Noncentral d2t-Distribution Applied to Measures of Replication Success

Nathan (Nat) Goodman

December 3, 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". The noncentral t-distribution underlies many of these tests. Here I describe my software that repackages R's built-in functions for the noncentral t-distribution to operate on sample size (instead of degrees of freedom), population effect size (instead of the noncentrality parameter), and observed effect size (instead of the t-statistic). I call this the d2t-distribution.

Introduction

Various authors have proposed tests for deciding whether a replication succeeds or fails. I call these tests "measures". I simulated many of these tests across a range of replication conditions and reported 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. I describe the measures themselves in a working paper Measures of Replication Success.

Several measures depends on nuances of the *noncentral t-distribution*. I learned the hard way that the classic (*central*) t-distribution only applies when the NULL is true. When the NULL isn't true, the correct sampling distribution is the *noncentral t-distribution*.

Statistics lessons usually parameterize the noncentral t-distribution by *degrees of freedom* and a *noncentrality parameter*, and the distribution 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* (aka *Cohens's d*) (instead of the t-statistic). For the limited context of my simulation, it's easy to convert between the concepts.

Here I present my software for the noncentral t-distribution parameterized by sample size, population effect size, and observed effect size. I call this the *noncentral d2t-distribution*. The code is in the file `R/stats.R`. I must emphasize that the d2t-distribution is not a new distribution in any sense; it merely repackages the t-distribution.

Simulation Scenario

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. 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. The result is a boolean matrix whose rows represent replication instances and columns represent the measures' results.

Notation

This section defines basic notation that I use throughout. Later sections define specialized notation for specific purposes.

Notation when working with individual studies

variable	meaning
`n`	sample size
`d.pop`	population effect size; this arises in discussion but not the software
`d`	standardized observed effect size; I use `d` in this code, instead of `d.sdz`, to save space since there's no possible ambiguity
`d0`	noncentrality parameter expressed in d_{sdz} units; this is the presumed population effect size
`pval`	p-value
`df`	degrees of freedom
`t`	t-statistic
`ncp`	noncentrality parameter expressed in t units
`sig.level`	significance level

Notation for base R t-distribution and d2t-distribution functions

My d2t-distribution repackages base R's t-distribution functions. The table below lists the base R and corresponding d2t functions.

base R	d2t	meaning
`dt`	`d_d2t`	density function
`pt`	`p_d2t`	cumulative probability function
`qt`	`q_d2t`	quantile function
`rt`	`r_d2t`	random generation function

Additional notation when working with pairs of studies, i.e., replications

The suffixes `1` and `2` denote the two studies, e.g, `n1`, `n2` are the sample sizes of the two studies. It is often convenient to think of study `1` as the original and `2` the replica but this isn't baked into the software.

Transformations between `t`, `d`, `pval`

For the limited context of my simulation, it's easy to convert between the t-statistic, standardized effect size, and p-value.

```
## t-statistic to Cohen's d & p-value
t2d=function(n,t) t*sqrt((2*n)/n^2)
t2pval=function(n,t) 2*pt(-abs(t),df=2*(n-1))

## Cohen's d to t-statistic, p-value, and noncentrality parameter
d2t=function(n,d) d*sqrt(n^2/(2*n))
```

```
d2pval=function(n,d) t2pval(n,d2t(n,d))
ncp=function(n,d) sqrt(n/2)*d

## p-value to t-statistic & Cohen's d
pval2t=function(n,pval) qt(pval/2,df=2*(n-1),lower.tail=F)
pval2d=function(n,pval) q_d2t(n,q=pval/2,lower.tail=F)        # see below for q_d2t
```

d2t-Distribution Functions

I use the transformations above to define a family of d2t-distribution functions analogous to R's t-distribution functions.

```
## density function of d2t-distribution
d_d2t=function(n,d,d0=NULL) {
  df=2*(n-1);
  t=d2t(n,d);
  if (!is.null(d0)) suppressWarnings(dt(t,df=df,ncp=ncp(n,d0)))
  else dt(t,df=df)
}
## distribution function of d2t-distribution
p_d2t=function(n,d,d0=NULL,lower.tail=TRUE) {
  df=2*(n-1);
  t=d2t(n,d);
  if (!is.null(d0)) suppressWarnings(pt(t,df=df,ncp=ncp(n,d0),lower.tail=lower.tail))
    else pt(t,df=df,lower.tail=lower.tail)
}
## quantile function of d2t-distribution
q_d2t=function(n,q,d0=NULL,lower.tail=TRUE) {
  df=2*(n-1);
  if (!is.null(d0)) t=suppressWarnings(qt(q,df=df,ncp=ncp(n,d0),lower.tail=lower.tail))
    else t=qt(q,df=df,lower.tail=lower.tail)
  t2d(n,t);
}
## random generation function of d2t-distribution
r_d2t=function(m,n,d0=NULL) {
  df=2*(n-1);
  if (!is.null(d0)) t=suppressWarnings(rt(m,df=df,ncp=ncp(n,d0))) else t=rt(m,df=df);
  t2d(n,t)
}
```

Mean and Standard Deviation of d2t-Distribution

I was surprised to learn that base R doesn't provide mean and standard deviation functions for even the central t-distribution, so I had to dig it all out from sources on the web. I first found code I could adapt in the UnivRNG package but subsequently found more sources including a Wikipedia's article.

Note the comments about R's `gamma` function and the workaround using `lgamma`. I'm sure I didn't devise this workaround on my own but my notes don't say where I found it. Sorry!

The code makes extensive use of the transformations above.

```
## mean of d2t-distribution
mean_d2t=function(n,d0=NULL) {
```

```
  df=2*(n-1);
  if (!is.null(d0)) {
    ncp=ncp(n,d0);
    ## Note: gamma blows up when n>100 or so. use lgamma instead
    ##       theo.mean=sqrt(df/2)*ncp*gamma((df-1)/2)/gamma(df/2)
    theo.mean=sqrt(df/2)*ncp*exp(lgamma((df-1)/2)-lgamma(df/2))
    t2d(n,theo.mean)
  } else 0;
}
## standard deviation of d2t-distribution
sd_d2t=function(n,d0=NULL) {
  df=2*(n-1);
  if (!is.null(d0)) {
    ncp=ncp(n,d0);
    ## Note: gamma blows up when n>100 or so. use lgamma instead
    ##       theo.mean=sqrt(df/2)*ncp*gamma((df-1)/2)/gamma(df/2)
    theo.mean=sqrt(df/2)*ncp*exp(lgamma((df-1)/2)-lgamma(df/2))
    theo.var=(1+ncp^2)*df/(df-2)-(theo.mean^2)
    theo.sd=sqrt(theo.var)
    t2d(n,theo.sd)
  } else
    (sqrt(2*n)/n)*sdt(2*(n-1));
}
## standard deviation of central d2t-distribution
sdt=function(df) sqrt(df/(df-2))
```

Confidence and Prediction Intervals of d2t-Distribution

Additional notation for confidence and predictions intervals

variable	math symbol	meaning
`d.lo`, `d.hi`	d_{lo}, d_{hi}	lower and upper bounds of interval
`p.lo`, `p.hi`	p_{lo}, p_{hi}	lower and upper probability cutoffs for interval
`conf.level`	c	confidence level for confidence interval

Confidence intervals

The bounds of the confidence interval are the smallest and largest *population* effect sizes for which the observed effect size d_{sdz} would be significant. I started out using the confidence intervals from base R's `t.test` function but later learned these are for raw, not standardized, effect sizes. I learned about confidence intervals for standardized effect sizes from Uri Simonsohn's blog post We cannot afford to study effect size in the lab and adapted the confidence interval function from Uri's code accompanying that post.

Note that $p\_d2t()$ used in the math (`p_dt` in the code) is the cumulative probability function for the d2t-distribution defined above.

Math

$$
\begin{aligned}
p_{lo} &= c/2 \\
p_{hi} &= 1 - p_{lo} \\
d_{lo} &= d0 \mid p\_d2t(n, d_{sdz}, d0) = p_{lo} \\
d_{hi} &= d0 \mid p\_d2t(n, d_{sdz}, d0) = p_{hi}
\end{aligned}
$$

Code

The code uses R's `uniroot` function to solve the equations for d_{lo} and d_{hi}. The call to `suppressWarnings` is to shut up an annoying warning from `pt` about possibly not attaining full precision.

```
## adapted from http://urisohn.com/sohn_files/BlogAppendix/Colada20.ConfidenceIntervalsForD.R
ci_d2t=function(n,d,conf.level=0.95) {
  p.lo=(1-conf.level)/2;
  p.hi=1-p.lo;
  d.lo=suppressWarnings(
    uniroot(function(d0) p_d2t(n,d,d0,lower.tail=F)-p.lo,interval=c(-10,10))$root);
  d.hi=suppressWarnings(
    uniroot(function(d0) p_d2t(n,d,d0,lower.tail=F)-p.hi,interval=c(-10,10))$root);
  c(d.lo,d.hi);
}
```

Prediction intervals

A prediction interval indicates the range of effect sizes we're likely to observe in a replication, given the sample sizes of the original and replica studies and the observed effect size in the original study.

I adapted the prediction interval function from Spence and Stanley's paper and Stanleys predictionInterval package. The method seems to work, but I have no intuition why.

Note that `ci_dt` is the confidence interval function for the d2t-distribution defined above.

```
## adapted from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5028066/ and predictionInterval package
pi_d2t=function(n1,n2,d,ci1=NULL,ci2=NULL,pred.level=0.95) {
  ## # confidence intervals for the two studies
  if (is.null(ci1)) ci1=ci_d2t(n1,d,pred.level);
  if (is.null(ci2)) ci2=ci_d2t(n2,d,pred.level);
  ## lower and upper bounds for the two confidence intervals
  d.lo1=ci1[1];
  d.hi1=ci1[2];
  d.lo2=ci2[1];
  d.hi2=ci2[2];
  ## lower and upper bounds for the prediction interval
  d.lo=d-sqrt((d-d.lo1)^2+(d.hi2-d)^2);
  d.hi=d+sqrt((d-d.lo2)^2+(d.hi1-d)^2);
  c(d.lo,d.hi);
}
```

Discussion

The noncentral t-distribution underlies many statistical tests (*measures*) for deciding whether a replication passes or fails. In this document I describe my software for the *noncentral d2t-distribution*; this software wraps R's built-in t-distribution functions to operate on sample size (instead of degrees of freedom), population effect size (instead of the noncentrality parameter), and observed effect size (instead of the t-statistic). The

code is in the file `R/stats.R`. The d2t-distribution is not a new distribution in any sense; it's merely a repackaging of the t-distribution.

I describe the measures themselves in a working paper Measures of Replication Success which relies heavily on the d2t functions in the present document. I describe the results of the simulation in another 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.