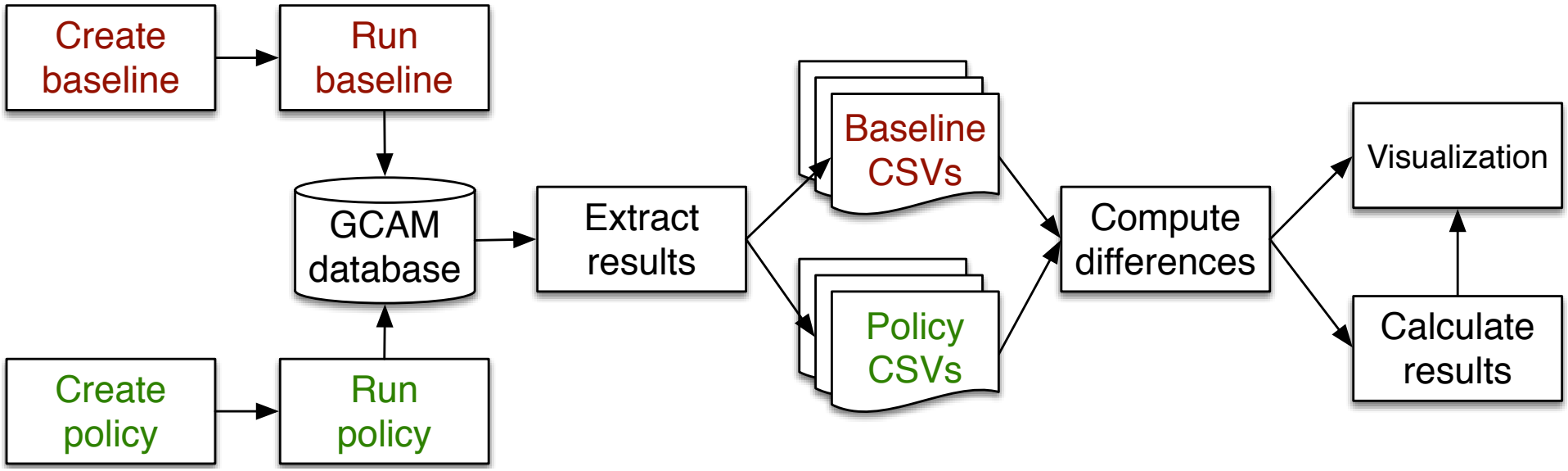
Introduction to pygcam

Richard Plevin, Ph.D.
Transportation Sustainability Research Center
UC Berkeley

October 13, 2016
GCAM Community Workshop

Berkeley
UNIVERSITY OF CALIFORNIA

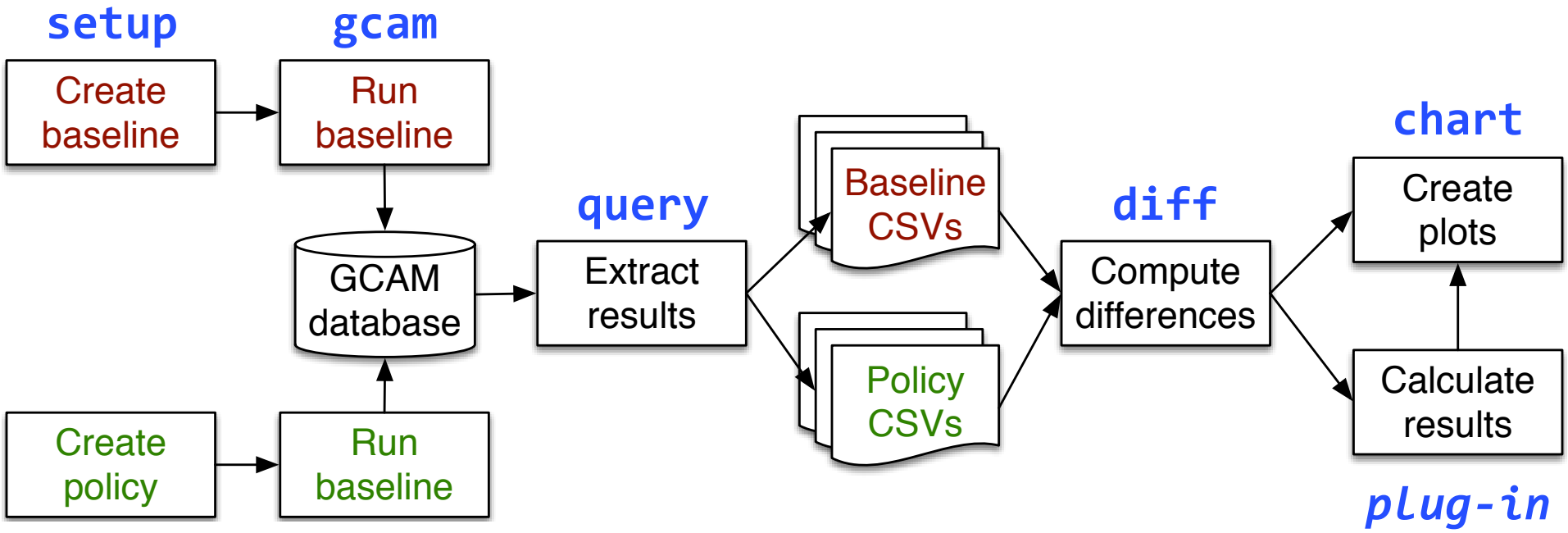
Generic GCAM workflow



Manual data manipulation

- Edit of XML input and configuration files
- Copy & paste into Excel or write batch query files
- Interpolate annual values between time-steps
- Calculate deltas from baseline
- Generate figures

First pass: a script for each step



But…

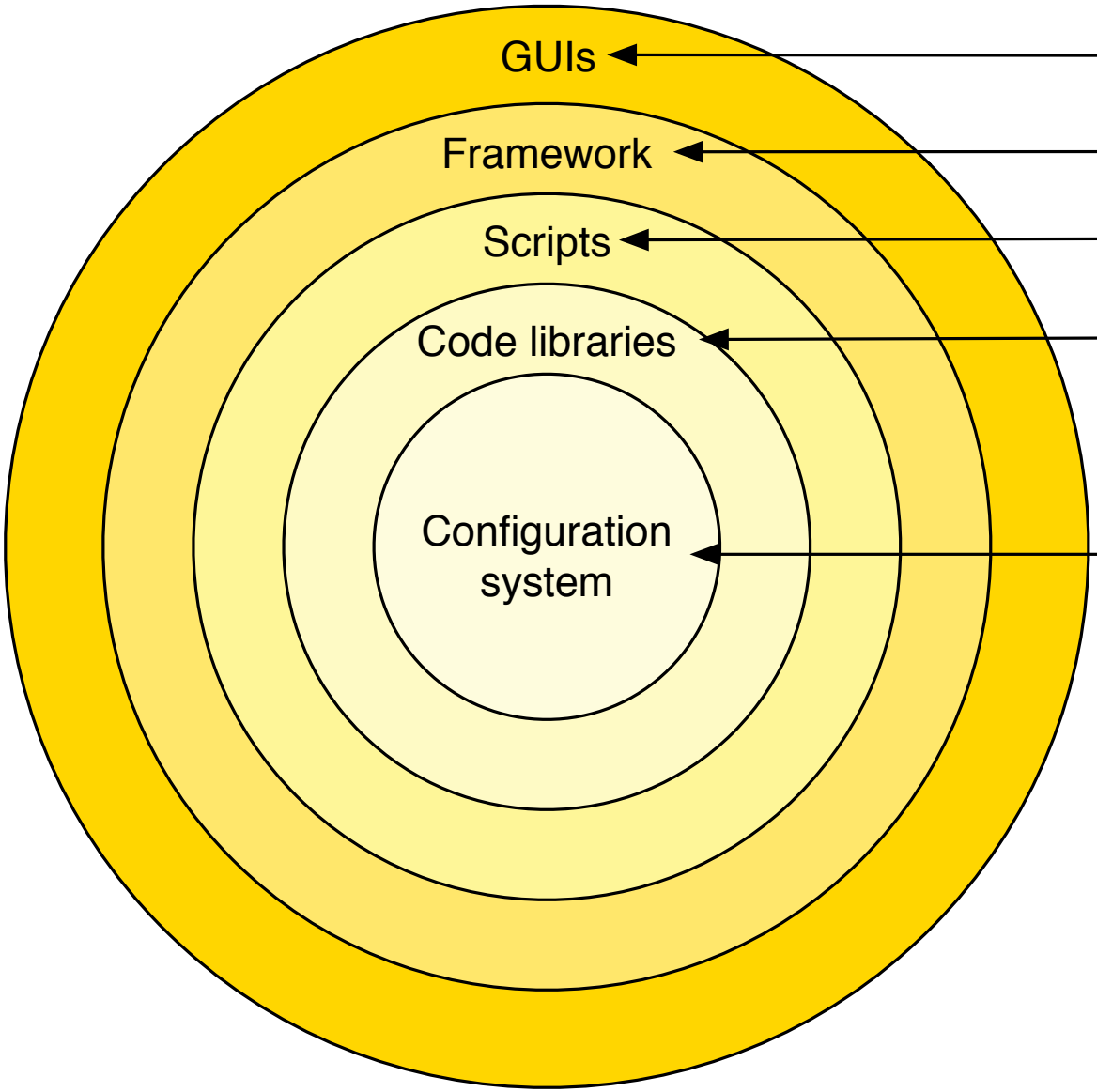
- Largely duplicated scripts for each project
- Additional scripts to combine steps
- Ad hoc input file format and many files
- Expanding pile of code and data
- Each user has to “roll their own”

pygcam

- Project workflow management framework
- Written in Python language
- Runs on Windows, Mac OS X, and Linux
- Open source, funded by JGCRI
- Command-line only for now; GUI to be developed

Project system – Goals

- Automate much of the GCAM workflow
- Automate modification of GCAM XML input files
- Easily customize for your preferences / environment
- Extensibility via plug-ins
- Define all GCAM project components in XML
- Selectively run workflow steps / scenarios as needed
- Support for high-performance computing systems



Interactive tools (still to come)

Workflow management system

Perform individual workflow tasks

High-level API to workflow steps

File locations and parameter values

Berkeley
UNIVERSITY OF CALIFORNIA

XML-based project file

- Steps to take to perform workflow
- GCAM XML file modification instructions
  - Manages input files and configuration file
- Queries to execute, including re-aggregation
- Scenario groups (baseline + policy scenarios)
- User-defined and automatic variables
- Calls internal sub-commands or external programs

Scenario definition “language”

```
<scenarios defaultGroup=“base-90”>
   <iterator name="protection" values="0, 90"/>

   <iterator name="tax" values="10,15,20,25"/>

    <scenarioGroup name="protect-{protection}" useGroupDir="0" iterator="protection">

        <scenario name="base-{protection}" baseline="1">
            <if value1="{protection}" value2="0">
                <delete name="protected_land_input_2"/>
                <delete name="protected_land_input_3"/>
            </if>
        </scenario>

        <!-- e.g., scenario “tax-10-0” => $10/tonne tax, 0% land protection -->
        <scenario name="tax-{tax}-{protection}" iterator="tax">
           <add name="carbon_tax">../input/policy/carbon_tax_{tax}_5.xml</add>
        </scenario>

    </scenarioGroup>
</scenarios>
```

Berkeley
UNIVERSITY OF CALIFORNIA

Command-line interface

- All functionality accessed via “gt” (gcamtool) script
- Sub-commands identify distinct functions, each with options
- Arguments follow (some) option specifiers

```
gt run -g base-0
```

```
gt run -s query,diff,plot -S tax-10-90
```

Berkeley
UNIVERSITY OF CALIFORNIA

gt sub-commands

Sub-command	Description
chart	generate figures
config	list configuration variables
diff	compute differences between CSV files
gcam	run GCAM on specified scenarios
new	create a new project using template files
protect	generate files with specified land protection
query	run XML queries to produce CSV files
run	run workflow steps defined in project.xml
setup	generate XML input and configuration files
other	anything you like, via custom plug-ins

Tutorial, Part 2

We begin Part 2 of the tutorial by creating a new project called `ctax` using the templates files provided by `pygcam`.

2.0 Create the project structure and initial configuration file

The first step in creating a new product is to run gcamtool new sub-command. To create the project `ctax` with the project directory `/Users/rjp/projects/ctax`, I would run the following command:

```
gt new -c -r /Users/rjp/projects ctax
```

This both creates the initial file structure in `/Users/rjp/projects/ctax`, and (because I specified the `-c` flag) adds a section for `ctax` to my configuration file, which is found in my home directory. In my case, it is in `/Users/rjp/projects/.pygcam.cfg`.

When `gt` runs, it checks whether this file exists. If the file is not found, it is created with all available configuration parameters shown in comments (i.e., lines starting with '#') explaining their purpose and showing their default values. To uncomment a line, simply remove the leading '#' character.

Here is the `.pygcam.cfg` file (with the long listing of default settings removed):

pygcam.diff

Functions for computing differences between CSV files and for generating CSV and XLSX from multiple CSV files.

API

pygcam.diff.computeDifference(*df1, df2, resetIndex=True*)

Compute the difference between two DataFrames.

Parameters:
- **df1** – a pandas DataFrame instance
- **obj2** – a pandas DataFrame instance
- **resetIndex** – (bool) if True (the default), the index in the DataFrame holding the computed difference is reset so that data in non-year columns appear in individual columns. Otherwise, the index in the returned DataFrame is based on all non-year columns.

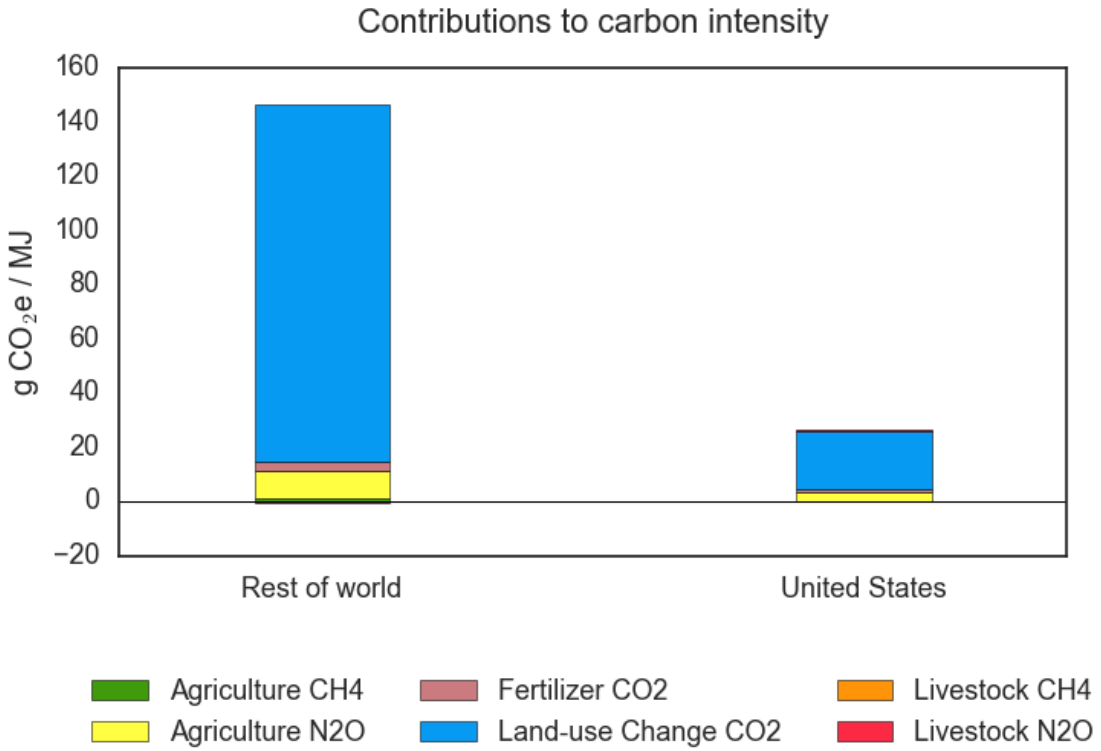
Returns: a pandas DataFrame with the difference in all the year columns, computed as (df2 - df1).

pygcam.diff.writeDiffsToCSV(*outFile, referenceFile, otherFiles, skiprows=1, interpolate=False, years=None, startYear=0*)

Compute the differences between the data in a reference .CSV file and one or more other .CSV files as (other - reference), optionally interpolating annual values

Plug-ins

- Example: compute "**carbon intensity**" of biofuels
  - Extract emissions and fuel production data
  - Calculate sum(emissions) / sum(fuel change)
  - Write results to CSV file for plotting



Berkeley
UNIVERSITY OF CALIFORNIA

Still to come…

- Monte Carlo Simulation framework
- Graphical User Interface
- More built-in chart types
- More documentation
- Test / validation suite
- Integration with software from other GCAM community members

Tutorial session Friday morning

How do I get it?

http://pygcam.readthedocs.io

for installation instructions and documentation

https://bitbucket.org/plevin/pygcam

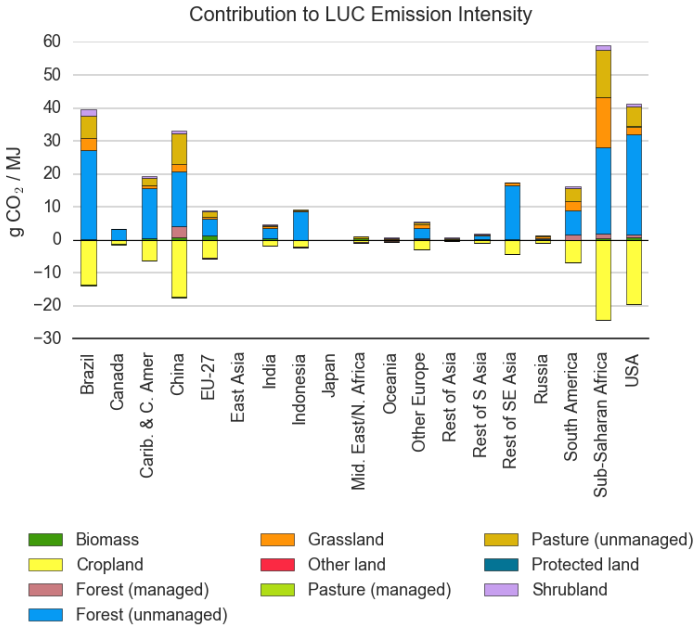
to access the source code repository

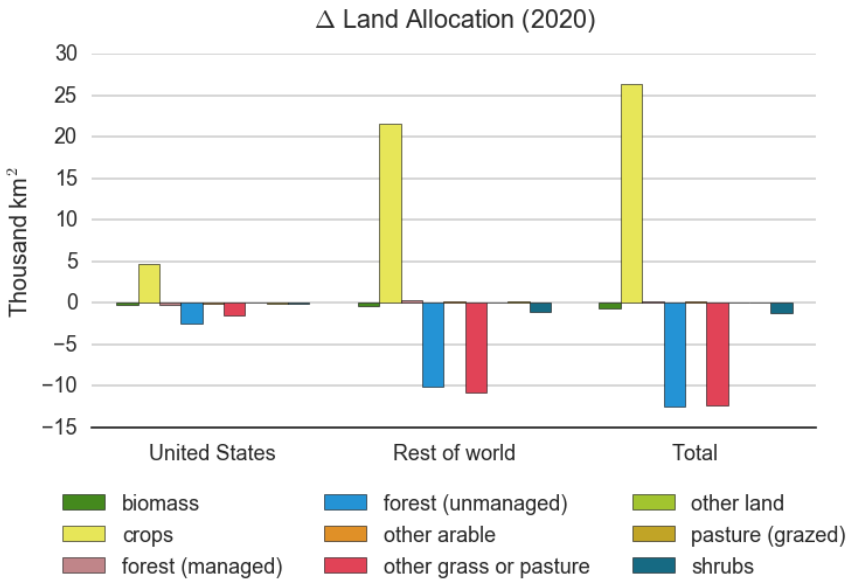
Thanks, JGCRI!

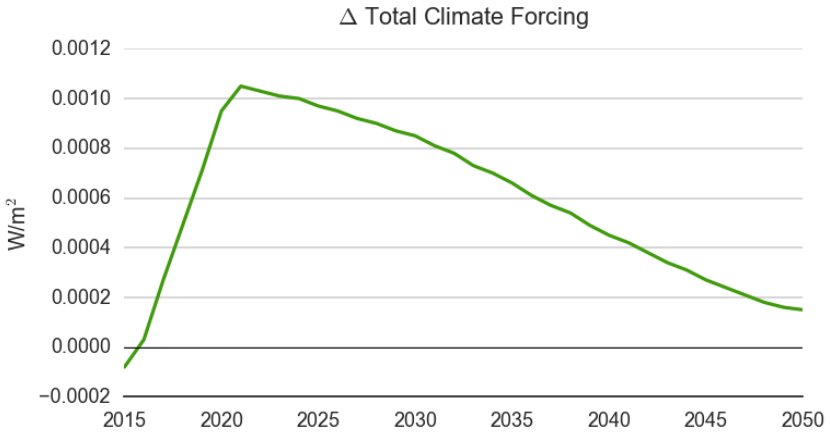
- To **Leon Clarke** for funding this effort
- To **Robert Link** for weekly discussion & feedback
- To **Pralit Patel** for technical support

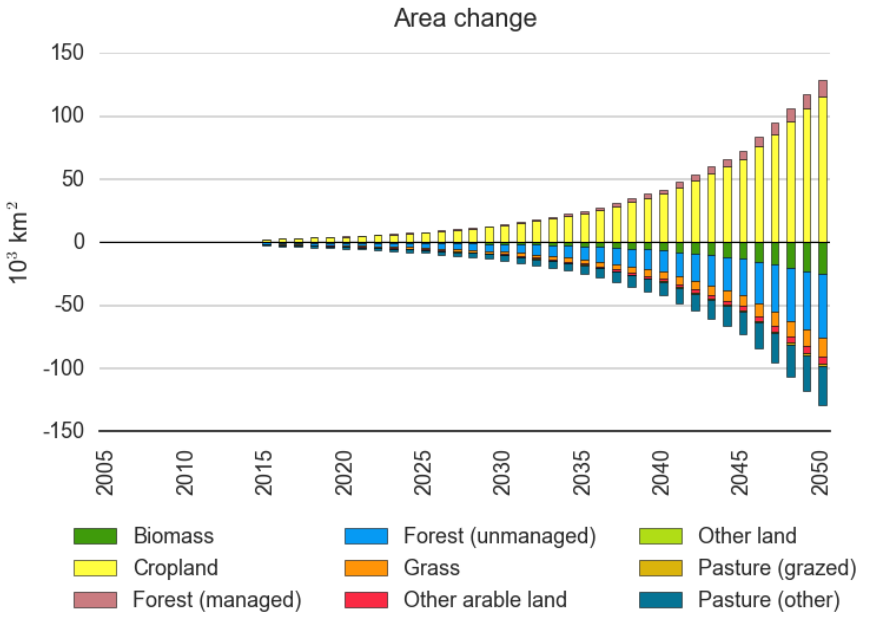
Berkeley
UNIVERSITY OF CALIFORNIA

BACKUP SLIDES









Berkeley
UNIVERSITY OF CALIFORNIA

gt new

- Create file structure for a new project
- Generates a project.xml file and initializes config
- Default structure matches default config settings
- File “template” can be customized per site / per user

```
gt new --projectRoot ~/projects myProject \
    --addToConfig
```

gt setup

- Automates modification of XML input files
- XML declarations define scenarios, with iteration over terms and conditional ("if") expressions.
  - Concisely define numerous related scenarios
- Supports customization via Python
  - Create new commands "callable" from XML

```
gt setup --baseline base-0 --scenario s1 \
   --workspace ~/ws/myProject/s1 --stop 2050 \
   --years 2015-2050
```

Berkeley
UNIVERSITY OF CALIFORNIA

gt gcam

- Run gcam in a separate “sandbox” directory
- Sandbox is created on-the-fly
- Queue to run on cluster: “gt --batch run …”

```
gt --batch --minutes=15 gcam --scenario s1 \
   -w ~/ws/myProject/s1
```

Berkeley
UNIVERSITY OF CALIFORNIA

gt query

- Automatic generation of batch-query files
- Extracts queries by name from, e.g., Main_Queries.xml
- On-the-fly aggregation via named “rewrite sets”
- Easily aggregate AEZ without verbose “rewrites”

```
gt query --outputDir ~ws/myProject/s1/results \
   --workspace ~/ws/myProject/s1 \
   refined_liquids_production_by_technology \
   Global_mean_temperature etc.
```

Berkeley
UNIVERSITY OF CALIFORNIA

gt diff

- Calculate differences between 2 or more CSV files
- Optionally limit years, interpolate annual values

```
gt diff --workingDir ~/ws/myProject \
   --years=2005-2050 --startYear=2015 \
   --queryFile --interpolate \
   --outFile ~/ws/myProject/s1/diffs
   base/Climate_forcing s1/Climate_Forcing
```

Berkeley
UNIVERSITY OF CALIFORNIA

gt chart

- Plot data from GCAM-generated (or "diff") CSV files
- Set defaults for a common look & feel
- On-the-fly unit conversions (e.g., C to CO_2)
- Control aesthetics: labels, titles, legend, and more

```
gt chart --workingDir ~/ws/s1/diffs \
   --outputDir figures --years 2005-2050 \
   --ygrid --zeroLine --reference base \
   --scenario s1 --fromFile charts.txt
```

Berkeley
UNIVERSITY OF CALIFORNIA

Custom scripts

- Pygcam is designed for use as a library
- Built-in sub-commands are based on the library
- Project system is based on the scripts
- Write “plug-ins” to customize gcamtool
  - Uses configuration and logging systems
  - Directly callable in project system
  - Automatically “batch runnable” on cluster

```
$ gt run -h
usage: gt run [-h] [-a] [-f PROJECTFILE] [-g GROUP] [-G] [-k SKIPSTEPS]
              [-K SKIPSCENARIOS] [-l] [-L] [-n] [-p PROJECT] [-q] [-s STEPS]
              [-S SCENARIOS] [--version] [--vars] [-x SANDBOXDIR]

optional arguments:
  -h, --help            show this help message and exit
  -a, --allGroups       Run all scenarios for all defined groups.
  -f PROJECTFILE, --projectFile PROJECTFILE
                        The XML file describing the project. If set, command-
                        line argument takes precedence. Otherwise, value is
                        taken from config file variable GCAM.ProjectXmlFile,
                        if defined, otherwise the default is './project.xml'.
  -g GROUP, --group GROUP
                        The name of the scenario group to process. If not
                        specified, the group with attribute default="1" is
                        processed.
  -G, --listGroups      List the scenario groups defined in the project file
                        and exit.
  -k SKIPSTEPS, --skipStep SKIPSTEPS
                        Steps to skip. These must be names of steps defined in
                        the project.xml file. Multiple steps can be given in a
                        single (comma-delimited) argument, or the -k flag can
                        be repeated to indicate additional steps. By default,
                        all steps are run.
  -K SKIPSCENARIOS, --skipScenario SKIPSCENARIOS
                        Scenarios to skip. Multiple scenarios can be given in
                        a single (comma-delimited) argument, or the -K flag
                        can be repeated to indicate additional scenarios. By
                        default, all scenarios are run.
```