Measuring Replication Success

Nathan (Nat) Goodman

December 5, 2018

Statisticians have devised numerous statistical tests for deciding whether a replication passes or fails, thus validating of refuting the original result. I call these tests "measures". Elsewhere I report results from simulating these tests across a range of conditions. Here I describe the measures themselves and provide implementation details for ones that aren't obvious.

Introduction

Various authors have proposed tests for deciding whether a replication succeeds or fails. These include significance testing of the replica study, seeing whether the observed effect size of one study falls in the confidence or prediction interval of the other, checking whether the confidence or prediction intervals of the two studies overlap, significance testing of the studies combined via meta-analysis, and comparing the observed effect size of the repeated study with a "small telescope" threshold deemed to be the smallest true effect size the original study could have plausibly detected.

I simulated these rules (and more) across many replication conditions and computed false positive and false negative rates for conditions of interest. I reported these results in a working paper, Systematic Replication Has Limited Power to Detect Bad Science, a shorter blog post, Systematic Replication May Make Many Mistakes, and a supplement to the blog post.

In this document I describe the measures themselves, providing implementation details where appropriate. The code implements 28 measures. These include all replication rules I found in the literature or blogosphere that I was able to implement with adequate performance, plus ones I added to satisfy my sense of completeness, and a few I needed for technical reasons. The Measures section below has them all, but here a few to give you a flavor.

- *sig1* (resp. *sig2*) - the first (resp. second) study of the pair has a significant p-value
- *d1.c2* (resp. *d2.c1*, *d1.p2*, *d2.p1*) - the standardized observed effect size of one study is in the confidence interval (resp. prediction interval) of the other study
- *sdir* - the observed effects sizes for the two studies are in the same direction, i.e., have the same sign

Several measures depends on nuances of the *noncentral t-distribution*. The usual treatment parameterizes this distribution by *degrees of freedom* and a *noncentrality parameter*, and operates on the *t-statistic*. I prefer to work with more concrete notions: *sample size* (instead of degrees of freedom), *(presumed) population effect size* (instead of the noncentrality parameter), and *standardized observed effect size* (instead of the t-statistic). For the limited context of my simulation, it's easy to convert between the concepts. I developed software that repackages base R's t-distribution functions to use my preferred concepts; I call this the *noncentral d2t-distribution*. To be clear: the d2t-distribution is not a new distribution in any sense; it merely repackages the t-distribution. I describe the d2t-distribution software in Noncentral d2t-Distribution Applied to Measures of Replication Success. The d2t software does much of the heavy lifting for the measures.

Simulation

I consider a basic replication scheme in which each original study is repeated once.

The software first simulates *studys* across a range of conditions, then combines pairs of studies into *pairwise replications*, applies rules (the *measures*) for deciding which pairwise replications pass, and finally computes true and false positive and negative rates for measures and conditions of interest.

The studies are simple two group comparisons parameterized by sample size n and population effect size d_{pop} ($d_{pop} \geq 0$). For each study, I generate two groups of random numbers, each of size n. One group, *group0*, comes from a standard normal distribution with $mean = 0$; the other, *group1*, is from a standard normal distribution with $mean = d_{pop}$. When I need to be pedantic, I use the term *study set* for the ensemble of studies for a given combination of n and d_{pop}.

To generate pairwise replications, I consider all (ordered) pairs of study sets. For each pair, the software permutes the instances of each study, then combines the instances row-by-row. It is often convenient to think of the first study of the pair as the *original* and the second as the *replication*, but the software doesn't rely on this distinction. A *pairwise replication set* is the ensemble of pairwise replication instances for a given pair of study sets. Four variables parameterize each pairwise replication set: $n1$, $n2$, $d1_{pop}$, $d2_{pop}$. These are, naturally enough, the sample and population effect sizes for the two study sets.

After forming the pairwise replications, I apply the measures. Each measure takes a pairwise replication set as input and returns a vector of boolean values telling which instances pass or fail. The result is a boolean matrix whose rows represent replication instances and columns represent the measures' results.

The next step summarizes the results by counting the number of true results for each measure in the replication set. I compute counts for the measures individually and a few combinations of interest. The combinations are

- *sig1* - instances that also satisfy the *sig1* measure, since most authors assume that the original study is significant
- *baseline* - instances that also satisfy the *sdir* measure, because most authors assume that the observed effect size are in the same direction
- *sig2* - instances that also satisfy the *sig2* measure, because Uri's small telescope rule requires this

To get pass rates, I divide the counts by the number of instances satisfying some baseline condition. Usually, the baseline condition is the one called *baseline* above (hence the name), but it can be the null condition (in which case the denominator is the total number of instances per set, i.e., 10^4 by default) or either of the other named conditions above.

The final step is to convert pass rates into true positive, false positive, true negative, and false negative rates. This requires a definition of *true* and *false* instances, in other words, an explicit statement as to which pairwise replication instances represent replications that should succeed vs. ones that should fail. I call these *correctness criteria*. See the Correctness Criteria section of the working paper for a discussion.

Measures

The table below lists the 28 measures implemented by the software. Each measure applies to a pairwise replication instance (although some only depend on one study of the pair). Measures and the subsequent discussion use the following notation.

- *s1* and *s2* refer to the first and second study of the pair; *sm* means the fixed-effect meta-analysis of the two studies
- *d1*, *d2*, *dm* mean the standardized observed effect size of *s1*, *s2*, *sm*
- *pval1*, *pval2*, *pvalm* mean the p-value computed from the t-distribution of *s1*, *s2*, *sm*
- *sig1*, *sig2*, *sigm* mean *pval1*, *pval2*, *pvalm* is significant
- *c1*, *c2*, *cm* are confidence intervals of *s1*, *s2*, *sm*
- *p1*, *p2* are prediction intervals of *s1*, *s2*; I don't have *pm* because I didn't find a definition in the literature
- *scp1*, *scp2* are Uri Simonsohn's small telescopes thresholds for *s1*, *s2*
- *scpd1*, *scpd2* are my extension to Uri's small telescope method that takes into account the observed effect size
- . (*dot*) combines two statistics to yield a complete measure. For example, *d1.c2* means that $d1_{sdz}$ is contained in *c2*; *c1.c2* means *c1* and *c2* overlap; *d1.scp2* means $d1_{sdz} \geq scp2$

Caution The notation d is potentially ambiguous. When used in measures it means d_{sdz}, the standardized observed effect size. Elsewhere it may d_{pop}, the population effect size. I try to be clear in the text.

measure	meaning
sig1	*pval1* is significant
sig2	*pval2* is significant
sigm	*pvalm* is significant
sdir	*d1* and *d2* are in same direction, i.e., have same sign
d1.c2	*d1* is contained in *c2*
d2.c1	*d2* is contained in *c1*
d1.cm	*d1* is contained in *cm*
d2.cm	*d2* is contained in *cm*
dm.c1	*dm* is contained in *c1*
dm.c2	*dm* is contained in *c2*
c1.c2	*c1* and *c2* overlap
c1.cm	*c1* and *cm* overlap
c2.cm	*c2* and *cm* overlap
d1.p2	*d1* is contained in *p2*
d2.p1	*d2* is contained in *p1*
dm.p1	*dm* is contained in *p1*
dm.p2	*dm* is contained in *p2*
p1.p2	*p1* and *p2* overlap
d1.scp2	*d1* exceeds *scp2*
d2.scp1	*d2* exceeds *scp1*; note that Uri's published small telescope method is this one and requires that *sig2* also hold
dm.scp2	*dm* exceeds *scp2*
dm.scp1	*dm* exceeds *scp1*
d1.scpd2	*d1* exceeds *scpd2*
d2.scpd1	*d2* exceeds *scpd1*
dm.scpd2	*dm* exceeds *scpd2*
dm.scpd1	*dm* exceeds *scpd1*
big1	$d1 \geq d2$
big2	$d2 \geq d1$

Precursor Statistics

Performance is critical, because I apply the measures to many, many replications (more than 100 million in default full runs). The code computes precursor statistics that make the measures very fast. It computes each precursor statistic as early as possible in the pipeline: it computes what it can before doing any simulations, then what it can on each simulated study individually, and finally statistics that depend on the actual replications. I took care to implement the code in a vectorized fashion to operate on individual studies and replications in a single gulp.

Statistics computed before simulation

Statistics computed before the simulation cannot depend on the data, of course. In addition to values that truly do not depend on the data (notably Uri Simonsohn's small telescope thresholds), I precompute expensive statistics on a parameter grid, then later interpolate to get the values for specific data. For interpolation, I use `aspline` from the akima package.

Small telescope thresholds

Uri Simonsohn's small telescope thresholds depend only on the sample sizes of the original and replica studies. In Uri's words, the method "evaluates the original study's *design* rather than its result — the telescope rather than the planet". I extended Uri's method to take into account the observed effect size in the original study. This started out as my misunderstanding of Uri's published method. Since I had the code, I decided to keep it.

Uri's method first computes the smallest population effect size that would confer 33% power to the original study for seeing the smallest possible significant effect; Uri calls this $d_{33\%}$. Then it computes the smallest effect size the replica study could see consistent with the hypothesis that the true effect size is $\geq d_{33\%}$. I call this d_{close}. If the replica observes an effect size $\geq d_{close}$, the method deems it close enough to the original effect size to be valid; conversely, if the replica effect size is $< d_{close}$, the method deems it too small to have been really detected by the original study.

My extended (effect size dependent) method differs only in the first step: it computes the smallest population effect size that would confer adequate power (default 33% as in Uri's method) for the original study to see the *original* effect size rather than the *smallest possible* significant effect size.

Notation for small telescope thresholds

math	code	meaning
$n1$, $n2$	`n1`, `n2`	sample sizes of the two studies
pwr	`scope.power`	required power of original study; Uri fixes this at 33%
α	`sig.level`	significance level
d_{sig}	`d.sig`	minimum significant effect size for a sample size of $n1$; used in Uri's version of the method
$d1_{sdz}$	`d1.sdz`	observed effect size in the original study; used in my effect size dependent version of the method
d_{scope}	`d.scope`	minimum population effect size that would confer pwr power to the original study; Uri calls this $d_{33\%}$
d_{close}	`d.close`	minimum acceptable observed effect size in the replica

Functions from d2t document

These functions use the identities $d = t\sqrt{2n/n^2}$ and $t = d\sqrt{n^2/2n}$ to convert between standardized effect sizes (d) and t-statistics (t) thes use base R's `pt` functions to convert t-statistics into p-values.

- `pval2d` converts p-value to standardized effect size
- `p_d2t()` is a wrapper for R's `pt()` that expresses the t-distribution cumulative probability in d units

Math

$$
\begin{aligned}
&\text{Uri's method} \\
d_{sig} &= pval2d(n1, \alpha) \\
d_{scope} &= d0 \mid p\_d2t(n1, d_{sig}, d0) = pwr \\
d_{close} &= d \mid p\_d2t(n2, d, d_{scope}) = \alpha \\
&\text{effect size dependent method} \\
d_{scope} &= d0 \mid p\_d2t(n1, d1_{sdz}, d0) = pwr \\
d_{close} &= d \mid p\_d2t(n2, d, d_{scope}) = \alpha
\end{aligned}
$$

R

For Uri's method, the software performs the above computation on all pairs of simulated sample sizes (`n1`, `n2`). For the effect size dependent version, the software adds a grid of $d1_{sdz}$ values.

I show here the code that implements the core computations and not the outer loop that invokes the core on all n1, n2 pairs or n1, n2, d1.sdz triples. The code uses R's uniroot function to solve the equations for d_{scope} and d_{close}. It wraps uniroot in suppressWarnings to shut up an annoying warning from pt about possibly not attaining full precision.

```
## Uri's method
d.sig=pval2d(n1,sig.level);
d.scope=uniroot(function(d0) p_d2t(n1,d=d.sig,d0,lower.tail=F)-scope.power,interval=c(0,10))$root;
d.close=suppressWarnings(uniroot(function(d) p_d2t(n2,d,d0=d.scope)-scope.close,c(-10,10))$root);

## effect size dependent method
d.scope=uniroot(function(d0) p_d2t(n1,d=d1.sdz,d0,lower.tail=F)-scope.power,interval=c(0,10))$root;
d.close=suppressWarnings(uniroot(function(d) p_d2t(n2,d,d0=d.scope)-scope.close,c(-10,10))$root);
```

The software stores the results of Uri's method in a matrix scope whose row and column names are the values of n1, n2 and whose values are d.close. It stores the effect size dependent results in a data frame scopd whose columns are n1, n2, d1.sdz, and d.close.

Confidence and prediction intervals

Confidence intervals approximately indicate the range of likely values for the population effect size. Technically, the confidence level is the probability that the interval contains the population effect size.

Confidence intervals depend on the sample size and observed effect size of an individual study. The software precomputes confidence intervals on a grid of these values. The main work is done by the ci_d2t function described in the d2t document. The result is a data frame confvl whose columns are as follows.

Column names of confvl data frame

name	meaning
n	sample size
d.sdz	standardized observed effect size
d.lo, d.hi	bounds of confidence interval

Prediction intervals estimate the range of effect sizes we're likely to observe in a replication.

Prediction intervals depend on the sample sizes of the original and replica studies and the observed effect size of the original study. The software precomputes prediction intervals on a grid of these values. The main work is done by the pi_d2t function in the d2t document. The result is a data frame predvl whose columns are as follows.

Column names of predvl data frame

name	meaning
n1, n2	sample sizes of the two studies
d1.sdz	standardized observed effect size of the original study
d.lo, d.hi	bounds of prediction interval

Statistics computed on individual studies

The following table lists the statistics computed on individual studies. Most are obvious.

variable	meaning	details
`mean0`, `mean1`, `sd0`, `sd1`	means and standard deviations of group0 and group1	
`d.raw`	raw effect size	`mean0-mean1`
`sd`	pooled standard deviation	`sqrt((sd0^2+sd1^2)/2)`; see explanation below
`d.sdz`	standardized effect size	`d.raw/sd`
`pval`	p-value computed from t-distribution	`d2pval(n,d.sdz)`; `n` is sample size, `d2pval` defined in d2t document
`d.abs`, `d.sign`	absolute value and sign of d.sdz	`d.sign` is -1 for negative, 1 for positive, 0 for the very unlikely case that `d.sdz` is exactly 0
`ci.lo`, `ci.hi`	lower and upper bounds of confidence interval	interpolated from precomputed `confvl` matrix
`w.meta`, `d.meta`	inverse variance weight and weighted effect size for fixed effect meta-analysis	see explanation below

In the current implementation, the software discards `mean0`, `mean1`, `sd0`, `sd1`, and `d.raw` early in the workflow, since they're not used later. This is a minor optimization.

Pooled standard deviation

My code for pooled standard deviation, though independent of the sample size n, is correct. Here's the derivation.

$$
\begin{aligned}
&\text{standard formula for sample sizes n0, n1}\\
pooled.sd &= sqrt((n0-1)sd0^2 + (n1-1)sd1^2)/(n0+n1-2)\\
&\text{when n0=n1=n}\\
&= \sqrt{((n-1)sd0^2 + (n-1)sd1^2)/(n+n-2)}\\
&= \sqrt{((n-1)(sd0^2 + sd1^2))/2(n-1)}\\
&= \sqrt{(sd0^2 + sd1^2)/2}
\end{aligned}
$$

Values for fixed-effect meta-analysis

The code computes the inverse variance weight and weighted effect size for fixed-effect meta-analysis on each study, then later combines these values to yield the final meta-analytic statistics for pairs of studies.

The variance in question is the sampling variance of the estimated effect size (d_{sdz}), not the variance of the data! The sampling distribution of d_{sdz} is a noncentral t-distribution centered on d_{sdz} with $2(n-1)$ degrees of freedom. The weighted effect size is simply d_{sdz} times the weight. I use the `sd_d2t` function in the d2t document to get the standard deviation, then square it to get the variance.

The R code is

```
## sd_d2t function in Statistical Notes section
w.meta=1/(sd_d2t(n,d.sdz)^2);
d.meta=d.sdz*w.meta;
```

Why fixed-effect meta-analysis? With two studies it's possible to do fixed- or random-effect but not mixed-effect which needs at least three studies. I chose fixed-effect for pragmatic performance reasons. I tested the random-effect calculation from metafor; it adds ~1 minute per replication which is way too slow for this simulation.

Statistics computed on replications

Several of the statistics computed on replications simply interpolate or directly read-out statistics precomputed before the simulation [see above]](#statistics-computed-before-simulation).

variable	meaning	details
`pi.lo`, `pi.hi`	bounds of prediction intervals	interpolated from precomputed `predvl` data frame
`scp1`, `scp2`	small telescope thresholds (Uri's original version)	directly read-out from precomputed `scope` matrix
`scpd1`, `scpd2`	effect size dependent small telescope thresholds (my extended version)	interpolated from precomputed `scopd` data frame

The rest are meta-analytic. A key point is that the sampling distribution of the meta-effect size is normal with standard deviation $\sqrt{1/sum\ of\ weights}$. I checked my results against metafor but don't have a good intuition for the approach.

variable	meaning	details
`w.sum`	sum of weights for the two studies	
`meta.d`	meta-effect size	sum of the weighted effect sizes divided by `w.sum`
`meta.abs`	absolute value of `meta.d`	`abs(meta.d)`
`meta.sd`	standard deviation of distribution of `meta.d`	`sqrt(1/(w.sum))`
`meta.pval`	p-value of `meta.d`	`2*pnorm(-abs(meta.d/meta.sd))`
`meta.ci`	distance from `meta.d` to bounds of confidence interval	`qnorm(p=0.5+conf.level/2,sd=meta.sd)`
`meta.ci.lo`, `meta.ci.hi`	bounds of meta-analytic confidence interval	`meta.d-meta.ci`, `meta.d+meta.ci`

Implementation of measures

With the precursor statistics in hand, the measures are trivial. Here is the code. In this code, `s1`, `s2` are data frames containing statistics for the two studies; `p1`, `p2` are prediction intervals relative to the effect sizes of the two studies; and `$` is R's standard notation for accessing the columns of a data frame. `between` is a simple function that tests whether the first argument is between the other two: `between=function(x,lo,hi) x>=lo&x<hi`.

```
## apply the rules. do them in-line for efficiency
## use concise terms so result won't be too wide to view
##   d{12m} means effect size for s1, s2, meta
##   c{12m} means confidence interval for s1, s2, meta
##   scp{12} is small telescope threshold for s1, s2
##   scps{12} is effect size dependent small telescope threshold for s1, s2
##   term.term means compare the two terms, eg, d1.c2 means s1 d.sdz in s2 confidence interval
##
## significant: s1,s2,meta
```

```
sig1=s1$pval<=sig.level;
sig2=s2$pval<=sig.level;
sigm=meta.pval<=sig.level;
## same direction (ie, sign): s1,s2
sdir=s1$d.sign==s2$d.sign;
## d in confidence interval: each vs other two
d1.c2=between(s1$d.sdz,s2$ci.lo,s2$ci.hi);
d2.c1=between(s2$d.sdz,s1$ci.lo,s1$ci.hi);
d1.cm=between(s1$d.sdz,meta.ci.lo,meta.ci.hi);
d2.cm=between(s2$d.sdz,meta.ci.lo,meta.ci.hi);
dm.c1=between(meta.d,s1$ci.lo,s1$ci.hi);
dm.c2=between(meta.d,s2$ci.lo,s2$ci.hi);
## confidence intervals overlap: all 3 unique pairs
##   thanks to https://fgiesen.wordpress.com/2011/10/16/checking-for-interval-overlap/
##   for simple calculation I should have known already :)
c1.c2=s1$ci.lo<=s2$ci.hi & s2$ci.lo<=s1$ci.hi;
c1.cm=s1$ci.lo<=meta.ci.hi & meta.ci.lo<=s1$ci.hi;
c2.cm=s2$ci.lo<=meta.ci.hi & meta.ci.lo<=s2$ci.hi;
## d in prediction interval: each d vs two intervals we have
d1.p2=between(s1$d.sdz,p2$pi.lo,p2$pi.hi);
d2.p1=between(s2$d.sdz,p1$pi.lo,p1$pi.hi);
dm.p2=between(meta.d,p2$pi.lo,p2$pi.hi);
dm.p1=between(meta.d,p1$pi.lo,p1$pi.hi);
## prediction intervals overlap
p1.p2=p1$pi.lo<=p2$pi.hi & p2$pi.lo<=p1$pi.hi;
## d >= small telescope boundary. use d.abs instead of d.sdz to handle negative values
d1.scp2=(s1$d.abs>=scp2)&sdir;
d2.scp1=(s2$d.abs>=scp1)&sdir;
dm.scp2=(meta.abs>=scp2)&sdir;
dm.scp1=(meta.abs>=scp1)&sdir;
## d >= effect size dependent small telescope boundary
##   need as.vector else R treats it as matrix. screws up naming in data.frame
d1.scpd2=as.vector((s1$d.abs>=scpd2)&sdir);
d2.scpd1=as.vector((s2$d.abs>=scpd1)&sdir);
dm.scpd2=as.vector((meta.abs>=scpd2)&sdir);
dm.scpd1=as.vector((meta.abs>=scpd1)&sdir);
## d2 bigger (actually more extreme) than d1
big1=(s1$d.abs>=s2$d.abs)&sdir;
big2=(s1$d.abs<=s2$d.abs)&sdir;
## return as data.frame. downstream code uses it as environment
data.frame(sig1,sig2,sigm,sdir,
  d1.c2,d2.c1,d1.cm,d2.cm,dm.c1,dm.c2,c1.c2,c1.cm,c2.cm,d1.p2,d2.p1,dm.p1,dm.p2,p1.p2,
  d1.scp2,d2.scp1,dm.scp2,dm.scp1,d1.scpd2,d2.scpd1,dm.scpd2,dm.scpd1,
  big1,big2);
```