Computing trial factors for Astronomy Point-Source Searches

F. Vazquez de Sola

`2026-01-26 14:36:33`

1 Introduction

This document describes an approach to compute trial factors for point-source searches, when using candidate lists partially derived from the same data used to look for those sources.

1.1 Basics of the analysis

In its most basic form, as far as the trial-factor topic is concerned, the current point-source search analysis approach is:

- For each candidate, we have 2D-histograms of the reconstructed energy vs the reconstructed angle from the candidate, both for background and signal;
- Pseudo-experiments (PEs) are generated based on these histograms, assuming no signal strength;
- Each PE is fit by optimizing the negative log-likelihood ratio (NLLR) between a free signal strength fit and one where the signal strength is fixed to zero[1];
- For each candidate, we thus construct a distribution of background-only NLLRs;
- The value obtained for each candidate in the *real* dataset is compared to this distribution to obtain its "p-value", the probability to obtain a NLLR value as high or higher in the null-hypothesis (i.e. no signal) scenario.

We refer to each candidate's p-value as its "local" p-value (referred to elsewhere as its "pre-trial" value), which would be the "true" p-value of the analysis if it was the only candidate we were looking at.

1.2 Naive trial factor

Since we are testing multiple candidates at once, the observed p-value of the most significant candidate will be artificially higher than if we were looking at it in isolation - this is called the "Look-Elsewhere Effect" in Physics ("multiple comparisons" or "multiple testing" problem in statistics). In the most simple scenario in which all candidates are independent from each other, and selected before looking at the data, you can convert the local p-value of the most significant source into a "true"/global one with:

$$\begin{aligned} 1 - p_{global}(p_{local}) &= \prod_{i=1}^{N} P_i\left(p > p_{local}\right) = \left(1 - p_{local}\right)^N \\ \textbf{if } Np_{local} \ll 1 \;\Rightarrow\; & p_{global} \simeq Np_{local} \end{aligned} \tag{1}$$

where N is the number of candidates tested, and $P_i\left(p > p_{local}\right) = 1 - p_{local}$ is the probability that the p-value for candidate i is larger (i.e. "less extreme") than p_{local}. The final approximation is why we call $T = \frac{p_{global}}{p_{local}}$ the "trial factor", the correction factor to apply onto p_{local} to obtain the global p-value (referred to elsewhere as its "post-trial" value)

[1]The histograms used for generation of PEs and fitting/"evaluating" them can be different, but this is not important for this discussion.

1.3 Trial factor in the ANTARCA analysis

My understanding of how the ANTARCA candidate list was obtained, after discussions with Vladimir Kulikovskiy, Matthieu Lamoureux and Paco Salesa Greus, is as follows:

For the ANTARCA analysis, the candidate list was reused from the first ARCA point source analysis. However, this list was partially derived from ANTARES data, which is now being folded back into with the ARCA data. Concretely, we use (per the ANTARCA internal note and the original ANTARES paper, to be double-checked):

- The most significant candidate out of a list of 1453 based on ANTARES data up to 2017/12/31;
- The candidates out of a list of 167 where the fit of the ANTARES data up to 2017/12/31 showed at least one signal-like event - this turned out to be 105.

This approach to generating our candidate list will artificially drag the local p-values down, especially since the ANTARES data up to 2017/12/31 represents a significant proportion of the total ANTARCA data. Clearly, taking $T = 1 + 105$ will underestimate the trial factor. Conversely, $T = 1453 + 167$ will overestimate it.

The generic approach to compute the true trial factor is to generate PEs as follows:

- Generate events for all candidates, assuming no signal, and keeping track of which events were generated during ANTARES up to 2017/12/31;
- Apply the same filtering based on the generated ANTARES PE information (most significant candidate from the 1453 catalog, candidates with at least one signal-like event from the 167 catalog);
- Compute p_{local} for the remaining candidates, and keep the most significant value.

By repeating this procedure, we obtain a distribution on p_{local}, which can be used to convert our observed p_{local} into p_{global}.

This approach works, but is not very flexible. If we modify the number of candidates considered, a new set of PEs must be generated. A semi-analytical approach is presented in the following section.

2 Semi-analytical approach: Theory

We will fist consider the two catalogs separately, since they are filtered in different ways. In both cases, we will need access to the distribution of the NLLR over the whole dataset based on some constraints on the NLLR during the "selection period" (here the ANTARES dataset up to 2017/12/31), but not in the same way.

For the following, we define the following conventions:

- p is the local p-value of the most significant candidate, $p_{global}(p)$ is its corresponding p-value after accounting for the trial factor;
- $\mathcal{L}$ the NLLR for the complete dataset for one candidate;
- $f(\mathcal{L})$ the probability density function of $\mathcal{L}$ for one candidate, and $F(\mathcal{L}) = \int_0^{\mathcal{L}} f(\mathcal{L}')\, d\mathcal{L}'$ its cumulative density function;
- $\mathcal{L}(p)$ the value of the NLLR such that $F(\mathcal{L}(p)) = 1 - p$, i.e. the value of the NLLR that would correspond to that given local p-value;
- subscript 1 to denote functions or variables including only the selection ("first") period, e.g. $\mathcal{L}_1$ is the NLLR for one candidate including only the selection period;
- subscript s to denote the functions relative to candidate s, with a comma to distinguish the subscript related to the selection period, e.g. $\mathcal{L}_s$, $F_{s,1}$, etc.

2.1 Filter: Candidates with signal over threshold

In the simpler scenario where all candidates are equivalent, the formula for p_{global} is :

$$1 - p_{global}(p) = \left(P_1(\neg C) \,+\, P_1(C)\, F(\mathcal{L}(p) \,|\, C) \right)^N \tag{2}$$

where p is the best local p-value observed, C is the constraint during the selection period (for ANTARCA, requiring the number of signal-like events larger than 1, but the formula is agnostic to the form of the constraint), and $P_1(C)$ (or $P_1(\neg C)$) is the probability that this constraint is (or not) verified during the first period. $F(\mathcal{L} \,|\, C)$ is the conditional cumulative distribution of the NLLR over the whole dataset, given that the constraint is verified during the selection period.

If the candidates are not equivalent, then we update the equation to:

$$1 - p_{global}(p) = \prod_{s=1}^{N} \left(P_{s,1}(\neg C_s) \,+\, P_{s,1}(C_s)\, F_s(\mathcal{L}_s(p) \,|\, C_s) \right) \tag{3}$$

where we add the subscript s for elements that might depend on the candidate (including potentially the constraint itself, although that is not the case for the ANTARCA analysis). Notably, we see that the formula can be separated per candidate: we just need to compute $P_{s,1}(C_s)$ (a number) and $F_s(\mathcal{L}_s(p) \,|\, C_s)$ (a 1D distribution/histogram) for each, which we can then plug into the formula to obtain p_{global} from the observed p_{local}. This can easily be done by simulating pseudo-experiments for each candidate separately. Note however that both elements depend on the constraint, so the PEs need to be repeated if we change them[2].

In practice, you likely cannot save $F_s(\mathcal{L}_s(p) \,|\, C_s)$ directly, since $\mathcal{L}_s(p)$ requires knowing $F_s(\mathcal{L})$ first, which you do not necessarily have before running the PEs. Instead, you can save $f_s(\mathcal{L})$ and $f_s(\mathcal{L} \,|\, C_s)$ as you run the PEs, e.g. by creating two 1D histograms of $\mathcal{L}$ and filling them as you go, and then use them to construct $F_s(\mathcal{L}_s(p) \,|\, C_s)$ after the fact. Some interpolations will be necessary, as the bin edges for $\mathcal{L}_s$ are unlikely to have matching values of p_{local} across different candidates.

2.2 Filter: Best of N candidates

In the simpler scenario where all candidates are equivalent, the formula for p_{global} is:

$$1 - p_{global}(p) = \int_0^{+\infty} N \left(F_1(\mathcal{L}'_1)\right)^{N-1} f_1(\mathcal{L}'_1) \,\cdot\, F(\mathcal{L}(p) \,|\, \mathcal{L}_1 = \mathcal{L}'_1)\, d\mathcal{L}'_1 \tag{4}$$

The formula comes from integrating over $\mathcal{L}'_1$, the best NLLR over all candidates during the selection period, where:

- the first term, $N \left(F_1(\mathcal{L}'_1)\right)^{N-1} f_1(\mathcal{L}'_1)$, corresponds to the probability that one arbitrary candidate has an NLLR during the selection period of exactly $\mathcal{L}'_1$, while the rest have a smaller NLLR,
- the second term, $F(\mathcal{L} \,|\, \mathcal{L}_1 = \mathcal{L}'_1)$, is the conditional cumulative distribution of the NLLR over the whole dataset, given that the NLLR over the selection period is equal to some value $\mathcal{L}'_1$.

If the candidates are not equivalent, then we need to do the comparison between candidates in the selection period in terms of their local p-value at the time, instead of their NLLR. The equation becomes:

$$1 - p_{global}(p) = \int_0^{1} (1 - p_1)^{N-1} \sum_{s=1}^{N} f_{s,1}(\mathcal{L}_{s,1}(p_1)) \,\cdot\, F_s\left(\mathcal{L}_s(p) \,|\, \mathcal{L}_{s,1} = \mathcal{L}_{s,1}(p_1)\right) dp_1 \tag{5}$$

where we are integrating over p_1, the best local p-value over all candidates during the selection period. Just as for the previous filtering approach, we see that we can also separate the components per candidate: we need $f_{s,1}(\mathcal{L}_{s,1}(p_1))$ (a 1D distribution/histogram) and $F_s\left(\mathcal{L}_s(p) \,|\, \mathcal{L}_{s,1} = \mathcal{L}_{s,1}(p_1)\right)$ (a 2D distribution/histogram) for each.

[2] If you know ahead of time which constraints you are considering, you can probably compute $P_{s,1}(C_s^{(k)})$ and $F_s(\mathcal{L}_s(p) \,|\, C_s^{(k)})$ for all of them at once with just one set of PEs.

In practice, since you likely do not have access to $\mathcal{L}_{s,1}(p_1)$ or $\mathcal{L}_s(p)$ before running the PEs, you can save instead $f_s(\mathcal{L}|\mathcal{L}_1)$, e.g. by filling a 2D histogram of $\mathcal{L}$ vs $\mathcal{L}_1$ as you go. Additionally, the formula changes when using discrete probability density functions, as is the case when we derive them from histograms instead of analytical functions. It becomes:

$$\begin{aligned}
&R_{s_{\mathcal{L}_1}}(i) = \sum_{i'=0}^{i} H_{s,1}[i'] \\
&R_{s_{\mathcal{L}|\mathcal{L}_1}}(i,j) = \sum_{j'=0}^{j} H_s[i,j'] \\
&g_s(\mathcal{L}^{(j)}) = R_{s_{\mathcal{L}_1}}(0)^N \cdot R_{s_{\mathcal{L}|\mathcal{L}_1}}(0,j) + \sum_{i=1}^{n_{\mathcal{L}_1}} (R_{s_{\mathcal{L}_1}}(i)^N - R_{s_{\mathcal{L}_1}}(i-1)^N) \cdot R_{s_{\mathcal{L}|\mathcal{L}_1}}(i,j) \\
&\Rightarrow 1 - p_{global}(p) = \frac{1}{N}\sum_{s=1}^{N} g_s(\mathcal{L}_s(p))
\end{aligned} \tag{6}$$

where $H_s[i,j]$ is the 2D counting histogram of $\mathcal{L}$ vs $\mathcal{L}_1$ for candidate s, with $H_{s,1}[i]$ and $H_{s,T}[j]$ its projection onto the $\mathcal{L}_1$ and $\mathcal{L}$ axes respectively. $\mathcal{L}^{(j)}$ is the $\mathcal{L}$ value of the upper edge of bin j, $p_s^{(j)} = 1 - \sum_{j'=0}^{j} H_{s,T}[j']$ is its corresponding local p-value for candidate s, $n_{\mathcal{L}_1}$ is the number of bins for $\mathcal{L}_1$, and we follow ROOT's bin numbering convention (0 is underflow, $1 : n$ are the regular bins, $n+1$ is the overflow). Note that both $g_s(\mathcal{L})$ and $\mathcal{L}_s(p)$ need to be interpolated for values in between bin edges of $\mathcal{L}$.

3 Semi-analytical approach: Implementation

3.1 Available code

Two C++ scripts were added to JPP to implement this approach, taking advantage of Maarten's JPseudoExperiment code for fast PE generation. They can be found in examples/JAstronomy.

The first one, JPseudoExperimentCondNLLR.cc, computes the "conditional likelihood" 2D histogram of $\mathcal{L}$ vs $\mathcal{L}_1$, both for all PEs (h2d_nllr) and only for those that pass the constraint (h2d_nllr_constr , requiring selection period fit gives $n_{signal} > 1$), for one candidate. The histograms are not rescaled after filling, so the ratio of the total count of the latter over the former gives the probability of passing the constraint.

The main input to this script is a list of signal/background histogram pairs (or quartets, if the generation and evaluation histograms are different), where we specify if they are part of the first/"filtering" period or of the second/"blinded" period. Each histogram might be rescaled, to test the effect of increasing exposure, or changing the ratio of signal to background. These will be used to compute the conditional likelihood histograms. Concretely, the arguments for JPseudoExperimentCondNLLR.cc are:

- E: inputs for 1st data period: signal and background histograms for generation and evaluation, exposure boost, and signal and background nuisances, provided as:

  "< file >:<hgen_s name> <file>:<hgen_b name> <file>:<heval_s name> <file>:<heval_b name> <boost> <type_s> (values) <type_b> (values)"

  Can be called repeatedly to add multiple datasets to this period;

- F: inputs for 2nd data period: signal and background histograms for generation and evaluation, exposure boost, and signal and background nuisances, provided as:

  "< file >:<hgen_s name> <file>:<hgen_b name> <file>:<heval_s name> <file>:<heval_b name> <boost> <type_s> (values) <type_b> (values)"

  Can be called repeatedly to add multiple datasets to this period;

- o: output file with histograms (CondNLLR.root by default);

- s: signal strength;
- b: background strength (1.0 by default);
- M: lookup table for CDFs (0/unused by default);
- R: signal-to-noise ratio threshold (0.0 by default)
- n: number of PEs;
- x: x-axis likelihood histogram ("200 0.0 +20.0" by default);
- S: seed (0 by default);
- d: debug (1 by default);

The second C++ script, JTrialFactorsFromCondNLLR.cc, takes a list of output files from this first script (one per candidate), and combines them to generate CDF histograms of the best p_{local} assuming either filter, and the resulting trial factor vs p_{local} plot. An option is given to duplicate the candidate outputs, as a way to estimate the effect of increasing the candidate list size. Additionally, the script also saves the naive trial factor curve, $T(p) = \frac{1-(1-p)^N}{p}$ assuming all of the original candidates were kept; this serves to estimate the improvement on the trial factor from the filtering procedure. Concretely, the arguments for JTrialFactorsFromCondNLLR.cc are:

- o: output file with histograms;
- i: file with list of H2D NLLR files to use;
- n: number of times to duplicate candidates (1.0 by default, i.e. no duplication);
- x: x-axis of -log(p) histogram ("80 0. 8." by default, but the max value should be set so that $\mathbf{min}(p) \gtrsim 1/n_{PEs}$);
- d: debug (1 by default);

One important caveat of this second script is that no particular thought was given to its numerical accuracy, i.e. limits due to double precision. The calculations might involve adding numbers very close to zero ("best-of" filter) or multiplying numbers very close to one ("constraint" filter). For the results described in the following sections, this did not appear to be an issue - as a sanity-check, we confirmed that, at low p-values, the computed trial factors do converge towards the total number of candidates, at least for the "constraint" filter. However, future users may want to take precautions if, for example, they want finer bin widths in their conditional likelihood histograms.

3.2 Inputs

As mentioned in the introduction (cf. Sec. 1.3), the ANTARCA analysis considered a candidate list based on two different catalogs, filtered differently:

- Catalog of 167 entries: Candidates were retained if the ANTARES-only likelihood fit preferred at least one signal-like event;
- Catalog of 1453 entries: Only the single candidate with the best ANTARES-only p-value was retained.

Following this procedure, a total of 106 $(105 + 1)$ candidates were ultimately selected. To compute the trial factor of this procedure, ideally we would have signal and background distributions available for all the candidates in the original catalogs. However, these are only available for the 106 retained ANTARCA candidates. To proceed, it was therefore assumed that these 106 candidates are representative of the full candidate population, and duplicated as needed (cf. Sec. 3.4). Concretely, the available distributions/histograms for each candidate are:

- hs_gen_ARCA6BP

- hs_gen_ARCA6GP
- hs_gen_ARCA8
- hs_gen_ARCA19
- hs_gen_ARCA21
- hs_gen_ANTARES_s
- hs_gen_ANTARES_t
- hs_eva_ARCA6BP
- hs_eva_ARCA6GP
- hs_eva_ARCA8
- hs_eva_ARCA19
- hs_eva_ARCA21
- hs_eva_ANTARES_s
- hs_eva_ANTARES_t
- hb_gen_ARCA6BP
- hb_gen_ARCA6GP
- hb_gen_ARCA8
- hb_gen_ARCA19
- hb_gen_ARCA21
- hb_gen_ANTARES_s
- hb_gen_ANTARES_t
- hb_eva_ARCA6BP
- hb_eva_ARCA6GP
- hb_eva_ARCA8
- hb_eva_ARCA19
- hb_eva_ARCA21
- hb_eva_ANTARES_s
- hb_eva_ANTARES_t

Notably, the input signal and background distributions were split by detector, but the ANTARES histograms were *not* split pre- and post- 2018. Thus, the PEs were generated as if the whole of the ANTARES data was used to filter the original catalogs, instead of just the data prior to 2017/12/31 - this will tend to overestimate the computed trial factor, compared to the real analysis.

3.3 Running the code

Once JPseudoExperimentCondNLLR.cc and JTrialFactorsFromCondNLLR have been compiled, you must run them one after the other. Here is an example of a bash script, px_condnllr.sh to produce the conditional NLLR histograms (adapt it to your specific needs):

```
#!/bin/bash

source $JPP_DIR/setenv.sh $JPP_DIR >& /dev/null
DEBUG=2

if (( $# != 2 )); then
    echo ``Usage: $script <number of PEs> <candidate ref number>''
    exit
fi

NUMBER_OF_PES=$argv[1]
REFCAND=$argv[2]

SIGNAL_STRENGTH=1
BOOST_EXP=1

INPUT_FOLDER=<folder_with_signal_and_background_histos_files>
OUTPUT_FOLDER=<folder_to_save_outputs>

FILE_NAME=${INPUT_FOLDER}/histos_${REFCAND}_det_A6BP_A6GP_A8_A19_A21_As_At_.root
OUTPUT_NAME=${OUTPUT_FOLDER}/CondNLLR_ref${REFCAND}_${NUMBER_OF_PES}PEs.root
if (( $BOOST_EXP != 1 )); then
  OUTPUT_NAME=${OUTPUT_FOLDER}/CondNLLR_ref${REFCAND}_boost${BOOST_EXP}_${NUMBER_OF_PES
fi

HS_antares_s=${FILE_NAME}:hs_gen_ANTARES_s
HB_antares_s=${FILE_NAME}:hb_gen_ANTARES_s
HS_antares_t=${FILE_NAME}:hs_gen_ANTARES_t
HB_antares_t=${FILE_NAME}:hb_gen_ANTARES_t
HS_arca6bp=${FILE_NAME}:hs_gen_ARCA6BP
HB_arca6bp=${FILE_NAME}:hb_gen_ARCA6BP
HS_arca6gp=${FILE_NAME}:hs_gen_ARCA6GP
HB_arca6gp=${FILE_NAME}:hb_gen_ARCA6GP
HS_arca8=${FILE_NAME}:hs_gen_ARCA8
HB_arca8=${FILE_NAME}:hb_gen_ARCA8
HS_arca19=${FILE_NAME}:hs_gen_ARCA19
HB_arca19=${FILE_NAME}:hb_gen_ARCA19
HS_arca21=${FILE_NAME}:hs_gen_ARCA21
HB_arca21=${FILE_NAME}:hb_gen_ARCA21

# pseudo experiments

if (( $NUMBER_OF_PES > 0 )); then
    ./JPseudoExperimentCondNLLR.cc \
        –o $OUTPUT_NAME  \
        –E ``$HS_antares_s $HB_antares_s $HS_antares_s $HB_antares_s 1 fixed fixed''
        –E ``$HS_antares_t $HB_antares_t $HS_antares_t $HB_antares_t 1 fixed fixed''
        –F ``$HS_arca6bp $HB_arca6bp $HS_arca6bp $HB_arca6bp $BOOST_EXP fixed fixed''
        –F ``$HS_arca6gp $HB_arca6gp $HS_arca6gp $HB_arca6gp $BOOST_EXP fixed fixed''
        –F ``$HS_arca8 $HB_arca8 $HS_arca8 $HB_arca8 $BOOST_EXP fixed fixed'' \
        –F ``$HS_arca19 $HB_arca19 $HS_arca19 $HB_arca19 $BOOST_EXP fixed fixed'' \
        –F ``$HS_arca21 $HB_arca21 $HS_arca21 $HB_arca21 $BOOST_EXP fixed fixed'' \
        –n $NUMBER_OF_PES         \
        –s $SIGNAL_STRENGTH       \
            -x "120 0 12"                \
            -S 0                         \
            -d $DEBUG

fi
echo ''Created $OUTPUT_NAME''
echo ''''
```

We then call this script (or an equivalent one) for all the 106 candidates, as follows:

```
$ for i in {0..105}; do ./px_condnllr.sh 1000000 $i; done
```

Depending on the number of candidates and the number of PEs, the previous step might take some time (a few hours in the specific example provided above) - use your local cluster resources if needed. Once the conditional likelihood files have been generated, we can quickly compute the trial factors:

```
$ ls <folder_with_outputs> >> candlist.txt
$ cd <folder_with_outputs>$
$ JTrialFactorsFromCondNLLR -i candlist.txt   \
    -n <duplication_factor> -x ''50 0 5'' -o  TrialFactors.root
```

A simple snakemake workflow could be developed to manage the two steps automatically, and to submit jobs to your local cluster automatically. **NB:** the bash code in this subsection is approximate. Do not copy the shell scripts as-is, but instead make sure you have defined sensible naming conventions, paths, etc. for every step.

3.4 Results

JPseudoExperimentCondNLLR.cc was used to generate the conditional-likelihood histograms. Initial tests were performed with 10^6 PEs per candidate. An example can be seen in Fig. 1. It confirms the strong correlation between the obtained NLLR in the "first" (i.e. ANTARES) period and in the complete dataset, justifying the need for proper computation of the trial factors.

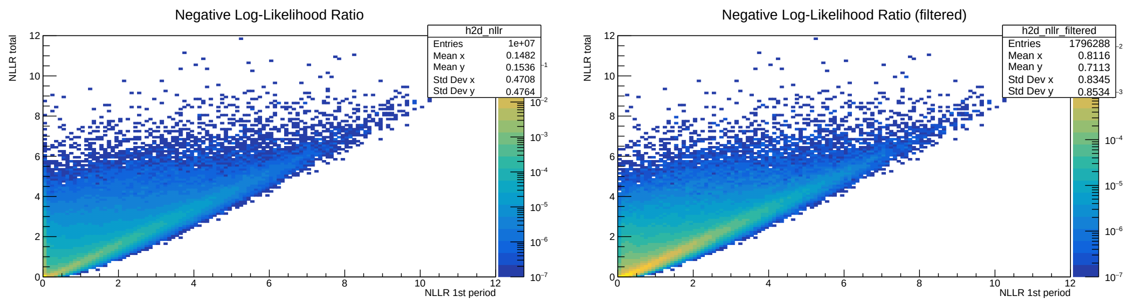

Figure 1: Conditional likelihood histograms. Left: For all PEs. Right: For those where the fit during the ANTARES period had $n_{signal} > 1$.

Based on the assumption that the remaining candidates are representative of the original candidate list, these conditional-likelihood histograms were duplicated to emulate the size of the original catalogs when generating the trial factor curves:

- $106 \times 2 = 212$ candidates for the “signal-over-threshold” (167-entry) catalog,
- $106 \times 13 = 1378$ candidates for the “best-of” (1453-entry) catalog.

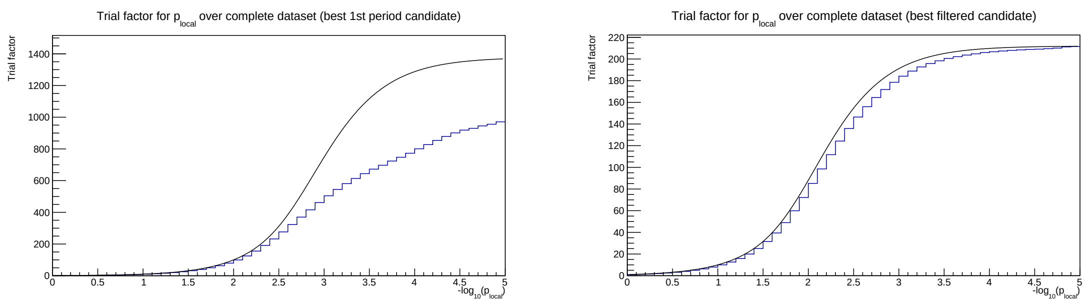

Figure 2: Trial factors. The blue histograms represent the trial factors obtained using the filtering strategies described above, while the smooth black curves represent the trial factor that would result from keeping all candidates without filtering. Left: For the 1378 artificial catalog, keeping only the “best” candidate during the selection period. Right: For the 212 artificial catalog, keeping only the candidates where the fit of the ANTARES data had $n_{signal} > 1$.

We use these trial factors to correct the local p-values reported in the ANTARCA internal note:

- “Best” candidate from the filtered 167-entry catalog: $p_{local} = 9 \cdot 10^{-4}$; $T_{c167}(9 \cdot 10^{-4} \simeq 150 \Rightarrow p_{c167} \simeq 0.14$. Compare with the currently reported value of 0.095.
- “Best” candidate from the filtered 1453-entry catalog: $p_{local} = 4 \cdot 10^{-5}$; $T_{c1453}(4 \cdot 10^{-5} \simeq 950 \Rightarrow p_{c1453} \simeq 3.6 \cdot 10^{-2}$. Compare with the currently reported value of $4 \cdot 10^{-3}$.

where the trial factors were rescaled to match the true catalog size, rather than the simulated one (i.e. 212 back to 167, 1378 back to 1453). Additionally, since these two catalogs correspond to two separate analyses searching for the same signal, the effective trial factor combines both approaches as follows:

$$\begin{aligned} 1 - T_{c167+c1453}(p) \cdot p &= (1 - T_{c167}(p) \cdot p)\,(1 - T_{c1453}(p) \cdot p) \\ \Rightarrow T_{c167+c1453}(p) &= T_{c167}(p) + T_{c1453}(p) - T_{c167}(p) T_{c1453}(p) \cdot p \end{aligned} \tag{7}$$

giving $T_{c167+c1453}(4 \cdot 10^{-5}) \simeq 1100$, and hence $p_{c167+c1453} \simeq 4.5 \cdot 10^{-2}$, which is roughly a factor of 11 worse than the smallest post-trial p-value quoted in the internal ANTARCA note. Note that this discrepancy would likely be reduced if the calculation could be repeated using signal and background distributions where the pre-2018 ANTARES data is properly split from the latter data.

We do observe a discrepancy between these results and the ones in the real ANTARCA analysis: the latter retains 105 candidates out of the 167-entry catalog, while the filtering applied here (requiring more than one fitted signal event in ANTARES data) retains only about 31.5 candidates out of 212. This suggests that the implemented filtering is not fully equivalent to the ANTARCA procedure, and therefore the derived trial factors for this catalog tend to underestimate those of the real analysis. However, since the trial factors for the 1453-entry catalog are much larger than those from the 167-entry catalog, this does not strongly affect the final numbers.

3.5 Effect of increased exposure

To explore the effect of a larger ARCA dataset, while keeping the same candidate filtering procedure based on ANTARES data, a larger exposure was approximated by scaling up the ARCA signal and background distributions; the ANTARES expectations are kept unchanged. Scaling factors of 5 and 20 were tested. The results are shown in Fig. 3.

The largest impact is observed in the “best-candidate” filtering approach. At $p_{local} = 10^{-5}$, the trial factor decreases from approximately ~ 970 (nominal exposure), to ~ 400 ($\times 5$ exposure), and

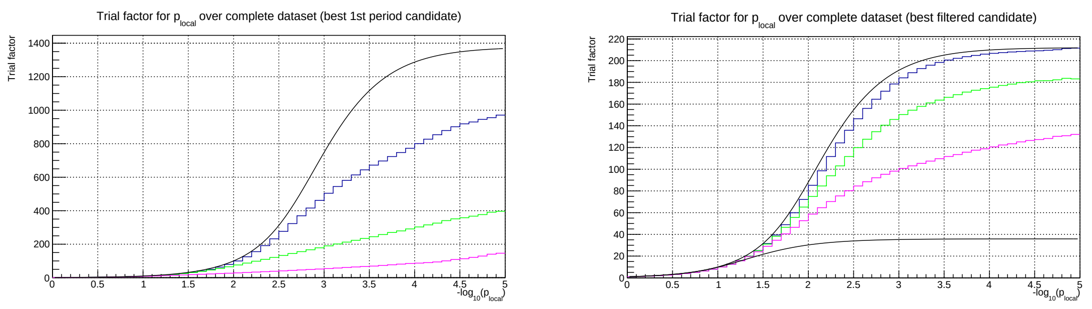

Figure 3: Trial factors when artificially increasing ARCA exposure by scaling up its signal and background expectations. The blue, green and purple histograms represent the trial factors obtained using the filtering strategies for 1, 5 and 20 times the ARCA exposure respectively. The (top) smooth black curves represent the trial factor that would result from keeping all candidates without filtering; for the right plot, the bottom smooth black curve is the (incorrect) naive trial factor, based only on the number of candidates that pass the filter. Left: For the 1378 artificial catalog, keeping only the "best" candidate during the selection period. Right: For the 212 artificial catalog, keeping only the candidates where the fit of the ANTARES data had $n_{signal} > 1$.

to $\sim$ 150 ($\times$20 exposure), starting from a candidate list of 1378 sources. However, the trial factor still appears to increase toward lower p-values in the boosted-exposure cases, indicating that higher-statistics simulations would be required to fully characterize this regime.

For the "signal-over-threshold" filtering approach, the improvement is limited. At $p_{local} = 10^{-5}$, the trial factor decreases from approximately $\sim$ 210 (nominal exposure), to $\sim$ 180 ($\times$5), and to $\sim$ 130 ($\times$20), starting from a list of 212 candidates.

3.6 Discussion

The present note provides indicative estimates of the trial factors associated with the ANTARCA candidate-filtering strategies, but is constrained by the lack of signal and background distributions for the full original candidate lists, and by the inability to separate ANTARES data before and after 2017/12/31. Updating either would provide a more accurate estimate of the true trial factor.

The main conclusion is that there is a "memory" effect of sorts for the trial factor due to using a candidate list filtered based on ANTARES data, and that this effect will decrease but not vanish for the whole duration of the KM3NeT experiment. It is thus critical to carefully define a candidate selection strategy for astronomical point-source searches. Any strategy that uses the "performance" of a candidate during part of our data-taking to include or exclude it from reported results should be considered with extreme caution.

On the flip-side, trial factors are only one of the elements to be considered when defining a robust strategy for future analyses. It is crucial to combine them with studies of the sensitivity to real sources, i.e. simulating signal strengths above zero, and how they are affected by candidate filters. While that falls beyond the scope of this note, the two *have* to be considered together. Trial factors, in isolation, can and should only be used for proper *post hoc* reporting of results of an experiment, and not to define future strategy.

A Appendix

Nothing to see here