




SigCorr: A Python package for studies of trials factors

V. Ananiev ^{1*} and A. L. Read ^{1*}¶

¹ Department of Physics, University of Oslo, Boks 1072 Blindern, Oslo, NO-0316, Norway ¶
Corresponding author * These authors contributed equally.

DOI: [10.21105/joss.04989](https://doi.org/10.21105/joss.04989)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: Øystein Sørensen  

Reviewers:

- @BSGalvan
- @peifengjing
- @gvieralopez

Submitted: 09 November 2022

Published: 30 June 2023

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).

Summary

The Trials factor (TF) is a generalization of the Bonferroni correction for the significance of repeated experiments. In a likelihood ratio scan that searches for the most significant peak of signal amplitude as a function of some parameter, e.g. the mass of a hypothetical particle, repeated experiments emerge effectively due to the localized nature of the hypothetical signal. The trials factor in such a search is the ratio between the probability of observing such an excess anywhere in the scanned search region to the probability of observing the excess at the point of maximum observed significance. For a more detailed introduction of the TF, we refer the reader to the work of Gross and Vitells ([Gross & Vitells, 2010](#)).

There are several ways to estimate the trials factor or an upper bound for it. Most of the approaches require a number of samples from the background distribution (Monte Carlo (MC) toys) to be generated and fitted with a statistical model. This procedure is separated in the upper block of [Figure 1](#), mainly because it is the most time-consuming part of the analysis.

The lower block in [Figure 1](#) shows various ways to estimate the trials factor from the fitted MC toys. When expressed as a pipeline, each way can be assembled from some simple building blocks, the implementation of which is at the core of SigCorr.

The communication between the upper (fitting) and the lower (analysis) components is conducted via HDF5 files with a well defined structure, which can be referred to in the documentation of SigCorr ([Ananiev & Read, 2023a](#)). Such a weak coupling allows to easily replace parts of the pipeline with more efficient tools if needed.

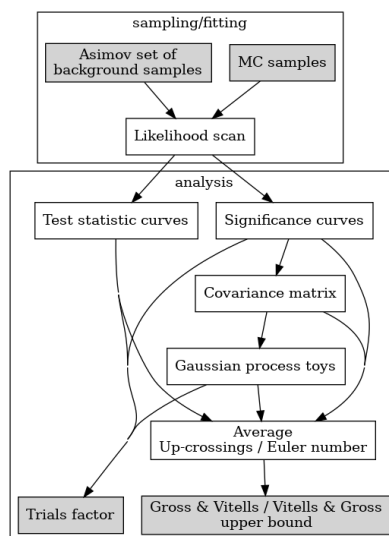


Figure 1: Various ways to estimate the trials factor.

SigCorr is a framework that implements a wide range of tools and was developed to simplify the process of construction of the pipelines shown above, and, therefore, to make studies of the global significance or a trials factor a straightforward exercise. It is worth mentioning, though, that the figure above is not exhaustive. We believe there are more ways to approach the trials factor, and some of them may also be possible to implement with the tools provided by SigCorr!

The project has a hybrid structure:

- From the perspective of fitting the Monte Carlo toys, it is a framework. Users will have to implement their own statistical models following the guides and examples in the documentation.
- From the perspective of analysis of the fitted toys, however, SigCorr is a Swiss Army knife that assembles into one tool the knowledge from the frequently cited HEP papers that studied statistical hypothesis testing and the trials factor (Cowan et al., 2011; Gross & Vitells, 2010; Vitells & Gross, 2011).

Statement of need

In high-energy physics it is a recurring challenge to efficiently and precisely (enough) calculate the global significance of, e.g., a potential new resonance. The Gross and Vitells trials factor approximation (Gross & Vitells, 2010) and (Vitells & Gross, 2011) is based on the average “up-crossings” of the significance in the search region, or generally on the average Euler characteristic of the set of significance measurements that exceed the threshold of the local significance. It has revolutionized the trials factor estimation for significances above 3 standard deviations, but the challenges of actually calculating the average up-crossings and the validity of the approximation for smaller significances remain.

In Ananiev & Read (2023b) a new method was proposed. It models the significance in the search region as a Gaussian process (GP). The method was developed to overcome the limitations of the Gross and Vitells approach via replacing expensive MC fits with lightweight GP toys.

Up-crossings, Euler characteristic and Gaussian processes are commonly relied on in this field of research. When studied together, they have many useful properties for the estimation of

trials factors. SigCorr is the first project that assembles all of them into one Python package, that also includes the tools for parallel analysis of the MC toys and GP toys in a unified fashion. Together with the fitting framework it allows the path from the statistical model definition to a TF estimate to be travelled.

SigCorr is a framework developed with a goal to cover a wide range of use cases for TF estimates. There are, however, popular and well-maintained packages which cover some aspects of this process. For example, RooFit ([Verkerke & Kirkby, 2006](#)) and PyHF ([Heinrich et al., n.d.](#)), which are widely used in particle physics, may significantly replace or augment the part of SigCorr that is responsible for optimizing the likelihoods of the statistical models. The pyBumpHunter package ([Vaslin & Donini, 2023](#)) may help with brute force estimates of the TF.

Although SigCorr covers just simplified versions of all the above, it also implements validated approximations and asymptotics for fast computations of the TF and its upper bound based on classical papers in the field. The possibility of doing this in a modular but consistent fashion makes SigCorr unique in the field, and will allow users to try different methods ([Figure 1](#)), cross-validate and choose or construct the most suitable approach to their analysis.

Examples of use

Average Euler characteristic propagation

For the Gaussian process field, for any chosen threshold on the field values, one can define a set of points above this threshold. The Euler characteristic of this set gives an estimation of the number of peaks above the threshold, which is a very important characteristic of the Gaussian process for the Trials factor studies.

A Gaussian process has a property that the average Euler characteristic at any threshold is a simple function of a few reference Euler characteristic values and the threshold ([Vitells & Gross, 2011](#)). The number of reference values required depends on the dimensionality of the Gaussian field.

Here is how to estimate the average Euler characteristic for arbitrary thresholds with SigCorr, knowing 2 reference values for a 2-dimensional Gaussian field:

```
import numpy as np
from sigcorr.tools.stats.gp.euler_number import GPEulerNumberPropagator

# 2D Gaussian process requires 2 reference values
ref_thresholds = np.array([1, 2])
ref_euler_nums = np.array([3.2, 1.2])
target_thresholds = np.array([0.5, 1.5, 2.5])
euler_number_propagator = GPEulerNumberPropagator(ref_thresholds,
                                                    ref_euler_nums)
target_euler_numbers = euler_number_propagator.calc(target_thresholds)
# array([3.10791656, 2.29280228, 0.46934814])
```

Batched statistics

When the number of samples is known in advance but not all samples are available at the moment (e.g. they don't fit into RAM), it is still possible to estimate mean, variance, covariance and statistical errors by processing the samples in batches.

```
import numpy as np
from sigcorr.tools.stats.batch_stats import BatchStats2

n_samples = 3
one_sample_shape = 2
samples = np.array([[1, 2], [2, 3], [3., 4.]])
```

```
bs = BatchStats2(n_samples, one_sample_shape)
bs.push(samples[:1]) # push first batch
bs.push(samples[1:]) # push second batch

# sample mean
mean = bs.get_mean()
# array([2., 3.])

# sample variance
variance = bs.get_var()
# array([0.66666667, 0.66666667])

# covariance matrix between observables
covariance = bs.get_cov()
# array([[0.66666667, 0.66666667],
#        [0.66666667, 0.66666667]])

# variance of the covariance matrix between observables,
# can be used to estimate statistical errors
# on the sample covariance estimate
covariance_variance = bs.get_cov_var()
# array([[0.22222222, 0.22222222],
#        [0.22222222, 0.22222222]])
```

Statistical analysis of GP samples (parallel)

A set of standard normal random variables when squared follow a chi-squared distribution with 1 degree of freedom. One can see the analogy with a test statistic curve emerging from a likelihood scan. If we ask the question, what is the fraction of curves that exceed some threshold, this would resemble the procedure used to estimate the trials factor via brute force. Below we estimate the fraction of test statistic curves that exceed the threshold 1.2.

```
import numpy as np
from sigcorr.mapreduce.gp import gp_batch_mapreduce
from sigcorr.tools.utils import get_last_from_iter
from sigcorr.mapreduce.map_reducers import ChainCalc
from sigcorr.mapreduce.map_reducers import MathCalc
from sigcorr.mapreduce.map_reducers import OverflowsCalc
from sigcorr.mapreduce.map_reducers import BatchStats1Reduce

cov = np.eye(3) + 0.1
result_iterator = gp_batch_mapreduce(
    cov, # covariance matrix of the GP
    100, # number of GP samples
    10, # batch size
    (3, ), # sample shape
    ChainCalc([MathCalc(np.square), OverflowsCalc(1.2)]), # apply to every batch
    BatchStats1Reduce()) # aggregate
resulting_bs, num_processed = get_last_from_iter(result_iterator)
resulting_bs.get_mean()
```

Here, ChainCalc, MathCalc, OverflowsCalc are building blocks of the pipeline used to square the GP sample and then to set the value 0 or 1 per curve in the batch depending on whether the curve exceeds the threshold 1.2. There are other building blocks that, for example, help to estimate the significance from the maximum likelihood values (SigsCalc), or to compute the Euler characteristic of the batch of samples (EulerNumberCalc), etc.

Acknowledgements

We would like to thank Ofer Vitells for several very helpful discussions, in particular about the details of the statistical model in the Gross and Vitells paper, the implementation of which became a part of the SigCorr package to serve as one of the examples of its use. We would also like to acknowledge the support of the ATLAS Collaboration. This research was supported by the European Union Framework Programme for Research and Innovation Horizon 2020 (2014–2021) under the Marie Skłodowska-Curie Grant Agreement No.765710.

References

- Ananiev, V., & Read, A. L. (2023a). *Documentation and examples for SigCorr*. <https://sigcorr.gitlab.io/sigcorr/latest/>
- Ananiev, V., & Read, A. L. (2023b). Gaussian process-based calculation of look-elsewhere trials factor. *Journal of Instrumentation*, 18(05), P05041. <https://doi.org/10.1088/1748-0221/18/05/P05041>
- Cowan, G., Cranmer, K., Gross, E., & Vitells, O. (2011). Asymptotic formulae for likelihood-based tests of new physics. *The European Physical Journal C*, 71(2). <https://doi.org/10.1140/epjc/s10052-011-1554-0>
- Gross, E., & Vitells, O. (2010). Trial factors for the look elsewhere effect in high energy physics. *The European Physical Journal C*, 70(1-2), 525–530. <https://doi.org/10.1140/epjc/s10052-010-1470-8>
- Heinrich, L., Feickert, M., & Stark, G. (n.d.). *Pyhf*. <https://doi.org/10.5281/zenodo.1169739>
- Vaslin, L., & Donini, J. (2023). pyBumpHunter. In *GitHub repository*. <https://github.com/scikit-hep/pyBumpHunter>; GitHub.
- Verkerke, W., & Kirkby, D. (2006, May). The RooFit Toolkit for data modeling. *Statistical Problems in Particle Physics, Astrophysics and Cosmology*. https://doi.org/10.1142/9781860948985_0039
- Vitells, O., & Gross, E. (2011). Estimating the significance of a signal in a multi-dimensional search. *Astroparticle Physics*, 35(5), 230–234. <https://doi.org/10.1016/j.astropartphys.2011.08.005>