Final Abstract Number
IN33B-2065

Session Date and Time
12/11/24 @ 13:40 - 17:30 EST

# Demonstrating New Tools to Foster and Expedite Standardized, Continental-Scale Hydrologic Evaluation

Matt Denno (PI), Katie van Werkhoven (co-PI) Sam Lamont (Developer), Sam Landsteiner (Developer), Jason Regina (Contributor), Andy Wood (Contributor), Josh Sturtevant (Contributor)

## Background

Operational hydrologic forecasts play a crucial role in guiding flood warnings, water supply operations, hydropower generation, and other essential water management activities. Recent federal investments and the public benefits of advanced forecasting capabilities have spurred numerous research initiatives aimed at furthering hydrologic forecasting. However, the influx of research poses a significant challenge: how to consistently and objectively measure forecast improvements from these new developments and determine which should transition into the operational forecast system. This challenge becomes even more complex when improving continental-scale forecasts, spanning thousands to millions of locations.

To address these challenges, the **Cooperative Institute for Research to Operations in Hydrology (CIROH)** initiated the development of a hydrologic model and forecast evaluation system known as Tools for Exploratory Evaluation in Hydrologic Research (TEEHR). TEEHR is designed to significantly enhance the evaluation of continental-scale hydrologic models and forecasts, promoting standardized assessment practices.

## Tools for Exploratory Evaluation in Hydrologic Research (TEEHR)

TEEHR is a Python package built for iterative and exploratory data analysis of hydrologic model performance. It offers scalability through PySpark integration, ensures robust data integrity with its internal framework, and provides flexibility for users to customize metrics, implement bootstrapping, and manipulate data with various grouping and filtering options. With its comprehensive data management and analytics capabilities, TEEHR facilitates efficient, standardized evaluation of continental-scale hydrologic data.

## Design Objectives

TEEHR's design objectives emphasize seamless integration into research workflows, utilization of modern data structures and computing platforms, scalability for large-domain evaluations, and simplified exploration of performance trends and potential drivers like climate and basin characteristics. It includes common and emerging evaluation methods such as error statistics, skill scores, hydrologic signatures, uncertainty quantification, and graphical methods. As an open-source, community-extensible project, TEEHR invites feedback and contributions from users to continuously improve its capabilities.

## Key Strengths

One of TEEHR's key strengths is its ability to allow users to ask complex questions of their hydrologic data. Users can compare simulations to a baseline during specific time periods, across different regions, or at locations with particular attributes. The tool supports limitless combinations of data grouping and filtering, enabling detailed and nuanced analyses. For example, a user might analyze how simulations compare to a baseline between specific dates, across different seasons, or between various ecoregions and land cover types.

## TEEHR is built on open-source python packages

dask, pandera, xarray, arch, PySpark, python, NumPy, GeoPandas, Pydantic, pandas, bokeh

...and many others

This project received funding under award NA22NWS4320003 from NOAA Cooperative Institute Program. The statements, findings, conclusions and recommendations are those of the author(s) and do not necessarily reflect the views of NOAA.

## TEEHR Features

### Evaluation

TEEHR enables conducting an exploratory evaluation through the concept of an Evaluation. In this context an Evaluation is a class in the TEEHR Python library and a directory with a defined set of subdirectories to hold all the relevant data that work together to make up an Evaluation. The Evaluation class provides users with methods to interact with the data that is in the Evaluation directory, including:

- Fetching data from external sources
- Loading data you already have
- Querying and visualizing Evaluation data

```
import teehr
ev = teehr.Evaluation(dir_path="/home/teehr/my_teehr_evaluation")
ev.enable_logging()
ev.clone_template()
```

The path can be a local path (e.g., /home/user/evaluation) or it can point to an S3 bucket (e.g., s3://bucket_name).

### Fetching

TEEHR has methods to efficiently fetch USGS streamflow and National Water Model (NWM) data to populate the dataset with common observed and simulated data.

```
ev.fetch.nwm_retrospective_points(
    nwm_version="nwm30",
    variable_name="streamflow",
    start_date=datetime(2022, 2, 22),
    end_date=datetime(2022, 2, 23)
)

ev.fetch.usgs_streamflow(
    start_date=datetime(2022, 2, 22),
    end_date=datetime(2022, 2, 23)
)
```

Both retrospective and near real-time NWM data can be fetched from AWS and GCS object storage, including point-based output (e.g., streamflow) and gridded data (e.g., forcings). TEEHR also has the capability to summarize gridded NWM data to polygons of interest, such as Mean Areal Precipitation estimates over catchment boundaries.

### Loading

TEEHR has built-in methods to ingest CSV, Parquet, NetCDF, and FEWS PI XML files, as well as various spatial file formats. These methods allow users to map the source schema (column names) to the TEEHR schema and validates the data and the schema before writing it to the TEEHR Evaluation dataset.

```
ev.locations.load_spatial(in_path="/home/datasets/usgs_gages.geojson")

ev.primary_timeseries.load_parquet(
    in_path="/home/files/my_primary_timeseries_data.parquet",
    field_mapping={
        "value_time": "source_value_time_field_name",
        "configuration": "source_configuration_field_name"
    }
)
```

### Data Validation

Before data is saved in the TEEHR dataset, it is validated against pre-defined schemas and domain variables.

### Joining Timeseries with UDF’s

TEEHR efficiently pairs the primary and secondary timeseries data with options to add location attributes and execute User-Defined Functions adding custom fields, which can be used in queries and metric calculations through filtering and grouping.

```
ev.joined_timeseries.create(add_attrs=True, execute_udf=True)
```

### Grouping and Filtering

The grouping and filtering capabilities in TEEHR provide the ability to explore models across different subsets of the data, allowing us to gain insight to model performance and iteratively explore an Evaluation Dataset.

### Event Detection

(*in development*) Methods are in development to identify hydrological events (e.g., flood peaks) in timeseries using multiple algorithms.

### Querying

The data within a TEEHR Evaluation dataset can be queried and returned as a Pandas DataFrame or a GeoDataFrame. The filtering and ordering arguments can be used to subset and order the data.

```
ts_df = ev.primary_timeseries.query(
    filters=[
        "value_time > '2022-01-01'",
        "value_time < '2022-01-02'",
        "location_id = 'gage-C'"
    ],
    order_by=["location_id", "value_time"]
).to_pandas()
```

### Metrics

TEEHR calculates four categories of metrics and can take advantage of a distributed cluster using PySpark to process huge amounts of data across large spatial and temporal scales.

- Timeseries Signatures**:** Operate on a single timeseries. (Average, Variance, …)
- Comparative**:** Operate on two timeseries, typically comparing a “modeled” or “secondary” timeseries”, to an “observed” or “primary” timeseries. (KGE, RMSE, …)
- Ensemble**:** (*in development)* Compare sets of forecast members to a single primary timeseries. (CRPS, …)
- Event-based: (*in development*) Evaluate performance of hydrological events within a timeseries (FAR, CSI, …)

```
from teehr import Metrics as m

kge = KlingGuptaEfficiency()
rmse = RootMeanSquareError()
bias = RelativeBias()

metrics_df = ev.metrics.query(
    include_metrics=[kge, rmse, bias],
    filters=["primary_location_id = 'gage-A'"],
    group_by=["primary_location_id"]
).to_pandas()
```

Metric queries can operate on population subsets using the filtering and grouping arguments.

### Bootstrapping Uncertainty

Comparative metrics in TEEHR can be customized to include Bootstrapping to quantify uncertainty in the metrics results.

```
from teehr import Bootstrappers as b

boot = b.CircularBlock(
    seed=40,
    block_size=100,
    quantiles=None,
    reps=500
)
kge = Metrics.KlingGuptaEfficiency(bootstrap=boot)
```

### Method Chaining

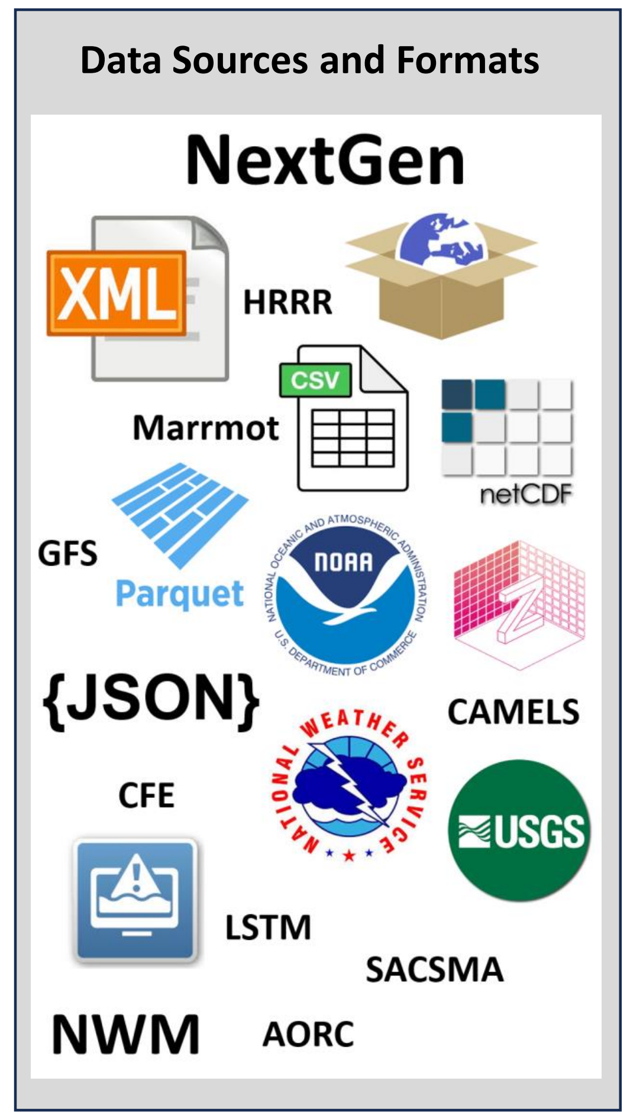
Methods can be chained to create workflows of successive steps.

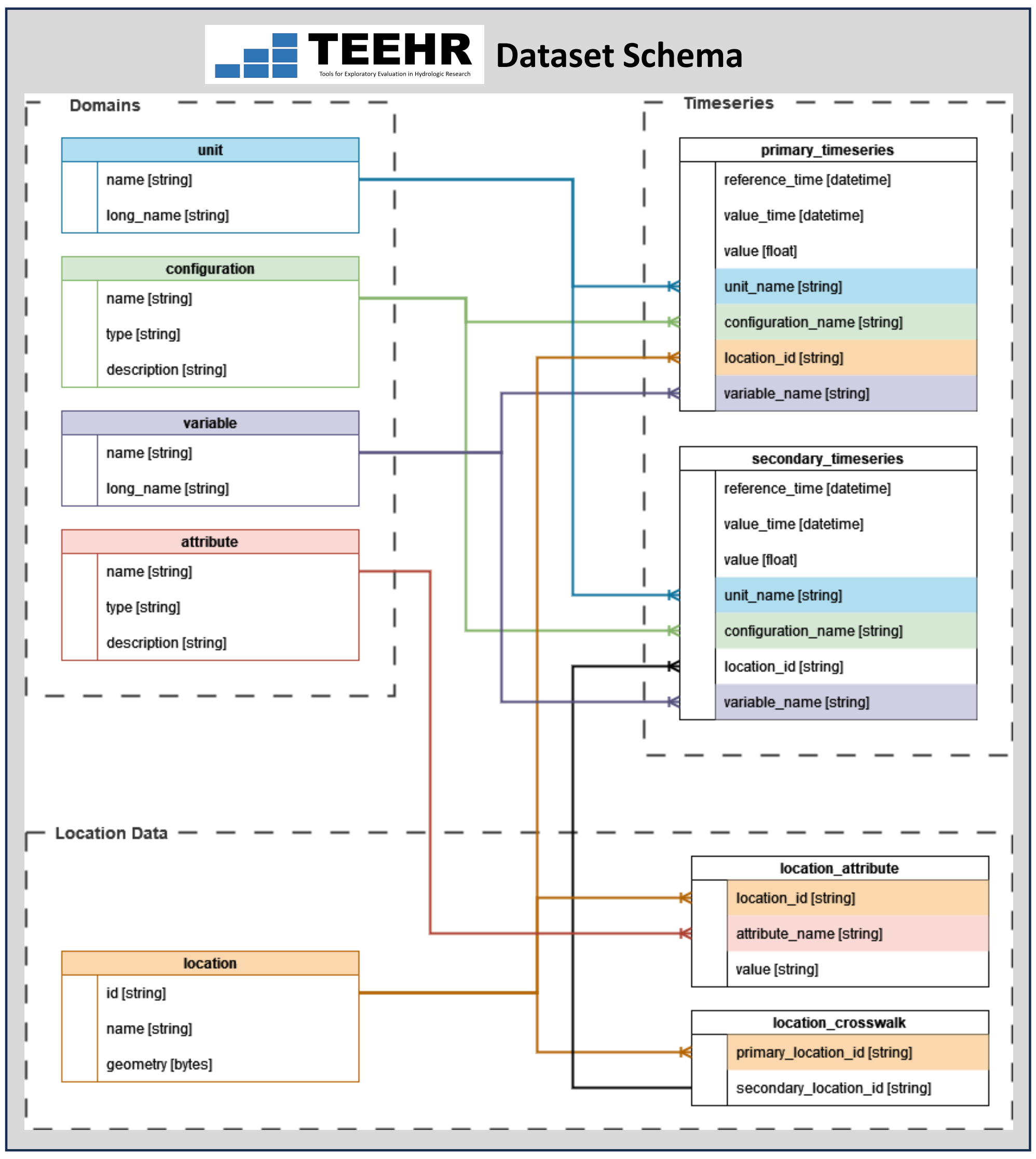
```
metrics_df = ev.metrics
.query(
    group_by=["configuration_name", "primary_location_id", "year"],
    filters=["primary_location_id = 'usgs-14138800'"],
    include_metrics=[
        teehr.Metrics.Maximum(
            input_field_names=["primary_value"],
            output_field_name="max_primary_value"
        ),
        teehr.Metrics.Maximum(
            input_field_names=["secondary_value"],
            output_field_name="max_secondary_value"
        )
    ]
)
.query(
    group_by=["configuration_name", "primary_location_id"],
    include_metrics=[
        teehr.Metrics.RelativeBias(
            input_field_names=["max_primary_value", "max_secondary_value"],
            output_field_name="annual_max_relative_bias"
        )
    ]
)
.to_pandas()
```

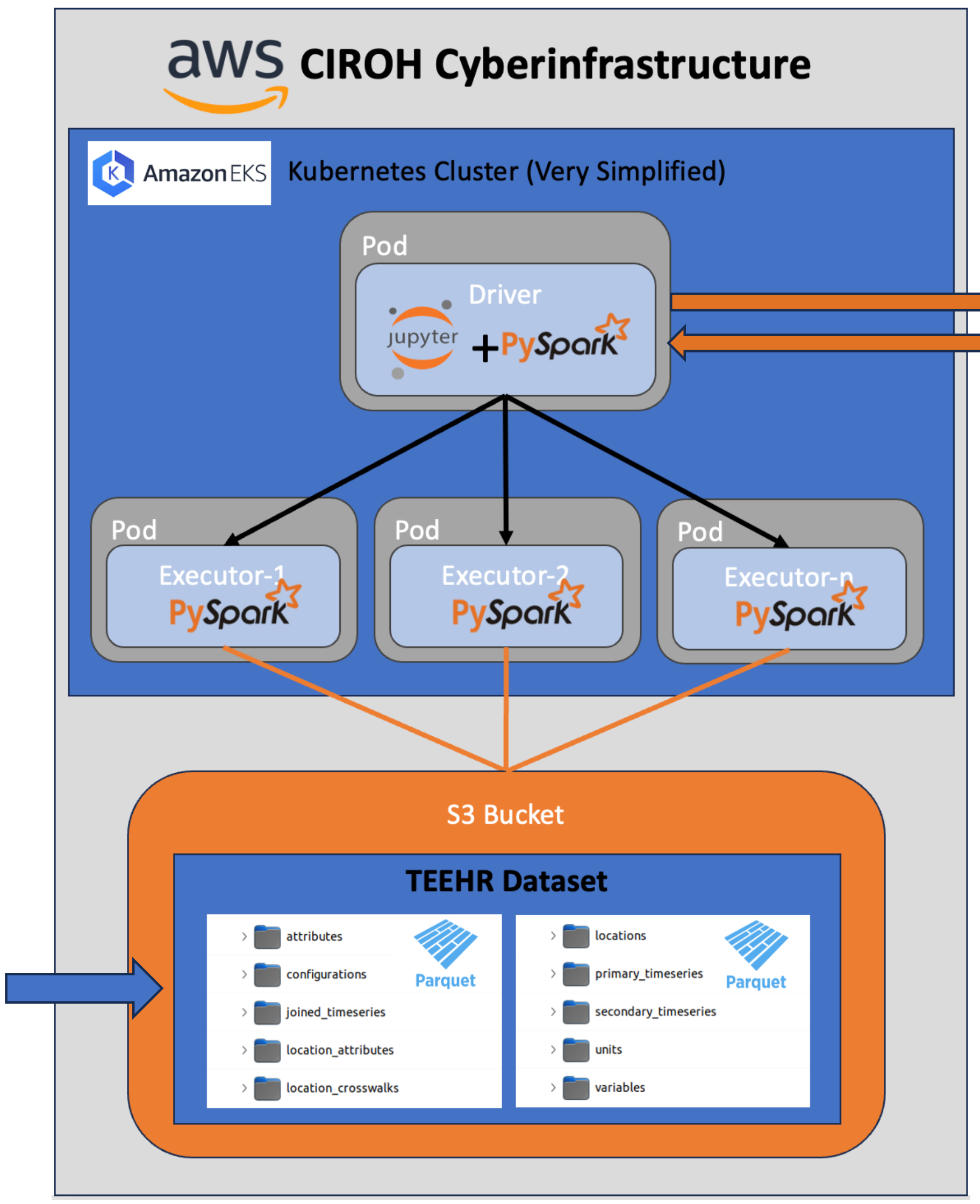
### Visualization

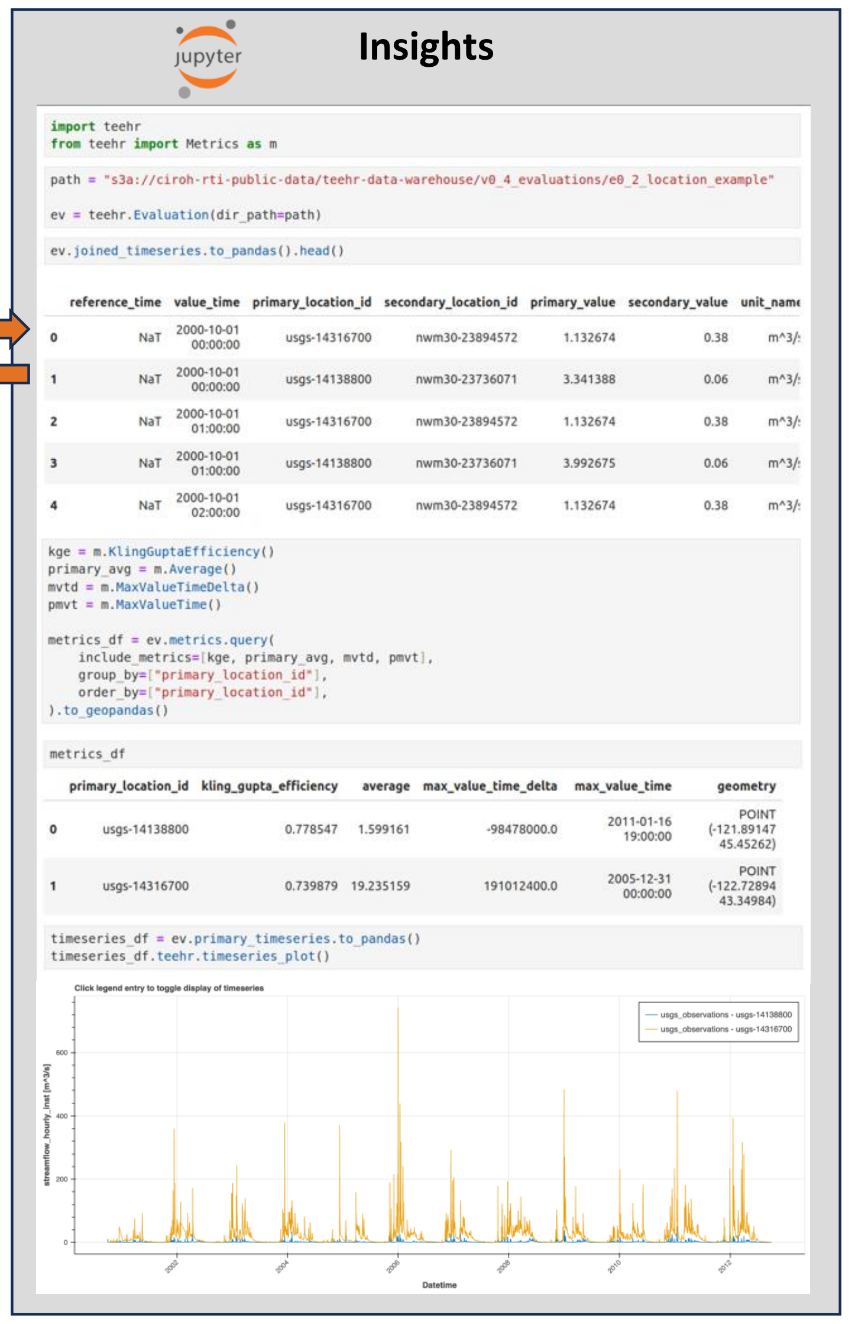
(*in development*) TEEHR provides both timeseries and metric visualization capabilities through use of a Pandas DataFrame Accessor.

```
ts_df.teehr.timeseries_plot()
```









```
import teehr
from teehr import Metrics as m
```

```
path = "s3a://ciroh-rti-public-data/teehr-data-warehouse/v0_4_evaluations/e0_2_location_example"

ev = teehr.Evaluation(dir_path=path)
```

```
ev.joined_timeseries.to_pandas().head()
```

| | reference_time | value_time | primary_location_id | secondary_location_id | primary_value | secondary_value | unit_name |
|---|---|---|---|---|---|---|---|
| 0 | NaT | 2000-10-01 00:00:00 | usgs-14316700 | nwm30-23894572 | 1.132674 | 0.38 | m^3/ |
| 1 | NaT | 2000-10-01 00:00:00 | usgs-14138800 | nwm30-23736071 | 3.341388 | 0.06 | m^3/ |
| 2 | NaT | 2000-10-01 01:00:00 | usgs-14316700 | nwm30-23894572 | 1.132674 | 0.38 | m^3/ |
| 3 | NaT | 2000-10-01 01:00:00 | usgs-14138800 | nwm30-23736071 | 3.992675 | 0.06 | m^3/ |
| 4 | NaT | 2000-10-01 02:00:00 | usgs-14316700 | nwm30-23894572 | 1.132674 | 0.38 | m^3/ |

```
kge = m.KlingGuptaEfficiency()
primary_avg = m.Average()
mvtd = m.MaxValueTimeDelta()
pmvt = m.MaxValueTime()

metrics_df = ev.metrics.query(
    include_metrics=[kge, primary_avg, mvtd, pmvt],
    group_by=["primary_location_id"],
    order_by=["primary_location_id"],
).to_geopandas()
```

```
metrics_df
```

| | primary_location_id | kling_gupta_efficiency | average | max_value_time_delta | max_value_time | geometry |
|---|---|---|---|---|---|---|
| 0 | usgs-14138800 | 0.778547 | 1.599161 | -98478000.0 | 2011-01-16 19:00:00 | POINT (-121.89147 45.45262) |
| 1 | usgs-14316700 | 0.739879 | 19.235159 | 191012400.0 | 2005-12-31 00:00:00 | POINT (-122.72894 43.34984) |

```
timeseries_df = ev.primary_timeseries.to_pandas()
timeseries_df.teehr.timeseries_plot()
```

Click legend entry to toggle display of timeseries
usgs_observations - usgs-14138800
usgs_observations - usgs-14316700
streamflow_hourly_inst [m^3/s]
Datetime
0
200
400
600
2002
2004
2006
2008
2010
2012

https://github.com/RTIInternational/teehr

TEEHR
Tools for Exploratory Evaluation in Hydrologic Research

## Case Study: Hourly CONUS NWM v3.0 Retrospective Evaluation

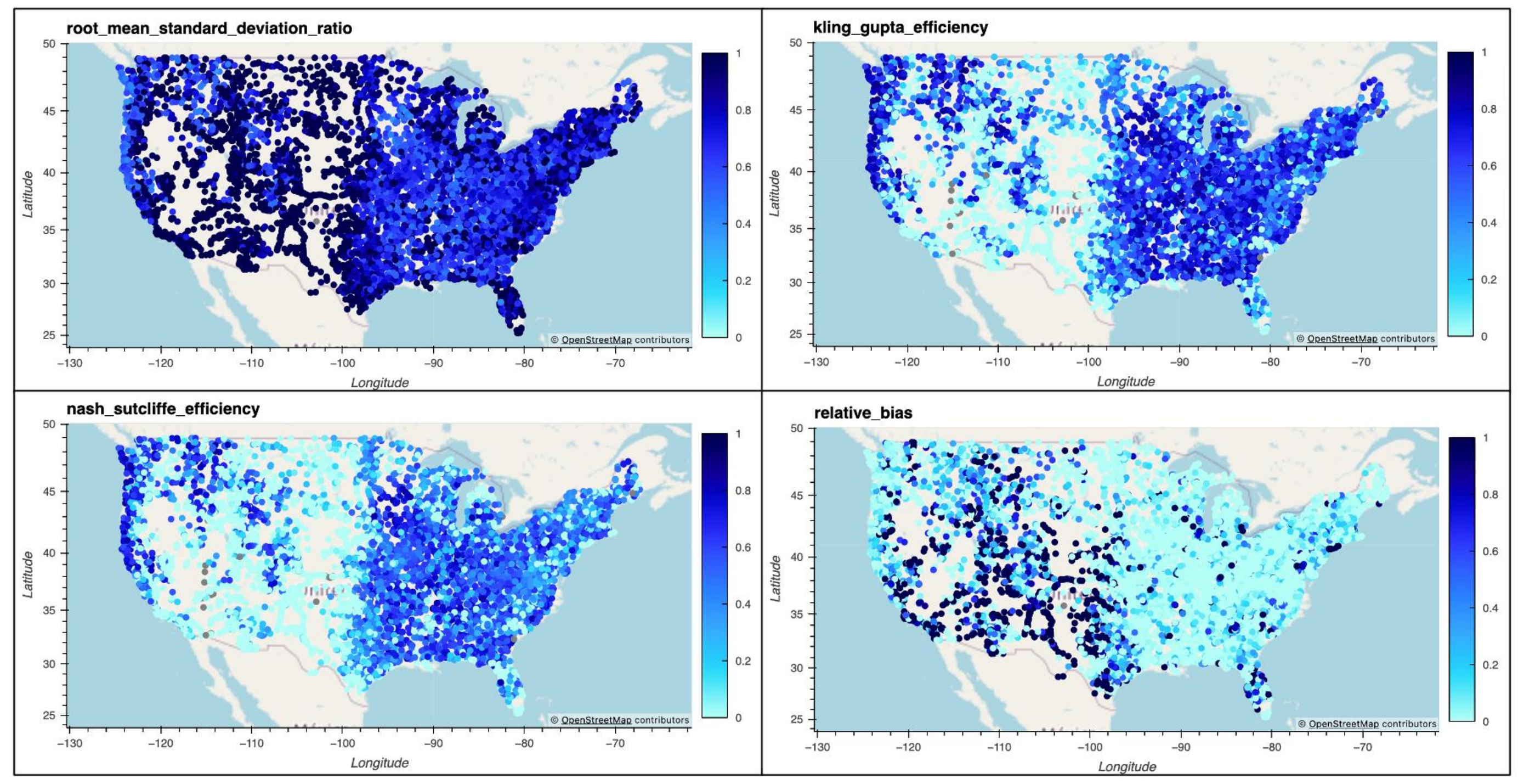
To demonstrate the power of TEEHR for conducting continental-scale evaluations we setup an evaluation of the hourly National Water Model v3.0 Retrospective Simulation data and ran several queries against the dataset to demonstrate the power of TEEHR and the underlying data structure (Apache Parquet) and the scalable compute engine (PySpark). The TEEHR dataset is stored in an AWS S3 bucket and queried using TEEHR running in a Spark cluster in AWS EKS (Kubernetes).

- Hourly National Water Model v3.0 Retrospective Simulation Data
- 8,132 USGS gages
- From 1980-01-01 00:00:00 to 2023-12-31 23:00:00 (44 years)
- 1.56 Billion joined (“paired”) timeseries records

### Query 1: 4 Comparative Metrics at CONUS USGS Gages

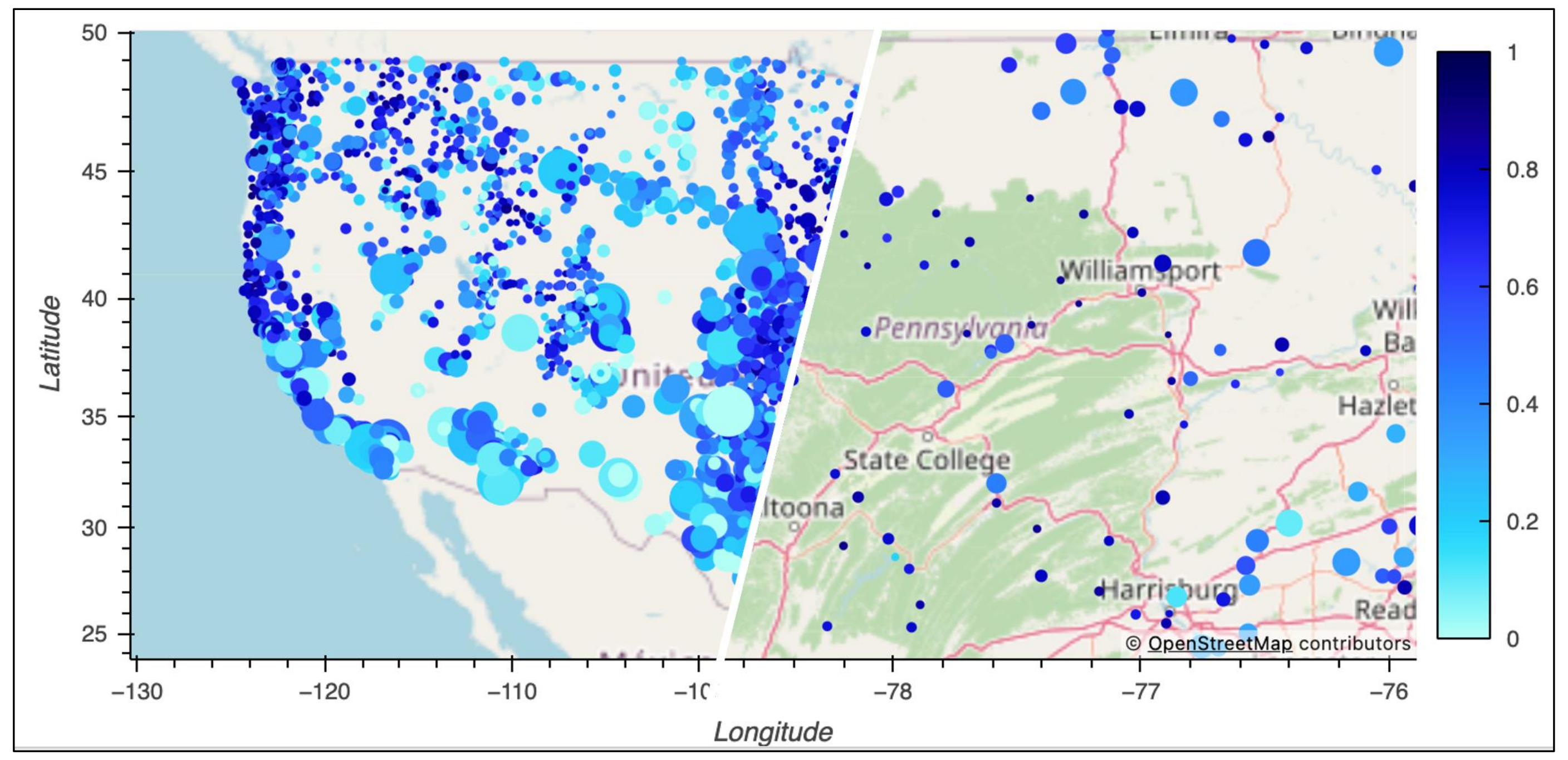
Query 1 calculated 4 standard comparative metrics at each of the USGS gages comparing the simulated NWM v3.0 to the USGS gage observations.
**Execution Time: ~1 minute**

| | configuration_name | primary_location_id | nash_sutcliffe_efficiency | kling_gupta_efficiency | relative_bias | root_mean_standard_deviation_ratio |
|---|---|---|---|---|---|---|
| 0 | nwm30_retrospective | usgs-01038000 | 0.686707 | 0.709544 | -0.058372 | 0.559726 |
| 1 | nwm30_retrospective | usgs-01401000 | 0.170434 | 0.185527 | -0.133946 | 0.910805 |
| 2 | nwm30_retrospective | usgs-01514850 | 0.656467 | 0.747603 | 0.170950 | 0.586117 |
| 3 | nwm30_retrospective | usgs-01643500 | 0.193687 | 0.434383 | -0.009543 | 0.897949 |
| 4 | nwm30_retrospective | usgs-02326526 | -10.448203 | -0.273122 | -0.907320 | 3.383519 |



### Query 2: CONUS-wide KGE with Bootstrapped Uncertainty Estimation

Query 2 calculated KGE with Bootstrapped Uncertainty at each of the USGS gages. A circular block bootstrap was used with a block size of 365 days and 500 repetitions. In the figure, the symbol color represents KGE based on total population, and symbol size represents the bootstrap confidence interval (0.05 to 0.95) – i.e., bigger has higher uncertainty.
**Execution Time: ~30 minutes**



### More Information

**Presenting author**: Matthew Denno
mdenno@rti.org
www.rti.org