Dear Reader,

This is a kind of white paper that tries to describe the software that is applicated in the paper:

<https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0299753>

However, it is important to understand that this software can with some very minor changes be used in all kinds of context including EEG fMRI ECG and NIRS.

My papers tend to feel heavy because they are packed with information. This is not enjoyable for anyone, least of all the author. To help you a bit, I will provide a sort of quick user guide to get through the paper in reasonable time. At this point, I would like to try to describe the relationship between the paper and what is truly important, namely the underlying software. I assume you are familiar with imaging, but I will keep things as simple as possible, as you may be working in a completely different field of science. Additionally, I am Dutch, and we tend to be very straightforward, if not childish.

The basic idea is that fMRI is quite unpopular in clinical practice. Most neurosurgeons and psychiatrists avoid fMRI like the devil avoids holy water. Why? Some reliability assessments are not that bad. Nonetheless, the discrepancy between what scientists claim and what clinicians do is quite large. Could this be due to the reliability measures used in science? Do we need different reliability measures, and if so, how can we escape the stinking swamp if the new reliability measures disappoint? In the following text, I will address this under the headings "Clinical Reliability Measures" and "How We Can Improve Clinical fMRI."

**Clinical Reliability Measures**

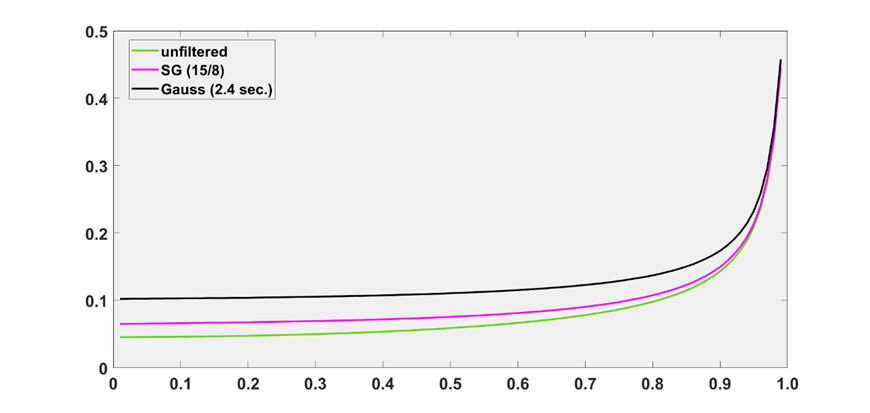
To determine how good fMRI is, correlations are typically made across a population of people. For example, one can measure a connectivity pathway between two areas (connectivity) in 25 people twice. The next step is to form a correlation across the individuals. In our experiment, the group reliability was quite high, certainly much higher than what is usually reported. However, one should be careful not to attribute too much significance to this. Unfortunately, such a group measure is not very suitable for patient research, as neuropsychology is concerned with whether one can reliably assess the patient being treated. Therefore, a reliability measure that works at the individual level is needed.

As you probably know, it is common in fMRI to alternate phases of cognitive actions with phases of rest to induce fluctuations in the fMRI signal. In the cognitive phase, the signal is high, while it is low during the rest phases. Ideally, these brain fluctuations should look similar when measuring the same person twice with the same experiment. To determine reliability within an individual, one simply needs to correlate the time series of a test experiment with a rest experiment. Until now, this has been done very little, probably because the naked truth is not particularly pleasant. Unfortunately, the test-retest reliability of the time series also affects the connectivity that can be detected between areas. Connectivity between two brain areas is defined as the correlation between a time series in area A and a time series in area B.

However, the correlation between two phenomena can never be higher than the test-retest reliability of phenomenon A and the test-retest reliability of phenomenon B. In other words, the connectivity between two areas can never be higher than the measurement accuracy (test-retest reliability) with which one can measure areas A and B. In the following paper, we attempted to capture the test-retest reliability of time series and the closely related detectable connectivity.

The important question is: Is connectivity research on the dodgy side? We fear so, at least if one wants to conduct it at the individual level (it may be different at the group level). However, proving this was significantly more difficult than expected. You are surely familiar with the fact that what sounds simple in theory can be much more challenging in practice. This was also the case in this paper, which is why it became so extensive. The main problem is that fMRI time series are autocorrelated. This means that if an fMRI time series is shifted by one or more data points and then correlated with the unshifted signal, one will find that fMRI time series exhibit quite high autocorrelations or self-similarities. This is, to put it bluntly this mixed shit. When one correlates an autocorrelated time series with another autocorrelated time series, it can happen that a correlation arises purely by chance. However, this random correlation has no significance, as it is caused by the autocorrelated structures of the time series.

To illustrate this, we programmed a small computer simulation, which is presented in another paper. The basic idea is quite simple: generate a randomly generated data series that exhibits autocorrelation and correlate it with another randomly generated data series that has the same autocorrelation. Do this a million times and see how the standard deviation of the correlation distribution looks compared to randomly generated correlations that are not based on autocorrelated time series.



The whole thing is presented up here, but just focus on the green line. What you see is that the standard deviations of the correlations increase quite dramatically as the autocorrelation of the underlying time series rises. On the horizontal axis, the level of autocorrelation is indicated, while the vertical axis shows the standard deviations of the correlations between two randomized time series after generating a million random correlations. In other words, when the autocorrelations in the time series are high, it can happen that high correlations between two purely random events arise purely by chance.

In Holland, we say: "That doesn't smell fresh." So, we calculate the test-retest reliability of two time series, and it can simply be conjured up by chance, a reliability that doesn't actually exist, just because the time series are autocorrelated. What do you do?

One way is to free the time series from autocorrelations, a procedure that is quite common in task-oriented fMRI but, strangely enough, not in reliability and connectivity research. Why? No one knows, but not everything is about logic.

So, we need to remove the autocorrelations from the entire time series using an AR(1) model, which is done with the function **autocorrfilter.m**. This function requires two time series as input. In this function, the weight Zeta (line 14) of the residual autocorrelation is estimated. Zeta is then applied to free the time series from autocorrelations (lines 17 and 18). The correlation between two time series should, in principle, be significantly lower when corrected for autocorrelation compared to uncorrected correlations. The test-retest reliability is indeed dramatically lower when an AR(1) corrector is applied, as can be easily seen in Figure 1f. The same applies, albeit to a lesser extent, to connectivity, as you can see from Table 1. However, it is rumored among economists that this process, although common in imaging, is not exactly the best solution.

An alternative method is to calculate correct confidence intervals using a so-called block bootstrap method. In this case, the level of correlation remains untouched, but the calculation of the confidence interval attempts to account for the fact that the confidence intervals are wider due to autocorrelations than is usually the case.

Doing this in imaging is no small feat, as bootstrapping is quite time-consuming. The problem is: so far, no one has applied block bootstrapping in imaging; it’s just foreign stuff cobbled together by economists. Unlike regular bootstrap methods, which draw random data points from a population, block bootstrapping is based on the idea of drawing larger amounts of neighboring points to retain the autocorrelation of the time series. Different blocks of length n are then concatenated to simulate a time series that exhibits autocorrelation. The length of a block is determined by the lag shift at which no significant autocorrelations occur anymore. The optimal block length is determined by the function **opt\_block.m**, but I should mention at this point that the lines come from Kevin Sheppard.

The determination of the autocorrelated behavior of the time series takes place in lines 49 to 61. Then, in the next step, the critical block length must be determined from the data. This critical step occurs in lines 101 and 105. The input of the **opt\_block.m** function is a time series, and the output is the estimated block length Bstar. Since we usually consider two time series (test, retest, reliability, or connectivity), Bstar is averaged.

The point at which a block starts and ends is taken over by the MATLAB function **stationary\_bootstrap.m**, which also contains code borrowed from Kevin Sheppard. This function requires Bstar as input (how long should the blocks be?), as well as the actual time series and the frequency with which the bootstrap should be executed. It is clear that if 100 bootstraps are drawn, a total of 100 synthetic time series will be created by the function, which are then passed to the output variable bsdata.

However, the real problem was not the block bootstrapping itself, but rather how to integrate block bootstrapping into the connectivity pipeline. The bootstrap method was applied to calculate the correct confidence intervals for the test-retest reliability of connectivity and the actually detectable connectivity. The relationship between the block bootstrap method and the imaging-relevant quantities is organized in the function **bootstrapBlock.m**. This function can be fed with a varying number of input variables. In its simplest form, the function receives two time series (from line 120) with which the confidence interval of the test-retest reliability can be calculated. However, you can also pass four time series (line 71), which will then bootstrap the detectable connectivity.

Using **TrueCon.m**, the detectable connectivity is calculated in line 93. The detectable connectivity is computed with the help of the "connectivity upper bound" as follows:



Where ρ node A represents the test-retest reliability of region A and ρ node B represents the test-retest reliability of region B. The actually detectable connectivity is then calculated from the connectivity upper bound using the following algorithm:

We calculated the detectable connectivity from the observed connectivity and the connectivity upper bound using the following procedure:

**We set observed connectivity to zero when one or two nodes exhibit negative time course reliability. This is a necessary operation since, given the formula above, one cannot take the root out of a negative reliability.**

**Otherwise,**

**If, the observed connectivity is a positive correlation: In this case, we compared the absolute connectivity and absolute connectivity upper bound correlations and took the smaller of the two correlations.**

**Else, if the observed connectivity is negative: we compared the absolute connectivity and connectivity upper bound correlations and took the smaller of the two correlations and made the sign of the result negative.**

**End**

So, that was the whole magic: Essentially, the observed connectivity is replaced by the connectivity upper bound when the observed connectivity is higher than the connectivity upper bound. Please note, the upper bound should not be confused with the upper bound of a confidence interval in this context. The algorithm in **TrueCon.m** starts at line 30 and ends at line 70. Unsurprisingly, the "detectable connectivity" is on average significantly lower than the "normal connectivity." It only gets really interesting when you look at the values in Table 1, which, as hinted, is not a pleasant task.

Although it may not be strictly necessary, the block bootstrap method was also applied to time series that were freed from autocorrelation to account for any minimal remnants of autocorrelation that were not removed by the AR(1) method. To ensure this, all the functions mentioned so far have been integrated into the functions **remove\_AutoBlockBoot\_short.m** and **remove\_AutoBlockBoot.m**. In these two functions, the reliability, normal connectivity, and detectable connectivity are calculated. So far, we have always calculated the bootstraps per brain region relationship-wise per path, but it is clear that a brain consists of multiple regions or paths. In the function **remove\_AutoBlockBoot\_short.m**, the n regions per subject are subjected to the bootstrap method, while in the function **remove\_AutoBlockBoot.m**, all paths of a connectome of a single person are subjected to the bootstrap method. This is done with both raw and AR(1) corrected time series, resulting in an enormous amount of information, which is why the output variable "dat" is quite swollen. To clarify: the abbreviations "test" stand for the test run, "retest" stands for the retest run. "Raw" stands for the normal, non-AR(1) corrected data. "RemAuto" stands for the time series from which the autocorrelation was removed. "Contrue" stands for the detectable connectivity, while "relmax" contains the detectable upper bound of connectivity (connectivity upper bound).

Furthermore, there are the confidence intervals "Low" and "Up," as well as "trans," which stands for the forward Fisher z-transformed data (the data will be averaged later). So, the abbreviation dat.contrue\_test\_remAutoRoi\_Low\_trans found in line 186 refers to the lower confidence interval of the detectable connectivity that has been freed from autocorrelation. So far, we have analyzed all regions or paths per person, but of course, we need this for all individuals. This tour de force takes place in the function **Estimate\_ResAuto\_TrueConBlockBoot.m**.

Now, I welcome you a bit more meticulously because this is where it gets serious. In line 13, the actual fMRI data is loaded. The data consists of the test run (Clean\_SPM\_p\_1) and the retest run (Clean\_SPM\_p\_2). The first dimension is the length of the time series, which in this case is 488 time points. The second dimension is the number of brain regions; in this case, there are 34. The third dimension is the number of subjects; in this case, there are 50. After all this stuff is loaded, there’s a bit of decoration, but the psychologist's eye wants something too. In line 15, the test-retest reliability of the reaction times per person is uploaded. This is a bit quirky because no one else does this, but the idea is quite simple. Suppose a person presses a button 48 times for a memory experiment at a measurement occasion 1. Does this person then show the same reaction times at measurement occasion 2 when the same experiment is conducted again? The trick is straightforward: correlate the 48 reactions at time point 1 with the reactions at time point 2. So, truly no witchcraft involved. We had 50 people, so there are also 50 such correlations. In line 17, the names of the 34 brain regions are uploaded (this is a kind of biological nerdology). If you are not a psychologist or a medical professional, please don’t be put off; they just talk in convoluted terms. In line 19, the codes of the selected subjects are uploaded.

In line 22, it is then determined how many brain regions are present per person. In line 23, the size of the connectivity matrix is determined. So, if there are 34 regions, there are essentially 34 \* 34 possible connectivity correlations. However, the correlations on the diagonal are of little interest because each region naturally correlates 1 with itself. Furthermore, the upper and lower triangles of the connectivity matrix are, of course, equal. Therefore, we only want to extract the lower triangle of the action, which we do in line 25. The extractor function has the advantage that the entire triangular beauty can be transformed into a 1D vector, which is certainly easier when we check the paths of individuals for significance. In line 28, the threshold span for the Dice overlap is determined. This comes a bit early because no dino knows what that is, but it will be explained later. Then in line 31, we have numBoot. This is simple; it refers to the number of block bootstraps conducted at the time series level. "numSimulation" in line 33 is the same thing in green as we say in German. So, pay attention. One can determine the confidence interval of a correlation at the time series level. We’ve talked enough about that, but of course, you can also estimate the confidence intervals of the mean at the population level. There’s nothing wrong with that; if we have 50 boys and 50 girls, we can certainly determine the confidence interval for the average size of the boys and girls. This is also bootstrapped, but not at the time course level, rather at the person level. No magic is needed here because we don’t have all that fuss with the autocorrelations. Easy peasy, as the Americans say. nSamp determines how many individuals are actually evaluated. In my case, it’s 50. The same goes for the bootstrap parameters, which were once set to 10,000. Then the whole thing takes, without exaggeration, 3 days on 20 cores. So, don’t try it; it’s not fun.

After the initial banter, a loop is initialized that collects the statistical stuff for the entire population. The loop starts at line 38 and, not surprisingly, ends at line 116. Well, now that all the data is collected, it needs to be organized into tables so we can explain it to the masses if such a thing exists in science.

After the loop is finished, all possible means are generated. In line 118, the creation of a table is initialized, aiming to calculate the average reliability across individuals and areas. Here, there’s a problem. Theoretically, negative test results regarding reliability are not possible. However, they can still occur. So, how do we deal with this? One way is to simply exclude areas with negative means and treat them as NaN. Another solution, commonly used in psychology, is to set negative correlations to zero. Of course, one might also argue that negative correlations indicate that the system is corrupt and not functioning, and thus, negative correlations must be accepted as such, which naturally lowers the mean. We will calculate all variants here. Furthermore, we calculate the average confidence interval of the reliability correlations.

And now it gets a bit confusing: we also calculate the confidence intervals of the means and standard deviations at the population level. This whole process is reported in Supporting Material Table S1. The averaging process and the determination of the population confidence intervals are driven by the function **average\_rel.m**, whose functionality can be determined with the input variable version. Version 1 calculates the means with negative reliabilities, while Version 2 sets the means to zero or excludes them. Within the function **average\_rel.m**, there is **sampboot.m**, which is used to perform bootstrapping at the population level. It appears in lines 45 and 119. In line 135, a table is generated that reports the average reliability per brain region. In line 147, the table is then saved. This table is reported in Supporting Material Table S2.

We than progresses to line 149 in which we attempt to explore the relationship between brain reliability and behavior. This is, as always, not particularly complicated. Individuals who exhibit unreliable brain activity should also display unreliable behaviors. One must stem from the other in some way. To do this, we correlate the reliability of the brain with the reliability of behavioral measures. Up to line 155, we do this per region, but we do not report it in the paper. In lines 157 to 178, we correlate the averaged brain reliability of a person with the reliability of behavior, setting negative correlations, as mentioned above, to zero. In the main paper, you will then see the scatterplot in Figure 1C that illustrates this relationship.

In line 172, we begin calculating the means of the connectivity matrices. Again, this is done across individuals and paths. For this, we need the **function average\_con.m**. This function not only averages the existing paths and their confidence intervals but also performs a bootstrapping procedure at the population level using the function **sampboot.m**. The entire output is then reported in Table 1 of the main paper and in Table S3 of the Supporting Materials.

Now, let’s talk about the Dice overlap. What is that? Well, it’s nothing painful. Basically, it’s super simple. The idea behind it is that we want to determine reliability at the group level using a test-based method. Imagine we have a connectome with 200 paths measured in 50 individuals. We can first test whether the paths significantly differ from zero at the group level. To do this, we test the 50 observations we have from path A against zero using a t-test. Since we have 200 paths, we conduct 200 tests. We do this not only for a test run but also for a retest run. This happens in lines 194 to 206.

Okay, now that we have that, let’s discuss the Dice overlap. Suppose, in the test run, one hundred paths have become significant, and the same goes for the retest run. However, upon closer inspection, it turns out that only 50 paths commonly differ in both the test and retest run. What both measurements have in common is considered the reproducible portion of the connectome, while what they do not share is regarded as the non-reproducible portion of the connectome. Of course, this can be written in a simple formula:

Dice-Overlap = 2 × (all common paths from test and retest) / (all paths from A + all paths from B)

So, it’s a measure of overlap: 0 means no overlap, and 1 means complete overlap. The problem with this whole thing is that the measure naturally depends on the significance threshold you accept for the p-value. This is always a tricky issue in the imaging field. When you conduct 561 tests, as is the case here, you typically divide the p-value by 561 to guard against false discoveries. But does such an approach even make sense? Does one have the liberty to ask? A little provocation here and there to shake up the old habits? This inquiry happens in the function **dice.m**. As mentioned, it’s a bit of a provocative piece. The provocation lies in the fact that we calculate reproducibility at the group level for a very large range of t-values. The usual Bonferroni correction (0.05/561) yields a reproducibility of 1, which is just as good as not having performed the correction at all. No wonder, when we use the corrected p-value, all paths are significant in both the test and retest.

Now, there’s this bizarre idea among scientists that very small p-values predict good group reproducibility. For the average scientist, the p value can’t be small enough. But that’s all nonsense; the exact opposite is true: the smaller the p-value, the less reproducible the result. This is easily understandable with common sense: the smaller the connectomes become, the less likely they are to overlap. That sounds plausible, but most scientists resist this idea. What about all those projects with such delightfully small p-values? We show the curve trends in Figure 3C of the main paper.

In line 214, we begin conducting the conjunction analysis, which is just a fancy term for taking the smallest t-value from a test and retest run. I don’t want to linger too long on this and explain the details in the paper, as it’s a rather banal process.

We’ve been at this for a while now, but perhaps you remember that at the very beginning of the paper, it was mentioned that reproducibility is usually calculated by correlating across the group. To put it casually: If mommy has high connectivity in path A during the test run and daddy has low connectivity, then you want to recognize the same behavior in the retest run. In our case, we correlate the 50 observations we have for path A in the retest run with the 50 observations in the test run. This begins in the code at line 230. By the way, these correlations are quite high compared to other studies. So, in the conventional sense, we are doing quite well.

Okay, now we’ve roughly gone through the script, and any normal person who is somewhat sane would say, "That’s nice." But as it often goes when you have someone in the group who can program a bit: suddenly, the co-authors have the tendency to get greeeeeat ideas. Once they’ve caught a whiff of it, they won’t let go, as they follow the unspoken motto: Wishing is easy, doing is hard.

The next idea is: How does group reproducibility depend on reliability and (detectable) connectivity, as well as sample size? To explore this, we need the script **bootRelSampleSize.m**, which is referenced in line 250. The script uses many of the routines discussed now, but they are integrated in a special way. It essentially consists of an outer loop that captures all sample sizes between 10 and 50. The outer loop starts in line 34 and ends in line 145. Then there’s an inner loop that repeatedly draws randomized samples for a given sample size. All connectivity and reliability statistics are then collected in 3D matrices. The output of this function is quite substantial, so you get the best impression by taking a look at Figure 5 of the main paper and Figures S5 and S6 of the supporting materials.

Of course, we can't forget about Figure 6 in the main paper! So far, we've adopted the coordinates for the connectome from a meta-analysis of a working memory study. This makes perfect sense, as certain metrics—especially the height of the correlations detectable between two pathways—are quite sensitive to the area you choose. When the connectomes are determined completely at random, the correlations between the areas drop significantly. However, if you're more interested in group reproducibility in terms of Dice overlap, it becomes relatively less important. The number of pathways you can uncover through conjunction is significantly influenced by whether you use normal connectivity or detectable connectivity to test for significance. I've provided the MATLAB script **montec\_func\_connect\_rel.m**, but I don't want to delve too deeply into that, as running the script requires the full FreeSurfer output, which is extremely large. Plus, we're dealing with human data that is legally protected. I hope you won't hold it against me, but even a rather open-minded Dutchie knows where to draw the line!

Yours ever,

Jan Willem Koten junior