Table of contents

Installation of a Linux virtual machine on PC or Laptop

How to install Windows Subsystem for Linux using Settings

If you want to run distributions of Linux on Windows 10, you must first enable the Windows Subsystem for Linux feature before you can download and install the flavor of Linux that you want to use.

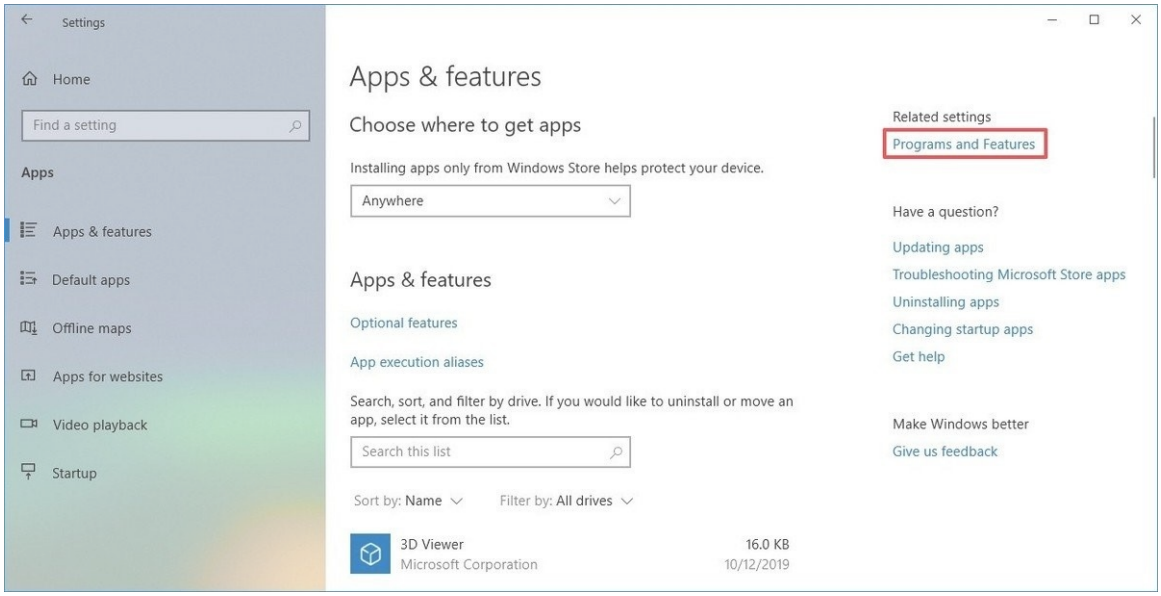
Enabling Windows Subsystem for Linux using Settings

To install WSL using Setting on Windows 10, use these steps:

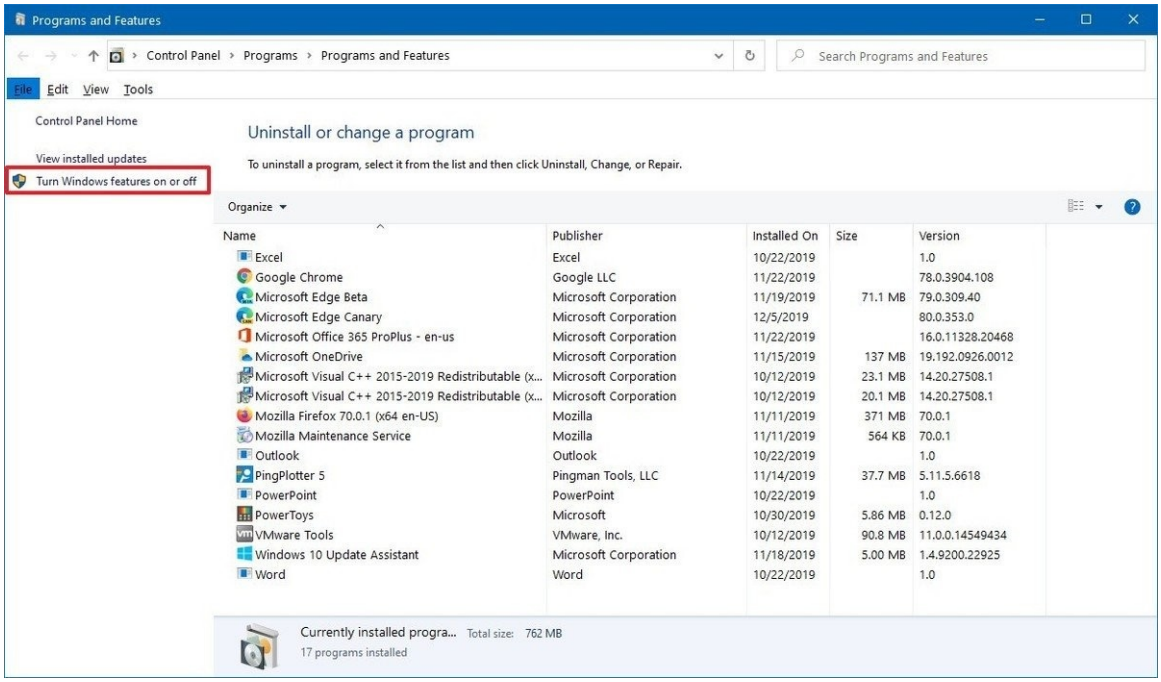
Open **Settings**.

Click on **Apps**.

Under the “Related settings” section, click the **Programs and Features** option.



Click the **Turn Windows features on or off** option from the left pane.



Check the **Windows Subsystem for Linux** option.

Click the **OK** button.

Click the **Restart now** button.

Once you complete the steps, the environment will be configured to download and run the distros of Linux on Windows 10.

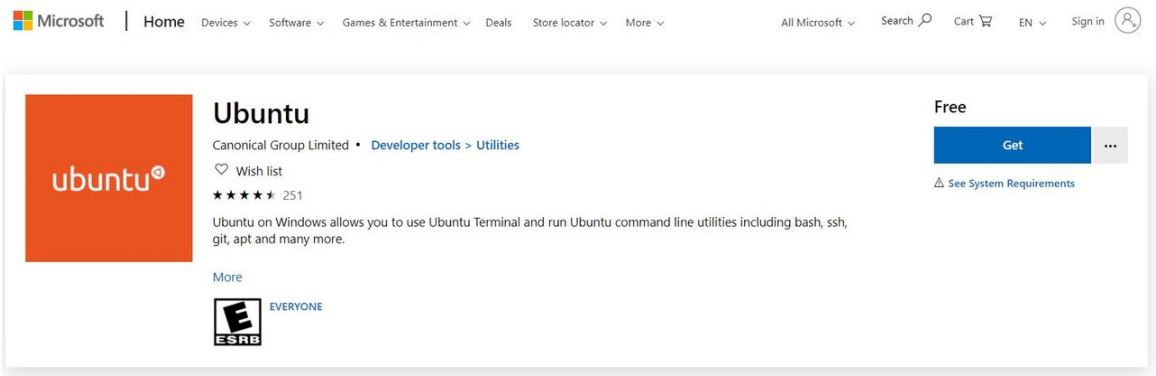
Installing Linux distros using Microsoft Store

To install a distribution of Linux on Windows 10, use these steps:

Open **Microsoft Store**.

Search for the Linux distribution that you want to install.

Select Ubuntu and **Get** it.



Once the software is downloaded you have to **LAUNCH** the server.
It takes a few minutes before your Linux virtual machine is installed and will be completed after:

```
enter a new UNIX username:vm-ubuntu
New password:plantbreeding
Retype new password:plantbreeding
```

Here we are, you have created a linux virtual environment and have a terminal open in which you can type or copy/paste the CoSSA pipeline commands.

For more info about the installation of linux virtual machines go to:
https://www.windowscentral.com/install-windows-subsystem-linux-windows-10

Setting up the CoSSA pipeline**

Create conda environment CoSSA

A convenient tool for the installation of software packages and setting up pipelines in Linux are the so called conda environments. The CoSSA conda environment will contain all the tools needed for running the pipeline. First you have to check if conda is already installed.

On the command line of the linux terminal type:

```
conda --version
```

If this commands returns the message `-bash: conda: command not found` you have to install it first. Miniconda is a free minimal installer of conda. It is a small, bootstrap version of Anaconda that includes only conda, python, the packages they depend on, and a small number of other useful packages.

On the command line type or copy/paste (right mouse click = paste)

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
```

The Miniconda3 package is downloaded and ready to install using the Unix shell and command language bash. You can have a look if it is there by typing the bash command `ls` ,and if it is try the following command line trick:
Type `bash` , followed by capital `M` and hit the tab key. As you will notice bash will fill in the file name for you so you don't have to type the filenames or directory names yourself in bash. Not only very convenient but will also save you a lot of typing errors ;-)

```
bash Miniconda3-latest-Linux-x86_64.sh
```

After three questions/remarks which can all be accepted Miniconda3 will be installed successfully and they will thank you for that. However there was also the message "=> For changes to take effect, close and re-open your current shell. <==" .

Type on the command line:

```
exit
```

and the Ubuntu terminal will close. Using the windows start button you can launch the virtual environment by a **Click** on the Ubuntu app.
Conda should now be available, so lets try again

```
conda --version
```

We will now create a conda environment specific for the CoSSA workflow

```
conda create --name CoSSA
```

To use a specific conda environment you first have to activate it by

```
conda activate CoSSA
```

```
(base) vm_ubuntu@L0142725:~$ conda create --name CoSSA
Collecting package metadata (current_repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.8.2
  latest version: 4.8.3

Please update conda by running

    $ conda update -n base -c defaults conda



## Package Plan ##

  environment location: /home/vm_ubuntu/miniconda3/envs/CoSSA



Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate CoSSA
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(base) vm_ubuntu@L0142725:~$ conda activate CoSSA
(CoSSA) vm_ubuntu@L0142725:~$
```

Installing the CoSSA software tools
Although it is possible to install the whole pipeline in one go, we will install the tools needed for the pipeline through conda, In this way you will get familiar with conda and can use it as well for install software packages outside the scope of CoSSA. The tools needed for the CoSSA pipeline are:

FastP – fastq preprocessor for quality control and filtering features

KMC3 – K-mer counter and KMC tools for operations with k-mer databases

BWA – alignment algorithm for aligning sequence reads against a reference genome

Sambamba – reading/writing/editing/indexing/viewing bam files

Mosdepth – Fast BAM depth calculation

Let's install the software packages using conda:

```
conda install -c bioconda/label/cf201901 fastp kmc bwa sambamba mosdepth
```

And of course we want to proceed:

```
Proceed ([y]/n)?  y
```

A few python packages for plotting the figures are needed as well

```
conda install numpy pandas matplotlib
```

It takes a few minutes but then all the tools needed should have been installed without major error messages.

The tools needed are installed but how to use them ? Most tools have a build-in help function which shows the usage of the tool, the options for this tool and the default settings of the options. e.g.

```
fastp --help

kmc --help

kmc_tools --help

sambamba --help

bwa –help

mosdepth --help
```

Downloading CoSSA scripts and toy data

The CoSSA pipeline is a set of bash scripts in which data is redirected to the above mentioned tools. These scripts are freely available, updated and maintained on GitHub. GitHub is a repository hosting service where you can find all kinds of useful, useless and freely available software.
To be able to test the pipeline and to get familiar with CoSSA , a toy data is created. Dataset 1 exists of simulated fastq data of a bulk of resistent samples, a bulk of susceptible samples and a reference chromosome. Dataset 2 consist simulated sequence data of a R-parent, R-bulk, S-parent and S-bulk and also makes use of the same reference chromosome.
To download the scripts and data from the PBR Github repository:

```
git clone https://github.com/PBR/CoSSA.git
```

and type the bash command `ls` and `ls CoSSA` on the command line.

```
(CoSSA) vm_ubuntu@L0142725:~$ git clone https://github.com/PBR/CoSSA.git
Cloning into 'CoSSA'...
remote: Enumerating objects: 44, done.
remote: Counting objects: 100% (44/44), done.
remote: Compressing objects: 100% (28/28), done.
remote: Total 44 (delta 15), reused 37 (delta 12), pack-reused 0
Unpacking objects: 100% (44/44), 50.19 MiB | 1.57 MiB/s, done.
Updating files: 100% (17/17), done.
(CoSSA) vm_ubuntu@L0142725:~$ ls
CoSSA  Miniconda3-latest-Linux-x86_64.sh  miniconda  miniconda3
(CoSSA) vm_ubuntu@L0142725:~$ ls CoSSA/
README.md  rawreads  reference  scripts
```

Instead of storing all data in one place, the data is structured into directories.
The directory CoSSA contains three sub directories. The reference genome sequence is found in directory **reference**, the sequence reads in directory **rawreads** and the CoSSA scripts in directory **scripts**. Later on after running the scripts of the CoSSA pipeline you will notice that new directories are created by the

pipeline. In these directories you will find the result files of the scripts.
Instead of staying in your so called home directory we will go to directory CoSSA from where we will run the scripts

```
cd CoSSA
```

You have used the bash commands `ls` and `cd` on the command line, but there are much more bash commands available which you can and will use when using the Linux command line. Here. you can find a complete list of the available bash commands.

Running the CoSSA pipeline

The CoSSA pipeline contains general and sample specific analysis. The general analysis like read quality checks, pre-processing of the reads and k-mer counting have to be done for all samples. Instead of running the analysis for each sample separately the CoSSA pipeline will do this using a loop. To do so you only have to add the name of the sample directory to a text file (e.g. genotypes.list but you can use whatever name) in directory **rawreads** and after running the scripts all the samples in the genotypes.list will have been processed.

The tool nano is an easy to use linux command line text editor which you can use to create a file like genotypes.list.

On the command line type

```
nano rawreads/genotypes.list
```

The file genotypes.list was already created and the sample names, one sample per line, were already added to this file

CoSSA_toy1
CoSSA_toy2

After adding all samples which you want to process, e.g the samples of dataset2, save and close the file `Ctrl x` followed by `y(es)`

Bash scripts

The CoSSA pipeline steps are executed on the server in a so called bash script. Bash reads the lines, one line at the time and interpreted and execute these lines as if they would have come directly from the keyboard (as you have be doing by typing nano) If the text in a bash script is commented (# in front of the word or sentence) the line is not executed. Usage of the bash scripts can be invoked by the `–help` function. If you're interested in to see how a bash script looks like you can use the command `less scripts/CoSSA_fastp_reads.sh`. You quit reading the file by typing `q` (q=quit).
The naming of the scripts is as follows:

scripts/ — path to the script
CoSSA_ — scripts developed for CoSSA
tool name_ — the tool which is called inside the script
data type — which type of data as input
.sh — extension for bash

As an example: scripts/CoSSA_fastp_reads.sh will run the tool fastp with reads as input but will throw an error if k-mer tables are used as input.

Quality check and pre-processing of sequence reads

Quality control and pre-processing of sequencing data are critical to obtaining high-quality and high-confidence variants in downstream data analysis. Data can suffer from adapter contamination, base content biases and over represented sequences. Fastp is a tool to provide fast all-in-one pre-processing of FASTQ files.The sequence quality of most sequence providers is already of high quality so we will do a quick quality check for one million read pairs and based on the results decide if analysis and trimming of the whole data set is required. The scripts needs two arguments, the list with samples to process and the number of reads to analyze

```
bash scripts/CoSSA_fastp_reads.sh genotypes.list 100000
```

As you will have noticed you already get some info about the quality on the screen but the folder **fastp** contains more info.

```
ls rawreads/CoSSA_toy1/fastp/
```

You will find two sequence files containing the quality filtered and trimmed sequences but there is also a fastp report made in html format which can be viewed using a browser. (If you are using a Linux server instead of the installed virtual machine use command `firefox rawreads/CoSSA_toy1/fastp/CoSSA_toy1_fastp.html`) On the command line type or copy/paste:

```
python3 -m http.server 8000
```

Next, copy/paste `[http://localhost:8000/]` in the address bar of your browser (Microsoft Edge, Chrome, Firefox etc.) and navigate to the html file (rawreads/CoSSA_toy1/fastp/CoSSA_toy1_fastp.html] and **Click** on the file (or use this link http://localhost:8000/rawreads/CoSSA_toy1/fastp/CoSSA_toy1_fastp.html)
All the information generated by the tool is now visualized in the browser.
You can do the same for CoSSA_toy2. (or use this link http://localhost:8000/rawreads/CoSSA_toy2/fastp/CoSSA_toy2_fastp.html)

After inspection of the data we have to decide if it is necessary to filter and trim the sequence data. Filtering a full dataset comes at a cost, the computational time of the CoSSA pipeline will increase substantially but the gained increase in the data quality is often negligible. However, if you decide to use a filtered set, you have to adjust the path of the reads files in the genotype.list
To continue the pipeline we have to go back to the command line. In most cases using **Ctrl+C** will do the trick.

K-mer counting

KMC3 is a memory efficient and fast algorithm for generating k-mer tables for sequence reads and comes with the kmc_tools suite for manipulating of these k-mer tables
The script for counting k-mers makes also use of a loop structure and it needs a list of samples to process as an argument:

```
bash scripts/CoSSA_kmc3_count_kmers_reads.sh genotypes.list
```

The scripts creates a new directory kmc within the sample directory which contains all the output files. Beside the unreadable binary k-mer table also a histogram file is created in the directory kmc. e.g CoSSA_toy1.kmc3.histo. This file contains the kmer frequency distribution of the sample reads:

```
less rawreads/CoSSA_toy1/kmc/CoSSA_toy1.kmc3.histo
```

This file is also used for plotting a simple k-mer profile (rawreads/CoSSA_toy1/kmc/CoSSA_toy1.kmc3.histo.png).
To view this file type on the command line

```
python3 -m http.server 8000
```

and go to http://localhost:8000 and navigate to the .png file

A more advanced analysis of the k-mer statistics is done by the online tool of Genomescope2 . There you can upload a kmer histo file and run the analysis. We will copy the CoSSA_toy1.kmc3.histo file to the windows environment and upload the file from there to Genomescope2;

```
cat rawreads/CoSSA_toy1/kmc/CoSSA_toy1.kmc3.histo
```

Select the content of the file using the left mouse button
copy **Ctrl+C**
and paste **Ctrl +V** the content in Notepad and save the file as CoSSA_toy1.kmc3.histo
Do the same for CoSSA_toy2.

The CoSSA_toy1.kmc3.histo file is now copied to your windows environment.
Start Genomescope2, upload the histo file(s) and have a look at the kmer profiles.
Indeed, it is a bit cumbersome to get the data to Genomescope2 but that has to do with the setup of this hands-on virtual environment. On the plantbreeding server the .histo file can be directly uploaded to Genomescope2.

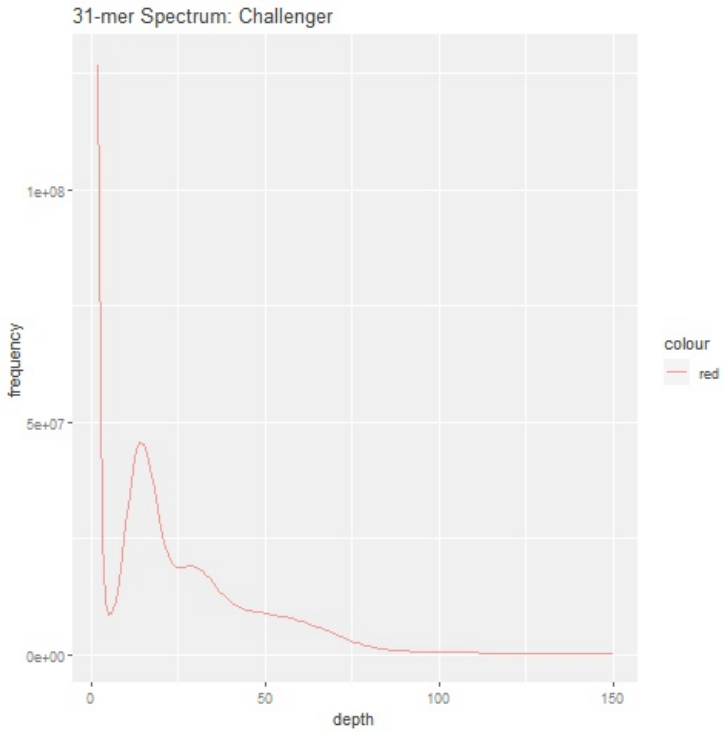


Fig1: k-mer profile of a real data set of a tetraploid potato cultivar

K-mer table manipulations using kmc_tools

In the CoSSA paper a part of the strategy is to subtract the k-mers from the susceptible bulk from the k-mers from the resistent bulk to obtained resistent bulk specific k-mers.
The CoSSA pipeline has a few scripts to play around with k-mer tables. Note that these scripts can be easily be adapted to your needs by changing parameters within the script using nano. The scripts needs as arguments the two names of the k-mer tables which you want to use.

Subtraction of k-mer tables:

```
bash scripts/CoSSA_kmc3_subtract_kmers.sh rawreads/CoSSA_toy1/kmc/CoSSA_toy1 rawreads/CoS
```

Intersection of k-mer tables can be made by:

```
bash scripts/CoSSA_kmc3_intersect_kmers.sh rawreads/CoSSA_toy1/kmc/CoSSA_toy1 rawreads/Co
```

For combining k-mer table of samples you can use the script CoSSA_kmer_union.sh

See the manual of kmc_tools for all the options.
The result files of the different scripts are stored in a newly created directory and can be viewed by:

```
ls k-mer_manipulations
```

Advanced filtering options

Using a kmer frequency plot generated by Genomescope2 or CoSSA, coverage cut off values can be determined for several reasons. The lower cut off eliminates error sequences Note that the k-mers with frequency 1 were already removed from the k-mer tables, but maybe a higher threshold is needed. The high cut off values eliminates k-mers from multiple copy regions that you may not want to be included in your analysis. Besides these cut offs also thresholds can be set for selecting specific ploidy k-mers, e.g unique k-mers (simplex) or the duplex, triplex etc k-mers. Thresholds could be defined as the 5th and 95th percentile of the coverage median (median coverage x -1.645 and median coverage x 1.645)

Aligning selected k-mers to a reference genome

When you have made selections of k-mers using these k-mer scripts and you probably want to know where these k-mers will align on a genome of interest. Therefore you will need a reference genome sequence. For the toy example there is already a reference sequence available in the directory **reference** e.g chr11.fa.

However , if you need other reference sequences and you know were to get it e.g. at NCBI or in case of the potato genome of Solyntus at https://www.plantbreeding.wur.nl/Solyntus/ you can download the sequences to the directory **reference**.
In general a genome sequence file is a compressed file and you first have to decompress the file to a fasta file before you can use it:

```
tar zxvf Solyntus_v1.1.fasta.tar.gz
```

To be able to use a genome sequence for aligning the original reads or k-mers the BWA alignment tool makes use of an index which has to created just once for each genome.
Here we will use the already decompressed reference file chr11.fa.

```
bash scripts/CoSSA_bwaindex_genome.sh reference/chr11.fa
```

Next the extracted k-mer set can be aligned to the reference sequence

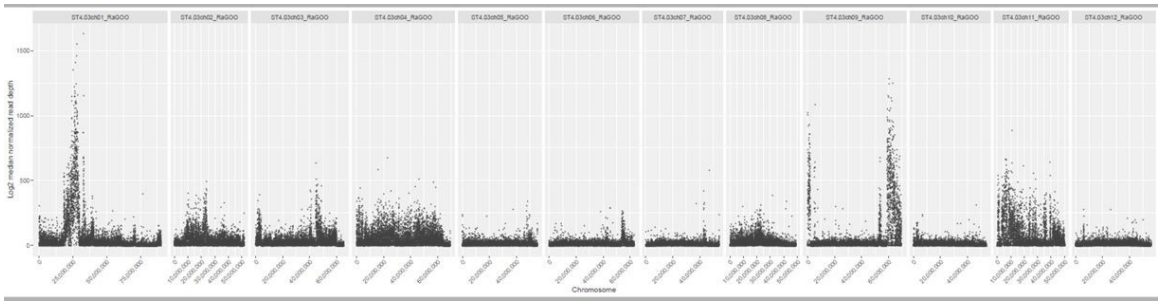
```
bash scripts/CoSSA_bwasamse_kmers.sh reference/chr11.fa k-mer_manipulations/CoSSA_toy1.Co
```

If not already present a new directory bamfiles is created and the alignment file, a so called bam file is stored in here. The next step is to identify the regions on the chromosomes were the selected k-mers are aligned. This is done by the tool mosdepth which takes the bamfile as an argument.

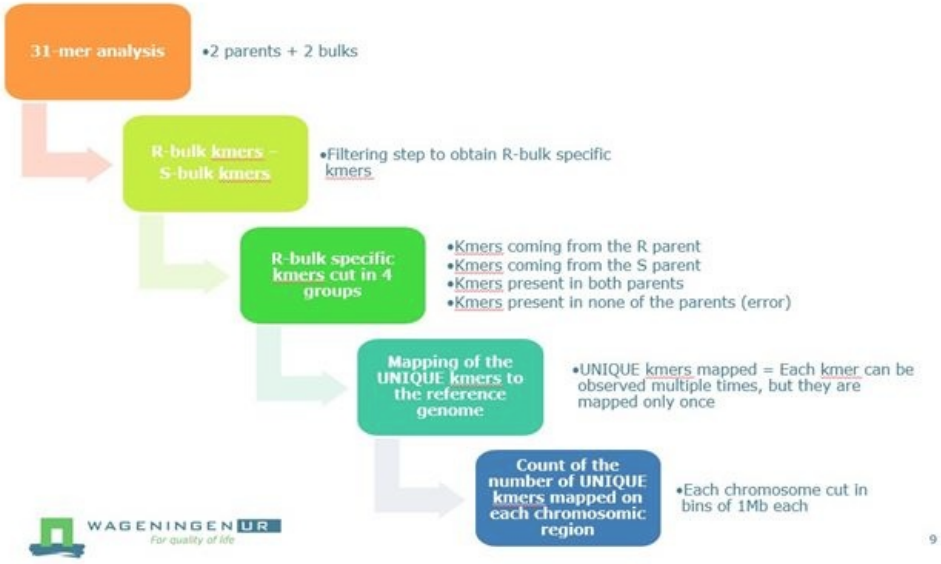
```
bash scripts/CoSSA_mosdepth_bam.sh bamfiles/CoSSA_toy1.CoSSA_toy2.subtract.bam
```

This command shows in the terminal the k-mer distribution over the first 20 bins of 1kb of chr11. The command `ls bamfiles` lists all the create files in the bamfiles directory including a .png file. This file shows the k-mer distribution over the whole chromosome . To have a look copy/paste `localhost:8000` in the browser address bar , navigate to the file and **Click** on it. For this hands-on we zoomed into the region of interest, instead of showing the whole chromosome.

In real life you're probably more interested in the k-mer distribution over a whole genome. In the next figure you see such an example as the result of the CoSSA workflow with k-mer peaks on chr1 en chr9



With the above examples of how to use the CoSSA scripts and have an idea about the results to expect, you can start your own CoSSA strategy using the other datasets which are available in the rawreads directory. You could for instance follow the strategy as described in the CoSSA paper according to the following flowchart



Filtering steps which can be performed on the available datasets using the CoSSA scripts are e.g:

- Rbulk - Sbulk = Rbulk specific k-mers
- Sbulk - Rbulk = Sbulk specific k-mers
- Rbulk specific k-mers ∩ Rbulk = R-bulk specific k-mers in common with Rbulk
- Rbulk specific k-mers ∩ Sbulk = R-bulk specific k-mers in common with Sbulk
- Rbulk specific k-mers – R-parent = Rbulk specific k-mers not in R-parent
- Rbulk specific k-mers – S-parent = Rbulk specific k-mers not in S-parent
- Rbulk specific k-mers not in R-parent - Rbulk specific k-mers not in S-parent = Rbulk specific k-mers inherited from S-parent
- Rbulk specific k-mers not in S-parent - Rbulk specific k-mers not in R-parent = Rbulk specific k-mers inherited from R-parent
- Rbulk specific k-mers not in R-parent - Rbulk specific k-mers in common S-parent = Rbulk specific k-mers inherited from none of the parents = sequencing errors and contamination
- Rbulk specific k-mers in common R-parent ∩ Rbulk specific k-mers in common S-parent = Rbulk specific k-mers inherited from both parents

The sum of these last 4 k-mer sets = Rbulk specific k-mers.

Maybe you want to know what the intermediate results are of the different filtering steps, want to know how the remaining k-mers look like or to count the number,. You can use the script CoSSA_kmc3_dump_kmers.sh to do so... It takes a kmc library and prints out a readable dump file. To have a look at the k-mers themselves use the command `less k-mer_manipulations/<name>.dump` To count the number of k-mers you can use the command `wc -l k-mer_manipulations/<name>.dump`
After selecting the specific k-mer sets you can continue with the mapping and coverage estimation scripts to find the physical position of the k-mers on a genome.

Using a virtual Linux environment is a convenient way to play around with scripts, scripting languages like Python and other Linux related secrets. However, if you are going to use the CoSSA on real data sets you have to switch to a Linux server since you will need more storage space and memory (RAM). The CoSSA Github repository will be updated with scripts and manuals for the CoSSA analysis on the WUR PlantBreeding Server.

Have fun ! and if you have any questions or comments please let us know.

Corentin Clot
Herman van Eck
Reinhard Simon
Danny Esselink
Richard Visser

With acknowledgement to Charlotte Prodhomme, Jack Vossen and Simone Dimitrova.

> Written with StackEdit.