**Tutorial on next generation sequencing (NGS) data analysis with emphasis on RNA sequencing (RNA-seq) technique**

**Prepared by :**

**Kennedy …, MSc student JKUAT**

**Mtakai Ngara, PhD**

**April 6, 2022**

**Scope**

· Bioinformatics background and resources

· Next-generation sequencing (NGS) concepts and applications

· Gene expression profiling using RNA-sequencing technique

· Basic command line for unix/linux users

· Conda environment installation and basics

· NGS datasets

· Read trimming

· Quality control (QC) analysis

· Reference mapping

· Mapping quality evaluation

· Gene expression quantification

· Differential expression analysis

· Basic visualisation

**Preparations:**

* Stress test the internet download speeds at MKU using parallel download function

**Basic command line (CLI) and conda package manager**

**Basic command line (CLI)**

What is a shell?

Wikipedia:

“In computing, a shell is a user interface for access to an operating system’s services. In general, operating system shells use either a command-line interface (CLI) or graphical user interface (GUI), depending on a computer’s role and particular operation…”

“CLI shells allow some operations to be performed faster in some situations, especially when a proper GUI has not been or cannot be created. However, they require the user to memorize all commands and their calling syntax, and also to learn the shell-specific scripting language, for example bash script.”

Some words regarding the Unix/Linux filesystem:

* The directory structure in a Linux system is not much different from any other system you worked with, e.g., Windows, MacOSX.
* However, on the command-line we navigate via commands and not via mouse clicks. Why is it necessary to use the command-line in the first place? Strictly speaking it is not, if you do not want to make use of programs on the command-line. However, the power of the Linux system becomes only obvious once we learn to make use of the command-line, thus navigating the directory structure via commands is one of the most important skills for you to learn.
* Examples of Linux environments: Ubuntu, etc.

**Open a terminal**

Open a terminal window and you are are ready to go.

On your linux desktop find: System Tools –> QTerminal (for LXDE environment) or type “Terminal” in the search box (Fig. ##).

**Proxy settings**

You might encounter problems connecting to the internet due to proxy settings at your institute..

The Unix shell for novice:

<https://swcarpentry.github.io/shell-novice/>

<https://amfdrey.github.io/2022-04-04-NiTheCS-Online/>

<https://software-carpentry.org/>

**$ conda update --yes conda**

**Installing conda channels to make tools available -** A channel is where conda looks for packages

**# Install some conda channels**

**$ conda config --add channels defaults**

**$ conda config --add channels bioconda**

**$ conda config --add channels conda-forge**

**Creating environments**

**$ conda create -n ngs python=3**

**# activate the environment**

**$ conda activate ngs**

**Install modules (software)**

**# Install more tools into the environment**

**$ conda install package**

**General conda commands**

**# to search for packages**

**$ conda search [package name]**

**# To update all packages**

**$ conda update --all –yes**

**# List all packages installed**

**$ conda list [-n env]**

**# conda list environments**

**$ conda env list**

**# create new env**

**$ conda create -n [name] package [package] ...**

**# activate env**

**$ conda activate [name]**

**# deactivate env**

**$ conda deactivate**

click download - python based. Download for mac users.

**Installing conda package manager on Mac OS**

To add….

**QC analysis of NGS data analysis**

**QC process**

There are a few steps one need to do when getting the raw sequencing data from the sequencing facility:

1. Remove PhiX sequences (we are not going to do this)

2. Adapter trimming

3. Quality trimming of reads

4. Quality assessment

**Get the data**

**# How about knowing the current directory before mkdir?**

**#printing working directory**

**Pwd**

**# create a working directory**

**$ mkdir analysis**

**# change into the directory**

**$ cd analysis**

**# download the data from SRA Explorer using the accession ‘PRJEB31541’ from in search field and select all the runs to the collection and download the ftp links. You should have a text file named: ‘sra\_explorer\_fastq\_urls.txt’.**

**For those using terminal:**

**#create a directory for the data**

**$ mkdir data**

**# change into the directory**

**$ cd analysis**

**$wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR319/004/ERR3197114/ERR3197114\_1.fastq.gz**

**$wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR319/004/ERR3197114/ERR3197114\_2.fastq.gz**

**$wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR319/003/ERR3197113/ERR3197113\_1.fastq.gz**

**$wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR319/003/ERR3197113/ERR3197113\_2.fastq.gz**

**$wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR319/002/ERR3197112/ERR3197112\_1.fastq.gz**

**$wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR319/002/ERR3197112/ERR3197112\_2.fastq.gz**

**For those having Windows:**

**Pasting the ftp links in your browser would not download the fastq files. Proceed as follows**

**>** Search SRA database > enter the accession number e.g. ERR3197114 > Click on Runs (your accession number) > click on data access (menu) - download your first file reads ~ 12Gbs under the Original file table.

> Use basic **mv** command to move your file to your working directory

Repeat the steps forl the other two files

**NB:** Should the download fail, what options can we provide.

**NB:** Test the data download in parallel to see if internet speeds are sufficient at MKU

Perhaps we can use a smaller dataset - this will not work unless we **have a server**

Maybe we could you use bam file from a different study for illustration - **for this study not available**

**NB:** Provide instruction on the dataset(s) to download in advance

The data is from a paired-end sequencing run data (see Fig. ##) from an Illumina HiSeq ## (ref ##). Thus, we have two files, one for each end of the read.

**Investigate the data**

1. Use the command-line to get some ideas about the file.

2. What kind of files are we dealing with?

3. How many sequence reads are in the file?

4. Assume a genome size of ~## MB. Calculate coverage based on this formula: C = LN / G

• C: Coverage

• G: is the haploid genome length in bp

• L: is the read length in bp (e.g., 2xL## paired-end = ##)

• N: is the number of reads sequenced

**The fastq file format**

The data we receive from the sequencing is in fastq format. Explain (##) fastq format.

A useful tool to decode base qualities can be found here:<http://broadinstitute.github.io/picard/explain-qualities.html>

**QC process**

**PhiX genome**

PhiX is a nontailed bacteriophage with a single-stranded DNA and a genome with 5386 nucleotides. PhiX is used as a quality and calibration control for sequencing runs. PhiX is often added at a low known concentration, spiked in the same lane along with the sample or used as a separate lane. As the concentration of the genome is known, one can calibrate the instruments. Thus, PhiX genomic sequences need to be removed before processing your data further as this constitutes a deliberate contamination. The steps involve mapping all reads to the “known” PhiX genome and removing all of those sequence reads from the data.

However, your sequencing provider might not have used PhiX, thus you need to read the protocol carefully, or just do this step in any case.

**Adapter trimming**

The process of sequencing DNA via Illumina technology requires the addition of some adapters to the sequences. These get sequenced as well and need to be removed as they are artificial and do not belong to the species we try to sequence. We have to deal with a trade-off between accuracy of adapter removal and speed of the process. Adapter trimming does take some time.

Also, we have generally two different approaches when trimming adapter:

1. We can use a tool that takes an adapter or list of adapters and removes these from each sequence read.

2. We can use a tool that predicts adapters and removes them from each sequence read.

For the first approach we need to know the adapter sequences that were used during the sequencing of our samples. Normally, you should ask your sequencing provider, who should be providing this information to you. Illumina itself provides a document that describes the adapters used for their different technologies. Also, tools such as FastQC tool, provides a collection of contaminants and adapters.

Here, we are going to use the second approach with a tool called fastp to trim adapters and do quality trimming. fastp has a few characteristics which make it a great tool, most importantly: it is pretty fast, provides good information after the run, and can do quality trimming as well, thus saving us to use another tool to do this.

Quality trimming of our sequencing reads will remove bad quality called bases from our reads.

**# create env and install tools -** Perhaps you can give an explanation as to why we have to create an environment when we can use the base - possible reason is overshadowing.

Basic command: conda create [environment name]

**$ conda create --yes -n qc fastp fastqc multiqc** # How about creating an env > activate > install the necessary tools for us as starters?

**# activate env**

**$ conda activate qc**

Here, as an example we are trimming the sequence reads:

**$ mkdir trimmed**

**$ fastp --detect\_adapter\_for\_pe**

**--overrepresentation\_analysis**

**--correction --cut\_right --thread 2**

**--html trimmed/anc.fastp.html --json trimmed/anc.fastp.json**

**-i data/##\_R1.fastq.gz -I data/##\_R2.fastq.gz**

**-o trimmed/##\_R1.fastq.gz -O trimmed/##\_R2.fastq.gz**

## is the path to your files

· **--detect\_adapter\_for\_pe**: Specifies that we are dealing with paired-end data.

· **--overrepresentation\_analysis**: Analyse the sequence collection for sequences that appear too often.

· **--correction**: Will try to correct bases based on an overlap analysis of read1 and read2.

· **--cut\_right**: Will use quality trimming and scan the read from start to end in a window. If the quality in the window is below what is required, the window plus all sequence towards the end is discarded and the read is kept if its still long enough.

· **--thread**: Specify how many concurrent threads the process can use.

· **--html** and **--json**: We specify the location of some stat files.

· **-i** data/anc\_R1.fastq.gz **-I** data/anc\_R2.fastq.gz: Specifies the two input read files

· **-o** trimmed/anc\_R1.fastq.gz **-O** trimmed/##\_R2.fastq.gz: Specifies the two desired output read files

**Quality assessment of sequencing reads**

**FastQC**

FastQC (## ref) is a very simple program to run that provides information about sequence read quality. It aims to provide a simple way to do some quality control checks on raw sequence data coming from high throughput sequencing pipelines. It provides a modular set of analyses which you can use to give a quick impression of whether your data has any problems of which you should be aware before doing any further analysis.

**$ fastqc --help**

The basic command:

**$ fastqc -o RESULT-DIR INPUT-FILE.fq (.gz) …**

· -o RESULT-DIR is the directory where the result files will be written

· INPUT-FILE.fq is the sequence file to analyse, can be more than one file

**$ mkdir fastqc\_report**

**$ fastqc -o fastqc\_report data/anc\_R1.fastq.gz**

NB: The result will be a HTML page per input file that can be opened in a web-browser.

For some help on FastQC results and interpretation see the links:

<http://www.bioinformatics.babraham.ac.uk/projects/fastqc/>

<http://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/>

**MultiQC**

MultiQC (## refs) is an excellent tool to put FastQC (and other tool) results of different samples into context. It compiles all FastQC (## ref) results and fastp stats into one nice webpage.

The use of MultiQC (## ref) is simple. Just provide the command with directories where multiple results are stored and it will compile a nice report, for example:

**$ multiqc DIRECTORY DIRECTORY ...**

Run FastQC and MultiQC on the trimmed data

To do:

1. Create a directory for the results –> trimmed-fastqc

2. Run FastQC on all trimmed files.

3. Visit the FastQC (## ref) website and read about sequencing QC reports for good and bad Illumina (## ref) sequencing runs.

4. Run MultiQC (## ref) on the trimmed-fastqc and trimmed directories

5. Compare your results to these examples (Fig. ## to Fig. ##) of a particularly bad run (taken from the FastQC (## ref) website) and write down your observations with regards to your data.

6. What elements in these example figures (Fig. ## to Fig. ##) indicate that the example is from a bad run?

**Read mapping**

Here we will use CLI to map our reads from the downloaded fastq files to the reference genome.

After studying this section, you should be able to:

1. Explain the process of sequence read mapping.

2. Use bioinformatics tools to map sequencing reads to a reference genome.

3. Filter mapped reads based on quality.

Create a directory for read mapping:

**$ cd ~/analysis**

**# Create a mapping result directory**

**$ mkdir mappings**

**$ ls -1F**

**$ cd ~/analysis**

**$ wget -O trimmed.tar.gz https://osf.io/m3wpr/download**

**$ tar xvzf trimmed.tar.gz**

**$ wget -O assembly.tar.gz https://osf.io/t2zpm/download**

**$ tar xvzf assembly.tar.gz**

**Mapping sequence reads to a reference genome**

Use the quality trimmed forward and backward DNA sequences and use a program called BWA (## ref) to map the reads.

**Installing read mapping software**

**$ conda create --yes -n mapping samtools bwa qualimap r-base**

**$ conda activate mapping**

**BWA overview**

BWA (### ref) is a short read aligner, that can take a reference genome and map single- or paired-end sequence data to it (### ref). It requires an indexing step in which one supplies the reference genome and BWA (### ref) will create an index that in the subsequent steps will be used for aligning the reads to the reference genome. While this step can take some time, the good thing is the index can be reused over and over. The general command structure of the BWA (### ref) tools we are going to use are shown below:

**# bwa index help**

**$ bwa index**

**# indexing**

**$ bwa index path/to/reference-genome.fa**

**# bwa mem help**

**$ bwa mem**

**# single-end mapping, general command structure, adjust to your case**

**$ bwa mem path/to/reference-genome.fa path/to/reads.fq.gz > path/to/aln-se.sam**

**# paired-end mapping, general command structure, adjust to your case**

**$ bwa mem path/to/reference-genome.fa path/to/read1.fq.gz path/to/read2.fq.gz > path/to/aln-pe.sam**

**Creating a reference index for mapping**

To do: Create an BWA (### ref) index for our reference genome assembly.

Attention! Remember which file you need to submit to BWA (### ref).

Note: Should you be unable to run BWA (## ref) indexing on the data, you can download the index from Downloads (##). Unarchive and uncompress the files with tar -xvzf bwa-index.tar.gz.

**Mapping reads in a paired-end manner**

Now that we have created our index, it is time to map the trimmed sequencing reads to the reference genome.

To do: Use the correct bwa mem command structure from above and map the reads of the two evolved line to the reference genome.

**The sam mapping file-format**

BWA (### ref), like most mappers, will produce a mapping file in sam-format. Have a look into the sam-file that was created by either program. A quick overview of the sam-format can be found here (### ref) and even more information can be found here (## ref). Briefly, first there are a lot of header lines. Then, for each read, that mapped to the reference, there is one line.

The columns of such a line in the mapping file are described in Table ##.

One line of a mapped read can be seen here ##:

It defines the read and the position within the reference genome, where the read mapped and a quality of the mapping.

**Mapping post-processing**

**Fix mates and compress**

Because aligners can sometimes leave unusual SAM flag (### ref) information on SAM records, it is helpful when working with many tools to first clean up read pairing information and flags with SAM tools (## ref). We are going to produce also compressed bam output for efficient storing of and access to the mapped reads. Note, samtools fixmate expects name-sorted input files, which we can achieve with samtools sort -n.

**$ samtools sort -n -O sam mappings/##.sam | samtools fixmate -m -O bam -mappings/##.fixmate.bam**

· -m: Add ms (mate score) tags. These are used by markdup (below) to select the best reads to keep

· -O bam: specifies that we want compressed bam output from fixmate

Attention: The step of sam to bam -file conversion might take a few minutes to finish, depending on how big your mapping file is.

A very useful tool to explain flags can be found here:<http://broadinstitute.github.io/picard/explain-flags.html>

Once we have bam-file, we can also delete the original sam-file as it requires too much space and we can always recreate it from the bam-file.

$ rm mappings/##.sam

**Sorting**

**# Convert to bam file and sort**

**$ samtools sort -O bam -o mappings/##.sorted.bam mappings/##.fixmate.bam**

**# Once it successfully finished, delete the fixmate file to save space**

**$ rm mappings/##. fixmate.bam**

**Remove duplicates**

In this step we remove duplicate reads. The main purpose of removing duplicates is to mitigate the effects of PCR amplification bias introduced during library construction.

**$ samtools markdup -r -S mappings/##.sorted.bam mappings/##.sorted.dedup.bam**

**# if it worked, delete the original file**

**$ rm mappings/##.sorted.bam**

**Mapping statistics**

Mapping statistics with SAMtools

**$ samtools flagstat mappings/##.sorted.dedup.bam**

To do: Look at the mapping statistics and understand their meaning. Discuss your results. Explain why we may find mapped reads that have their mate mapped to a different chromosome/contig? Can they be used for something?

For the sorted bam-file we can get read depth for at all positions of the reference genome, e.g. how many reads are overlapping the genomic position.

**$ samtools depth mappings/evol1.sorted.dedup.bam | gzip > mappings/evol1.depth.txt.gz**

Todo: Extract the depth values for chromosome ## and load the data into R, calculate some statistics of our scaffold.

Now we quickly use some R (ref ##) to make a coverage plot for chromosome ##. Open a R shell by typing R on the command-line of the shell.

**$ zcat mappings/##.depth.txt.gz | egrep '^NODE\_##\_' | gzip > mappings/NODE\_##.depth.txt.gz**

x <- read.table('mappings/NODE\_##.depth.txt.gz', sep='\t', header=FALSE, strip.white=TRUE) # Look at the beginning of x

head(x)

# calculate average depth

mean(x[,3])

# std dev

sqrt(var(x[,3]))

# mark areas that have a coverage below 20 in red

plot(x[,2], x[,3], col = ifelse(x[,3] < 20,'red','black'), pch=19, xlab='postion', ylab='coverage')

# to save a plot

png('mappings/covNODE##.png', width = 1200, height = 500)

plot(x[,2], x[,3], col = ifelse(x[,3] < 20,'red','black'), pch=19, xlab='postion', ylab='coverage')

dev.off()

The result plot will be looking similar to the one in Fig. ##.

To do: Look at the created plot. Explain why it makes sense that you find relatively bad coverage at the beginning and the end of the contig.

**Statistics with QualiMap**

For a more in-depth analysis of the mappings, one can use QualiMap (ref##).

QualiMap examines sequencing alignment data in SAM/BAM files according to the features of the mapped reads and provides an overall view of the data that helps to the detect biases in the sequencing and/or mapping of the data and eases decision-making for further analysis.

Run QualiMap with:

**$ qualimap bamqc -bam mappings/##.sorted.dedup.bam**

**# Once finsished open reult page with**

**$ firefox mappings/##.sorted.dedup\_stats/qualimapReport.html**

To do: Investigate the mapping of the sample. Write down your observations.

**Sub-selecting reads**

The mapping commands we used above, without additional parameters to sub-select specific alignments (e.g. for Bowtie2 (## ref) there are options like --no-mixed, which suppresses unpaired alignments for paired reads or --no-discordant, which suppresses discordant alignments for paired reads, etc.), are going to output all reads, including unmapped reads, multi-mapping reads, unpaired reads, discordant read pairs, etc. in one file. We can sub-select from the output reads we want to analyse further using SAMtools (## ref).

To do: Explain what concordant and discordant read pairs are?

We can select read-pair that have been mapped in a correct manner (same chromosome/contig, correct orientation to each other, distance between reads are sensible).

**$ samtools view -h -b -f 3 mappings/##.sorted.dedup.bam > mappings/##.sorted.dedup.concordant.bam**

-b: Output will be bam-format

-f 3: Only extract correctly paired reads

-f extracts alignments with the specified SAM flag (ref ##) set

**Quality-based sub-selection**

In this section we want to sub-select reads based on the quality of the mapping. It seems a reasonable idea to only keep good mapping reads. As the SAM-format contains at column 5 the 𝑀𝐴𝑃𝑄 value, which we established earlier is the “MAPping Quality” in Phred-scaled, this seems easily achieved. The formula to calculate the 𝑀 𝐴𝑃 𝑄 value is: 𝑀 𝐴𝑃 𝑄 = −10 \* 𝑙𝑜𝑔10(𝑝), where 𝑝 is the probability that the read is mapped wrongly. However, there is a problem! While the MAPQ information would be very helpful indeed, the way that various tools implement this value differs. A good overview can be found here (<https://sequencing.qcfail.com/articles/mapq-values-are-really-useful-but-their-implementation-is-a-mess/>). Bottom-line is that we need to be aware that different tools use this value in different ways and the it is good to know the information that is encoded in the value. Once you dig deeper into the mechanics of the 𝑀 𝐴𝑃 𝑄 implementation it becomes clear that this is not an easy topic. If you want to know more about the 𝑀 𝐴𝑃 𝑄 topic, please follow the link above.

**Unmapped reads**

We can use Kraken2 (<https://www.ccb.jhu.edu/software/kraken2/>) to classify all un- mapped sequence reads and identify the species they are coming from and test for contamination.

Command for getting unmmaped reads

**$samtools view -b -f 4 mappings/##.sorted.dedup.bam > mappings/##.sorted.unmapped.bam**

**# we are deleting the original to save space,**

**# however, in reality you might want to save it to investigate later**

**$ rm mappings/##.sorted.dedup.bam**

**# count the unmapped reads**

**$ samtools view -c mappings/##.sorted.unmapped.bam**

• -b: indicates that the output is BAM.

• -f INT: only include reads with this SAM flag (## ref) set.

• -c: count the reads.

**$ samtools fastq -1 mappings/evol1.sorted.unmapped.R1.fastq.gz -2 mappings/evol1.sorted.unmapped.R2.fastq.gz mappings/##.sorted.unmapped.bam**

**# delete not needed files**

**$ rm mappings/##.sorted.unmapped.bam**

**Contamination investigation**

We want to investigate if there are sequences of other species in our collection of sequenced DNA pieces. This might be a way of quality control, e.g. have the samples been contaminated?

We will use the tool Kraken2 (###ref) to assign taxonomic classifications to our sequence reads. This tool uses k-mers to assign a taxonomic labels in form of NCBI Taxonomy (## ref) to the sequence (if possible). The taxonomic label is assigned based on similar k-mer content of the sequence in question to the k-mer content of reference genome sequence. The result is a classification of the sequence in question to the most likely taxonomic label. If the k-mer content is not similar to any genomic sequence in the database used, it will not assign any taxonomic label.

**Installing kraken**

**$ conda create --yes -n kraken kraken2 bracken**

**$ conda activate kraken**

**# make sure you are in your analysis root folder**

**$ cd ~/analysis**

**# create dir**

**$ mkdir kraken**

**$ cd kraken**

Create or download a Kraken2 (## ref) database that can be used to assign the taxonomic labels to sequences.

**$ curl -O ftp://ftp.ccb.jhu.edu/pub/data/kraken2\_dbs/minikraken2\_v2\_8GB\_201904\_UPDATE.tgz**

**# alternatively we can use wget**

**$ wget ftp://ftp.ccb.jhu.edu/pub/data/kraken2\_dbs/minikraken2\_v2\_8GB\_201904\_UPDATE.tgz**

**# once the download is finished, we need to extract the archive content:**

**$ tar -xvzf minikraken2\_v2\_8GB\_201904\_UPDATE.tgz**

Note: The “minikraken2” database was created from bacteria, viral and archaea sequences.

**Running Kraken2**

Now that we have installed Kraken2 (##ref) and downloaded and extracted the minikraken2 database, we can attempt to investigate the sequences we got back from the sequencing provider for other species as the one it should contain. We call the Kraken2 (###ref) tool and specify the database and fasta-file with the sequences it should use. The general command structure looks like this:

$ kraken2 --use-names --threads 4 --db PATH\_TO\_DB\_DIR --report example.report.txt example.fa >example.kraken

Each sequence classified by Kraken2 (## ref) results in a single line of output. Output lines contain five tab- delimited fields; from left to right, they are:

1. C/U: one letter code indicating that the sequence was either classified or unclassified.
2. The sequence ID, obtained from the FASTA/FASTQ header.
3. The taxonomy ID Kraken2 (## ref) used to label the sequence; this is 0 if the sequence is unclassified and otherwise should be the NCBI Taxonomy (##ref) identifier.
4. The length of the sequence in bp.
5. A space-delimited list indicating the lowest common ancestor (in the taxonomic tree) mapping of each k-mer in the sequence. For example, 562:13 561:4 A:31 0:1 562:3 would indicate that:

· the first 13 k-mers mapped to taxonomy ID #562

· the next 4 k-mers mapped to taxonomy ID #561

· the next 31 k-mers contained an ambiguous nucleotide

· the next k-mer was not in the database

· the last 3 k-mers mapped to taxonomy ID #562

**Investigate taxa**

We can use the webpage NCBI TaxIdentifier (ref ##) to quickly get the names to the taxonomy identifier. However, this is impractical as we are dealing potentially with many sequences. Kraken2 (ref ##) has some scripts that help us understand our results better.

Because we used the Kraken2 (ref ###) switch --report FILE, we have also got a sample-wide report of all taxa found. This is much better to get an overview of what was found.

The output of kraken-report is tab-delimited, with one line per taxon. The fields of the output, from left-to-right, are as follows:

1. Percentage of reads covered by the clade rooted at this taxon

2. Number of reads covered by the clade rooted at this taxon

3. Number of reads assigned directly to this taxon

4. A rank code, indicating (U)nclassified, (D)omain, (K)ingdom, (P)hylum, (C)lass, (O)rder, (F)amily, (G)enus, or (S)pecies. All other ranks are simply “-“.

5. NCBI Taxonomy (##ref) ID

6. The indented scientific name

**Installing Ubuntu on Virtual Box in Windows**

1. *Install Virtual Box*

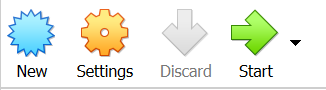
Download Virtual Box from this source: <https://www.virtualbox.org/wiki/Downloads>. Click on the downloaded executable file and follow the prompts to install Virtual Box on your Windows laptop.

2. *Download the Ubuntu ISO file*

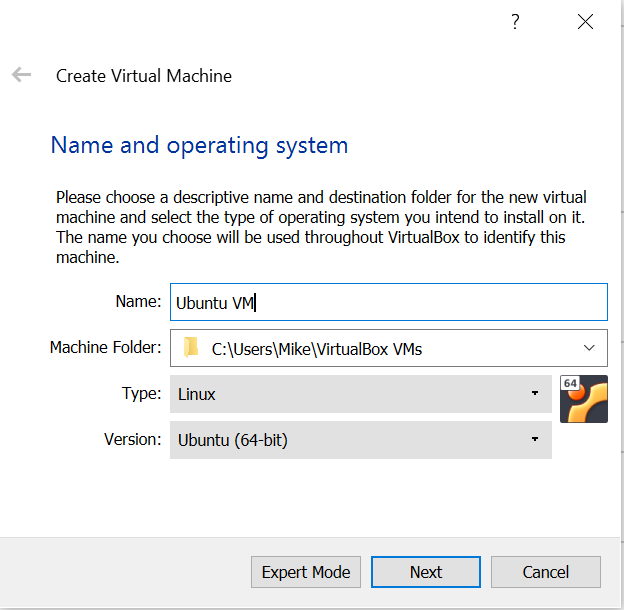
This may take some time because the file is relatively large (about 3 GB). The ISO file can be downloaded from this source: <https://ubuntu.com/download/desktop>. Click on the “Download” button beside the Ubuntu 20.04.4 LTS version to download.

3. *Begin the installation of Ubuntu on Virtual Box*

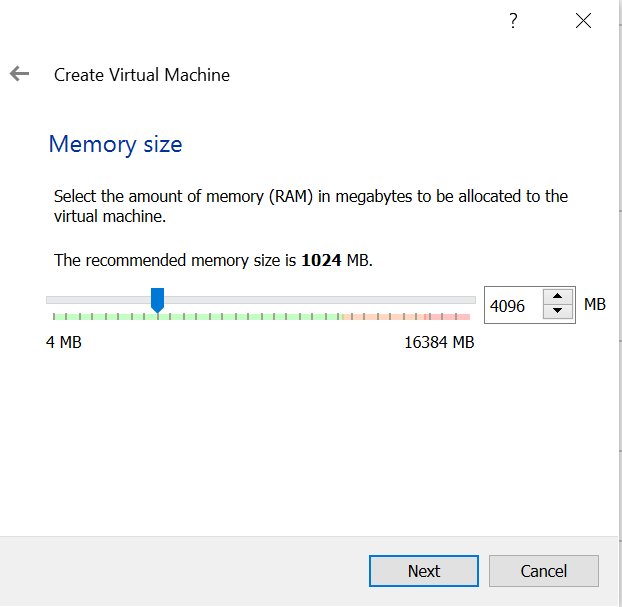
Click on the Virtual Box icon to launch Virtual Box and click the “New” button at the top of VirtualBox Manager.



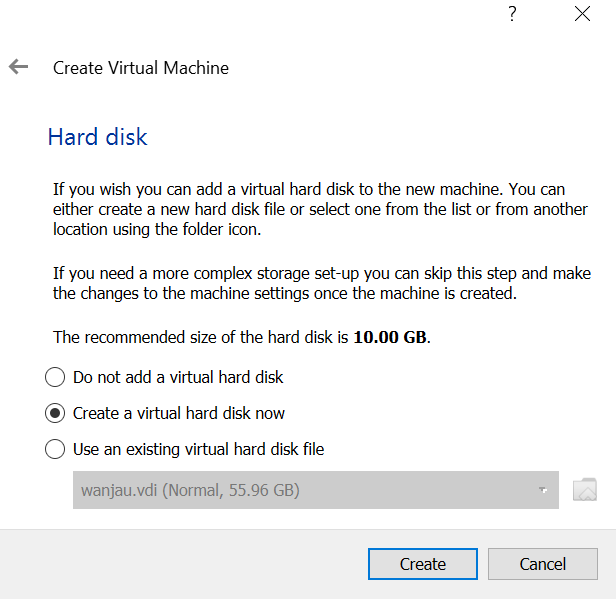
In the pop up window, enter the name of the virtual machine, for example “Ubuntu VM”. Choose the type of operating system as “Linux” and lastly, select the 64 bit version of Ubuntu Linux from the dropdown menu.



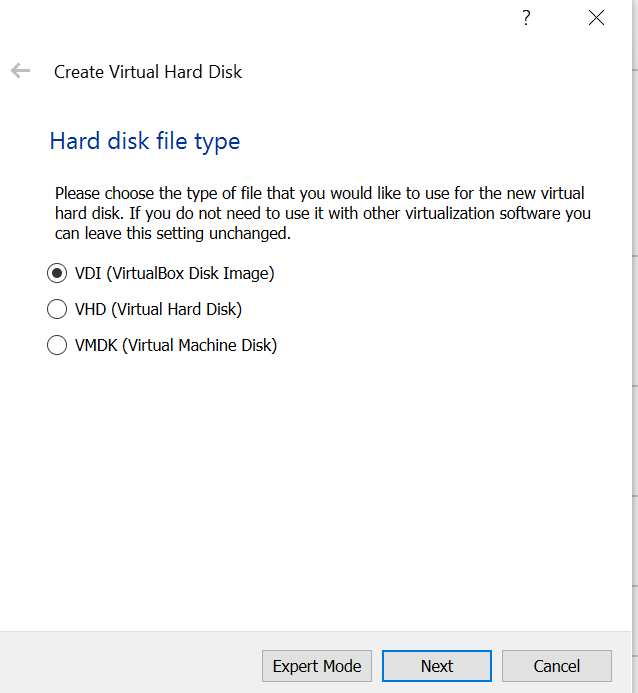
Click on “Next” and select the memory size. Depending on the RAM of your machine, assign an appropriate size of RAM to your virtual machine such that it will not make your Windows OS run slow. For example, for a Windows machine with 8GB RAM, you may assign your virtual machine 4GB of RAM.



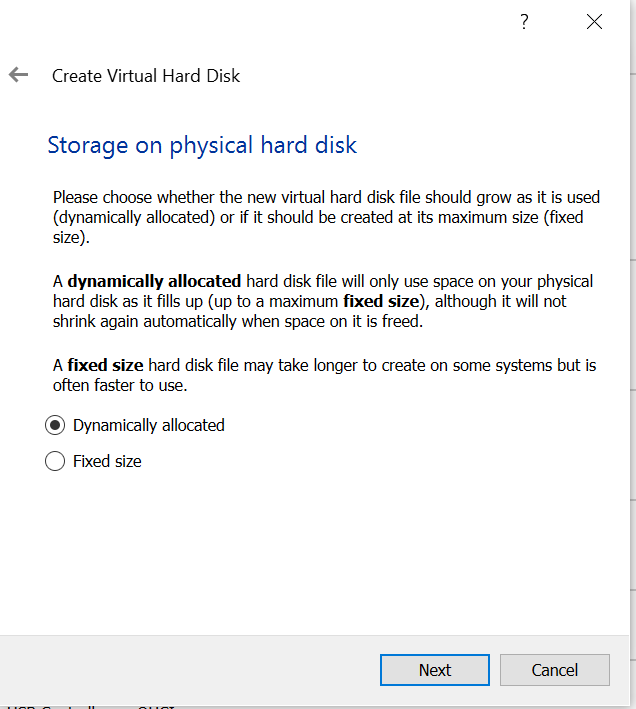
Next, allocate hard drive disk space (virtual drive) for your virtual machine. About 60GB of disk space will be enough for a machine with 500GB of disk space. Just be sure that you have enough space on your actual hard drive to handle the size of your virtual drive. Click “Create” to create the virtual disk space.



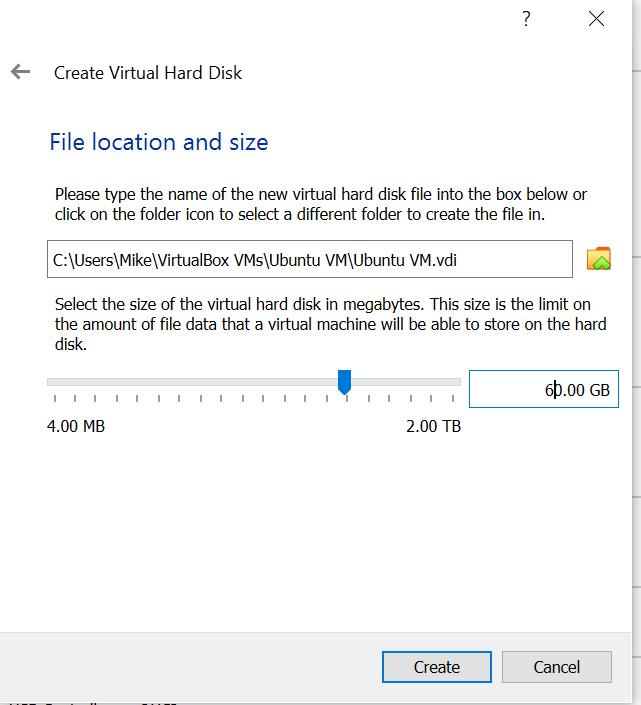
On the next pop-up window, select “VDI (VirtualBox Disk Image)” and click “Next”.



Select “Dynamically allocated” and click “Next”



Assign an appropriate virtual disk space for the Ubuntu virtual machine and click “Create”.

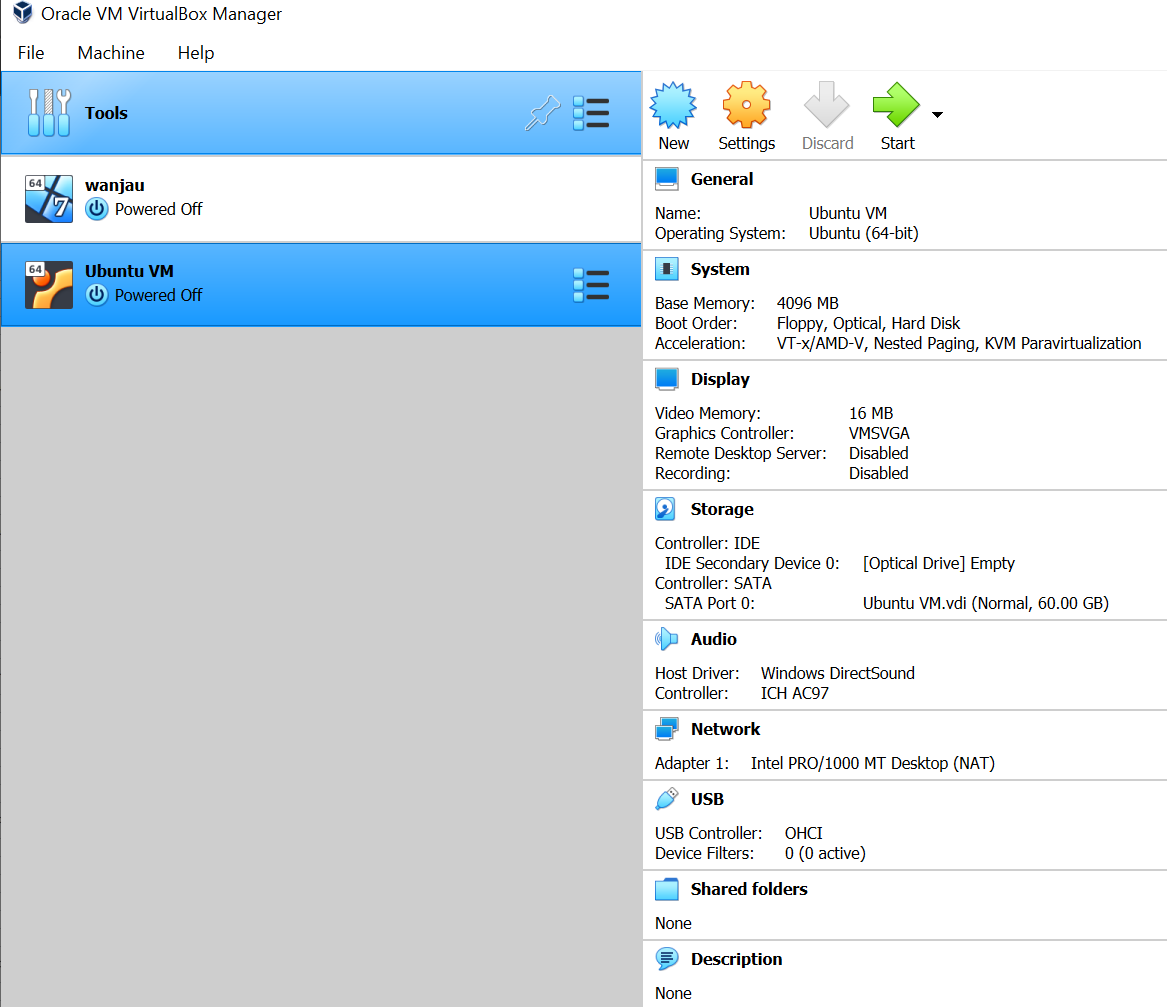


You will see a Virtual machine called “Ubuntu VM” or the name you gave to your virtual machine.

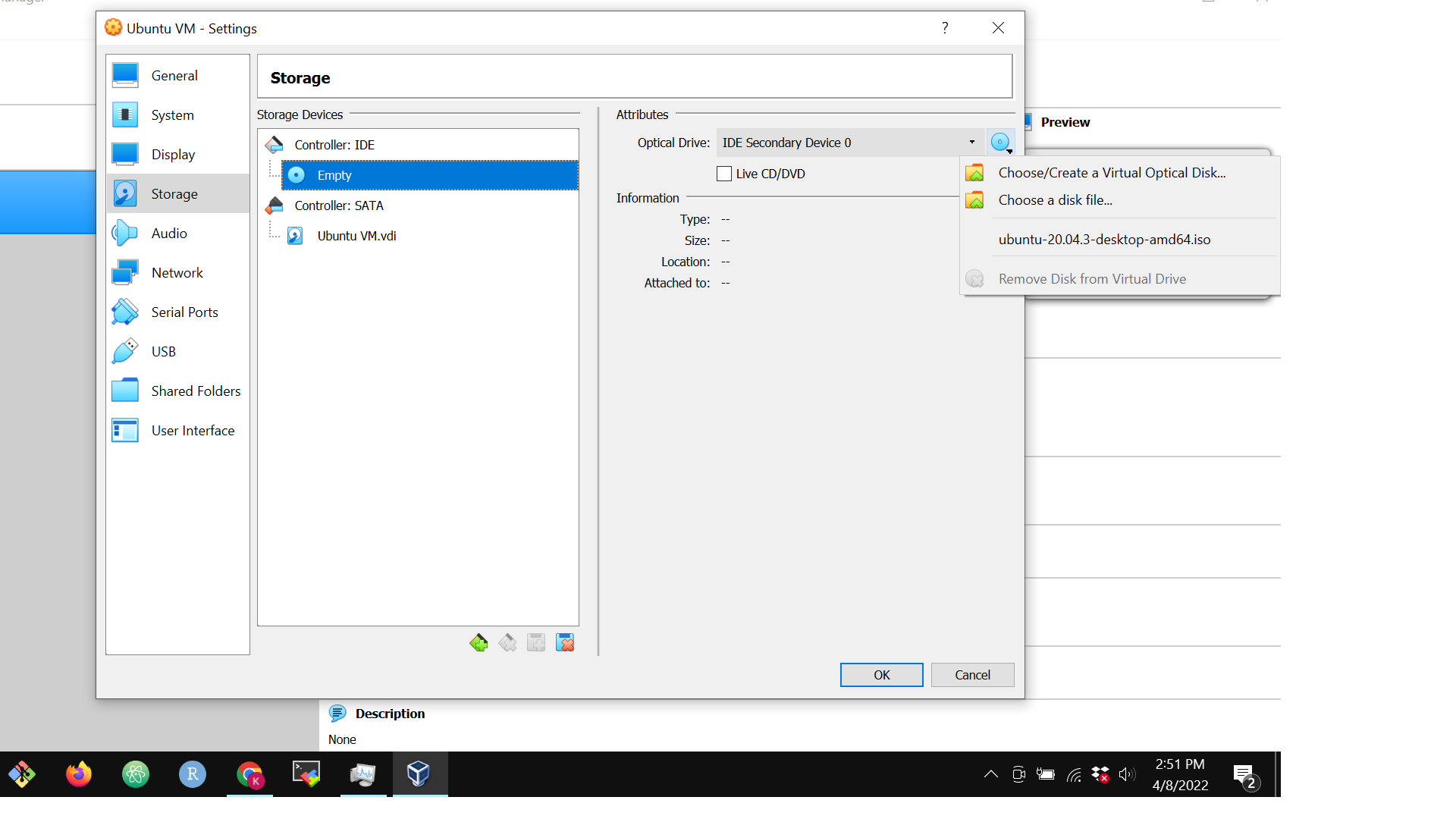
How do we install VM Ubuntu on Windows OS?

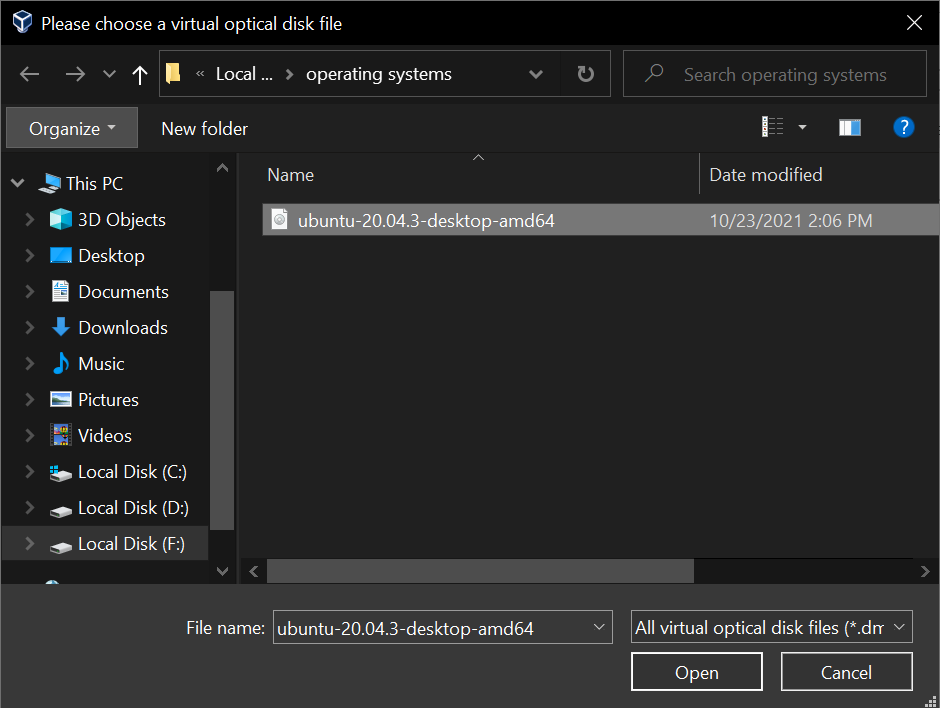
I am not sure about this but………

Caution/Disclaimer to participants: It will delete your Windows OS - meaning youwsl will lose everything on your PC. Make sure they back up their data on their own PC.

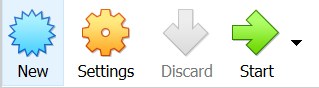


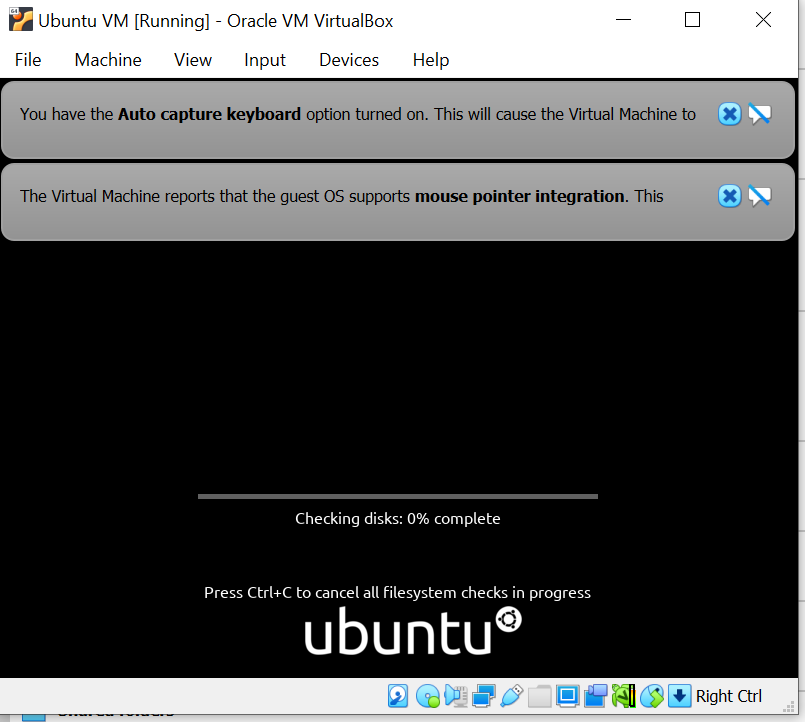
Finally, install the Ubuntu OS by going into the “Settings” icon in the menu bar for this virtual machine. In the Storage Devices section on the left, select “Empty” under “Controller: IDE.” Click the Blue CD Icon to the right of the optical drive dropdown. Click “Choose disk,” and then locate the Ubuntu ISO file you downloaded earlier. Select the ISO image and click “Open” and finally “Ok”





Press the “Start” button in VirtualBox, and begin the Ubuntu OS installation process. Follow the instructions on the screen to finish up the installation. Choose default settings during the installation.





**Step by step installation of Miniconda in VB with installed Ubuntu**

1. Open your Linux terminal and copy and paste this command on the terminal: wget <https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh>

Press “Enter” on your keyboard to execute the command. That will download the script used to install Miniconda, which is a lightweight installer for conda.

2. Once the script has downloaded, copy and paste the following command on the terminal and press “Enter” on your keyboard to execute: bash Miniconda3-latest-Linux-x86\_64.sh

That will install the conda package management tool. Accept default options if prompted to choose while the tool installs.

3. Once installation is complete, close and re-open the terminal for the changes to take effect. If the installation is successful, you should see a list of packages installed by default by typing and executing the following on the terminal: conda list

4. Next, we will install the required packages to run our analysis. On the terminal, run the following command one after the other:

* conda update --yes conda
* conda config --add channels defaults
* conda config --add channels bioconda
* conda config --add channels conda-forge
* conda create -n ngs python=3
* conda activate ngs
* conda install multiqc fastqc samtools bwa qualimap r-base igv kraken2 centrifuge bracken -y

**Installing the Windows Subsystem for Linux (WSL)**

If you have a laptop with Windows 10 Build 16215 or later, you can install Windows Subsystem for Linux (WSL) which offers a Linux terminal with the functionalities we need to run the analysis. Please follow the instructions here to install the WSL: <https://ubuntu.com/tutorials/install-ubuntu-on-wsl2-on-windows-10#1-overview>

Once you install WSL, follow the instructions listed under “**Step by step installation of Miniconda in VB with installed Ubuntu”** to install Miniconda and the required packages.