Uncertainty at Scale
How computer science *hinders* climate science

Patrick Ferris
Department of Computer Science and Technology, University of Cambridge
<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Introduction</td>
</tr>
<tr>
<td>2. Uncertainty in Data</td>
</tr>
<tr>
<td>3. Uncertainty in Code &amp; Dependencies</td>
</tr>
<tr>
<td>4. Uncertainty in Ecology</td>
</tr>
<tr>
<td>5. Reflections</td>
</tr>
</tbody>
</table>
Intro
Just as yesterday’s uncommon knowledge becomes today’s common knowledge, so yesterday’s unrecognized ignorance becomes today’s specified ignorance.

Figure 1: Additional carbon as calculated from a counterfactual analysis of a tropical moist forest avoided deforestation project
Uncertainty in Data
Figure 2: European Commission JRC Tropical Moist Forest Dataset from 2021 (left) and 2022 (right) for the year 2008 in Indonesia [6]. Legend:
- undisturbed, - degraded, - deforested, - regrowth, - water, - other

Source code: https://github.com/quantifyearth/jrc-diff
### Table 1: Changes in Land Use Class (LUC) as a percentage of the total area of Indonesia (1.9 million km$^2$)

<table>
<thead>
<tr>
<th>LUC</th>
<th>2021</th>
<th>2022</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undisturbed</td>
<td>58.39</td>
<td>57.80</td>
</tr>
<tr>
<td>Degraded</td>
<td>14.47</td>
<td>12.47</td>
</tr>
<tr>
<td>Deforested</td>
<td>7.25</td>
<td>10.30</td>
</tr>
<tr>
<td>Regrowth</td>
<td>0.37</td>
<td>0.55</td>
</tr>
<tr>
<td>Water</td>
<td>2.30</td>
<td>2.30</td>
</tr>
<tr>
<td>Other</td>
<td>17.22</td>
<td>16.58</td>
</tr>
</tbody>
</table>

Deforested misclassification is approximately the size of Togo!
• Delineation of new tree plantation mainly in Central, West Africa, Indonesia and Malaysia.
• Improvements and corrections of errors in the Annual Change collection in the sequence of values for deforestation of old regrowth forest.
• Etc.

European Commission Joint Research Committee [2].
Uncertainty in Code & Dependencies
System Dependencies I

$ gdaldem tri foo.tif bar2.tif
$ gdalinfo --version
3.2
$ gdaldem tri foo.tif bar3.tif
$ gdalinfo --version
3.3
$ gdal_calc.py --cal="A - B" -A bar2.tif bar3.tif
Figure 3: A case where the default algorithm for a CLI command was corrected between two minor versions of GDAL [1].
Figure 4: Numpy floating point number inconsistencies arising from optimisations in the SVML library\(^2\).

\(^2\)Details: https://github.com/numpy/numpy/issues/25269
Figure 5: Whilst subepsilon values on their own may be insignificant, they can accumulate across many computations to become significant.
Uncertainty in Ecology
PACT Tropical Moist Forest Accreditation Methodology

Draft for comments

Version 2.0

Published: December 15, 2023
Expires: June 15, 2024


About this document
6.2 Additionality & Leakage

The calculation of additionality and leakage are very similar and share a core algorithm. In summary, this takes an area of interest (the project or the project’s leakage zone) and runs the matching algorithm to find counterfactual pixels for this area. With this set of pairs we then analyse the carbon stock changes in these two regions to make an estimate on the amount of additional carbon (which may be negative in the case of leakage).

**Inputs:**
1. Area of Interest (AOI)
2. \( t_0 \) the start year and \( t_{now} \) the year of assessment (I1, I2)
3. The project polygon with leakage buffer (this is explicitly needed as part of the Find Potential Matches section to exclude it from the matching zone, whether calculating additionality or leakage both will remove this entire region). (I3)

**Algorithm:**
1. Let \( T \) be the set of Pixels in the AOI at time \( t_0 \) (A24, A25). Let \(|T|\) be the number of pixels in \( T \).
2. For each pixel in \( T \) find its land use class \( V \).
3. For each year \( t \) of time series from \( t_0 \) to \( t_{now} \), where \( t_0 = \) year of project start and \( t_{now} = \) the year of assessment.
   a. Let \( N_{TV}(t) \) be the number of pixels in each class \( V \) in \( T \) in year \( t \)
   b. The proportion of the AOI in class \( V \) in year \( t \) is \( N_{TV}(t)/|T| \).
   c. For each land use class \( V \)
      i. Find the total area in class \( V \) by multiplying \( N_{TV}(t)/|T| \) by the total AOI area in ha.
      ii. \( S_{TV}(t) = \) Carbon stock per ha in class \( V \) * total area of class \( V \) in the AOI.
   d. \( P_{tot}(t) = \) total carbon stock for year \( t \) in the AOI = \( \sum_{V} S_{TV}(t) \)
   e. In preparation for finding counterfactuals, call procedure Find Potential Matches to generate the set of potential treatment pixels \( (K) \) and potential control pixels \( (M) \) with:
def generate_additionality:
    project_area_msq: float,
    project_start: str,
    end_year: int,
    density: np.ndarray,
    matches_directory: str,
    expected_number_of_iterations: int,
    ) -> Dict[int, float]:
    """Calculate the additionality (or leakage) of a project from the counterfactual pair matchings alongside the carbon density values and some project specific metadata.""
    logging.info("Project area: %.2fmsq", project_area_msq)

    matches = glob.glob("*.parquet", root_dir=matches_directory)
    matches = [x for x in matches if is_not_matchless(x)]
    assert len(matches) == expected_number_of_iterations

    treatment_data: Dict[int, np.ndarray] = {}

    for pair_idx, pairs in enumerate(matches):
        logging.info("Computing additionality in treatment for %s", pairs)
        matches_df = pd.read_parquet(os.path.join(matches_directory, pairs))

        columns = matches_df.columns.to_list()
        columns.sort()

        earliest_year = find_first_loc(columns)
Communicating Methodologies IV

**Undisturbed Forest Trajectories**

The graph maps the trajectories of levels of undisturbed forests for the treatment and control regions. For the latest data for this project there’s a difference of +1.89%. 

<table>
<thead>
<tr>
<th>CONTROL</th>
<th>TREATMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elevation</td>
<td>180 m</td>
</tr>
<tr>
<td>Slope</td>
<td>3°</td>
</tr>
<tr>
<td>Accessibility</td>
<td>154 min</td>
</tr>
<tr>
<td>AGB*</td>
<td>93 Mg ha⁻¹</td>
</tr>
</tbody>
</table>
Reflections
...no-one with the same casual attitude to experimental instrumentation as many researchers have to code would be allowed anywhere near a lab.

— Baxter et al. [3]
**Researcher-Developer** wants to be judged on their scientific output, is a researcher at heart, however develops a lot of code due to the nature of their research...

**Research Software Engineer** comes from a research background but is also a skilled software developer... They want to be recognised for producing tools which others rely on for research.

— Baxter et al. [3]
### Figure 6: Six myths from Mary Shaw on software, developers and programming languages [5].

<table>
<thead>
<tr>
<th>Myth</th>
<th>se mythos</th>
<th>se pragmos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Professional Programmer</td>
<td>Programs are written by highly skilled professional programmers.</td>
<td>Vernacular software developers vastly outnumber professional developers. Professional developers now (mostly) do things other than write code.</td>
</tr>
<tr>
<td>The Code IS the Software</td>
<td>Software is simply the symbolic program text.</td>
<td>Software systems are coalitions of many types of elements from many sources with sketchy specifications and unannounced behavior change.</td>
</tr>
<tr>
<td>The Purity myths:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mathematical Tractability</td>
<td>Soundness of programming languages is essential.</td>
<td>Task-specific expressiveness is more important than completeness or soundness.</td>
</tr>
<tr>
<td>Correctness</td>
<td>Correctness of software is also essential.</td>
<td>Fitness for task usually matters more than absolute correctness.</td>
</tr>
<tr>
<td>Specifications</td>
<td>Thus formal specifications are also essential.</td>
<td>Much software is developed to discover what it should do, not to satisfy a prior specification.</td>
</tr>
<tr>
<td>...and a new one... AI Revolution</td>
<td>...a new one going around... Artificial intelligence (actually machine learning) is so special that it will break normal software development.</td>
<td>Artificial intelligence has long been an incubator for disruptive programming ideas; the issues of current concern have variants in conventional software, where there are established ways to deal with them.</td>
</tr>
</tbody>
</table>
Thank you

• Prof. Anil Madhavapeddy and Prof. Srinivasan Keshav
• Colleagues in Plant Sciences and Zoology especially Dr. Tom Swinfield
• Colleagues in Computer Science especially Michael Dales, Derek Sorensen, Ryan Gibb, and Madeline Lisaius
GDAL.  
https://gdal.org/.

List of updates integrated in the TMF 2022 products.  

The research software engineer.  
Sept. 2012. 
Digital Research 2012 ; Conference date: 10-09-2012 Through 12-09-2012.
R. K. Merton.  
Three fragments from a sociologist’s notebooks: Establishing the phenomenon, specified ignorance, and strategic research materials.  

M. Shaw.  
Myths and mythconceptions: what does it mean to be a programming language, anyhow?  