, https://packagecloud.io/github/git-lfs/install. That didn't cut it, and neither did trying to install git by using `git lfs install`, as I suspect that will only work once lfs is actually on the system (it gives an error saying it doesn't recognize the command). So I downloaded the tar archive, changed the directory to the Downloads folder, and extracted it with tar (v = verbose, x = extract - otherwise, tar tries creating an archive, z = indicate that archive is in gzip format, f = name to use for the extracted file or folder)

```
`tar -vxzf git-lfs-linux-amd64-v3.5.1.tar.gz`
```

I then navigated to the decompressed folder, made the install.sh script file inside of the folder executable with `chmod u+x install.sh`, then navigated to my inter-OS folder and ran the install.sh with sudo permissions (it told me I could not install it where I had downloaded it, the regular Downloads folder of Ubuntu, hence my choice to change to my folder). I tried now to use `git lfs install` and got the same message as before, `Updated Git hooks. Git LFS initialized.` It is now ready to use...but, in the end, does not seem so useful for my plans. I don't have individual files at the moment that are extremely large - I simply have folders with lots of relatively small files in them.

To see which files git is currently tracking, I used `git ls-files`, `git ls-tree -d HEAD` (directories only) and `git ls-tree -r HEAD` (directories and files contained therein) to make sure I am now really only tracking the files that I want to track, and not the FastQ files or genome files.

```
(base) iweber@iweber-VirtualBox:/mnt-win-ubu-shared$ git ls-tree -r HEAD
100644 blob 0ffed7141f6001530a77a07fec3a32e8533ad99c    .gitignore
100644 blob e69de29bb2d1d6434b8b29ae775ad8c2e48c5391    Datasets/2024-03-24_runMultiQC.txt
100644 blob a179d00b7db49fc9ed8cafc762beb5278bcd2188    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761520_1_fastqc.html
100644 blob 98353bcdfccf26f2d9212af41da6792907f0586a    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761520_1_fastqc.zip
100644 blob 46da2b3455d917d526fd4e26c6e46db0560d3b6c    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761520_2.fastq
100644 blob b7b85c975c169e1b08b72267ee2dc0433592f974    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761520_2_fastqc.html
100644 blob e0cd473727f4717e05cff0558985d71401b2bd2b    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761520_2_fastqc.zip
100644 blob 2be6c9af1e2b75c3c1b075ee52bb7c0b9aaa1fc8    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761521_1_fastqc.html
100644 blob e993dc4f05fdfc74a9f6efa40a1dc03c6c932265    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761521_1_fastqc.zip
100644 blob 2188a15c0de0a04d7407b6e4e995e302a799a8a3    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761521_2_fastqc.html
100644 blob 271ff6523d376745c06032e9beca813215179daf    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761521_2_fastqc.zip
100644 blob a63f91ec8411be49e3f86430bfb1b284093845a1    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761522_1_fastqc.html
100644 blob fdb69f8f4fb2a279596fa4c43a2ce3f141b2be6a    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761522_1_fastqc.zip
100644 blob a7076f98c161aefea9c7cdc604c697cdb9565834    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761522_2_fastqc.html
100644 blob 2bfb7281fc96a306923e1acf24134cad75ffab56    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761522_2_fastqc.zip
100644 blob 011784fae72d0a5af50bf45521407989f4627314    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761523_1_fastqc.html
100644 blob 206187d6bdd0156cb13a6893fb94ed282a1aa146    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761523_1_fastqc.zip
100644 blob 54b5d26c265f73a1c271027494abb3123bef274e    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761523_2_fastqc.html
100644 blob a5be721b84be25956713c28ee026b46b4eb9b72e    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761523_2_fastqc.zip
100644 blob fa633c9c19f6816b1aa08351adfc763939cca298    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761524_1_fastqc.html
100644 blob 1937c15ff7f89a32f33b4179d1ecb448d6008017    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761524_1_fastqc.zip
100644 blob 7a54a41b7c0b3b30085dfd94018db33b8ca52d0a    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761524_2_fastqc.html
100644 blob 58766fdd2bf4e8a09cd644e16da658568f236a0a    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761524_2_fastqc.zip
100644 blob da9065a04ad531169372d79b2805acc034d4d913    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761525_1_fastqc.html
100644 blob 1534512a9f7930f49b439be52b63536212133e85    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761525_1_fastqc.zip
100644 blob c1f62da1a934b6c63d4b5c7b7d8aacd922ec05c6    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761525_2_fastqc.html
100644 blob ce9f9d352028066dbfec9f989ad501d20e65b917    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761525_2_fastqc.zip
100644 blob 2c17a0d35018c97b8d4dd53cd36cc815362b6f0b    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761526_1_fastqc.html
100644 blob 897e1c7413dfba513ea5c9b1143cef1bf822b762    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761526_1_fastqc.zip
100644 blob 0bb58f5cbe42a3e16220f565c074e8fd96250791    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761526_2_fastqc.html
100644 blob 9f1513f346cdeca487fee45f3aa5e45e3b1cf626    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761526_2_fastqc.zip
100644 blob 279aa7008335132f255e411a58fccf5a8f4931d6    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761527_1_fastqc.html
100644 blob dfe6a72a3c52e7d1ac44c2774166f7f544b3633f    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761527_1_fastqc.zip
100644 blob 3c15dcad61e6e84755b2c33067a99420c53a87c8    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761527_2_fastqc.html
100644 blob 1a748edd6b96d5517f3ef38c96dcbe547a505b8b    Datasets/Pre_eclampsia_mice/FastQC_results/SRR13761527_2_fastqc.zip
100644 blob c929271f51bf6384b23efc26982717eeb64b7bb9    Datasets/Pre_eclampsia_mice/GSE167193_All.counts.txt/GSE167193_All.counts.txt
100644 blob 8c303448bb1034cdd42bc017a25d58421f507302    Datasets/Pre_eclampsia_mice/GSE167193_series_matrix.txt/GSE167193_series_matrix.txt
100644 blob d318b6c1568a59721545fedcb3459819388c9fc8    Datasets/Pre_eclampsia_mice/MultiQC_results/2024-04-07_multiqc_report_pre-eclampsia_cortices.html
100644 blob 7158dda253c445e61518e6b6f2f052186f89497e    Datasets/Pre_eclampsia_mice/MultiQC_results/2024-05-07_MultiQC_res_pre-e_general_stats.PNG
100644 blob 707fcf79628e8c66e9d701df69f9b1e4bab944b4    Datasets/Pre_eclampsia_mice/MultiQC_results/2024-05-07_MultiQC_res_pre-e_overrepresented-seqs.PNG
100644 blob 18158e1f45bc034e20781b98b19d8367c9dbf8c8    Datasets/Pre_eclampsia_mice/MultiQC_results/2024-05-07_MultiQC_res_pre-e_per-base-N-content.PNG
100644 blob 91769a22e5571f014485d44dc3028dc8b75f24f5    Datasets/Pre_eclampsia_mice/MultiQC_results/2024-05-07_MultiQC_res_pre-e_per-base-seq-content.PNG
100644 blob 686134f691269ab067c71c5a634024e4a1ddb1b2    Datasets/Pre_eclampsia_mice/MultiQC_results/2024-05-07_MultiQC_res_pre-e_per-seq-GC-content.PNG
100644 blob 7c3b1fb55887921262912ec8a3135025b1afc243    Datasets/Pre_eclampsia_mice/MultiQC_results/2024-05-07_MultiQC_res_pre-e_seq_dupes.PNG
100644 blob e080a1007475d0aaea1818852f3d768635e03962    Datasets/Pre_eclampsia_mice/MultiQC_results/2024-05-07_MultiQC_res_pre-e_seq_quality.PNG
100644 blob b8f1c8e9c0dfbe7b25f46defd3f627f69f774864    Datasets/Pre_eclampsia_mice/MultiQC_results/2024-05-07_MultiQC_res_pre-e_success.PNG
100644 blob ce5cd668d6adb89fe4efcc7ae72527bd8535ba28    Datasets/Pre_eclampsia_mice/MultiQC_results/multiqc_data/fastqc-status-check-heatmap.csv
100644 blob 60b782719eb5453437e415540ab94d0ba22b195c    Datasets/Pre_eclampsia_mice/MultiQC_results/multiqc_data/fastqc_adapter_content_plot.csv
100644 blob 90acb1a09ac468624a064812a99955d455380ca9    Datasets/Pre_eclampsia_mice/MultiQC_results/multiqc_data/fastqc_per_base_n_content_plot.csv
```

It looks like I am indeed now avoiding the really large files. I added again everything with `git add .`, then checked the status. It said it has nothing to commit, confirmed when I still tried to commit with `git commit -m "Add remaining changes"` so I went on to push the changes with `git push origin main`. The runtime started at 13:30, this time the "Writing objects" buffer went up to 10%, as opposed to the 7% it used to go up to yesterday. But it crashed again, with the error

"Writing objects: 100% (140/140), 2.04 GiB | 1.08 MiB/s, done.
Total 140 (delta 26), reused 1 (delta 0), pack-reused 0
fatal: the remote end hung up unexpectedly
Everything up-to-date",

so I think I need to reduce the buffer size again. Did it with `git config --global http.postBuffer 157286400`, let's see if this is of any help...nope.

I went through all of the files and removed anything that is not absolutely critical. I then reset the git staging area, and also removed any of the cached versions of any files with `git rm -r --cached .` Then, I used `find . -type f` to find all files that could be added at that moment. I checked the sizes - nothing is above 100 MB at the moment, not even folders.

Tried again to add and push in small batches, e.g., in the Presentations folder, only some of the PNG images. It stopped at 96% in the writing process and would not progress. I increased the http buffer with `git config --global http.postBuffer 524288000` to 500 MB, but it then got stuck at 76% in the writing process.

This is when I gave up and decided to do a fresh start. I deleted the .git folder, everything in the online version of the repository, and all of the data inside the inter-OS shared folder, since I had it backed up in a different location anyway. I re-initialized a git repo inside of the same folder, added the remote repo I had previously created and which was basically empty at this point, and then started adding the files and folders piecemeal back into the inter-OS shared folder, adding them to the staging area, committing, and pushing after each smaller batch of files. Worked like a charm, took maybe 15 min and solved all of my problems. Presentations folder, at 33 MB, took from 21:36 to 21:37 - nothing like the full hour it took in the previous days and earlier today!

Getting the repo to work under Windows as well

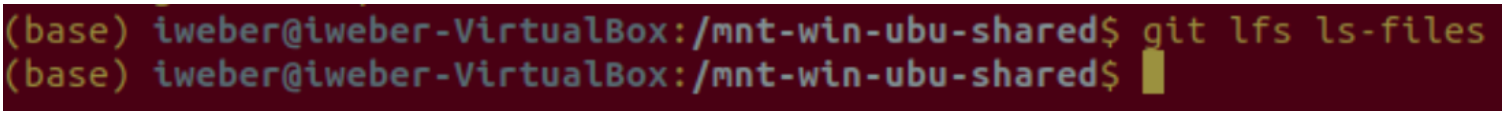
Since I later want to take the data that the analysis pipeline generates and work with it under Windows, where I have R fully set up and ready to rumble, I want to have access to the repo from this OS as well. Additionally, I read that Visual Studio Code allows for a very nice integration with GitHub, including an extension that shows when commits were made and how different branches of a project relate to one another (Git Graph). So I opened the repo in Visual Studio Code, installed the extension, and now can open it with Ctrl+Shift+P --> Git Graph: View Git Graph from the dropdown menu. Now I can very easily track the changes to the repo visually, and also have access to the repo from the Windows side - if I select the folder where the repo is in the Explorer part of VSC, I can perform all of the usual git commands from the terminal.

Trying to use Git LFS to track FastQ and other large files

As I want to track changes to my FastQ files as well once I start trimming them and processing them, but also to other large files, such as bam files, that I will generate further in the process, I set up Git LFS to track such files.

I had previously installed git LFS on my VM, and I now went to my inter-OS folder and hit `git lfs install` to initialize it (yes, the naming is somewhat confusing).

I then proceeded to track .fastq files by using `git lfs track "*.fastq"` and it returned `Tracking "*.fastq"`. I checked which files are now tracked by git LFS using `git lfs ls-files`, and it returned...nothing xD.



I checked whether the .gitattributes file was correctly created, and it seems so - it is present in my repo and contains `*.fastq filter=lfs diff=lfs merge=lfs -text`, as it should.

I pasted my FastQ files into a raw data folder (Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/), together with the FastQC and MultiQC files that resulted from their analysis. I then created a .gitignore file to, for now, ignore the FastQs but still be able to stage, commit and push the new directory structure. After doing the usual git steps, I had an updated online repo including the FastQC and MultiQC files, but not the .fastq ones.

I removed the .fastq rule from the .gitignore file. When hitting `git status` now, the only files left unchanged were indeed the newly added FastQ ones.

I ran the `git lfs track "*.fastq"` command again, and got a message saying `"*.fastq" already supported`.

I next added the first FastQ file to the staging area, SRR13761520_1.fastq. I started at 12.07, and it took around 5 min to finish. It also took a moment to commit, but, when I tried pushing it to the repo, it failed, saying

```
[f12181f07712304b1aed2f1876165f549942ad8144457878ca37fbe90356a731] Size must be less than or equal to 21474836480: [422] Size must be less than or equal to 21474836480
error: failed to push some refs to 'https://github.com/i-weber/Internship_project_RNAseq'
```

It sounds to me like git LFS is still not managing the FastQ files as it should. In the previous step, Git tried to push a file larger than 270 MB to the repo, which is clearly not what it should be doing, if git LFS were working.

> **What's happening here? there's some logical connection missing.**
>
> The solution may involve using `git lfs migrate` to force the objects that are now managed by git to be managed by git LFS instead. I used `git lfs ls-files --all` to understand if git lfs was already tracking anything

2:17 git status
2:19: It looks like the file that now appears to already be tracked by git lfs does not appear any longer under "untracked files".

Tried adding the pair of this FastQ file (the one ending in `_2`) to the staging area and committing. then running `git lfs ls-files --all` to see if git LFS takes over this file after the commit.

```
(base) iweber@iweber-VirtualBox:/mnt-win-ubu-shared$ git status
Refresh index: 100% (118/118), done.
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761520_2.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761521_1.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761521_2.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761522_1.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761522_2.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761523_1.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761523_2.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761524_1.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761524_2.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761525_1.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761525_2.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761526_1.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761526_2.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761527_1.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761527_2.fastq

nothing added to commit but untracked files present (use "git add" to track)
(base) iweber@iweber-VirtualBox:/mnt-win-ubu-shared$ git add Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761520_2.fastq
(base) iweber@iweber-VirtualBox:/mnt-win-ubu-shared$ git commit -m "Pair of first FastQ file used to test if git LFS works"
[main b5b4cc7] Pair of first FastQ file used to test if git LFS works
 1 file changed, 3 insertions(+)
 create mode 100644 Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761520_2.fastq
(base) iweber@iweber-VirtualBox:/mnt-win-ubu-shared$ git lfs ls-files --all
f12181f077 * Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761520_1.fastq
d3e4f0d7f7 * Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761520_2.fastq
(base) iweber@iweber-VirtualBox:/mnt-win-ubu-shared$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761521_1.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761521_2.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761522_1.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761522_2.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761523_1.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761523_2.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761524_1.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761524_2.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761525_1.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761525_2.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761526_1.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761526_2.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761527_1.fastq
        Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761527_2.fastq

nothing added to commit but untracked files present (use "git add" to track)
(base) iweber@iweber-VirtualBox:/mnt-win-ubu-shared$
```

And success! The files now don't show up in the regular git tracked files, but do show up as managed by git LFS!

So, what have I learned? Git LFS needs a definition of what kinds of files to look for (the pattern specified in the .gitattributes file). It then keeps a lookout for them, and, when such files are added to the staging area, it picks up on them and creates so-called pointers (pointer files) to these files and the changes in them, which are recorded in the repository instead of the files themselves. A bit like a group of elite dogs getting a sniff at a fabric scrap from persons they should track, then alerting when those persons are nearby, staying with their snouts pointed towards them.

However, they still will not be pushed online. So, even with git LFS, the problem seems to be on the side of GitHub.

```
(base) iweber@iweber-VirtualBox:/mnt-win-ubu-shared$ git lfs ls-files --all
f12181f077 * Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761520_1.fastq
d3e4f0d7f7 * Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761520_2.fastq
(base) iweber@iweber-VirtualBox:/mnt-win-ubu-shared$ git push origin main
Username for 'https://github.com': i-weber
Password for 'https://i-weber@github.com':
[f12181f07712304b1aed2f1876165f549942ad8144457878ca37fbe90356a731] Size must be less than or equal to 2147483648: [422] Size must be less than or equal to 2147483648
[d3e4f0d7f716d1839f63a3e02f3e5caa22ceaa6597304d6edb362ee9930f877a] Size must be less than or equal to 2147483648: [422] Size must be less than or equal to 2147483648
error: failed to push some refs to 'https://github.com/i-weber/Internship_project_RNAseq'
```

ChatGPT says that GitHub has a maximal limit of 2 GB even for files managed by git LFS, and this seems to fit the number the push command returns (2147483648 bytes is 2.14 GB).

So I guess I have to add these files to .gitignore and be done with them...

I added `*.fastq` to my .gitignore file, then used `git rm --cached *.fastq` to remove any previously tracked fastq files. That took a few minutes in which git-lfs was using a hefty portion of my CPU (77%). It seems to have worked:

```
(base) iweber@iweber-VirtualBox:/mnt-win-ubu-shared$ git rm --cached *.fastq
rm 'Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761520_1.fastq'
rm 'Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761520_2.fastq'
(base) iweber@iweber-VirtualBox:/mnt-win-ubu-shared$
```

I checked the git status, just in case:

```
(base) iweber@iweber-VirtualBox:/mnt-win-ubu-shared$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761520_1.fastq
        deleted:    Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/SRR13761520_2.fastq
```

It looks like the next commit should delete these files from my staging area, without touching the files per se in my computer. Tried it out, and, indeed, I could still see my files in my file system using the file explorer and `ls -lat`.

I also removed `*.fastq` files from the git LFS tracking using `git lfs untrack "*.fastq"`. Got a message in return saying `Untracking "*.fastq"`, and saw that the .gitattributes file is now also empty again.

> **Important to know for the future:**
>
> Apparently, trying to push files from Windows to GitHub that are larger than 4 GB truncates them and causes them getting corrupted!

Setting up Nextflow to run bioinformatic analysis pipelines

Installing Nextflow and nf-core

First, installed Java:

```
        sudo apt install default-jre
```

I created a dedicated environment in which i will work with Nextflow, that I called as such, using

```
conda create Nextflow
```

After activating it, I installed the actual Nextflow workflow manager with `conda install -c bioconda nextflow`, and it worked like a charm. Then, to get access to the curated Nextflow pipelines for bioinformatic analyses, I installed nf-core in the same environment with `conda install nf-core`, which also completed without any overt errors.

I also activated shell completions for nf-core by adding the recommended command to my .bashrc file:

```
eval "$(_NF_CORE_COMPLETE=bash_source nf-core)"
```

and restarted my shell.

And now I could easily list all of the curated pipelines available in nf-core:

```
(base) iweber@iweber-VirtualBox:~$ conda activate Nextflow
(Nextflow) iweber@iweber-VirtualBox:~$ nf-core list
```

```
nf-core/tools version 2.14.1 - https://nf-co.re
```

Pipeline Name	Stars	Latest Release	Released	Last Pulled	Have latest release?
funcscan	62	1.1.6	3 weeks ago	-	-
variantben…	6	dev	yesterday	-	-
demultiplex	37	1.4.1	5 months ago	-	-
ampliseq	166	2.10.0	4 weeks ago	-	-
pairgenome…	0	dev	yesterday	-	-
crisprseq	23	2.2.1	4 days ago	-	-
eager	131	2.5.2	4 weeks ago	-	-
chipseq	177	2.0.0	2 years ago	-	-
nascent	14	2.2.0	5 months ago	-	-
oncoanalys…	22	dev	3 days ago	-	-
denovotran…	0	dev	3 days ago	-	-
rangeland	3	dev	3 days ago	-	-
magmap	1	dev	3 days ago	-	-
smrnaseq	70	2.3.1	3 months ago	-	-
sarek	357	3.4.2	3 months ago	-	-
mcmicro	4	dev	1 week ago	-	-
fastquorum	13	1.0.0	2 months ago	-	-
rnaseq	824	3.14.0	7 months ago	-	-
multiplese…	11	dev	1 weeks ago	-	-
demo	1	1.0.0	1 months ago	-	-
phaseimpute	16	dev	1 weeks ago	-	-
airrflow	46	4.1.0	2 months	-	-

The pipeline I am interested in is called `rnasplice`, and we'll get back to that in a moment.

Containerization with Docker for nf-core/Nextflow?

What caught my eye when reading more about nf-core was the following:

Pipeline software

An analysis pipeline chains the execution of multiple tools together. Historically, all tools would have to be manually installed — often a source of great frustration and a key step where reproducibility between analyses is lost. nf-core pipelines utilise the built-in support for software packaging that Nextflow offers: all can work with Docker and Singularity, and most pipelines also support Conda.

To use any of the below, simply run your nf-core pipeline with `-profile <type>`. For example, `-profile docker` or `-profile conda`.

- Docker
  - Typically used locally, on single-user servers, and the cloud
  - Analysis runs in a *container*, which behaves like an isolated operating system
  - Typically requires system root access, though a *"rootless mode"* is available
- Singularity
  - Often used as an alternative to Docker on HPC systems
  - Also runs *containers*, and can optionally create these from Docker images
  - Does not need root access or any daemon processes
- Apptainer
  - Open source version of Singularity (split from Singularity in 2021)
  - **⚠ Warning**

    Currently, nf-core pipelines run with `-profile apptainer` will build using docker containers instead of using pre-built singularity containers.
    To use the singularity containers, use `-profile singularity` instead. This works because `apptainer` simply defines `singularity` as an alias to the `apptainer` command.
- Podman, Charliecloud and Shifter
  - All alternatives to Docker, often used on HPC systems
- Conda
  - Packaging system that manages environments instead of running analysis in containers
  - Poorer reproducibility than Docker / Singularity
    - There can be changes in low-level package dependencies over time
    - The software still runs in your native operating system environment and so core system functions can differ
- Mamba
  - A faster reimplementation of Conda

I started working with conda because everyone in the bioinformatic community swears by it, and so did the economics researchers that I had my very first Python course with. It has served me very well so far, but I do also know [quote Adi Dilita] that software developers prefer to work with Docker, Singularity, or Kubernetes to create containers so that their software can always be run, that is, at any time, on any machine. I read more about how Docker, Conda, and Nextflow relate to one another and found out that conda is used so widely in the bioinformatic/scientific community because it integrates seamlessly with Python and R and is a more lightweight solution for reproducibility. This is due to the fact that it manages only the dependencies specifically required by these programming languages. However, Docker containers also incorporate information about the OS and all of the apps and packages installed on it that are required to run a specific program/application. In essence, Docker containers mirror the environment that the developer of a particular software worked in in order to program that software/app etc *and* isolate this mirror image within another user's OS. This increases the degree of reproducibility dramatically BUT is also more bulky, because a Docker container then stores all of this extra information related to the OS.

Since I want to try out Docker AND because it might be the less buggy option here, I decided to install it on my Linux virtual machine.

As per the instructions on the website, I first ran

```
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

to install the required packages....and it doesn't work. What I get is:

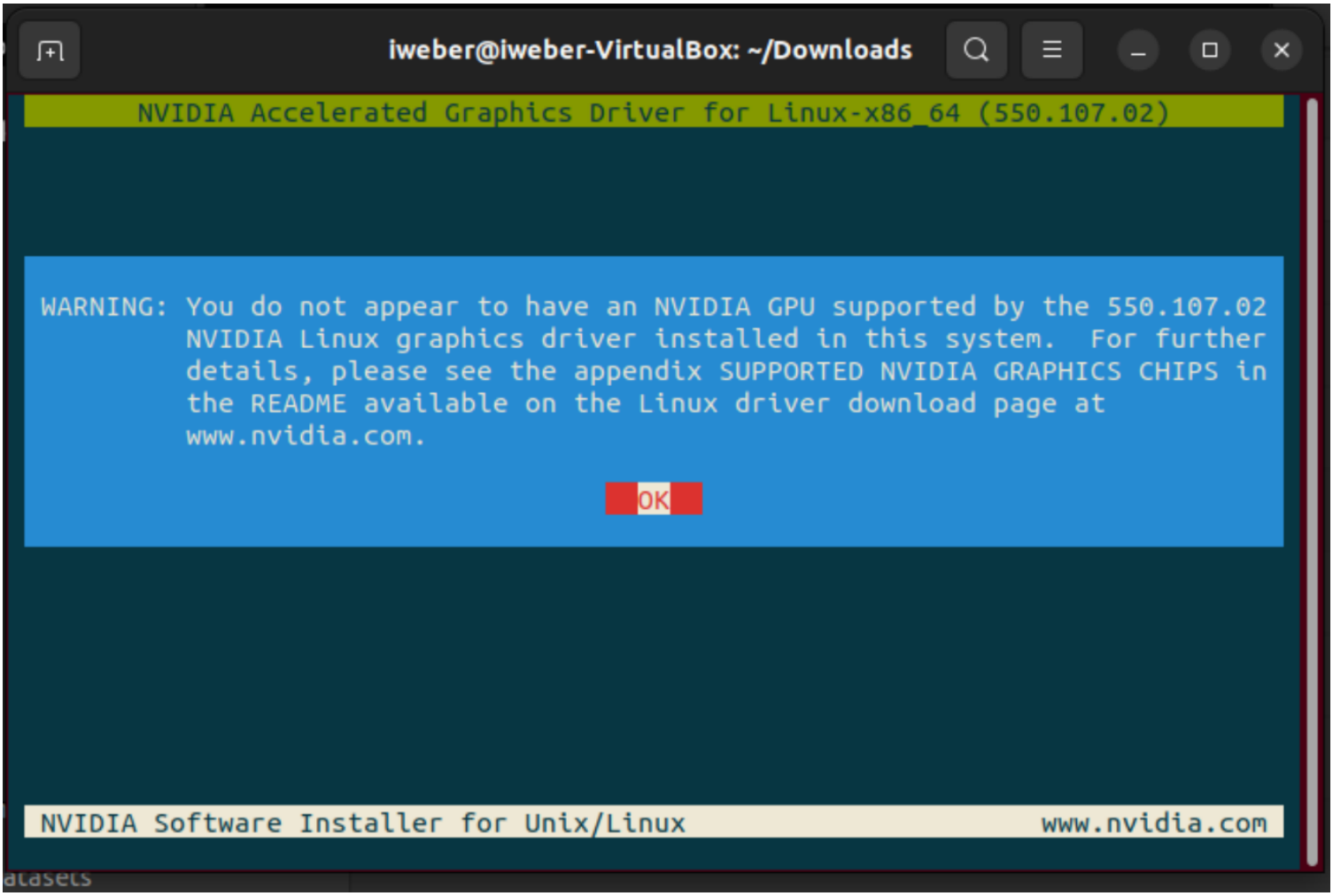
```
Get:1 file:/var/cuda-repo-ubuntu2004-11-7-local  InRelease [1.575 B]
Get:1 file:/var/cuda-repo-ubuntu2004-11-7-local  InRelease [1.575 B]
Hit:2 http://de.archive.ubuntu.com/ubuntu jammy InRelease
Hit:3 https://cloud.r-project.org/bin/linux/ubuntu jammy-cran40/ InRelease
Get:4 http://de.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Hit:5 https://packages.microsoft.com/repos/code stable InRelease
Hit:6 http://de.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:7 https://ppa.launchpadcontent.net/c2d4u.team/c2d4u4.0+/ubuntu jammy InRelease
Get:8 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Hit:9 https://packagecloud.io/github/git-lfs/ubuntu jammy InRelease
Fetched 257 kB in 2s (139 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ca-certificates is already the newest version (20230311ubuntu0.22.04.1).
ca-certificates set to manually installed.
curl is already the newest version (7.81.0-1ubuntu1.16).
The following packages were automatically installed and are no longer required:
  libwpe-1.0-1 libwpebackend-fdo-1.0-1
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 11 not upgraded.
8 not fully installed or removed.
After this operation, 0 B of additional disk space will be used.
Do you want to continue? [Y/n] y
Setting up nvidia-dkms-515 (515.65.01-0ubuntu1) ...
update-initramfs: deferring update (trigger activated)

A modprobe blacklist file has been created at /etc/modprobe.d to prevent Nouveau
from loading. This can be reverted by deleting the following file:
/etc/modprobe.d/nvidia-graphics-drivers.conf

A new initrd image has also been created. To revert, please regenerate your
initrd by running the following command after deleting the modprobe.d file:
`/usr/sbin/initramfs -u`

*******************************************************************************
*** Reboot your computer and verify that the NVIDIA graphics driver can   ***
*** be loaded.                                                             ***
*******************************************************************************

INFO:Enable nvidia
DEBUG:Parsing /usr/share/ubuntu-drivers-common/quirks/lenovo_thinkpad
DEBUG:Parsing /usr/share/ubuntu-drivers-common/quirks/put_your_quirks_here
DEBUG:Parsing /usr/share/ubuntu-drivers-common/quirks/dell_latitude
Removing old nvidia-515.65.01 DKMS files...
Deleting module nvidia-515.65.01 completely from the DKMS tree.
Loading new nvidia-515.65.01 DKMS files...
Building for 6.5.0-44-generic
Building for architecture x86_64
```

```
Building initial module for 6.5.0-44-generic
ERROR: Cannot create report: [Errno 17] File exists: '/var/crash/nvidia-dkms-515.0.crash'
Error! Bad return status for module build on kernel: 6.5.0-44-generic (x86_64)
Consult /var/lib/dkms/nvidia/515.65.01/build/make.log for more information.
dpkg: error processing package nvidia-dkms-515 (--configure):
 installed nvidia-dkms-515 package post-installation script subprocess returned error exit status 10
dpkg: dependency problems prevent configuration of cuda-drivers-515:
 cuda-drivers-515 depends on nvidia-dkms-515 (>= 515.65.01); however:
  Package nvidia-dkms-515 is not configured yet.

dpkg: error processing package cuda-drivers-515 (--configure):
 dependency problems - leaving unconfigured
dpkg: dependency problems prevent configuration of cuda-drivers:
 cuda-drivers depends on cuda-drivers-515 (= 515.65.01-1); however:
  Package cuda-drivers-515 is not configured yet.

dpkg: error processing package cuda-drivers (--configure):
 dependency problems - leaving unconfigured
dpkg: dependency problems prevent configuration of nvidia-driver-515:
 nvidia-driver-515 depends on nvidia-dkms-515 (= 515.65.01-0ubuntu1); however:
  Package nvidia-dkms-515 is not configured yet.

dpkg: error processing package nvidia-driver-515 (--configure):
 dependency problems - leaving unconfigured
dpkg: dependency problems prevent configuration of cuda-runtime-11-7:
 cuda-runtime-11-7 depends on cuda-drivers (>No apport report written because the error message indicates its a followup error
from a previous failure.
                                            No apport report written because the error message indicates its a followup
error from a previous failure.
                                             No apport report written because MaxReports is reached already
    No apport report written because MaxReports is reached already
                                                                  No apport report written because MaxReports is reached
already
                       No apport report written because MaxReports is reached already
                                                                                  No apport report written because
MaxReports is reached already
                                         = 515.65.01); however:
  Package cuda-drivers is not configured yet.

dpkg: error processing package cuda-runtime-11-7 (--configure):
 dependency problems - leaving unconfigured
dpkg: dependency problems prevent configuration of cuda-demo-suite-11-7:
 cuda-demo-suite-11-7 depends on cuda-runtime-11-7; however:
  Package cuda-runtime-11-7 is not configured yet.

dpkg: error processing package cuda-demo-suite-11-7 (--configure):
 dependency problems - leaving unconfigured
dpkg: dependency problems prevent configuration of cuda-11-7:
 cuda-11-7 depends on cuda-runtime-11-7 (>= 11.7.1); however:
  Package cuda-runtime-11-7 is not configured yet.
 cuda-11-7 depends on cuda-demo-suite-11-7 (>= 11.7.91); however:
  Package cuda-demo-suite-11-7 is not configured yet.

dpkg: error processing package cuda-11-7 (--configure):
 dependency problems - leaving unconfigured
dpkg: dependency problems prevent configuration of cuda:
 cuda depends on cuda-11-7 (>= 11.7.1); however:
  Package cuda-11-7 is not configured yet.

dpkg: error processing package cuda (--configure):
 dependency problems - leaving unconfigured
Processing triggers for initramfs-tools (0.140ubuntu13.4) ...
update-initramfs: Generating /boot/initrd.img-6.5.0-44-generic
Errors were encountered while processing:
 nvidia-dkms-515
 cuda-drivers-515
 cuda-drivers
 nvidia-driver-515
 cuda-runtime-11-7
 cuda-demo-suite-11-7
 cuda-11-7
 cuda
E: Sub-process /usr/bin/dpkg returned an error code (1)
```

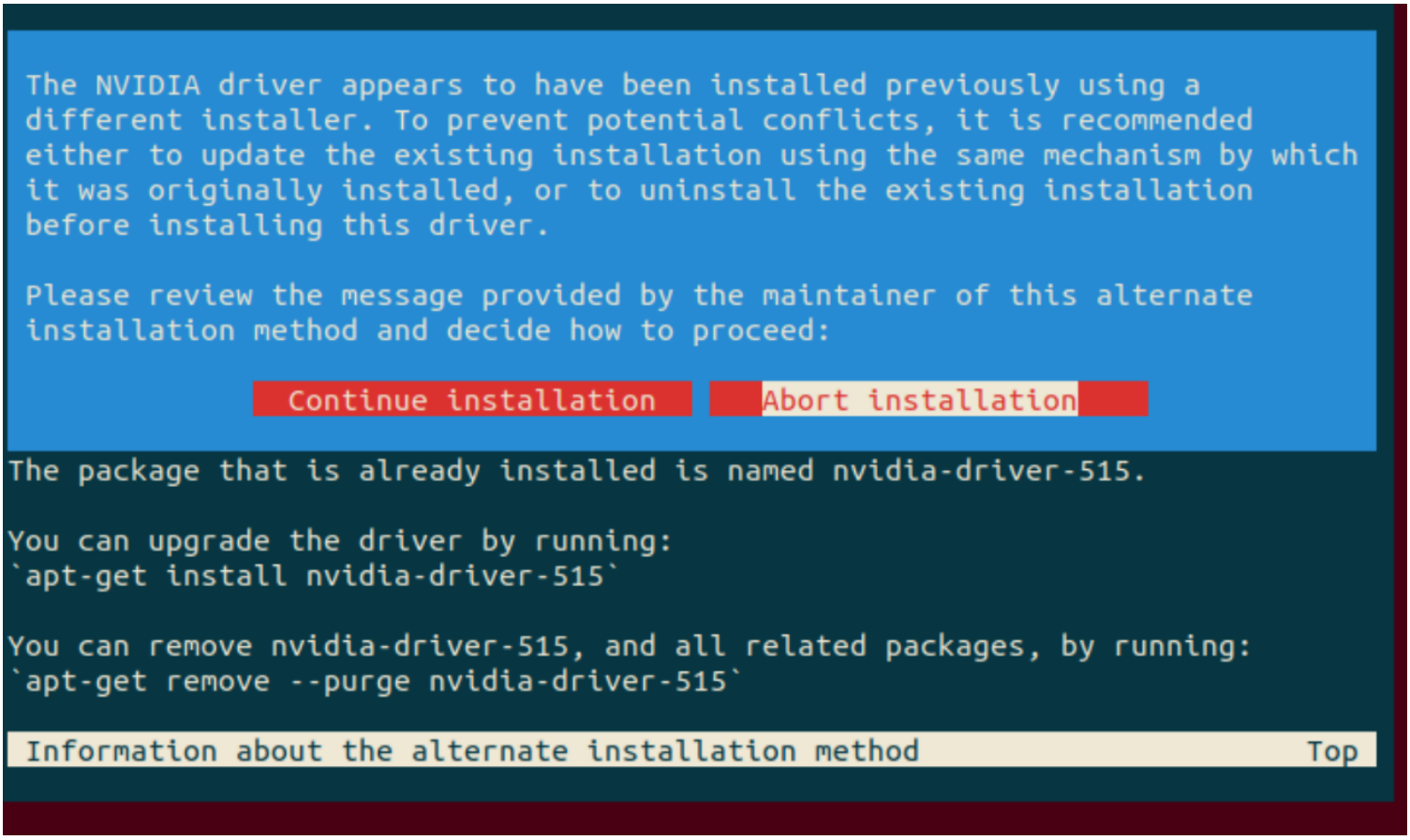
I went online to try and find what all of these issues with Nvidia and Cuda are about, and, sure enough, I found an Nvidia page about it.

I went through the preliminary checks. Yes, I have an OS version that's supported, yes, I have a GPU that supports Cuda. I also have gcc installed and `uname -r` returned "6.5.0-44-generic".

I tried finding this mlnx_ofed driver to install, but it was not available anywhere on the NVIDIA pages. I downloaded the dedicated Linux 64-bit driver for my graphics card (RTX A4000) from the NVIDIA website (it's a bash script) and ran it with sudo, but I got an error saying



Upon further reading, I saw that Ubuntu is a bit peculiar in the way in which it uses NVIDIA drivers and that, for some stability reasons, it has its own driver, based on the current driver's predecessor (number 535 instead of 550). But, after hitting OK on this error message, I got another one saying:



I hit "abort installation" and peeked at whether version 515 is indeed the one installed on my system using `ubuntu-drivers list`. Annnnd all I got was

```
(base) iweber@iweber-VirtualBox:~/Downloads$ ubuntu-drivers list
open-vm-tools-desktop
(base) iweber@iweber-VirtualBox:~/Downloads$
```

As far as I understand, the driver list is different from what a normal, non-virtual OS would have.

I tried upgrading the driver by running `sudo apt-get install nvidia-driver-515` ....annnnd got the exact same big stream of errors as above, which suggest rebooting computer and making sure NVIDIA graphics driver can be loaded. This seems to be a general issue with trying to run Cuda on a virtual machine, as, as far as my meagre understanding takes me, the virtual machine does not have direct access to the GPU through the host, especially not on VirtualBox virtual machines (apparently, VMWare is somewhat better, but that's not what I have).

I'm considering trying Singularity instead. Singularity is the containerization app often used on high-performance computing clusters (HPCs), and it makes things a little bit easier because it also does not require root access privileges.

...apparently, Singularity has changed names and is now called Apptainer. I started following the installation instructions , and already the very first step for a setuid installation, `sudo apt install -y software-properties-common` immediately threw the same sequence of errors about NVIDIA and Cuda as above. So I decided the time has come to see if I can clean that up.

Before doing so, I backed up all of my conda environments into my GitHub repository. I created a new folder "Conda_environment_yamls", and pushed that to GitHub

I also created a snapshot of my virtual machine in its current state using VirtualBox, and saved my .vbox file in a safe location. Additionally, I saved the entire virtual machine folder in another location, and exported my virtual machine as an OVF/OVA file (in VirtualBox, File --> Export Appliance). Finally, I made a full clone of my virtual machine to test changing the drivers in, so that I can always just remove it and go back to my unchanged version, should major problems arise (did that also from VirtualBox from the machine options).

It seems like I got myself into another technical rabbithole here: apparently, the root of the problems is that VirtualBox cannot take the GPU available to my Windows host and pass it through to my virtual machine. This might just be what is causing all of the issues: given the errors the Docker installation throws, it probably needs to access the GPU itself, because it helps it speed up computation quite a lot (inasmuch as my NVIDIA RTX A4000 can do that - it's a good GPU, not the top of the tops, but surely better than the virtualized version that VirtualBox creates). And because the Docker containers are so resource-intensive, this is likely why it won't work without it.

I can likely transfer my virtual machine from VirtualBox to another virtual machine software, VMware, but that will likely require some more tweaking as well. For the time being, I will try running the pipeline as is in Nextflow under Conda, and then see about anything else.

Transferring entirety of virtual machine to VMware

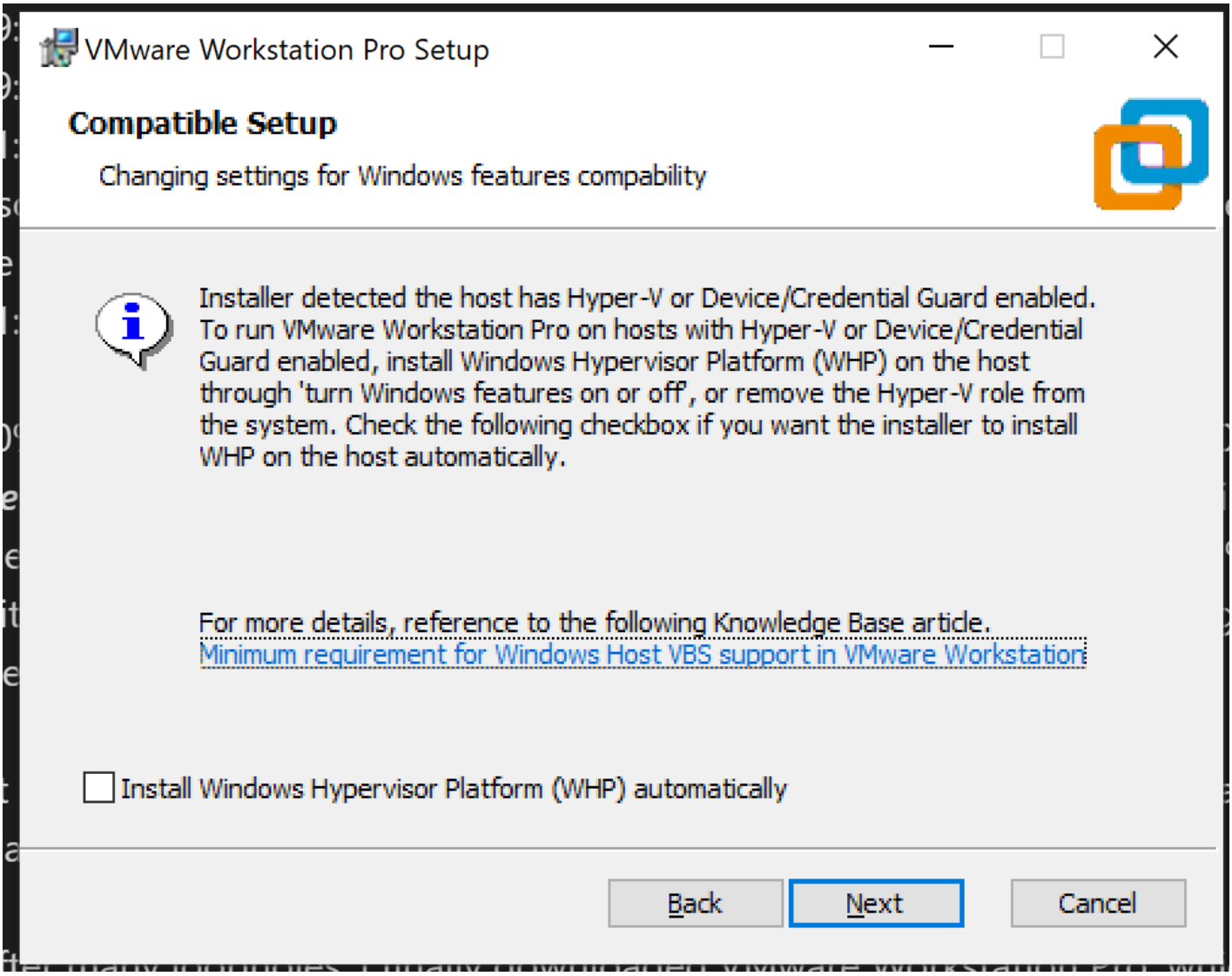
I took a snapshot of my current VM (2024-08-01) using VirtualBox, and, in the same program, hit "Export appliance" for this very same machine from the File menu. Started around 18:40, reading about the pipeline in parallel. It's mightily slow - at 19.05, so 25 min later, it was at barely 2%, meaning this will probably take until Saturday...
18:40 - start
19:05 - 2 % --> 1%/12.5 min
19:42 - 4 % --> 1%/18.5 min
21:00 - 7 % --> 3%/1h15, eq to 1%/25 min
...somehow, it seems to be slowing down quite a lot. Maybe because I opened more apps. Should be faster over night, when I'm not actively working on the laptop.
21:42 - 10% --> 7%/42 min, eq to 1%/6 min

90% more to go. At best, this should take 6x90 min = 9 h. At worst, 25x90 min = 37.5 h ***insert clenched teeth smiley here*** Of course, I closed any app that might be eating any of my laptop's memory/processing power, but the virtual machine was only using 16-30% of my CPU to begin with, so maybe that's not the issue here. Maybe, as with many other things, VirtualBox is rather inefficient in its resource utilization...

At around 2 AM, it had finally finished creating the OVF file (the file format of the so-called virtual machine appliance I just created, which is an [universal format](How To Convert Virtual Machines Between VirtualBox and VMware (howtogeek.com)) and can be shared between different VM software).

After many loopholes, I finally downloaded VMware Workstation Pro, which, to my joy, is now available for free for personal use since May of this year.

During the installation of VMware Workstation Pro, I got a message saying



I researched this further and found that, in my Windows Features, Hyper-V was actually switched off. However, since I had previously used WSL, I suspect this is not entirely true. I went on to fully switch it off in an elevated command prompt using

```
bcdedit /set hypervisorlaunchtype off
```

Seconds later, I realized it was stupid to not make a Windows restore point right before that, and I tried reverting the command using "on" instead of "off", but it just gave me an error message saying "The integer data is not valid as specified. Run "bcdedit /?" for command line assistance. The parameter is incorrect.". I asked ChatGPT and found I was supposed to run

```
bcdedit /set hypervisorlaunchtype auto
```

which, indeed, completed successfully. I swiftly created another restore point. Additionally, I updated my bootable Windows USB drive, which I had created some time back, to be able to easily boot the system in case I suddenly get a BSOD (this happened to me some years back, on the second day of owning this laptop, when I wanted to set up VirtualBox...better safe than sorry).

I also downloaded the VMware OVF Tool to make sure that the OVF I previously generated with VirtualBox is compatible with VMware Workstation Pro. I added its installation directory to my PATH variable under Windows, and could then access it from my command line in PowerShell.

I tried converting the old virtual machine OVA file to a VMware-compatible type using " ovftool ubuntu22x64.ova E:\VMware_ubuntu_VM " (into a new directory I created for the purpose), but got an error: "Opening OVA source: ubuntu22x64.ova Opening VMX target: E:\VMware_ubuntu_VM Error: OVF Package is not supported by target: - Line 25: Unsupported hardware family 'virtualbox-2.2'. Completed with errors "

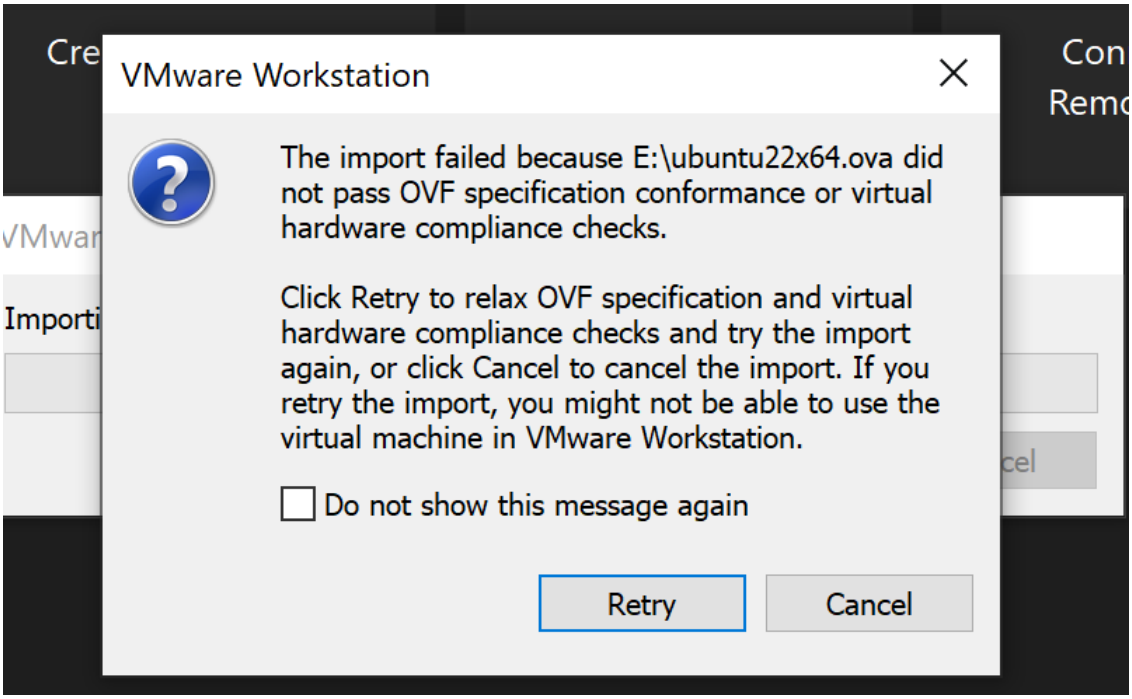
To convert this hardware family to one that's compatible with VMware Workstation Pro, ChatGPT suggested unpacking the OVA archive and directly editing the OVF file that contains the info. While 7zip was doing the unpacking, I did a [bit more reading online on the topic](How To Convert Virtual Machines Between VirtualBox and VMware (howtogeek.com)](https://www.howtogeek.com/125640/how-to-convert-virtual-machines-between-virtualbox-and-vmware/)). Because this website says it should be possible to open the VM as is in VMware, once it is installed, so I decided to postpone modifying anything in the OVF until I shut off Hyper-V for good with bcdedit, restart my PC, and attempt to install VMware.

7zip kept showing "Unexpected end of data" all the way through.

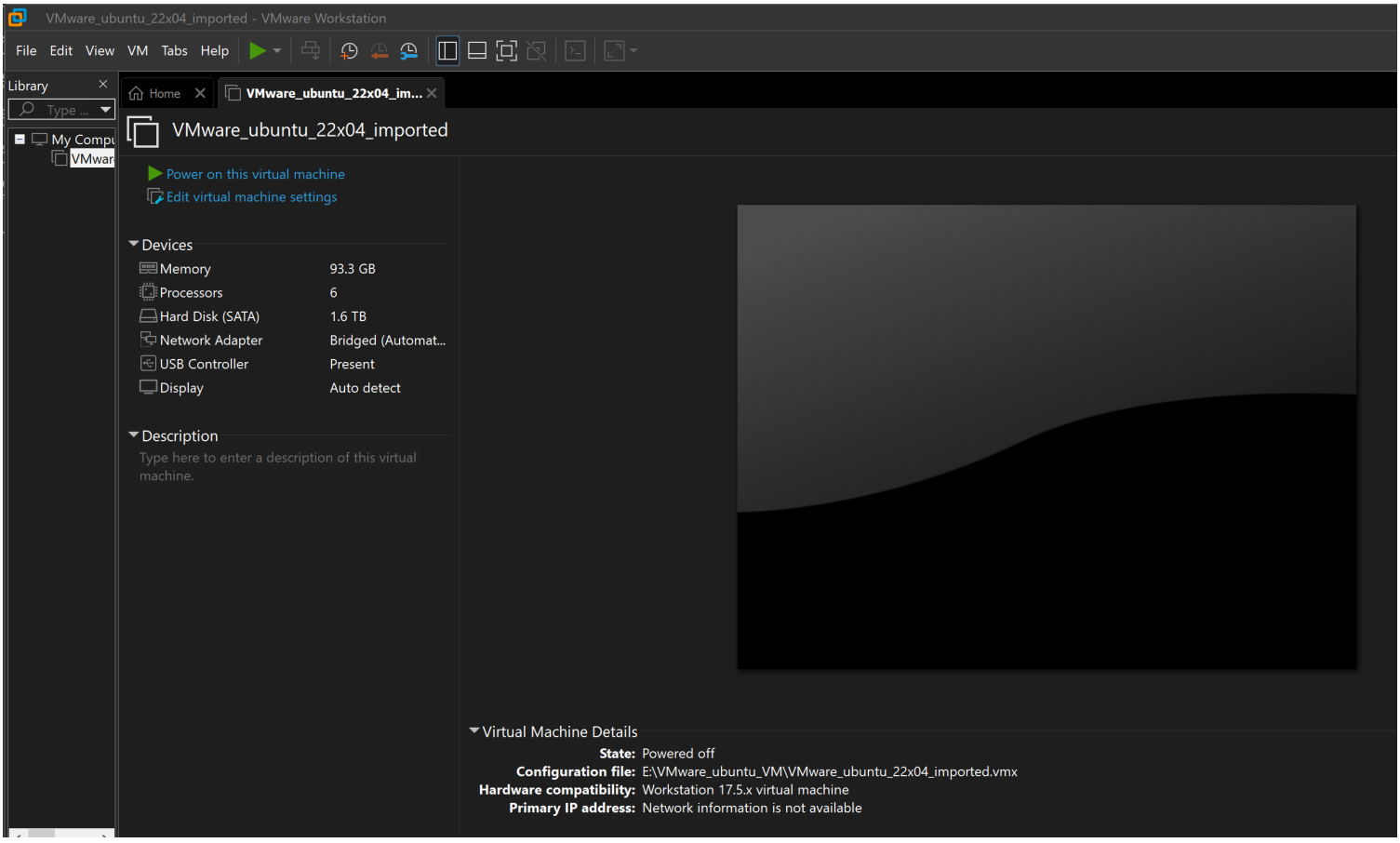
And then I restarted my PC. And got no BSOD, yay! *insert party smiley*

I re-ran the installer for VMware Workstation Pro, and didn't get that error relating to Hyper-V, which means it's now finally switched off and I don't need to worry any longer. I chose to skip adding the optional "Enhanced Keyboard Driver", as I did not find any overtly important reason to do so. The installation completed successfully and I could now finally open the software ***!yay!***

I followed the instructions from How To Geek to import the VM from VirtualBox into VMware. I got the error the website predicted:



I hit "Retry" as instructed, which, as it says, makes the importing criteria less strict, and the actual import began. Annnnnnnd it worked!



I started the machine and *even conda was still working as expected in the terminal*. What was left to do was to install the VMware Guest Additions from the Linux ISO. For this, with the VM powered off, I went to its settings and added a CD/DVD, which I pointed to the VMware Linux ISO that should contain the additional tools for this Ubuntu virtual machine. The behavior was rather strange after that:

```
(base) iweber@iweber-VirtualBox:~$ vmware-toolbox-cmd -v
12.3.5.46049 (build-22544099)
(base) iweber@iweber-VirtualBox:~$ sudo vmware-uninstall-tools.pl
sudo: vmware-uninstall-tools.pl: command not found
(base) iweber@iweber-VirtualBox:~$ sudo mount /dev/cdrom /mnt
mount: /mnt: WARNING: source write-protected, mounted read-only.
```

Maybe I have a broken installation of the Toolbox, because the option to reinstall it is greyed out in the VM menu, and, while some part of it is installed, returning me some version number, somehow, the uninstallation command isn't. So I tried to remove the Toolbox in its entirety from the system, running these commands one by one:

```
sudo rm -rf /usr/lib/vmware-tools # completed with no errors
sudo rm -rf /etc/vmware-tools # completed with no errors
sudo rm -rf /usr/lib/vmware-tools/modules # same
sudo rm -rf /usr/bin/vmware* # same
```

I then re-mounted the ISO image with

```
sudo mount /dev/cdrom /mnt
```

, which told me that it's already mounted.

Then, I used tar to extract the archive from the ISO:

```
tar -zxvf /mnt/VMwareTools-*.tar.gz -C /tmp
```

, which generated an onslaught of files, but completed fine and gave me a new command prompt.

I changed to the extracted archive directory, made the installation script executable, and ran it. It didn't work, and I realized, based on the messages, that the problem was that the toolbox of VirtualBox was still present on the system:

```
(base) iweber@iweber-VirtualBox:/tmp/vmware-tools-distrib$ sudo ./vmware-install.pl
The installer has detected an existing installation of open-vm-tools packages
on this system and will not attempt to remove and replace these user-space
applications. It is recommended to use the open-vm-tools packages provided by
the operating system. If you do not want to use the existing installation of
open-vm-tools packages and use VMware Tools, you must uninstall the
open-vm-tools packages and re-run this installer.
The packages that need to be removed are:
open-vm-tools
Packages must be removed with the --purge option.
The installer will next check if there are any missing kernel drivers. Type yes
if you want to do this, otherwise type no [yes] yes

INPUT: [yes]

Creating a new VMware Tools installer database using the tar4 format.

Installing VMware Tools.

In which directory do you want to install the binary files?
[/usr/bin] /usr/bin

INPUT: [/usr/bin]
```

!!! I started this kernel driver installer, but I soon realized this was nonsense, and aborted it. I re-ran the rm operations above to purge any existing installed pieces of VMware Toolbox. Then, I proceeded to remove the open-vm-tools installation:

```
sudo apt-get remove --purge open-vm-tools
sudo apt-get autoremove
```

Or...at least I thought I would be doing so. Instead, I got that very same nice error message regarding Nvidia and Cuda that I struggled with on VirtualBox. I tried:

```
sudo dpkg --purge open-vm-tools
```

and at least this one seems to have worked (I got an active command prompt after this):

```
(base) iweber@iweber-VirtualBox:/$ sudo dpkg --purge open-vm-tools
(Reading database ... 265933 files and directories currently installed.)
Purging configuration files for open-vm-tools (2:12.3.5-3~ubuntu0.22.04.1) ...
```

I ran a few more commands ChatGPT suggested to completely purge any pieces left from either Toolbox or open-vm-tools:

```
(base) iweber@iweber-VirtualBox:/$ sudo dpkg --purge open-vm-tools
(Reading database ... 265933 files and directories currently installed.)
Purging configuration files for open-vm-tools (2:12.3.5-3~ubuntu0.22.04.1) ...
(base) iweber@iweber-VirtualBox:/$ sudo rm -rf /etc/vmware-tools
(base) iweber@iweber-VirtualBox:/$ sudo rm -rf /usr/lib/open-vm-tools
(base) iweber@iweber-VirtualBox:/$ sudo rm -rf /usr/share/open-vm-tools
(base) iweber@iweber-VirtualBox:/$ sudo rm -rf /usr/bin/vmware*
(base) iweber@iweber-VirtualBox:/$ sudo systemctl disable open-vm-tools
Failed to disable unit: Unit file open-vm-tools.service does not exist.
(base) iweber@iweber-VirtualBox:/$ sudo systemctl stop open-vm-tools
Failed to stop open-vm-tools.service: Unit open-vm-tools.service not loaded.
(base) iweber@iweber-VirtualBox:/$ sudo rm /etc/systemd/system/multi-user.target.wants/open-vm-tools.service
rm: cannot remove '/etc/systemd/system/multi-user.target.wants/open-vm-tools.service': No such file or directory
(base) iweber@iweber-VirtualBox:/$
```

It seems no piece were left anyway - the system said it failed to find the things that I was trying to delete. Also, my shared clipboard between my Windows host and my VM stopped working, as was to be expected.

I attempted to reinstall VMware tools, when I got an interesting message:

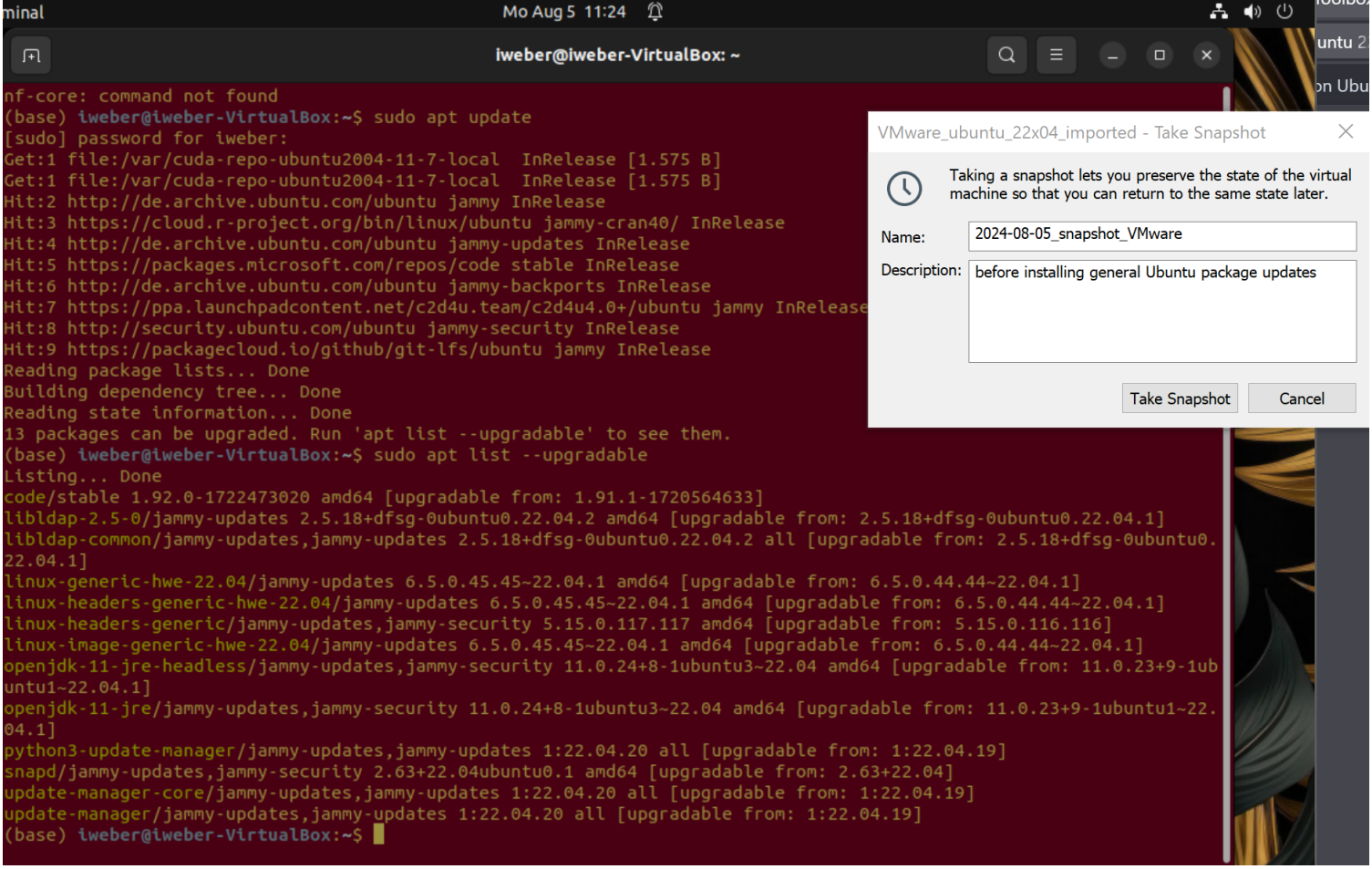
```
(base) iweber@iweber-VirtualBox:/$ sudo mount /dev/cdrom /mnt
mount: /mnt: /dev/sr0 already mounted on /media/iweber/VMware Tools.
(base) iweber@iweber-VirtualBox:/$ cd /tmp/vmware-tools-distrib/
(base) iweber@iweber-VirtualBox:/tmp/vmware-tools-distrib$ ls -lat
total 408
drwxrwxrwt 20 root   root     4096 Aug  2 20:26 ..
drwxr-xr-x  9 iweber iweber   4096 Jul 18  2020 .
-rw-r--r--  1 iweber iweber 147002 Jul 18  2020 FILES
drwxr-xr-x  2 iweber iweber   4096 Jul 18  2020 bin
drwxr-xr-x  2 iweber iweber   4096 Jul 18  2020 doc
drwxr-xr-x  5 iweber iweber   4096 Jul 18  2020 etc
-rw-r--r--  1 iweber iweber   2538 Jul 18  2020 INSTALL
drwxr-xr-x  2 iweber iweber   4096 Jul 18  2020 installer
-rwxr-xr-x  1 iweber iweber 227024 Jul 18  2020 vmware-install.pl
drwxr-xr-x  5 iweber iweber   4096 Jul 18  2020 caf
drwxr-xr-x 14 iweber iweber   4096 Jul 18  2020 lib
drwxr-xr-x  3 iweber iweber   4096 Jul 18  2020 vgauth
(base) iweber@iweber-VirtualBox:/tmp/vmware-tools-distrib$ sudo ./vmware-install.pl
open-vm-tools packages are available from the OS vendor and VMware recommends
using open-vm-tools packages. See http://kb.vmware.com/kb/2073803 for more
information.
Do you still want to proceed with this installation? [no]
```

I am postponing decisionmaking about this to tomorrow, I'll have to do a bit more research to confirm. This is the website it is sending me to: https://knowledge.broadcom.com/external/article?legacyId=2073803 and it is last updated in February of this year, so seems fairly recent...

After some further research, I see that VMware indeed recommends open vm tools as the open source implementation of... VMware tools. They also recommend installing it as per the instructions of the OS producer.

After reading this , I chose to go with the desktop version of open-vm-tools, as that offers extended capability for GUI-based systems, whereas, apparently, the simple open-vm-tools package is meant for bare-bones Linux servers.

I first checked what packages need updating before that with `sudo apt update` and listed the packages with `sudo apt list --upgradable`. I got this list, and decided to take a snapshot of the system before the update, just in case something destabilizes it:

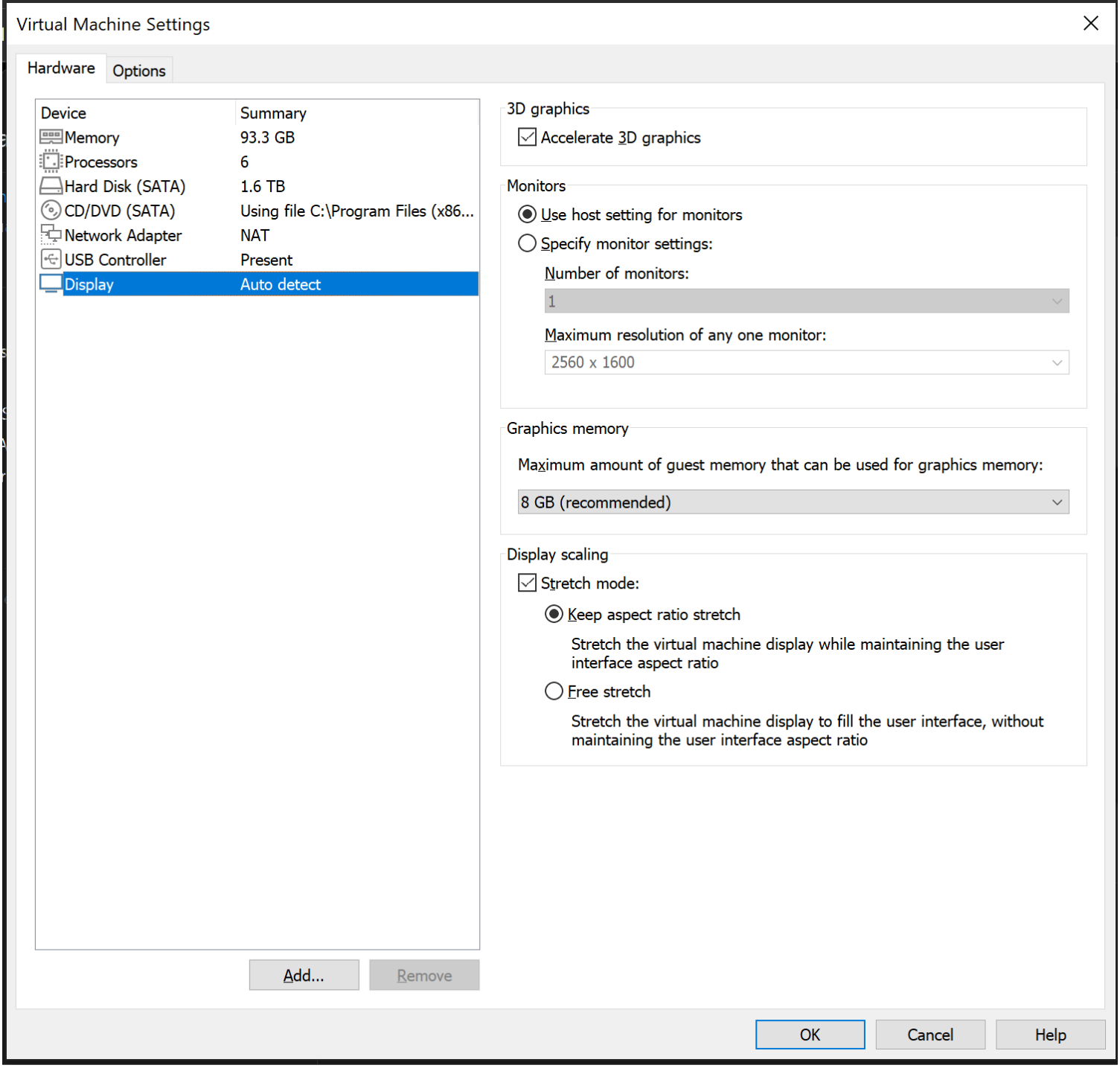


I hit `sudo apt upgrade` to upgrade all of the packages listed above, and ran into the same issue with the NVIDIA and Cuda drivers not working. And when I tried the actual command to install open-vm-tools,

```
sudo apt install open-vm-tools-desktop
```

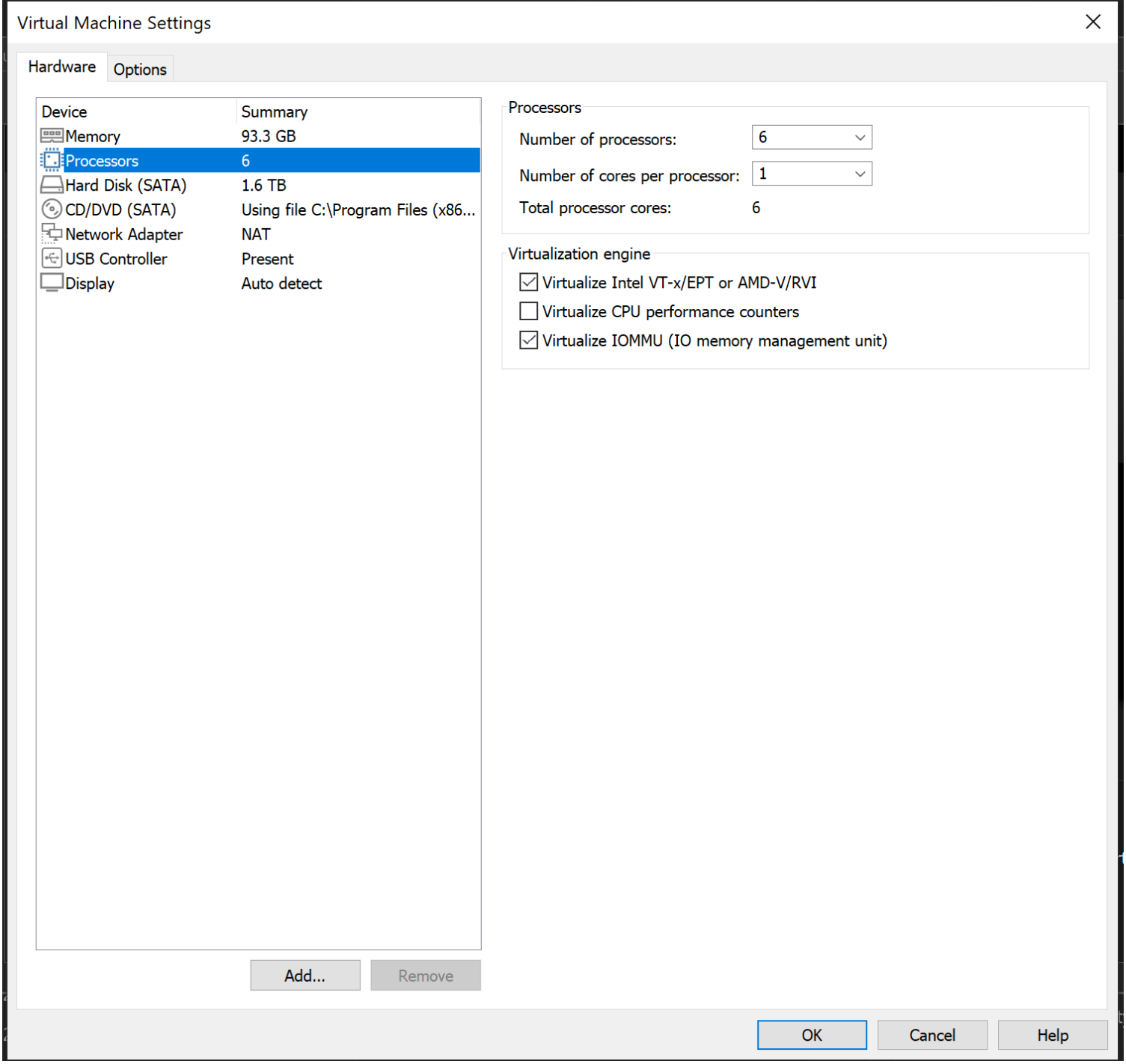
it happened again...

I had a look in the VMware graphics settings and found that I could change some options regarding 3d acceleration, which I know Nvidia is in charge of, so I switched that on. This is how the updated settings look like:



After this, I tried switching the machine back on to see if this maybe allows the use of Nvidia and Cuda and lets me install open-vm-ware....nope. Process failed with same problem.

Then, I also saw I have direct virtualization options under "Processors" in the virtual machine's settings. I changed those as follows:



Now, when peeking at the Windows task manager, I can actually see Vmware using the GPU!

But I still can't install anything, not even Ubuntu's very own tool , without the same error about nvidia and cuda popping up again.

I ran

```
sudo apt-get remove --purge '^nvidia-.*'
sudo apt-get remove --purge '^cuda-.*'
sudo apt-get remove --purge '^libcuda-.*'
sudo apt-get remove --purge '^libnvidia-.*'
sudo apt-get autoremove --purge
sudo apt-get clean
```

to clear any installation I might have left on the system for any nvidia and cuda-related packages, and start afresh. Confirmed no more packages left with `lsmod | grep nvidia`, which didn't return anything, and then added the graphics drivers repository to my system with

```
sudo add-apt-repository ppa:graphics-drivers/ppa
sudo apt-get update
```

I then went to Ubuntu's official Nvidia driver page to see how they recommend installing these drivers from the command line interface (CLI)...no help there. All i got when checking for the recommended drivers for my machine with `sudo ubuntu-drivers install` is...drumroll...open-vm-tools *laugh-cry*

I suspect this won't do much, but, in sheer despair, I tried forcing the installation of an Nvidia driver with `sudo apt-get install nvidia-driver-535` and rebooted with `sudo reboot`. But running `nvidia-smi` to check the driver gave the same message as before the install, "NVIDIA-SMI has failed

because it couldn't communicate with the NVIDIA driver. Make sure that the latest NVIDIA driver is installed and running."

I shut off the virtual machine and went scouting in my BIOS options to see if anything else relating to Intel VT needs switching on, but it does not look like it - "VT-d" is activated in my BIOS setup.

Again out of lack of better options, I tried installing the nvidia-cuda toolkit directly using `sudo apt install nvidia-cuda-toolkit`. Oddly enough, it downloaded all of the packages and completed successfully. I rebooted the machine and typed `nvidia-smi` to check if the GPU is recognized, but got the same error as above, saying it has failed because it couldn't communicate with the NVIDIA driver.

I also tried installing the CUDA toolkit from https://developer.nvidia.com/cuda-downloads?target_os=Linux&target_arch=x86_64&Distribution=Ubuntu&target_version=22.04&target_type=deb_local , but I do suspect this won't do much if Nvidia can't communicate from my virtual machine with the Nvidia driver.

For the umpteenth time, I purged everything Nvidia related, and installed the Nvidia driver version 550, apparently the latest recommended by Nvidia for Linux 64-bit systems like my VM (I currently have version 535 installed) - `sudo apt-get install nvidia-driver-550`. I also re-installed the CUDA toolkit according to the link above.

And, again, out of sheer desperation, I tried installing Docker as I had previously done on my old VirtualBox virtual machine with

```
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

and...it gave no errors?!??

I then ran, as indicated on the Docker website,

```
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

, which also completed without errors, and, finally:

```
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

It...completed?!?? Miracles????

IT WORKED. It actually worked. The test command completed successfully:

```
(base) iweber@iweber-VirtualBox:~$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:1408fec50309afee38f3535383f5b09419e6dc0925bc69891e79d84cc4cdcec6
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

So...now I have Docker on the system. Will perhaps also the pigz installation work now?

It did. Or a prior installation is now working (it looks rather like it tried to update it but didn't find any more recent version):

```
(base) iweber@iweber-VirtualBox:~$ sudo apt install pigz
[sudo] password for iweber:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
pigz is already the newest version (2.6-1).
pigz set to manually installed.
The following packages were automatically installed and are no longer required:
  libatomic1:i386 libdrm-amdgpu1:i386 libdrm-intel1:i386 libdrm-nouveau2:i386
  libdrm-radeon1:i386 libdrm2:i386 libedit2:i386 libelf1:i386 libexpat1:i386 libgl1:i386
  libgl1-mesa-dri:i386 libglapi-mesa:i386 libglvnd0:i386 libglx-mesa0:i386 libglx0:i386
  libicu70:i386 libllvm15:i386 libpciaccess0:i386 libsensors5:i386 libstdc++6:i386
  libx11-xcb1:i386 libxcb-dri2-0:i386 libxcb-dri3-0:i386 libxcb-glx0:i386
  libxcb-present0:i386 libxcb-randr0:i386 libxcb-shm0:i386 libxcb-sync1:i386
  libxcb-xfixes0:i386 libxfixes3:i386 libxml2:i386 libxshmfence1:i386 libxxf86vm1:i386
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
```

I ***immediately*** started creating another snapshot of the machine, just in case anything else goes awry.

Ensuring Docker can use GPU

ChatGPT had me test whether Docker can actually use my GPU, in case that's needed (and it likely will be, with the RNA-Seq pipelines) using `sudo docker run --rm --gpus all nvidia/cuda:11.5-base nvidia-smi`, which didn't work ("Unable to find image 'nvidia/cuda:11.5-base' locally docker: Error response from daemon: manifest for nvidia/cuda:11.5-base not found: manifest unknown: manifest unknown. See 'docker run --help'.") - not surprising, considering that `nvidia-smi` still shows the same error message saying "NVIDIA-SMI has failed because it couldn't communicate with the NVIDIA driver. Make sure that the latest NVIDIA driver is installed and running."

So it suggested installing the nvidia-docker2 package

```
distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key add -
curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | sudo tee /etc/apt/sources.list.d/nvidia-docker.list
sudo apt-get update

sudo apt-get install -y nvidia-docker2

sudo systemctl restart docker
```

and then pulling a Docker image that's easily retrievable to test if the GPU is working:

```
sudo docker pull nvidia/cuda:11.0-base
sudo docker run --rm --gpus all nvidia/cuda:11.0-base nvidia-smi
```

Which...did not work. ChatGPT said the exact name of the docker container (its tag) might be at fault, and I then went on the official GitLab repo of Nvidia with Docker containers to test different Cuda versions Here, I noticed that the Cuda toolkit version that I have (`nvcc -V` -> "Cuda compilation tools, release 11.5, V11.5.119") is not listed under "Ubuntu 22.04" but under earlier versions. I did install the toolkit as recommended on the Cuda toolkit website but, somehow, still installed version 11.5 instead of 12.6, which seems to be current recommended one.

How did that happen? I did follow the instructions on the Cuda toolkit website to a T - I cycled back to previous commands, and I for sure ran the commands for version 12.6, the latest recommended one. I now ran, out of curiosity, "sudo apt-get -y install cuda-toolkit-12-6", and what I got back was

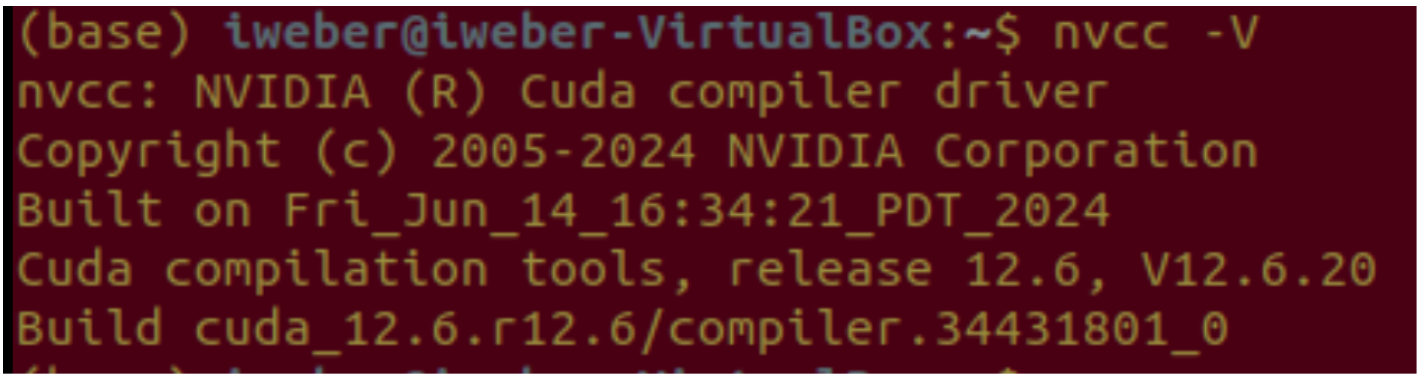
```
[sudo] password for iweber: Reading package lists... Done Building dependency tree... Done Reading state information... Done

***cuda-toolkit-12-6 is already the newest version (12.6.0-1)***.

The following packages were automatically installed and are no longer required: libatomic1:i386 libdrm-amdgpu1:i386 libdrm-intel1:i386 libdrm-nouveau2:i386 libdrm-radeon1:i386 libdrm2:i386 libedit2:i386 libelf1:i386 libexpat1:i386 libffi8:i386 libgl1:i386 libgl1-mesa-dri:i386 libglapi-mesa:i386 libglvnd0:i386 libglx-mesa0:i386 libglx0:i386 libicu70:i386 libllvm15:i386 libnvidia-egl-wayland1 libnvidia-egl-wayland1:i386 libpciaccess0:i386 libsensors5:i386 libstdc++6:i386 libwayland-client0:i386 libwayland-server0:i386 libx11-xcb1:i386 libxcb-dri2-0:i386 libxcb-dri3-0:i386 libxcb-glx0:i386 libxcb-present0:i386 libxcb-randr0:i386 libxcb-shm0:i386 libxcb-sync1:i386 libxcb-xfixes0:i386 libxfixes3:i386 libxml2:i386 libxshmfence1:i386 libxxf86vm1:i386 nvidia-firmware-550-550.107.02 Use 'sudo apt autoremove' to remove them. 0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
```

Still, when I do the `nvcc -V` check, what I get is that I have `Cuda compilation tools, release 11.5, V11.5.119`. I wondered if it is possible that I have two versions on my system, both 12.6 and 11.5, and, if yes, how I would check that. ChatGPT had me run `ls /usr/local/` to see what Cuda folders and present there, and, lo and behold, it was three: cuda, cuda-12, and cuda-12.6. So it is indeed possible that they're all different versions. However, my system apparently can't access either of them directly, since they seem to not be in the PATH variable - running `echo $PATH | grep -o "/usr/local/cuda[^:]*/bin"` to find anything cuda-related in the PATH didn't return anything.

To add the latest version to the PATH variable, I edited my .bashrc file and added, through the console, the path to the folder of the latest Cuda version:

```
sudo echo 'export PATH=/usr/local/cuda-12.6/bin${PATH:+:${PATH}}' >> ~/.bashrc
sudo echo 'export LD_LIBRARY_PATH=/usr/local/cuda-12.6/lib64${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}' >> ~/.bashrc
```

, then re-loaded my .bashrc with `source ~/.bashrc`. Now, when running `echo $PATH | grep -o "/usr/local/cuda[^:]*/bin"`, I got back "/usr/local/cuda-12.6/bin" -> success in adding it to the PATH. When I now checked the version with `nvcc -V`, I finally saw



I now tried pulling an existing Cuda Docker container from their GitLab repository using

```
sudo docker pull nvidia/cuda:12.5.1-base-ubuntu22.04
```

and saw that it worked:

```
(base) iweber@iweber-VirtualBox:~$ sudo docker pull nvidia/cuda:12.5.1-base-ubuntu22.04
12.5.1-base-ubuntu22.04: Pulling from nvidia/cuda
3713021b0277: Pull complete
1a4c6357a507: Pull complete
c13a925c73f2: Pull complete
c6961d6f1272: Pull complete
85bb4fbc01b0: Pull complete
Digest: sha256:47181750209e68cab2e36c3e0e8ab0ab8d76bdabbb418e56e4b0ac7b08b97d34
Status: Downloaded newer image for nvidia/cuda:12.5.1-base-ubuntu22.04
docker.io/nvidia/cuda:12.5.1-base-ubuntu22.04
```

(I didn't see a version for Cuda 12.6, so I chose a container running the latest available version, 12.5)

I next tried running the container to see if GPU usage actually works using

```
sudo docker run --rm --gpus all nvidia/cuda:12.5.1-base-ubuntu22.04 nvidia-smi
```

...and it didn't. I restarted the machine, thinking that maybe this will help certain settings load properly, and tried again. I also tried running it without the `nvidia-smi` part. Either way, I got the same error message: "docker: Error response from daemon: failed to create task for container: failed to create shim task: OCI runtime create failed: runc create failed: unable to start container process: error during container init: error running hook #0: error running hook: exit status 1, stdout: , stderr: Auto-detected mode as 'legacy'
nvidia-container-cli: initialization error: nvml error: driver not loaded: unknown."

This is apparently a problem, again, with the Nvidia driver. I could try pulling some other docker container to test this further, but I suspect it wouldn't change much if `nvidia-smi` doesn't show anything useful. And I could go on and on trying this out, but maybe I'm lucky again and maybe the pipeline works even without the GPU usage. So the next important step is to generate some small test data to run a pilot test with the pipeline.

Accessing shared folder from VMware virtual machine

I noticed that I could see my former inter-OS shared folder from the VMware virtual machine, but it appeared as empty, even though I set the path to it in the VM settings in VMware Workstation Pro. I unmounted this folder and then checked the VMware website and ChatGPT for how to mount the folder. I realized I need to first check what Linux kernel version I have, so I used

```
uname -srm
```

so I could confirm I have the `Linux 6.5.0-45-generic x86_64` kernel.

I created a new mount point with

```
sudo mkdir /mnt/mnt-win-ubu-shared
```

and then, to access my shared folder at this mount point, used:

```
sudo /usr/bin/vmhgfs-fuse .host:/ /mnt/mnt-win-ubu-shared -o subtype=vmhgfs-fuse,allow_other
```

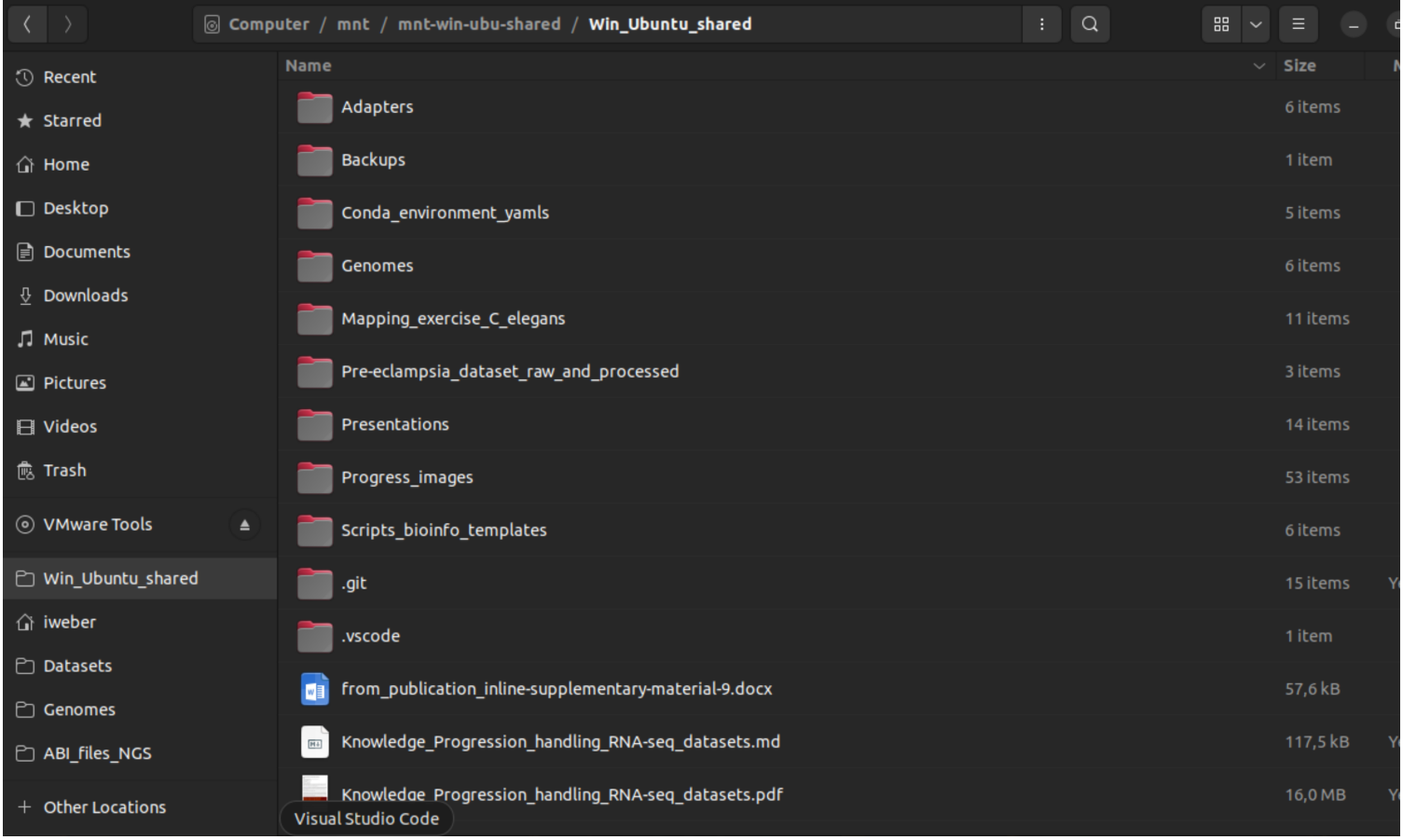
I attempted to make the mounted folder not ask me for my password every time by adding this to `/etc/fstab` (sudo access):

```
.host:/ /mnt/mnt-win-ubu-shared fuse.vmhgfs-fuse defaults,allow_other 0 0
```

and then finally mounted

```
sudo mount -a
```

And restarted my system. I had to go to the mounting point /mnt/mnt-win-ubu-shared from the address bar of my file explorer (ctrl+L) but could then see the folder. I swiftly added it to my bookmarks in the left-hand-side panel.



I also changed the alias in my .bashrc for the command needed to switch to this inter-OS shared folder:

```
# some more handy aliases
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'
alias mydata='cd /home/iweber/Documents'
alias abidata='cd /home/iweber/Documents/ABI_files_NGS'
alias inter-OS='cd /mnt/mnt-win-ubu-shared/Win_Ubuntu_shared'
alias inter-OS_data='cd /mnt/mnt-win-ubu-shared/Win_Ubuntu_shared/Datasets/'
```

I reloaded the .bashrc with `source ~/.bashrc` and set the permissions on the folder for everyone (including my user) to be able to use it:

```
sudo chmod 755 /mnt/mnt-win-ubu-shared
```

Checking that the git repository within the newly mounted inter-OS folder still works

I immediately hit `git status` to make sure my git repo still works, and what I got was an error saying "fatal: detected dubious ownership in repository at '/mnt/mnt-win-ubu-shared/Win_Ubuntu_shared' To add an exception for this directory, call: git config --global --add safe.directory /mnt/mnt-win-ubu-shared/Win_Ubuntu_shared".

To hopefully resolve this, I ran:

```
git config --global --add safe.directory /mnt/mnt-win-ubu-shared/Win_Ubuntu_shared
```

and then tested if it pops up as safe using `git config --global --get-all safe.directory` and its address does show up in the list.

Next, I tested if git is running as it should using `git status`, and it does - I immediately got an overview of untracked changes :) (I added the folders relating to my pilot pipeline run, "Adapters", "Genomes" and "C elegans mapping exercise", to .gitignore because I don't care about tracking them. When I repeated the git status command, they did not show up any longer). Everything was updated as it should on GitHub :)

The rnasplice pipeline

`rnasplice` is a fairly new pipeline on the nf-core, published just a few months back. This is partially the reason for which I really badly want to run it in a container rather than just in conda: I suspect it probably can still throw a number of bugs when run on a system even remotely different than whatever the developers were using.

I pulled the pipeline from nf-core using

```
nextflow pull nf-core/rnasplice
```

```
(Nextflow) iweber@iweber-VirtualBox:/mnt/mnt-win-ubu-shared/Win_Ubuntu_shared$ nextflow pull nf-core/rnasplice
Checking nf-core/rnasplice ...
 downloaded from https://github.com/nf-core/rnasplice.git - revision: 1d0494ae34 [master]
```

While I am trying to transfer the entirety of my virtual machine from VirtualBox to VMware (see section Attempting to transfer entirety of virtual machine to VMware), I read about the pipeline to better understand what the individual steps involve.

Prerequisites for the rnasplice pipeline

Create contrast sheet

--contrasts contraststssheet.csv: I first need to create a contrast sheet, which tells the pipeline what kind of an experiment has been performed and how the conditions are called.

Straight from the pipeline's "contrastsheet input" section:

The contrastsheet has to be a comma-separated file with 3 columns, and a header row as shown in the examples below.

```
contrast,treatment,control
TREATMENT_CONTROL,TREATMENT,CONTROL
```

The contrastsheet can have as many columns as you desire, however, there is a strict requirement for the first 3 columns to match those defined in the table below.

Column	Description
`contrast`	An arbitrary identifier, will be used to name contrast-wise output files.
`treatment`	The treatment/target level for the comparison.
`control`	The control/base level for the comparison.

Therefore, my `contrastssheet_preeclampsia.csv` looks like:

```
contrast,treatment,control
PREECLAMPSIA_CONTROL,PREECLAMPSIA,CONTROL
```

Creating the sample sheet

--input samplesheet.csv: Next, I created a sample sheet (a .csv file telling the pipeline what the sample files are called and which ones are from the control and which ones from the treated animals). [Note from "Source configuration": when using FastQ files as input, file must be structured exactly as shown here]

Straight from the pipeline's "samplesheet input" section:

```
sample,fastq_1,fastq_2,strandedness,condition
CONTROL_REP1,AEG588A1_S1_L002_R1_001.fastq.gz,AEG588A1_S1_L002_R2_001.fastq.gz,forward,control
CONTROL_REP2,AEG588A2_S2_L002_R1_001.fastq.gz,AEG588A2_S2_L002_R2_001.fastq.gz,forward,control
CONTROL_REP3,AEG588A3_S3_L002_R1_001.fastq.gz,AEG588A3_S3_L002_R2_001.fastq.gz,forward,control```
```

The samplesheet can have as many columns as you desire, however, there is a strict requirement for at least 3 columns to match those defined in the table below.

Column	Description	
`sample`	Custom sample name. This entry will be identical for multiple sequencing libraries/runs from the same sample. Spaces in sample names are automatically converted to underscores (`_`).	
`fastq_1`	Full path to FastQ file for Illumina short reads 1. File has to be gzipped and have the extension “.fastq.gz” or “.fq.gz”.	
`fastq_2`	Full path to FastQ file for Illumina short reads 2. File has to be gzipped and have the extension “.fastq.gz” or “.fq.gz”.	
`strandedness`	Sample strand-specificity. Must be one of `unstranded` , `forward` or `reverse` .	

Column	Description	
`condition`	The name of the condition a sample belongs to (e.g. ‘control’, or ‘treatment’) - these labels will be used for downstream analysis.	
`genome_bam`	Full path to aligned BAM file, derived from splicing aware mapper (STAR, HiSat, etc). File has to be in “.bam” format.	
`transcriptome_bam`	Full path to aligned transcriptome file, derived from splicing aware mapper (STAR, HiSat, etc). File has to be in “.bam” format.	
`salmon_results`	Full path to the result folder produced by salmon quantification.	"

I created the sample sheet in Notepad++, and I even remembered to set the line breaks to Unix line breaks (LF instead of CR LF).

	1 Run	2 BioSample	5 Experiment	6 GEO_Accession	9 tissue
1	SRR13761520	SAMN18024800	SRX10148223	GSM5098819	Cortex offspring from control mother mice
2	SRR13761521	SAMN18024799	SRX10148224	GSM5098820	Cortex offspring from control mother mice
3	SRR13761522	SAMN18024798	SRX10148225	GSM5098821	Cortex offspring from control mother mice
4	SRR13761523	SAMN18024797	SRX10148226	GSM5098822	Cortex offspring from control mother mice
5	SRR13761524	SAMN18024796	SRX10148227	GSM5098823	Cortex offspring from preeclampsia mother mice
6	SRR13761525	SAMN18024795	SRX10148228	GSM5098824	Cortex offspring from preeclampsia mother mice
7	SRR13761526	SAMN18024794	SRX10148229	GSM5098825	Cortex offspring from preeclampsia mother mice
8	SRR13761527	SAMN18024793	SRX10148230	GSM5098826	Cortex offspring from preeclampsia mother mice

Complete once established

Therefore, my `samplesheet_preeclampsia.csv` looks like:

Stranded information?

But where is the strandedness of the libraries indicated? There is no mention of this neither in the publication, nor in its Supplemental Data 1, nor in the Supplementary Materials and Methods. I only found a mention in their regular Materials and Methods section that they used the TruSeq Stranded mRNA Library Prep Kit from Illumina, in whose protocol I read that it encompasses a first-strand reverse transcription and then the generation of the second strand of the cDNA, complementary to the first. This means that the resulting cDNA has the forward strand identical to the reverse complement of the original mRNA and so should be treated as "reverse" and specified as such in the subsequent process. However, how can I make super sure that this is true? Giving the wrong strandedness will render the entire analysis essentially useless, this is highly important.
[explain the underlying biology]

Side note: how does the kit distinguish between what is the first strand (reverse strand) and the forward strand? According to the TruSeq protocol, this is done by using a desoxyribonucleotide in the second-strand preparation that is different from the first-strand preparation (dUTP instead of dTTP for the first strand - they will both hybridize with adenosine, but the presence of Ts in one cDNA strand and the presence of Us in the other makes it possible to tell apart which strand the reads stem from)

The blessings of online searching pointed me to a Python package called How are we stranded here , which analyzes fastq files precisely to understand this. The knack: it also needs some additional information about the organism at hand, such as a gtf annotation file and a kallisto transcriptomic index. This seems like a very complicated option, so I asked Ioana Lemnian whether that is necessary. She said no, because it can only be that the `_R1` FastQ files are reverse and `_R2` files forward, so the strandedness should be specified as `RF` (she also sent me some useful resources: this one from ECSeq and this from the Broad Institute

One question remained in this context, though: how should one specify this in the sample sheet? It only takes one single indication of strandedness for each sample and both of its `_R1` and `_R2` files with reads together, and that indication can only be either `forward`, `reverse`, or `unstranded`. I realized nf-core has its very own Slack channel, where I swiftly got a detailed answer from Thomas Danhorn , indicating that the strandedness in this case has to follow that of the R1 file, so, in my case, be `reverse`.

This is how the sample sheet looks like in the end:

samplesheet_preeclampsia.csv

```
sample,fastq_1,fastq_2,strandedness,condition
CONTROL_REP1,SRR13761520_1.fastq,SRR13761520_1.fastq,reverse,CONTROL
CONTROL_REP2,SRR13761521_1.fastq,SRR13761521_2.fastq,reverse,CONTROL
CONTROL_REP3,SRR13761522_1.fastq,SRR13761522_2.fastq,reverse,CONTROL
CONTROL_REP4,SRR13761523_1.fastq,SRR13761523_2.fastq,reverse,CONTROL
PREECLAMPSIA_REP1,SRR13761524_1.fastq,SRR13761524_2.fastq,reverse,PREECLAMPSIA
PREECLAMPSIA_REP2,SRR13761525_1.fastq,SRR13761525_2.fastq,reverse,PREECLAMPSIA
PREECLAMPSIA_REP3,SRR13761526_1.fastq,SRR13761526_2.fastq,reverse,PREECLAMPSIA
PREECLAMPSIA_REP4,SRR13761527_1.fastq,SRR13761527_2.fastq,reverse,PREECLAMPSIA
```

Configuration of the type of source files

Since I have FastQ files, I can leave the source configuration to the default, `--source fastq`.

Gzipping FastQ files for pipeline with pigz

I noticed in the parameters that the pipeline doesn't work with FastQ files directly, but only with their gzipped versions. I started the `gzip SRR13761520_1.fastq` command at 13:09, took around 10 min. So I know that doing the same with the remaining 15 files in sequence should take something like 150 minutes ~ 3 h. Which is why I'll only start after my recovery drive is complete, I switch off hyper-V, and I dare to restart my PC to see if it is indeed off and confirm my system didn't turn into blue pulp (BSOD).

The script I want to use to gzip all files at once: (~~pigz, which allows hyperthreading, won't work on my VirtualBox VM because of the same issues with GPU passthrough, so I'll have to take this slow route...~~):

gzip_several_fastqs.sh

```
#!/bin/bash

# Specify directory containing FASTQ files
DIRECTORY="/mnt-win-ubu-shared/Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq"

# Find all .fastq files in the specified directory and subdirectories. Write line by line, pipe each line with one file name into while loop that compresses it.
find "$DIRECTORY" -type f -name "*.fastq" | while read -r FILE; do
  echo "Compressing $FILE"
  gzip "$FILE"
done
```

Scratch the above, I finally fixed all of these issues and now can use pigz, taking advantage of its parallel processing capabilities. So I need only replace the command with pigz:

pigz_several_fastqs.sh

```
#!/bin/bash

# Specify directory containing FASTQ files
DIRECTORY="/mnt/mnt-win-ubu-shared/Win_Ubuntu_shared/Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_raw_fastq/"

# Find all .fastq files in the specified directory and subdirectories. Write line by line, pipe each line with one file name into while loop that compresses it.
find "$DIRECTORY" -type f -name "*.fastq" | while read -r FILE; do
  echo "Compressing $FILE"
  pigz "$FILE"
done
```

GTF file for the mouse mm10 genome

rnasplice also needs a GTF annotation file that indicates where genes, transcripts, exons, etc are located within the mouse genome. Turns out, I had already gotten the mouse gtf a few months back, when I downloaded the mouse genome from NCBI (https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/001/635/GCF_000001635.27_GRCm39/). This is good because I have the latest assembly of the mouse genome from NCBI, so it is more consistent to also use the NCBI annotations, especially since I am interested in looking at potentially disease-relevant genes, and I know NCBI has very good integration of its genomic data with its clinical databases, whereas Ensembl is somewhat better at having extremely detailed annotations, which come in handy for comparative/evolutionary studies.

However, the annotation file from NCBI is in GFF3 format, so I might need to convert it to GTF. I found an overview of what these two file types are made of at Ensembl, and it seems that GTF2 and GFF2 files are identical. BUT: reading on on the parameters page, I found that -gff is also an option that allows the use of GFF files instead of GTF, so yay for no conversion needed!

Even better: the GTF file is also included in the NCBI FTP with downloads for this genome assembly.

To get the mouse GTF from the NCBI FTP (already had the genome from them), used `wget https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/001/635/GCF_000001635.27_GRCm39/GCF_000001635.27_GRCm39_genomic.gtf.gz` and then `pigz -d` to unzip it.

Pipeline description for rnasplice

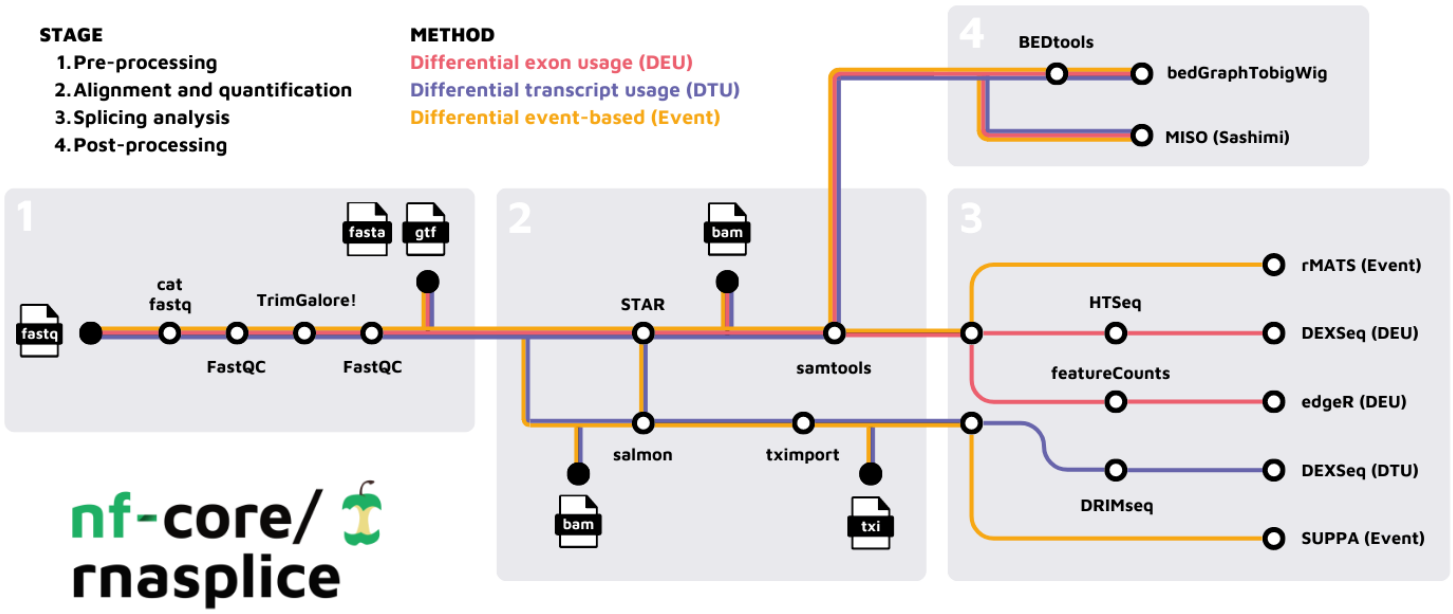
Calling the pipeline could be as easy as typing

```
nextflow run nf-core/rnasplice \
--input samplesheet.csv \
--contrasts contrastsheet.csv \
--genome GRCh37 \
--outdir my/result/directory \
-profile docker
```

but what does it do, and what other parameters need to be set?

Graphical overview

I tried creating an overview here that is somewhat easier to describe with words than the image on the website:



Parts of the pipeline

> ***Part 1 of pipeline: preprocessing***
>
> > **cat fastq**
> > is something I would use if I had several fastq files per sample, coming from several runs of sequencing. This is usually done to increase sequencing depth, but the SRA archives I downloaded contained precisely two files per sample, containing the two parts of a pair of reads each.
> >
> > **FastQC**
> > *own addition*: **MultiQC** to summarize results
> > I already performed these steps on the WSL, so will not need to do so again. I already know I need to trim 10 bases from the 5' end of the reads in order to solve any problems that the suboptimal per-base distribution of the four nucleotides may create.
> >
> > > **TrimGalore**! - GitHub and user guide
> > > - **COMMAND**: `trim_galore [options] <filename(s)>`
> > > - will set `--cores 4` to make maximal use of all of my cores but following the indication on the website that says this is a sweet spot (I set 6 cores to be available to the VM)
> > > - will use the `--paired` option because I have paired reads as input

Install

```
- However, this requires pigz instead of gzip, so I will have to eventually install it
```

```
- see above - need to trim 10 bases. I previously worked with Trimmomatic for this kind of operation, but TrimGalore also has the option `--clip_R1 10` and `--clip_R2 10`, which will remove these 10 bases from the 5' ends of all of the reads.
```

Install

```
- TrimGalore is a wrapper around FastQC and another program, meant for adapter trimming, `cutadapt`, which needs to be previously installed - DO IT
```

```
- Since it employes FastQC, TrimGalore does an automated check of low quality and trims accordingly. I don't need to specify the Phred score cutoff for base call quality - it automatically sets it to the most modern version, --phred33.
- in my previous checks, FastQC did not detect any significant adapter contamination in the reads. If there are any adapters left, they will be of the Illumina type, since this is what kit was employed. However, I am a little reluctant to just use the --illumina parameter, as this trimming is extremely stringent and an overlap of even only 1 base between the end of the read and the adapter sequences will be removed, potentially losing useful information in the process. I will rather let TrimGalore automatically look for potential overlaps with adapter sequences and set `--stringency` to `5` to lose less information, since the risk of real adapter contamination seems extremely low AND I am anyway trimming the 5' ends because of the biased  base distribution
-  it also automatically filters the resulting sequences and, if a read is then shorter than 20 bp, it is automatically removed from the results.
- output: [***does it also create separate files for reads that could successfully be paired up from the two input FastQ files and those that could not?***]

>>>
```

FastQC

- these two steps are done to check whether the trimming worked exactly as expected, so, in my case, I'd expect the new report generated by MultiQC to show me 140 bp reads with no more issues at the 5' ends.
- to run FastQC again on the results, use `--fastqc_args "--outdir /TrimGalore_output"` (TrimGalore automatically creates the output directory if it doesn't exist)
  -*own addition*: **MultiQC** to summarize results

To do

```
I will then need to add a MultiQC step to summarize the results, but I can do that manually
```

input files: fasta and gtf - see sections Downloading genomes and Initial creation of genome and transcriptome indices

[this certainly needs more research for accuracy!]
The rest of the pipeline is divided in two branches. Each branch starts with one of two different algorithms, STAR and Salmon, whose mechanics sound similar at first, but work quite differently.

As explained here by Devon Ryan, an aligner like STAR genuinely looks at the base-to-base match of a read to a region in the target genome ("does this read align with this bit of the genome?") and also saves information on whether the read matches the genomic region as-is, from one end of the read to the other, or whether there are gaps or insertions ("does every one of the 150 bases in the read match 150 bases at the location where it's mapped in the genome? Or are, say, only bases 1-57 aligned, with an unmatching portion from bases 58-92, and then 93-150 match again?"). This means that it it can accurately quantify which genes are expressed at what levels. Additionally, it is built so that it can easily identify reads that span exon-exon boundaries, thereby making it able to distinguish which of the different transcripts that can come from the same genomic locus the read could be derived from. This information is usually collected in a BAM file.

To do:

Read more on how Salmon works - the Harvard Chan Bioinfo Core PDF has good visualizations.

STAR is the gold standard of the old school of transcript abundance analysis. Salmon is the new star on the block because it can be used in two different ways: as a pseudo-aligner or as a real aligner. It truly shines as a pseudo-aligner, a mode in which it does not care about the exact fit of the read base-to-base, and does a quantification of "is this read likely to come from this transcript?" based on some more complex background mathematics. This is called pseudo-alignment, and is much faster than an actual base-to-base alignment, like STAR performs. In this mode, Salmon does not compare the reads to a reference genome but to a reference transcriptome, for which it needs a so-called index of the transcriptome. Often, such reference indices are readily available for download for well-studied organisms such as the mouse.

Salmon also makes some adjustments for normalizing the read counts, e.g. to the length of the transcript, in order to produce accurate estimates of how abundantly different splice isoforms are expressed from a particular gene. It can take the BAM files generated by STAR in order to give these results, which we will obtain with STAR in the first part of the pipeline. [how does tximport play into this?]

Salmon can also operate on raw reads from scratch, as an aligner like STAR would, and I could theoretically let it start its branch of the pipeline from scratch, based on the raw reads, but I will anyway get the BAM files from the STAR branch, so I could save some time on that.

The creators of the dataset/authors of the original study used Hisat2 instead of STAR to align the reads to the mm10 mouse genome, and HTSeq to obtain read counts aka expression levels for the genes. The rnasplice pipeline also runs HTSeq after the STAR step, and I am curious how the results will compare to the ones published in the paper.

> ***Part 2a of pipeline: read alignment and read count quantification starting with STAR***
>
> > **STAR**
> > aligns the reads
> > Parameters to consider:
> > - `--star_index` : Provide the path to the STAR index built for your reference genome.
> > - `--genome` : Specify the reference genome (e.g., GRCh38).
> > - `--sjdb_overhang` : Adjust based on your read length. A common choice is `read_length - 1`, or else it defaults to 100 bp. So I will use ***149***, because I am dealing with 150 bp reads.
> > - Here, Renesh Bedre explains more of the other parameters that STAR can use, such as `--runThreadN`, which sets on how many cores the aligner should be run (I'll use 6, because that's how many I have)
>
> > > **samtools**
> > >
> > > > ***Part 3a of pipeline: splicing quantification with edgeR, DEXSeq***
> > > >
> > > > **rMATS** = splicing event quantification
> > > > `--rmats_threads` : Set the number of threads to use for `rMATS`
> > > >
> > > > Then follow two branches that deal with finding differentially expressed exons in the data:
> > > >
> > > > > **HTSEq** - part of the DEXSeq package
> > > > >
> > > > > > **DEXSeq** = differential exon usage. A package, activated when the parameter `--dexseq_exon` is enabled.
> > > > > >
> > > > > > - "Using the `--aggregation` parameter the pipeline will combine overlapping genes into a single aggregate gene. This approach can alternatively be skipped and any exons that overlap other exons from different genes will be skipped. Other important options to take note of are the `--alignment_quality` parameter which can be set by the user and defines the minimum alignment quality required for reads to be included (defined in 5th column of a given SAM file) (default: 10). Prior to quantification, DEXSeq provides an annotation preparation script which takes a GTF file as input and returns a GFF file. Users may instead wish to define their own GFF file and skip this annotation preparation skip by supplying it using the `--gff_dexseq` parameter."
> > > > > >
> > > > > > **featureCounts**
> > > > >
> > > > > ```
> > > > > - provides a quantification that edgeR requires
> > > > > - " is activated when the parameter `--edger_exon` is enabled. Please note that as this is aimed at
> > > > > differential exon usage feature type is set as `exon` and cannot be changed. Please take care to use a
> > > > > suitable attribute to categorize the featureCounts attribute type in your GTF using the option `--
> > > > > gtf_group_features` (default: `gene_id`)."
> > > > > ```
> > > > >
> > > > > **edgeR** provides info on differential exon usage, based on featureCounts

Part 4 of the pipeline does some basic processing of the BAM files resulting from STAR in order to make them easy to display in a genome browser, together with a quantification of how frequently certain exon-exon junctions are encountered. This last step is performed by MISO, which creates so-called Sashimi plots with the exons and their junctions.

> > > ***Part 4 of pipeline: post-processing after STAR/samtools branch***
> > >
> > > > **BEDtools**
> > > >
> > > > > **bedGraphTobigWig**
> > > > >
> > > > > **MISO/Sashimi**

Part 2b of pipeline: read psued-alignment and read count quantification per transcript using Salmon

Salmon

tximport

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4712774/ tximport solves issues with artificially inflated gene counts derived from transcript isoforms

Part 3b of pipeline: actual splicing junction quantification

DRIMSeq

DEXSeq = differential transcript usage

SUPPA = differential event-based splicing

Initial creation of genome and transcriptome indices

STAR independent installation and mouse genome index

I decided to create my STAR and Salmon indices in advance in order to remove potential sources of pipeline getting stuck, so I had to install both independent of the pipeline.

From the ~~horse's~~ scientist's mouth who developed STAR, Alexander Dobin, I took the instructions on how to create one's own STAR index, and did so before running the pipeline (see also here)

To first get STAR separately from the Nextflow pipeline, I went into my Nextflow environment and ran:

```
wget https://github.com/alexdobin/STAR/archive/2.7.11b.tar.gz
tar -xzf 2.7.11b.tar.gz
cd STAR-2.7.11b/source
```

Then, as instructed, I ran `make STAR` for the gcc C++ compiler to create a ready-to-run version of STAR on my system. Started around 11:57, took around 3 min to complete.
Got a warning "make: warning: Clock skew detected. Your build may be incomplete."

I reset my system time with `sudo apt-get install ntp` and `sudo service ntp restart`, then re-made the STAR build with `make clean` and `make STAR`, which now worked with no further errors.

I had to add the path to the folder where I compiled the executable to the PATH variable, which I did by adding to my .bashrc `export PATH=/home/iweber/Documents/Software/STAR-2.7.11b/source:$PATH`

To make the mouse STAR index, ran:

```
STAR --runThreadN 6 --runMode genomeGenerate --genomeDir /mnt/mnt-win-ubu-
shared/Win_Ubuntu_shared/Genomes/genome_M_musculus/GCF_000001635.27/STAR_index_M_musculus --genomeFastaFiles /mnt/mnt-win-ubu-
shared/Win_Ubuntu_shared/Genomes/genome_M_musculus/GCF_000001635.27/GCF_000001635.27_GRCm39_genomic.fna --sjdbGTFfile
/mnt/mnt-win-ubu-shared/Win_Ubuntu_shared/Genomes/genome_M_musculus/GCF_000001635.27/GCF_000001635.27_GRCm39_genomic.gtf --
sjdbOverhang 149
```

Annnnnnnd success! Took all of 27 min to complete.

```
        STAR --runThreadN 6 --runMode genomeGenerate --genomeDir /mnt/mnt-win-ub
u-shared/Win_Ubuntu_shared/Genomes/genome_M_musculus/GCF_000001635.27/STAR_index
_M_musculus --genomeFastaFiles /mnt/mnt-win-ubu-shared/Win_Ubuntu_shared/Genomes
/genome_M_musculus/GCF_000001635.27/GCF_000001635.27_GRCm39_genomic.fna --sjdbGT
Ffile /mnt/mnt-win-ubu-shared/Win_Ubuntu_shared/Genomes/genome_M_musculus/GCF_00
0001635.27/GCF_000001635.27_GRCm39_genomic.gtf --sjdbOverhang 149
        STAR version: 2.7.11b   compiled: 2024-08-07T12:28:19+02:00 :/home/iwebe
r/Documents/Software/STAR-2.7.11b/source
Aug 07 17:23:32 ..... started STAR run
Aug 07 17:23:32 ... starting to generate Genome files
Aug 07 17:24:02 ..... processing annotations GTF
Aug 07 17:24:24 ... starting to sort Suffix Array. This may take a long time...
Aug 07 17:24:31 ... sorting Suffix Array chunks and saving them to disk...
Aug 07 17:39:52 ... loading chunks from disk, packing SA...
Aug 07 17:41:12 ... finished generating suffix array
Aug 07 17:41:12 ... generating Suffix Array index
Aug 07 17:44:38 ... completed Suffix Array index
Aug 07 17:44:39 ..... inserting junctions into the genome indices
Aug 07 17:48:41 ... writing Genome to disk ...
Aug 07 17:48:44 ... writing Suffix Array to disk ...
Aug 07 17:49:05 ... writing SAindex to disk
Aug 07 17:49:07 ..... finished successfully
```

Salmon installation and mouse transcriptome index

For the Salmon index creation, I first got the latest Salmon version independent of the Nextflow pipeline as instructed on the Salmon website using

```
wget https://github.com/COMBINE-lab/salmon/releases/download/v1.10.0/salmon-1.10.0_linux_x86_64.tar.gz
```

I decompressed the tar archive with `tar -xvzf salmon-1.10.0_linux_x86_64.tar.gz`, and then added the path to Salmon's bin folder to my PATH variable by including `export PATH=/home/iweber/Documents/Software/salmon-latest_linux_x86_64/bin:$PATH` in my .bashrc file. I could then see the command autocomplete, so I knew it works on my system :) `salmon --version` also returned "1.10.0", which is correct.

I also tried to get a Docker image with

```
sudo docker pull combinelab/salmon
```

> **Note**
>
> which...did not seem to download anything to my "Software" folder that I ran the command in ***pondering smiley*** . I'm sure this rather has something to do with me not understanding how the Docker image is supposed to work.

Next, I went looking for information on how to set up the transcriptomic index with Salmon. The Salmon manual sent me to this RefGenie server/repository that has ready-made indices for well-studied species, and it indeed hosts a Salmon index for the mouse, matching the NCBI genome, here. I chose to download the complete Salmon index generated with selective alignment method, because the description says that using it will increase the accuracy of quantification.

I made a new folder, Salmon_index_M_musculus_mm10, in the genome folder (/mnt/mnt-win-ubu-shared/Win_Ubuntu_shared/Genomes/genome_M_musculus/GCF_000001635.27/), then used wget to download the index from RefGenie:

```
wget http://refgenomes.databio.org/v3/assets/archive/0f10d83b1050c08dd53189986f60970b92a315aa7a16a6f1/salmon_sa_index?tag=default
```

which...didn't work. Even though I saw the 12 GB download and it took around 30 min to do so, I had no files at the end in my working directory and only an odd file that popped up with `ls -lha`:

```
s/genome_M_musculus/GCF_000001635.27/Salmon_index_M_musculus_mm10$ ls -lha
ls: cannot access 'salmon_sa_index?tag=defaul': No such file or directory
total 4,0K
drwxrwxrwx 1 root root    0 Aug  7 19:11 .
drwxrwxrwx 1 root root 4,0K Aug  7 19:09 ..
-????????? ? ?    ?       ?            ? 'salmon_sa_index?tag=defaul'
```

ChatGPT told me that the file with the utterly weird privilege indicators (question marks) and the truncated name is likely the fault of a corrupted download, and that that likely happened because there are special characters (question mark) in the archive name on the website. So it suggested re-

downloading the archive with a clearly defined name using

```
wget "http://refgenomes.databio.org/v3/assets/archive/0f10d83b1050c08dd53189986f60970b92a315aa7a16a6f1/salmon_sa_index?tag=default" -O salmon_sa_index.tgz
```

...but that only downloaded a tiny file, nothing close to the 12 GB I was expecting from the tgz archive. So I went the new-school way and just downloaded it through the browser *sweat drop smiley*

Side note: *I found out later that RefGenie is actually a Python package that allows downloading files related to some reference genome assemblies from the command line; here's how to use it for further reference.*

Final setup of the rnasplice pipeline

```
nextflow run nf-core/rnasplice -r 1.0.4

--input samplesheet_preeclampsia.csv
--contrasts contrastssheet_preeclampsia.csv

--fasta /mnt/mnt-win-ubu-shared/Win_Ubuntu_shared/Genomes/genome_M_musculus/GCF_000001635.27/GCF_000001635.27_GRCm39_genomic.fna
--gtf /mnt/mnt-win-ubu-shared/Win_Ubuntu_shared/Genomes/genome_M_musculus/GCF_000001635.27/GCF_000001635.27_GRCm39_genomic.gtf
--outdir /mnt/mnt-win-ubu-shared/Win_Ubuntu_shared/Pre-eclampsia_dataset_raw_and_processed/Pre_eclampsia_mice_rnasplice_results


--aligner star_salmon
--star_index /mnt/mnt-win-ubu-shared/Win_Ubuntu_shared/Genomes/genome_M_musculus/GCF_000001635.27/STAR_index_M_musculus/
--salmon_index path/to/salmon/index
--save_align_intermeds true # needed if wanting to use rMATS, because otherwise pipeline does not save SAM files, which rMATS needs
--save_reference true # saves anything the pipeline downloads by itself, e.g. STAR indices

--salmon_quant_libtype # this might need to go into Salmon's chunk in the workflow AND is optional - Salmon can infer this automatically from the samplesheet information


-profile docker # or conda, depending on what works
```

Pipeline pilot experiment: *C. elegans* data from the course

The C. elegans data

In the NGS part of the bioinformatics course, we mapped some reads from the C. elegans genome. I'd like to use these as a test dataset for the pipeline, because the data consisted of far fewer reads than what I have for my mouse experiment, and so I should be able to see fast if the pipeline works or not.

I have the reads files stored as fastq.gz files under
`/mnt/mnt-win-ubu-shared/Win_Ubuntu_shared/Mapping_exercise_C_elegans/Fastq_original_C_elegans/`

and the C. elegans genome stored under
`/mnt/mnt-win-ubu-shared/Win_Ubuntu_shared/Genomes/genome_C_elegans/GCF_000002985.6_WBcel235_genomic.fna`

C. elegans GTF file

To get the GTF file, an annotation file that specifies where genes and other genomic elements start and end within the genome, I went to Ensembl's FTP page and found that the *C. elegans* GTFs are stored under https://ftp.ensembl.org/pub/current_gtf/caenorhabditis_elegans/Caenorhabditis_elegans.WBcel235.112.gtf.gz. I navigated to my *C. elegans* genome folder and downloaded the file with

```
wget  https://ftp.ensembl.org/pub/current_gtf/caenorhabditis_elegans/Caenorhabditis_elegans.WBcel235.112.gtf.gz
```

For the relatively small *C. elegans* genome, this took all of 1 minute. I then unzipped it with pigz:

```
pigz -d Caenorhabditis_elegans.WBcel235.112.gtf.gz
```

STAR index for C. elegans

I tried to prepared the STAR index for C. elegans using

```
STAR --runThreadN 6 --runMode genomeGenerate --genomeDir /mnt/mnt-win-ubu-
shared/Win_Ubuntu_shared/Genomes/genome_C_elegans/STAR_index_C_elegans --genomeFastaFiles /mnt/mnt-win-ubu-
shared/Win_Ubuntu_shared/Genomes/genome_C_elegans/GCF_000002985.6_WBcel235_genomic.fna --sjdbGTFfile /mnt/mnt-win-ubu-
shared/Win_Ubuntu_shared/Genomes/genome_C_elegans/Caenorhabditis_elegans.WBcel235.112.gtf --sjdbOverhang 149
```

but got an error saying something is not right with the GTF file: "Fatal INPUT FILE error, no valid exon lines in the GTF file: /mnt/mnt-win-ubu-shared/Win_Ubuntu_shared/Genomes/genome_C_elegans/Caenorhabditis_elegans.WBcel235.112.gtf
Solution: check the formatting of the GTF file. One likely cause is the difference in chromosome naming between GTF and FASTA file."

As I learned from the Harvard Chan Bioinformatics Core course on RNA-Seq, I need to use the GFF file from NCBI instead, or, at the very least, rename the chromosomes in the GTF files so that they have the same names as they do in the genome. GTF files and genomic files need to be not only from the same build of the genome (so mm10, hg38, etc) but also from the same source, such as NCBI or Ensembl, because the different databases have somewhat different ways of formatting the names of elements in their genomes, such as the names of chromosomes.

So I got the GTF file from NCBI with `wget` `https://ftp.ncbi.nlm.nih.gov/genomes/all/GCF/000/002/985/GCF_000002985.6_WBcel235/GCF_000002985.6_WBcel235_genomic.gtf.gz`, then unzipped it with `pigz`. I then repeated the index creation using the GTF file from NCBI:

```
STAR --runThreadN 6 --runMode genomeGenerate --genomeDir /mnt/mnt-win-ubu-
shared/Win_Ubuntu_shared/Genomes/genome_C_elegans/STAR_index_C_elegans --genomeFastaFiles /mnt/mnt-win-ubu-
shared/Win_Ubuntu_shared/Genomes/genome_C_elegans/GCF_000002985.6_WBcel235_genomic.fna --sjdbGTFfile /mnt/mnt-win-ubu-
shared/Win_Ubuntu_shared/Genomes/genome_C_elegans/GCF_000002985.6_WBcel235_genomic.gtf --sjdbOverhang 149
```

It worked, except for a warning: "!!!!! WARNING: --genomeSAindexNbases 14 is too large for the genome size=100286401, which may cause seg-fault at the mapping step. Re-run genome generation with recommended --genomeSAindexNbases 12"

I looked in the STAR manual to gain a better understanding of why this is happening. Apparently, this is because the C. elegans genome is very small, hence the request to run the index generation again with this parameter set to 12. So I deleted all of the files that STAR generated for the index, and started afresh with:

```
STAR --runThreadN 6 --runMode genomeGenerate --genomeDir /mnt/mnt-win-ubu-
shared/Win_Ubuntu_shared/Genomes/genome_C_elegans/STAR_index_C_elegans --genomeFastaFiles /mnt/mnt-win-ubu-
shared/Win_Ubuntu_shared/Genomes/genome_C_elegans/GCF_000002985.6_WBcel235_genomic.fna --sjdbGTFfile /mnt/mnt-win-ubu-
shared/Win_Ubuntu_shared/Genomes/genome_C_elegans/GCF_000002985.6_WBcel235_genomic.gtf --sjdbOverhang 149 --
genomeSAindexNbases 12
```

And ta-da! Two minutes later, with no more warnings, I had the index, a set of files that total some 1.4 GB in size ***insert meow_party smiley***

Salmon index for C. elegans

I had less luck with finding a pre-assembled C. elegans Salmon (transcriptomic) index than I had with the mouse one, so I needed to build this index from scratch.

Getting transcriptome file for C. elegans

For this, according to the Salmon manual, I need a file with all of the annotated C. elegans transcripts, so something with RNA or cDNA in the name. I went to NCBI's genome FTP repository for C. elegans and found two files, called "GCF_000002985.6_WBcel235_rna.fna.gz" and "GCF_000002985.6_WBcel235_rna_from_genomic.fna.gz", 13 and 14 MB in size, respectively. Which one to use, though? I checked the FTP FAQ page of the NCBI, but the descriptions about the two files are not necessarily helpful. Since this is just a pilot experiment, I decided to go with the simple rna.fna.gz file and downloaded it with wget.

> **To do:**
>
> clarify this for the future

Getting dependencies for index building

MashMap

To create the index, the Salmon manual talks about two ways to build a so-called *decoy-aware* transcriptome, and, for that, it first needs to build something called a *decoy file*. To do so, it needs a tool called MashMap. I already had GCC and g++, which are needed for compiling the MashMap binaries, but didn't have cmake.

cmake

To get an installation script, I ran

```
wget https://github.com/Kitware/CMake/releases/download/v3.30.2/cmake-3.30.2-linux-x86_64.sh
```

then set the script to executable with `chmod u+x`, ran it with `./cmake-3.30.2-linux-x86_64.sh`, and added the path to the newly created binaries file to my PATH variable in .bashrc. I could then call it from the command line.

GNU GSL

MashMap also needs the GNU GSL tool, which I got in its latest stable version, 2.8, from https://www.gnu.org/software/gsl/ as a tar archive, and immdiately unpacked it. I went to the folder that generated, ran `./configure` to read my system properties, `make` to make a configuration suited to my system, which...took a moment, and many lines of code running in the background. It ended with the lines "`make[2]: Leaving directory '/home/iweber/Documents/Software/gsl-2.8'` make[1]: Leaving directory '/home/iweber/Documents/Software/gsl-2.8`, and, finally,` sudo make install `to run the installation program that was just compiled. I now have access to` gsl-config --version`, for instance :) The binary is in /usr/local/bin, so I did not need to add it to my PATH variable.

Building MashMap

I downloaded MashMap from its git repository with `git clone https://github.com/marbl/MashMap.git` and went into the newly-created directory. Then, as per the instructions on MashMap's GitHub repo, typed `cmake -H. -Bbuild -DCMAKE_BUILD_TYPE=Release` and then `cmake --build build`. This is what I got:

```
(base) iweber@iweber-VirtualBox:~/Documents/Software/MashMap$ cmake --build buil
d
[ 16%] Building CXX object CMakeFiles/mashmap.dir/src/common/utils.cpp.o
[ 33%] Building CXX object CMakeFiles/mashmap.dir/src/map/mash_map.cpp.o
[ 50%] Linking CXX executable bin/mashmap
[ 50%] Built target mashmap
[ 66%] Building CXX object CMakeFiles/mashmap-align.dir/src/common/utils.cpp.o
[ 83%] Building CXX object CMakeFiles/mashmap-align.dir/src/align/align.cpp.o
[100%] Linking CXX executable bin/mashmap-align
[100%] Built target mashmap-align
(base) iweber@iweber-VirtualBox:~/Documents/Software/MashMap$
```

Looking into the MashMap folders, I saw a /bin folder and added it to my PATH variable, and now I can call it from the command line.

Building the decoys for *C. elegans*

Get SalmonTools

To create the decoys, I got SalmonTools from GitHub with `git clone https://github.com/COMBINE-lab/SalmonTools.git` and went into the newly created directory, SalmonTools. ChatGPT advised to make a separate directory for building the compiled tool, so I did with `mkdir build` and changed into it. I configured the build with `cmake ..`, and then made it with `make`. It was indeed made, but with quite some errors:

"[7%] Creating directories for 'libspdlog'
[15%] Performing download step for 'libspdlog'
% Total % Received % Xferd Average Speed Time Time Time Current
Dload Upload Total Spent Left Speed
0 0 0 0 0 0 0 0 --:--:-- --:--:-- --:--:-- 0
100 147k 0 147k 0 0 121k 0 --:--:-- 0:00:01 --:--:-- 922k
spdlog-v0.12.0.tar.gz: OK
[23%] No update step for 'libspdlog'
[30%] No patch step for 'libspdlog'
[38%] No configure step for 'libspdlog'
[46%] No build step for 'libspdlog'
[53%] Performing install step for 'libspdlog'
[61%] Completed 'libspdlog'
[61%] Built target libspdlog
[69%] Building CXX object src/CMakeFiles/salmon_tools_core.dir/FastxParser.cpp.o
[76%] Building CXX object src/CMakeFiles/salmon_tools_core.dir/ExtractUnmapped.cpp.o
In file included from /home/iweber/Documents/Software/SalmonTools/include/zstr.hpp:16,
from /home/iweber/Documents/Software/SalmonTools/src/ExtractUnmapped.cpp:10:
/home/iweber/Documents/Software/SalmonTools/include/strict_fstream.hpp: In static member function ‘static void strict_fstream::detail::static_method_holder::check_peek(std::istream*, const string&, std::ios_base::openmode)’:
/home/iweber/Documents/Software/SalmonTools/include/strict_fstream.hpp:128:39: warning: catching polymorphic type ‘class std::ios_base::failure’ by

value [-Wcatch-value=]
128 | catch (std::ios_base::failure e) {}
| ^
/home/iweber/Documents/Software/SalmonTools/src/ExtractUnmapped.cpp: In function 'void ExtractUnmapped(const string&, std::vector<std::cxx11::basic_string >::const_iterator, std::vector<std::cxx11::basic_string >::const_iterator)':
/home/iweber/Documents/Software/SalmonTools/src/ExtractUnmapped.cpp:210:18: warning: catching polymorphic type 'class args::Help' by value [-Wcatch-value=]
210 | catch (args::Help)
| ^~~~
/home/iweber/Documents/Software/SalmonTools/src/ExtractUnmapped.cpp:215:29: warning: catching polymorphic type 'class args::ParseError' by value [-Wcatch-value=]
215 | catch (args::ParseError e)
| ^
In file included from /home/iweber/Documents/Software/SalmonTools/src/ExtractUnmapped.cpp:8:
/home/iweber/Documents/Software/SalmonTools/include/sparsepp/spp.h: In instantiation of 'void spp::sparsetable<T, Alloc>::resize(spp::sparsetable<T, Alloc>::size_type) [with T = std::pair<const std::cxx11::basic_string, std::cxx11::basic_string >; Alloc = spp::libc_allocator<std::pair<const std::cxx11::basic_string, std::cxx11::basic_string > >; spp::sparsetable<T, Alloc>::size_type = long unsigned int]':
/home/iweber/Documents/Software/SalmonTools/include/sparsepp/spp.h:2942:25: required from 'void spp::sparse_hashtable<Value, Key, HashFcn, ExtractKey, SetKey, EqualKey, Alloc>::_move_from(spp::sparse_hashtable<Value, Key, HashFcn, ExtractKey, SetKey, EqualKey, Alloc>::MoveDontCopyT, spp::sparse_hashtable<Value, Key, HashFcn, ExtractKey, SetKey, EqualKey, Alloc>&, spp::sparse_hashtable<Value, Key, HashFcn, ExtractKey, SetKey, EqualKey, Alloc>::size_type) [with Value = std::pair<const std::cxx11::basic_string, std::cxx11::basic_string >; Key = std::cxx11::basic_string; HashFcn = spp::spp_hash<std::cxx11::basic_string >; ExtractKey = spp::sparse_hash_map<std::cxx11::basic_string, std::cxx11::basic_string >::SelectKey; SetKey = spp::sparse_hash_map<std::cxx11::basic_string, std::cxx11::basic_string >::SetKey; EqualKey = std::equal_to<std::cxx11::basic_string >; Alloc = spp::libc_allocator<std::pair<const std::cxx11::basic_string, std::__cxx11::basic_string > >; spp::sparse_hashtable<Value, Key, HashFcn, ExtractKey, SetKey, EqualKey, Alloc>::size_type = long unsigned int]'
/home/iweber/Documents/Software/SalmonTools/include/sparsepp/spp.h:3093:9: required from 'spp::sparse_hashtable<Value, Key, HashFcn, ExtractKey, SetKey, EqualKey, Alloc>::sparse_hashtable(spp::sparse_hashtable<Value, Key, HashFcn, ExtractKey, SetKey, EqualKey, Alloc>::MoveDontCopyT, spp::sparse_hashtable<Value, Key, HashFcn, ExtractKey, SetKey, EqualKey, Alloc>&, spp::sparse_hashtable<Value, Key, HashFcn, ExtractKey, SetKey, EqualKey, Alloc>::size_type) [with Value = std::pair<const std::cxx11::basic_string, std::cxx11::basic_string >; Key = std::cxx11::basic_string; HashFcn = spp::spp_hash<std::cxx11::basic_string >; ExtractKey = spp::sparse_hash_map<std::cxx11::basic_string, std::cxx11::basic_string >::SelectKey; SetKey = spp::sparse_hash_map<std::cxx11::basic_string, std::cxx11::basic_string >::SetKey; EqualKey = std::equal_to<std::cxx11::basic_string >; Alloc = spp::libc_allocator<std::pair<const std::cxx11::basic_string, std::__cxx11::basic_string > >; spp::sparse_hashtable<Value, Key, HashFcn, ExtractKey, SetKey, EqualKey, Alloc>::size_type = long unsigned int]'
/home/iweber/Documents/Software/SalmonTools/include/sparsepp/spp.h:2881:26: required from 'bool spp::sparse_hashtable<Value, Key, HashFcn, ExtractKey, SetKey, EqualKey, Alloc>::_resize_delta(spp::sparse_hashtable<Value, Key, HashFcn, ExtractKey, SetKey, EqualKey, Alloc>::size_type) [with Value = std::pair<const std::cxx11::basic_string, std::cxx11::basic_string >; Key = std::cxx11::basic_string; HashFcn = spp::spp_hash<std::cxx11::basic_string >; ExtractKey = spp::sparse_hash_map<std::cxx11::basic_string, std::cxx11::basic_string >::SelectKey; SetKey = spp::sparse_hash_map<std::cxx11::basic_string, std::cxx11::basic_string >::SetKey; EqualKey = std::equal_to<std::cxx11::basic_string >; Alloc = spp::libc_allocator<std::pair<const std::cxx11::basic_string, std::__cxx11::basic_string > >; spp::sparse_hashtable<Value, Key, HashFcn, ExtractKey, SetKey, EqualKey, Alloc>::size_type = long unsigned int]'
/home/iweber/Documents/Software/SalmonTools/include/sparsepp/spp.h:3413:21: required from 'spp::sparse_hashtable<Value, Key, HashFcn, ExtractKey, SetKey, EqualKey, Alloc>::value_type& spp::sparse_hashtable<Value, Key, HashFcn, ExtractKey, SetKey, EqualKey, Alloc>::find_or_insert(const key_type&) [with DefaultValue = spp::sparse_hash_map<std::cxx11::basic_string, std::cxx11::basic_string >::DefaultValue; Value = std::pair<const std::cxx11::basic_string, std::cxx11::basic_string >; Key = std::cxx11::basic_string; HashFcn = spp::spp_hash<std::cxx11::basic_string >; ExtractKey = spp::sparse_hash_map<std::cxx11::basic_string, std::cxx11::basic_string >::SelectKey; SetKey = spp::sparse_hash_map<std::cxx11::basic_string, std::cxx11::basic_string >::SetKey; EqualKey = std::equal_to<std::cxx11::basic_string >; Alloc = spp::libc_allocator<std::pair<const std::cxx11::basic_string, std::cxx11::basic_string > >; spp::sparse_hashtable<Value, Key, HashFcn, ExtractKey, SetKey, EqualKey, Alloc>::value_type = std::pair<const std::cxx11::basic_string, std::cxx11::basic_string >; spp::sparse_hashtable<Value, Key, HashFcn, ExtractKey, SetKey, EqualKey, Alloc>::key_type = std::cxx11::basic_string]'
/home/iweber/Documents/Software/SalmonTools/include/sparsepp/spp.h:3933:57: required from 'spp::sparse_hash_map<Key, T, HashFcn, EqualKey, Alloc>::mapped_type& spp::sparse_hash_map<Key, T, HashFcn, EqualKey, Alloc>::operator[](const key_type&) [with Key = std::cxx11::basic_string; T = std::cxx11::basic_string; HashFcn = spp::spp_hash<std::cxx11::basic_string >; EqualKey = std::equal_to<std::cxx11::basic_string >; Alloc = spp::libc_allocator<std::pair<const std::cxx11::basic_string, std::cxx11::basic_string > >; spp::sparse_hash_map<Key, T, HashFcn, EqualKey, Alloc>::mapped_type = std::cxx11::basic_string; spp::sparse_hash_map<Key, T, HashFcn, EqualKey, Alloc>::key_type = std::cxx11::basic_string]'
/home/iweber/Documents/Software/SalmonTools/src/ExtractUnmapped.cpp:117:31: required from here
/home/iweber/Documents/Software/SalmonTools/include/sparsepp/spp.h:2214:23: warning: 'void* memcpy(void*, const void*, size_t)' writing to an object of type 'spp::sparsetable<std::pair<const std::cxx11::basic_string, std::cxx11::basic_string >, spp::libc_allocator<std::pair<const std::cxx11::basic_string, std::cxx11::basic_string > > >::group_type' {aka 'class spp::sparsegroup<std::pair<const std::cxx11::basic_string, std::cxx11::basic_string >, spp::libc_allocator<std::pair<const std::cxx11::basic_string, std::cxx11::basic_string > > >'} with no trivial copy-assignment; use copy-initialization instead [-Wclass-memaccess]
2214 | memcpy(first, _first_group, sizeof(*first*) (std::min)(sz, old_sz));
| ^~~~
~~In file included from /home/iweber/Documents/Software/SalmonTools/src/ExtractUnmapped.cpp:8:~~
~~/home/iweber/Documents/Software/SalmonTools/include/sparsepp/spp.h:1027:7: note: 'spp::sparsetable<std::pair<const std::cxx11::basic_string, std::cxx11::basic_string >, spp::libc_allocator<std::pair<const std::cxx11::basic_string, std::cxx11::basic_string > > >::group_type' {aka 'class~~

~~spp::sparsegroup<std::pair<const std::**cxx11::basic_string, std::**cxx11::basic_string >, spp::libc_allocator<std::pair<const std::**cxx11::basic_string, std::**cxx11::basic_string > > >'} declared here~~
~~1027 | class sparsegroup~~
~~| ^~~
[84%] Linking CXX static library libsalmon_tools_core.a
[84%] Built target salmon_tools_core
[92%] Building CXX object src/CMakeFiles/salmontools.dir/SalmonTools.cpp.o
/home/iweber/Documents/Software/SalmonTools/src/SalmonTools.cpp: In function ‘int main(int, char**)’:
/home/iweber/Documents/Software/SalmonTools/src/SalmonTools.cpp:39:18: warning: catching polymorphic type ‘class args::Help’ by value [-Wcatch-value=]
39 | catch (args::Help) {
| ^~~~
/home/iweber/Documents/Software/SalmonTools/src/SalmonTools.cpp:43:24: warning: catching polymorphic type ‘class args::Error’ by value [-Wcatch-value=]
43 | catch (args::Error e) {
| ^
[100%] Linking CXX executable salmontools
[100%] Built target salmontools
"

Build decoys

/home/iweber/Documents/Software/MashMap/build/bin/

Building the decoy-aware transcriptome index for *C. elegans*

I found this excellent [RNA-seq analysis course](#)from the Harvard Chan Bioinformatics Core ([Zenodo](#)). In the [Salmon lesson](#) , they explain how to set the parameters:

"

```
$ salmon index \
-t /n/groups/hbctraining/rna-seq_2019_02/reference_data/Homo_sapiens.GRCh38.cdna.all.fa \
-i salmon_index \
-k 31
```

- `-t` **:** the path to the transcriptome file (in FASTA format)
- `-i` **:** the path to the folder to store the indices generated
- `-k` **:** the length of kmer to use to create the indices (will output all sequences in transcriptome of length k) - **NOTE:** Default kmer size for salmon is -k 31, so we do not need to include the `-k` parameter in the index command. However, the kmer default of 31 is optimized for read sizes of 75bp or longer and if your reads are shorter, you will want a smaller kmer (kmer size needs to be an odd number).
  "

Sample sheet and contrasts sheet

I, of course, still need the sample sheet and the contrast sheet for this experiment as well. I have reads from two samples, CHS_1063 and CHS_1109. I tried searching for their names to see what kind of experiment they stemmed from and couldn't find any info on the quick, so I will simply assume that CHS_1063 is "control" and CHS_1109 is "treatment".

Here's how the sample sheet looks like:

```
sample,fastq_1,fastq_2,strandedness,condition
CONTROL_REP1,CHS_1063_R1.fastq.gz,CHS_1063_R2.fastq.gz,reverse,CONTROL
TREATMENT_REP1,CHS_1109_R1.fastq.gz,CHS_1109_R1.fastq.gz,reverse,TREATMENT
```

And the contrasts sheet:

```
contrast,treatment,control
TREATMENT_CONTROL,TREATMENT,CONTROL
```

Pipeline parameters for C. elegans

To run it for the C. elegans data, I will use

```
#!/bin/bash

nextflow run nf-core/rnasplice -r 1.0.4

--input samplesheet_C_elegans.csv
--contrasts contrastssheet_C_elegans.csv

--fasta /mnt/mnt-win-ubu-shared/Win_Ubuntu_shared/Genomes/genome_C_elegans/GCF_000002985.6_WBcel235_genomic.fna
~~--gtf /mnt/mnt-win-ubu-shared/Win_Ubuntu_shared/Genomes/genome_C_elegans/Caenorhabditis_elegans.WBcel235.112.gtf~~ NO. USE
GFF INSTEAD.
--gff /mnt/mnt-win-ubu-shared/Win_Ubuntu_shared/Genomes/genome_C_elegans/

--save-trimmed true # to save the reads after trimming

--aligner star_salmon
--star_index path/to/index
--salmon_index path/to/index

--save-unaligned true # to save, if possible, any reads that couldn't be aligned in the results directory for later inspection
--save-align-intermeds # to save BAM files separately

--rmats true to
--outdir pipeline_test_results
-profile docker
```

```
nextflow run nf-core/rnasplice -r 1.0.4
--input samplesheet.csv
--contrasts contrastsheet.csv
--genome WBcel235 OR --fasta? as path to genome and --gtf as path to annotation file?
--outdir my/result/directory


--aligner star_salmon
--star_index /path/to/star/index
--save_align_intermeds # needed if wanting to use rMATS, because otherwise pipeline does not save SAM files, which rMATS needs
--save_reference # saves anything the pipeline downloads by itself, e.g. STAR indices

--salmon_quant_libtype # this might need to go into Salmon's chunk in the workflow AND is optional - Salmon can infer this
automatically from the samplesheet information

--gtf /path/to/annotations.gtf

-profile docker
```

ACTIVELY WORKING ON

Trimming the reads

After MultiQC aggregated the FastQC results, I wanted to look specifically if anything needed trimming. I installed Trimmomatic from conda with `conda install bioconda::trimmomatic`

Then, I wrote a tiny script to run Trimmomatic automatically over all sequences and then have FastQC go over the now-trimmed sequences, plus MultiQC to aggregate the results:
[adapter trimming? check MultiQC!]

```
`!#/bin/bash

`trimmomatic PE -phred33 *.fastq -baseout "output" LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15 MINLEN:3

`fastqc output_* --threads 8 --memory 40000 --outdir FastQC_results

`multiqc FastQC_results/ --outdir MultiQC_results/`
```

I found out in another analysis that one can create at most 4 threads on an octo-core PC like mine[source with explanation], and that trimmomatic has some restriction that only allows the use of a max of 10 GB of RAM, and not 40, as I had indicated to it here, so I adjusted the parameters for this case. [can max RAM be overridden with some option of trimmomatic?]

Title

Contents
Since the reads all had issues up to around the 10th base from the 5' end, I set the

Installing R and RStudio on the virtual machine

From https://cran.r-project.org/

```
# update indices
sudo apt update -qq
# install two helper packages we need
sudo apt install --no-install-recommends software-properties-common dirmngr
# add the signing key (by Michael Rutter) for these repos
# To verify key, run gpg --show-keys /etc/apt/trusted.gpg.d/cran_ubuntu_key.asc
# Fingerprint: E298A3A825C0D65DFD57CBB651716619E084DAB9
wget -qO- https://cloud.r-project.org/bin/linux/ubuntu/marutter_pubkey.asc | sudo tee -a
/etc/apt/trusted.gpg.d/cran_ubuntu_key.asc
# add the R 4.0 repo from CRAN -- adjust 'focal' to 'groovy' or 'bionic' as needed
sudo add-apt-repository "deb https://cloud.r-project.org/bin/linux/ubuntu $(lsb_release -cs)-cran40/"
```

License