Replication Power

Nathan (Nat) Goodman

August 22, 2018

A collection of R scripts and documents exploring the power of replication to detect bad science. I treat replication as a statistical test, simulate proposed replication methods across a wide range of conditions, and estimate error rates for conditions of interest. The main point is that replication is a poor statistical test with unacceptable error rates under most conditions. It works as a validation test only when the original and replica studies are sampling nearly identical populations. Methods for testing whether the populations are similar work poorly under all conditions analyzed.

THE SOFTWARE IS STILL ROUGH and SOFTWARE DOCUMENTATION NONEXISTENT. PLEASE GET IN TOUCH IF YOU NEED HELP

Overview

The program explores the power of replication to detect bad science. The software simulates *studies* across a range of conditions, combines pairs of studies into *pairwise replications*, applies rules (called *measures*) for deciding which pairwise replications pass, summarizes the results as counts and pass rates, and finally computes false positive and negative rates for measures and conditions of interest. The main conclusion is that replication is a poor statistical test with unacceptable error rates under most conditions. Significance testing of the replica works fine as a validation test for exact and near-exact replications, but error rates increase rapidly as the populations diverge. All other tests have excessive error rates under all conditions analyzed. All tests have unacceptable error rates when used to check whether the original and replica studies are similar.

To calculate error rates, it's necessary to define explicit correctness criteria. The ones I use are

1. *non-zero*: a replication instance is *true* if the population effect size of the first study is non-zero
2. *same-effect*: a replication instance is *true* if both studies have the same population effect size; with *tolerance* δ, a replication instance is *true* if the two population effect sizes differ by at most δ

The measures appearing in this README document are

- *sig2*: the replica study has a significant p-value
- *sigm*: the fixed effect meta-analysis of the studies has a significant p-value
- *d1.c2*, *d2.c1*: the standardized observed effect size (aka *Cohen's d*) of one study is in the confidence interval of the other
- *d1.p2*, *d2.p1*: *Cohen's d* of one study is in the prediction interval of the other
- *c1.c2* (resp. *p1.p2*): the confidence (resp. prediction) intervals of the two studies overlap
- *d2.scp1*: Uri Simonsohn's *small telescopes* method

All measures assume that the original study is significant (*sig1* in my notation) and the observed effect sizes of the two studies have the same sign. *Small telescopes* also assumes that *sig2* holds.

Installation and Usage

The software is **not a package** and cannot be installed by `devtools::install_github` or related. Sorry. The simplest way to get the software is to download or clone the entire repo.

The code mostly uses base R capabilities but has a few dependencies: `RColorBrewer`, `akima`, and `pryr`. Since it's not a package, you have to manually install these packages if you don't already have them.

The recommended way to run the program is to `source` the file `R/repwr.R` into your R session; `R/repwr.R` will source the rest. Once loaded, you can run the program by executing the statement `run()` as shown below.

```
## This code block assumes your working directory is the root of the distribution.

source('R/repwr.R');
run();
```

This runs the program in a demo-like mode that quickly generates the simulated data and produces the figures that appear in this README document. The default computation simulates 625,000 replications and produces 16 figures. The simulation takes about 40 seconds on my small Linux server; the figures take about 60 seconds, much of which is spent rendering the plots over a remote X11 connection.

To rerun the program from scratch you must specify `clean=T`, since by default the program reuses existing data.

```
## This code block assumes your working directory is the root of the distribution
## and you've already sourced R/repwr.R into your session

run(clean=T);        # delete data from previous run and rerun program
```

You can run each part separately by running one of the statements below.

```
## This code block assumes your working directory is the root of the distribution
## and you've already sourced R/repwr.R into your session

dodata(clean=T);     # delete data from previous run and generate the simulated data
dodoc();             # generate the figures
dodoc(save.fig=F);   # generate the figures without saving them
dodoc(figscreen=F);  # generate and save the figures without plotting to the screen. faster!
```

The program can also generate the data and figures for the blog post associated with the project. **CAUTION: this takes much longer to run**: about an hour on my small Linux server. To generate these, execute `run()` with a suitable `doc` argument as shown below.

```
## This code block assumes your working directory is the root of the distribution.

source('R/repwr.R');
run(doc='repwr');    # generate data and figures for blog post
```

Figures

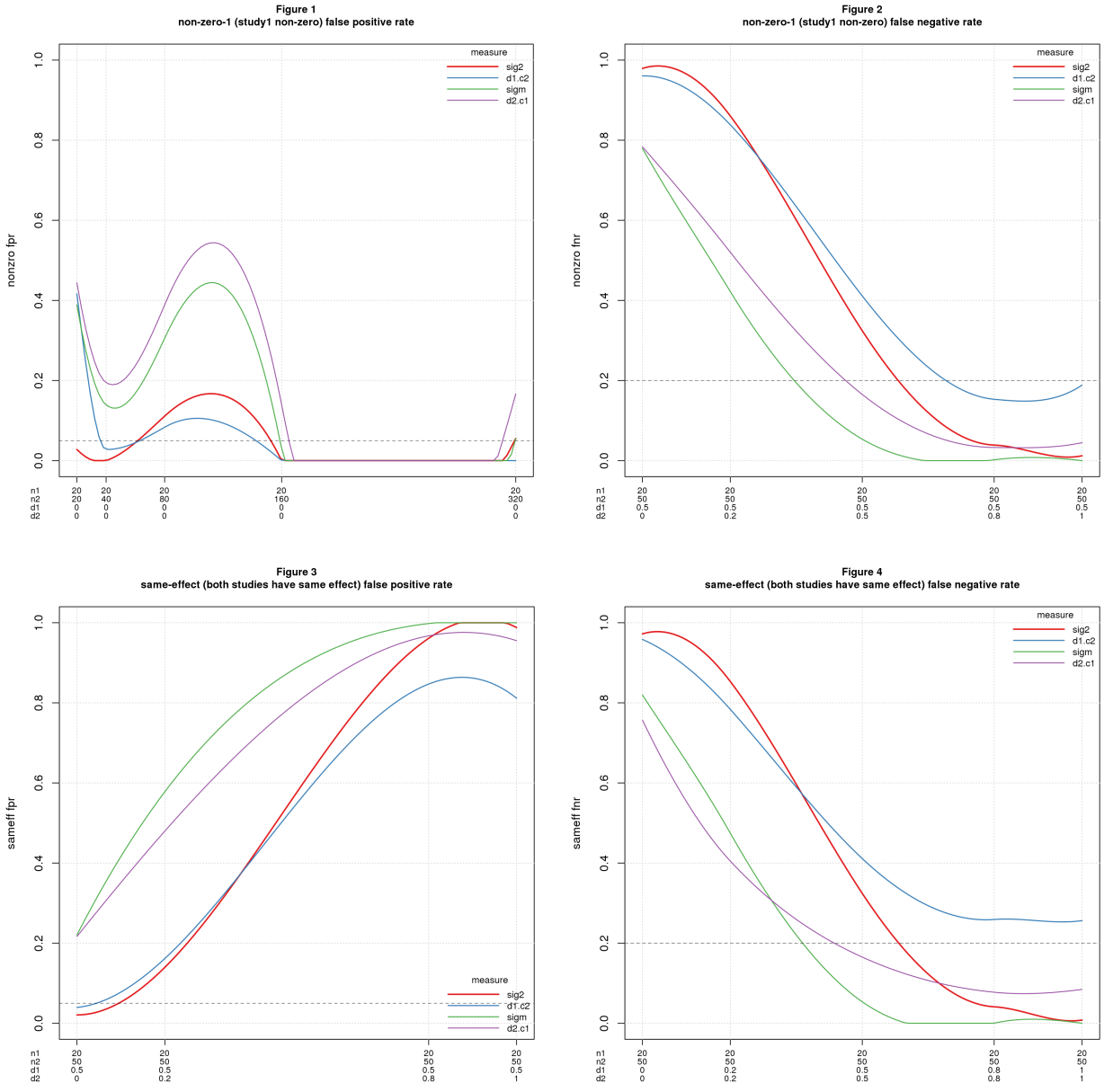
The default mode produces figures that illustrate the kinds of graphs the program can produce.

1. line graphs showing error rates for a set of measures across chosen conditions: simple and intuitive but poor at showing data for too many measures and parameters
2. heatmaps showing the same kind of data: still reasonably intuitive and somewhat better at depicting more measures and parameters
3. rate-vs-rate scatter plots: able to display error rates across large swaths of parameter space but with less parameter resolution and perhaps less intuitive clarity; inspired by *receiver operating characteristic (ROC)* curves
4. aggregate line graphs: same data as rate-vs-rate scatter plots but for fewer measures and with better parameter resolution

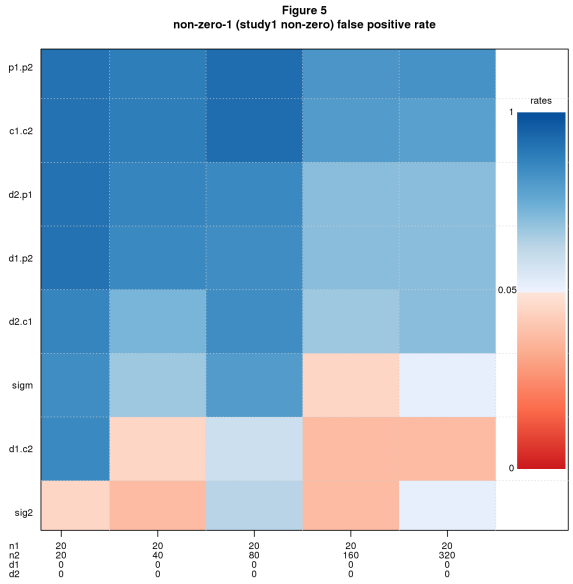
The first group of figures are line graphs showing false positive and false negative rates for a few measures across a few conditions. The labels on the x-axis show the conditions: *n1*, *n2* are the sample sizes; *d1*, *d2* are

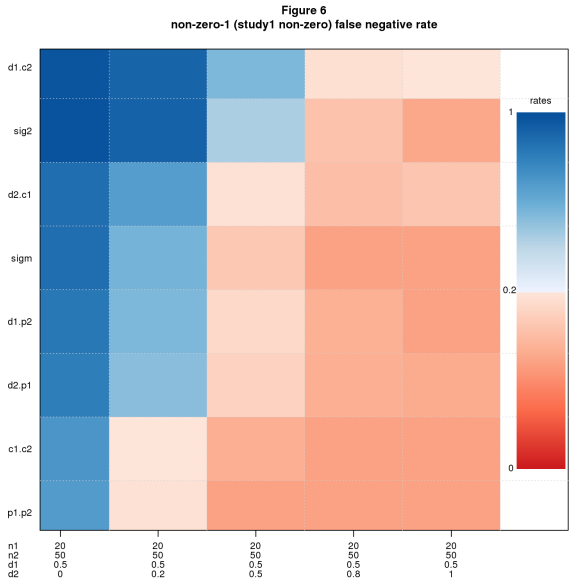
the population effect sizes. The horizontal dashed lines demark the conventionally accepted thresholds of 0.05 for false positives and 0.20 for false negatives.

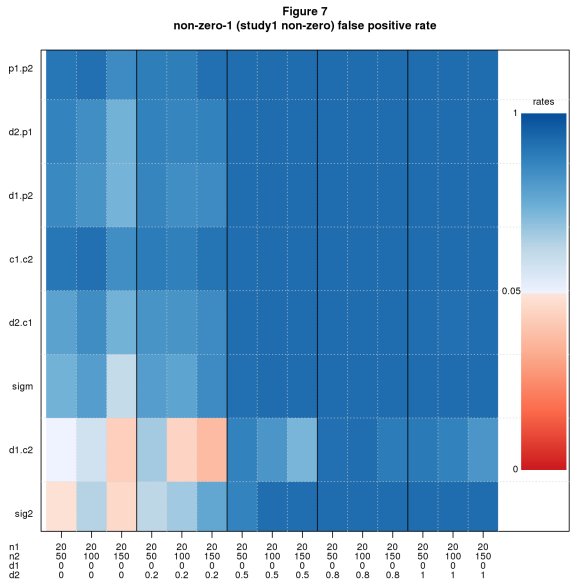
Figures 1-2 are for the *non-zero* correctness criterion; figures 3-4 are for *same-effect*.

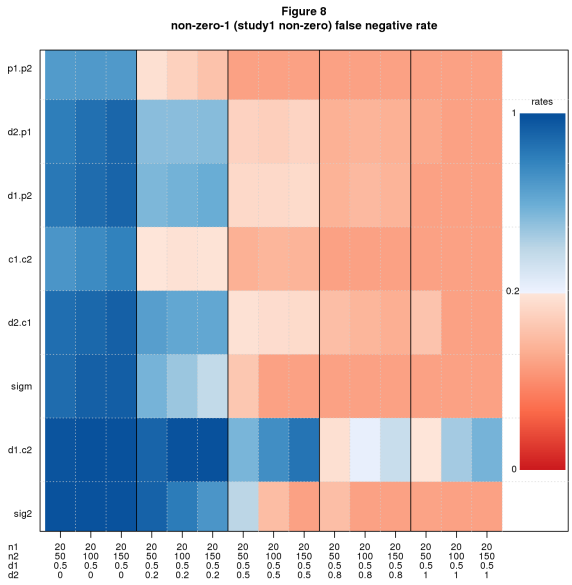


The next figures are heatmaps. Figures 5-6 show the same conditions as figures 1-2 but for more measures; figures 7-8 show more conditions. The red-to-blue transition is set at the conventionally accepted thresholds of 0.05 for false positives and 0.20 for false negatives. The dark vertical lines in figures 7-8 visually split each plot into separate "panels" for each value of *d2*.

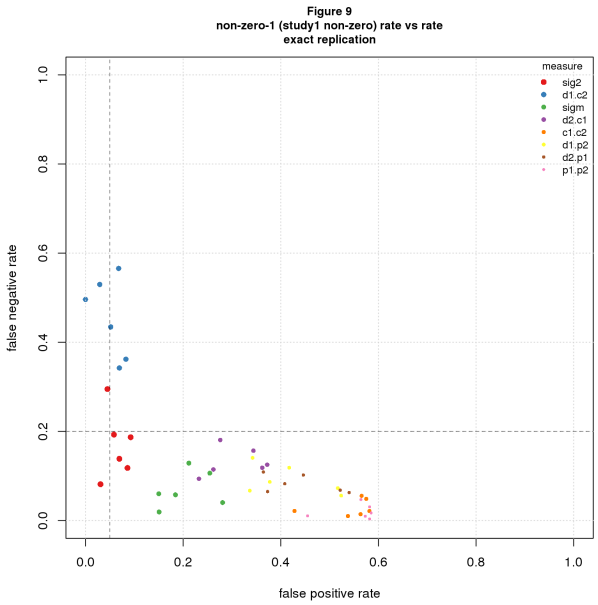


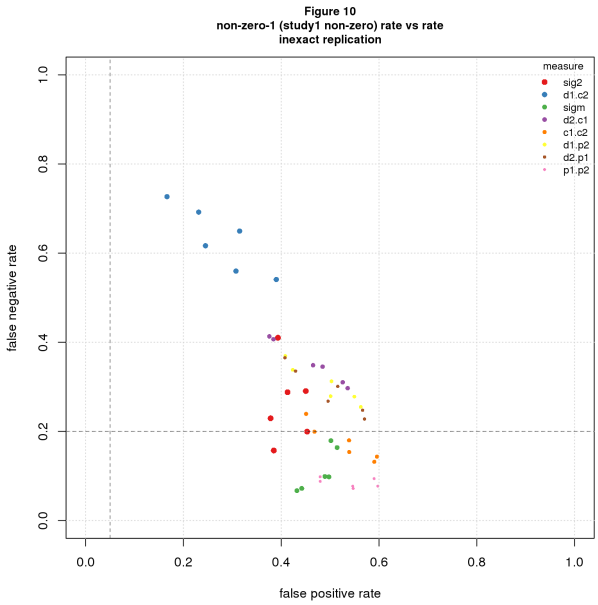




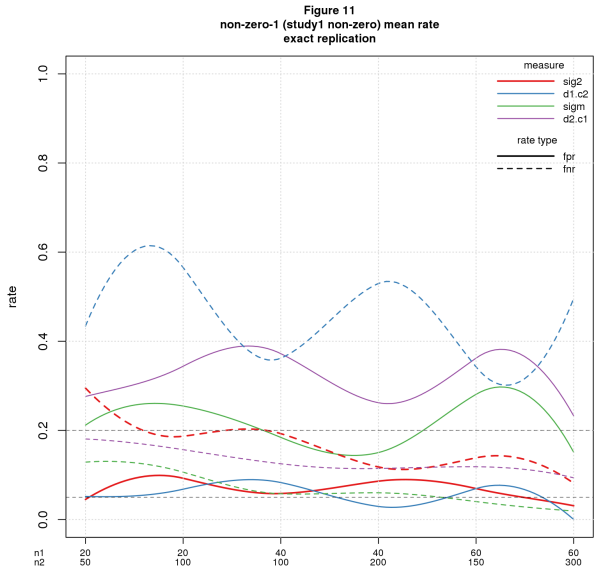


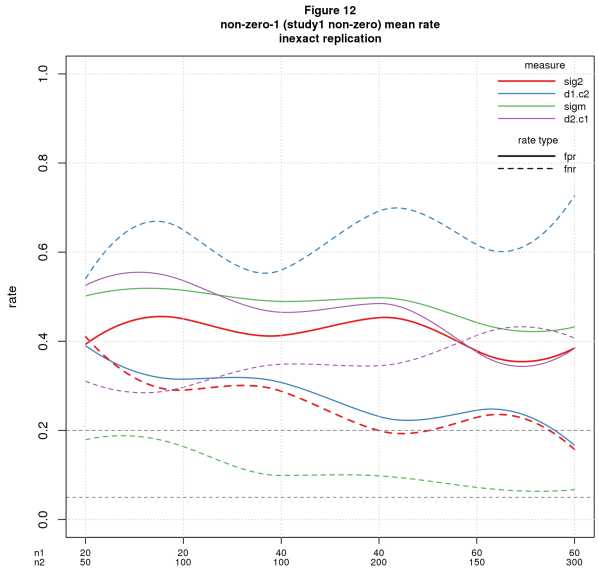
The next two figures (figures 9-10) are rate-vs-rate graphs for *exact* and *inexact* replications. Each point shows the mean false negative vs. mean false positive rate for specific conditions grouped by $n1, n2$. The dashed lines demark the conventionally acceptable error rates; the bottom left hand corner is the region where both error rates are acceptable. You'll note that for *exact*, *sig2* is the only measure with points in or near the acceptable region; for *inexact*, no points are even close.



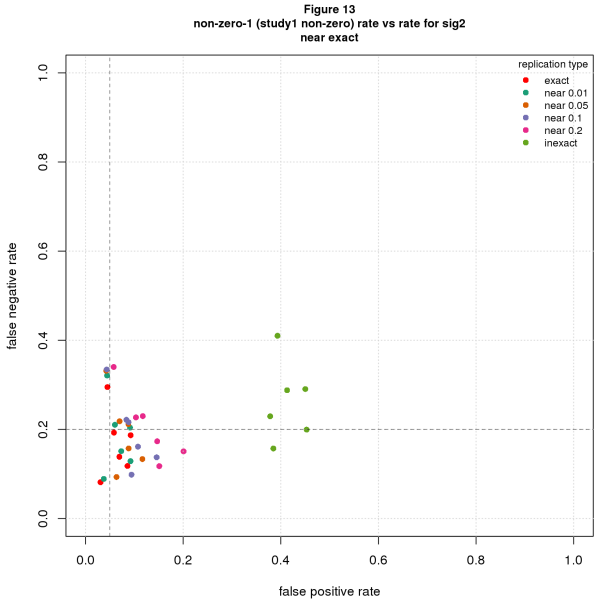


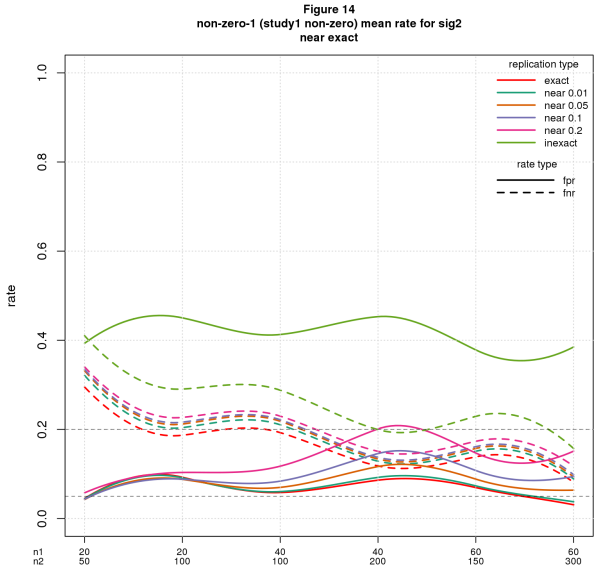
Figures 11-12 are aggregate line graphs showing the same data as the rate-vs-rate graphs above for fewer measures.



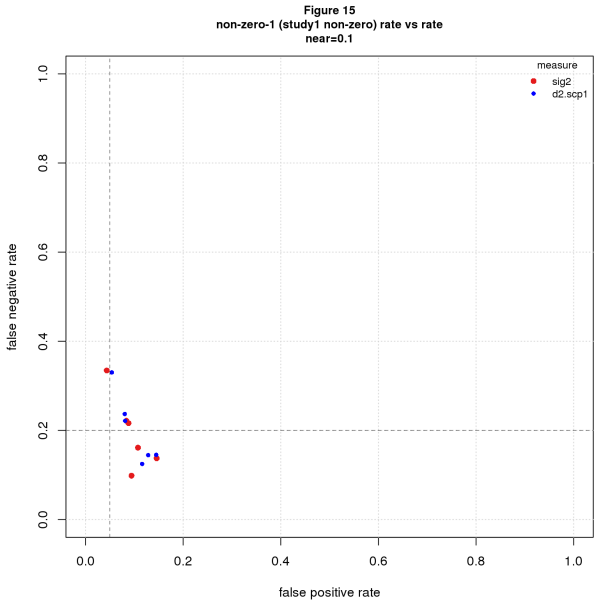


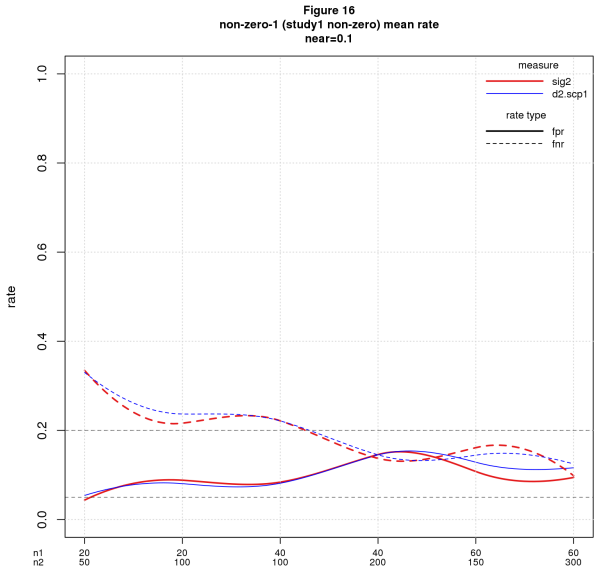
Recall that *sig2* works pretty well in exact replications but poorly in inexact ones (see figures 9-10). The next two figures (figures 13-14) show how *sig2* performs in *near exact* replications, ones where the population effect sizes differ slightly. The first is a rate-vs-rate graph showing *sig2* across various nearness values; the second is an aggregate line graph showing the same data.





The final two figures (figures 15-16) compare *sig2* and *d2.scp1* (Uri Simonsohn's *small telescopes* method). The differences are quite small.





See Also

A blog post discussing the approach and results is available in html and pdf on the GitHub Pages site associated with this repository and will soon be posted on a blog site TBD. It's also in the repository as files repwr.stable.html and repwr.stable.pdf. (But note that GitHub, unlike GitHub Pages, renders html files as raw text). The previous version of the post (version 1.00) are available as html and pdf

Author

Nathan (Nat) Goodman, (natg at shore.net)

Bugs and Caveats

Please report any bugs, other problems, and feature requests using the GitHub Issue Tracker. I will be notified, and you'll be apprised of progress. As already noted, the software is still rough and software documentation nonexistent.

Copyright & License