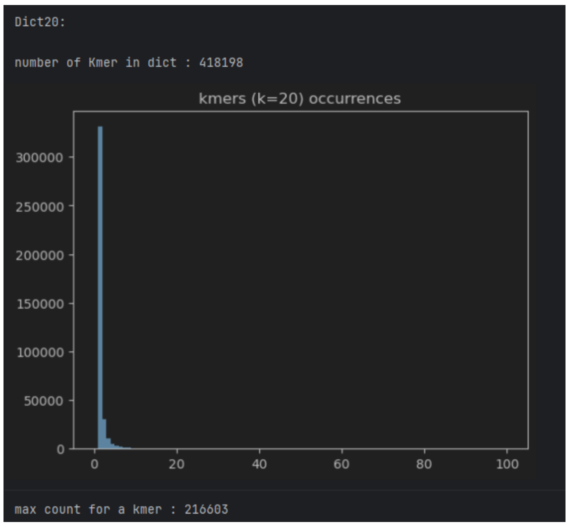
SARSCoV2 *De Novo* assembly

This project consist in assembling the genome of SARSCoV2 from cleaned reads using the De Bruijn Graph (DBG) assembly method. In this project, I will not build a full graph but construct contigs using the K-1 overlap method used in DBG.

1) Kmer Dictionnary

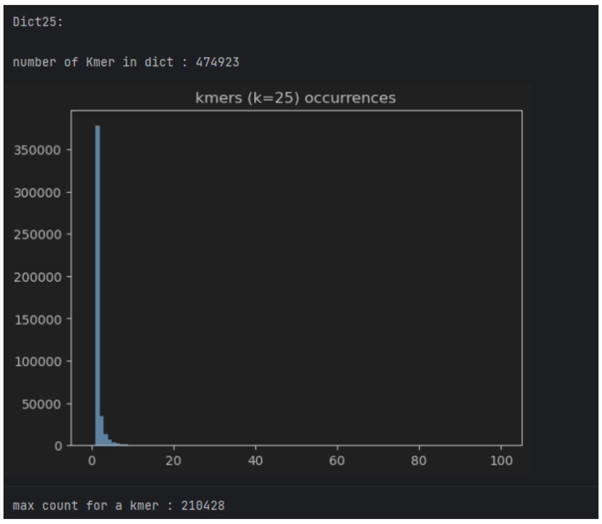
The first step is to build the Kmer dictionary from the reads fastq file, we will test 3 length of Kmer : 20,25, and 30.
Only one Kmer is kept for both forward or reverse complement variant. For each Kmer in the dictionary we keep the number of occurrences (either in forward or reverse complement form) it appear in the reads.



```
Dict20:

number of Kmer in dict : 418198
```

```
max count for a kmer : 216603
```

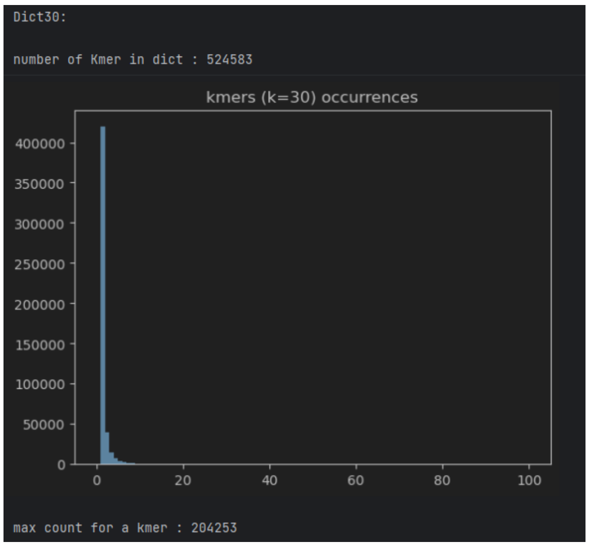


```
Dict25:

number of Kmer in dict : 474923
```

```
max count for a kmer : 210428
```

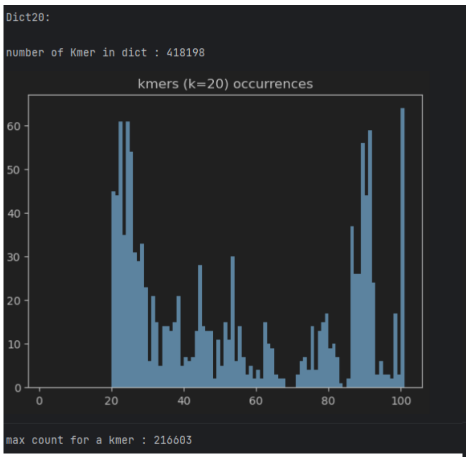
In all the dictionaries, the vast majority of contigs appear only once in the reads. This is due to errors in sequencing. By removing these Kmer we greatly reduce the number of Kmer to compute during the assembly phase, plus we avoid integrating these Kmer in our contigs. With a threshold of 2 occurences a very large proportion of remaining Kmer have low frequency. We can see a better frequency distribution from around 20 as threshold (shown after)



```
Dict30:

number of Kmer in dict : 524583
```

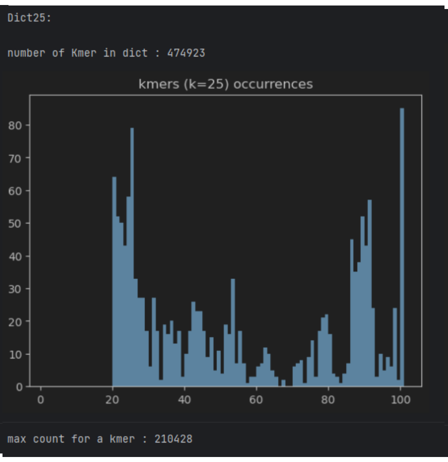
```
max count for a kmer : 204253
```



```
Dict20:

number of Kmer in dict : 418198
```

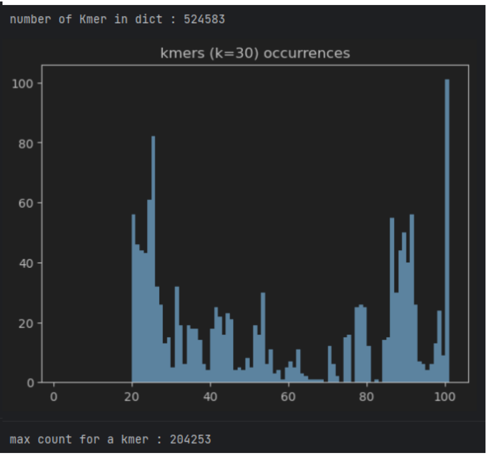
```
max count for a kmer : 216603
```



```
Dict25:

number of Kmer in dict : 474923
```

```
max count for a kmer : 210428
```



```
number of Kmer in dict : 524583
```

```
max count for a kmer : 204253
```

From this point in the script, kmer dictionaries from the genome are built. They are not used in the assembling phase, they are just used as quality assessment.
Removing low occurrences Kmer greatly increase the accuracy (~90% increase) of the dictionary but also slightly reduce the coverage (>1% see script output for more details)

2)Contigs Building

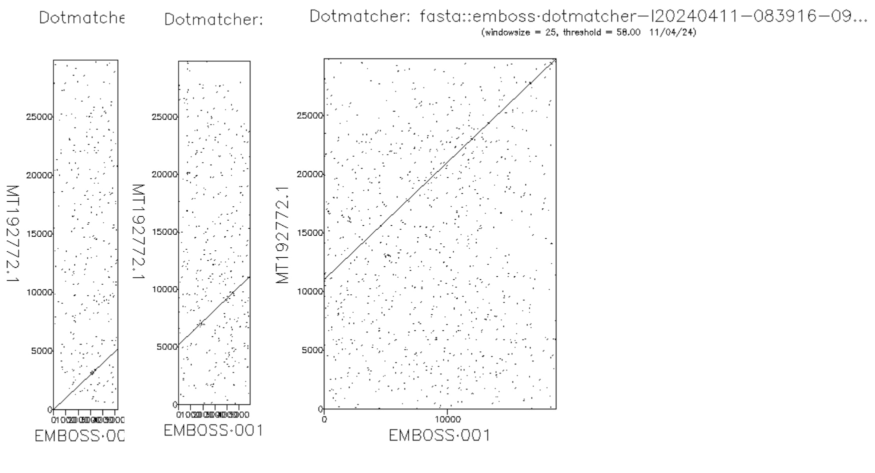
Contigs are built from the kmer dictionary, each active Kmer with an occurrence score superior or equal to 60 can serve as "seed". A seed get elongated and become a branch, a list of kmer ordered in 3' to 5' sens. The ContigConstructor function check for every Kmer in dictionary for seeding. Once a Kmer has been selected as seed, it is elongated (= growth step) from both ends with K-1 overlapping kmer found in dict (reverse complement of Kmer are also tested since only one version of Kmer is kept in dict).
If two or more Kmer are found to be added to the branch, the Kmer with the greater occurrence score is added and the second serves as new seed and is appended to a list of "branches" (containing the first seed) so that each branch in this list get elongated. Since the number of branches rapidly increase, at each iteration branches that haven't growth (full branches) are removed and re-added after the growth step to reduce computational impact. When a kmer is appended in a branch it gets deactivated and can no longer be used unless it get reactivated.
At the end of the growth step, the completed branches are returned and all the branches that are long enough (at least 2 Kmer long to prevent SNP contigs to be kept) and have a minimum branch scoring are assembled into contig sequence .
The branch score gets the mean score of all Kmer occurrences in the branch. Normally Kmer within the genome should appear more frequently in the reads and a contig should contain mostly high score kmer, this prevent having contig only formed from low occurrence kmer which are unexpected (except for low coverage region). Experimentally, the score have been put to 400 (which is close to the mode of the distribution of genome kmer occurrences, see script for details)
If a branch doesn't meet the requirement to be assembled as contig, the high occurrence Kmer are reactivated to be used in new contigs (a copy of the dictionary is used in the function, to allow this reversion from the original).
The contigs are stored in a Fasta file.

3)Assembly quality assessment

After the contig building phase we can retrieve our fasta file with the contigs and assess the quality of the assembly.
First step is to ensure the quality of the kmer used in the contigs, we compare the Ker content of the assembly with the Kmer content of the genome as seen previously.
The score are great but the assembly with k=30 as less contigs with a better accuracy and coverage (only 55 genome Kmer over 29 150 have been "lost" during the contig building phase)
According to these metrics, L50 and N50 are also better with K=30

Maping the contigs on the genome using blastn on proksee.ca confirm that the contigs covers most of the SARS-CoV-2 genome (figures on the last page). But it shows that contigs are on both strand (non-sens for virus, so some contigs need to be reverted in reverse complement). Another interesting point is that contigs seems to overlap (especially for k = 30). This suggest that overlapping contigs could contains same shorter Kmer, and that it is possible to merge them by reassemble contigs from the assembly fasta file and a lower kmer length.

To do so we re-create a Kmer dictionary of length 25 from the k30 assembly, re-score the occurrences found by multiplying by 200 (to allow seeding and re-activating in ContigConstruct function), if a Kmer is shared between contigs, it should be scored at 2 then at 400 and should be selected first in branch elongation allowing contig fusion.
By repeating this process multiple times we can reduce to 3 the number of contigs. Allowing for L50 = 1 and N50 = 18808.



Here are the Dotplots for the contigs 3,2,1 from the full assembly, contigs 1 and 2 are in reverse complement.

4)Discussion

The contigs formed still contains errors such as SNP and using only one on the two reads files doesn't allow to retrieve all the Kmer from the genome.

Furthermore, the threshold on the number of occurrences to obtain the cleared dictionary also mask Kmer from genome, maybe using both files could help resolve these issues.

However there will always have low coverage regions from the sequencing due to under representation of reads containing these region or other technical issues.
To overcome this we can only sequence multiple time the specie until we get enough information.

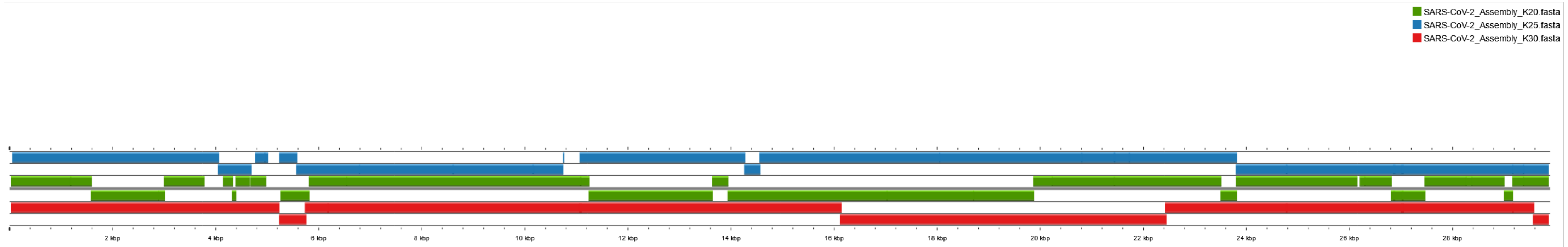
The way the contigs are build could induce errors, since the number of occurrence determine the integration of a Kmer, we could imagine a low covered Kmer that get rejected, also we don't keep data about the Kmer variants rejected, which make it difficult to retrieve the correct sequence.

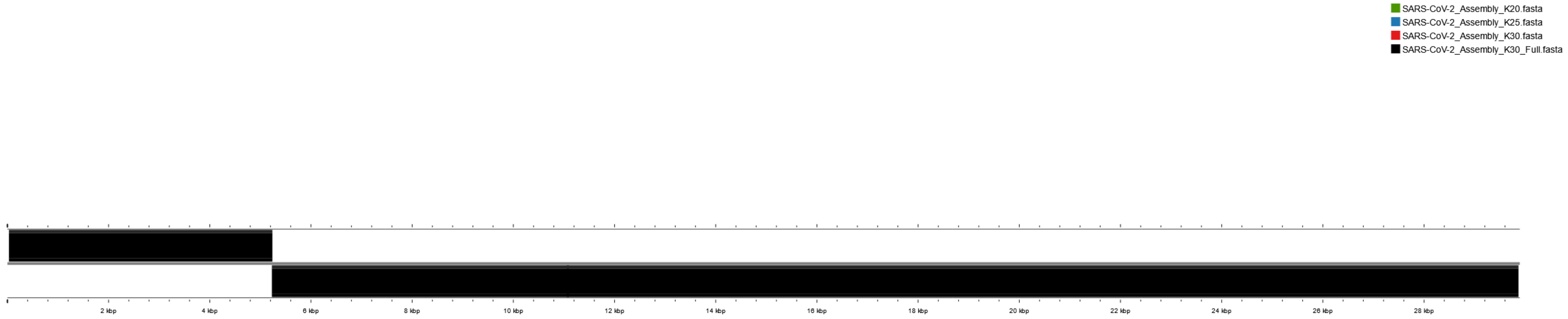
Another issue is the possibility that a Kmer is redundant or correspond to another Kmer in reverse complement, in this case the first contig in which it gets included prevent the others to integrate it, preventing elongation or forcing the integration of an bad Kmer in place.

SNP could lead to mismatching contigs too, they are most likely to have a pretty high occurrence score making them hard to find the good variant to integrate but further occurrence analysis could easily find this type of variance since the global coverage of these regions gets split into few variants.

An interesting way could be to attempt to elongate the contig obtained with the full dictionary to pass through eventual low coverage regions.

Here the assembly quality assessment was easier because we already got a consensus sequence to compare to. But for real *De Novo* assembly, unless using close specie for which we have a genome , it require more steps and more tools to have this quality. For instance the merge of the contigs could have used global alignment algorithm to find overlaps.


🟩 SARS-CoV-2_Assembly_K20.fasta
🟦 SARS-CoV-2_Assembly_K25.fasta
🟥 SARS-CoV-2_Assembly_K30.fasta
2 kbp
4 kbp
6 kbp
8 kbp
10 kbp
12 kbp
14 kbp
16 kbp
18 kbp
20 kbp
22 kbp
24 kbp
26 kbp
28 kbp


🟩 SARS-CoV-2_Assembly_K20.fasta
🟦 SARS-CoV-2_Assembly_K25.fasta
🟥 SARS-CoV-2_Assembly_K30.fasta
⬛ SARS-CoV-2_Assembly_K30_Full.fasta
2 kbp
4 kbp
6 kbp
8 kbp
10 kbp
12 kbp
14 kbp
16 kbp
18 kbp
20 kbp
22 kbp
24 kbp
26 kbp
28 kbp