GCAM-MCS

Distributed Monte Carlo Simulation framework for GCAM

Richard Plevin, Ph.D.

Institute of Transportation Studies
University of California – Davis
plevin@ucdavis.edu

GCAM Community Workshop
December 3, 2015

ITS UCDAVIS INSTITUTE OF TRANSPORTATION STUDIES

CHALLENGES OF MONTE CARLO SIMULATION WITH GCAM

- 10-15 min run-time precludes MCS on desktop system
- Running MCS on a cluster is technically challenging

Carver @ NERSC.gov: 1202 compute nodes (9,984 processor cores)

CHALLENGES OF MONTE CARLO SIMULATION WITH GCAM (cont.)

Step in performing MCS with GCAM

CHALLENGES OF MONTE CARLO SIMULATION WITH GCAM (cont.)

Step in performing MCS with GCAM

1. Identify “parameters” in GCAM

CHALLENGES OF MONTE CARLO SIMULATION WITH GCAM (cont.)

Step in performing MCS with GCAM

1. Identify “parameters” in GCAM
2. Specify probability distributions

CHALLENGES OF MONTE CARLO SIMULATION WITH GCAM (cont.)

Step in performing MCS with GCAM

1. Identify “parameters” in GCAM
2. Specify probability distributions
3. Use random values to perturb parameters

CHALLENGES OF MONTE CARLO SIMULATION WITH GCAM (cont.)

Step in performing MCS with GCAM

1. Identify “parameters” in GCAM
2. Specify probability distributions
3. Use random values to perturb parameters
4. Setup separate workspace to run GCAM

CHALLENGES OF MONTE CARLO SIMULATION WITH GCAM (cont.)

Step in performing MCS with GCAM

1. Identify “parameters” in GCAM
2. Specify probability distributions
3. Use random values to perturb parameters
4. Setup separate workspace to run GCAM
5. Run baseline and policy case(s)

CHALLENGES OF MONTE CARLO SIMULATION WITH GCAM (cont.)

Step in performing MCS with GCAM

1. Identify “parameters” in GCAM
2. Specify probability distributions
3. Use random values to perturb parameters
4. Setup separate workspace to run GCAM
5. Run baseline and policy case(s)
6. Extract results from each scenario

CHALLENGES OF MONTE CARLO SIMULATION WITH GCAM (cont.)

Step in performing MCS with GCAM

1. Identify “parameters” in GCAM
2. Specify probability distributions
3. Use random values to perturb parameters
4. Setup separate workspace to run GCAM
5. Run baseline and policy case(s)
6. Extract results from each scenario
7. Calculate differences (policy – baseline)

CHALLENGES OF MONTE CARLO SIMULATION WITH GCAM (cont.)

Step in performing MCS with GCAM

1. Identify “parameters” in GCAM
2. Specify probability distributions
3. Use random values to perturb parameters
4. Setup separate workspace to run GCAM
5. Run baseline and policy case(s)
6. Extract results from each scenario
7. Calculate differences (policy – baseline)
8. Store results for analysis

CHALLENGES OF MONTE CARLO SIMULATION WITH GCAM (cont.)

Step in performing MCS with GCAM

1. Identify “parameters” in GCAM
2. Specify probability distributions
3. Use random values to perturb parameters
4. Setup separate workspace to run GCAM
5. Run baseline and policy case(s)
6. Extract results from each scenario
7. Calculate differences (policy – baseline)
8. Store results for analysis
9. Analyze and plot results

DISTRIBUTED MONTE-CARLO SIMULATION FRAMEWORK

– Goal –
Enable modelers to run MCS and robustness analysis with models that in practice require cluster computing.

Open-source framework designed for flexibility and extensibility.

and many open-source Python modules

Available at https://bitbucket.org/plevin/

CORE PACKAGE + MODEL-SPECIFIC FEATURES

coremcs

- Database management
- Latin-hypercube sampling
- Generalized parameter file handling
- Creation of model execution "sandboxes"
- Execution of model on Linux cluster
- Extraction of model results
- Storage of scalar / timeseries results in DB
- Uncertainty importance analysis
- Several types of plots

gcammcs

- Handle XML input files
- Set up GCAM workspace
- Run batch queries via ModelInterface
- Handle cleanup of several large files
- Run post-processing script

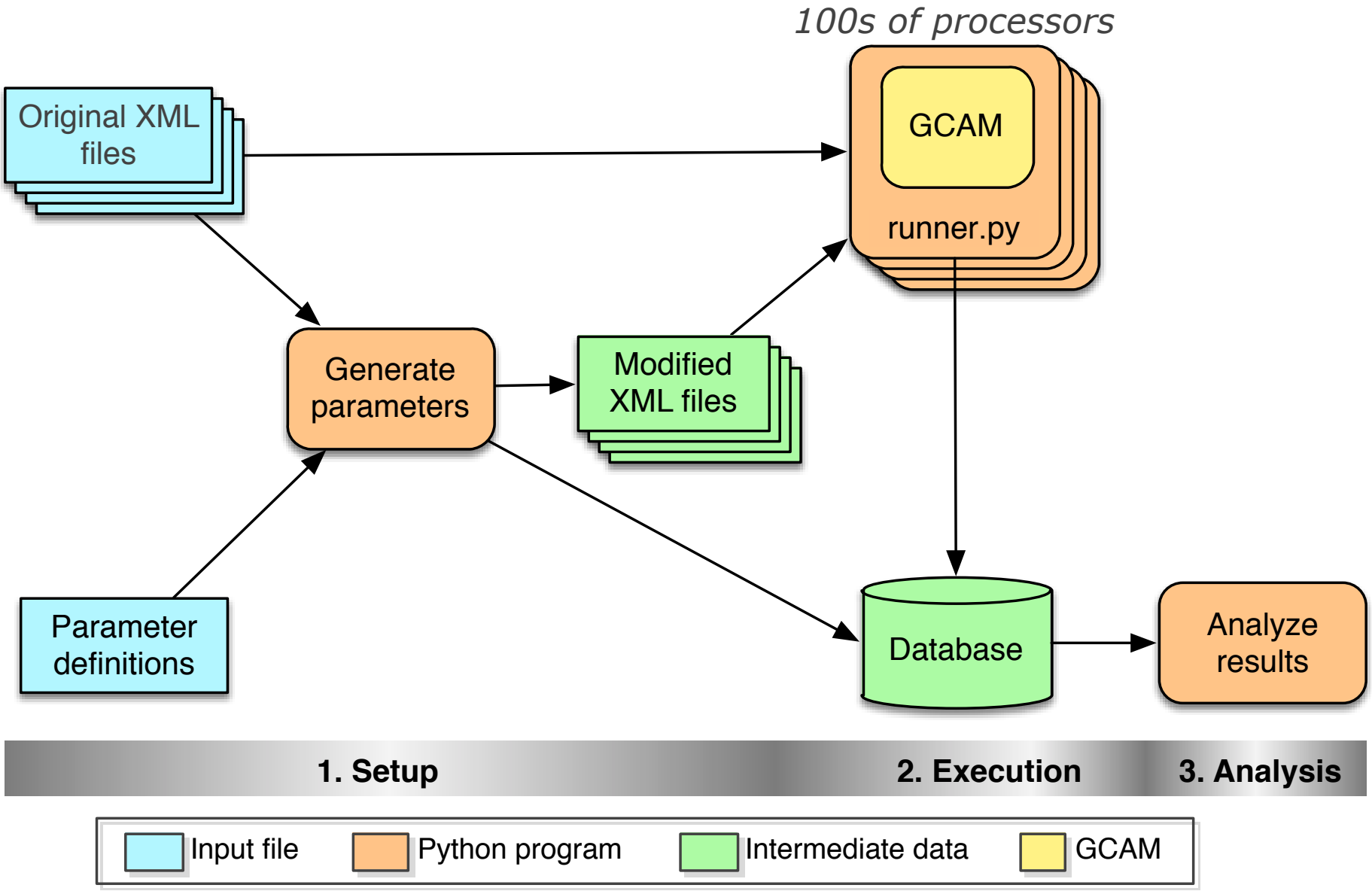
gtapmcs

- Handle "HAR" input & output files
- Handle sets used to iterate over matrices
- Set up GTAP workspace
- Construct proper GTAP command-line
- Extract "accuracy" from GTAP log file

ilucmcs

- Extensions to handle multiple sub-models
- LUC carbon accounting via AEZ-EF
- Calculation of carbon intensity etc.

GCAM MONTE-CARLO SIMULATION FRAMEWORK



MAIN GCAM-MCS COMMANDS

- gcammcs *global-args subcmd subcmd-args*
- gcammcs new *args*
  - Initialize a new user-application run-time structure
- gcammcs gensim *args*
  - Generate a simulation
- gcammcs queue *args*
  - Queue batch runs of GCAM on Linux cluster
- gcammcs analyze *args*
  - Generate statistics and figures from MCS results
- gcammcs config *args*
  - Get, set, and list MCS configuration parameters

Allows user to control:

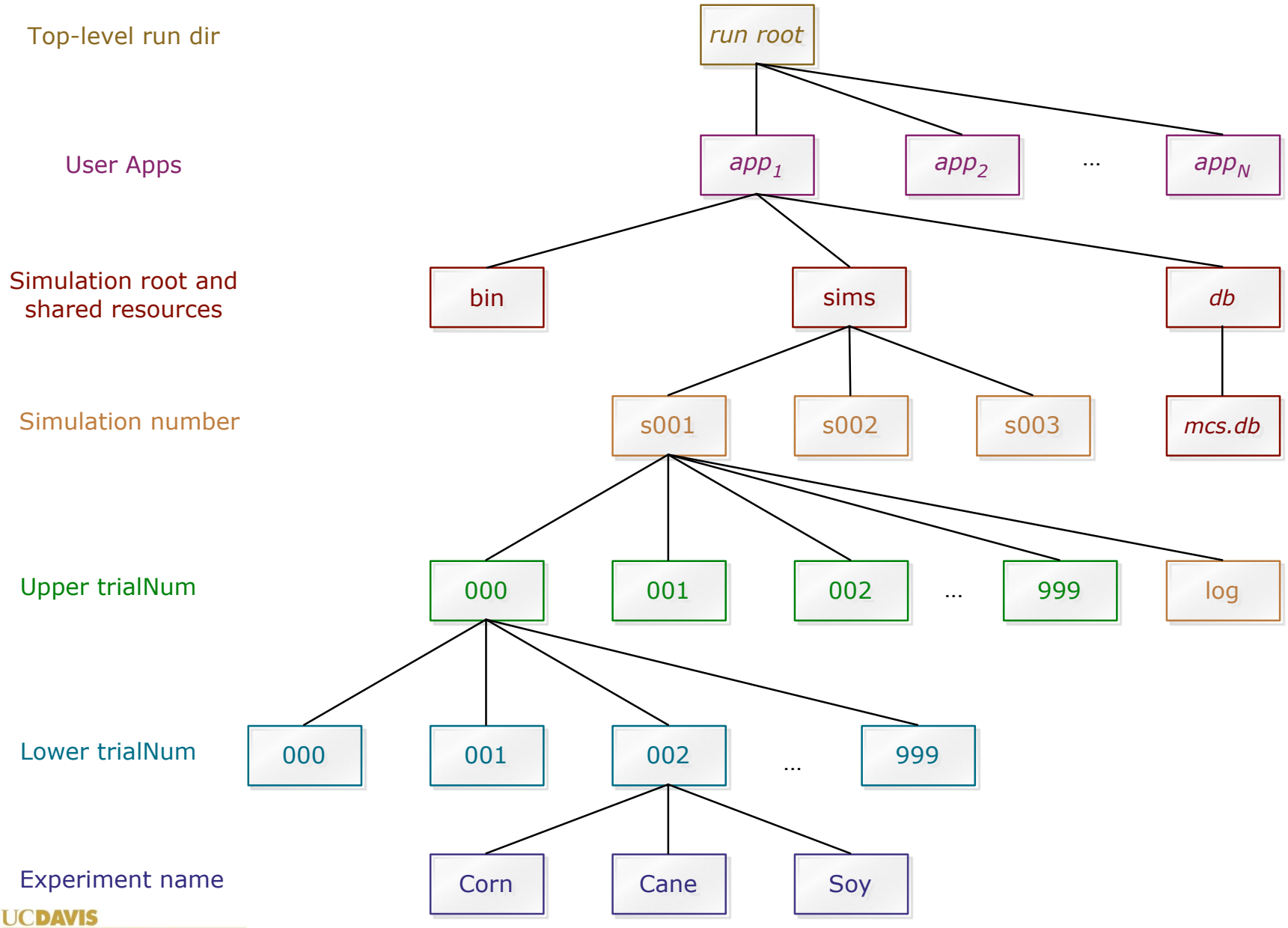
- System configuration (database, Mac/Linux differences)
- Location of key files and directories
- Run-time parameters
- Batch queries to run when GCAM completes
- Post-processing script to run after queries complete
- Plotting defaults
- User-application configuration info

EXTENSIVE CONFIGURATION SYSTEM ALLOWS CUSTOMIZATION

```
[DEFAULT]
Core.LogLevel     = INFO         ; must be one of {DEBUG, INFO, WARNING, ERROR, FATAL}
Core.LogConsole   = True
Core.MinPerTask   = 30

GCAM.RootDir      =%(Home)s/GCAM
GCAM.ModIntDir    = %(GCAM.RootDir)s/current/ModelInterface

GCAM.RunDbDir     = %(Core.RunDir)s/db
GCAM.RunWorkspace = %(Core.RunDir)s/workspace
GCAM.RunCvsDir    = %(GCAM.RunWorkspace)s/cvs
GCAM.RunExeDir    = %(GCAM.RunWorkspace)s/exe
GCAM.RunInputDir  = %(GCAM.RunWorkspace)s/input
GCAM.RunLibsDir   = %(GCAM.RunWorkspace)s/libs
GCAM.RunQueryDir  = %(GCAM.RunWorkspace)s/queries

GCAM.LinuxJarFile = %(GCAM.ModIntDir)s/ModelInterface.jar
#GCAM.MacJarFile  = %(GCAM.ModIntDir)s/ModelInterface.app/Contents/Resources/Java/ModelInterface.jar
GCAM.JarFile      = %(GCAM.LinuxJarFile)

GCAM.QueryPath    = .:%(GCAM.RunQueryDir)s:%(GCAM.RunQueryDir)s/Main_queries.xml

# A file with the names of queries to run, one per line
GCAM.BatchQueryFile =

# Optional regional mapping to use for extracted query definitions
GCAM.RegionMapFile =

# Executable to run when GCAM completes; default is none
#GCAM.PostProcessor = gcammcs diff -s{simId} -t{trialNum} -p{scenario}
GCAM.PostProcessor  =

GCAM.Years = 1990,2005-2100:5    ; years to evaluate
```

This is just a small sample of configuration options

GCAM-MCS RUN-TIME FILE STRUCTURE



RELATIONAL DATABASE

- Supports sqlite3 and postgresql
- Stores:
  - Meta-data (simulation, run, scenario, inputs, outputs)
  - Run-time status
  - Input and output values (scalar and timeseries)

simId	expName	status	count(*)
1	base-1	alarmed	14
1	base-1	failed	35
1	base-1	running	1
1	base-1	succeeded	950
1	corn-1	alarmed	19
1	corn-1	failed	26
1	corn-1	succeeded	955
1	stover-1	failed	51
1	stover-1	succeeded	949

Parameters.xml

- Defines parameters and their distributions
  - parameter => elements returned by XPath query
- Original values are added to, multiplied by, or replaced by random values from distribution
- "Hooks" to call Python functions for custom processing per parameter or per XML file

EXAMPLE PARAMETER DEFINITIONS

```
<ParameterList>
  <InputFile name="demand">
    <!-- Price elasticity of food crop demand. Currently zero everywhere, so can't multiply -->
    <Parameter mode="shared" name="food-crop-price-elast">
      <Query>//energy-final-demand[@name="FoodDemand_Crops"]/price-elasticity[@year>=2015]</Query>
      <Distribution apply="direct" name="triangle" min="-0.2" mode="0" max="0"/>
    </Parameter>

    <!-- Price elasticity of meat demand -->
    <Parameter mode="shared" name="meat-price-elast">
      <Query>//energy-final-demand[@name="FoodDemand_Meat"]/price-elasticity[@year>=2015]</Query>
      <Distribution apply="multiply" name="triangle" factor="0.2"/>
    </Parameter>

    <!-- Income elasticity of food crop demand -->
    <Parameter mode="shared" name="food-crop-income-elast">
      <Query>//energy-final-demand[@name="FoodDemand_Crops"]/income-elasticity[@year>=2015]</Query>
      <Distribution apply="add" name="triangle" range="0.2"/>
    </Parameter>
  </InputFile>
</ParameterList>
```

EXAMPLE DISTRIBUTIONS

```
name="Normal" mean="50" stdev="10"
name="Normal" type="delta"  mean="50" stdev="10"
name="Normal" type="factor" mean="50" stdev="10"

name="Uniform" min="10" max="20"
name="Uniform" type="factor" min="10" max="20"
name="Uniform" type="delta" range="2"

name="LogUniform" type="factor" factor="2"

name="LogNormal" mean="50" std="5"
name="LogNormal" low95="3" high95="50"

name="Triangle"  min="20" max="40" mode="35"

name="Binary"

name="Grid" min="5" max="30" count="5"
```

PROCESSING RESULTS

User identifies scalar / timeseries values to save in database

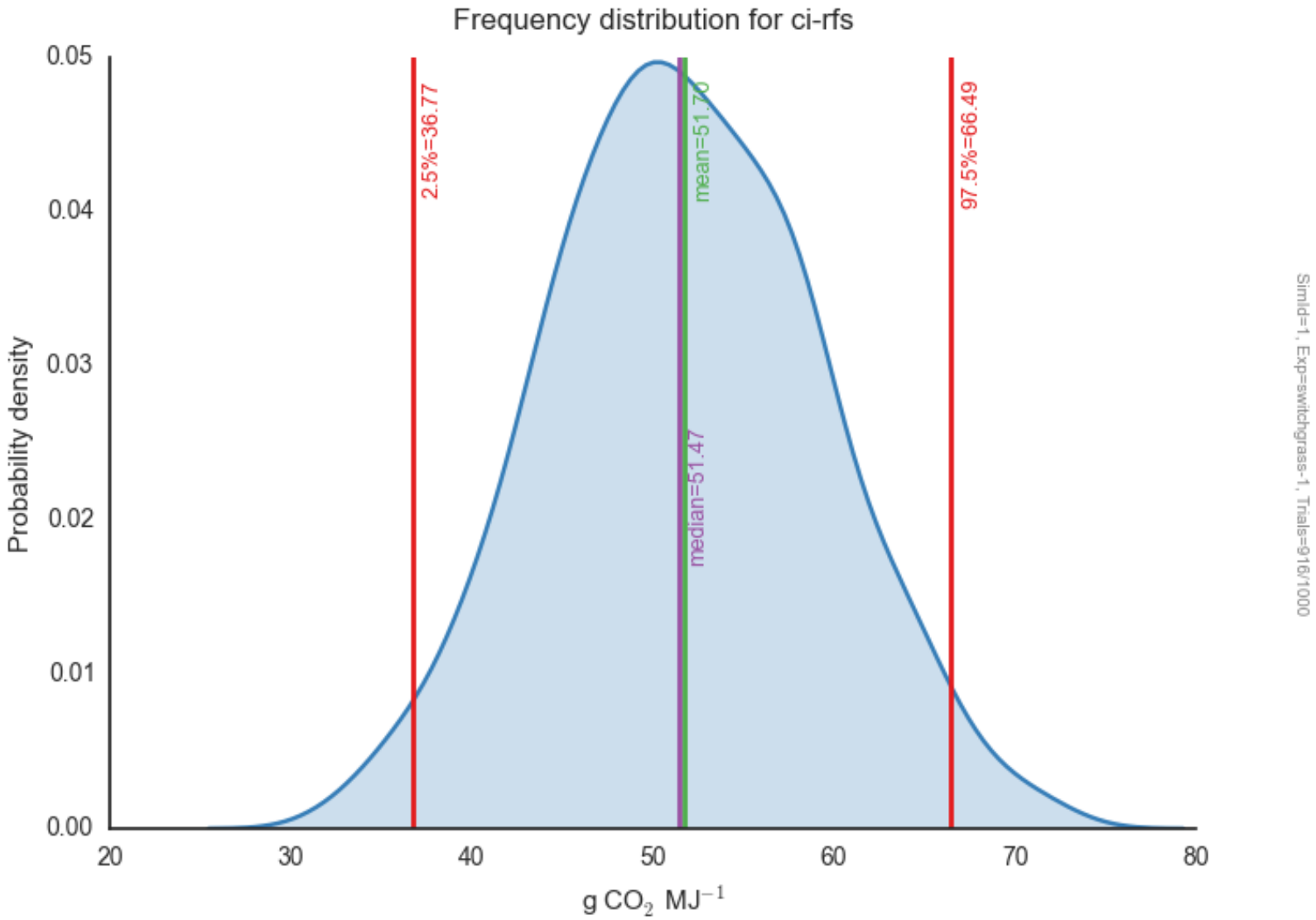
```
<ResultList>
    <Result name="ci-rfs" type="diff"
            desc="Carbon intensity, g CO2e/MJ">
        <File name="carbon_intensity"/>
        <Column name="value"/>
        <Constraint column="method" op="equal" value="ci-rfs"/>
    </Result>

    <Result name="delta-forcing" type="diff"
            desc="Change in RF (W/m^2)">
        <File name="Climate_forcing"/>
    </Result>
</ResultList>
```

EXAMPLE SCENARIO DEFINITIONS

Scenario files can be generated dynamically from baseline

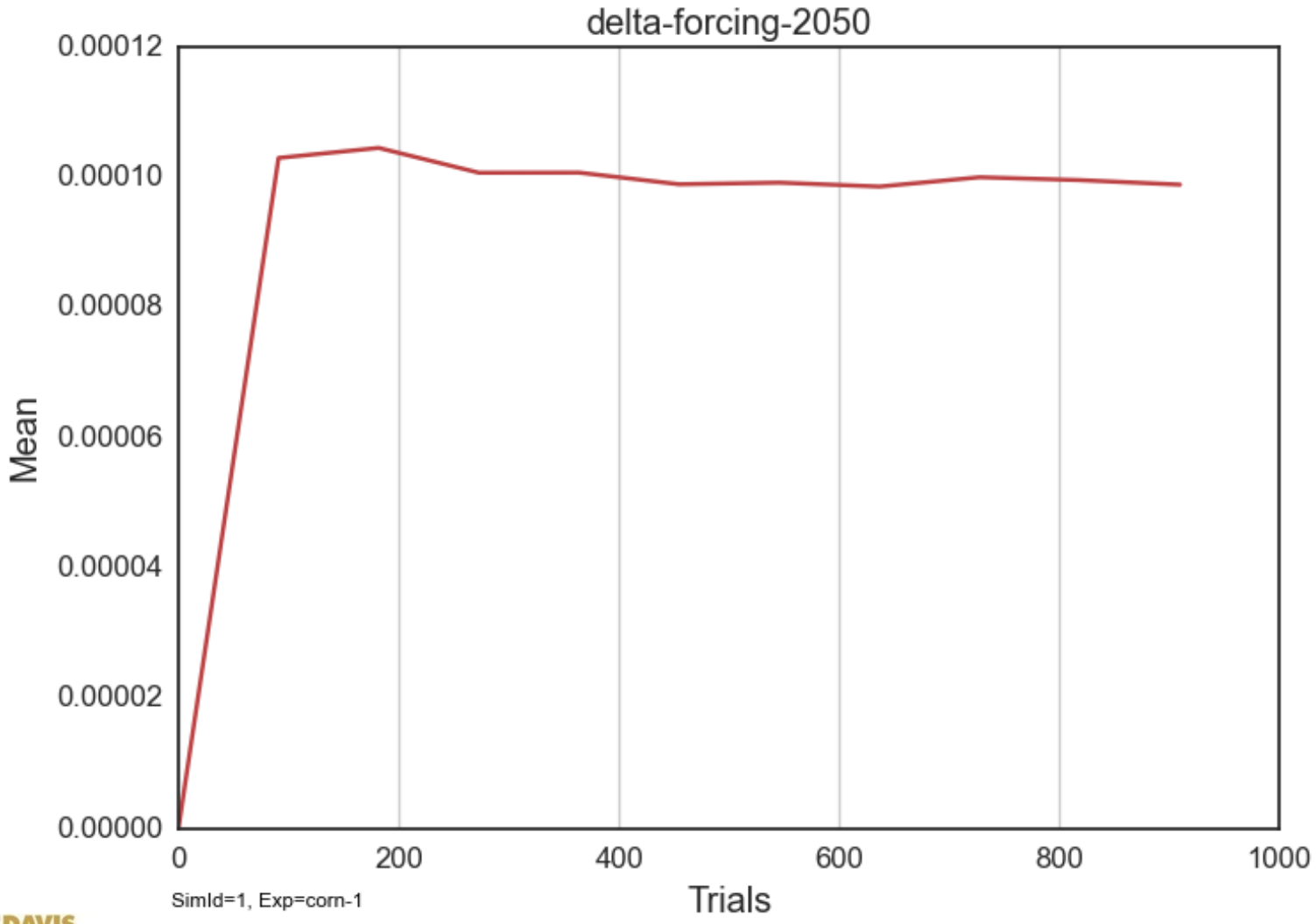
```
<Scenarios>
  <Scenario name="base-1" type="baseline">
    <Description>Paper1 control case</Description>
  </Scenario>

  <Scenario name="corn-1" parent="base-1">
    <Description>Corn ethanol shock</Description>
  </Scenario>

  <Scenario name="switchgrass-1" parent="base-1">
    <Generator>
      genConstraints.py
            --baseline {baseline}
            --policy {policy}
            --xmlOutputDir "{trialDir}/dyn-xml"
    </Generator>
    <Description>Switchgrass ethanol shock</Description>
  </Scenario>
</Scenarios>
```

DISTRIBUTION PLOTS

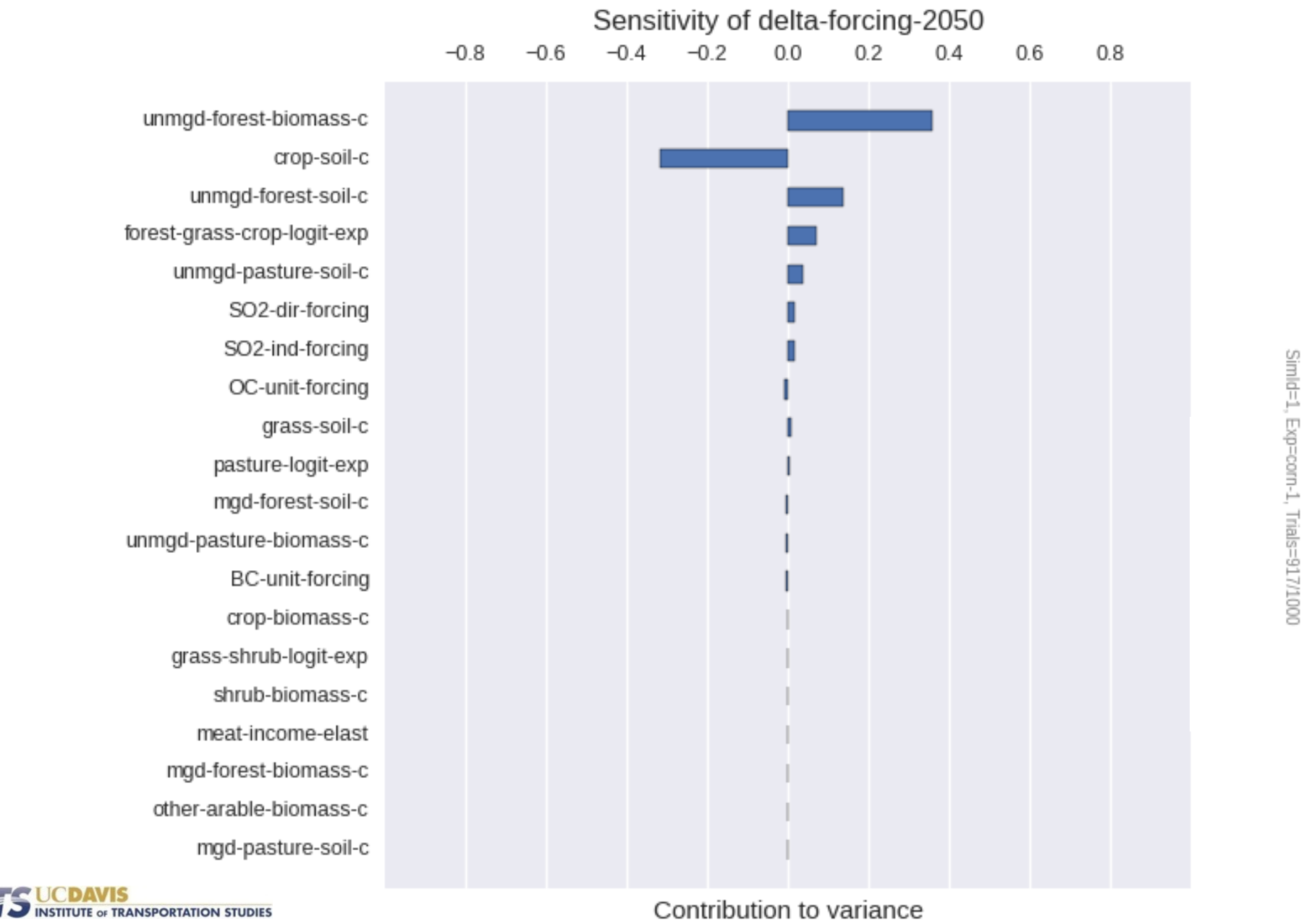


CONVERGENCE PLOTS

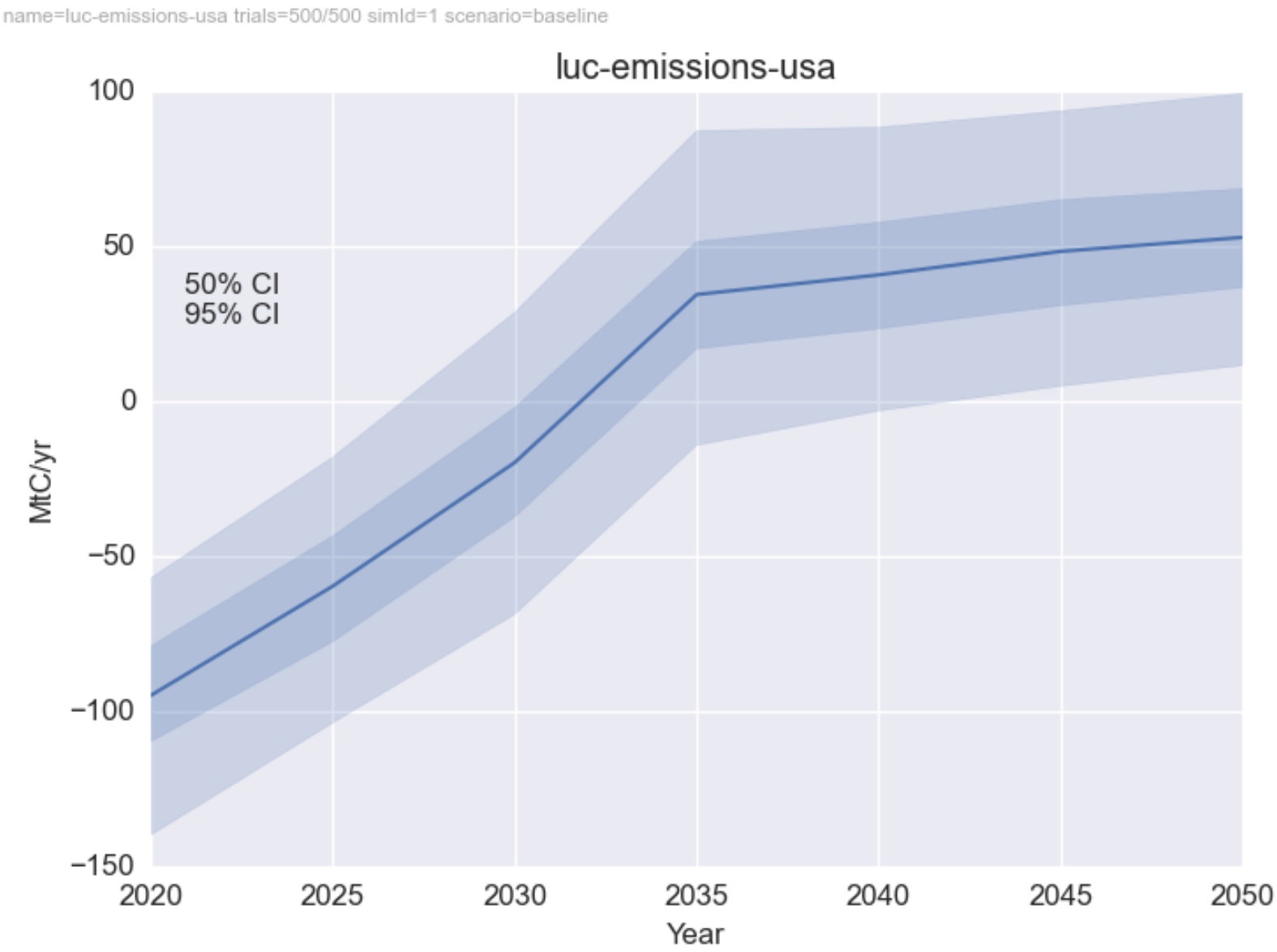
Show convergence of mean, stdev, skewness, 95% CI



UNCERTAINTY IMPORTANCE – CORN ETHANOL ΔRF IN 2050



SAMPLE TIME-SERIES PLOT



Features in common with MCS

- Generate input parameters
- Run on computing cluster
- Collect results into database
- Analyze results

Additional features

- Non-probabilistic methods
- PRIM or similar analysis
- Visualization features

Suite of flexible Python scripts:

- **queueGCAM**: Run GCAM in new workspace, locally or by queueing on Linux cluster
- **batchQuery**: Query GCAM's database to produce a CSV files with on-the-fly region specification & aggregation
- **csvDiff**: Generate a CSV/XLSX file of difference between GCAM query results with optional annual interpolation
- **chartGCAM**: Plots data from GCAM-style CSV files with several plot styles with various aesthetic options

https://bitbucket.org/plevin/gcam-utils