Reproducible Forecast Evaluation with the Solar Forecast Arbiter

Antonio Lorenzo
Assistant Research Scientist
UA Dept. of Hydrology & Atmospheric Sciences

DOE Solar Forecasting 2 Topic Area 1

William F. Holmgren, Univ. of Arizona
Clifford W. Hansen, Sandia Nat. Labs
Aidan Tuohy, EPRI
Justin Sharp, Sharply Focused

Leland J. Boeman, Univ. of Arizona
Adam Wigington, EPRI
David P. Larson, EPRI
Qin Wang, EPRI
Anastasios Golnas, DOE EERE

Purpose of the Solar Forecast Arbiter

- Provide a transparent, reproducible way to analyze solar power forecasts
- Forecast users (primarily electric utility companies) want to an easy way to compare forecasts from multiple vendors
- Forecast providers want to make sure their forecasts are evaluated fairly and accurately in a transparent way
- Other point forecasts can also be analyzed including air temperature, wind speed, etc.

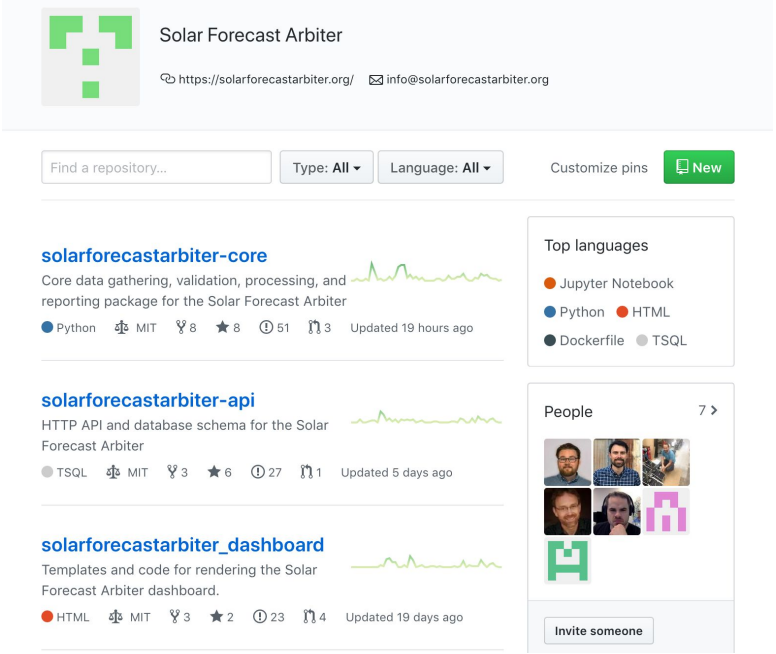
Typical User Story

- Utility Company needs a weather and solar power forecast from a vendor for each of its solar power plants
- Utility Company wants to run a trial over a period of a month of many different vendors to choose the best one
- Vendor X wants to earn Utility Company's business but wants to make sure it understands how its forecast will be evaluated and that the evaluation will be fair
- Solar Forecast Arbiter is built to facilitate this kind of trial

What is the Solar Forecast Arbiter?

Tool for analyzing point forecast time-series

- Web-based user interface
- HTTP RESTful API for scripting
- Python software package for analysis
- Scripts to redeploy entire software stack
- Detailed supporting documents
- Supported by stakeholder input, feedback

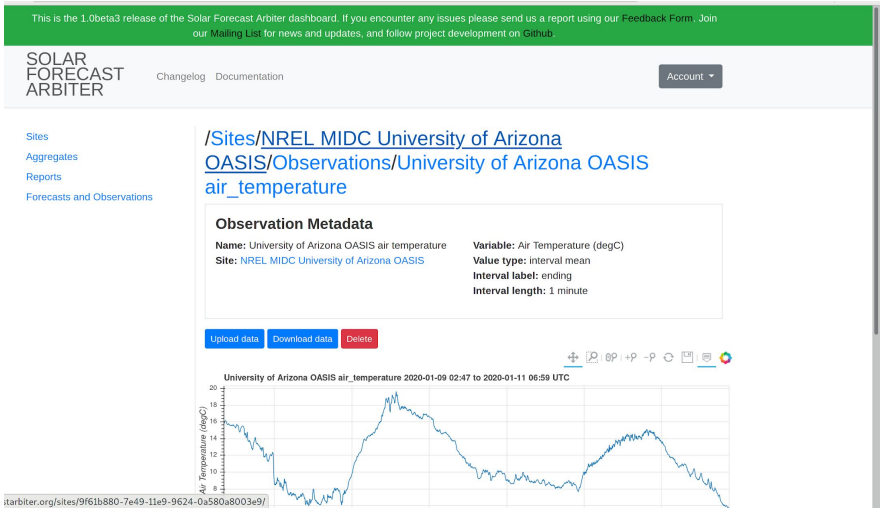


Open source. Transparently developed on GitHub

API & Dashboard Overview

- API
  - Provides data to dashboard
  - Enables programmatic access to observations, forecasts, reports
  - Background jobs automatically fetch data, generate forecasts and reports
  - Data stored in MySQL
  - Flask app
  - Role based access control
- Dashboard
  - Web GUI for most users
  - Visualize data, data validation, reports
  - Flask app

https://{api,dashboard}.solarforecastarbiter.org



Primary Output: Reports

Intro/Metadata

surfrad ghi hrrr gfs

This report of solar forecast accuracy was automatically generated using the Solar Forecast Arbiter.

Download as html or pdf (coming soon). The download is a ZIP archive that includes checksums for the report file and a PGP signature that can be used to verify the authenticity of the report. The Solar Forecast Arbiter PG key ID is 0x22bd497c0930f8b0.

Please see our GitHub repository for known issues with the reports or to create a new issue.

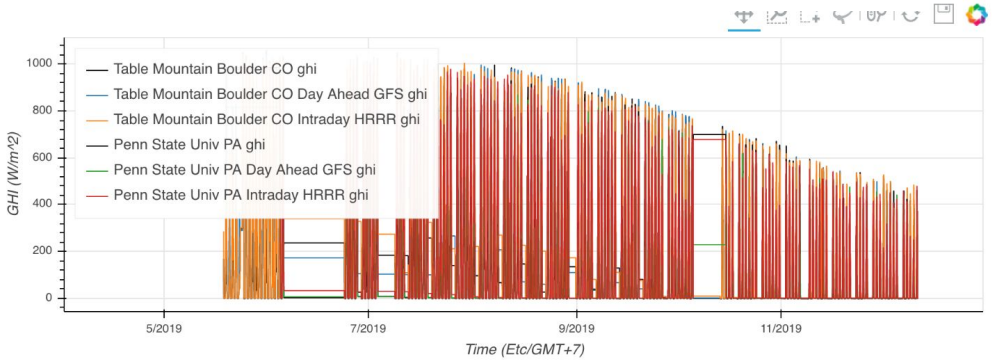
Contents:

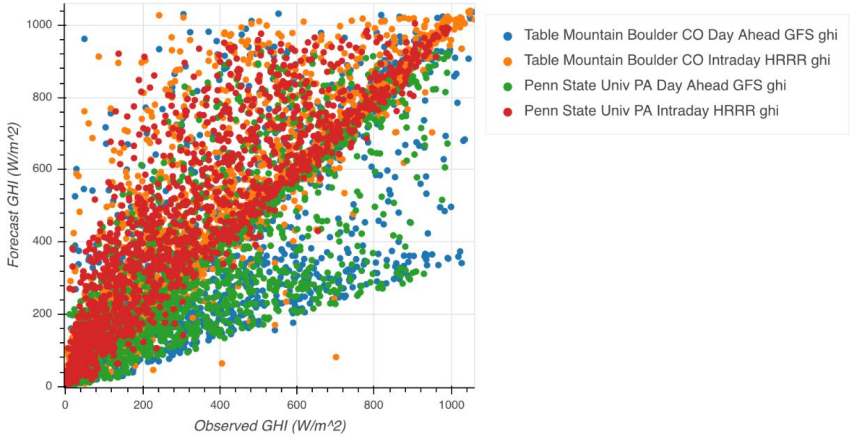
- Report metadata
- Data
  - Observations and forecasts
  - Data validation
- Metrics
  - Total analysis
  - Year analysis
  - Month of the year analysis
  - Hour of the day analysis
  - Date analysis
- Versions

Report metadata

- Name: surfrad ghi hrrr gfs
- Start: 2019-04-01 05:00:00 +0000
- End: 2019-12-31 03:00:00 +0000
- Generated at: 2019-12-16 22:57:19 +0000

Time-series and scatter plots



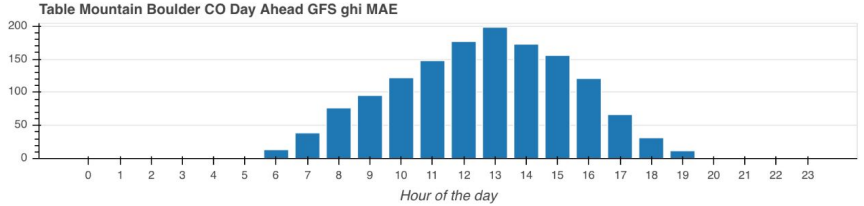


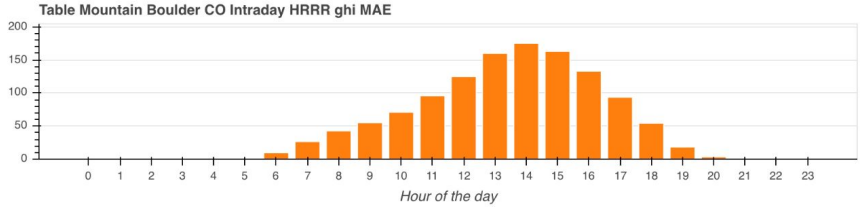
Primary Output: Reports

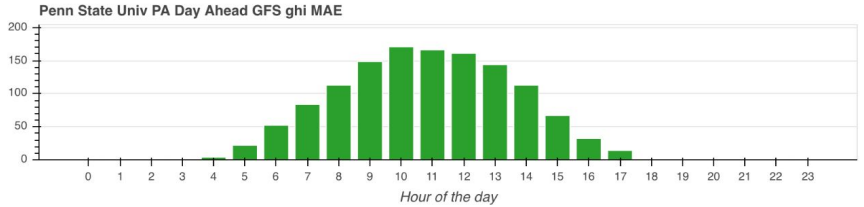
Metrics by hour

Hour of the day analysis

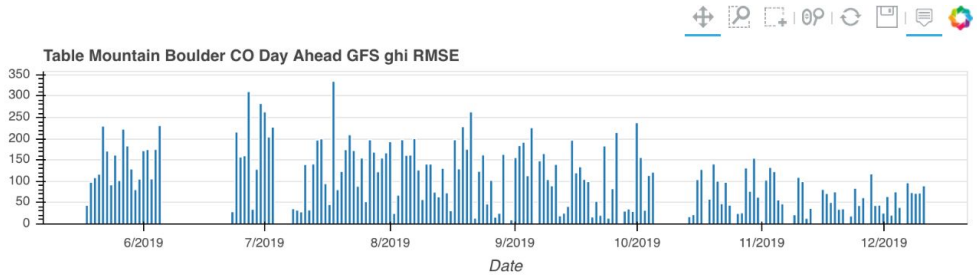
Metrics for each hour (0-23) of the analysis period are displayed in tables and figures below.

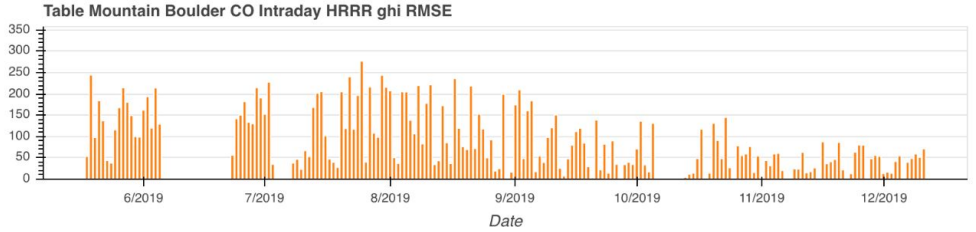


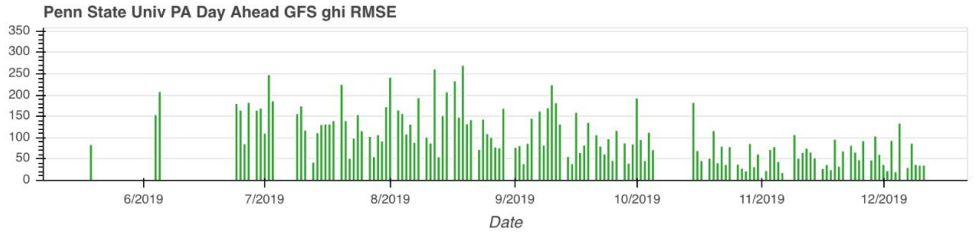




Metrics by day







Primary Output: Reports

Data Validation Results

The Solar Forecast Arbiter's data validation toolkit identified the following issues with the unprocessed observation data:

- USER FLAGGED: 0 intervals
- NIGHTTIME: 39855 intervals
- LIMITS EXCEEDED: 318 intervals
- STALE VALUES: 12104 intervals
- INTERPOLATED VALUES: 5598 intervals
- INCONSISTENT IRRADIANCE COMPONENTS: 0 intervals

These intervals were removed from the raw time series before resampling and realignment. For more details on the data validation results for each observation, please see the observation page linked to in the table above. Data providers may elect to reupload data to fix issues identified by the validation toolkit. The metrics computed in this report will remain unchanged, however, a user may generate a new report after the data provider submits new data. The online version of this report verifies that the data was not modified after the metrics were computed.

Versions

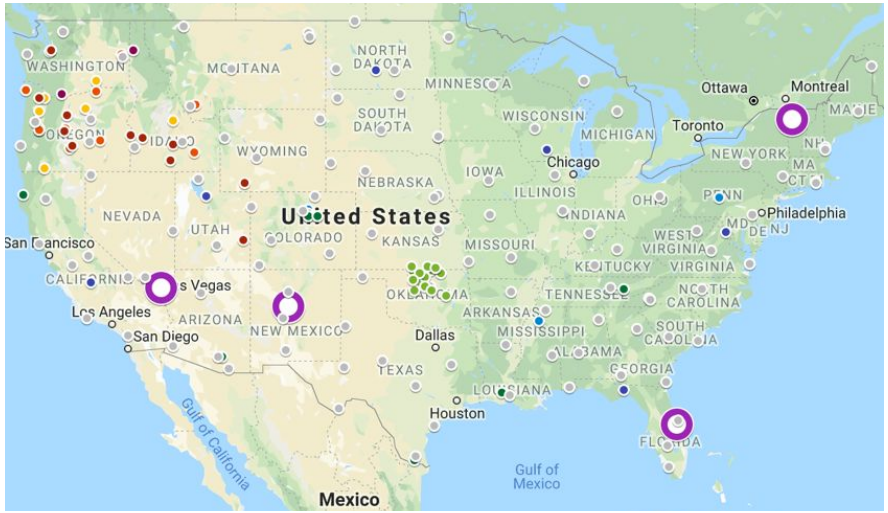
Package	Version
solarforecastarbiter	1.0b3
pvlib	0.6.3
pandas	0.25.3
numpy	1.17.4
bokeh	1.4.0
netcdf4	1.5.3
xarray	0.14.1
tables	3.6.1
numexpr	2.7.0
bottleneck	None
jinja2	2.10.3
python	3.7.5
platform	Linux-3.10.0-957.5.1.el7.x

solarforecastarbiter-core Package

- Python >= 3.7 only
- Primarily procedural
- Open source, MIT licensed
- Documented
- Well tested (~4k test statements of ~20k total)
- Openly developed on GitHub
- https://github.com/solararbiter/solarforecastarbiter-core

Reference Data & Forecasts

- Reference data and forecasts stored in API with retrieval functionality in core
- Reference data from a number of networks:
  - NOAA SURFRAD, SOLRAD, CRN
  - NREL MIDC
  - DOE RTC
  - U. Oregon
- Ability to retrieve/process NWP models: GFS, NAM, RAP, HRRR, GEFS



Data Model

- Metadata defined in Python 3.7 frozen dataclasses
- Ensures required metadata is present throughout processing
- Easier to track information throughout the processing and report generation code

```
@dataclass(frozen=True)
class Forecast(BaseModel, _ForecastDefaultsBase, _ForecastBase):
    """
    A class to hold metadata for Forecast objects.

    Parameters
    ----------
    name : str
        Name of the Forecast
    issue_time_of_day : datetime.time
        The time of day that a forecast run is issued, e.g. 00:30. For
        forecast runs issued multiple times within one day (e.g. hourly),
        this specifies the first issue time of day. Additional issue times
        are uniquely determined by the first issue time and the run length &
        issue frequency attribute.
    lead_time_to_start : pandas.Timedelta
        The difference between the issue time and the start of the first
        forecast interval, e.g. 1 hour.
    interval_length : pandas.Timedelta
        The length of time between consecutive data points, e.g. 5 minutes,
        1 hour.
    run_length : pandas.Timedelta
        The total length of a single issued forecast run, e.g. 1 hour.
        To enforce a continuous, non-overlapping sequence, this is equal
        to the forecast run issue frequency.
    interval_label : str
        Indicates if a time labels the beginning or the ending of an interval
        average, or indicates an instantaneous value, e.g. beginning, ending,
        instant.
    interval_value_type : str
        The type of the data in the forecast, e.g. mean, max, 95th percentile.
    variable : str
        The variable in the forecast, e.g. power, GHI, DNI. Each variable is
        associated with a standard unit.
```

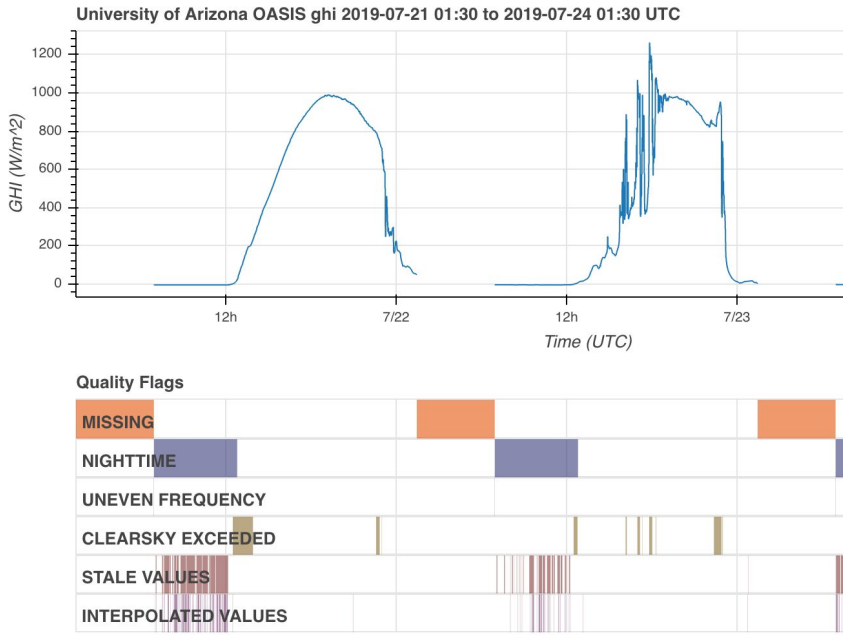
```
plant = datamodel.SolarPowerPlant(
    name='My Solar Plant', latitude=32.0, longitude=-110.0, elevation=1200,
    timezone='US/Arizona', modeling_parameters=datamodel.FixedTiltModelingParameters(
        ac_capacity=100, dc_capacity=90, temperature_coefficient=-0.003,
        dc_loss_factor=0, ac_loss_factor=0, surface_tilt=32.0, surface_azimuth=180.0
    )
)

ghi_observation = datamodel.Observation(
    name='My Solar Plant GHI', variable='ghi', interval_value_type='instantaneous',
    interval_length=pd.Timedelta('1min'), interval_label='instant',
    site=plant, uncertainty=0.1
)

ghi_forecast = datamodel.Forecast(
    name='My Solar plant GHI Forecast X', issue_time_of_day=dt.time(0, 0),
    lead_time_to_start=pd.Timedelta('1h'), interval_length=pd.Timedelta('1h'),
    run_length=pd.Timedelta('24h'), interval_label='beginning',
    interval_value_type='mean', variable='ghi', site=plant
)
```

```
In [3]: ghi_observation.interval_length
Out[3]: Timedelta('0 days 00:01:00')

In [4]: ghi_observation.interval_length = pd.Timedelta('3min')
---------------------------------------------------------------------------
FrozenInstanceError                       Traceback (most recent call last)
<ipython-input-4-534e86fad8c6> in <module>
----> 1 ghi_observation.interval_length = pd.Timedelta('3min')

<string> in __setattr__(self, name, value)

FrozenInstanceError: cannot assign to field 'interval_length'
```

```python
@dataclasses.dataclass(frozen=True)
class Observation(BaseModel):
    """
    A class for keeping track of metadata associated with an observation.
    Units are set according to the variable type.

    Parameters
    ----------
    name : str
        Name of the Observation
    variable : str
        Variable name, e.g. power, GHI. Each allowed variable has an
```

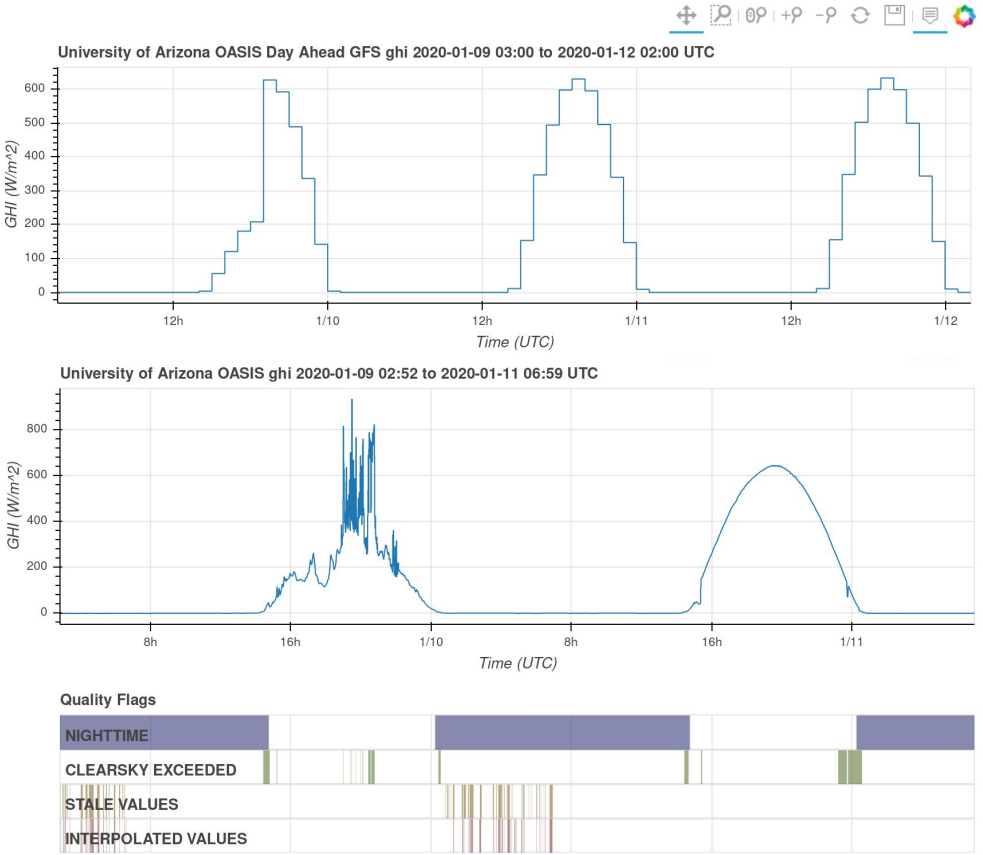
Data Validation

- Flags potential problems with user and reference data
- Automatically applied
- Report options control how flags should be used (e.g. exclude data, fill with 0)



Data Pre-processing

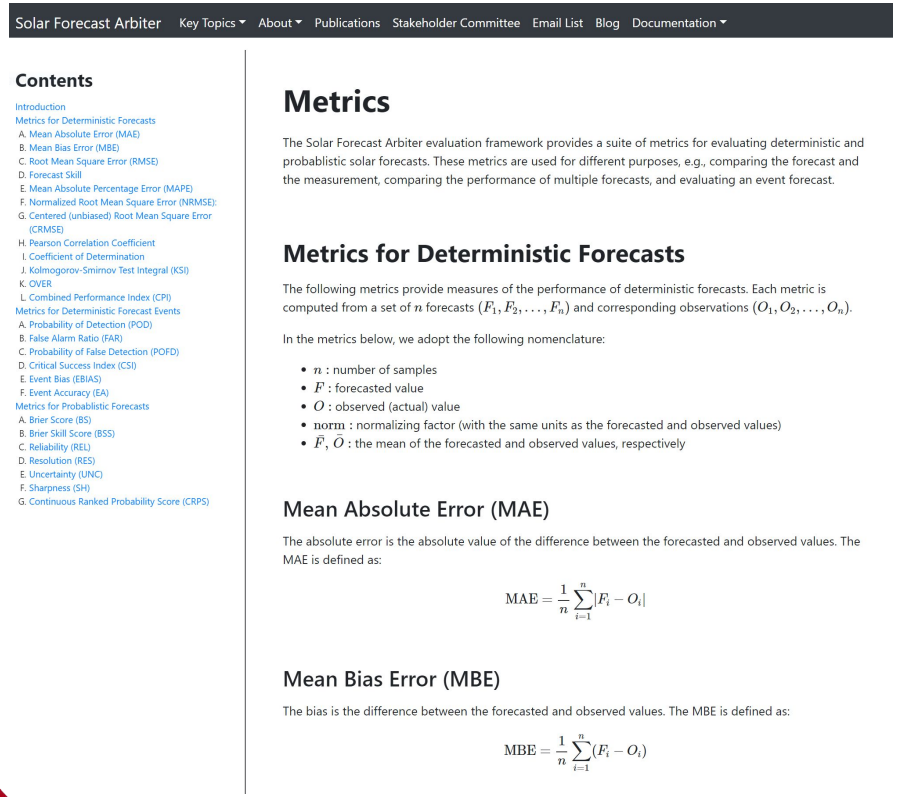
- Observations/forecasts must be pre-processed before comparing
- Pre-processing includes:
  - Aligning frequency
  - Handling data quality flags
  - Handling missing values
  - Restricting based on other filters (time of day, observation value)
- Each step is documented



Metrics

Stakeholder selections of:

- Deterministic Forecasts
- Event Forecasts
- Probabilistic Forecasts



Solar Forecast Arbiter | Key Topics | About | Publications | Stakeholder Committee | Email List | Blog | Documentation

Contents

Introduction
Metrics for Deterministic Forecasts
A. Mean Absolute Error (MAE)
B. Mean Bias Error (MBE)
C. Root Mean Square Error (RMSE)
D. Forecast Skill
E. Mean Absolute Percentage Error (MAPE)
F. Normalized Root Mean Square Error (NRMSE):
G. Centered (unbiased) Root Mean Square Error (CRMSE)
H. Pearson Correlation Coefficient
I. Coefficient of Determination
J. Kolmogorov-Smirnov Test Integral (KSI)
K. OVER
L. Combined Performance Index (CPI)
Metrics for Deterministic Forecast Events
A. Probability of Detection (POD)
B. False Alarm Ratio (FAR)
C. Probability of False Detection (POFD)
D. Critical Success Index (CSI)
E. Event Bias (EBIAS)
F. Event Accuracy (EA)
Metrics for Probablistic Forecasts
A. Brier Score (BS)
B. Brier Skill Score (BSS)
C. Reliability (REL)
D. Resolution (RES)
E. Uncertainty (UNC)
F. Sharpness (SH)
G. Continuous Ranked Probability Score (CRPS)

Metrics

The Solar Forecast Arbiter evaluation framework provides a suite of metrics for evaluating deterministic and probablistic solar forecasts. These metrics are used for different purposes, e.g., comparing the forecast and the measurement, comparing the performance of multiple forecasts, and evaluating an event forecast.

Metrics for Deterministic Forecasts

The following metrics provide measures of the performance of deterministic forecasts. Each metric is computed from a set of n forecasts $(F_1, F_2, \ldots, F_n)$ and corresponding observations $(O_1, O_2, \ldots, O_n)$.

In the metrics below, we adopt the following nomenclature:

- n : number of samples
- F : forecasted value
- O : observed (actual) value
- norm : normalizing factor (with the same units as the forecasted and observed values)
- $\bar{F}$, $\bar{O}$: the mean of the forecasted and observed values, respectively

Mean Absolute Error (MAE)

The absolute error is the absolute value of the difference between the forecasted and observed values. The MAE is defined as:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|F_i - O_i|$$

Mean Bias Error (MBE)

The bias is the difference between the forecasted and observed values. The MBE is defined as:

$$\mathrm{MBE} = \frac{1}{n}\sum_{i=1}^{n}(F_i - O_i)$$

Report Generation

- Includes:
  - data summary
  - filter results
  - time-series and scatter plots
  - metrics plots and tables
- Output to:
  - HTML
  - Jupyter notebook
  - PDF

- Interactive plots with Bokeh (HTML, Jupyter)
- Package versions included to reproduce report
- Checksum of data to verify it hasn't changed
- Download includes PGP signature to verify report authenticity

Conclusion

- Solar Forecast Arbiter includes a REST API, web dashboard, and python library enabling reproducible analysis of point forecasts

- Provides:
  - Reference forecasts and data
  - Data validation
  - Data pre-processing
  - Metrics calculation
  - Report generation
- Openly developed on GitHub
- https://github.com/solararbiter