EnsScores

Ensemble Scores

User's guide

Jean-Michel Brankart

`https://github.com/brankart/ensdam`

Institut des Géosciences de l'Environnement
Université Grenoble Alpes, CNRS, France

The purpose of EnsScores is to provide tools to compute probabilistic scores for ensemble simulations. The scores evaluates the reliability and/or the resolution of an ensemble simulation by comparison to verification data. For posterior ensembles (conditioned to observations), there is also a score to evaluate the optimality of the result, by testing its compatibility with observation uncertainty.

The tools are provided as a library of modules, which can be easily plugged in an existing software. This library includes:

- the computation of the Continuous Rank Probability Score (CRPS);
- the computation of the Reduced Centered Random variable (RCRV) score;
- the computation of scores based on the relative entropy of user-defined events;
- the computation of an optimality score for posterior ensembles.

1 Description of the methods

The standard protocol to evaluate the performance of ensemble simulations is to measure the *reliability* and the *resolution* of the ensemble using *verification data* (Toth et al., 2003; Candille and Talagrand, 2005; Candille et al., 2007). These scores can be used to compare ensemble forecasting systems, ensemble analyses, or to evaluate the impact of observations in these systems (using observation system simulation experiments). In the case of a posterior ensemble (conditioned on observations), it is also possible to check that the observations have been used optimally. A necessary condition of *optimality* is indeed that the distance between the observations and the ensemble members is compatible with the probability distribution of observation errors.

1.1 Reliability

Reliability is a measure of the consistency between the ensemble and the verification data. The idea is to check the hypothesis that the verification data are drawn from the same probability distribution as the ensemble members. For instance, a necessary condition of reliability is that verification data have the same probability to fall in every interval defined by the ensemble members. This can be checked by drawing a *rank histogram* of the verification data (Anderson, 1996), as used and described for instance in Candille et al. (2015) or Garnier et al. (2016), and as can be computed by the **score_ranks** module of this package.

To obtain a more compact score, we can center/reduce the verification data with the ensemble mean and ensemble standard deviation. A necessary condition of reliability (less stringent than the rank histogram) is then that this Reduced Centered Random Variable (RCRV) has zero mean and unit standard deviation. This is what is checked by the RCRV score (Candille et al., 2007), as computed by the **score_rcrv** module of this package.

The Continuous Rank Probability Score (CRPS), as computed by the **score_crps** module of this package, also provides an evaluation of reliability using the decomposition of the score into reliability and resolution, as described in Hersbach (2000): CRPS = Reli + Resol. In this case, it is important to check that the reliability component of the CRPS is much smaller than the resolution component.

Reliability is indeed the most important feature of an ensemble simulation. There is no point having good resolution or optimality scores if the ensemble is not reliable.

1.2 Resolution

Resolution is a measure of the accuracy of the ensemble, or the amount of information that it provides about the system. For instance, if the ensemble is perfectly reliable, a small ensemble spread will provide a better resolution than a large ensemble spread.

The Continuous Rank Probability Score (CRPS), as computed by the **score_crps** module of this package, provides an evaluation of resolution that complements the reliability component to explain the total CRPS score. The total CRPS score is a measure of the misfit between the Cumulative Distribution Functions (cdf) described by the ensemble (stepwise cdf) and the verification data (Heavside cdf). The score is thus expressed in the unit of the input variable, and the smaller the score, the better. For a perfectly reliable ensemble (Reli=0), this misfit (i.e. the total CRPS score) is only due to the ensemble spread. For an ensemble without spread, this misfit is only due to a shift between the zero-spread ensemble and the verification data (a very bad reliability, unless this shift if zero).

Providing that reliability has been verified, information theory can also provide means of evaluating the resolution of an ensemble simulation in terms of the amount of information that it provides about specific events (Roulston and Smith, 2002). From the probability distribution of an event described by the ensemble, it is for instance possible to evaluate how much

information has been gained as compared to a climatological distribution or as compared to a prior distribution. This is the purpose of the **score_entropy** module of this package, which implements the method described in Germineaud et al. (2019). In this paper, the method was used in the context of an Observation System Simulation Experiment, to evaluate how much information was gained about specific events for different quality of observation systems (in terms of coverage and accuracy).

1.3 Optimality

When a prior ensemble is conditioned to observations to produce a posterior ensemble, what is expected from the observational update is that the resolution can be improved (by the information brought by the observations), without degrading reliability. These scores tell us how much the updated ensemble has improved as compared to the prior ensemble. However, they do not tell us if we made the best possible use of the available observations. Is the updated ensemble close enough to observations to be consistent with the probability distribution of observation errors?

To evaluate this, we can use the optimality score proposed in Brankart (2019), as implemented in the **score_optimality** module of this package. In short, this score is obtained by computing the rank r^o_{ij} of every observation y^o_j, $j = 1, \ldots, p$ in the probability distribution for observation errors $p(y^o_j|\mathbf{x}_i)$, conditioned on every member $\mathbf{x}_i$, $i = 1, \ldots, m$ of the ensemble. If optimality is achieved, this rank is uniformly distributed between 0 and 1. To obtain a single score, we transform this uniform number into a $\mathcal{N}(0, 1)$ number, and take the mean square. This defines the optimality score, which is expected to be equal to 1 (for $p \to \infty$ and $m \to \infty$).

This score does not make any assumption on the probability distribution described by the ensemble or on observation error probability distribution. However, in the particular case of Gaussian distributions, the score is equivalent to computing the mean square misfit between the observations and all *individual ensemble members*, and check that the result is equal to the observation error variance. In the Gaussian case, this mean square misfit can be decomposed as the sum of the variance of the residual error (ensemble spread) and the mean square observation misfit with the ensemble mean (Talagrand, 1999). The more observational information, the smaller the ensemble spread, and the larger the observation misfit to the ensemble mean. It is indeed the sum of these two components that must be equal to the observation error variance. By looking at the misfit between observations and individual ensemble members (rather than the ensemble mean), we can directly compare to the observation error distribution and generalize the optimality check to non-Gaussian distributions.

1.4 About verification data

In the above discussion about reliability and resolution, it was assumed that the verification data are error free, as can be the case in twin assimilation experiments or in observation system simulation experiments. In this case, the reliability of the ensemble can be evaluated directly using the modules provided in this package.

However, if the verification data are observations, the observation equivalent of the ensemble members must be perturbed with a random observation error before checking reliability (with a rank histogram or the RCRV score). This can be very important if observation errors are not negligible as compared to the spread of the ensemble.

2 Description of the modules

In this section, the modules are described one by one, giving for each of them: the method that has been implemented, the list of public variables and public routines (with a description of input and output data), the MPI parallelization, and an estimation of the computational cost as a function of the size of the problem.

2.1 Module: score_ranks

The purpose of this module is to compute the rank of the verification data in the ensemble simulation and to produce a rank histogram.

Method

The algorithm loops on the verification data, sort the ensemble members for each of them, and compute the rank of each element of the verification data within the sorted ensemble values. From the vector of ranks, a rank histogram is produced.

Public variables

mpi_comm_score_ranks: MPI communicator to use (default=mpi_comm_world).

Public routines

compute_ranks: compute CRPS score (with an option to partition the data and compute the score for each subset of data):

`ens` **(input)** : ensemble to evaluate (model equivalent to verification data);

`verif` **(input)** : verification data;

`ranks` **(output)** : ranks of the verification data within the ensemble;

`rank_histogram` **(output, optional)** : rank histogram.

MPI parallelization

Parallelization is obtained by making each processor call the routine for a different chunk of the verification data. The routine combines the contributions from all processors to compute a global rank histogram.

Computational cost

The cost mainly results from the sorting of the ensemble members for each variable in the verification data. It is thus proportional to $nm \log_2 m$, if m is the size of the ensemble and n, the size of the verification dataset.

2.2 Module: score_crps

The purpose of this module is to compute the CRPS of an ensemble simulation by comparison to verification data.

Method

The algorithm loops on the verification data, to accumulate the contribution of each of them (in the intermediate arrays `aa` and `bb`), and then compute the overal CRPS score, together with the reliability and resolution components.

Public variables

mpi_comm_score_crps: MPI communicator to use (default=mpi_comm_world).

crps_missing_value: missing value to use where no valid data is available (default=-9999.).

Public routines

crps_score: compute CRPS score (with an option to partition the data and compute the score for each subset of data):

`crps` **(output)** : CRPS score for each subset of data;

`reliability` **(output)** : reliability part of CRPS;

`resolution` **(output)** : resolution part of CRPS;

`ens` **(input)** : ensemble to evaluate (model equivalent to verification data);

`verif` **(input)** : verification data;

`partition` **(input, optional)** : partition of verification data (giving the index of the subset for each element of the data).

crps_cumul: accumulate data to prepare the final computation of the score (for advanced use, if the full ensemble is only made progressively available).

crps_final: compute final score from accumulated data (for advanced use, if the full ensemble is only made progressively available).

MPI parallelization

Parallelization is obtained by making each processor call the routine for a different chunk of the verification data. The routine combines the contributions from all processors to compute a global score.

Computational cost

The cost mainly results from the sorting of the ensemble members for each variable in the verification data. It is thus proportional to $nm \log_2 m$, if m is the size of the ensemble and n, the size of the verification dataset.

2.3 Module: score_rcrv

The purpose of this module is to compute the RCRV of an ensemble simulation by comparison to verification data.

Method

The algorithm loops on the verification data, to accumulate the contribution of each of them (to the reduced bias and reduce spread), and then compute the overal RCRV score (bias component and spread component). If the anamorphosis option is activated (see below), the reduced variable (zero mean, unit standard deviation, Gaussian distribution) is obtained by anamorphosis (using quantiles of the input ensemble) rather than center/reduction (using the ensemble mean and standard deviation).

Public variables

mpi_comm_score_rcrv: MPI communicator to use (default=mpi_comm_world).

rcrv_missing_value: missing value to use where no valid data is available (default=-9999.).

rcrv_with_anamorphosis: use anamorphosis to compute reduced variable (default=.FALSE.).

rcrv_number_of_quantiles: number of quantiles to perform anamorphosis (default=11).

Public routines

rcrv_score: compute RCRV score (with an option to partition the data and compute the score for each subset of data):

`ens_bias` **(output)** : bias component of RCRV (should be 0);

`ens_spread` **(output)** : spread component of RCRV (should be 1);

`ens` **(input)** : ensemble to evaluate (model equivalent to verification data);

`verif` **(input)** : verification data.

`partition` **(input, optional)** : partition of verification data (giving the index of the subset for each element of the data).

rcrv_cumul: accumulate data to prepare the final computation of the score (for advanced use, if the full ensemble is only made progressively available).

MPI parallelization

Parallelization is obtained by making each processor call the routine for a different chunk of the verification data. The routine combines the contributions from all processors to compute a global score.

Computational cost

Without anamorphosis, the cost grows linearly with the ensemble size (m) and the size of the verification dataset (n). It is thus proportional to nm.
With anamorphosis, the cost mainly results from the sorting of the ensemble members for each variable in the verification data (to compute the quantiles). It is thus proportional to $nm \log_2 m$.

2.4 Module: score_entropy

The purpose of this module is to compute scores based on the relative entropy of user-defined events.

Method

To use this module, the user must provide a routine computing the outcome of one or several events from a given ensemble member. It is provided as the callback routine `events_outcome` (see below). The events must be discrete, with a finite number of possible outcomes (`jpo`, which should be much smaller than the ensemble size). The callback routine must provide the index of the outcome (between 1 and `jpo`) for each event. From this routine, the module can then compute:

- the probability distribution of the events in a given ensemble (routine `events_probability`): this is just computed as the number of members with a given outcome, divided by the size of the ensemble;

- the entropy associated to each event in a given ensemble (routine `events_entropy`), which can be directly computed from the probability distribution;
- the cross entropy between the ensemble distribution and a reference distribution provided by the user (routine `events_cross_entropy`);
- the relative entropy between the ensemble distribution and a reference distribution provided by the user (routine `events_relative_entropy`);
- a score (between 0 and 1) describing the gain obtained for each event, as compared to the reference distribution (routine `events_scores`), which is computed as the ratio between entropy and cross entropy.

For example, the reference distribution can be a climatological distribution or a prior distribution. The score describes the gain in resolution provided by the ensemble as compared to this distribution.

Public variables

score_entropy_base: base to use in the computation of logarithms (default=2.).

Public routines

events_score: compute entropy based score:

`score` **(output)** : ensemble score for each event;

`ens` **(input)** : ensemble to evaluate;

`pref` **(input)** : reference probability distribution for each event;

`events_outcome` **(input)** : callback routine providing the outcome of the events for a given member.

events_relative_entropy: compute relative entropy:

`relative_entropy` **(output)** : relative entropy (with respect to reference distribution);

`ens` **(input)** : ensemble to evaluate;

`pref` **(input)** : reference probability distribution for each event;

`events_outcome` **(input)** : callback routine providing the outcome of the events for a given member.

events_cross_entropy: compute cross entropy:

`cross_entropy` **(output)** : cross entropy (with reference distribution);

`ens` **(input)** : ensemble to evaluate;

`pref` **(input)** : reference probability distribution for each event;

`events_outcome` **(input)** : callback routine providing the outcome of the events for a given member.

events_entropy: compute ensemble entropy:

`entropy` **(output)** : ensemble entropy;

`number_outcome` **(input)** : number of possible outcomes for the events;

`ens` **(input)** : ensemble to evaluate;

`events_outcome` **(input)** : callback routine providing the outcome of the events for a given member.

events_probability: compute events marginal probability distributions from the ensemble:

`pens` **(output)** : ensemble probability distribution for each event;

`ens` **(input)** : ensemble to evaluate;

`events_outcome` **(input)** : callback routine providing the outcome of the events for a given member.

MPI parallelization

No parallelization is implemented.

Computational cost

The cost mainly depends on the evaluation of the outcome of an event in the callback routine provided by the user. This routine is called for each ensemble member.

2.5 Module: score_optimality

The purpose of this module is to compute a score evaluating the optimality of a posterior ensemble by comparison to observations.

Method

The algorithm loops on the observation data to accumulate the contribution of each of them (a 'normalized distance' between each observation and each ensemble member) and then compute the overal optimality score. The computation of this 'normalized distance' requires an evaluation of the observation error cdf (to compute the rank of the observation) and the inverse Gaussian cdf (to tranform this rank into a Gaussian number). In the case of Gaussian observation errors, this computation is simplified by computing the difference between each observation and each ensemble member, divided by the observation error standard deviation.

Public variables

mpi_comm_score_optimality: MPI communicator to use (default=mpi_comm_world).

optimality_missing_value: missing value to use where no valid data is available (default=-9999.).

Public routines

optimality_score: compute optimality score (with an option to partition the data and compute the score for each subset of data):

`ens_optimality` **(output)** : optimality score (should be 1);

`ens` **(input)** : ensemble to evaluate (model equivalent to observations);

`obs` **(input)** : observation data;

`partition` **(input, optional)** : partition of observation data (giving the index of the subset for each element of the data);

`std_obs` **or** `obs_cdf` **(input)** : standard deviation of observation errors (in the case of Gaussian observation errors) or callback routine providing the cdf of observation errors (in the case of non-Gaussian observation errors).

optimality_cumul: accumulate data to prepare the final computation of the score (for advanced use, if the full ensemble is only made progressively available).

MPI parallelization

Parallelization is obtained by making each processor call the routine for a different chunk of the verification data. The routine combines the contributions from all processors to compute a global score.

Computational cost

The cost mainly depends on the evaluation of the observation error cdf (callback routine provided by the user) and the inverse Gaussian cdf. The number of calls of these two routines is nm, where n is the number of observations and m, the ensemble size.

3 Examples

Two examples are provided to illustrate these modules: (i) a very simple idealized example illustrating how to call the routines, and how the results can depend on the number of verification data or the size of the ensemble, and (ii) a more sophisticated example, which corresponds to the use of these modules to evaluate the MCMC sampler in Brankart (2019).

3.1 A simple idealized test case

The code for this example is **score_idealized_example**. It illustrates all score modules: **scor_crps**, **scor_rcrv**, **scor_entropy** and **scor_optimality**.

In this example, a synthetic ensemble is generated by sampling independent Gaussian random number with zero mean and unit standard deviation. The size of the state vector is $n = 1000$ and the ensemble size is $m = 100$. The use of independent variables is overly simplistic, but it is sufficient to illustrate the computation of the scores, since all methods proceed in the same way whether the variables are independent or not.

One additional ensemble member is drawn from the same distribution to be used as the reference truth. This reference truth is used to generate one pseudo-observation for each of the variables, with an error standard deviation equal to $\sigma = 0.3$. A posterior ensemble is then computed by conditioning the prior ensemble to these pseudo-observations.

All modules are then used to evaluate the reliability, resolution and optimality of the prior and posterior ensemble, using the reference truth as verification data for reliability and resolution. It is then easy to see how the statistics of the scores can change with n, m and σ in the case of independent variables. One can also check how the score deteriorates when the reference truth is sampled from a distribution that is different from the prior ensemble, or when the observational update is made non-optimal (for instance using a wrong parameterization of observation errors).

Two events that can occur in the ensemble members will be used to illustrate the behaviour of the entropy score:

1. The first event is defined from the mean square of the ensemble member: above or below 1. In a real system, this could be the energy in the system above or below a user-requested level.

2. The second event is defined from the maximum absolute value of the ensemble member: above or below 3.3. In a real system, this could be the occurence of an extreme event above or below a user-requested level.

These are two binary events which have two outcomes with a similar probability (close to 0.5) in the prior ensemble. There is thus no much prior information about them, and we can check how much information can be gained by the observations.

- With the default setting of the example code ($m = 100$, $n = 1000$ and $\sigma = 0.3$) and with an optimal observational update, the output of the example is the following:

```
Prior CRPS reliability and resolution:     0.00104 0.56067
Posterior CRPS reliability and resolution: 0.00030 0.16223
Prior RCRV bias and spread:      0.02900  0.99723
Posterior RCRV bias and spread: -0.03375  1.00673
Prior probability distribution (event 1):     0.480 0.520
Prior probability distribution (event 2):     0.450 0.550
Posterior probability distribution (event 1): 0.810 0.190
Posterior probability distribution (event 2): 0.900 0.100
```

```
Entropy score (posterior vs prior, event 1): 0.676
Entropy score (posterior vs prior, event 2): 0.418
Prior optimality score:     4.81227
Posterior optimality score: 1.00351
```

We see that reliability score is good in the prior ensemble (by construction), and the it does not get worse after the observational update (posterior ensemble). The resolution of the ensemble has been improved by the observations: the CRPS resolution is smaller in the posterior ensemble, and the entropy score also shows an improvement (score well below 1). The probability concentrates on the first outcome of the two events. The optimality score decreases to 1, which means that the ensemble members are moved to the right typical 'distance' from the observations.

- If we decrease the observation error standard deviation with respect to the default setting to $\sigma = 0.05$, the output of the example is the following:

```
Prior CRPS reliability and resolution:     0.00104 0.56067
Posterior CRPS reliability and resolution: 0.00006 0.02847
Prior RCRV bias and spread:      0.02900  0.99723
Posterior RCRV bias and spread: -0.04206  1.01476
Prior probability distribution (event 1):     0.480 0.520
Prior probability distribution (event 2):     0.450 0.550
Posterior probability distribution (event 1): 1.000 0.000
Posterior probability distribution (event 2): 1.000 0.000
Entropy score (posterior vs prior, event 1): 0.000
Entropy score (posterior vs prior, event 2): 0.000
Prior optimality score:    28.12474
Posterior optimality score: 1.00250
```

Reliability and optimality are still good. Resolution has strongly improved by the much better information provided by the observations. Entropy has decreased to zero for the two events: all posterior members have the same outcome for each of them. Uncertainty about them has been removed by the observations (or, more precisely, the remaining uncertainty is too small to be resolved by a 100-member ensemble).

- Starting again from the default setting, we now modify the ensemble observational update algorithm by omitting the perturbations applied to the observations to perform the update of each ensemble member.

```
Prior CRPS reliability and resolution:     0.00104 0.56067
Posterior CRPS reliability and resolution: 0.03838 0.15373
Prior RCRV bias and spread:      0.02900  0.99723
Posterior RCRV bias and spread: -0.12757  3.46117
Prior probability distribution (event 1):     0.480 0.520
Prior probability distribution (event 2):     0.450 0.550
Posterior probability distribution (event 1): 1.000 0.000
Posterior probability distribution (event 2): 1.000 0.000
Entropy score (posterior vs prior, event 1): 0.000
Entropy score (posterior vs prior, event 2): 0.000
Prior optimality score:     4.81227
Posterior optimality score: 0.40346
```

In this case, the scheme is wrong, and we can see this in the scores. Reliability and optimality of the posterior ensemble have been lost. The posterior ensemble is no more compatible with the reference truth (e.g. RCRV spread = $3.46 \gg 1$), and it is too close to the observations (optimality score = $0.4 \ll 1$). Resolution is apparently improved (in the CRPS and entropy scores), but it does not serve any purpose since reliability is lost.

3.2 Evaluation of the MCMC sampler

The code for this example is **mcmc_ensemble_update**. It uses the score modules: **scor_crps** and **scor_optimality**.

This code is an implementation of the work done in Brankart (2019). Refer to this paper to understand what is done in the code.

The score modules are used to evaluate the ensemble that results from the application of the MCMC sampler. Calls to the routines to evaluate reliability, resolution and optimality of the ensemble can be found in the code and the results are displayed in the figures of the paper.

References

Anderson J. 1996. A method for producing and evaluating probabilistic forecasts from ensemble model integrations. *J. Climate*, **9**, 1518–1530.

Brankart J.-M., 2019: Implicitly Localized MCMC Sampler to Cope With Non-local/Non-linear Data Constraints in Large-Size Inverse Problems. *Front. Appl. Math. Stat.*, 5:58.

Candille G., and O. Talagrand, 2005: Evaluation of probabilistic prediction systems for a scalar variable. *Quart. J. Roy. Meteor. Soc.*, **131**, 2131–2150.

Candille G., C. Côté, P. L. Houtekamer, and G. Pellerin, 2007: Verification of an ensemble prediction system against observations. *Mon. Wea. Rev.*, **135**, 2688–2699.

Candille G., J.-M. Brankart, and P. Brasseur, 2015: Assessment of an ensemble system that assimilates Jason-1/Envisat altimeter data in a probabilistic model of the North Atlantic ocean circulation. *Ocean Science*, **11**, 425–438.

Garnier F., J.-M. Brankart, P. Brasseur and E. Cosme, 2016: Stochastic parameterizations of biogeochemical uncertainties in a 1/4° NEMO/PISCES model for probabilistic comparisons with ocean color data. *Journal of Marine Systems*, **155**, 59–72.

Germineaud, C., J.-M. Brankart, and P. Brasseur, 2019: An Ensemble-Based Probabilistic Score Approach to Compare Observation Scenarios: An Application to Biogeochemical-Argo Deployments. J. Atmos. Oceanic Technol., 36, 2307-2326.

Hersbach H., 2000: Decomposition of the continuous ranked probability score for ensemble prediction systems. *Wea. Forecasting*, **15**, 559–570.

Roulston M. S. and L. Smith, 2002. Evaluating probabilistic forecast using information theory. *Mon. Wea. Review*, **130**, 1653–1660.

Talagrand, O., 1999: A posteriori verification of analysis and assimilation algorithms, in: Workshop on diagnosis of data assimilation systems, 2–4 November 1998, ECMWF, Reading, UK, 1999.

Toth Z., O. Talagrand, G. Candille, and Y. Zhu. 2003. Probability and ensemble forecasts, in Forecast Verification: A Practitioner's Guide in Atmospheric Science, Jolliffe I. and D. B. Stephenson (eds), Wiley: UK. ISBN: 0-471-49 759-2, 137–163.