To the Independent Climate Change Email Review Committee,

I am making this submission with regard to the discussions of software in the hacked CRU emails, and to the small amounts of software source code also published in the same incident. I am a software expert with long industrial experience. I also have some expertise in climate science software. I hope that my submission may clarify some of the issues raised.

1. Background

I have been a professional software engineer since 1990. In 1997, I founded a software consultancy firm in Cambridge called Ravenbrook Limited, which I continue to run today. We work in various industry sectors, including desktop data analysis, games development, memory management, software tool integration, and robotic control. I have a first-class degree in mathematics, and a post-graduate diploma in computer science, both from the University of Cambridge. I have a life-long interest in science and in computers, and I have known many working scientists.

2. The Clear Climate Code Project

In 2008 I started Clear Climate Code, a volunteer pro-bono project, not suggested or commissioned by anyone else, because I saw that the clarity of software published by climate scientists was disrupting the public debate about climate change. Such disruption has greatly worsened in the wake of the CRU email hack.

The goals of the Clear Climate Code project are:

1. To produce clear climate science software;
2. To encourage the production of clear climate science software;
3. To increase public confidence in climate science results.

We aim to increase public confidence in results by publishing and clarifying the software which produces those results. That process will certainly uncover and correct errors. These corrections are essential to the goal. We don't want the public to trust incorrect results; any incorrect results should be visibly corrected, so that the public may trust them.

Several colleagues and other people around the world have joined and helped me in our efforts. We publish all of our work at http://clearclimatecode.org/ and make presentations about it at conferences.

3. GISTEMP

The main focus of Clear Climate Code so far has been on GISTEMP. GISTEMP is a gridded dataset of global historical surface temperature anomalies produced by NASA's Goddard Institute for Space

Studies (GISS). It is substantially similar to the HadCRUT3 dataset produced jointly by the Met Office Hadley Centre and by CRU. GISTEMP produces a "Global Land-Ocean Temperature Index", a chart that shows how global temperatures have changed since 1880. see http://data.giss.nasa.gov/gistemp/graphs/ . This chart is the analogue of HadCRUT3's "Global average temperature" chart, see http://hadobs.metoffice.com/hadcrut3/diagnostics/global/nh+sh/ .

The source code that produces the GISTEMP analysis is published, see http://data.giss.nasa.gov/gistemp/sources/. All the data used in the GISTEMP analysis is also published. Clear Climate Code have been reimplementing this analysis, with an emphasis on clarity. The intention is that anyone interested in the subject, and capable of understanding a program, should be able to download our software and easily follow it. I have closely examined every part of the original GISTEMP source code, and collectively we have written a new version that produces exactly the same results as GISTEMP. We continue to refine our version, to improve its clarity. Dr Reto Ruedy, a mathematician at NASA GISS with responsibility for GISTEMP, has said recently that they would like in future to use our version.

As part of this process, I have read several of the scientific papers that describe the GISTEMP analysis, including:

J. Hansen, R. Ruedy, J. Glascoe, Mki. Sato, J. Geophys. Res. 104, 30997-31022, doi:10.1029/1999JD900835 (1999).

J.E. Hansen, S. Lebedeff, J. Geophys. Res. 92, 13345-13372 (1987).

In making our new implementation we have found a few minor problems or "bugs" in the GISTEMP software. I have reported these to Dr Ruedy at NASA GISS who has in all cases fixed them and thanked us for reporting them. The problems we have found either do not affect the published results, or only affect them by tiny amounts, far less than the uncertainty. For instance, one month's reading might change by a hundredth of a degree.

As our work continues, we have also investigated some questions raised by critics of GISTEMP, for instance whether the warming signal of the results could be due to rounding numbers in the GISTEMP system, or due to the urban heat-island adjustment, or due to changes in weather station numbers in the 1990s. We have been able to show very easily that these factors are not significant - the effect of each factor is tiny, and the effects are often in the opposite direction to that suggested by critics.

Regarding GISTEMP I conclude:

* the source code could be clearer;

* the source code is free of any major errors;

* it performs as described in the scientific literature;

* it produces a result which is substantially the same as HadCRUT3,

and other similar datasets;

* the result is also substantially the same when the factors commonly mentioned by critics are removed;

* I have no reason to doubt the output of the GISTEMP analysis.

By "substantially the same" I mean that the conclusions regarding warming in the latter part of the 20th century that one can draw from the GISTEMP analysis are the same as the conclusions one can draw from the HadCRUT3 analysis.

Of course, none of our work supporting these conclusions would have been possible without the full publication of the original GISTEMP software and the data which it uses.

4. Science Software in General

It is obvious to me that the original GISTEMP software is not engineered to a high quality. However, this is exactly what I expected when I started the Clear Climate Code project. I have seen other bits and pieces of science software over the decades - for instance when friends have asked for help with programming problems - and most of it is very much of a piece with GISTEMP, and with the software published in the CRU emails.

I have known working scientists for decades, and although they are mostly very expert in their domain, very few of them are software experts. Writing software well is a complex and demanding profession, with a very large body of knowledge and art, in which scientists generally receive little or no training. Yet almost every scientist must write some software, and many of them spend a considerable proportion of their time doing so. They must write software for data collection, analysis, processing, and modelling, for drawing charts and other visualisations, for controlling lab equipment. This is an integral part of the work of many scientists today.

If scientists are very lucky, their establishment will have staff to help with writing some of this software. However, such staff are often equally untrained: many are graduate students in the same scientific discipline, others may be long-standing lab staff who have gradually been assigned software tasks over many years with no opportunity for training. Many junior scientists and staff members who show particular flair for software development quickly leave, when they realise that salaries and budgets in the software industry often greatly exceed those in science labs.

As a result, science software development lacks many features which are commonplace in the software industry. Detailed written specifications are rare - as is all documentation, even comments in the code. Test suites are not developed. The use of tools such as automated testing, source code management, configuration management, and defect trackers is very uncommon. Code style is often rudimentary, and many powerful features of computer languages and

systems are not used.

For this reason, many software professionals encountering science software for the first time may be horrified. How, they ask, can we rely on this crude software, developed in primitive conditions - by amateurs, working with such poor tools and such poor understanding of the field? This is a common reaction to GISTEMP, and is exactly the reaction which many critics have had, some very publicly, to the software published with the CRU emails.

Such critics do have a point. Science software should be better than it is. Scientists should be provided with more training, and more support. But consider the uses to which science software is put. Most software written by scientists:

* consists of tiny programs;
* which will only ever be run a small number of times;
* over the course of a few weeks as it is being developed;
* by the scientist who wrote it;
* on data gathered by that scientist's team;
* concerning a scientific field in which that scientist is expert;
* to perform data processing on which that scientist is expert;

and will be discarded, never to be used again, as soon as the paper containing the results is accepted for publication.

In comparison, commercial software has a huge range of complex requirements, which are completely irrelevant to most science software: usability, robustness, flexibility, maintainability, size, speed, compatibility, portability, consistency, documentation, and so on. Much of the software industry, and the art of computer programming, is motivated by and devoted to meeting these requirements (which, even so, it often utterly fails to do). The tools and skills which I have spent my professional life mastering are of little consequence to the working scientist, who is simply trying to get the charts ready for his or her next publication.

However, some of this small throw-away software written by scientists does survive beyond its original purpose. A sequel paper is proposed, and the author digs up the code for the earlier analysis, to amend it. Then a colleague expresses an interest, and develops a new feature for a paper of his or her own. More years pass, more data, more hypotheses to test, more papers are written, more analysis is added. Each step is taken in the same style as the first: write a small module to perform some additional analysis, run by hand, check the results, draw a couple of charts, submit the paper for publication. The result grows in an incoherent, undesigned fashion. Very occasionally science based on such software produces results which are of great public concern, and so the over-grown misshapen software comes to light.

GISTEMP is an example of this, and I don't doubt that other examples exist at CRU and many other places. The nucleus of GISTEMP was developed in the 1980s, I believe for the paper Hansen & Lebedeff 1987 (cited above) - certainly the code appears to be of that vintage and

the paper describes the behaviour of parts of the code very well. Other parts of the algorithm were certainly not developed until the Hansen et al 1999 paper. Some parts of GISTEMP are still under development today: in January 2010 there was a change to the way in which urban weather stations are identified.

I gather that some of GISTEMP was developed by a member of staff who is no longer at GISS. This is a very common problem: that no one person knows how a whole system operates. People leave, change jobs or careers, die, or simply forget. This appears to be the background to the infamous HARRY READ ME file published with the CRU emails: it appears to be a very detailed and frank record of a painstaking effort to work out and document the operation of a complex piece of software which has been developed over several years by several different people.

Despite all our much-vaunted expertise and skill, this problem is also commonplace in the software industry. I myself have written broadly similar documents to HARRY READ ME, for very similar reasons - although rarely at such great length. Sometimes software is simply abandoned, because the cost of figuring out its operation can be far greater than the benefit.

I should note before moving on that some software written by scientists does have to meet some of the same requirements as commercial software. Some groups of scientists, such as the Met Office, and some entire fields, such as computational physics, develop large, complex, long-lived, robust, immensely sophisticated pieces of software. Many companies and other enterprises exist to provide reusable and flexible software tools for scientists. Typically, such groups, fields, and enterprises do follow industry practice in software development.

5. Open Science

Plainly, the Clear Climate Code project would be very much harder, and our work on GISTEMP would have been completely impossible, if the GISTEMP code had not been published by NASA GISS, or if the data used by that code were not freely available from NASA and NOAA.

I believe that our work has advanced climate science in a small way, by identifying and correcting some minor bugs in GISTEMP. I believe that it has contributed, and will continue to contribute significantly, to the understanding of this aspect of science by interested members of the public. For these reasons, the publication of GISTEMP has surely been a good thing.

However, normal practice in most sciences is not to publish the software which performs the analysis or produces the charts for a paper. The software may not even be available to the peer reviewers of the paper. The science in the paper is supposed to stand on its own, to speak for itself. The data processing performed by the code should be described in the paper anyway.

This principle may have made sense fifty years ago, when the software would have mostly been straight-forward arithmetic (calculating an average, or a linear regression), and when publication of the software - or the data - might have meant distributing stacks of print-out with every journal. But in recent decades the software has become larger and more complex - its function less obvious even to peers, with too many fine details to cover in the paper - and the problem of publication and distribution has been solved entirely by the internet. In short, the need for publication has increased, and the reasons not to publish have dwindled away.

In some fields provision of all supporting code and data is the norm. Some journals make it a prerequisite for publication - or even review - of a paper, and have online repositories of code and data. Some journals even have free-access repositories, while still charging for access to the papers. This is a very positive trend, which could become the norm very rapidly if suitable action was taken by funding bodies. For instance, the UK Research Councils could make it a requirement for any research grant.

It remains the case that true scientific replication, a cornerstone of the scientific method, means gathering one's own data and writing one's own code. Running published code on published data will almost certainly reproduce published results, and for almost all science this may not be worthwhile. But when a publication is unclear, or called into question, examination of the fine detail of the code can provide further information which is not available in any other way.

In particular when science results are used to inform important public policy debates, exactly such reproduction can be immensely important.

6. Conclusions

The results of the HadCRUT3 analysis are compatible with those of the GISTEMP analysis, with which I am very familiar.

We have precisely reproduced the results of GISTEMP with our own work, which is freely available to the public.

The CRU emails, and the other documents published with them, in their discussion of software, do not show any evidence of any manipulation or suppression of data. They appear to be entirely in keeping with normal software practice in science.

CRU have followed common scientific practice of publishing results, without publishing the software which computes those results. However, this practice can inhibit effective communication among scientists and can severely impede public understanding of science. In the case of HadCRUT3 it has caused a great deal of public mistrust of scientists and disbelief of science results, with possibly serious consequences.

If you have any questions regarding this submission, or our work more broadly, I am of course available to give further evidence, in person

if necessary.

Nicholas Barnes, Director, Ravenbrook Limited