Operational Forecasting of Induced Seismicity (ORION) Toolkit

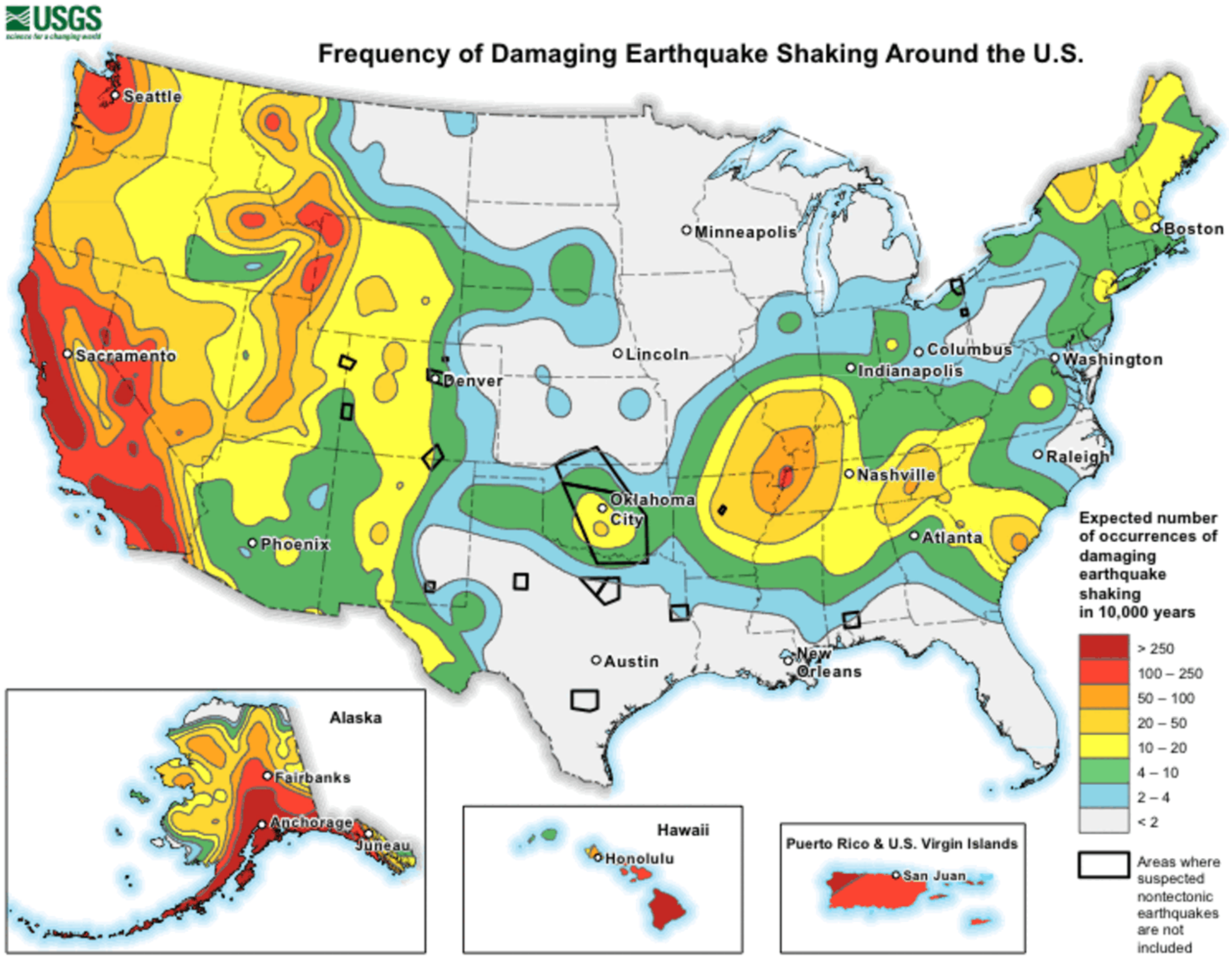
Background and Application to the Illinois Basin Decatur Project

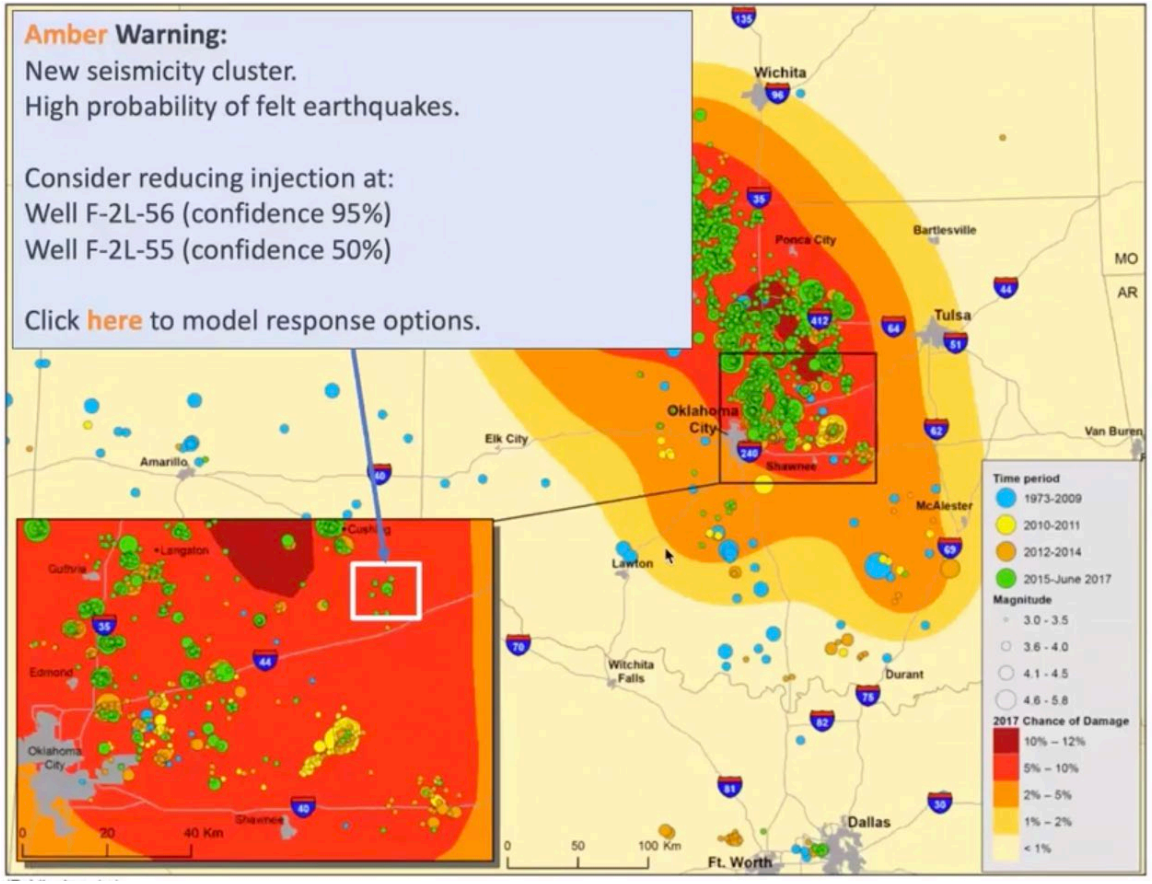
Lead Developers: Kayla Kroll[1] and Christopher Sherman[1]
Co-developers: Gina Geffers[1], Chaoyi Wang[1], Keurfon Luu[2]

[1]Lawrence Livermore National Laboratory
[2]Lawrence Berkeley National Laboratory

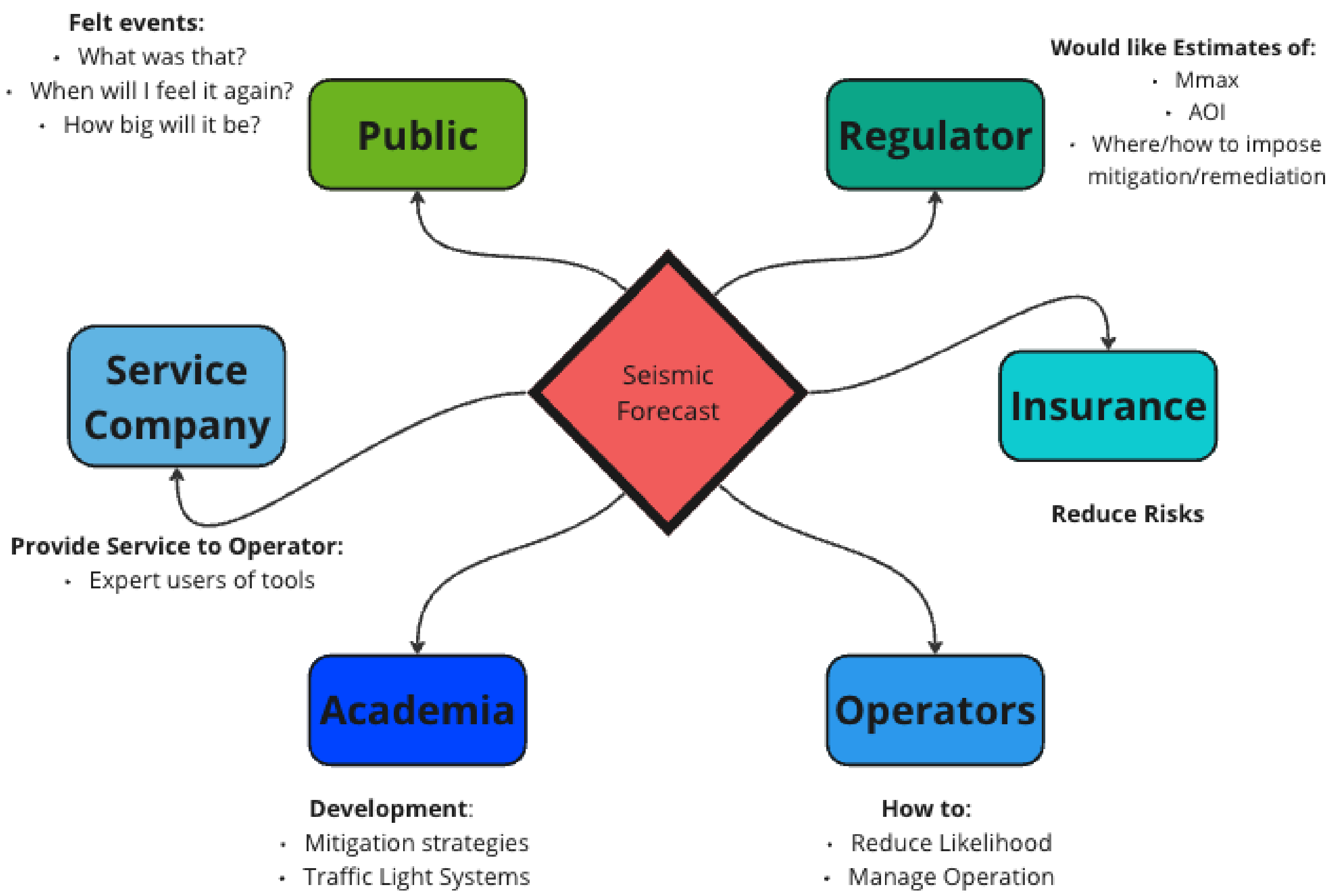
U.S. DEPARTMENT OF ENERGY

Seismic Forecasting

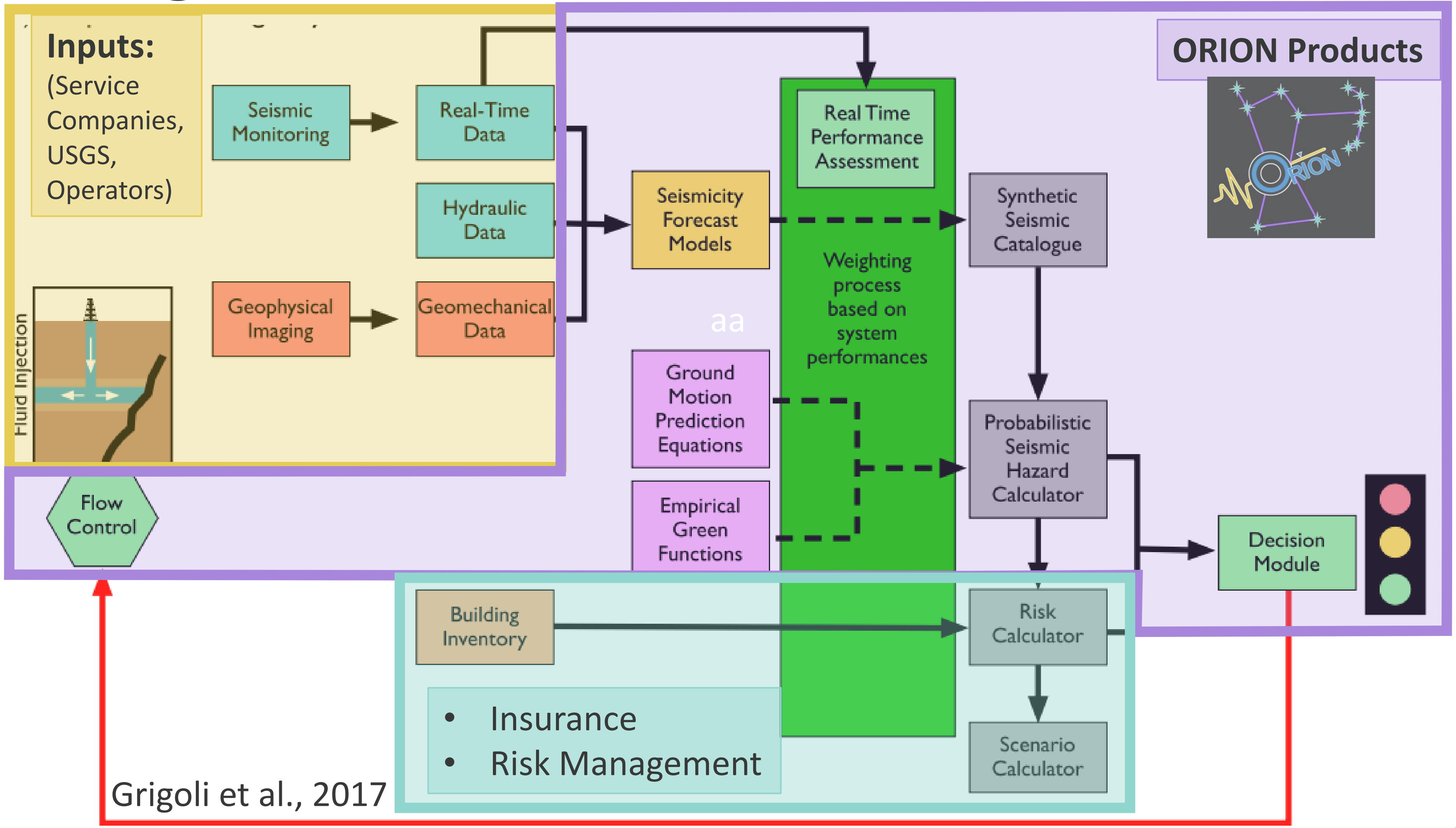




Forecast Stakeholders

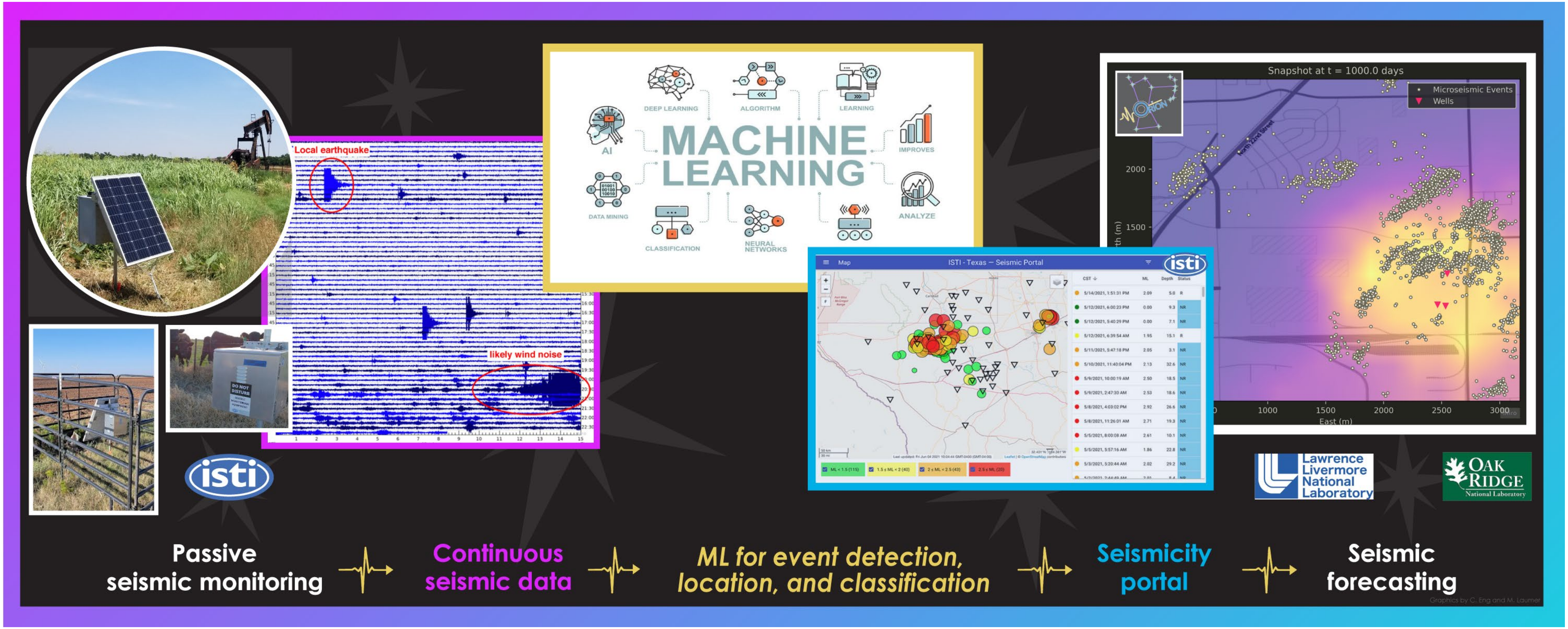


Forecasting Requirements



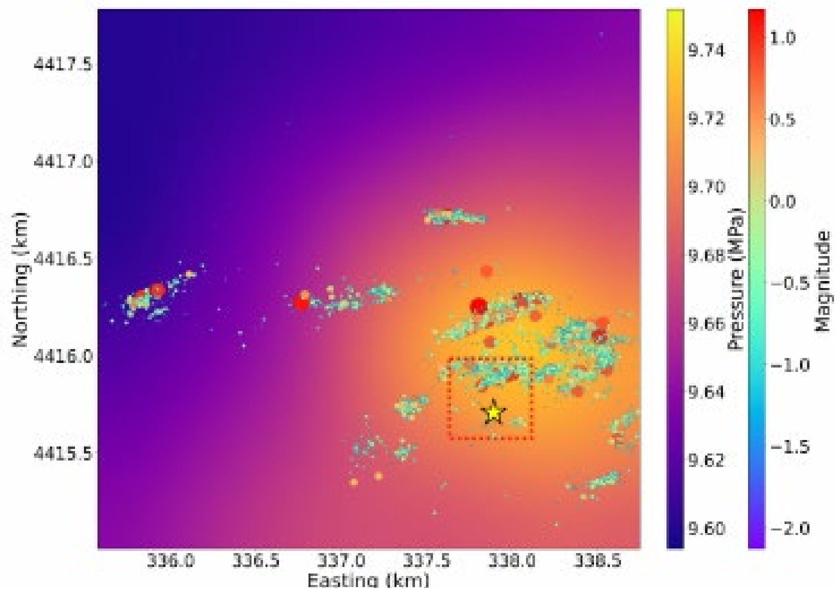
Grigoli et al., 2017

Required Input: Seismic Catalog



Required Input: Pressure Model

- **Pre-calculated pressure table**
  - ORION can support a variety of formats (using the USM as a converter), but prefers hdf5-format files
- **ML/ROM Pressure Model**
  - Support through USM/other tools
- **Radial flow model (Theis)**
  - Assumes an infinite homogeneous half-space
  - Requires constant or time-varying flow rate data into wells



The fluid pressure, p, is given by the superposition of each well contribution:

$$p = \frac{\rho g}{4piT}\left(z + \sum_{i=1}^{N} q_i W(u_i)\right) \qquad u_i = \frac{r_i^2 s}{4T(t - t_i)}$$

Ensemble of Induced Seismicity Forecasting Models

Physics-based Model

Coupled Coulomb Rate-State Model

(Dieterich, 1994; Kroll et al., 2017)

$$R = \exp\left(\frac{\Delta CFS}{a\sigma_{eff}}\right) \quad \text{Coseismic}$$

$$R(t) = \frac{1}{\frac{1}{\dot{S}f_r} + \left(\frac{1}{R_0} - \frac{1}{\dot{S}f_r}\right)\exp\left(\frac{-\dot{S}t}{a\sigma_{\text{eff}}}\right)};\ f_r = \frac{r}{\dot{S}_r} \quad \text{Interseismic}$$

Rate-State ODE

(Dieterich, 1994; Segall and Lu, 2015)

$$\frac{dR}{dt} = \frac{R}{t_a}\left(\frac{\dot{S}}{\dot{S}_0} - R\right)$$

*Declustered

Statistical Models

ETAS Models (Ogata, 1988; Bachmann et al., 2013)

$$\lambda_i(t) = \rho(M_i)\Phi(t - t_i)$$

$$\lambda_0(t) = \mu + c_f Q$$

$$\lambda(t) = \lambda_0' + \sum_{\{i;\, t_i<t\}} \lambda_i(t)$$

Gutenberg-Richter Law:

$$\rho(M_i) = k10^{\alpha(M-M_0)}$$

Omori's Law:

$$\Phi(\Delta_t) = \frac{k}{(t+c)^p}$$

Seismogenic Index (Shapiro et al., 2010)

$$SI = \log_{10}N - \log_{10}\sum \dot{S} + bM$$

$$R = \dot{S}^2\, 10^{SI - bM}$$

Ensemble of Induced Seismicity Forecasting Models

Physics-based Model

$$\dot{S} = \dot{\tau} + \mu(\dot{\sigma} - \dot{p})$$

Statistical Models

Coupled Coulomb Rate-State Model
(Dieterich, 1994; Kroll et al., 2017)

$$R = \exp\left(\frac{\Delta CFS}{a\sigma_{eff}}\right)$$ Coseismic

$$R(t) = \frac{1}{\frac{1}{\dot{S}f_r} + \left(\frac{1}{R_0} - \frac{1}{\dot{S}f_r}\right)\exp\left(\frac{-\dot{S}t}{a\sigma_{\text{eff}}}\right)};\quad f_r = \frac{r}{\dot{S}_r}$$ Interseismic

Rate-State ODE
(Dieterich, 1994; Segall and Lu, 2015)

$$\frac{dR}{dt} = \frac{R}{t_a}\left(\frac{\dot{S}}{\dot{S}_0} - R\right)$$

*Declustered

ETAS Models (Ogata, 1988; Bachmann et al., 2013)

$$\lambda_i(t) = \rho(M_i)\Phi(t - t_i)$$

$$\lambda_0(t) = \mu + c_f Q$$

$$\lambda(t) = \lambda_0' + \sum_{\{i;\, t_i<t\}} \lambda_i(t)$$

Gutenberg-Richter Law:

$$\rho(M_i) = k10^{\alpha(M-M_0)}$$

Omori's Law:

$$\Phi(\Delta_t) = \frac{k}{(t+c)^p}$$

Seismogenic Index (Shapiro et al., 2010)

$$SI = \log_{10}N - \log_{10}\sum \dot{S} + bM$$

$$R = \dot{S}^2\, 10^{SI - bM}$$

Ensemble of Induced Seismicity Forecasting Models

Physics-based Model

Coupled Coulomb Rate-State Model
(Dieterich, 1994; Kroll et al., 2017)

$$R = \exp\left(\frac{\Delta CFS}{a\sigma_{eff}}\right)$$ Clustering (EQ interaction)

$$R(t) = \frac{1}{\frac{1}{\dot{S}f_r} + \left(\frac{1}{R_0} - \frac{1}{\dot{S}f_r}\right)\exp\left(\frac{-\dot{S}t}{a\sigma_{\text{eff}}}\right)};\ f_r = \frac{r}{\dot{S}_r}$$ Interseismic

Rate-State ODE
(Dieterich, 1994; Segall and Lu, 2015)

$$\frac{dR}{dt} = \frac{R}{t_a}\left(\frac{\dot{S}}{\dot{S}_0} - R\right)$$

*Declustered

Statistical Models

ETAS Models (Ogata, 1988; Bachmann et al., 2013)

$$\lambda_i(t) = \rho(M_i)\Phi(t - t_i)$$

$$\lambda_0(t) = \mu + c_f Q$$

$$\lambda(t) = \lambda_0' + \sum_{\{i;\, t_i<t\}} \lambda_i(t)$$

Gutenberg-Richter Law:

$$\rho(M_i) = k10^{\alpha(M-M_0)}$$

Omori's Law:

$$\Phi(\Delta_t) = \frac{k}{(t+c)^p}$$

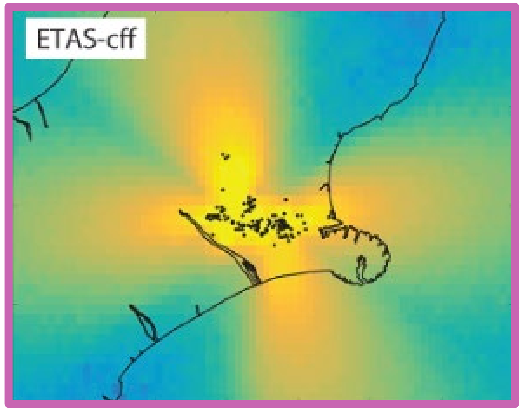
Seismogenic Index (Shapiro et al., 2010)

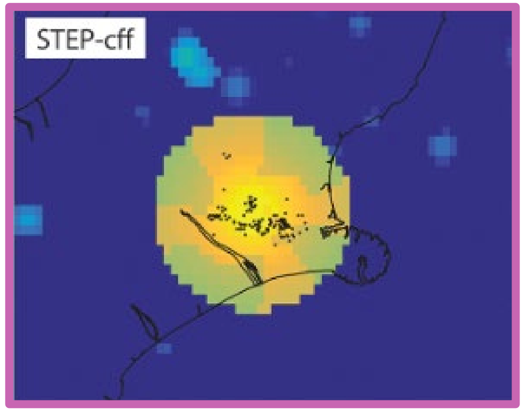
$$SI = \log_{10}N - \log_{10}\sum \dot{S} + bM$$

$$R = \dot{S}^2\, 10^{SI - bM}$$

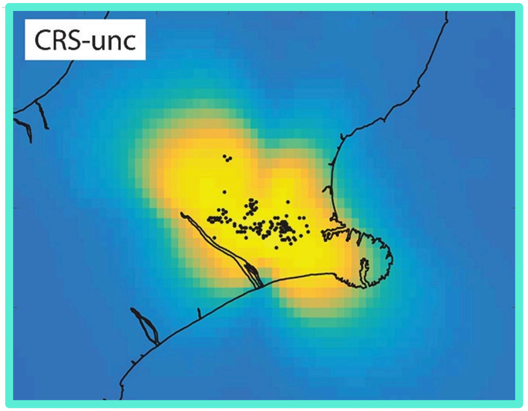
Various Induced Seismicity Forecasting Models

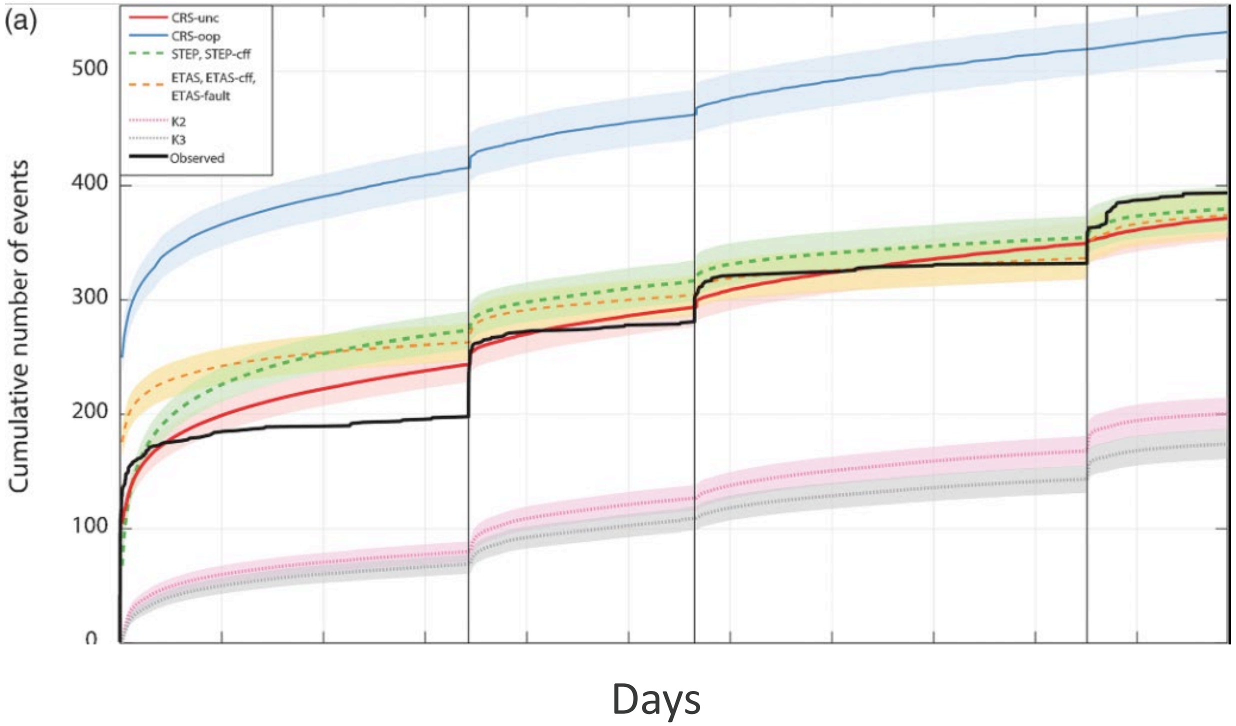
Statistical Models





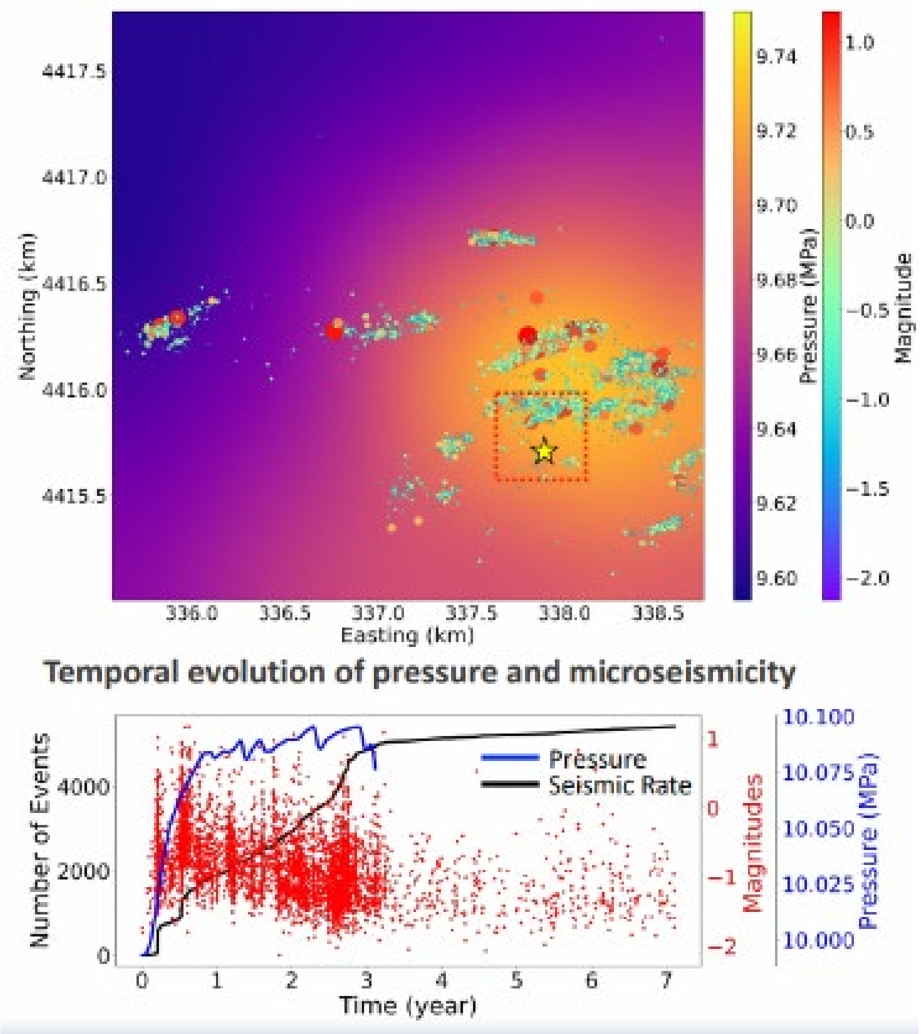
Physics-based Model



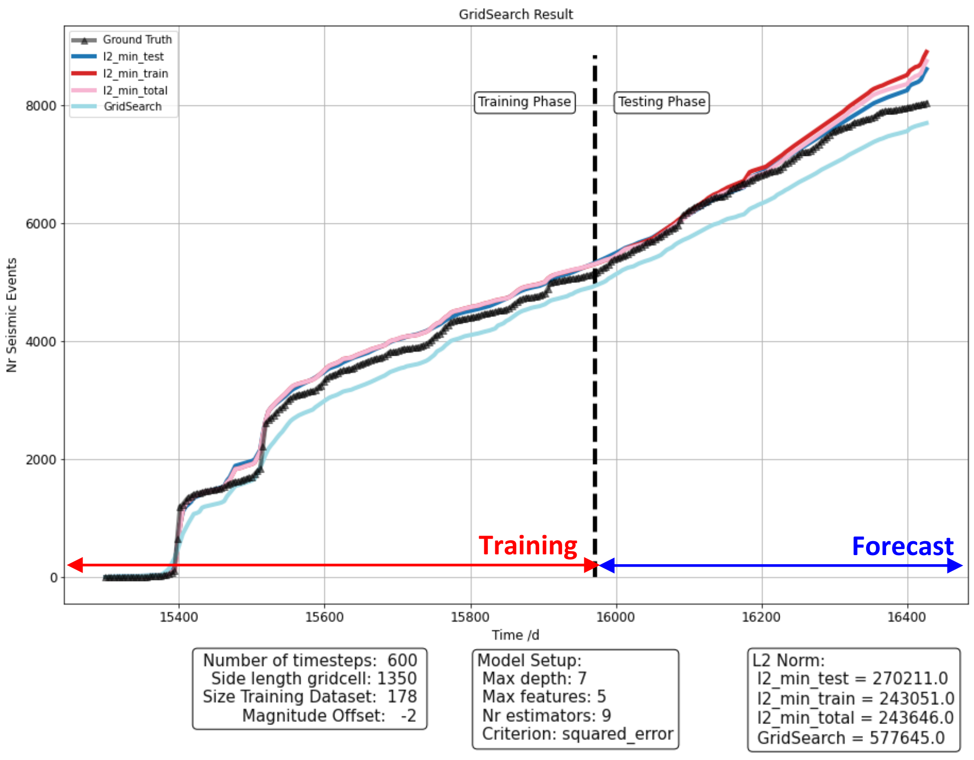


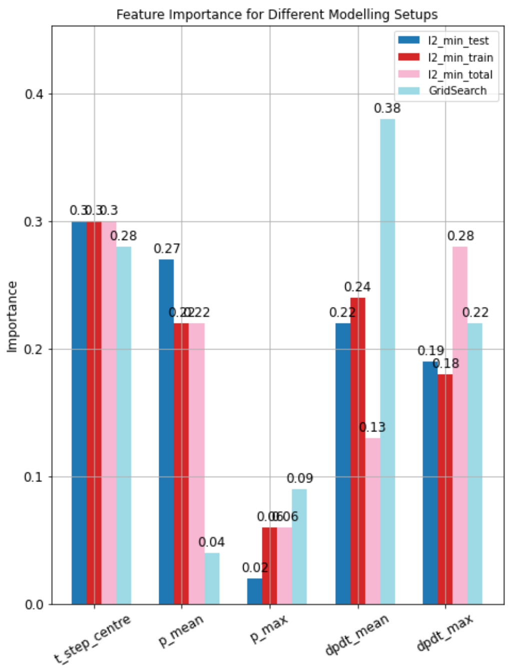
Cattania et al., 2018

ML-based Forecasts



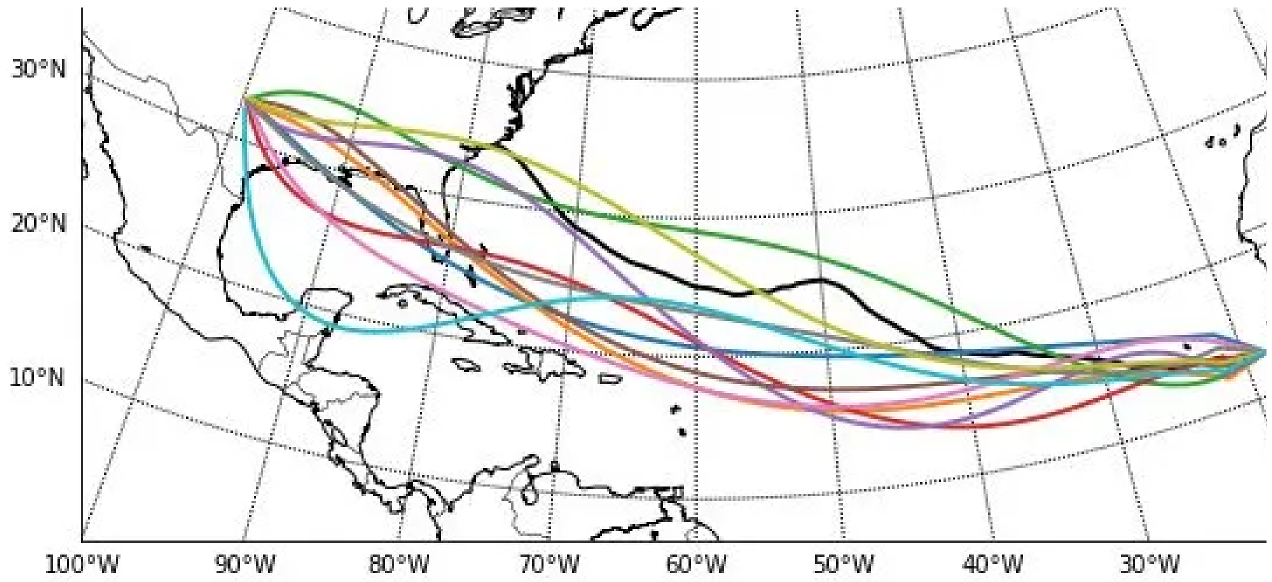
Random Forest

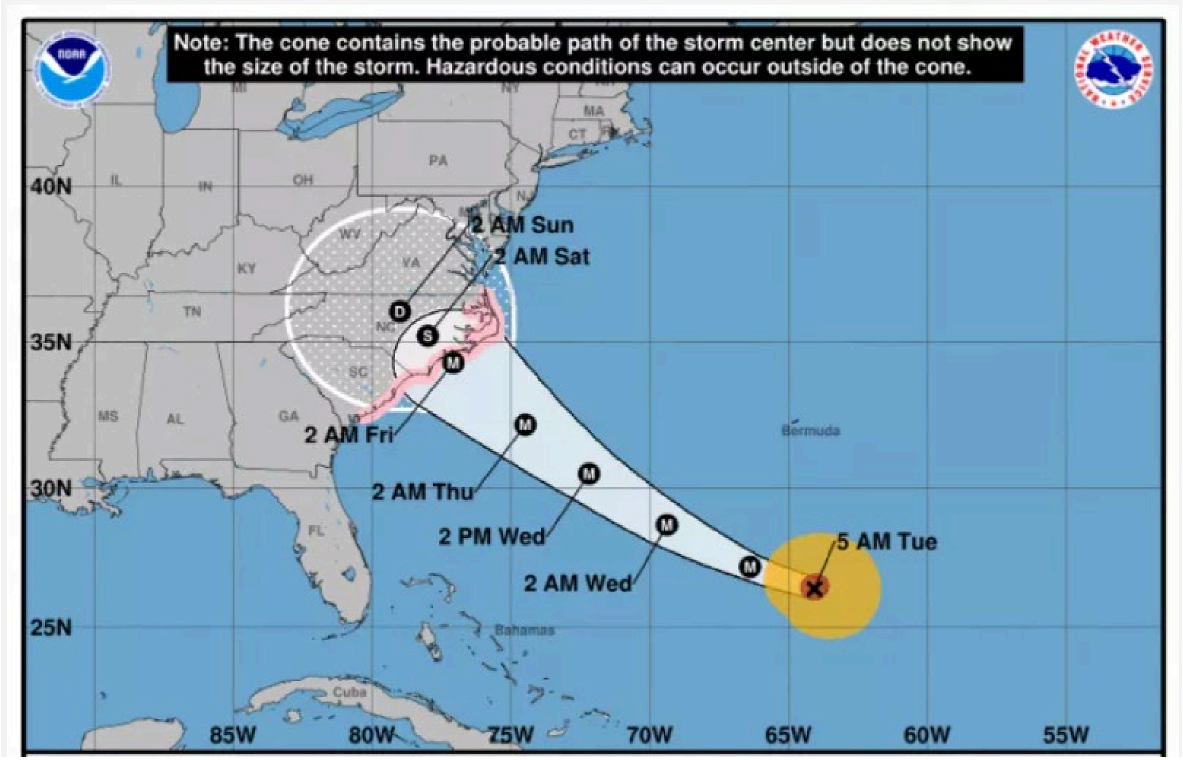




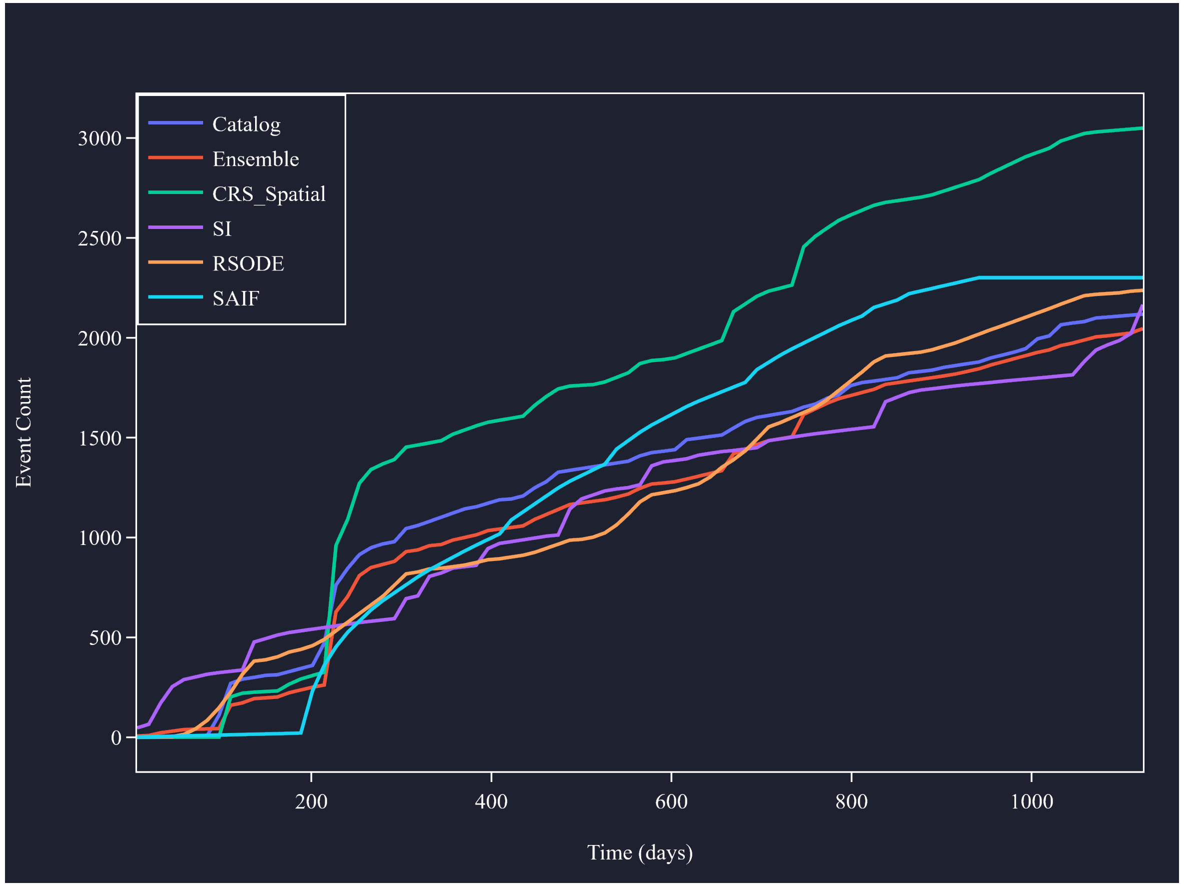
Walter et al., 2023

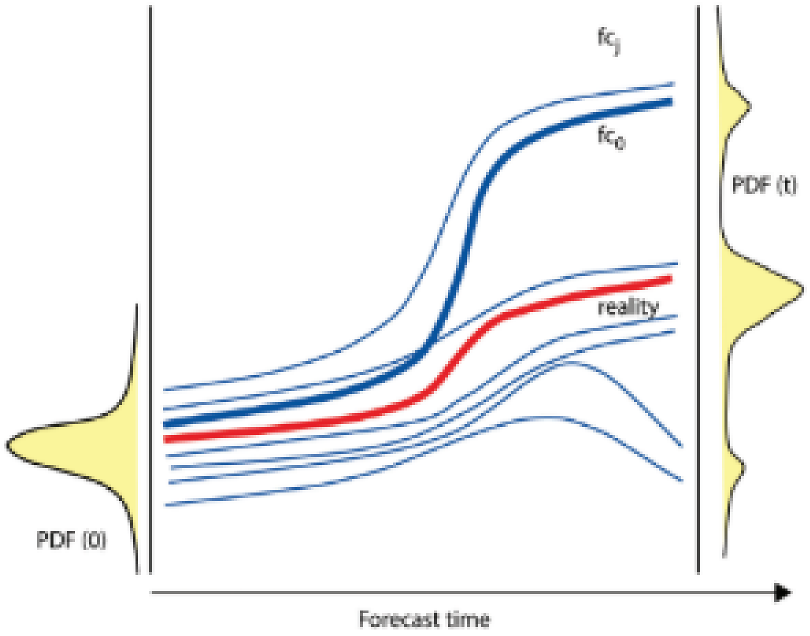
Ensemble Forecasting Approach





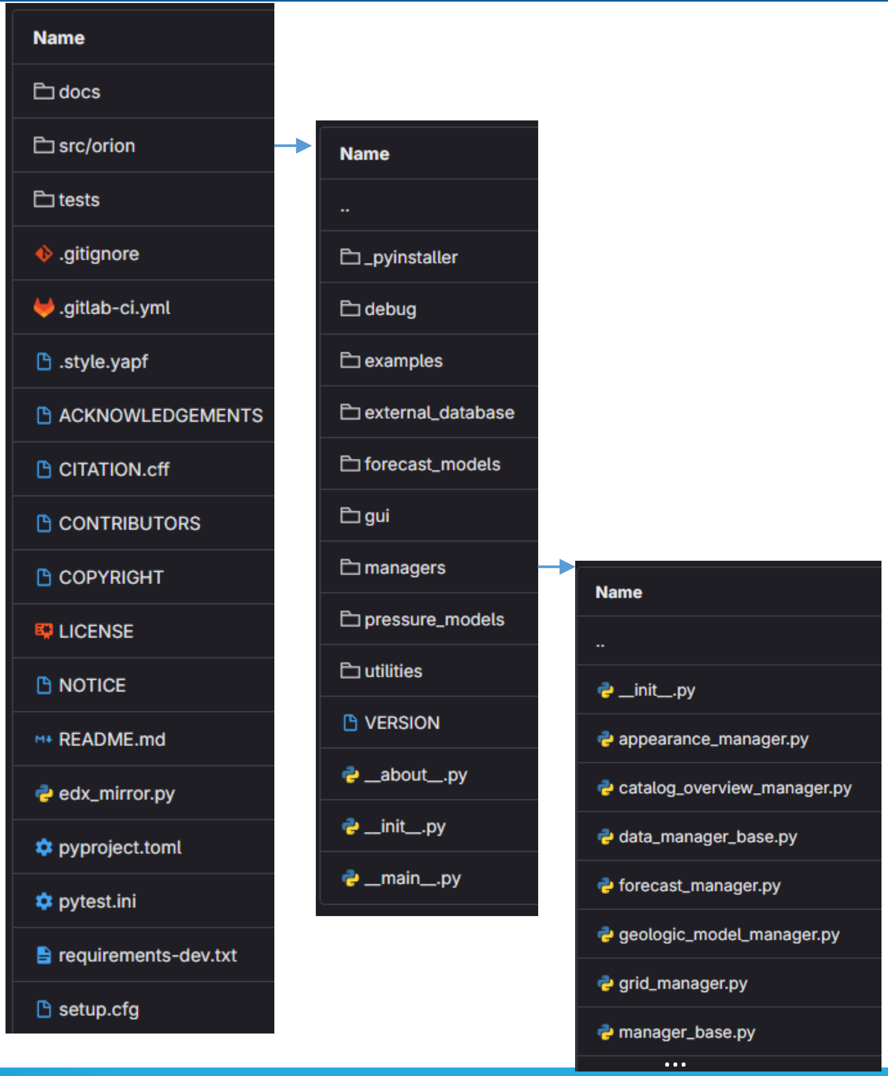
Ensemble Forecasting Approach





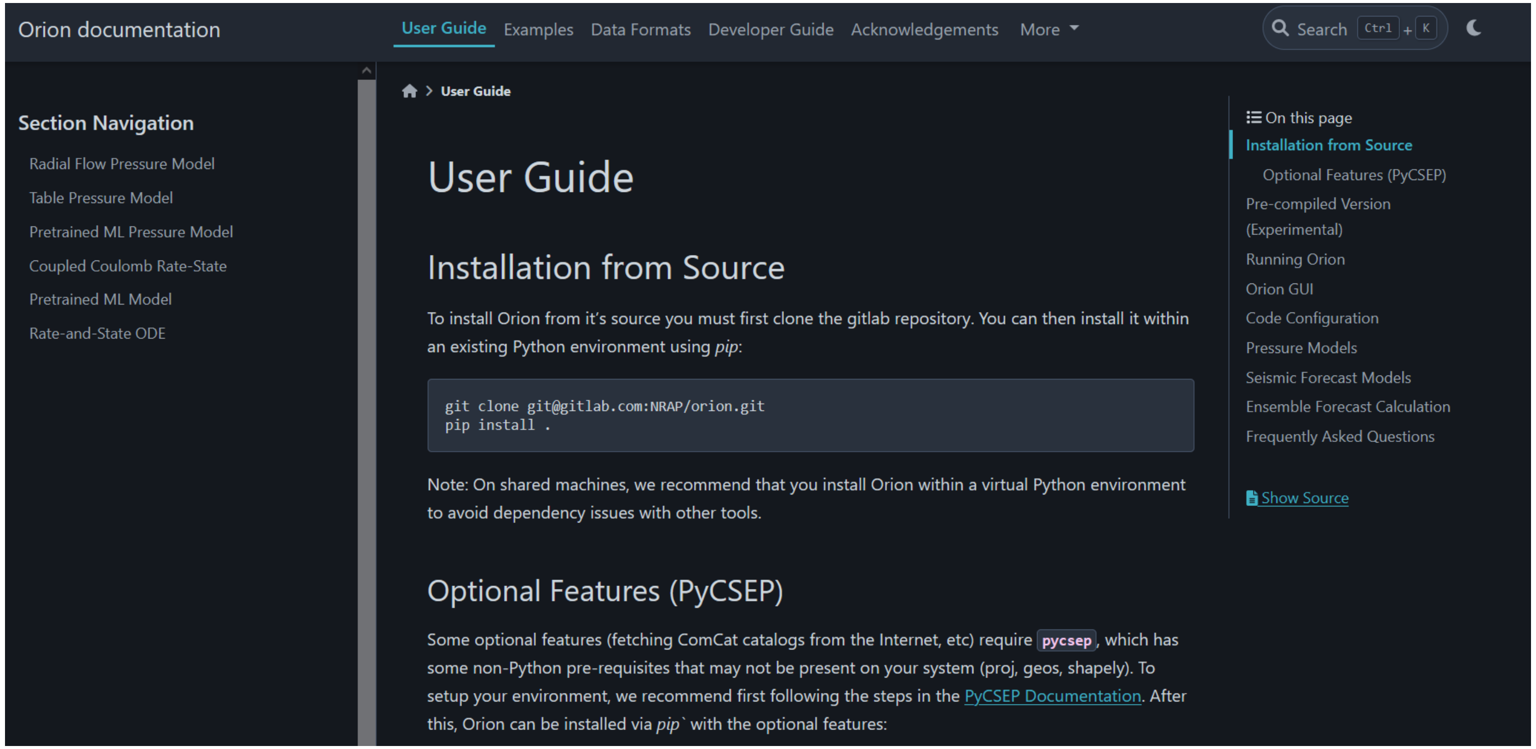
ORION Code Structure

- **ORION is organized as a pure-python, pip-installable package**
- **Hosted on Gitlab:**
  - https://gitlab.com/NRAP/orion
- **Submodules contain tools for managing data, forecasting, visualization, etc.**
- **Code formatting/Structure guidelines:**
  - PEP8 standards, enforced using the YAPF tool
  - Developers are expected to use human-readable variables, modern object-oriented coding practices
  - Classes, functions, methods, etc. include google format docstrings



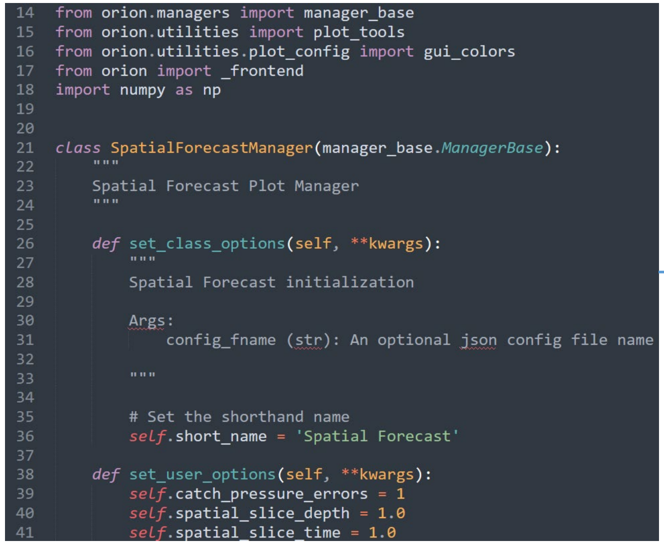
ORION Documentation

- **Documentation is written in .rst format, and built using Sphinx**
  - The build/hosting process is managed automatically by the Gitlab CI
  - https://nrap.gitlab.io/orion/user_guide.html
- **Components:**
  - User guide
  - Examples
  - Description of data formats
  - Developer guide
  - ORION API

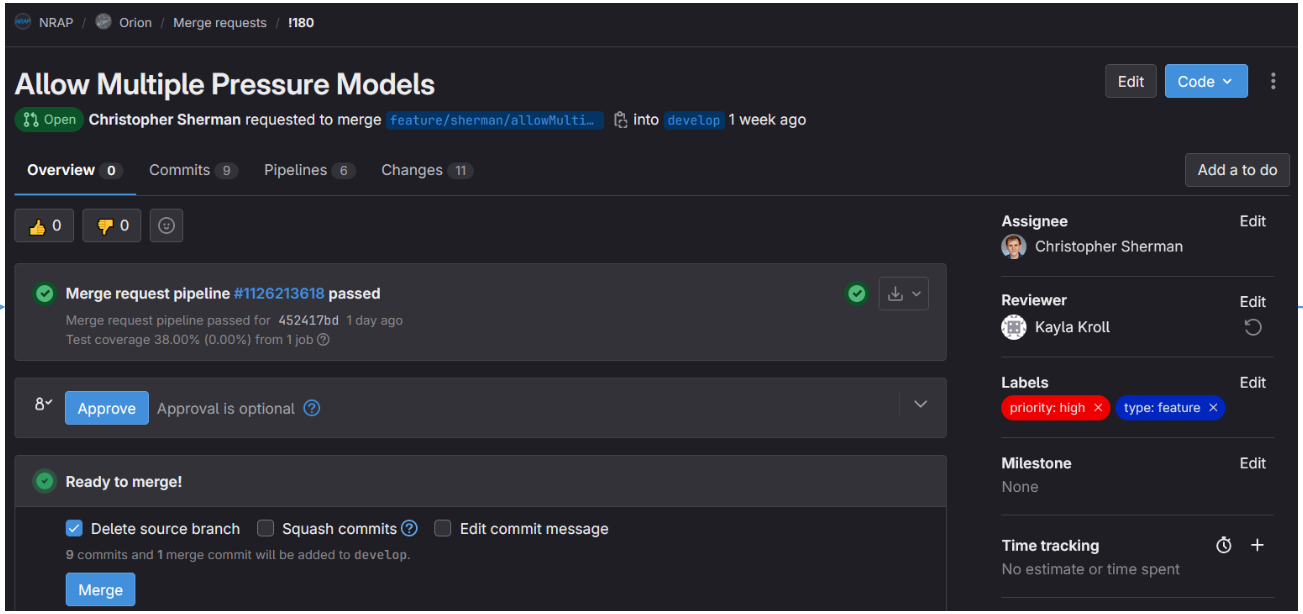


Version Control and Testing Pipeline

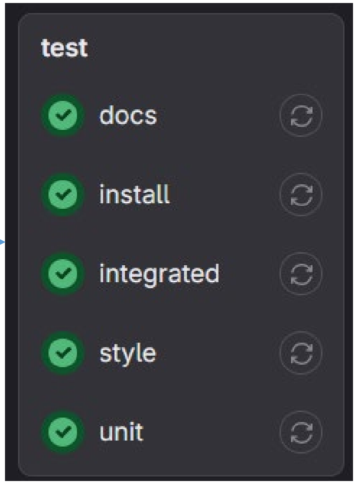
- Developers commit/push local code changes using git
- Merge Requests are used for code review, testing, record keeping, etc.
- The CI pipeline runs a variety of tests to ensure the quality of the code



Local code modification



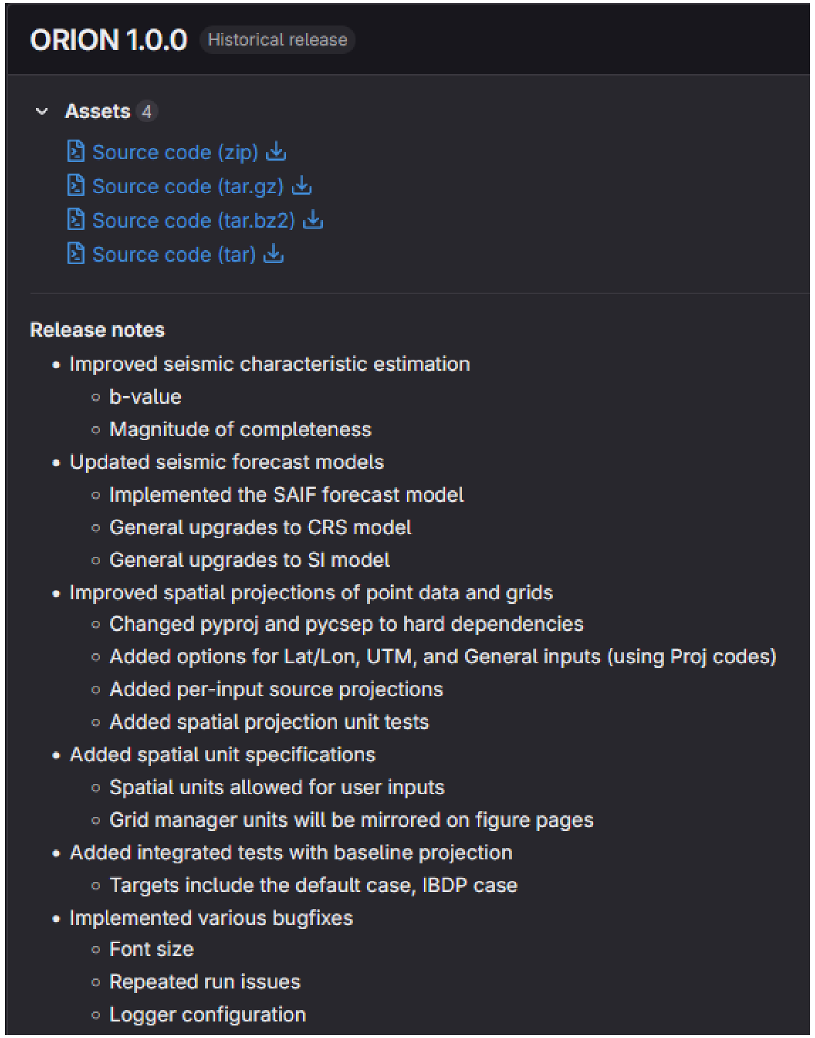
Gitlab Merge Request



CI Pipeline

ORION Distribution

- **Depending on a user's skill level / interest, they can work with ORION in a few different ways**
- **From easiest to hardest:**
  - Pre-compiled executable (built using pyinstaller, includes all of the necessary pre-requisites to run the code)
  - Install the package from a package manager (e.g. pypi)
  - Install the package from source (git)
- **Work in progress:**
  - For STRIVE-enabled ORION, users should simply need to visit a website



ORION 1.0.0 Historical release

Assets 4

- Source code (zip)
- Source code (tar.gz)
- Source code (tar.bz2)
- Source code (tar)

Release notes

- Improved seismic characteristic estimation
  - b-value
  - Magnitude of completeness
- Updated seismic forecast models
  - Implemented the SAIF forecast model
  - General upgrades to CRS model
  - General upgrades to SI model
- Improved spatial projections of point data and grids
  - Changed pyproj and pycsep to hard dependencies
  - Added options for Lat/Lon, UTM, and General inputs (using Proj codes)
  - Added per-input source projections
  - Added spatial projection unit tests
- Added spatial unit specifications
  - Spatial units allowed for user inputs
  - Grid manager units will be mirrored on figure pages
- Added integrated tests with baseline projection
  - Targets include the default case, IBDP case
- Implemented various bugfixes
  - Font size
  - Repeated run issues
  - Logger configuration

Basin Scale Spatio-temporal Forecasts

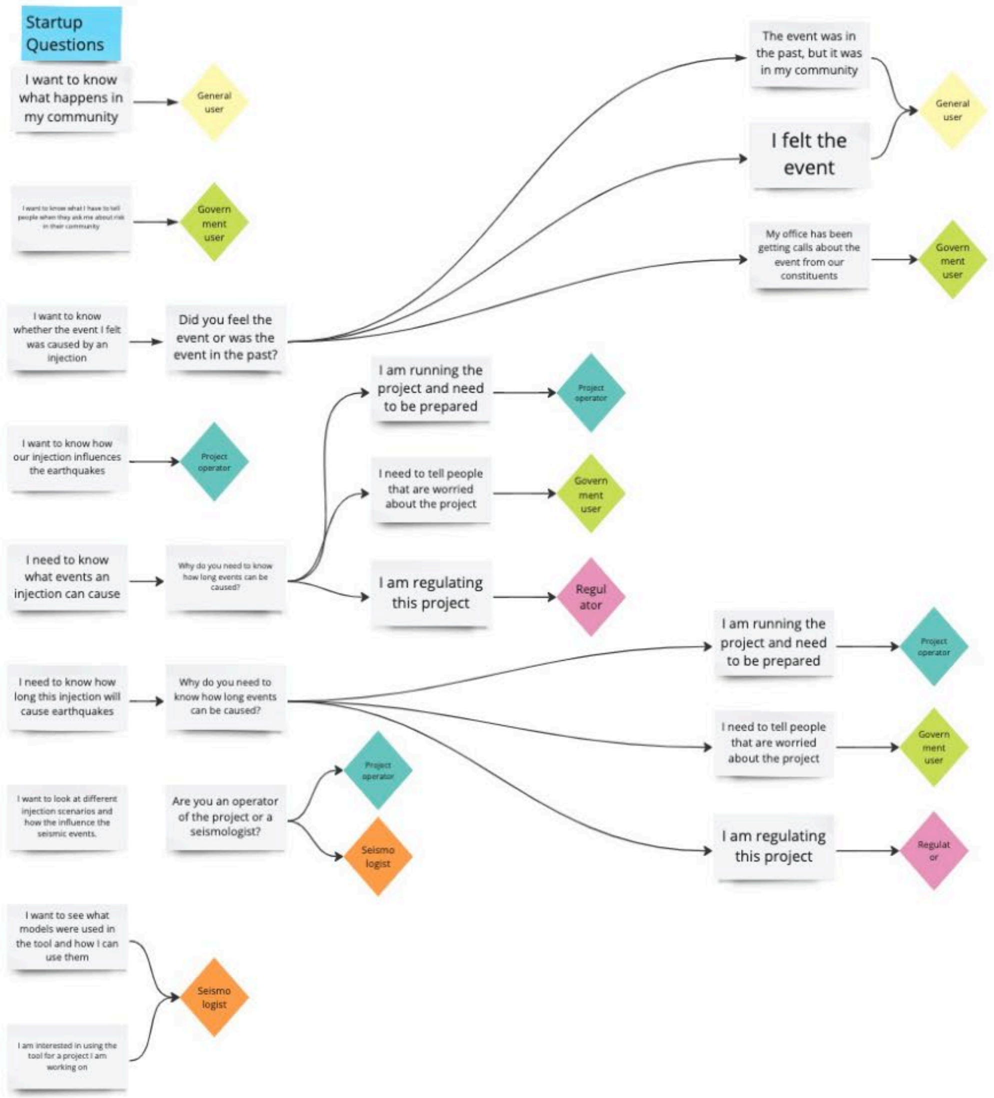
- Tailoring forecasting capabilities to various end-users:

Start up questionnaire

General user	Government user	Project operator	Seismologist	Regulator

miro

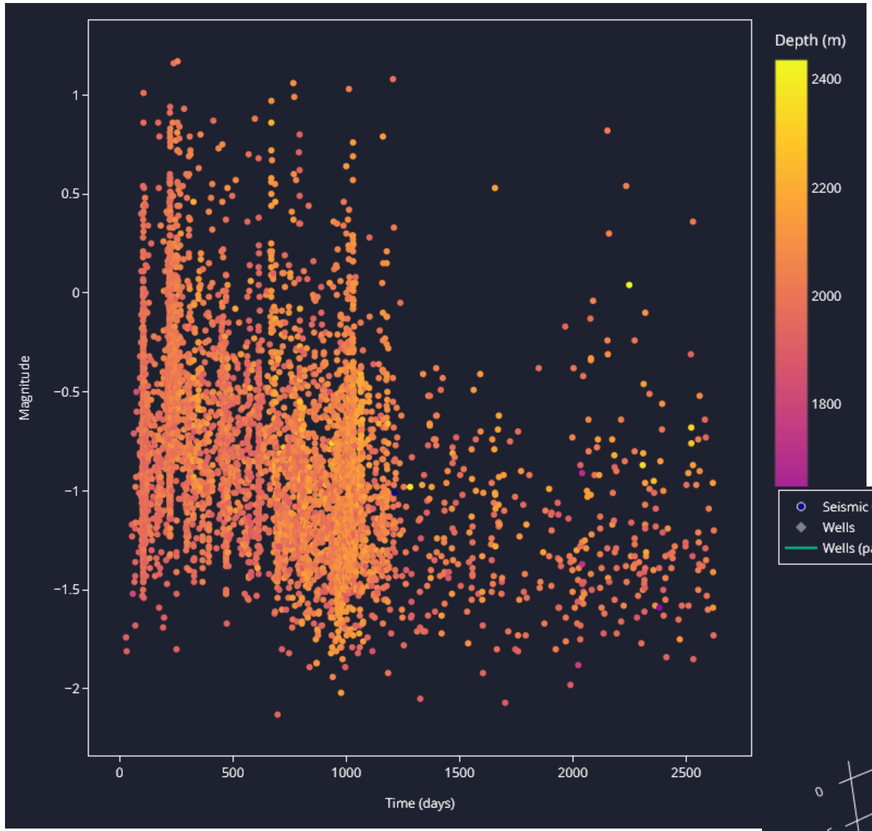
- Define user groups / stakeholders
  - Define access each group will have
  - Define use cases for each group
- Start with set of simple questions
  - Answer lead to different groups
    - Refine questions to get more specific / distinguish different groups
- Do not overcomplicate it
  - Max three to four layers

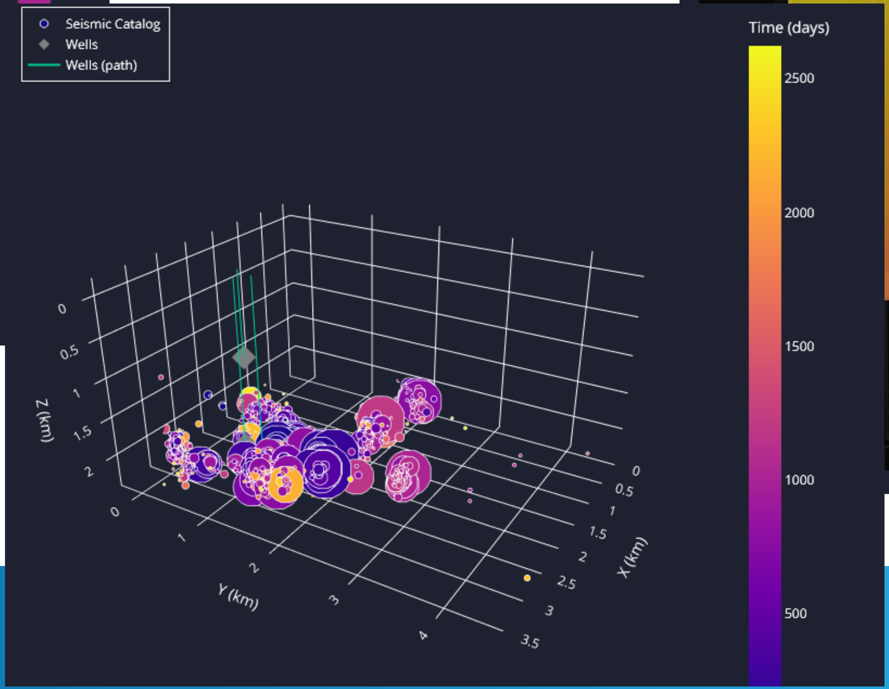


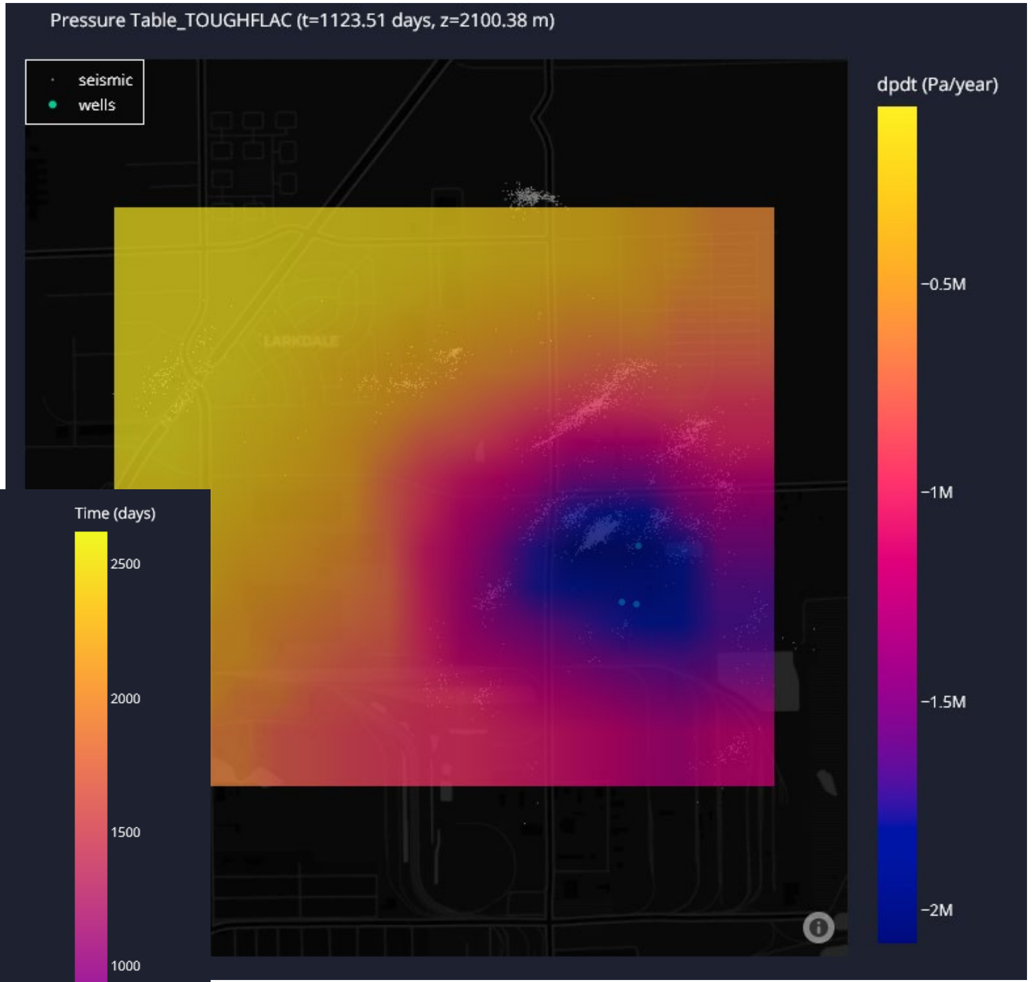
ORION Application Example: IBDP

Local seismic catalog, well information

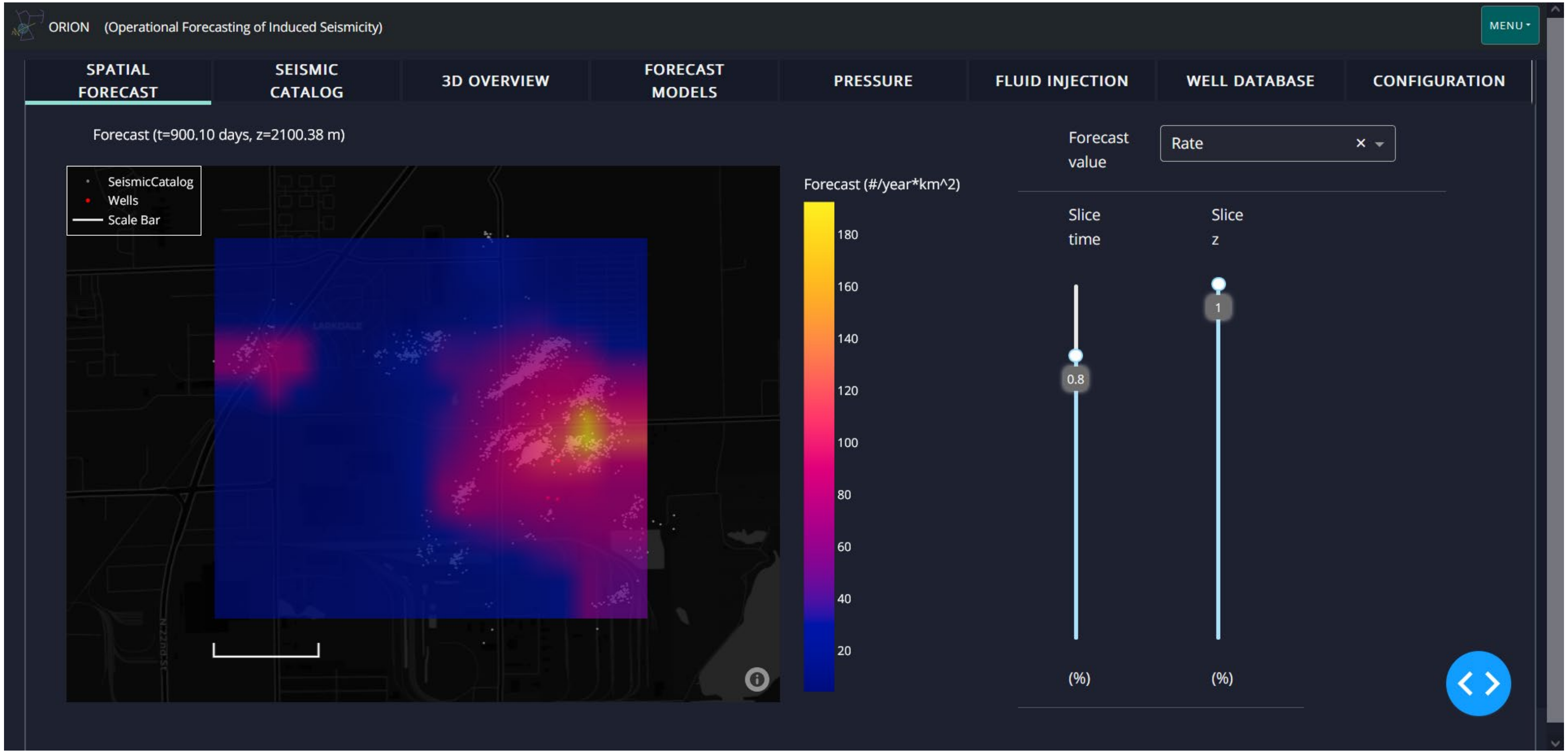
Pressure model (source: TOUGH+FLAC)



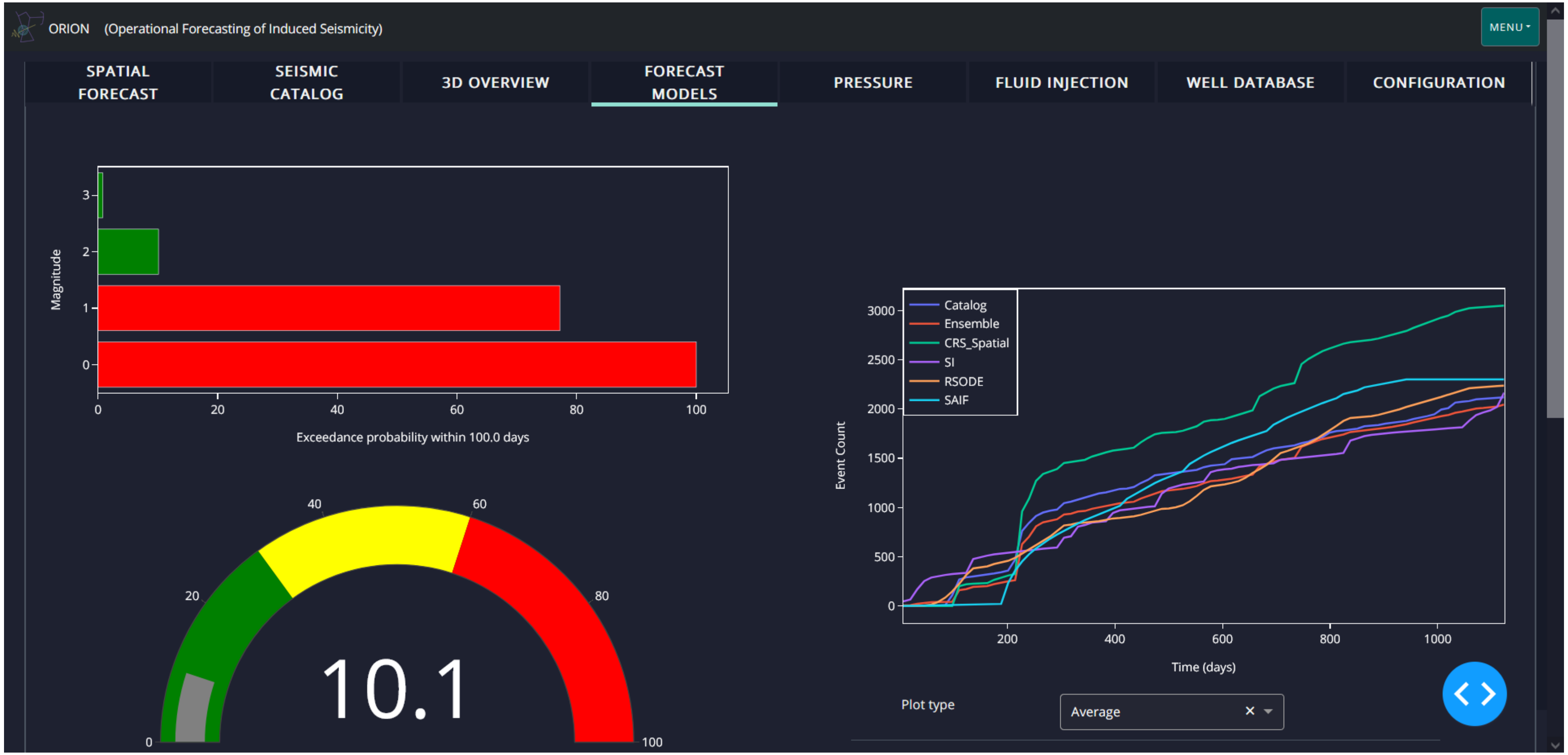




ORION Application Example: IBDP

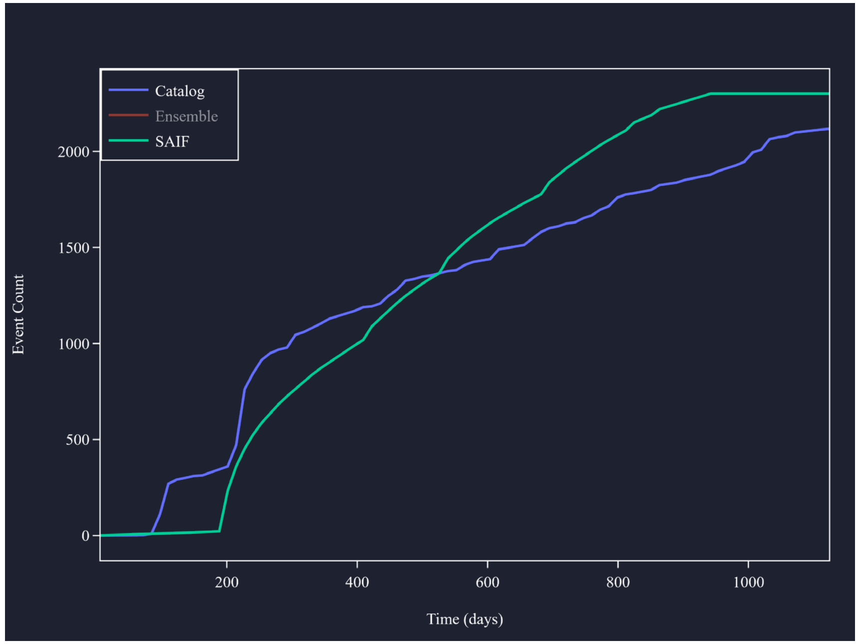


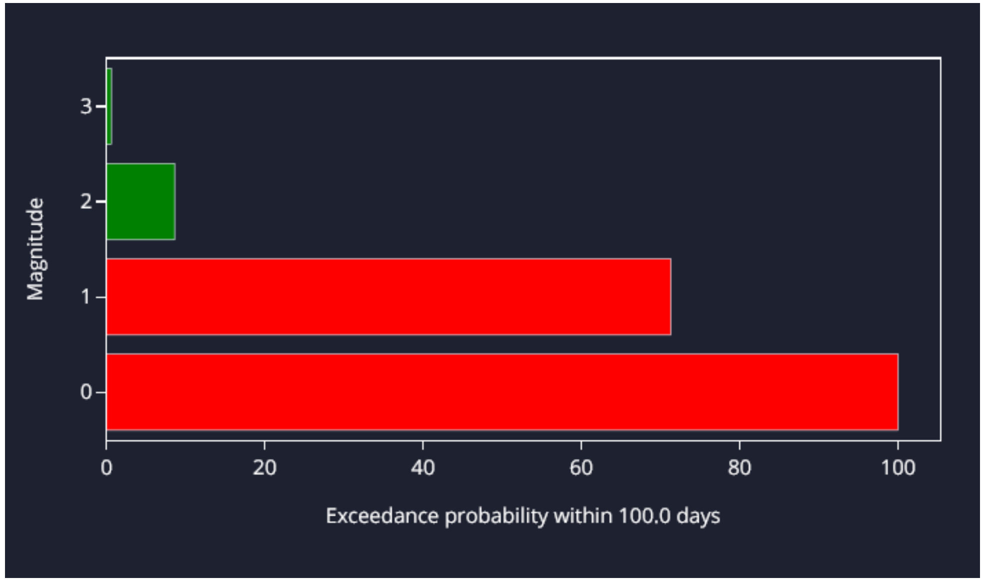
ORION Application Example: IBDP



ORION+SAIF

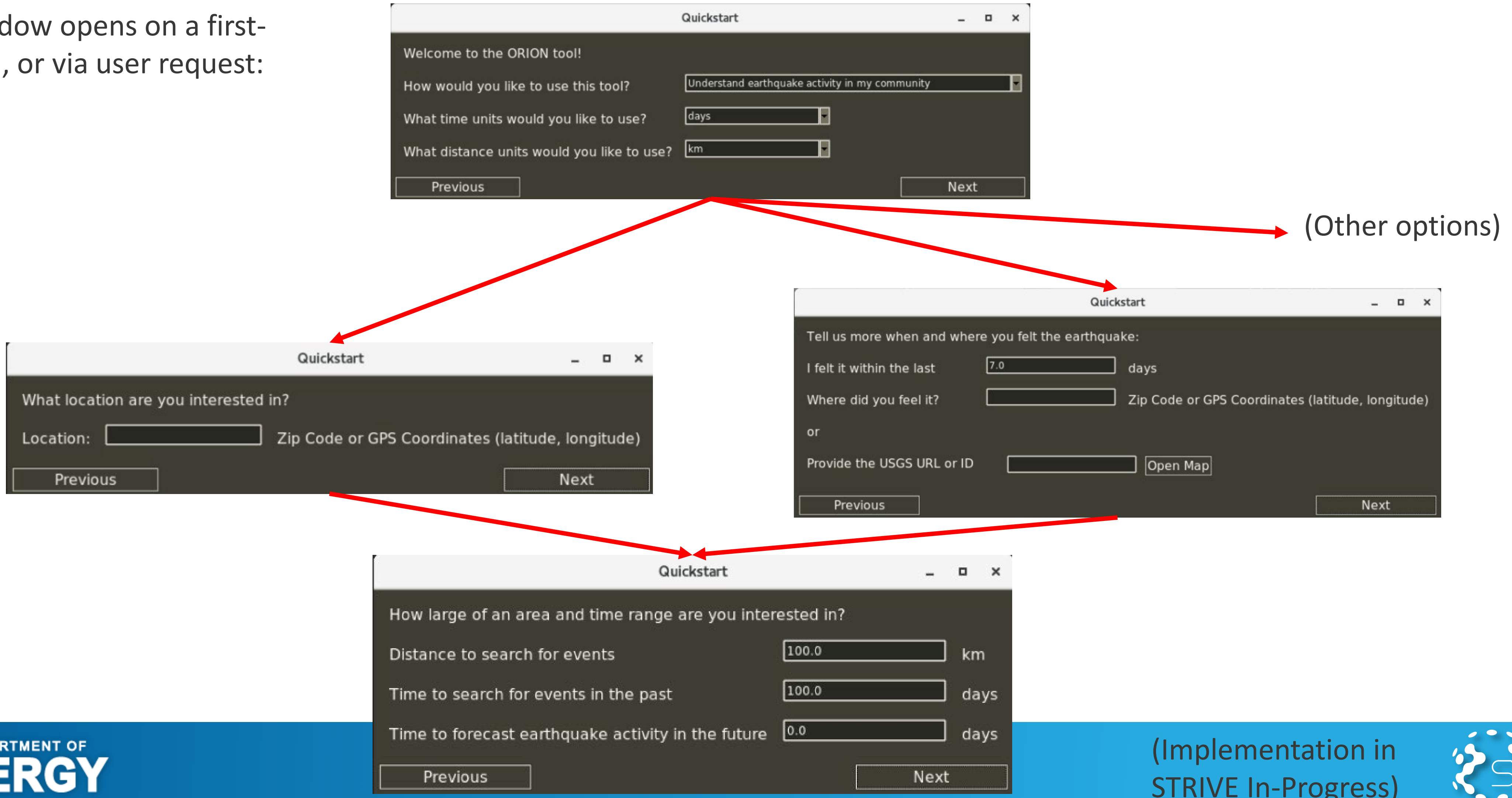
- Direct re-implementation of physics-based forecast model in pyTorch
- Extremely fast run time
- Enables automatic optimization of forecast parameters





ORION Startup Wizard

This window opens on a first-time run, or via user request:



(Implementation in STRIVE In-Progress)