Initial evaluation of samplers

Ben Bolker

11:58 27 December 2016

This version of the document has most of the technical details hidden; if you want to explore these details, look at the `.rmd` version (which can be opened in any text editor, or more conveniently in RStudio).

Extract the total number in the population, and the number diseased, from a population output file:

```
fn <- "multi2.p00.json"
get_counts(fn)
```

```
## Warning: running command 'grep -m 2 _count multi2.p00.json' had status 2
```

```
## numeric(0)
```

Single sample

This follows the `readme` file to generate a single population (with one town) and a single sample and get the counts from each.

```
system("./generate -c config/default_population.conf test")
system("./strip_epi_sample test.json strip")
get_counts("test.json")   ## population
```

```
## Warning: running command 'grep -m 2 _count test.json' had status 2
```

```
## numeric(0)
```

```
get_counts("strip.json")  ## sample
```

```
## Warning: running command 'grep -m 2 _count strip.json' had status 2
```

```
## numeric(0)
```

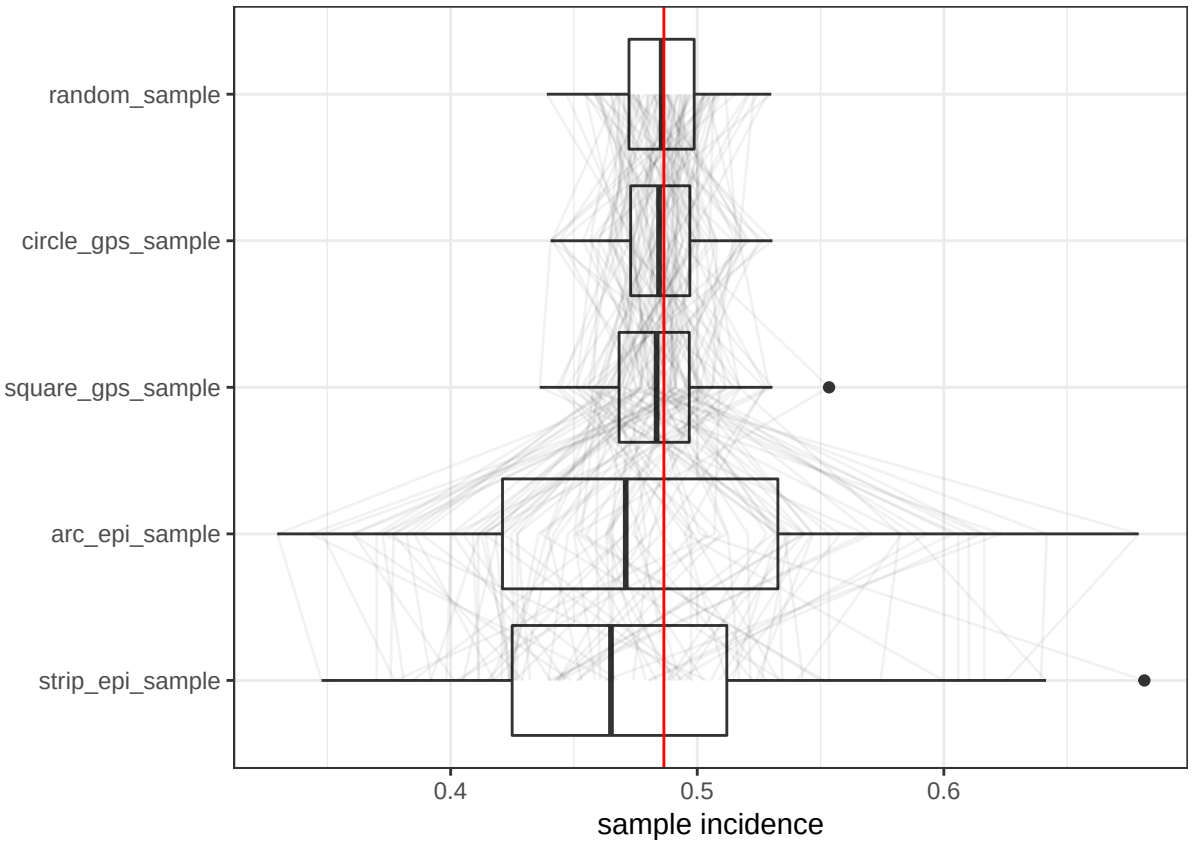
Multiple samples

Here we are generating 99 populations; for each population we are running each sampler.

For each population, extract the counts; then try every sampler and record the counts. We are (now) using a different random-number seed (the same for every sampler) for each population; this should address the problem that the results were presumably biased by the interaction between the fixed geographic gradient and the fixed (if seed was unchanged) starting direction for some samplers.

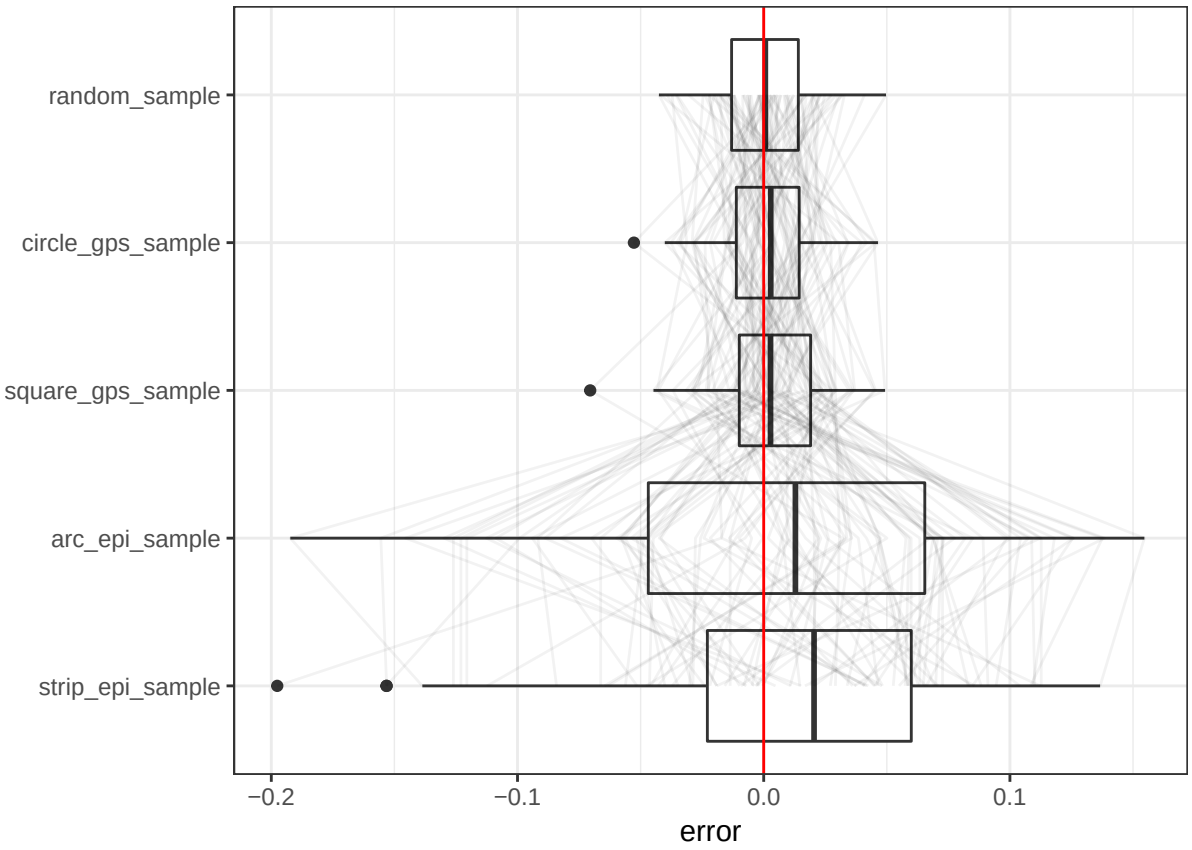
Doing this size sample (99 populations $\times$ 5 samplers) takes on the order of a half an hour on my laptop (I haven't checked carefully); there are probably some ways we can speed it up.

Basic results. Boxplots show overall distributions of estimated proportions; red line shows overall population mean (averaged across populations); gray lines connect results for the same populations across samplers.



Bias (distribution of differences between true and estimated proportion): blue points/whiskers represent means and 95% bootstrap CIs. The `arc_epi` and `strip_epi` samplers are obviously more variable, but they may not be biased (95% CIs of `strip_epi_sample` are just touching zero).

```
## Warning: Computation failed in `stat_summary()`:
## Hmisc package required for this function
```




```
##             sampler    N     bias      var   RMSE coverage
## 1  strip_epi_sample 1001 0.012457 0.004503 1.2333    0.323
## 2    arc_epi_sample 1002 0.007701 0.005642 0.7624    0.293
## 3 square_gps_sample 1001 0.002391 0.000475 0.2367    0.869
## 4 circle_gps_sample 1002 0.001548 0.000404 0.1532    0.869
## 5     random_sample 1002 0.000563 0.000357 0.0558    0.889
```

Despite being unbiased, the coverage of a naive binomial confidence interval is terrible for the EPI samplers and a little bit bad for the GPS and random samples, probably due to the clustering/spatial heterogeneity in the sample (theory suggests that SRS should be unbiased no matter what – but what does it say about the variance of the estimator?)

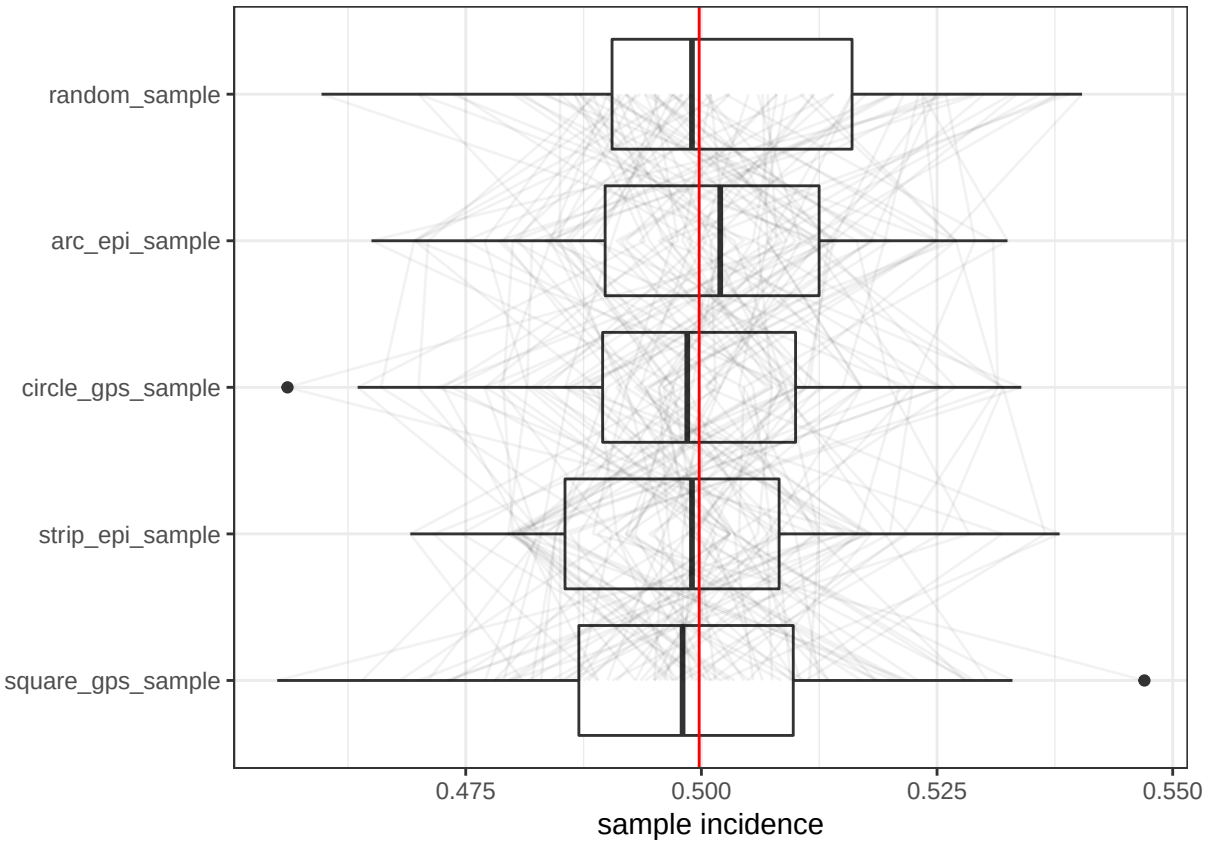
Vanilla pops

Re-run all of this with a "vanilla" population configuration; all interesting settings (pocketing, geographic variation, etc. etc.) are set to zero.

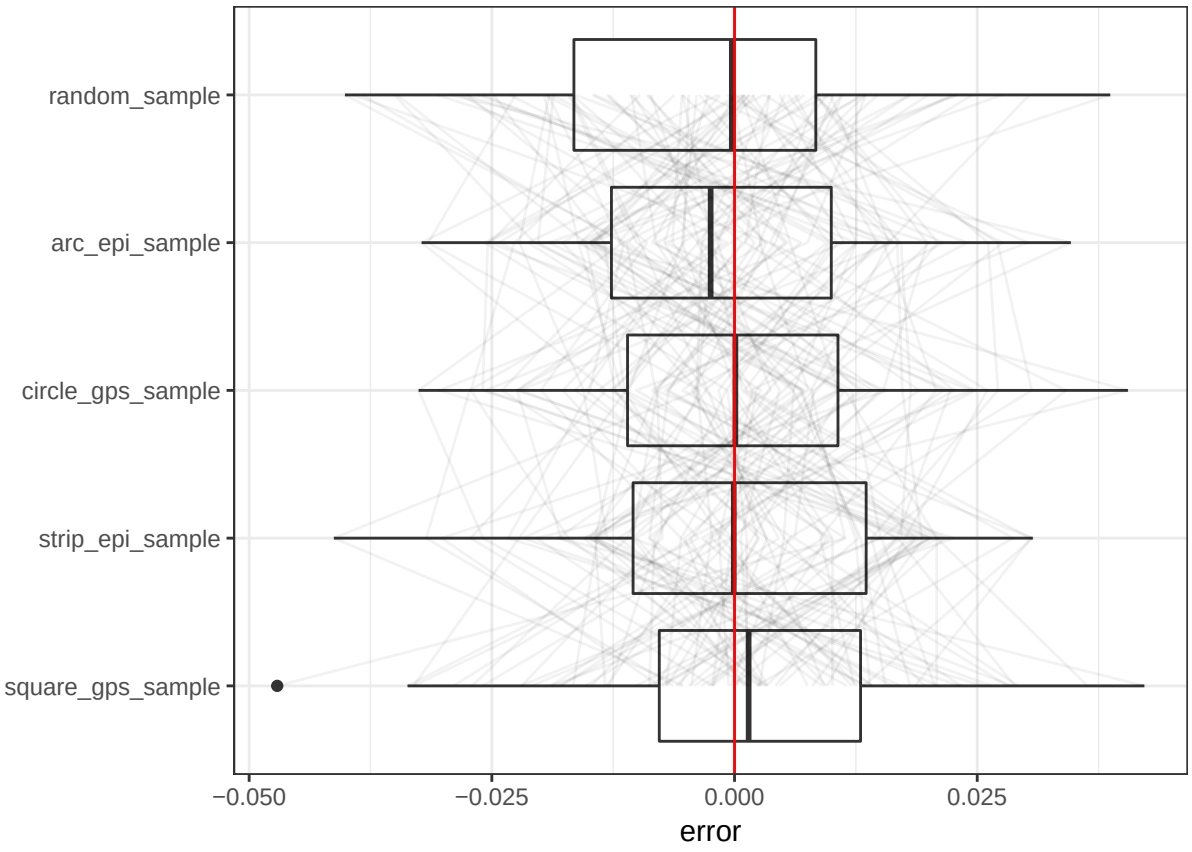
To see differences:

```
diff config/default_population.conf config/vanilla_population.conf
```

Re-do plots with data from "vanilla" configurations.




```
## Warning: Computation failed in `stat_summary()`:
## Hmisc package required for this function
```




```
##             sampler    N      bias      var   RMSE coverage
## 1 square_gps_sample 1001  0.001181 0.000287 0.1169    0.939
## 2  strip_epi_sample 1001  0.000731 0.000222 0.0724    0.980
## 3 circle_gps_sample 1001  0.000244 0.000268 0.0242    0.970
## 4    arc_epi_sample 1001 -0.001439 0.000248 0.1424    0.970
## 5     random_sample 1001 -0.003338 0.000308 0.3305    0.939
```

Now all methods are more or less comparable, and unbiased, and well-covered: the standard error of the binomial estimate for coverage from 99 samples should (?) be about $\sqrt{0.95 \cdot 0.05/99} \approx 0.02$, so anything within a few percentage points of 0.95 seems good. (Could do more runs if we needed to be more precise.)

To do/fix me:

- Makefile/otherwise avoid recomputation in a smarter way
- Think about how we can do more runs, more automatically ...
- It would be very convenient if we could run all samplers for the same population (in C++ code) automatically; not sure how much efficiency gain we would get. At present we can run the *same* sampler for lots of populations, without writing out the population information, but we like to be able to do the inverse (one config file → one population → many samplers, or even one config file → many populations → many samplers, or [hypercube config file] → many populations → many samplers ...).
- Does it make sense to allow an NA/NULL value for the random number seed that picks an *arbitrary* starting seed, i.e. for casual use?