Supplemental: Simulation code for Antibodies to SARS-CoV-2 in All of Us Research Program Participants, January 2-March 18, 2020

David Schlueter, PhD

4/19/2021

The simulation is as follows.

Step 1. Generate an underlying true population of size 1000000 with disease prevalence = 9/24079 (what we observed in the sample).

Step 2. Generate test results of first stage: Among true positives generated in Step 1, flip a coin with probability of positive test = estimated sensitivity of Abbott testing (more specifically, we use the LOWER BOUND of the CI). Among true negatives generated in Step 1, flip a coin with probability of positive test = 1 - estimated specificity of Abbott testing (more specifically, we use the LOWER BOUND of the CI).

Step 3. Among those that tested positive in step 2. we next perform the EI test. Specifically, among the true positives, we flip a coin with probability of positive test = estimated sensitivity of EI test (more specifically, we use the LOWER BOUND of the CI). Among true negatives generated in Step 1, flip a coin with probability of positive test = 1 - estimated specificity of EI (more specifically, we use the LOWER BOUND of the CI).

Step 4. At the end of step 3, we look at the individuals who tested positive both on Abbott and EI and find the proportion who were truly disease negative. This is our trial-specific estimated probability of disease negative given test positive on both (i.e false positives).

We repeat steps 2 to 4 1000 times, then plug these probabilities into the binomial cdf to get the probabilities of at least X for the 9 that were positive on both Abbott and EI.

```
set.seed(1234)
trials<-1000
# using the prevalence from the samples
prevs<-9/24079
population<-1000000

###
# 1. Abbott, 2. EI
worst_case_sens<-c(.966, .835)
worst_case_spec<-c(.988, .991)

## Now put into list

sens<-list()
sens[["worst"]]<-worst_case_sens

spec<-list()
spec[["worst"]]<-worst_case_spec
```

Sequential Testing Simulation under worst case sens/specificity

```r
all_results<-NULL
for(prev in prevs){
  #prev = prevs[1]
  print("prev:")
  print(prev)
  truth<-c(rep(1, floor(population*prev)), rep(0, population-floor(population*prev)))
  df_results<-data.frame(id = 1:population,truth = truth)

  ## first calculate the probability of finding a true negative when you scored 2 positi
ves
  for(case in c("worst")){
    #case = "worst"
    print(case)
    # abbott
    # sens_1 = P(T1+|D+)
    sens_1 = sens[[case]][1]
    print(sens_1)
    #ei
    # sens_2 = P(T2+|D+)
    sens_2 = sens[[case]][2]
    print(sens_2)
    #abbott
    # spec_1 = P(T1-|D-)
    spec_1 =spec[[case]][1]
    print(spec_1)
    #ei
    # spec_2 = P(T2-|D-)
    spec_2 =spec[[case]][2]
    print(spec_2)


    for(trial in 1:trials){
      # perform test on positives
      positives<-subset(df_results, truth == 1)

      positives$test_1<-rbinom(n = nrow(positives), size =1, p=sens_1)
      # perform test on negatives
      negatives<-subset(df_results, truth == 0)
      negatives$test_1<-rbinom(n = nrow(negatives), size =1, p=1-spec_1)
      # stack
      df_full<-rbind(positives,negatives)

      # calculate probability that a given positive on both is disease negative P(D-|T1+
&T2+)
      p_disease_neg_pos_both<-NA
      ## First round: positive abbott
      positive_abb<-subset(df_full, test_1==1)

      ## now perform test 2
      ## true positives
      ## among the individuals who were truly positive, simulate with probability sensit
ivity EI
      positive_abb_true_pos<-positive_abb[positive_abb$truth==1, ]
      positive_abb_true_pos$test_2<-rbinom(n = nrow(positive_abb_true_pos), size =1, p=s
ens_2)

      ## True negatives
      ## among the folks who were truly negative, simulate a positive with probability 1
-specificity EI
      positive_abb_true_neg<-positive_abb[positive_abb$truth==0, ]
      positive_abb_true_neg$test_2<-rbinom(n = nrow(positive_abb_true_neg), size =1, p=1
-spec_2)
      #stack
      positive_abb_second_round<-rbind(positive_abb_true_pos,positive_abb_true_neg)

      positive_abb_second_round$both<-(positive_abb_second_round$test_1==1)&(positive_ab
b_second_round$test_2==1)
      # Now positive if EI also positive, otherwise negative.
      positive_abb_second_round$overall<-(positive_abb_second_round$test_2==1)+0
      # calculate specificity and sensitivity for this run (sequential)
      sens_seq<-mean(subset(positive_abb_second_round, truth==1)$overall)

      spec_seq<-mean(subset(positive_abb_second_round, truth==0)$overall==0)

      # now calc the probability in sequential #P(D-|T1+&T2+)
      positive_both_seq<- subset(positive_abb_second_round, both == TRUE)
      n_both_pos<-nrow(positive_both_seq)
      #View(positive_both_seq)
      #P(D-|T1+&T2+)
      n_neg_both_pos<-sum(positive_both_seq$truth == 0)
      p_disease_neg_pos_both_seq<-mean(positive_both_seq$truth == 0)
      res<-c(case, trial, prev, p_disease_neg_pos_both,p_disease_neg_pos_both_seq,sens_s
eq,spec_seq,n_both_pos,n_neg_both_pos)
      all_results<-rbind(all_results, res)
    }

  }

}
```

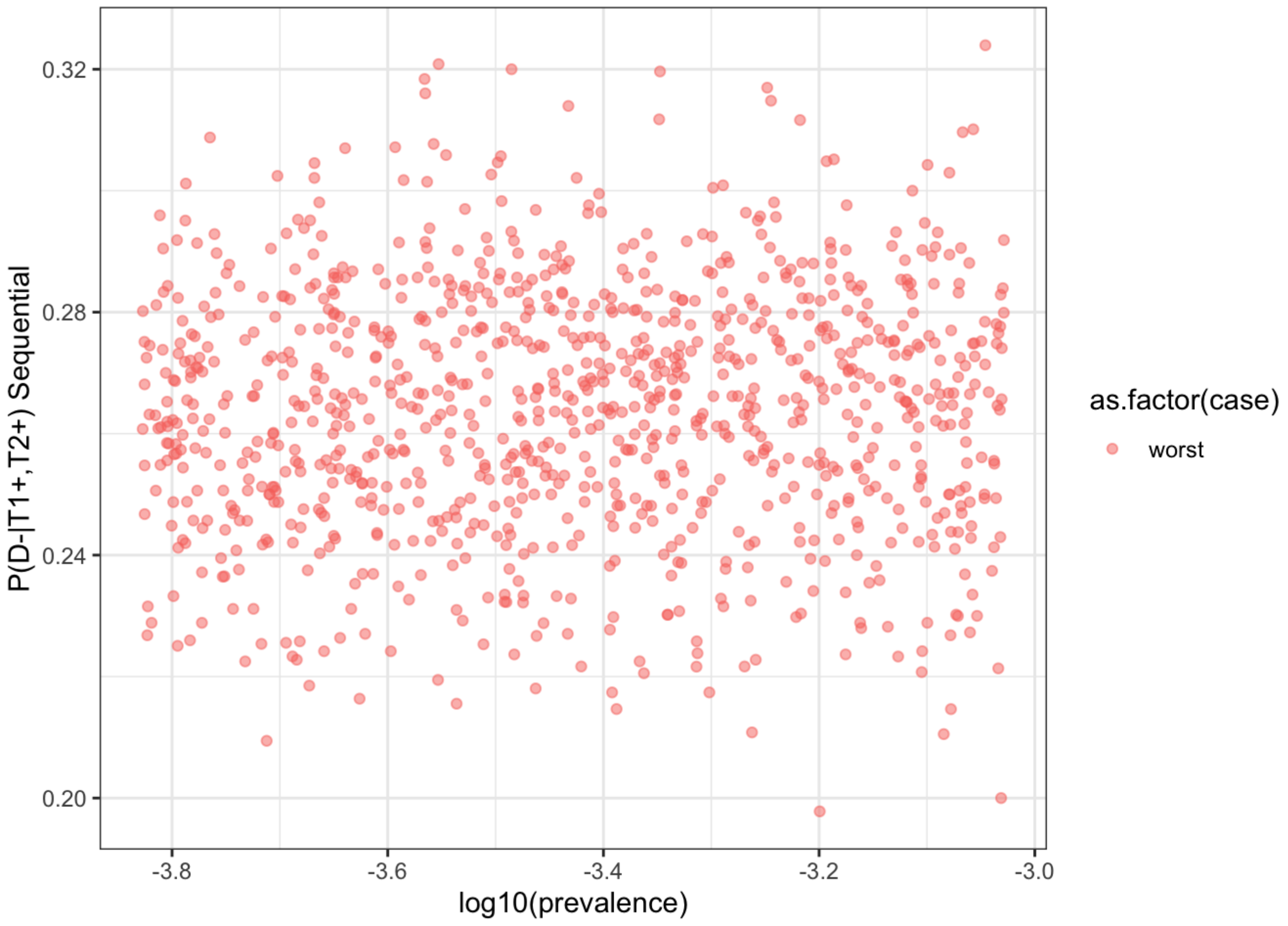
```
## [1] "prev:"
## [1] 0.0003737697
## [1] "worst"
## [1] 0.966
## [1] 0.835
## [1] 0.988
## [1] 0.991
```

```
# now name appropriately
all_results_df<-as.data.frame(all_results)
names(all_results_df)<-c("case", "trial","prev","prob", "prob_seq","sens_seq", "spec_se
q", "n_both_pos","n_neg_both_pos")
```

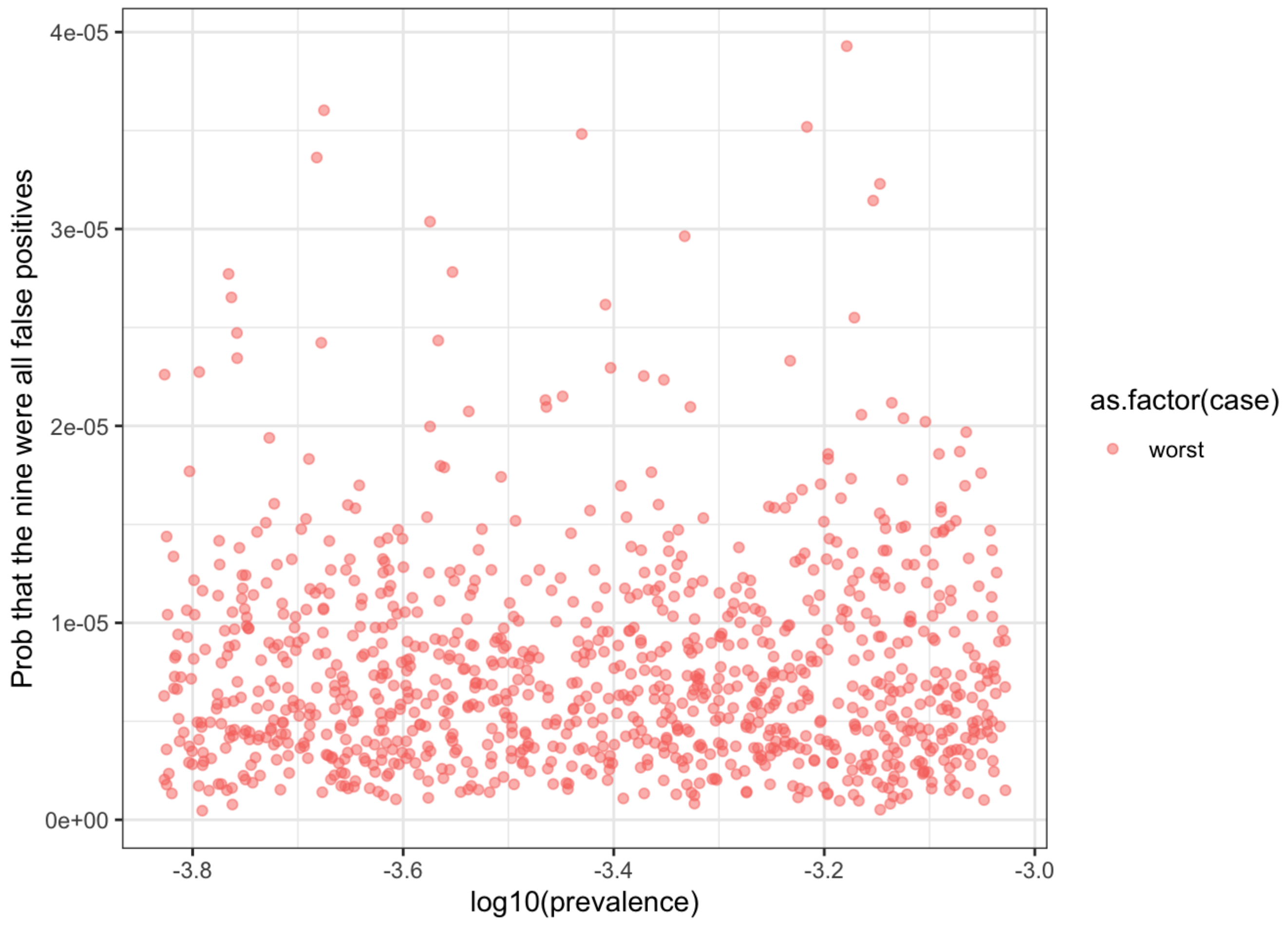
```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.0.2
```

```
ggplot(data=all_results_df, aes(x=log10(as.numeric(prev)), y = as.numeric(prob_seq), col
or = as.factor(case)))+geom_jitter(alpha=0.5)+
  theme_bw()+xlab("log10(prevalence)")+ylab("P(D-|T1+,T2+) Sequential")
```



```
ggplot(data=all_results_df, aes(x=log10(as.numeric(prev)), y = as.numeric(prob_seq)^9, c
olor = as.factor(case)))+geom_jitter(alpha=0.5)+
 theme_bw()+xlab("log10(prevalence)")+ylab("Prob that the nine were all false positives"
)
```



```r
library(tidyverse)
```

```
## ── Attaching packages ───────────────────────────────────────────────────────────────
─────────── tidyverse 1.3.0 ──
```

```
## ✓ tibble  3.0.3     ✓ dplyr   1.0.2
## ✓ tidyr   1.1.1     ✓ stringr 1.4.0
## ✓ readr   1.3.1     ✓ forcats 0.5.0
## ✓ purrr   0.3.4
```

```
## Warning: package 'tidyr' was built under R version 4.0.2
```

```
## Warning: package 'dplyr' was built under R version 4.0.2
```

```
## ── Conflicts ────────────────────────────────────────────────────────────────────────
───── tidyverse_conflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
all_results_df$prob_all<-as.numeric(all_results_df$prob_seq)^9

all_results_df%>%
  group_by(case) %>%
  summarise(tibble(min = min(prob_all), max = max(prob_all), mean = mean(prob_all),  med
ian = median(prob_all),lower = quantile(prob_all,.025), upper = quantile(prob_all,.975
)))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 1 x 7
##   case          min       max       mean     median      lower     upper
##   <chr>       <dbl>     <dbl>      <dbl>      <dbl>      <dbl>     <dbl>
## 1 worst 0.000000464 0.0000393 0.00000763 0.00000647 0.00000138 0.0000213
```

Probability of at least x false positives is equal to $P(X \geq x) = 1 - P(X \leq x - 1)$

```
at_least<-function(p, num, trials){
  return(1-pbinom(q = num-1, size = trials, prob = p ))
}

add_cols_probs<-function(df, prob_var,trials){

  for(success in 1:trials){
    var_name <- paste("p_atleast",success,sep="_")
    df[,var_name]<-at_least(p=as.numeric(df[,prob_var]),num=success, trials = trials)

  }
  return(df)
}

head(add_cols_probs(all_results_df, "prob_seq", trials = 9))
```

```
##        case trial                 prev prob         prob_seq          sens_seq
## res   worst     1 0.000373769674820383 <NA> 0.268948655256724 0.839887640449438
## res.1 worst     2 0.000373769674820383 <NA> 0.285714285714286 0.858725761772853
## res.2 worst     3 0.000373769674820383 <NA> 0.273381294964029 0.851123595505618
## res.3 worst     4 0.000373769674820383 <NA> 0.276018099547511 0.888888888888889
## res.4 worst     5 0.000373769674820383 <NA> 0.258536585365854 0.835164835164835
## res.5 worst     6 0.000373769674820383 <NA> 0.252955082742317 0.870523415977961
##                spec_seq n_both_pos n_neg_both_pos     prob_all p_atleast_1
## res   0.991030658838878        409            110 7.362485e-06   0.9403609
## res.1 0.989737647935115        434            124 1.268784e-05   0.9515997
## res.2 0.990445859872611        417            114 8.529413e-06   0.9435376
## res.3 0.989849405108578        442            122 9.299039e-06   0.9453551
## res.4 0.991175491175491        410            106 5.160575e-06   0.9322659
## res.5  0.99113798244161        423            107 4.240242e-06   0.9275363
##       p_atleast_2 p_atleast_3 p_atleast_4 p_atleast_5 p_atleast_6 p_atleast_7
## res     0.7428937   0.4523066   0.2028616  0.06520808  0.01456630 0.002145797
## res.1   0.7773588   0.4985733   0.2383735  0.08225366  0.01980571 0.003152928
## res.2   0.7523482   0.4646175   0.2120223  0.06946860  0.01583452 0.002381750
## res.3   0.7578549   0.4719171   0.2175516  0.07208633  0.01662770 0.002531960
## res.4   0.7197055   0.4232397   0.1820361  0.05588035  0.01189181 0.001666406
## res.5   0.7067055   0.4076056   0.1712914  0.05126471  0.01062276 0.001448317
##        p_atleast_8  p_atleast_9
## res   0.0001874756 7.362485e-06
## res.1 0.0002981642 1.268784e-05
## res.2 0.0002125620 8.529413e-06
## res.3 0.0002288173 9.299039e-06
## res.4 0.0001383618 5.160575e-06
## res.5 0.0001169435 4.240242e-06
```

```r
all_results_df_p<-add_cols_probs(all_results_df, "prob_seq", trials = 9)

# by case
for(case in unique(all_results_df_p$case)){
  case_df <-all_results_df_p[all_results_df_p$case==case,]
  print(case)
  print(round(colMeans(case_df[,11:19]), digits = 5))
}
```

```
## [1] "worst"
## p_atleast_1 p_atleast_2 p_atleast_3 p_atleast_4 p_atleast_5 p_atleast_6
##     0.93492     0.72959     0.43841     0.19487     0.06251     0.01405
## p_atleast_7 p_atleast_8 p_atleast_9
##     0.00210     0.00019     0.00001
```