we would download genomes from Gisaid EpiCoV https://www.epicov.org/epi3/frontend
Download the fasta which contains all fasta files (access must be requested)
We would then use scripts that are part of the augur pipeline
https://docs.nextstrain.org/projects/augur/en/stable/ to sanitze sequences and metadata files:

e.g.:

```
sanitize_sequences.py --sequences SEQUENCES.tar.xz --strip-prefixes 'hCoV-19/'
--output SEQUENCES.fasta.gz

sanitize_metadata.py --metadata METADATA.tar.xz --database-id-columns 'Accession
ID' --parse-location-field Location --rename-fields 'Virus name=strain' 'Accession
ID=gisaid_epi_isl' 'Collection date=date' --strip-prefixes 'hCoV-19/' --output
METADATA.sanitized.gz
```

This produces a very large compressed fasta file. To help parallelize the downstream analyses, the file is split into smaller parts using GNU Parallel, pigz, and bioawk. We create a tab separated file (TSV) with the sequence accession and the sequence. For example:

```
pigz -dc SEQUENCES.fasta.gz | parallel --pipe -j 8 --block 2000M --recend '\n'
"bioawk -c fastx '{print \$name\"\t\"\$seq}' | pigz -p 4 >
split-fastas/sequences_gisaid_sanitized_split_{#}.tsv.gz"
```

to organize the files by date, we used a custom R script. This script will read in the TSV file (chunked), merge it with metadata tsv file. Then output fasta files into directories, one for each day:

```
library(data.table)
args <- commandArgs(trailingOnly = TRUE)
if (length(args) == 0) {stop("No file provided. Exiting.")}
metafile <- args[1]
if (!file.exists(metafile)) {stop(paste("File", metafile, "not found. Exiting."))}
seqfile <- args[2]
if (!file.exists(seqfile)) {stop(paste("File", seqfile, "not found. Exiting."))}
meta<-fread(metafile, sep='\t', header=T)
seqs<-fread(seqfile, sep='\t', header=F)
colnames(seqs)<-c("strain", "seq")
meta.seqs<-merge(meta,seqs, by="strain")
if(nrow(meta.seqs) < 1){stop(paste("File", seqfile, "does not have samples.
Exiting."))}
meta.seqs$strain<-gsub("[/|]", "_", meta.seqs$strain)
for (i in 1:nrow(meta.seqs)){
   file_name <- paste0("fastas-by-date/",as.character(meta.seqs$date[i]), "/",
meta.seqs$strain[i],".fa");
   file_content<-paste0(">", meta.seqs$strain[i],"\n",meta.seqs$seq[i]);
   write(file_content, file=file_name)
```

```
}
```

and then run the Rscript for each split fasta to organize the fastas by date. e.g.:

```
Rscript organize-fastas-by-date.R METADATA.sanitized.gz split_fasta/[IN].fa
```

note: We would parallelize this with PBS scripts specific to our HPC system.

After files are organized into directories by date, we would run the Khill Script available from: https://github.com/deanbobo/khill

We would also run pangolin available from here: https://github.com/cov-lineages/pangolin
pangolin installed with a conda environment
note: it's faster to concatenate all the fasta files for a particular day and run them through the pangolin pipeline all at once. Again, we would parallelize this with our HPC system.

```
conda activate pangolin
pangolin --update
pangolin --update-data
pangolin --threads 4 --outfile PANGOLIN.csv IN.fa
```

#We would then load Khill and pangolin results and overlay summarized pangolin clade data using R.
#load and summarize pangolin:

```
library(data.table)
library(tidyverse)
# load pangolin results
pang<-fread("PANGOLIN.csv", sep=",", header=F)

#rename columns
colnames(pang)<-c("date",
"taxon","lineage","conflict","ambiguity_score","scorpio_call","scorpio_support","sc
orpio_conflict","scorpio_notes","version","pangolin_version","scorpio_version","con
stellation_version","is_designated","qc_status","qc_notes","note")

#filter out header rows
pang<-pang[!pang$taxon=="taxon",]

#format date
pang$date<-as.Date(pang$date)

#select only date and clade
pang<-pang[,c(1,3)]
colnames(pang)<-c("Date", "Clade")

#summarize the results by date and clade
pang.freqs<-pang %>% group_by(Date, Clade) %>% summarise(count = n()) %>%
mutate(freq=count/sum(count))
#format date
```

```
pang.freqs$Date<-as.Date(pang.freqs$Date)

# load khill results:
khill<-fread("khill.results.tsv", sep="\t", header=F)
colnames(khill)<-c("date", "khill")

#format date
khill$date<-as.Date(khill$date)

#for plotting purposes, normalize khill values 0 to 1
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
khill$khill.norm<-normalize(khill$khill)

# Plot results
ggplot() +
  theme_classic() +
  geom_bar(data=pang.freqs, aes(x=Date, y=freq, colour=Clade), stat="identity",
width=1) +
  geom_point(data=khill, aes(x=Date, y=khill.norm, fill=count), colour="black",
pch=21, size=3) +
  scale_y_continuous(name = "Clade Frequency", position="right", sec.axis =
sec_axis(~ scales::rescale(., range(khill$khill), range(khill$khill.norm)) ,
name="KHILL")) +
  scale_x_date(date_labels = "%Y %b", date_breaks = "2 month") +
  theme(legend.position="none", axis.text.x=element_text(angle=60, hjust=1),
panel.background = element_blank()) +
  labs(title="K-Hill", x="Date")
```