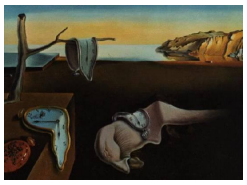

Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

Efficient Calculation of Improved Simultaneous Testing Bands for QQplots

Eric Weine
University of Chicago

Mary Sara McPeek
University of Chicago

Abstract

Quantile-Quantile plots (QQplots) are often difficult to interpret because it is unclear how large the deviation from the theoretical distribution must be to indicate a lack of fit. Current packages and algorithms towards this end either do not ensure a robust Type I error rate, are too slow, or are under-powered to deviations in the tails of the distribution. In this paper, we present an efficient algorithm that computes simultaneous testing bands for QQplots. These bands have a variety of desirable properties, including being fast to compute and being equally sensitive to deviations in all parts of the null distribution, including the tails.

Keywords: QQplots, Equal Local Levels, Kolmolgorov-Smirnov, GWAS, Multiple Testing.

1. Introduction

Quantile-Quantile plots (QQplots) are a common statistical tool used to judge if a sample comes from a specified distribution (Wilk and Gnanadesikan 1968). Despite their ubiquity, they are often difficult to interpret because it is challenging to determine if the magnitude of the deviation from the specified distribution is large enough to indicate a lack of fit as opposed to sampling variance. To make this determination, it is useful to put testing bands on a QQplot.

A number of methods have been created towards this end. First, some available software puts pointwise testing bands onto QQplots, which conduct α level tests on each order statistic (Almeida, Loy, and Hofmann 2017). This method is clearly insufficient, as it ignores the multiple testing problem and thus provides no guarantee on the Type I error rate of testing the global null hypothesis that all of the data come from the specified distribution. Second, the Komolgorov-Smirnov (KS) test is a common method for creating joint bands (Kolmogorov 1941; Smirnov 1944). While this method ensures

the correct Type I error by using the distribution of the KS statistic, this test suffers from very low power under a variety of reasonable alternatives (Aldor-Noiman, Brown, Buja, Rolke, and Stine 2013). In the same work, Aldor-Noiman et al. introduced a much more powerful test (under most alternatives) called the Tail Sensitive (TS) test. This test uses simulation to conduct an η level test on each order statistic such that the overall test has an α level Type I error. This test has a variety of desirable properties, but because it requires simulation it is extremely slow in large datasets.

In what follows, we present a method that yields the same results as the TS test but does so substantially faster because of an efficient recursive calculation. We then demonstrate this test on a few multiple testing scenarios including Type I error calibration and genetic studies.

2. Methods

2.1. Local Levels For Global Null Hypothesis Testing

Suppose we have observations

$$X_1, ..., X_n \overset{iid}{\sim} F,$$

and we are interested in conducting the following hypothesis test

$$H_0 : F = F_0 \text{ vs. } H_A : F \neq F_0$$

with level α, where all parameters of F_0 are known. One approach to this problem, referred to as "local levels," is to conduct separate hypothesis tests on each of the order statistics $X_{(i)}$, where each test has Type I error rate η_i (Gontscharuk, Landwehr, Finner *et al.* 2016). Then, we reject the global null hypothesis if at least one of the n tests results in a rejection. That is, we construct a set of intervals

$$(h_1, g_1), ..., (h_n, g_n),$$

and we reject H_0 if for any i

$$X_{(i)} \notin (h_i, g_i).$$

If we know the local level η_i, then

$$h_i = F_i^{-1}(\eta_i/2) \text{ and } g_i = F_i^{-1}(1 - \eta_i/2), \tag{1}$$

where F_i is the CDF of the i^{th} order statistic. Thus, the difficulty is in determining an appropriate vector of local levels $(\eta_1, ..., \eta_n)$ for which the resulting global level is α. In our case, we want to create testing bands that are "agnostic" to any alternative distribution. By this, we mean that we would like to design a test that applies equal scrutiny to each order statistic, and thus we set

$$\eta_1 = \eta_2 = ... = \eta_n = \eta. \tag{2}$$

This is known as "equal local levels."

Given the sample size n and the desired level α we present the following algorithm for obtaining η in two steps:

1. Given a proposed testing region $(h_1, g_1), ..., (h_n, g_n)$, where $h_i < g_i$ for $i = 1, ..., n$, find the probability of $X_{(1)}, ..., X_{(n)}$ falling outside of this region under the null.

2. Using the algorithm described in step 1, conduct a binary search on the space of the local level η. This involves proposing a value of η, calculating the corresponding region $(h_1, g_1), ..., (h_n, g_n)$ using equation (1), and then using step 1 to check if the region achieves the desired global Type I error rate α.

We now describe these two steps in more detail:

Step 1: First, note that under the null hypothesis,

$$F_0(X_1), ..., F_0(X_n) \overset{iid}{\sim} U(0, 1).$$

Thus, without loss of generality, we can build the algorithm for step 1 assuming that

$$X_1, ..., X_n \overset{iid}{\sim} U(0, 1)$$

We are trying to calculate the following probability

$$\alpha = P_0\Big(\bigcup_{i=1}^{n} X_{(i)} \notin (h_i, g_i)\Big)$$

Where P_0 is the probability under the null hypothesis. It is easier to calculate this as

$$\alpha = 1 - P_0\Big(\bigcap_{i=1}^{n} X_{(i)} \in (h_i, g_i)\Big)$$

To calculate this value, it is easier to transform the events over order statistics to multinomial events. To do this, we introduce the following notation:

Let $b_1, ..., b_{2n}$ be the sorted values of $h_1, ..., h_n, g_1, ..., g_n$ in ascending order. We also define $b_0 = 0$ and $b_{2n+1} = 1$. We will divide the interval (b_0, b_{2n+1}) into $2n + 1$ bins, where bin 1 is (b_0, b_1), bin 2 is (b_1, b_2), ..., and bin $2n+1$ is (b_{2n}, b_{2n+1}). Note that because the uniform distribution is continuous, the probabilities associated with the endpoints of the bins are 0, so it does not matter if the left and right boundaries of these intervals are open or closed. Let $N_j = \sum_{i=1}^{n} \mathbb{1}\Big(X_i \in (b_{j-1}, b_j)\Big)$ denote the random variable that counts the number of X's falling into bin j, for $1 \leq j \leq 2n + 1$, and let $S_k = \sum_{j=1}^{k} N_j$ be the kth partial sum of the N's, for $1 \leq k \leq 2n + 1$. The key to the algorithm is that the following two events are the same:

$$\{X_{(i)} \in (h_i, g_i) \text{ for } i = 1, ..., n\} = \{l_k \leq S_k \leq u_k \text{ for } k = 1, 2, ..., 2n\},$$

where

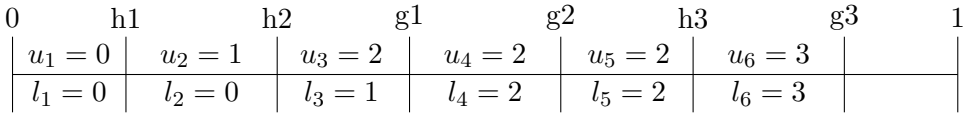

Figure 1: Example of how l_k and u_k, $k = 1, ..., 2n$, depend on the relative positions of $h_1, ..., h_n$ and $g_1, ..., g_n$, for a case with n = 3.

$$u_k = \begin{cases} 0, & \text{if } k = 1 \\ \sum_{i=1}^{k-1} \mathbb{1}(b_i \in \{h_1, ..., h_n\}), & \text{otherwise} \end{cases}$$

$$l_k = \sum_{i=1}^{k} \mathbb{1}(b_i \in \{g_1, ..., g_n\})$$

(See Figure 1 above for intuition). Finally, we define

$c_w^{(k)} = P_0(S_k = w \text{ and } l_j \leq S_j \leq u_j \text{ for } j = 1, ..., k-1), \text{for } k = 1, ..., 2n \text{ and } w = 1, ..., n.$

Our basic approach to step 1 is a recursive calculation of these $c_w^{(k)}$ values. Then, the global level α could be computed as $1 - c_n^{(2n)}$ in the general case. However, for the special case in which $(h_1, g_1), ..., (h_n, g_n)$ are derived from equal local levels, i.e. equation (2) holds, then as a result of the symmetry in the problem, we need only calculate $c_w^{(k)}$ for $k = 1, ..., n + 1$ instead of $k = 1, ..., 2n$. This shortcut is implemented below in the pseudocode for step 1. The algorithm calculates the probability of a valid allocation (under the null hypothesis) of points in bins by looping over each bin and making a recursive calculation.

Algorithm 1 Calculate Type I error α from proposed rejection region

Input: Vector of lower bound values $(h_1, ..., h_n)$, vector of upper bound values $(g_1, ..., g_n)$, where we require $h_i < g_i$ for $i = 1, ..., n$.

get_level_from_bounds_two_sided$(h_1, ..., h_n, g_1, ..., g_n)$

1: $b_1, ..., b_{2n} \leftarrow$ **sort**$(h_1, ..., h_n, g_1, ..., g_n)$
2: $c_0^{(1)} \leftarrow (1 - b_1)^n$
3: $l_k \leftarrow 0$
4: $u_k \leftarrow 0$
5: **for** $k = 2, ..., n + 1$ **do**
6: **if** $b_{k-1} \in \{h_1, ..., h_n\}$ **then**
7: $u_k \leftarrow u_{k-1} + 1$
8: **end if**
9: **if** $b_k \in \{g_1, ..., g_n\}$ **then**
10: $l_k \leftarrow l_{k-1} + 1$
11: **end if**
12: **for** $j = l_k, ..., u_k$ **do**
13: $c_j^{(k)} \leftarrow 0$
14: **for** $m = l_{k-1}, ..., min(u_{k-1}, j)$ **do**
15: $c_j^{(k)} \leftarrow c_j^{(k)} + c_m^{(k-1)} * dbinom(x = j - m, size = n - m, prob = \frac{(b_k - b_{k-1})}{(1 - b_{k-1})})$
16: **end for**
17: **end for**
18: **end for**
19: $1 - \alpha \leftarrow 0$
20: **for** $l = l_n, ..., u_n$ **do**
21: $1 - \alpha \leftarrow 1 - \alpha + \frac{c_l^{(n)} * c_{n-l}^{(n+1)}}{dbinom(x=l, size=n, prob=b_n)}$
22: **end for**
23: **return** α

end

Now, we derive the algorithm. **Initialization:** (Lines 2 through 4 of Algorithm 1)

$$
\begin{aligned}
c_0^{(1)} &= P_0(S_1 = 0) \\
&= P_0\Big(\bigcap_{i=1}^{n} \{X_i \in (b_1, 1)\}\Big) \\
&= \prod_{i=1}^{n} P_0\Big(X_i \in (b_1, 1)\Big) && \text{(by independence)} \\
&= (1 - b_1)^n && \text{(Since each } X_i \sim\ U(0,1) \text{ under the null)}
\end{aligned}
$$

Also, we initialize $l_1 = 0$ and $u_1 = 0$ because the only allowable partial sum is 0 in the

first bin. **Recursion:** (Line 15 of Algorithm 1)

$$
\begin{aligned}
c_j^{(k)} &= P_0(S_k = j \text{ and } l_q \le S_q \le u_q \text{ for } q = 1, ..., k-1) \\
&= P_0\Big(\bigcup_{m=l_{k-1}}^{min(j,u_{k-1})} \{S_{k-1} = m \text{ and } N_k = j - m \text{ and } l_q \le S_q \le u_q \text{ for } q = 1, ..., k-2\}\Big) \\
&= \sum_{m=l_{k-1}}^{min(j,u_{k-1})} P_0(S_{k-1} = m \text{ and } N_k = j - m \text{ and } l_q \le S_q \le u_q \text{ for } q = 1, ..., k-2) \\
&= \sum_{m=l_{k-1}}^{min(j,u_{k-1})} P_0(S_{k-1} = m \text{ and } l_q \le S_q \le u_q \text{ for } q = 1, ..., k-2) * \\
&\quad P_0(N_k = j - m | S_{k-1} = m \text{ and } l_q \le S_q \le u_q \text{ for } q = 1, ..., k-2) \\
&= \sum_{m=l_{k-1}}^{min(j,u_{k-1})} c_m^{(k-1)} * P_0(N_k = j - m | S_{k-1} = m) \\
&= \sum_{m=l_{k-1}}^{min(j,u_{k-1})} c_m^{(k-1)} * P(B = j - m), \text{where } B \sim Binomial\left(n - m, \frac{b_k - b_{k-1}}{1 - b_{k-1}}\right),
\end{aligned}
$$

which is easily computed.

As noted above, in the general case, we could use this recursion to obtain $c_n^{(2n)}$, and then compute $\alpha = 1 - c_n^{(2n)}$. However, in the special case of equal local levels, we can use symmetry to speed the calculation by only computing the $c_w^{(k)}$ values for $k = 1, ..., n+1$ and $w = 1, ..., n$, and performing the calculation on line 21 of Algorithm 1, as we now demonstrate. First, we define the following values:

$$
\begin{aligned}
\tilde{u}_k &= \begin{cases} 0, & \text{if } k = 2n \\ \sum_{i=k+1}^{2n} \mathbb{1}(b_i \in \{g_1, ..., g_n\}), & \text{otherwise} \end{cases} \\
\tilde{l}_k &= \sum_{i=k}^{2n} \mathbb{1}(b_i \in \{h_1, ..., h_n\}) \\
T_k &= \sum_{j=k+1}^{2n+1} N_j = n - S_k
\end{aligned}
$$

Now, we make the following observations:

(1) With equal local levels, $g_i = 1 - h_{n+1-i}$ for $i = 1, ..., n$ since $F^{-1}_{Beta(i,n+1-i)}(1 - \frac{\eta}{2}) = 1 - F^{-1}_{Beta(n+1-i,i)}(\frac{\eta}{2})$.

(2) $b_k = 1 - b_{2n+1-k}$ for $k = 1, ..., 2n$ by (1).

(3) $u_k = \tilde{u}_{2n+1-k}$ and $l_k = \tilde{l}_{2n+1-k}$, for $k = 1, ..., 2n$ by (1) and (2).

(4) The random vector $(N_1, ..., N_k)$ has the same distribution as $(N_{2n+1}, ..., N_{2n+2-k})$ for $k = 1, ..., 2n+1$. This follows from the fact that the vector $(X_1, ..., X_n)$ has the same distribution as $(1 - X_1, ..., 1 - X_n)$ (since each X_i is independent uniform) and (2).

(5) The random vector $(S_1, ..., S_k)$ has the same distribution as $(T_{2n}, ..., T_{2n+1-k})$ for $k = 1, ..., 2n$. This follows from (4).

(6)

$$\begin{aligned} c_j^{(k)} &= P_0(S_k = j \text{ and } l_q \le S_q \le u_q \text{ for } q = 1, ..., k-1) \\ &= P_0(T_{2n+1-k} = j \text{ and } \tilde{l}_r \le T_r \le \tilde{u}_r \text{ for } r = 2n+2-k, ..., 2n). \end{aligned}$$

This follows from (3) and (5).

(7) Conditional on S_k, the random vector $(X_{(S_k+1)}, ..., X_{(n)})$ is distributed as the order statistics of $n - S_k$ $U(b_k, 1)$ independent random variables.

(8) The random vector $(S_1, ..., S_r)$ and the random vector $(T_r, ..., T_n)$ are independent conditional on S_r. This follows directly from (7).

Combining the above results, we can write

$$\{X_{(i)} \in (h_i, g_i) \text{ for } i = 1, ..., n\} = \{l_k \le S_k \le u_k \text{ for } k = 1, ..., 2n\}$$

Also, observe that for any $2 \le r \le 2n-1$, we have

$$\{l_k \le S_k \le u_k \text{ for } k = 1, ..., 2n\} =$$

$$\{l_k \le S_k \le u_k \text{ for } k = 1, ..., r \text{ and } T_r = n - S_r \text{ and } \tilde{l}_k \le T_q \le \tilde{u}_k \text{ for } q = r, ..., 2n\}$$

Thus, we can write

$$\begin{aligned} &\sum_{j=l_r}^{u_r} P(S_r = j \text{ and } l_k \le S_k \le u_k \text{ for } k = 1, ..., r-1) \\ &\quad \cdot I(\tilde{l}_r \le n - j \le \tilde{u}_r) \cdot P(\tilde{l}_q \le T_q \le \tilde{u}_r \text{ for } q = r+1, ..., 2n | T_r = n - j) \\ &= \sum_{j=l_r}^{u_r} c_j^{(r)} \cdot I(\tilde{l}_r \le n - j \le \tilde{u}_r) \cdot P(\tilde{l}_q \le T_q \le \tilde{u}_r \text{ for } q = r+1, ..., 2n | T_r = n - j) \\ &= \sum_{j=l_r}^{u_r} c_j^{(r)} \cdot I(\tilde{l}_r \le n - j \le \tilde{u}_r) \cdot \frac{P(\tilde{l}_q \le T_q \le \tilde{u}_r \text{ for } q = r+1, ..., 2n \text{ and } T_r = n - j)}{T_r = n - j} \\ &= \sum_{j=l_r}^{u_r} c_j^{(r)} \cdot \frac{c_{n-j}^{(2n+1-r)}}{\binom{n}{j} b_r^j (1 - b_r)^{n-j}} \end{aligned}$$

Now, if we let $r = n$ above, then we get

$$P_0\Big(\bigcap_{i=1}^{n} X_{(i)} \in (h_i, g_i)\Big) = \sum_{j=l_n}^{u_n} c_j^{(n)} \cdot \frac{c_{n-j}^{(n+1)}}{\binom{n}{j} b_n^j (1 - b_n)^{n-j}}$$

which is line 21 in the psuedo-code above.

Empirically, it seems that our algorithm has computational complexity $O(n^2)$. For

a dense grid of values of n between 10 and $50,000$, the number of recursive steps required (line 15 in algorithm 1 above) scales with n^2. More precisely, it appears that the number of recursive steps required is very close to $8n^2$. While each recursive step itself requires calculating a binomial probability which is $O((n+1)!)$ due to the calculation of the binomial coefficient multiplied by a quantity with powers as large as n, these calculations can be memoized with $O(n^2)$ cost. Thus, these binomial probability calculations do not change the big O complexity of the algorithm.

Step 2: This step is done with a simple binary search over η. We begin by setting $\eta_{upper} = \frac{-log(1-\alpha)}{(2*log(log(n))*log(n))}$ since this is an upper bound on the local level as shown by Gontscharuk et al., and we set $\eta_{lower} = \frac{\alpha}{n}$, as this is the lower bound given by the Bonferroni correction. The pseudocode is shown below:

Algorithm 2 Calculate testing bounds from global level α and sample size n

Input: Local level α, sample size n, tolerance ϵ

$get\_bounds\_two\_sided(\alpha, n, \epsilon)$

1: $\eta_{upper} \leftarrow \frac{-log(1-\alpha)}{(2*log(log(n))*log(n))}$
2: $\eta_{lower} \leftarrow \frac{\alpha}{n}$
3: $\alpha_{mid} \leftarrow \infty$
4: **while** $\frac{\alpha_{mid}-\alpha}{\alpha} > \epsilon$ **do**
5: $\quad \eta_{mid} \leftarrow \frac{\eta_{upper}+\eta_{lower}}{2}$
6: $\quad h_1, ..., h_n \leftarrow qbeta(p = \frac{\eta_{mid}}{2}$, shape1 = c(1:n), shape2 = c(n:1))
7: $\quad g_1, ..., g_n \leftarrow qbeta(p = 1 - \frac{\eta_{mid}}{2}$, shape1 = c(1:n), shape2 = c(n:1))
8: $\quad \alpha_{mid} \leftarrow get\_level\_from\_bounds\_two\_sided(h_1, ..., h_n, g_1, ..., g_n)$
9: **end while**
10: **return** $h_1, ..., h_n, g_1, ..., g_n$

end

3. Local Level Approximations in Large Samples

For sufficiently large values of the number of tests n, it can be expedient to apply an accurate asymptotic approximation of the local level η corresponding to global level α in place of exact computation. Previously, Gontscharuk and Finner showed that the asymptotic local level for global level α and number of tests n is

$$\eta_{asymp} = \frac{log(1-\alpha)}{2log(log(n))log(n)}$$

(Gontscharuk and Finner 2017). However, as they note, this approximation gives poor performance for n even as large as 10^4. To improve this approximation, they propose to add a smaller order correction term, resulting in an approximation of the form

$$\eta_{approx} = \frac{log(1-\alpha)}{2log(log(n))log(n)} \left[1 - c_\alpha \frac{log(log(log(n)))}{log(log(n))}\right], \tag{3}$$

where c_α is chosen empirically. For the values $\alpha = .01, .05$, and $.1$ they chose $c_\alpha = 1.6, 1.3$, and 1.1, respectively. To select these c_α values, the authors calculated the values of η to high precision on a grid of values up until $n = 10,000$. To further test

these approximations, we decided to calculate the values of η with high precision on a grid of values up to $n = 500,000$ for $\alpha = .01, .05$. Based on our evaluation, we found that $c_\alpha = 1.3$ is satisfactory for $\alpha = .05$, but that $c_\alpha = 1.6$ for $\alpha = .01$ was not sufficiently accurate for our purposes (see Figure 2). This approximation begins anti-conservative for small values of n, becomes very accurate for values of n around 2,500, and then becomes and remains substantially conservative for even very large values of n.

In addition to implementing algorithms 1 and 2, which, in principle, can be applied for any n and α, our package offers a faster approximate approach specifically for $\alpha = .01$ and .05. For smaller values of n, up to $100,000$, we have pre-calculated values of η on a dense grid of values of n. If the user inputs a value of n less than or equal to $100,000$, we either return back the pre-computed value of η if n happens to be a grid point, or we use linear interpolation if the value of n is between grid points, which leads to a highly accurate approximation. If the user inputs a value of n greater than $100,000$, we use the asymptotic approximation given in (3), but with $c_\alpha = 1.591$ and 1.3 for $\alpha = .01$ and .05, respectively. We found that $c_\alpha = 1.591$ led to much better performance for $\alpha = .01$ for large values of n, as can be seen in Figure 2. Our package also implements the approximation given in equation (3) for $\alpha = .1$ with $c_\alpha = 1.1$, but we did not pre-compute any grid for these values due to limited computational resources.

4. Examples

As noted above, the algorithm presented is only correct if all of the parameters of F_0 are known, and all $X_1, ..., X_n$ are independent. While both of these assumptions may only hold in a small number of cases, the methods above are still useful in a variety of scenarios.

One of the main advantages of the local levels method, is that it can easily be used to put testing bands onto QQplots by simply graphing each (h_i, g_i) interval. This allows us to examine how a dataset might deviate from some null distribution much better than simply applying a test that yields a binary conclusion. Below, we present a few examples where a QQplot is useful, and where the local levels test seems ideal for assessing deviation from a global null hypothesis.

4.1. Examining the P-value Distribution for Testing Procedures

Suppose we have devised a new testing procedure to test a null hypothesis H_0 with test statistic $T(X_1, ..., X_n)$, and we wish to establish that this test has correct Type I error rate.

Typically, the verification of Type I error rate is done using the following procedure:

(1) Generate m simulated datasets: $(X_1^{(1)}, ..., X_n^{(1)}), ..., (X_1^{(m)}, ..., X_n^{(m)})$ under H_0.

(2) Apply the test T to each of these datasets at level α to yield m reject / not-reject conclusions: $c_1, ..., c_m$, where $c_i = 1$ if H_0 is rejected and 0 otherwise.

(3) Conduct a test to confirm that the mean of $c_1, ..., c_m$ is α.

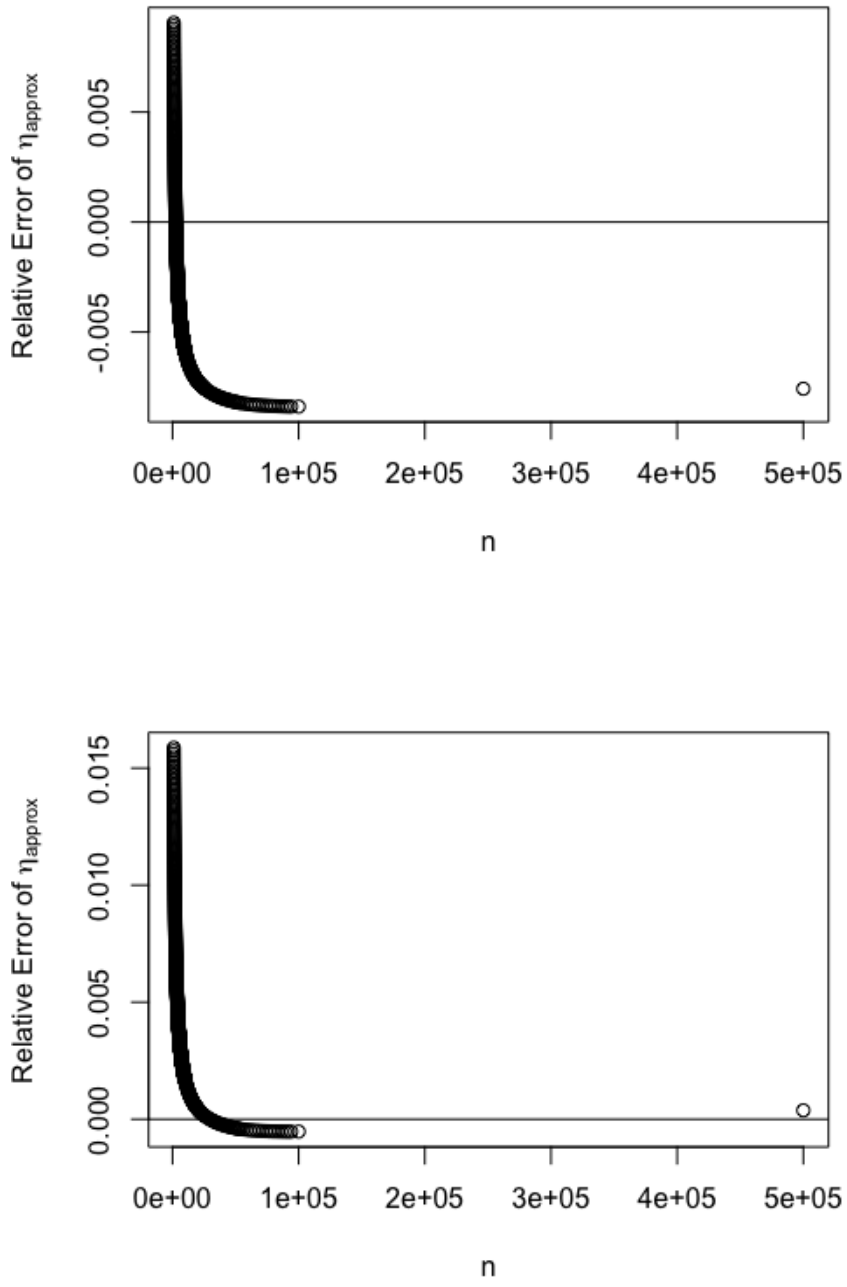

Figure 2: Relative Error of Local Level vs. Number of Tests for η_{approx} for $\alpha = .01$ with $c_\alpha = 1.6$ (top) $c_\alpha = 1.591$ (bottom).

While the above procedure provides reliable information about the Type I error calibration for one level of α, it provides little information about the global calibration of p-values. Instead, we suggest the following procedure:

(1) As above.

(2) Apply the test T to each of these datasets to yield m p-values: $p_1, ..., p_m$.

(3) Apply the local levels procedure to test if $p_1, ..., p_m \overset{iid}{\sim} U(0, 1)$.

This allows us to easily visualize the global calibration of the p-values with just one graph and diagnose any issues if they exist. In Step 3, one could use many different testing bands. However, in the calibration of p-values, we typically don't have the expectation that our p-values would be more likely to deviate from uniform in some regions than others, and so it makes sense to use the local levels test because it is agnostic to the alternative distribution. Moreover, since it is generally most concerning if p-values are not calibrated in the lower tail of the distribution, the local levels test is preferable to the standard KS test because it is much more sensitive in the tails (Aldor-Noiman 2013).

Chi-Square Test for Independence in a 2×2 *Table*

References

Aldor-Noiman S, Brown LD, Buja A, Rolke W, Stine RA (2013). "The power to see: A new graphical test of normality." *The American Statistician*, **67**(4), 249–260.

Almeida A, Loy A, Hofmann H (2017). *qqplotr: Quantile-Quantile Plot Extensions for 'ggplot2'*. R package version 0.0.3 initially funded by Google Summer of Code 2017, URL https://github.com/aloy/qqplotr.

Gontscharuk V, Finner H (2017). "Asymptotics of goodness-of-fit tests based on minimum p-value statistics." *Communications in Statistics - Theory and Methods*, **46**(5), 2332–2342. doi:10.1080/03610926.2015.1041985. https://doi.org/10.1080/03610926.2015.1041985, URL https://doi.org/10.1080/03610926.2015.1041985.

Gontscharuk V, Landwehr S, Finner H, *et al.* (2016). "Goodness of fit tests in terms of local levels with special emphasis on higher criticism tests." *Bernoulli*, **22**(3), 1331–1363.

Kolmogorov A (1941). "Confidence limits for an unknown distribution function." *The annals of mathematical statistics*, **12**(4), 461–463.

Smirnov NV (1944). "Approximate laws of distribution of random variables from empirical data." *Uspekhi Matematicheskikh Nauk*, (10), 179–206.

Wilk MB, Gnanadesikan R (1968). "Probability Plotting Methods for the Analysis of Data." *Biometrika*, **55**(1), 1–17. doi:10.1093/biomet/55.1.1.

Affiliation:

Mary Sara McPeek
Department of Statistics
University of Chicago
924 E 57th St., Chicago, IL
E-mail: mcpeek@uchicago.edu
URL: https://galton.uchicago.edu/~mcpeek/

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd