GDSCN Book: SARS with RStudio on AnVIL

May 07, 2025

Table of Contents

# Overview

This book provides resources for instructors to engage students in a cloud-based RStudio activity on AnVIL, focused on the evolutionary relationships among the SARS-CoV-2 variants.

There is a growing need for undergraduate students to learn cutting-edge concepts in genomics data science, including performing analysis on the cloud instead of a personal computer. This lesson aims to introduce basic tree building and interpretation using publicly available genetic samples of SARS-CoV-2. Students will be introduced to the sequencing revolution, variants, the basics of tree building and reading phylogenies, and essentials of cloud computing prior to the lab activity. During the lesson, students will work hands-on with RStudio on the [AnVIL](https://anvilproject.org/) cloud computing resource to check data, build trees, and visualize their results.

## Skills Level

*Genetics*  
**Beginner**: minimal genetics knowledge needed

*Programming skills*  
**Beginner**: minimal programming experience needed

## Learning Objectives

Learning objectives for this activity come from the [Genetics Core Competencies](https://genetics-gsa.org/education/genetics-learning-framework/):

* Generate and interpret trees displaying experimental results
* Use bioinformatics to assess genetics data
* Tap into the interdisciplinary nature of science

## GDSCN Collection

This exercise is part of a collection of teaching resources developed through the *Genomic Data Science Community Network* (GDSCN). GDSCN works towards a vision where researchers, educators, and students from diverse backgrounds are able to fully participate in genomic data science research. Learn more about GDSCN by visiting <https://www.gdscn.org/home> or reading the [article in Genome Research](https://doi.org/10.1101/gr.276496.121).

Please check out our full collection of AnVIL and related resources: <https://hutchdatascience.org/AnVIL_Collection/>

# 1 Student Activity Guide

This chapter contains the student instructions for the SARS-CoV-2 Phylogeny with RStudio activity.

## 1.1 Introduction

Since the beginning of the Covid-19 pandemic in December 2019, the world has experienced waves of increased cases caused by new variants. This activity walks you through a simple phylogeny to explore how the new variants are related. If you are interested in learning more about this topic, we recommend you check out the SARS-CoV-2 resources on [Nextstrain.org](https://nextstrain.org/sars-cov-2/).

### 1.1.1 Before You Start

If you do not already have a Google account that you would like to use for accessing Terra, [create one now](https://accounts.google.com/SignUp).

If you would like to create a Google account that is associated with your non-Gmail, institutional email address, follow [these instructions](https://support.terra.bio/hc/en-us/articles/360029186611).

### 1.1.2 Objectives

This activity will teach you how to use the AnVIL platform to:

1. Get started working on AnVIL
2. Launch RStudio
3. Import data into RStudio
4. Examine fasta and phyDat files
5. Build a neighbor-joining phylogeny
6. Interpret the topology and branch lengths of a phylogeny

## 1.2 Getting Started

In the next few steps, you will walk through how to get set up to use RStudio on the AnVIL platform. AnVIL is centered around different “Workspaces”. Each Workspace functions almost like a mini code laboratory - it is a place where data can be examined, stored, and analyzed. The first thing we want to do is to copy or “clone” a Workspace to create a space for you to experiment.

Use a web browser to go to the AnVIL website. In the browser type:

anvil.terra.bio

**Tip** At this point, it might make things easier to open up a new window in your browser and split your screen. That way, you can follow along with this guide on one side and execute the steps on the other.

A workspace for this activity on AnVIL coming soon!

### 1.2.1 Set Up

### 1.2.2 Video overview of RStudio on AnVIL

Here is a video tutorial that describes the basics of using RStudio on AnVIL.

### 1.2.3 Objectives

* Start compute for your RStudio environment
* Tour RStudio on AnVIL
* Stop compute to minimize expenses

### 1.2.4 Slides

The slides for this tutorial are are located [here](https://docs.google.com/presentation/d/1eypYLLqD11-NwHLs4adGpcuSB07dYEJfAaALSMvgzqw).

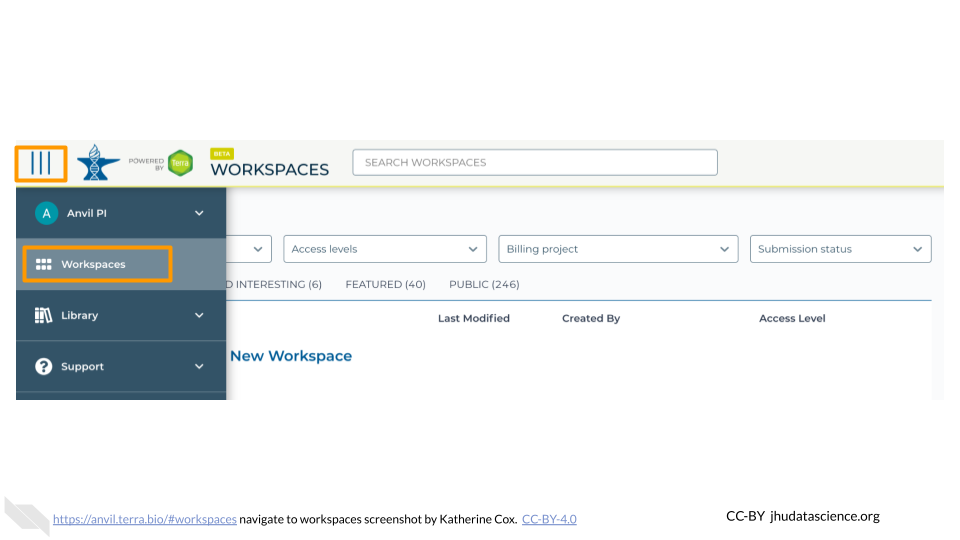
#### 1.2.4.1 Launching RStudio

## Warning: Chunk option fig.alt is not supported for docx output  
## Warning: Chunk option fig.alt is not supported for docx output  
## Warning: Chunk option fig.alt is not supported for docx output  
## Warning: Chunk option fig.alt is not supported for docx output  
## Warning: Chunk option fig.alt is not supported for docx output  
## Warning: Chunk option fig.alt is not supported for docx output  
## Warning: Chunk option fig.alt is not supported for docx output  
## Warning: Chunk option fig.alt is not supported for docx output  
## Warning: Chunk option fig.alt is not supported for docx output

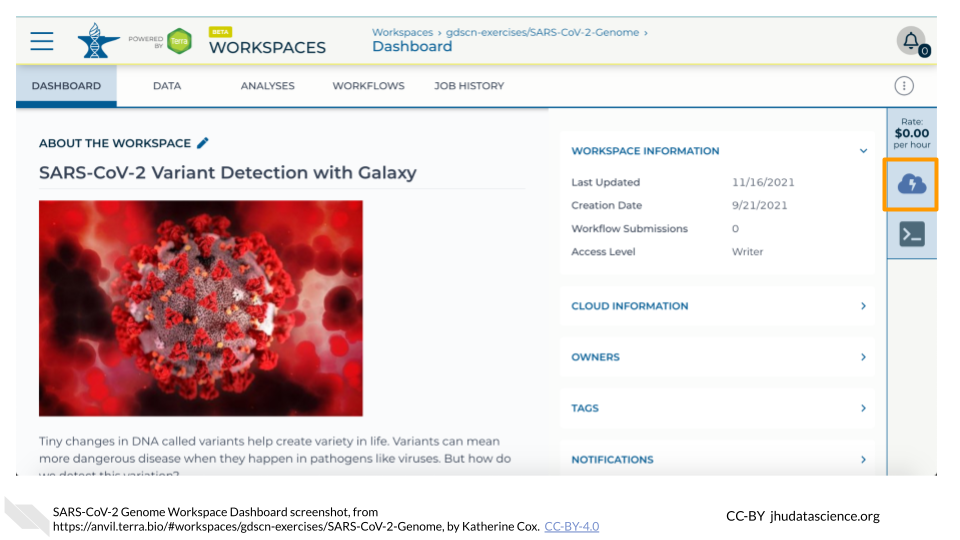
AnVIL is very versatile and can scale up to use very powerful cloud computers. It’s very important that you select a cloud computing environment appropriate to your needs to avoid runaway costs. If you are uncertain, start with the default settings; it is fairly easy to increase your compute resources later, if needed, but harder to scale down.

Note that, in order to use RStudio, you must have access to a Terra Workspace with permission to compute (i.e. you must be a “Writer” or “Owner” of the Workspace).

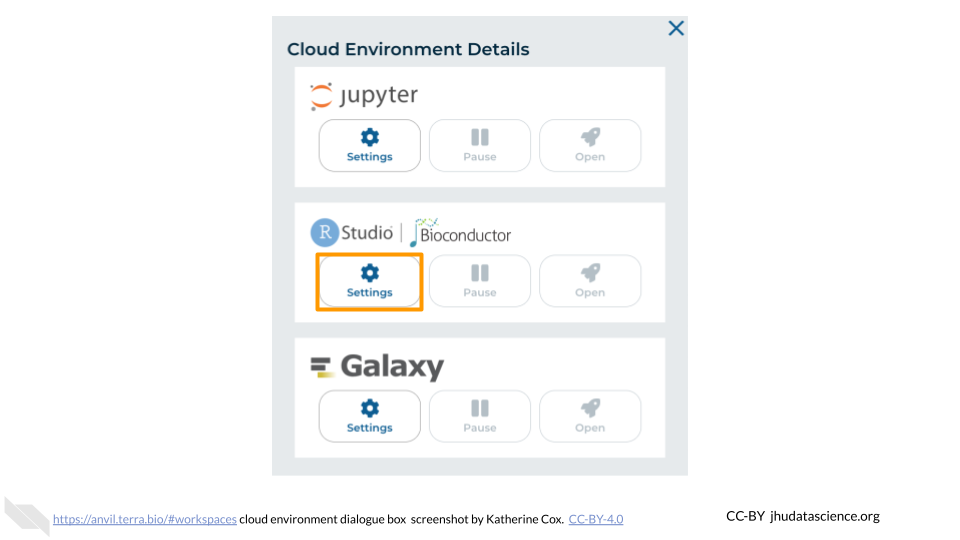
1. Open Terra - use a web browser to go to [anvil.terra.bio](https://anvil.terra.bio/)
2. In the drop-down menu on the left, navigate to “Workspaces”. Click the triple bar in the top left corner to access the menu. Click “Workspaces”.

* 

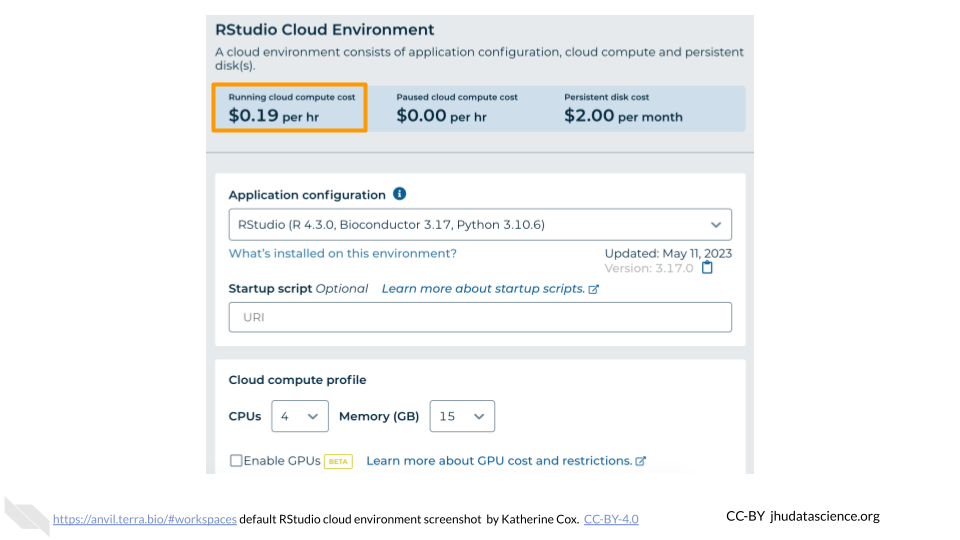
1. Click on the name of your Workspace. You should be routed to a link that looks like: https://anvil.terra.bio/#workspaces/<billing-project>/<workspace-name>.
2. Click on the cloud icon on the far right to access your Cloud Environment options. If you don’t see this icon, you may need to scroll to the right.

* 

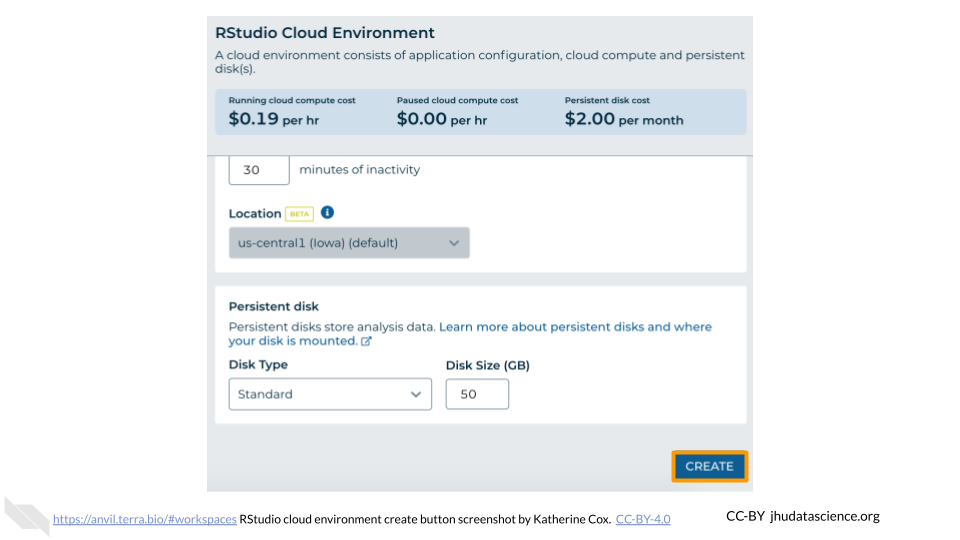
1. In the dialogue box, click the “Settings” button under RStudio.

* 

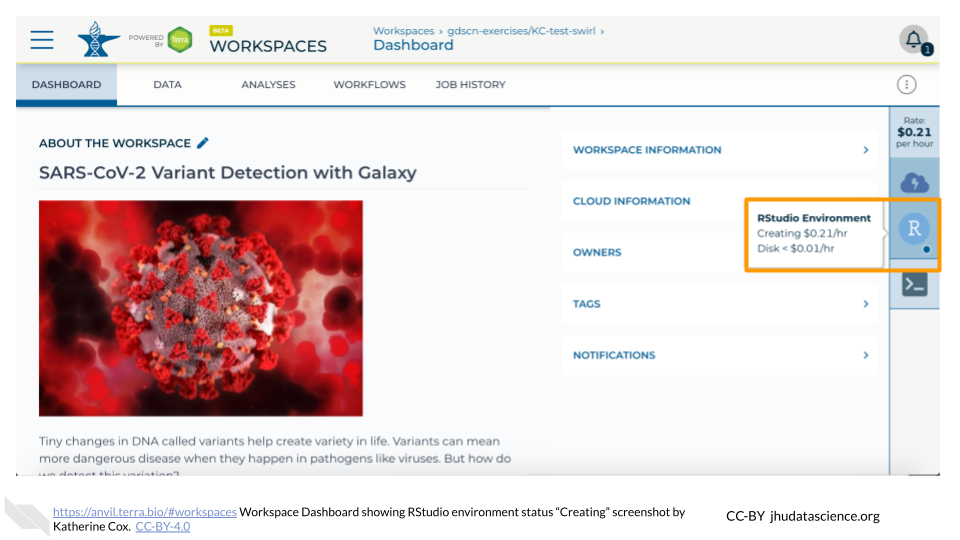
1. You will see some configuration options for the RStudio cloud environment, and a list of costs because it costs a small amount of money to use cloud computing.

* 

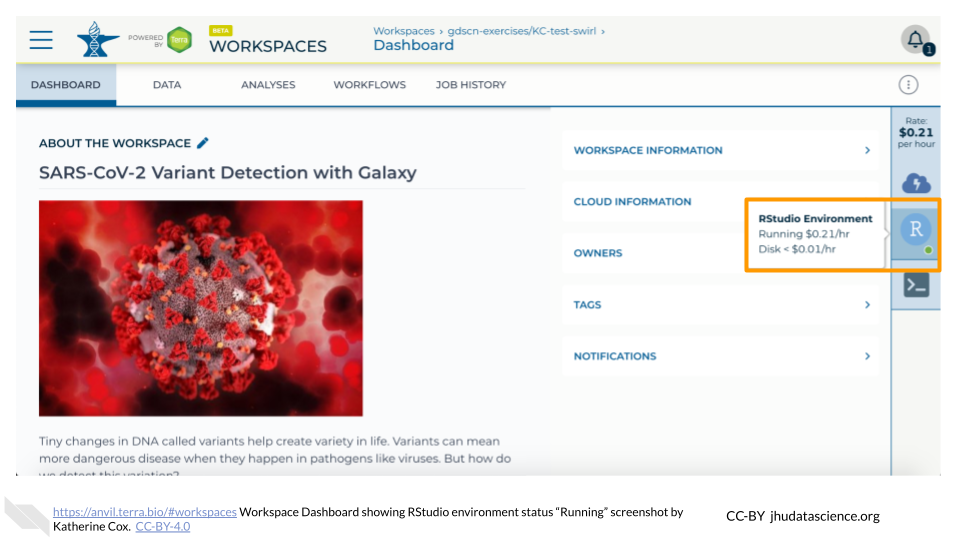
1. Configure any settings you need for your cloud environment. If you are uncertain about what you need, the default configuration is a reasonable, cost-conservative choice. It is fairly easy to increase your compute resources later, if needed, but harder to scale down. Scroll down and click the “CREATE” button when you are satisfied with your setup.

* 

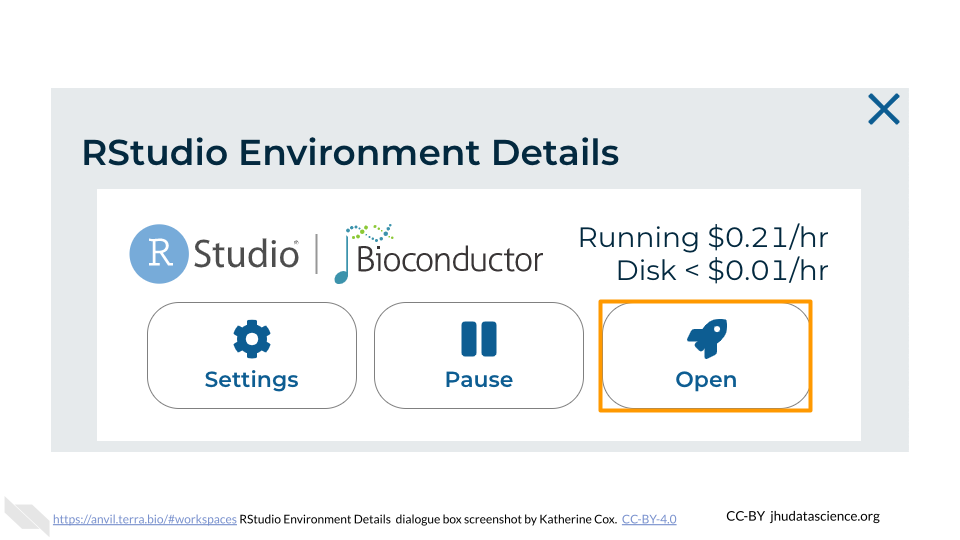
1. The dialogue box will close and you will be returned to your Workspace. You can see the status of your cloud environment by hovering over the RStudio icon. It will take a few minutes for Terra to request computers and install software.

* 

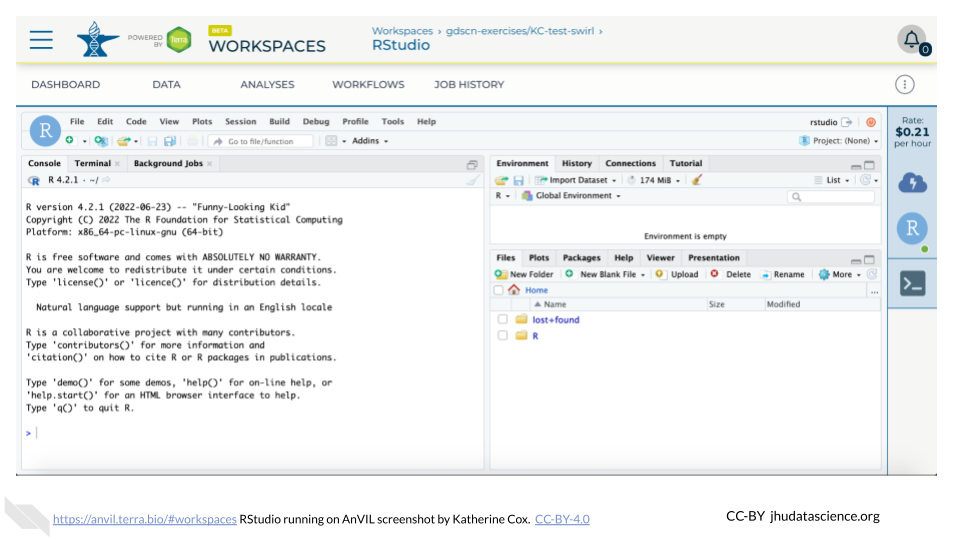
1. When your environment is ready, its status will change to “Running”. Click on the RStudio logo to open a new dialogue box that will let you launch RStudio.

* 

1. Click the launch icon to open RStudio. This is also where you can pause, modify, or delete your environment when needed.

* 

1. You should now see the RStudio interface with information about the version printed to the console.

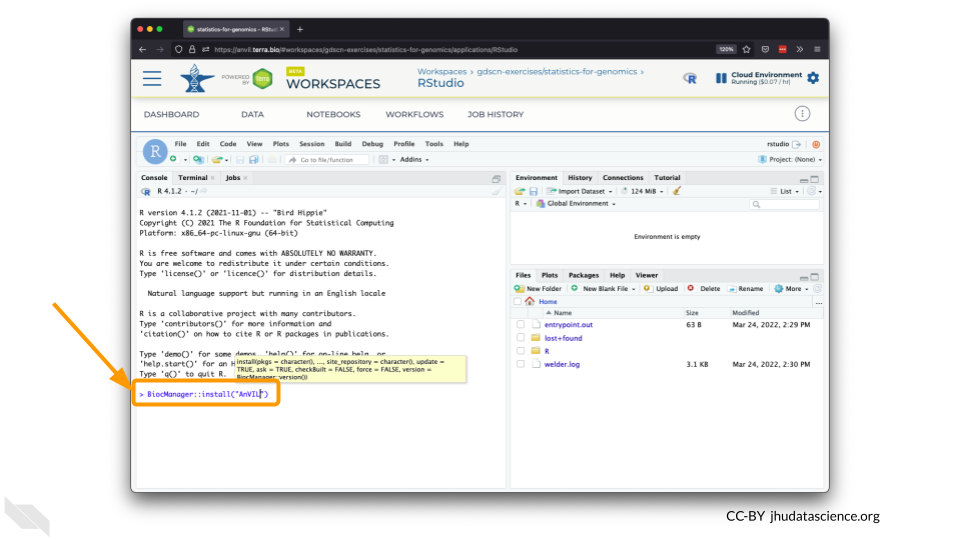
* 

#### 1.2.4.2 Touring RStudio

## Warning: Chunk option fig.alt is not supported for docx output  
## Warning: Chunk option fig.alt is not supported for docx output  
## Warning: Chunk option fig.alt is not supported for docx output  
## Warning: Chunk option fig.alt is not supported for docx output  
## Warning: Chunk option fig.alt is not supported for docx output  
## Warning: Chunk option fig.alt is not supported for docx output  
## Warning: Chunk option fig.alt is not supported for docx output  
## Warning: Chunk option fig.alt is not supported for docx output  
## Warning: Chunk option fig.alt is not supported for docx output  
## Warning: Chunk option fig.alt is not supported for docx output

Next, we will be using RStudio and the package Glimma to create interactive plots. See [this vignette](https://bioconductor.org/packages/release/bioc/vignettes/Glimma/inst/doc/limma_edger.html) for more information.

1. The Bioconductor team has created a very useful package to programmatically interact with Terra and Google Cloud. Install the AnVIL package. It will make some steps easier as we go along.

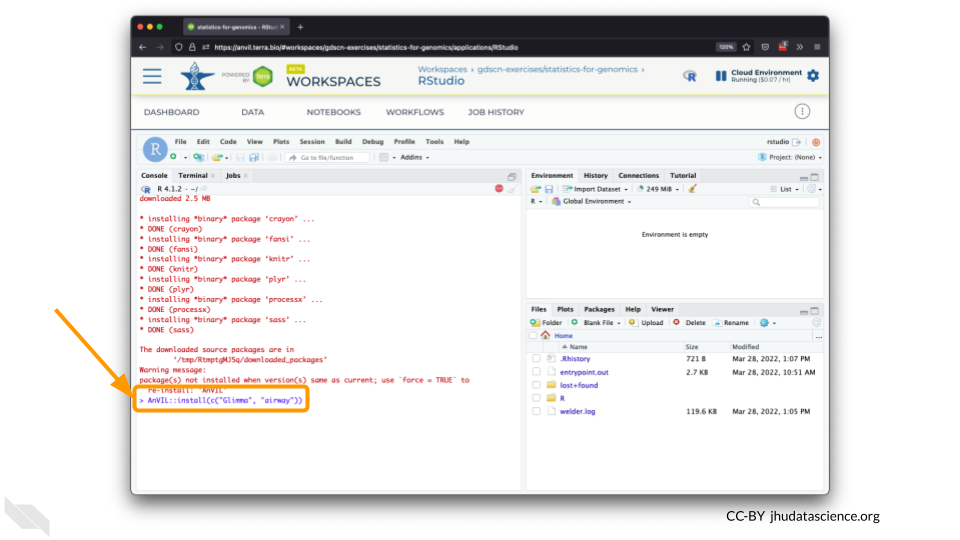
* 

1. You can now quickly install precompiled binaries using the AnVIL package’s install() function. We will use it to install the Glimma package and the airway package. The airway package contains a SummarizedExperiment data class. This data describes an RNA-Seq experiment on four human airway smooth muscle cell lines treated with dexamethasone.

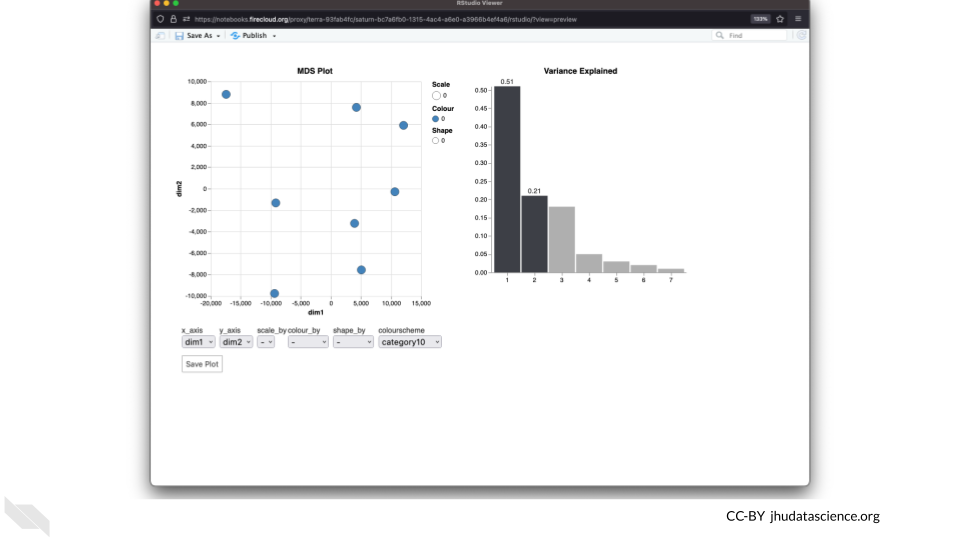
{Note: for some of the packages, you will have to install packaged from the CRAN repository, using the install.packages() function. The examples will show you which install method to use.}

![](resources/images/09-student\_guide\_files/figure-docx//1BLTCaogA04bbeSD1tR1Wt-mVceQA6FHXa8FmFzIARrg\_g11f12bc99af\_0\_56.png)

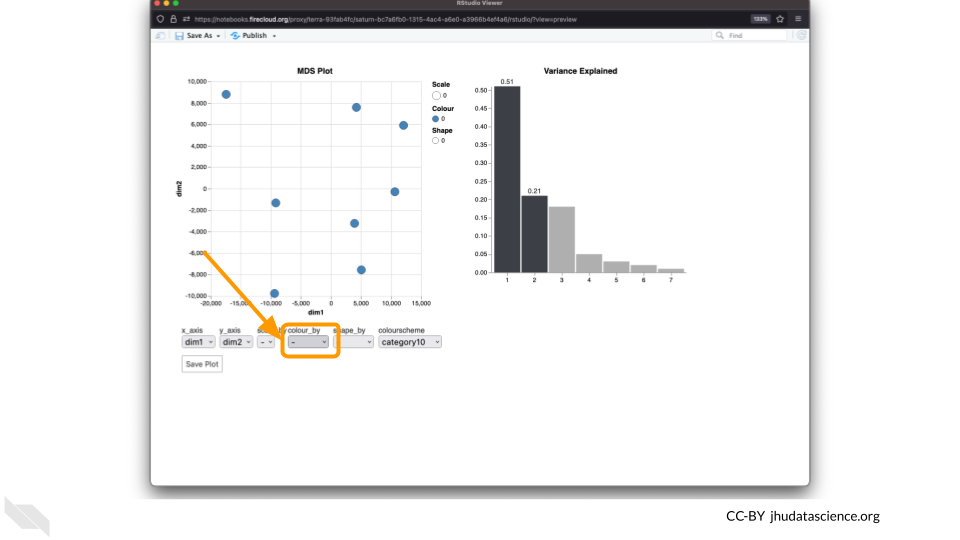
1. Load the example data.

* 

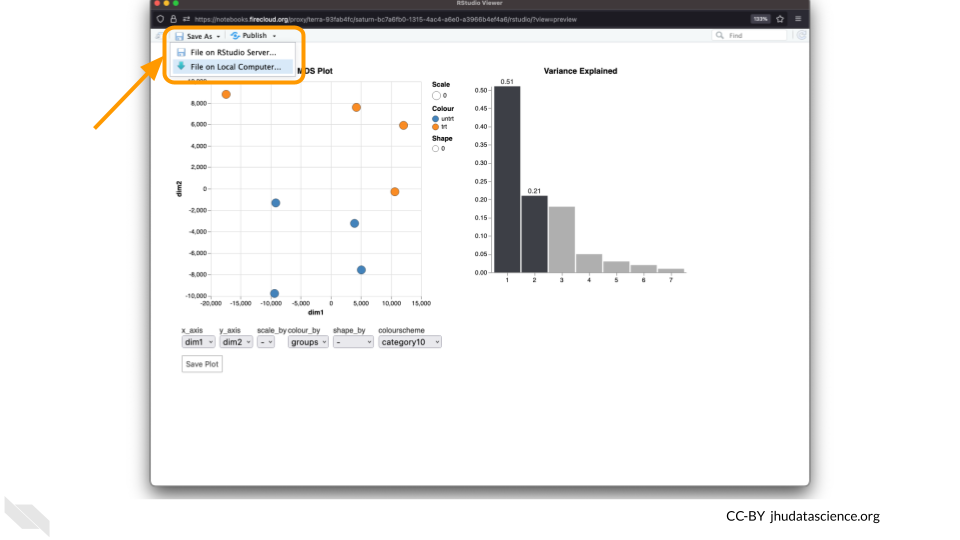
1. The multidimensional scaling (MDS) plot is frequently used to explore differences in samples. When this data is MDS transformed, the first two dimensions explain the greatest variance between samples, and the amount of variance decreases monotonically with increasing dimension. The following code will launch a new window where you can interact with the MDS plot.

* 

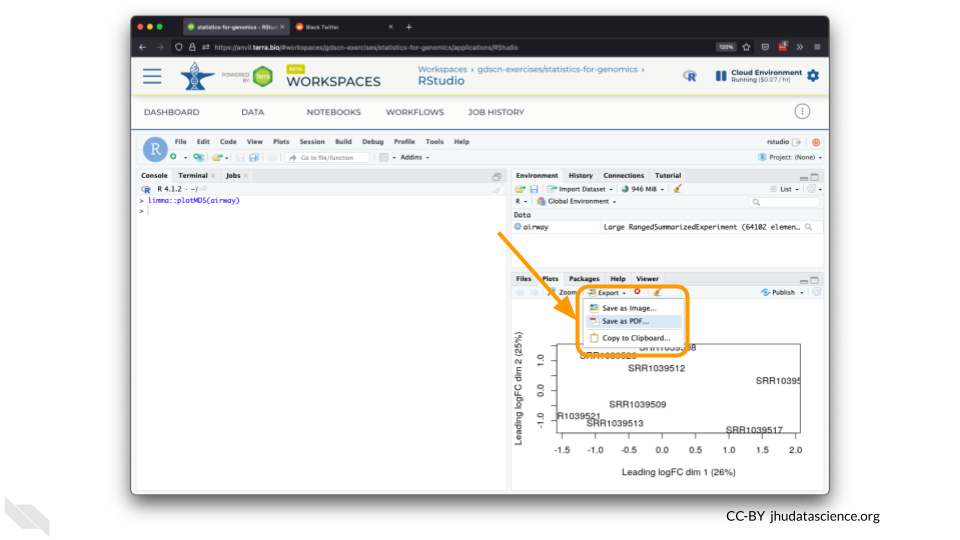
1. Change the colour\_by setting to “groups” so you can easily distinguish between groups. In this data, the “group” is the treatment.

* 

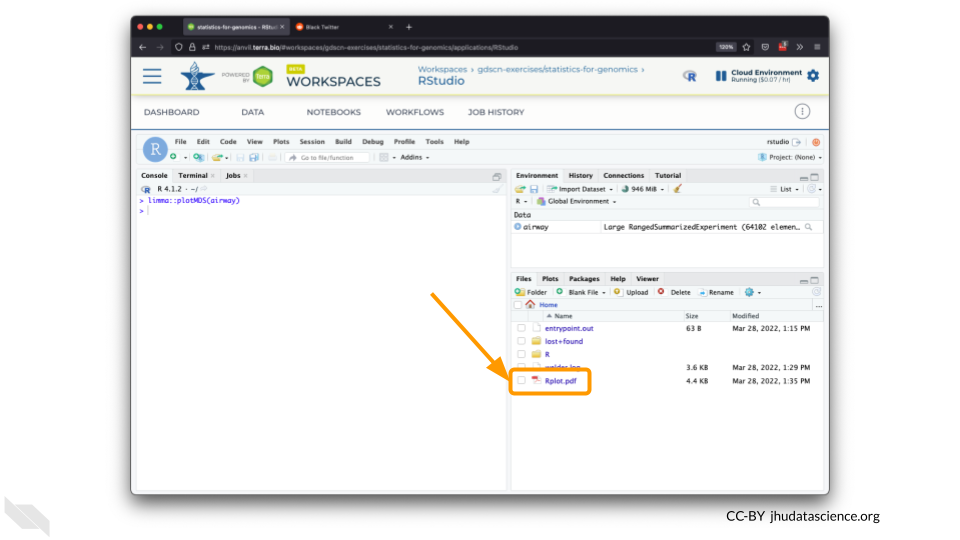
1. You can download the interactive html file by clicking on “Save As”.

* 

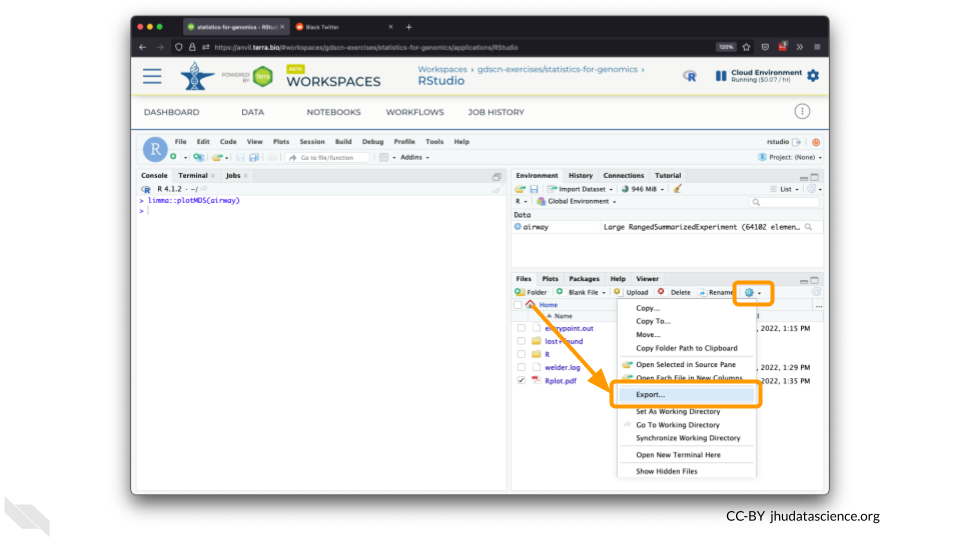
1. You can also download plots and other files created directly in RStudio. To download the following plot, click on “Export” and save in your preferred format to the default directory. This saves the file in your cloud environment.

* 

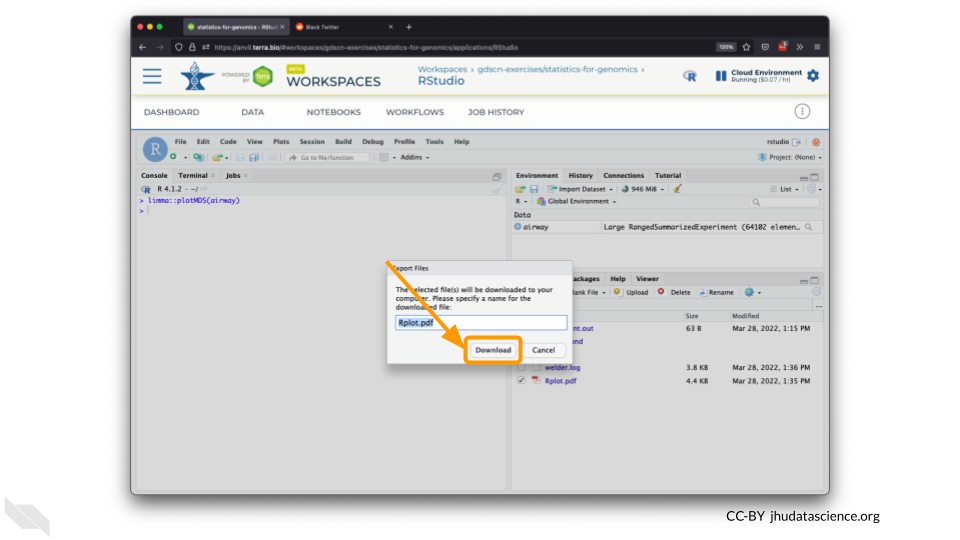
1. You should see the plot in the “Files” pane.

* 

1. Select this file and click “More” > “Export”

* 

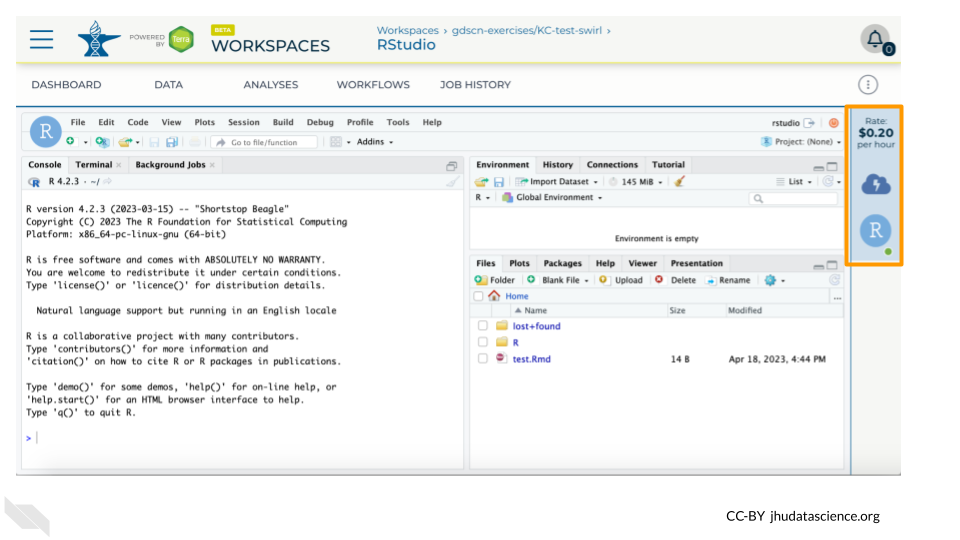
1. Select “Download” to save the file to your local machine.

* 

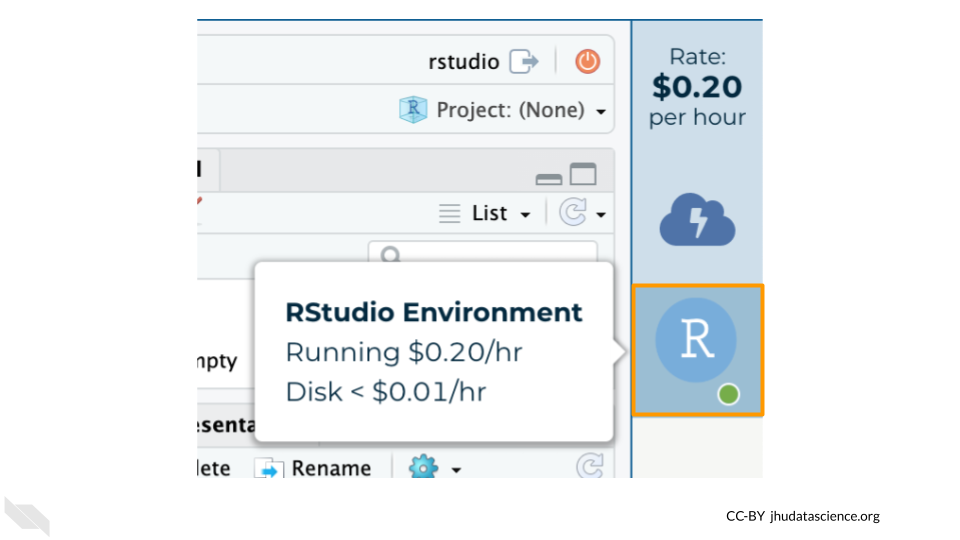
#### 1.2.4.3 Pausing RStudio

## Warning: Chunk option fig.alt is not supported for docx output  
## Warning: Chunk option fig.alt is not supported for docx output  
## Warning: Chunk option fig.alt is not supported for docx output  
## Warning: Chunk option fig.alt is not supported for docx output  
## Warning: Chunk option fig.alt is not supported for docx output  
## Warning: Chunk option fig.alt is not supported for docx output

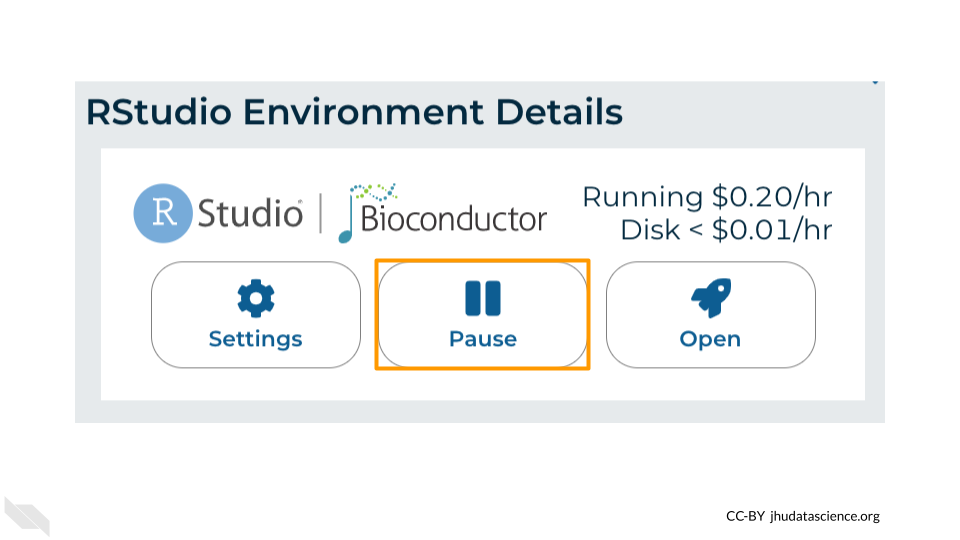
1. You can view costs and make changes to your cloud environments from the panel on the far right of the page. If you don’t see this panel, you may need to scroll to the right. Running environments will have a green dot, and paused environments will have an orange dot.

* 

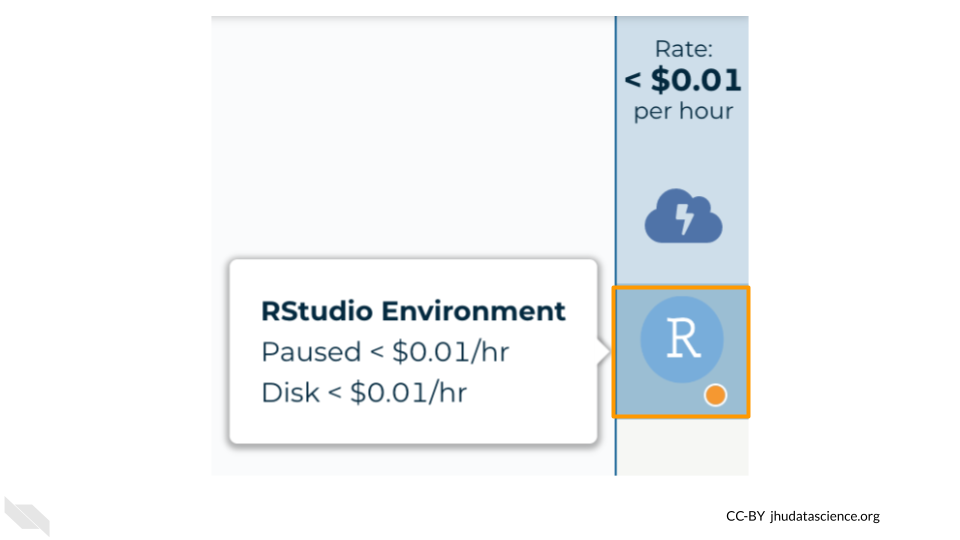
1. Hovering over the RStudio icon will show you the costs associated with your RStudio environment. Click on the RStudio icon to open the cloud environment settings.

* 

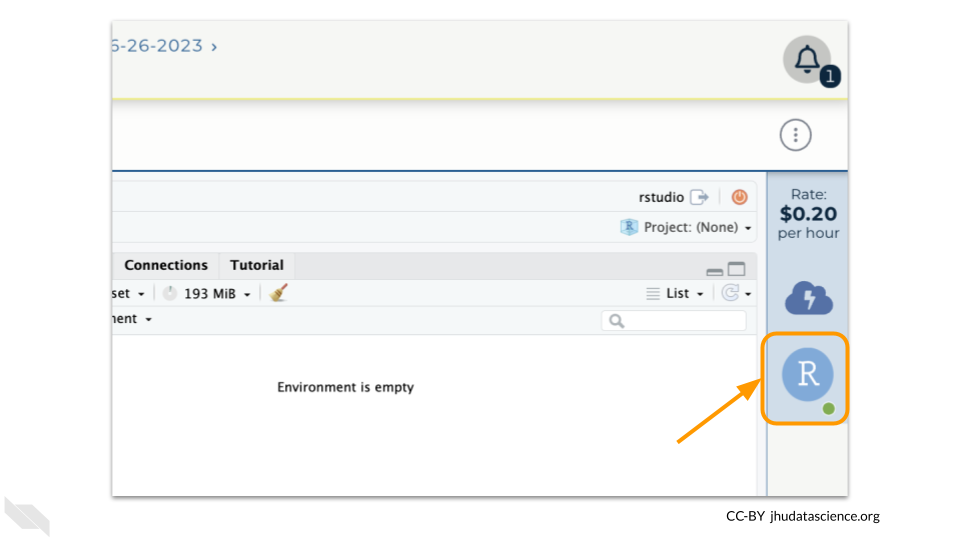
1. Click the Pause button to pause RStudio. This will take a few minutes.

* 

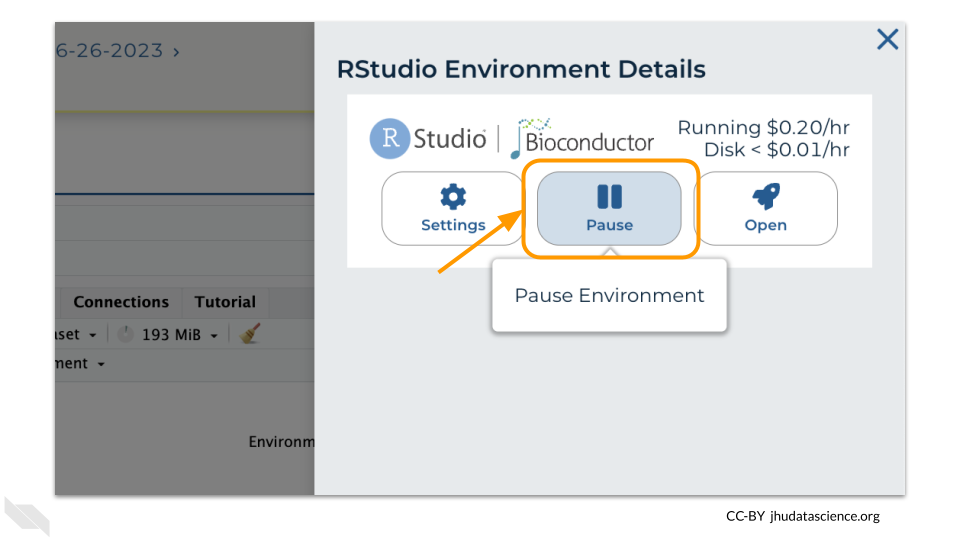
1. When the environment is paused, an orange dot will be displayed next to the RStudio icon. If you hover over the icon, you will see that it is paused, and has a small ongoing cost as long as it is paused. When you’re ready to resume working, you can do so by clicking the RStudio icon and clicking Resume.

* 

1. The right-hand side icon reminds you that you are accruing cloud computing costs. If you don’t see this icon, you may need to scroll to the right.

* 

1. You should minimize charges when you are not performing an analysis. You can do this by clicking on the RStudio icon and selecting “Pause”. This will release the CPU and memory resources for other people to use. Note that your work will be saved in the environment and continue to accrue a very small cost. This work will be lost if the cloud environment gets deleted. If there is anything you would like to save permanently, it’s a good idea to copy it from your compute environment to another location, such as the Workspace bucket, GitHub, or your local machine, depending on your needs.

* 

You can also pause your cloud environment(s) at <https://anvil.terra.bio/#clusters>.

## 1.3 Exercise One: Loading libraries in RStudio

Before we can start our analysis of how the SARS-CoV-2 variants are related to each other, we need to prepare the RStudio workspace and load the data.

R is an open-source statistical programming language and anyone can contribute to it. People have written programs in R to do a ton of different things, and they can make those programs (known as packages, or libraries) available to everyone. Generally, when someone has created a package they want to share, they will submit it to a repository, where anyone using R can download it.

For this lesson, we are using the [CRAN](https://cran.r-project.org/) (Comprehensive R Archive Network) repository. There are a series of servers around the world that store the up-to-date packages. When you open R, you can access those servers and download any package you want. If we are downloading a package that has been stored on CRAN, we use the command install.packages.

We need to install two packages: [ape](https://academic.oup.com/bioinformatics/article/20/2/289/204981) and [phangorn](https://academic.oup.com/bioinformatics/article/27/4/592/198887). Both of these packages were written specifically for phylogenetic analysis in R.

To install the packages, we type the following code into the RStudio console:

install.packages('ape')  
  
install.packages('phangorn')

Once you’ve downloaded a package, it will be saved on your computer (or, in the case of AnVIL, on your persistent disk space) so that you don’t have to download it again. Anytime you want to use the set of commands that are stored in a particular package, you’ll tell R to open the package with the library command.

Let’s open both packages now:

library(ape)  
library(phangorn)

You can verify that both packages have been loaded by looking at the Packages tab in the lower left-hand window of the RStudio interface. Packages that have been loaded are checked. You can search specifically for each package, or scroll down the entire list.

## 1.4 Exercise Two: Examining fasta files in RStudio

Now we need to retrieve the data. We’ll start by loading a type of data file called a fasta file. The fasta format is a common way to store sequences (either DNA or protein). Each sample in a fasta file has two sections. The sample ID and other descriptive information is on the first line (the description line). This line begins with either a > or a ;. The sample sequence is on the line immediately after the description. The sequence is written in standard IUCAC codes for either nucleic acids (for DNA sequence) or amino acids (for protein sequence). The sequence can also include unknown bases or gaps.

The fasta file we’re loading first contains the aligned sequences for the spike protein of 5 SARS-CoV-2 samples. This is what the top of the file looks like in a text editor:

We can load this file into RStudio using the read.FASTA command and save it as the object “spike.fasta”.

spike.fasta <- read.FASTA("sars\_spike\_protein.fasta")

After we’ve created an object in RStudio, we can get information about the object by typing the object’s name.

spike.fasta

## 5 DNA sequences in binary format stored in a list.  
##   
## All sequences of same length: 3827   
##   
## Labels:  
## alpha  
## beta  
## delta  
## gamma  
## Wuhan\_reference  
##   
## Base composition:  
## a c g t   
## 0.294 0.188 0.184 0.333   
## (Total: 19.14 kb)

Here we can see a summary of what this object contains, as well as how long the sequences are and the sequence names.

Notice that RStudio has saved the information in the fasta file as binary data. This means the sequence information has been converted from “ATCG” into something easier for RStudio to work with, but harder for humans to interpret.

The phangorn package uses a special data format called phyDat, which is derived from the fasta format. A phyDat object provides some additional information about the samples we upload.

spike.phydat <- read.phyDat("sars\_spike\_protein.fasta", format = "fasta")  
  
spike.phydat

## 5 sequences with 3827 character and 46 different site patterns.  
## The states are a c g t

Different site patterns refers to sites that differ between sequences. In this small dataset, 46 of the 3827 possible bases (characters) show differences among these SARS-CoV-2 samples.

We’ve been working with a dataset that contains the original SARS-CoV-2 sequence (the Wuhan reference sample), as well as samples of the alpha, beta, delta, and gamma variants. Another variant, the omicron variant, was first identified in late 2021 and quickly became a variant of concern. Let’s look at a dataset that contains additional omicron samples.

spike.omicron <- read.phyDat("sars\_spike\_protein\_omicron.fasta", format = "fasta")  
  
spike.omicron

## 9 sequences with 3827 character and 96 different site patterns.  
## The states are a c g t

QUESTIONS:

1. What is some information saved in a .fasta object that RStudio tells us that we don’t get from a .phyDat object?
2. How many omicron sequences are there in the second file (loaded into the object spike.omicron)?
3. Do you think there is more variability in the omicron sequences than in other variants (alpha, beta, delta, and gamma)? Why or why not?

## 1.5 Exercise Three: How are variants related to the original SARS-CoV-2 reference strain?

We’ll work with both the smaller spike protein dataset (saved as spike.phydat) and the spike protein dataset with omicron samples (saved as spike.omicron) in this activity.

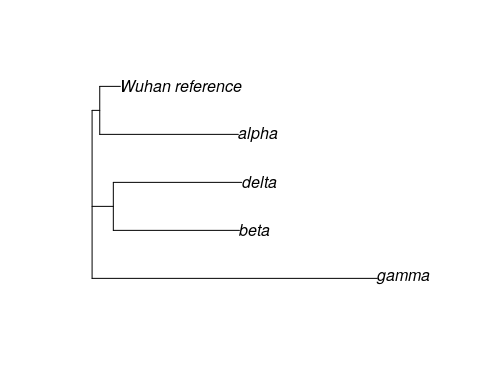
### 1.5.1 Building a neighbor-joining tree

We can use phyDat objects to build a type of phylogenetic tree called a neighbor-joining tree. We can do this in two steps in RStudio using the phangorn package.

dist.small <- dist.ml(spike.phydat)  
nj.small <- nj(dist.small)

The first command (dist.ml) converts the genetic data into a distance matrix, while the second command (nj) uses the matrix to group taxa based on the genetic distance between them. We plot the object created by the nj command to visualize our phylogenetic tree.

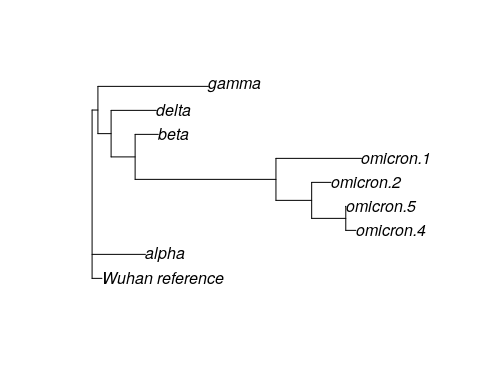
plot(nj.small)



When reading phylogenetic trees, it’s important to remember that relatedness is determined by the number of nodes between two taxa. The delta variant is most closely related to the beta variant because they share a common node (where the branches join together). We always look at common ancestors of clades, so we essentially read ancestry on a tree from right (where the taxa are) to left. Thus, the delta variant is more closely related to the beta variant because the delta branch connects to a node shared by the beta branch before it connects to a node shared with the gamma branch or the Wuhan reference/alpha variant branch.

Now let’s look at where the omicron sequences are in the SARS-CoV-2 tree.

dist.omicron <- dist.ml(spike.omicron)  
nj.omicron <- nj(dist.omicron)  
plot(nj.omicron)



QUESTIONS:

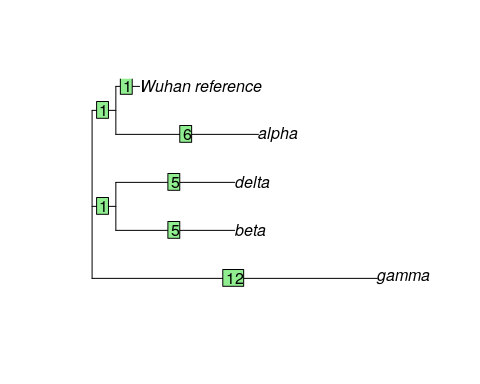
1. Which variant(s) is the most distantly related to the original Wuhan reference sequence?
2. Which variant shares the most recent common ancestor with the omicron variants? What does this mean for determining where the omicron variant came from?

### 1.5.2 Examining branch lengths

Now we will add branch lengths to our trees. Information about branch lengths are saved in our previous trees, but those branch lengths are equal to the number of substitutions per site, which can be difficult to interpret. Instead, we will look at trees where the branch lengths have been scaled so that each branch is equal to the total number of substitutions (changes in the DNA sequence).

First, let’s load and view a tree containing the five non-omicron sequences.

spike.tree\_small <- read.tree("sars\_tree\_small.tre")  
  
plot(spike.tree\_small)  
edgelabels(spike.tree\_small$edge.length)



The numbers in green boxes represent the number of DNA changes along a particular branch. For example, the Wuhan reference sequence is estimated to have only one DNA change from the spike protein sequence of the most recent common ancestor of all SARS-CoV-2 strains, while the alpha variant has 6 bases different from the most recent common ancestor. This also means the distance between Wuhan reference sequence and the alpha sample is 7 substitutions (1 substitution for the Wuhan sample + 6 substitutions for the alpha variant sample).

**Breakout Box: Learn more about phylogenetically informative sites**

You may be wondering why the total branch length changes (31) is different from the number of site patterns listed in the phyDat summary (46). This is because only some of the 46 site patterns are what we call *phylogenetically informative*.

Let’s look at a short example:

cat: GC**A** TT**C**

dog: GC**C** TT**C**

frog: GC**C** TT**A**

snake: GC**C** TT**A**

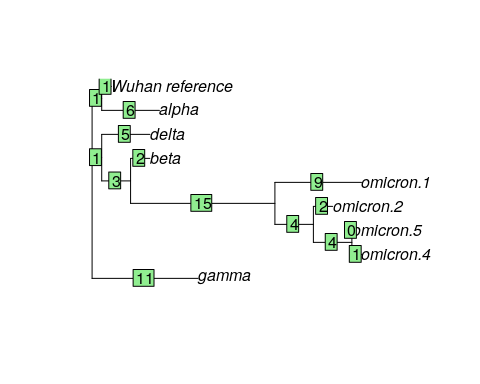
In this collection of sequences, there are 2 site patterns (both in bold), but only one is phylogenetically informative because it tells us about the relatedness between the species. While the cat sequence is different from the others at the third site, this doesn’t actually give us information about the relationships among the four species. The sixth site, however, tells us that the cat and dog share a mutation that frog and snake do not. We can interpret that to mean the cat and dog are more closely related to each other than either is to the frog or the snake. Thus, this particular site pattern gives us information we can use to build the phylogenetic tree.

QUESTIONS:

1. what is the branch length distance between the beta variant and the alpha variant?
2. What is the longest branch length in the tree? What does this mean for the number of mutations (compared to the Wuhan reference sequence) seen in that variant versus the others?

Now let’s load and view a tree containing all nine SARS-CoV-2 samples.

spike.tree\_full <- read.tree("sars\_tree\_full.tre")  
  
plot(spike.tree\_full)  
edgelabels(spike.tree\_full$edge.length)



What do these branch lengths tell us about the relationship among the SARS-CoV-2 variants?

QUESTIONS:

1. Did adding the omicron samples change the branch length distances between the original five samples? What is the branch length distance between the alpha and beta variants now?
2. What is the length of the branch connecting the omicron group to the rest of the tree
3. The covid vaccines were originally designed based on the Wuhan reference sequence of the spike protein. The immune system learns to recognize the spike protein from the vaccine and can identify and destroy any invading Covid-19 viruses. Using what you have learned about the phylogenetic tree of SARS-CoV-2 variants, can you explain why these initial vaccines were less effective at protecting against the omicron variants than they were against the delta variant?

## 1.6 Exercise Four: Do other protein-coding regions show us the same phylogenetic relationships?

The SARS-CoV-2 genome contains 6 protein-coding regions. So far, we’ve been working with genetic data from the region that codes for spike protein. (The spike protein is the part of the virus that sticks out to form the characteristic spikes on the outside of the SARS-CoV-2 virus.) However, you could also use any of the other 5 protein-coding regions for this exercise.

Let’s look at sequences from the region that codes for the membrane glycoprotein.

membrane.omicron <- read.phyDat("sars\_membrane\_protein\_omicron.fasta", format = "fasta")  
  
membrane.omicron

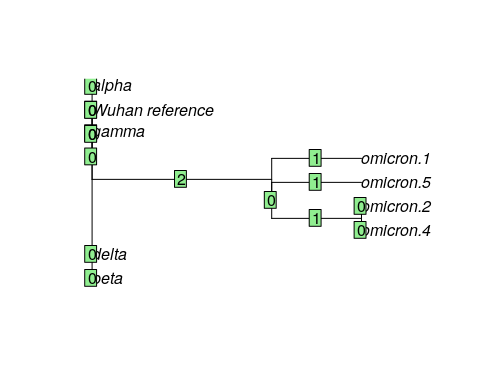
## 9 sequences with 670 character and 10 different site patterns.  
## The states are a c g t

QUESTIONS:

1. Based on the information from the phydat files for the spike protein dataset and the membrane protein dataset, is there the same amount of variation in each protein-coding region?
2. Do you think the spike protein dataset or the membrane protein dataset contains a greater number of phylogenetically-informative sites?

Now let’s look at a neighbor-joining tree built using the membrane protein dataset.

nj.membrane <- read.tree("sars\_membrane\_tree.tre")  
  
plot(nj.membrane)  
edgelabels(nj.membrane$edge.length)



QUESTIONS:

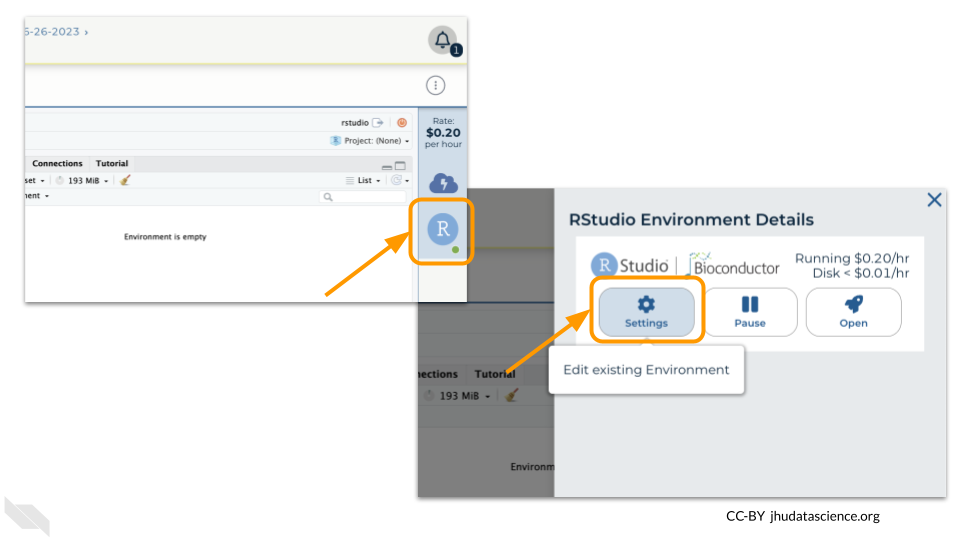
1. Is the tree built from the membrane protein data the same as the tree built from the spike protein data?
2. Why is a greater number of phylogenetically-informative sites better for tree building?

## 1.7 Wrap-up

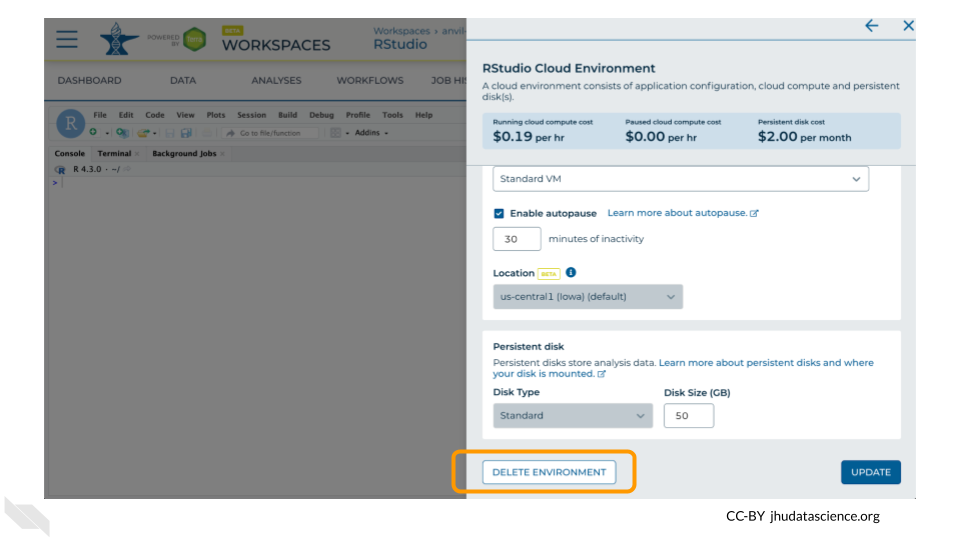
Once you are done with the activity, you’ll need to shut down your RStudio cloud environment. This frees up the cloud resources for others and minimizes computing cost. The following steps will delete your work, so make sure you are completely finished at this point. Otherwise, you will have to repeat your work from the previous steps.

## Warning: Chunk option fig.alt is not supported for docx output  
## Warning: Chunk option fig.alt is not supported for docx output  
## Warning: Chunk option fig.alt is not supported for docx output  
## Warning: Chunk option fig.alt is not supported for docx output

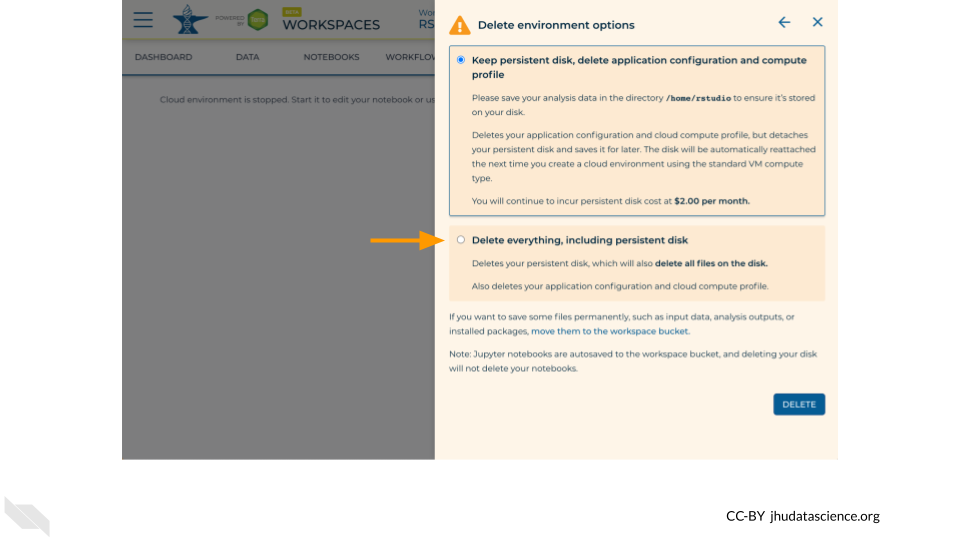
1. Pausing your cloud environment only temporarily stops your work. When you are ready to delete the cloud environment, click on the RStudio icon on the right-hand side and select “Settings”. If you don’t see this icon, you may need to scroll to the right.

* 

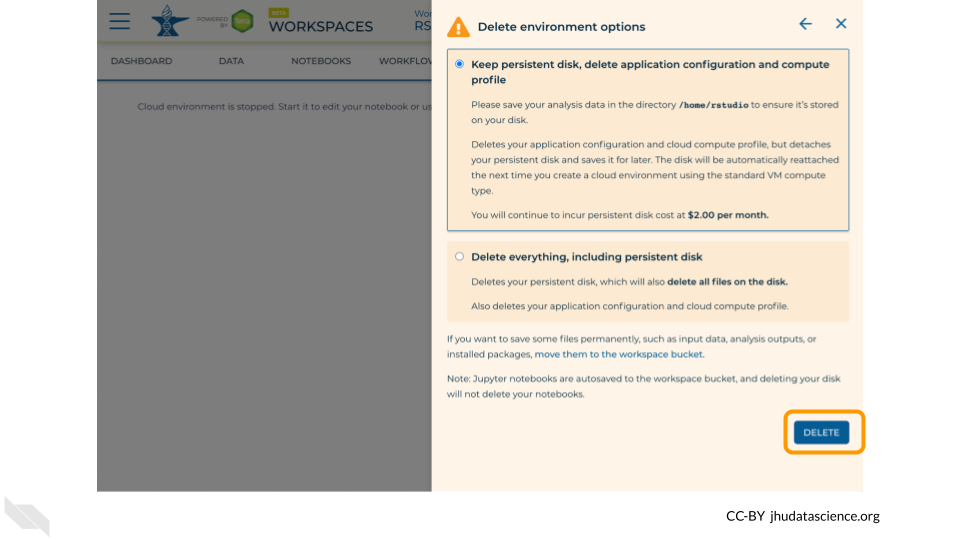
1. Click on “Delete Environment”.

* 

1. If you are certain that you do not need the data and configuration on your disk, you should select “Delete everything, including persistent disk”. If there is anything you would like to save, open the compute environment and copy the file(s) from your compute environment to another location, such as the Workspace bucket, GitHub, or your local machine, depending on your needs.

* 

1. Select “DELETE”.

* 

You can also delete your cloud environment(s) and disk storage at <https://anvil.terra.bio/#clusters>.

# About the Authors

These credits are based on our [course contributors table guidelines](https://github.com/jhudsl/OTTR_Template/wiki/How-to-give-credits).

| Credits | Names |
| --- | --- |
| **Pedagogy** |  |
| Lead Content Instructor(s) | Elizabeth Humphries |
| Content Editor(s)/Reviewer(s) | [Ava Hoffman](https://www.avahoffman.com/) |
| Content Directors | [Jeff Leek](https://jtleek.com/), [Frederick Tan](https://carnegiescience.edu/frederick-tan) |
| Content Consultants (General) | [Jeff Leek](https://jtleek.com/), Tiffany Rolle, [Frederick Tan](https://carnegiescience.edu/frederick-tan), [Carrie Wright](https://carriewright11.github.io/) |
| **Production** |  |
| Content Publisher | [Ira Gooding](https://publichealth.jhu.edu/faculty/4130/ira-gooding) |
| **Technical** |  |
| Template Publishing Engineers | [Candace Savonen](https://www.cansavvy.com/), [Carrie Wright](https://carriewright11.github.io/) |
| Publishing Maintenance Engineer | [Candace Savonen](https://www.cansavvy.com/) |
| Technical Publishing Stylists | [Carrie Wright](https://carriewright11.github.io/), [Candace Savonen](https://www.cansavvy.com/) |
| Package Developers ([Leanbuild]) | [John Muschelli](https://johnmuschelli.com/), [Candace Savonen](https://www.cansavvy.com/), [Carrie Wright](https://carriewright11.github.io/) |
| **Funding** |  |
| Funder | [National Human Genome Research Institute (NHGRI)](https://www.genome.gov/) |
| Funding Staff | Fallon Bachman, Jennifer Vessio, Emily Voeglein |

## ─ Session info ───────────────────────────────────────────────────────────────  
## setting value  
## version R version 4.3.2 (2023-10-31)  
## os Ubuntu 22.04.4 LTS  
## system x86\_64, linux-gnu  
## ui X11  
## language (EN)  
## collate en\_US.UTF-8  
## ctype en\_US.UTF-8  
## tz Etc/UTC  
## date 2025-05-07  
## pandoc 3.1.1 @ /usr/local/bin/ (via rmarkdown)  
##   
## ─ Packages ───────────────────────────────────────────────────────────────────  
## package \* version date (UTC) lib source  
## bookdown 0.41 2024-10-16 [1] CRAN (R 4.3.2)  
## cachem 1.0.8 2023-05-01 [1] RSPM (R 4.3.0)  
## cli 3.6.2 2023-12-11 [1] RSPM (R 4.3.0)  
## devtools 2.4.5 2022-10-11 [1] RSPM (R 4.3.0)  
## digest 0.6.34 2024-01-11 [1] RSPM (R 4.3.0)  
## ellipsis 0.3.2 2021-04-29 [1] RSPM (R 4.3.0)  
## evaluate 0.23 2023-11-01 [1] RSPM (R 4.3.0)  
## fastmap 1.1.1 2023-02-24 [1] RSPM (R 4.3.0)  
## fs 1.6.3 2023-07-20 [1] RSPM (R 4.3.0)  
## glue 1.7.0 2024-01-09 [1] RSPM (R 4.3.0)  
## htmltools 0.5.7 2023-11-03 [1] RSPM (R 4.3.0)  
## htmlwidgets 1.6.4 2023-12-06 [1] RSPM (R 4.3.0)  
## httpuv 1.6.14 2024-01-26 [1] RSPM (R 4.3.0)  
## knitr 1.48 2024-07-07 [1] CRAN (R 4.3.2)  
## later 1.3.2 2023-12-06 [1] RSPM (R 4.3.0)  
## lifecycle 1.0.4 2023-11-07 [1] RSPM (R 4.3.0)  
## magrittr 2.0.3 2022-03-30 [1] RSPM (R 4.3.0)  
## memoise 2.0.1 2021-11-26 [1] RSPM (R 4.3.0)  
## mime 0.12 2021-09-28 [1] RSPM (R 4.3.0)  
## miniUI 0.1.1.1 2018-05-18 [1] RSPM (R 4.3.0)  
## pkgbuild 1.4.3 2023-12-10 [1] RSPM (R 4.3.0)  
## pkgload 1.3.4 2024-01-16 [1] RSPM (R 4.3.0)  
## profvis 0.3.8 2023-05-02 [1] RSPM (R 4.3.0)  
## promises 1.2.1 2023-08-10 [1] RSPM (R 4.3.0)  
## purrr 1.0.2 2023-08-10 [1] RSPM (R 4.3.0)  
## R6 2.5.1 2021-08-19 [1] RSPM (R 4.3.0)  
## Rcpp 1.0.12 2024-01-09 [1] RSPM (R 4.3.0)  
## remotes 2.4.2.1 2023-07-18 [1] RSPM (R 4.3.0)  
## rlang 1.1.4 2024-06-04 [1] CRAN (R 4.3.2)  
## rmarkdown 2.25 2023-09-18 [1] RSPM (R 4.3.0)  
## sessioninfo 1.2.2 2021-12-06 [1] RSPM (R 4.3.0)  
## shiny 1.8.0 2023-11-17 [1] RSPM (R 4.3.0)  
## stringi 1.8.3 2023-12-11 [1] RSPM (R 4.3.0)  
## stringr 1.5.1 2023-11-14 [1] RSPM (R 4.3.0)  
## urlchecker 1.0.1 2021-11-30 [1] RSPM (R 4.3.0)  
## usethis 2.2.3 2024-02-19 [1] RSPM (R 4.3.0)  
## vctrs 0.6.5 2023-12-01 [1] RSPM (R 4.3.0)  
## xfun 0.48 2024-10-03 [1] CRAN (R 4.3.2)  
## xtable 1.8-4 2019-04-21 [1] RSPM (R 4.3.0)  
## yaml 2.3.8 2023-12-11 [1] RSPM (R 4.3.0)  
##   
## [1] /usr/local/lib/R/site-library  
## [2] /usr/local/lib/R/library  
##   
## ──────────────────────────────────────────────────────────────────────────────

# 2 References

All sequences were downloaded from NCBI’s GenBank. We used sequence data from the following accession numbers: OX315675.1 (omicron.2), OP093374.1 (omicron.4), OP093373.1 (omicron.5), OK091006.1 (delta), NC\_045512.2 (Wuhan\_reference), OX003129.1 (beta), OX315743.1 (omicron.1), OW998592.1 (alpha), MW911054.1 (gamma).

1. Benson DA, Cavanaugh M, Clark K, Karsch-Mizrachi I, Lipman DJ, Ostell J, Sayers EW. GenBank. Nucleic Acids Res. 2013 Jan;41(Database issue):D36-42. doi: 10.1093/nar/gks1195. Epub 2012 Nov 27. PMID: 23193287; PMCID: PMC3531190.
2. Cella E, Benedetti F, Fabris S, Borsetti A, Pezzuto A, Ciotti M, Pascarella S, Ceccarelli G, Zella D, Ciccozzi M, Giovanetti M. SARS-CoV-2 Lineages and Sub-Lineages Circulating Worldwide: A Dynamic Overview. Chemotherapy. 2021;66(1-2):3-7. doi: 10.1159/000515340. Epub 2021 Mar 18. PMID: 33735881; PMCID: PMC8089399.
3. Paradis E, Claude J, Strimmer K. APE: Analyses of Phylogenetics and Evolution in R language. Bioinformatics. 2004 Jan 22;20(2):289-90. doi: 10.1093/bioinformatics/btg412. PMID: 14734327.
4. Schliep KP. phangorn: phylogenetic analysis in R. Bioinformatics. 2011 Feb 15;27(4):592-3. doi: 10.1093/bioinformatics/btq706. Epub 2010 Dec 17. PMID: 21169378; PMCID: PMC3035803.

SARS-CoV-2 image credit: Davian Ho for the [Innovative Genomics Institute](https://innovativegenomics.org/free-covid-19-illustrations/)