Towards Accelerated CO_2 Inversion Modeling for Understanding Uncertainties through Multicore Processing and a Reusable Code Base

Sequoia Andrade*, Naomi Asimow**, Ron Cohen**, Ryan Giordano*
*UC Berkeley Department of Statistics
**UC Berkeley Department of Earth and Planetary Sciences

December 17th, 2025
A32C: Data-Driven Methods for Quantifying Atmospheric Composition: Advances in Computation and Statistical Learning

Motivation: Measuring Emissions

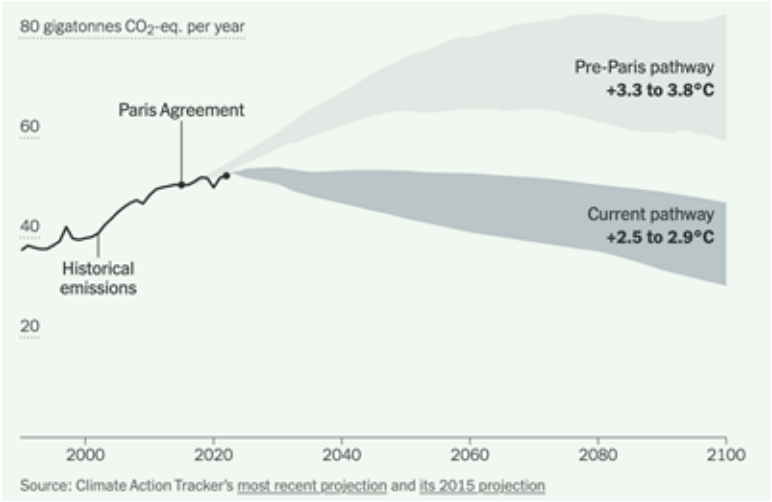
Why:

- Need to quantify the status carbon from anthropogenic sources, or "**flux**"
- Need to quantify any reductions in carbon for determine if emissions goals are feasible

How:

- **Bottom-up**: reporting-based inventories
- **Top-down**: measurement-based
  - E.g. In-Situ sensor networks, satellite observations

Figure 1: relation between warming and CO_2 [1]



We use a **top-down** approach while leveraging a **bottom-up** inventory as a prior

BEACO$_2$N Sensor Network [2]

- Berkeley Environmental Air Quality and CO_2 Network
- 2km network of ground-based sensors clustered over city centers
- Collects information on 6 gasses and particulate matter
- Has been used to measure sustained emissions trends [3] and COVID-19 trends [4]
- **Sensors are not where the emissions are!**

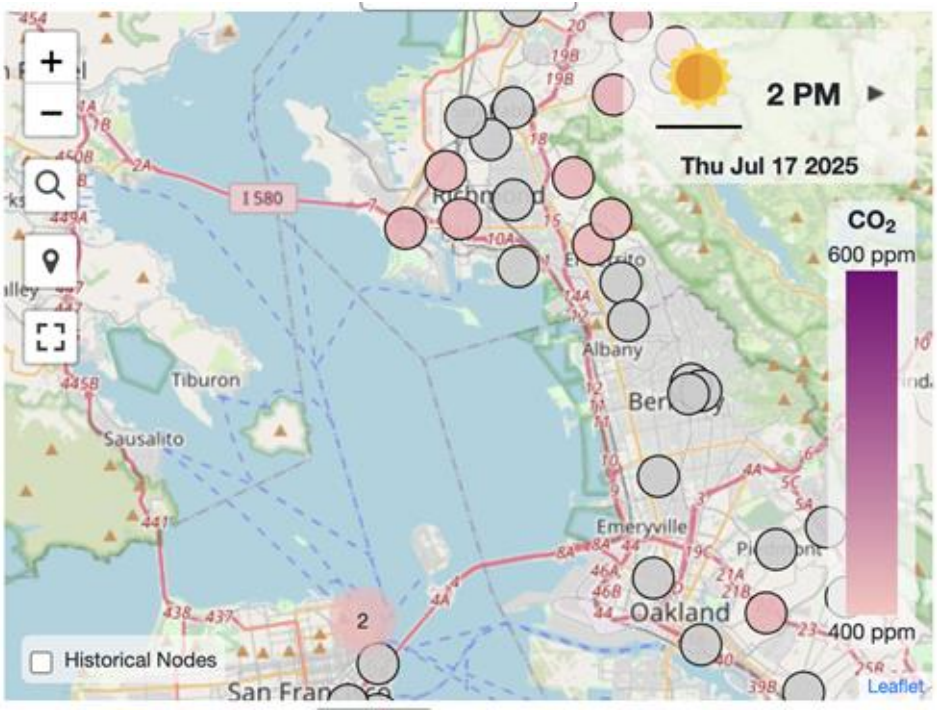


Figure 2: BEACO$_2$N sensor network

Conventional Flux Batch Inversion Modeling

- **Goal: estimate *x* the carbon "flux"**
- **Prior** guess of emissions flux from bottom-up inventory

$$x \sim N(x_a, B)$$

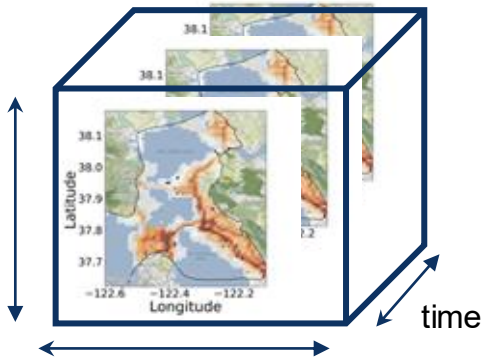


Figure 3: Prior inventory

- **Measurements, *Y*,** used to update prior

$$Y = Hx + \epsilon$$
$$Y|x \sim N(Hx, R)$$

Figure 2: BEACO$_2$N sensor network

Conventional Flux Batch Inversion Modeling

- **Goal: estimate *x* the carbon "flux"**
- Together we have a normal-normal Bayesian model:

$$\begin{pmatrix} X \\ Y \end{pmatrix} \sim N\left(\begin{pmatrix} x_a \\ Hx_a \end{pmatrix}, \begin{pmatrix} B & B^T H^T \\ HB & R + HBH^T \end{pmatrix}\right)$$

- With conditioning, we then can analytically solve for the estimated flux:

$$\implies x|Y \sim N(\hat{x}, \hat{\Sigma})$$

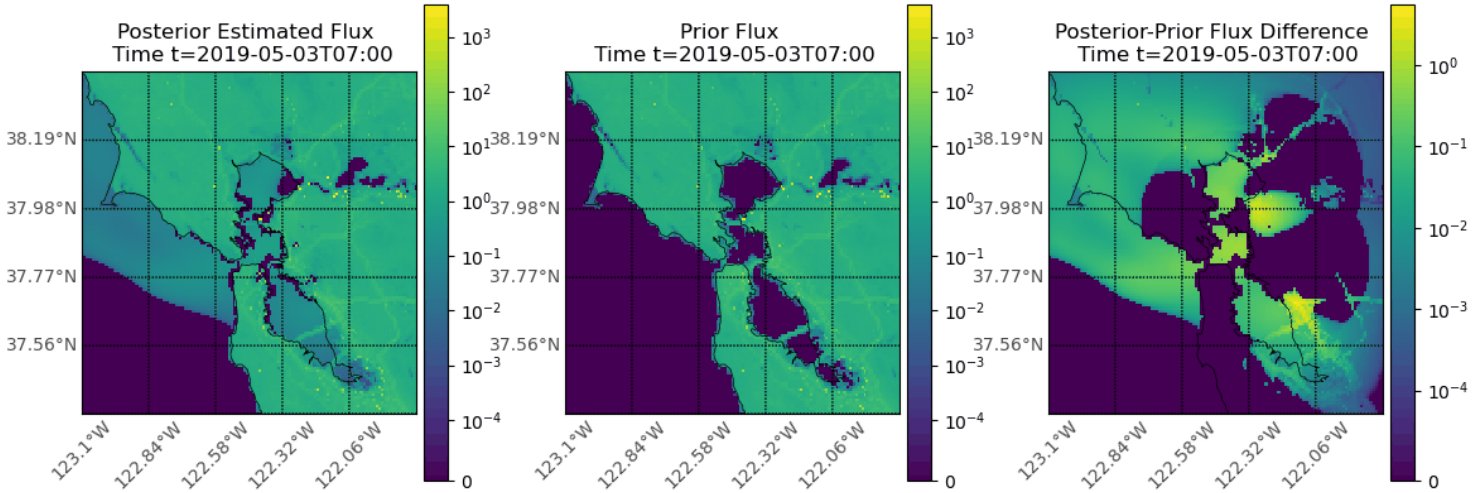
$$\hat{x} = x_a + (HB)^T (HBH^T + R)^{-1}(y - Hx_a)$$

$$\hat{\Sigma} = (H^T R^{-1} H + B^{-1})^{-1} = B - (HB)^T (HBH^T + R)^{-1} HB$$

Current analyses are limited to only calculating $\hat{x}$

Conventional Flux Batch Inversion Modeling Example

Figure 4: Example posterior and prior

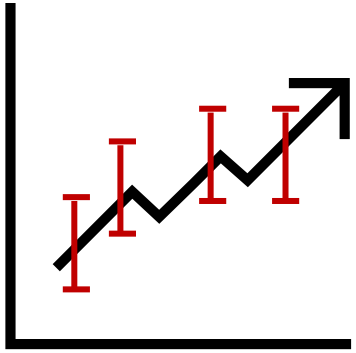


$$\hat{x} = x_a + (HB)^T(HBH^T + R)^{-1}(y - Hx_a)$$

Current Challenges

Statistical analyses we would like to have, but do not due to **computational cost:**

- Posterior uncertainty
- Sensitivity to prior specification
- Identifiability checks



Code Development:

- Code bases are not structured in **packages**
- Code is **siloed** – no common code base for multiple labs
- Code specialized for **one** type of inversion model, e.g., batch inversion
- Not always optimized for efficiency

Example inversion has a (50x96) x (96x127x157) dimension H matrix

Proposed Framework

We are working on an **open-source** object-oriented python code base to enable **reusability** that:

- Utilizes configuration files for **reusable** code
- Uses **Ray** for parallel computations
- Uses **sparse** matrix operations
- Will include functions for **statistical analyses**

This is currently a work in progress, and we welcome collaboration and feedback

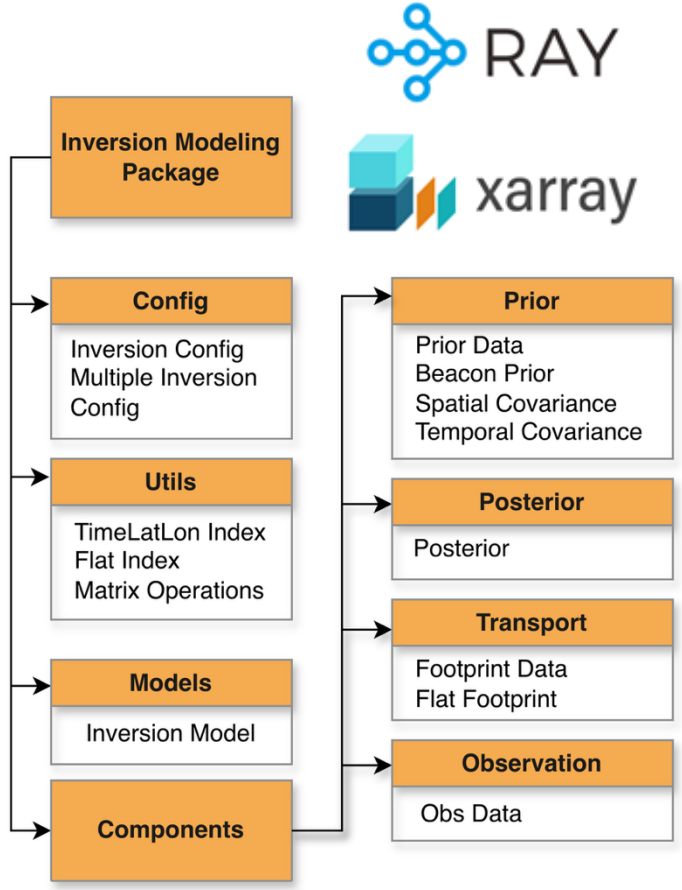


Figure 6: Overview of python package structure

Future Statistical Analyses

These can all be computed by **reusing** calculations from the mean

Posterior variance computation:

- Usually interested in a **size-reducing function** of the posterior

Influence function:

- Identify if **small changes** in a segment, i.e., a road, **can be detected**

Prior sensitivity:

- Compute the derivative of summed fluxes with respect to a **prior hyperparameter**

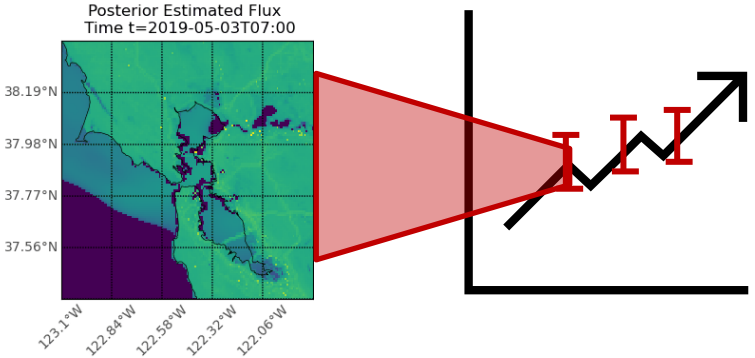


Figure 5: Posterior variance intuition

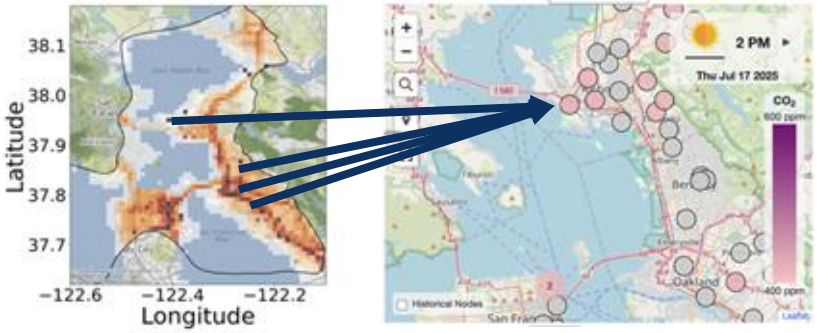
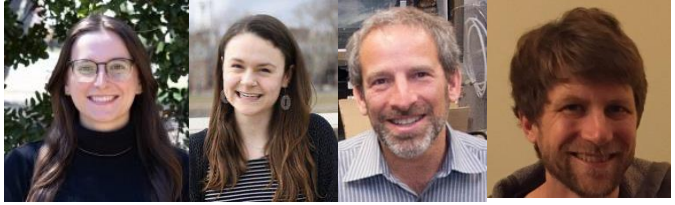


Figure 6: Influence function intuition

Thank You



Emails:

- Srandrade@berkeley.edu
- nasimow@berkeley.edu
- rgiordano@berkeley.edu
- rccohen@berkeley.edu

Additional Resources:

- $BEACO_2N$: https://beacon.berkeley.edu/overview/
- Python Toolkit:

References:

[1] S. Sengupta, H. Stevens, and M. Rojanasakul. 10 Years After a Breakthrough Climate Pact, Here's Where We Are. New York Times. https://www.nytimes.com/interactive/2025/11/07/climate/paris-agreement-climate.html, 2025. Accessed: 2025-12-07

[2] BEACO2N Network. Beaco2n: A network of atmospheric co2 and ch4 measurements. https://beacon.berkeley.edu/about/ , 2025. Accessed: 2025-07-17

[3] N. Asimow, A. Turner, and R. Cohen. Sustained Reductions of Bay Area CO2 Emissions 2018–2022. 58(15):6586–6594. ISSN 0013-936X. doi: 10.1021/acs.est.3c09642. URL https://doi.org/10.1021/acs.est.3c09642 .

[4] A. Turner, J. Kim, H. Fitzmaurice, C. Newman, K. Worthington, K. Chan, P. Wooldridge, P. K¨oehler, C. Frankenberg, and R. Cohen. Observed Impacts of COVID-19 on Urban CO Emissions. 47(22):e2020GL090037. ISSN 1944-8007. doi: 10.1029/2020GL090037. URL https://onlinelibrary.wiley.com/doi/abs/10.1029/2020GL090037.

Thank you

Proposed Solution

Current batch model is costly because of:

1. Large RAM requirements for storing matrices
2. Slow dense matrix multiplication and inversion

Proposed solution:

1. Avoid explicitly storing large matrices and use **Ray** distributed calculations
2. Leverage **sparsity** for matrix operations
3. Mindfully **cache** repeated computations

Our solution uses object-oriented programming to enable **reusability** and prioritize **efficiency**

Python Toolkit

- Object-oriented open-source code base
- Utilizes configuration files for reusable code
- Currently specialized for $BEACO_2N$ data but extendable to other data
- Will be extended to incorporate novel statistical analyses

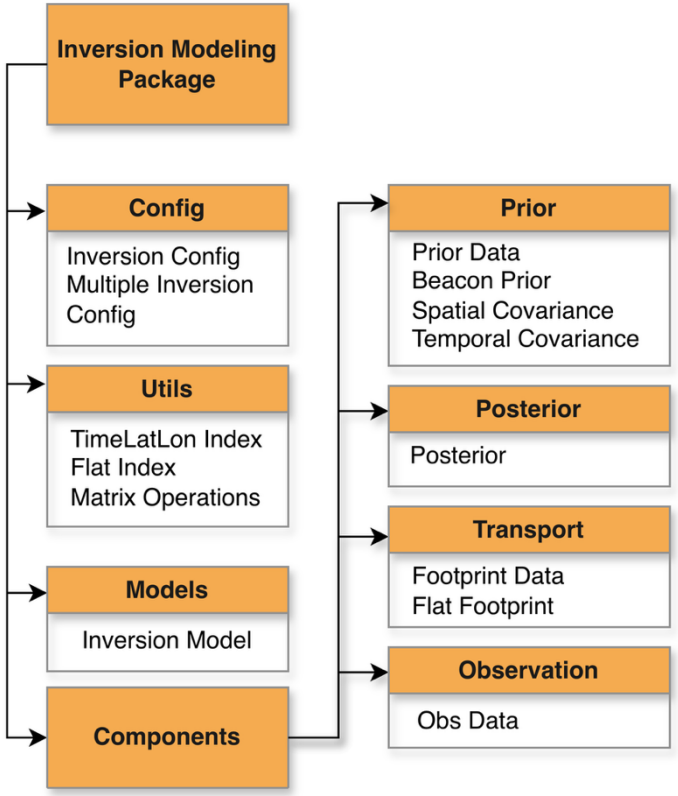


Figure 6: Overview of python package structure

Conventional Flux Inversion Modeling

- Flux: change in CO2 gas per unit time and area
- Uses normal-normal Bayesian model
- Forward model: $Y = Hx$

Inversion Assumes:

$$x \sim N(x_a, B)$$

- $Y|x \sim N(Hx, R)$

$$\implies x|Y \sim N(\hat{x}, \hat{\Sigma})$$

$$\hat{x} = x_a + (HB)^T(HBH^T + R)^{-1}(y - Hx_a)$$

This involves a (50x96) x (96x127x157) H matrix for a full inversion!!!

- Y - sensor readings $n \times 1$
- X - surface fluxes - e.g., emissions $(m_x m_y + 4)m_t \times 1$
- H - Jacobian - transport based on wind $n \times (m_x m_y + 4)m_t$
- Xa - prior fluxes - e.g., bottom up inventory $(m_x m_y + 4)m_t \times 1$
- B - prior spatio-temporal covariance $(m_x m_y + 4)m_t \times (m_x m_y + 4)m_t$
- R - model data mismatch -e.g., obs covariance $n \times n$