A meta-analysis of SARS-CoV-2 prevalence

using the Stan probabilistic programming language

Bob Carpenter

Center for Computational Mathematics, Flatiron Institute

May 2020

What is prevalence?

- A condition's **prevalence** is the proportion of the population that has it
  - e.g., if 32 of a population of 1000 has a condition, its prevalence is 3.2%.
- We'd like to **estimate** prevalence of individuals
  1. with SARS-Cov-2 **virus**,
  2. with COVID-19 **disease**,
  3. who have developed **antibodies** to SARS-Cov-2, and
  4. who are **infectious**.
- Viral infection (1) is the focus of this talk

Why is estimation challenging?

- Conditions form multiple **scales**
  - how much virus? which symptoms? how infectious? which antibodies?
- Measurements are **noisy**
  - **error**: inaccurate tests, varying accuracy across sites, human judgement, ...
  - **sampling**: extrapolate from sample to population
- Population **heterogeneity**
  - **demographics**: sex, age, existing medical conditions ...
  - **behavior**: social distancing, protective measures, food, travel, ...
  - **geo-political**: location, (local) government, climate, ...
  - **temporal**: prevalance evolves over time
  - **testing**: availability, assignment, self selection, ...

Understanding sampling uncertainty

- **Simulate**: false positive results; N = 100, 2% false positive rate
  - **simulated false positives** (100 simulations): 1 2 1 2 0 2 4 4 2 3 3 2 3 0 1 1 1 4 2 1 1 4 0 1 1 3 1 0 2 1 8 2 4 2 2 4 1 4 0 1 0 0 3 1 5 1 3 3 4 0 3 5 0 3 1 3 2 3 1 0 1 4 2 2 1 0 2 1 1 1 2 1 1 3 2 2 3 2 0 1 2 3 1 1 1 2 2 0 2 4 2 2 2 3 3 1 1 4 3 2
  - **min** 0 (0%); **max** 8 (8%); **std dev** 1.4 (1.4%)
- **Simulate**: positive status; N = 3000, 1.5% prevalence
  - **simulated positives** (100 simulations): 39 51 42 43 52 52 37 47 41 51 43 47 47 41 49 43 40 44 46 44 49 50 54 48 31 44 57 40 46 40 51 49 48 46 51 40 47 47 42 42 42 40 55 34 40 48 35 39 45 48 42 42 45 54 43 40 40 39 48 42 45 36 41 47 40 42 43 41 39 52 47 46 43 38 46 31 49 27 39 42 43 46 37 38 36 45 36 47 41 35 49 43 51 45 47 34 46 43 46 49
  - **min** 27 (0.9%); **max** 57 (1.9%); **std dev** 5.5 (0.2%)

Sensitivity and specificity of diagnostic tests

- Split accuracy based on status of individuals to account for test biases
- **sensitivity** is accuracy with positive status $\Pr[\text{test} = 1 \mid \text{status} = 1]$
  - sensitive tests have low false negative rates
- **specificity** is accuracy on negative status $\Pr[\text{test} = 0 \mid \text{status} = 0]$
  - specific tests have low false negative rates
- Examples from breast cancer diagnosis
  - **mammogram, MRI**: high sensitivity, low specificity
  - **puncture biopsy**: low sensitivity, high specificity
  - this profile can't catch breast cancer reliably until it's **too late**

Analyzing Serum PCR tests for SARS-CoV-2

- **Sensitivity** tests (known positives)

positives	total	sensitivity
78	85	92%
27	37	73%
25	35	71%

- **Prevalence test** (unknown status)

positives	total	prevalence
50	3300	1.5%

- **Goal**: estimate of **SARS-Cov-2 prevalence**

Specificity tests (known negatives)

negatives	total	specificity
368	371	99%
30	30	100%
70	70	100%
1102	1102	100%
300	300	100%
311	311	100%
500	500	100%
198	200	99%
99	99	100%
29	31	94%
146	150	97%
105	108	97%
50	52	96%

Adjust for test sensitivity & specificity

- Proportion of positive tests in sample must be **adjusted**.
  - for test sensitivity and specifity
- Expected **proportion of positive tests** is

$$\begin{aligned} \Pr[\text{test} = 1] &= \Pr[\text{status} = 1] \times \Pr[\text{test} = 1 \mid \text{status} = 1] \\ &\quad + \Pr[\text{status} = 0] \times \Pr[\text{test} = 1 \mid \text{status} = 0] \\ &= \text{prev} \times \text{sens} + (1 - \text{prev}) \times (1 - \text{sens}). \end{aligned}$$

- **Solve for expected prevalence** given sensitivity, specificity, positive tests.

$$\text{prev} = \frac{\text{pos} + \text{spec} - 1}{\text{sens} + \text{spec} - 1}$$

Uncertainty behind prevalence estimates

- Previous slide assumes sensitivity and specificity are known.
- Three forms of uncertainty lead to uncertainty in prevalence:
  - test **sensitivity and specificity are unknown** and estimated from data,
  - the result of a **test is uncertain** given the status of an individual, and
  - tests are applied to **only a sample** of a population.
- The job of statistics is to **adjust for bias** and **quantify uncertainty**
  - it's **not magic**—it's **assumption driven**

Test sensitivty and specificity varies by site

- sensitivity and specificity are intrinsically **anti-correlated**
  - adjusting thresholds trades one for the other
- sensitivity and specificity are **correlated by site**
  - good procedures increase both; bad procedures decrease both
- perform a **meta-analysis** with a **hierarchical model** to
  - estimate **mean sensitivity and specificity** of the test,
  - estimate **each site's** sensitivity and specificity,
  - let amount of variation among sites control how much to **pool data**, and
  - predict behavior in new test sites with no control cases.

Stan Code (Data & Parameters)

```
data {
  int<lower = 0> K_pos;
  int<lower = 0> N_pos[K_pos];
  int<lower = 0> n_pos[K_pos];
  int<lower = 0> K_neg;
  int<lower = 0> N_neg[K_neg];
  int<lower = 0> n_neg[K_neg];

  int<lower = 0> N_unk;
  int<lower = 0> n_unk;
}
```

```
parameters {
  real<lower = 0, upper = 1> prev;
  vector<lower = 0, upper = 1> sens[K_pos];
  vector<lower = 0, upper = 1> spec[K_neg];
  real<lower = 0, upper = 1> mu_sens;
  real<lower = 0> kappa_sens;
  real<lower = 0, upper = 1> mu_spec;
  real<lower = 0> kappa_spec;
  vector<lower = 0, upper = 1> sens_unk
  vector<lower = 0, upper = 1> spec_unk;
}
```

Stan Code (Model)

```
model {
  // hyperprior
  prev ~ uniform(0, 1);
  mu_spec, mu_sens ~ beta(9, 1);
  kappa_sens, kappa_spec ~ exponential(0.5);

  // prior (hierarchical)
  sens, suns_unk ~ beta(mu_sens * kappa_sens, (1 - mu_sens) * kappa_sens);
  spec, spec_unk ~ beta(mu_spec * kappa_spec, (1 - mu_spec) * kappa_spec);

  // likelihood
  n_pos ~ binomial(N_pos, sens);
  n_neg ~ binomial(N_neg, spec);
  n_unk ~ binomial(N_unk, prev * sens_unk + (1 - prev) * spec_unk);
}
```

Running Stan Code

- Can be run from R, Python, Julia, MATLAB, Mathematica, or shell

  Output for justthe prevalence estimate

	mean	se_mean	sd	2.5%	50%	97.5%	n_eff	Rhat
prev	0.013	0	0.003	0.007	0.012	0.019	7795	1

- **95% posterior interval** is (0.007, 0.019)
- Result is highly dependent on breadth of **sensitivity hyperprior**
  - only 3 sensitivity tests available
- Result does not vary among a range of **weakly regularizing** hyperpriors
  - e.g, assumiming variation among sites is on the order of 1–20%, but not 50%.
- Assuming no variation **underestimates uncertainty**

Adjusting for non-representative samples

- Prevalence **varies in subpopulations**
  - exposure risk by demographics; geographically by population density/travel; differing metabolism by age, sex; political and social effects
- May not have a random sample
  - because of purposeful stratified design; or convenience opt-in sample
- Either way, we use **multilevel regression** and **post-stratification** to adjust

  Step 0. fit a multilevel regression to the data (for regularization/pooling)

  Step 1. estimate prevalence in each demographic subgroup

  Step 2. weight prevalence in subgroups by their size
- Simulations in paper; real results awaiting Stanford IRB approval

Further Reading

- **Project home page**: `https://bob-carpenter.github.io/diagnostic-testing`
- **Stan home page**: `https://mc-stan.org`
- **Reports** (comments welcome!)

  - Gelman, A. & B. Carpenter. 2020. Bayesian analysis of tests with unknown specificity and sensitivity. *DRAFT.*
  - Carpenter, B. & A. Gelman. 2020. Case study of seroprevalence meta-analysis. *DRAFT.*
  - Carpenter, B., A. Gelman, M. D. Hoffman, et al. (2017). Stan: A probabilistic programming language. *J. Stat. Soft.* 76(1).
  - Carpenter, B. 2016. Stan case study: Hierarchical partial pooling for repeated binary trials. `https://mc-stan.org/users/documentation/case-studies`

Stan Availability and Usage

- **Platforms:** Linux, Mac OS X, Windows
- **Interfaces:** R, Python, Julia, MATLAB, Mathematica
- **Developers (academia & industry):** 40+ (15+ FTEs)
- **Users:** tens or hundreds of thousands
- **Companies using:** hundreds or thousands
- **Downloads:** millions
- **User's Group:** 3000+ registered; 6000+ non-bot views/day
- **Books using:** 10+
- **Courses using:** 100+
- **Case studies about:** 100+
- **Articles using:** 5000+
- **Conferences:** 4 (800+ attendance); **StanCon 2020** will be online

Some published applications of Stan

- **Physical sciences**: astrophysics, statistical mechanics, particle physics, organic chemistry, physical ehmistry, geology, hydrology, oceanography, climatology, biogeochemistry, materials science, ...
- **Biological sciences**: molecular biology, clinical drug trials, entomology, pharmacology, toxicology, opthalmology, neurology, genomics, agriculture, botany, fisheries, epidemiology, population ecology, neurology, psychiatry, ...
- **Social sciences**: econometrics (macro and micro), population dynamics, cognitive science, psycholinguistics, social networks, political science, survey sampling, anthropology, sociology, social work, ...
- **Other**: education, public health, A/B testing, government, finance, machine learning, transportation logistics, electrical engineering, mechanical engineering, civil engineering and transportation, actuarial science, sports analytics, advertising attribution, marketing, ...

Industries using Stan

- **Marketing attribution**: Google, Domino's Pizza, Legendary Ent.
- **Demand forecasting**: Facebook, Salesforce
- **Financial modeling**: Two Sigma, Point72
- **Pharmacology & CTs**: Novartis, Pfizer, Astra Zeneca
- **(E-)sports analytics**: Tampa Bay Rays, NBA, Sony Playstation
- **Survey sampling**: YouGov, Catalist
- **Agronomy**: Climate Corp., CiBO Analytics
- **Real estate pricing models**: Reaktor
- **Industrial process control**: Fero Labs

Why is Stan so Popular?

- **Community**: large, friendly, helpful, and sharing
- **Documentation**: novice to expert; breadth of fields
- **Robustness**: industrial-strength code; user diagnostics
- **Flexibility**: highly expressive language; large math lib
- **Portability**: popular OS, language, and cloud support
- **Extensibility**: developer friendly; derived packages
- **Speed**: $2 - \infty$ orders of magnitude faster
- **Scalability**: 2+ orders of magnitude more scalable
- **Openness**: permissive code and doc licensing