New England
Pathogen Genomics Center of Excellence

Coding framework for applying the *phylosamp* sample size calculation framework for genomic surveillance of viral variants

Authors: Carolyn Casiello[1], Zachary Thompson[2]

Last updated: 19 November 2025

I. Purpose and Scope

This document provides a coding example of variant sample size analysis for surveillance of SARS-CoV-2 in Massachusetts and is intended to be referenced in conjunction with the accompanying Methods and Example Outputs documents. This code assumes users have already generated variant prevalence estimates per time period of interest.

II. Data Structure

A single csv containing the variant prevalence estimates for each time period of interest (e.g., weekly, biweekly, monthly), along with sample size, is required. These data should be structured to be a row for each variant per time period of interest and described by four columns containing variant name, prevalence estimate, the time period of interest, and number of successfully sequenced samples from that time period. The names of the columns should be as follows:

- **variant**: character column with the variant name
- **prevalence**: numeric date in decimal format representing the prevalence estimate of that variant for that time period
- **date**: date variable representing the time period of interest (e.g., weekly end date)
- **n**: numeric variable of the total number of successfully sequenced samples in the period of interest

III. Code Instructions

Step 1. Get sample size targets

Step 1a. Load the necessary libraries.

```
library(dplyr)
library(readxl)
library(parsedate)
library(phylosamp)
```

[1] Massachusetts Department of Public Health, Division of Surveillance, Analytics, and Informatics
[2] Massachusetts Department of Public Health, Division of Sequencing, Bioinformatics, and Capacity Expansion

Step 1b. Import the data, making sure to change the file name to the proper name.

```
variants <- read.csv("filename.csv",
                     stringsAsFactors=F, as.is=TRUE)
```

Step 1c. R reads date variables in as characters so we can use the parse_date() function to theoretically take any date and transform it into the desired format of year-month-day. In this example, the time period of interest is weekly and the date variable represents the week end date.

```
variants$date <- as.Date(parse_date(variants$date),
                         format="%Y-%m-%d")
```

Step 1d. Calculate the sample size target if the detection ratio is 1 (just as likely to detect the variant of interest compared to remaining variants).

```
nC1 <- ceiling(vartrack_samplesize_detect(p_v1=0.03,
                                          prob=0.8,
                                          omega=1,
                                          c_ratio=1,
                                          sampling_freq="xsect"))
```

Step 1e. Calculate the sample size target if the detection ratio is 0.5 (half as likely to detect the variant of interest compared to remaining variants).

```
nC0.5 <- ceiling(vartrack_samplesize_detect(p_v1=0.03,
                                            prob=0.8,
                                            omega=1,
                                            c_ratio=0.5,
                                            sampling_freq="xsect"))
```

Step 2. Get probability of detection for each time period and compare to target n

Step 2a. Using the successfully sequenced sample size, calculate the probability of detection if the detection ratio is 1 (just as likely to detect the variant of interest compared to remaining variants).

```
variants$probC1 <- vartrack_prob_detect(p_v1=0.03,
                                        n=variants$n,
                                        omega=1,
                                        c_ratio=1,
                                        sampling_freq="xsect")
```

Step 2b. Using the successfully sequenced sample size, calculate the probability of detection if the detection ratio is 0.5 (half as likely to detect the variant of interest compared to remaining variants).

```
variants$probC0.5 <- vartrack_prob_detect(p_v1=0.03,
                                          n=variants$n,
                                          omega=1,
                                          c_ratio=0.5,
                                          sampling_freq="xsect")
```

Step 2c. This step adds the target sample sizes for detection ratios of 1 and 0.5 to the original imported data.

```
variants$nC1 <- rep(nC1, times=nrow(variants))
variants$nC0.5 <- rep(nC0.5, times=nrow(variants))
```

Step 2d. Calculate the sample size increase(s), if any, needed to achieve targets for detection ratios of 1 and 0.5.

```
variants$nIncreaseC1 <- ifelse((variants$nC1 - variants$n)>0,
                               paste0("+", variants$nC1 - variants$n>0),
                               "No change")
variants$nIncreaseC0.5 <- ifelse((variants$nC0.5 - variants$n)>0,
                                 paste0("+", variants$nC0.5 - variants$n>0),
                                 "No change")
```

Step 3. Get Wald 95% CIs

Step 3a. Function for 95% Wald confidence intervals.

```
wald <- function(estimate, n) {
  # Error calculation for 95% confidence based on your sample size and point
estimate
  error <- 1.96*(sqrt((1-estimate)/(n*estimate)))

  # This is the smallest value of the 95% interval (rounded to 0 if <0)
  min <- ifelse(round((estimate-estimate*error), 3)<0,
                0,
                round((estimate-estimate*error), 3))

  # This is the largest value of the 95% interval (rounded to 1 if >1)
  max <- ifelse(round((estimate+estimate*error), 3)>1,
                1,
                round((estimate+estimate*error), 3))

  # Creating display variable for the point estimate and 95% CI [XX (XX-XX)].
  display <- paste0(round(estimate, 3), " (", min, "-", max, ")")

  return(display)
}
```

Step 3b. Now applying the above CI function to the imported data, which will create an extra column with the CI data.

```
variants$ci <- wald(variants$prevalence, variants$n)
```