Reduced Order Models (ROMs)

RAVEN Workshop

www.inl.gov

Idaho National Laboratory

Outline

- Brief introduction on ROMs
- Application examples of ROMs
- ROMs and RAVEN
  - Available ROMs
  - RAVEN ROM workflow
- RAVEN examples
  - Create ROMs
  - Perform sampling of ROMs

ROMs: a Quick Introduction

- Consider a set of *N* data points

Output: Simulation outcome (success/failure, max clad temperature)

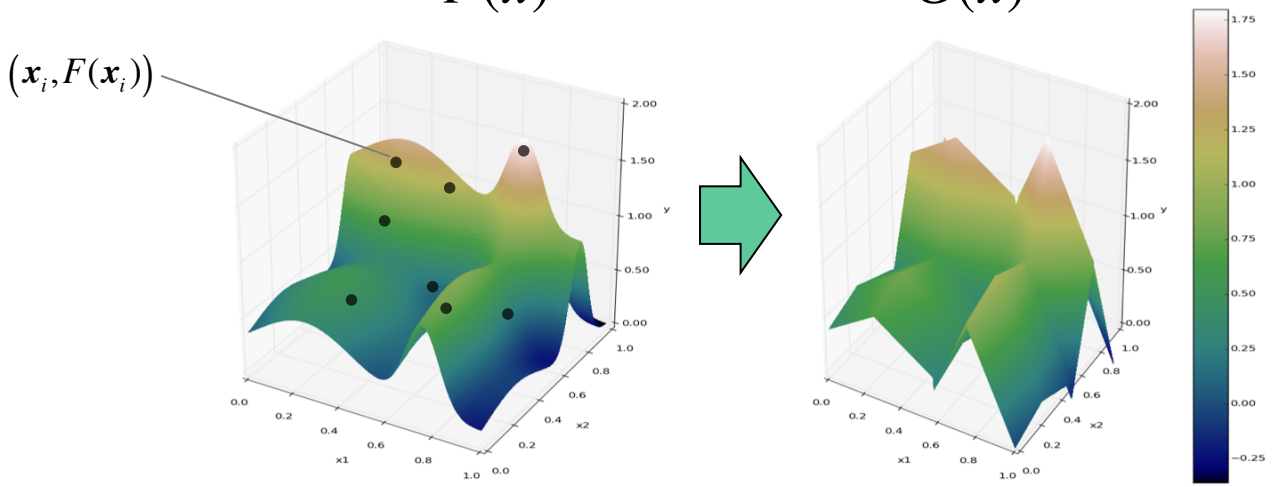
DataObjects (PointSet)

$$\left(x_i, F(x_i)\right) \quad i = 1,\ldots,N$$

Inputs: Initial and boundary conditions

- Build a surrogate model
  - Reduced Order Model $G(x) : x_i \longrightarrow G(x_i) \cong \mathrm{F}(x_i)$

$F(x)$ $G(x)$



ROMs: a Quick Introduction

- Basically we are trying to reduce the complexity of the original model

$$G(x) : x_i \longrightarrow G(x_i) \cong \mathrm{F}(x_i)$$

ROM (e.g., quadratic regressor)

Original data (e.g., RELAP5)

- Pros:
  - Much faster computation of the output variable
- Cons:
  - Presence of error in the ROM computed values

Classes of ROMs

- Model-Based
  - Prediction is performed using a blend of interpolation and regression algorithms
  - Examples:
    - Gaussian Process Models (GPMs)
    - Support Vector Machines (SVM)

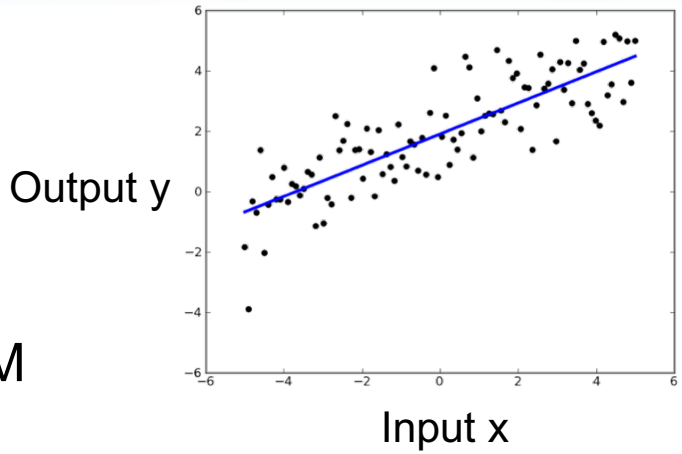
- Data-Based
  - Prediction is performed by solely considering the input data by using data searching algorithms
  - Example:
    - K nearest neighbor (KNN)

ROMs: Applications

- Basic steps:
  1. Sample original model
  2. Train the ROM

$$y = mx + c$$

  3. Perform desired analysis with the ROM instead of the original model



- Range of applications:
  - Uncertainty quantification / Sensitivity analysis
  - Probabilistic Risk Analysis (PRA)
  - Accelerator for stochastic analysis (adaptive sampling)
  - Prediction models

ROMs Available in RAVEN

- External dependency: Scikit-learn (http://scikit-learn.org)
  - Open source machine learning library for `Python`
  - Library for data mining and data analysis
  - Built on `NumPy` and `SciPy`

- Internally `C++` developed libraries: CROW
  - Generalized Polynomial Chaos
  - Multi-dimensional interpolators
  - Synthetic Time-Series Algorithms (e.g. F-ARMA)
  - Dynamic Mode Decomposition-based algorithms
  - Polynomial Regression (Poly-Tensor spline, Poly-exponential, etc.)

Supervised and Unsupervised Library

- Available:
  - Classification: identifying to which category an object belongs
  - Regression: predicting a continuous-valued attribute
  - Data clustering: grouping similar objects into sets
  - Dimensionality reduction: reducing the number of random variables
  - Data pre-processing: feature extraction and normalization

- Examples:
  - Linear regression models
  - Support Vector Machines
  - Multi-Class classifiers
  - Naïve Bayes
  - Neighbors classifiers
  - Tree classifiers

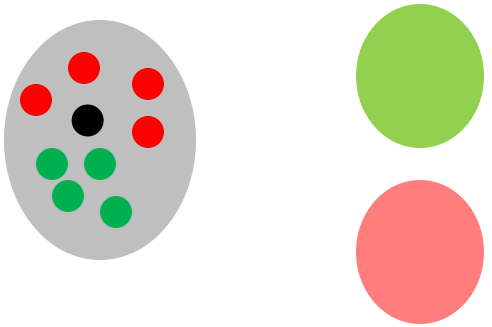
scikits learn
machine learning in Python

Supervised and Unsupervised Library

- Classification
  - Starting point: set of data points with labels

    $[\text{features, class}]_i$

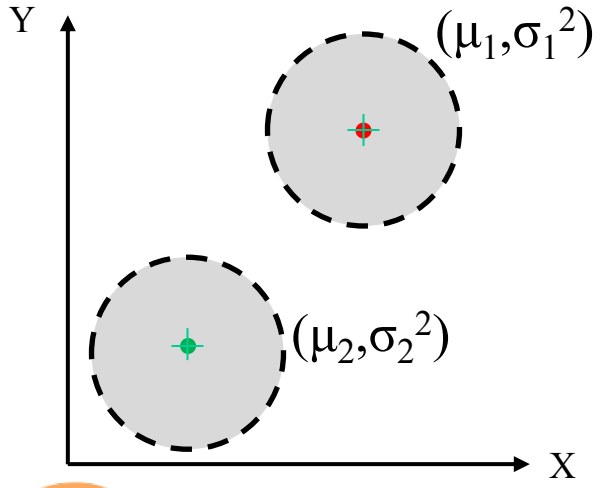
  - Objective: identify which class a new point [features] belong



- Clustering
  - Starting point: set of data points

    $[\text{features}]_I$

  - Objective: group data points based on a specific distance metrics



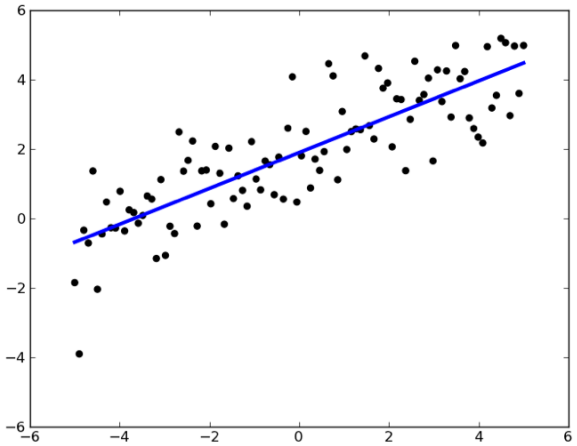
scikits learn
machine learning in Python

Supervised and Unsupervised Library

Starting point: set of data points

$[features]_i$

- Cardinality reduction
  - Objective: identify the most relevant features that keep data points unique
  - Outcome: Location of the points on the reduced space (e.g., line)



Source: scikit-learn.org

- Regression
  - Objective: estimate the relationships among variables via a statistical process
  - Outcome: coefficients of the reduced space (e.g., *m* and *c* for linear interpolator $y = mx + c$)

scikits learn
machine learning in Python

Generalized Polynomial Chaos

- Objective: overcome limitations of Monte-Carlo sampling
  - High number of samples
  - Computationally expensive

- Polynomial representation of an output variable $\Phi_k(Y)$
  - Simpler to evaluate
  - Easy to get statistical moments
  - Less effort and more accurate than Monte Carlo

$$u(Y) \approx \sum_{k \in \Lambda(L)} u_k \Phi_k(Y)$$

$$\Phi_k(Y) = \prod_{n=1}^{N} \Phi_{k_n}^{(n)}(y_n)$$

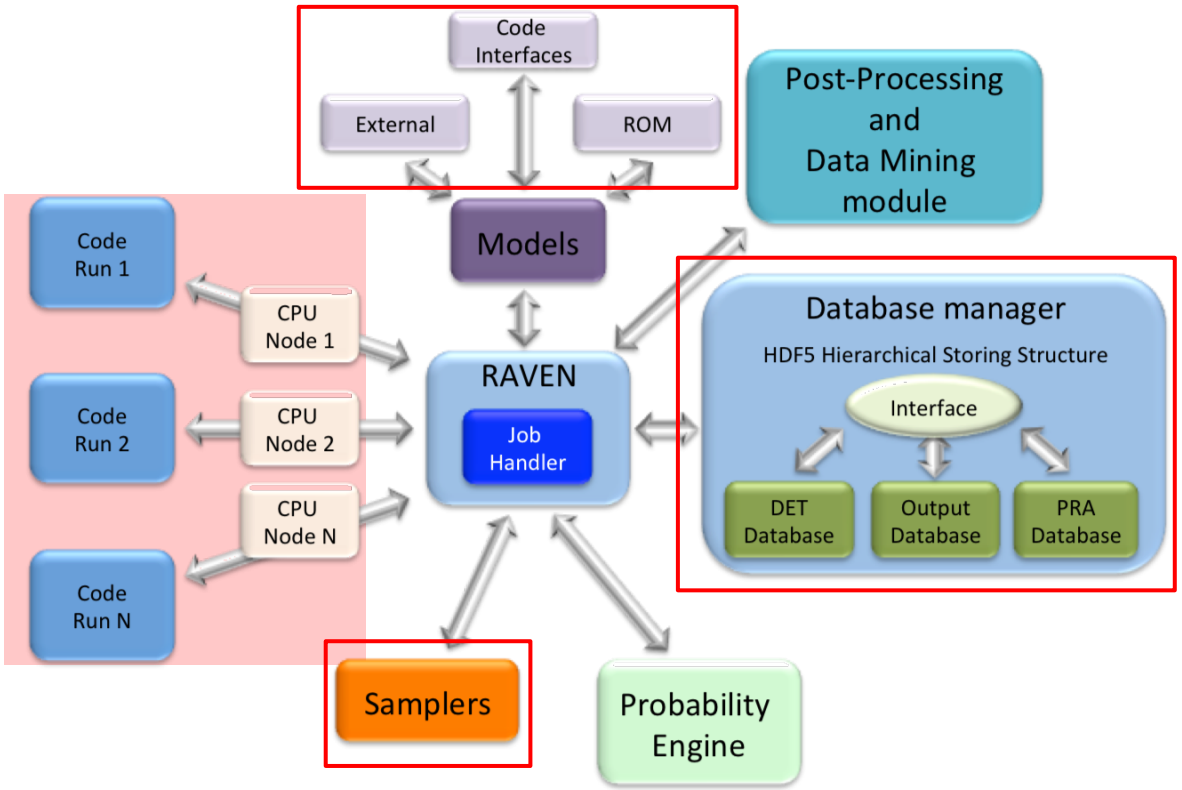
index set of all desired polynomial orders up to order L

Multi-Dimensional Interpolators

- CROW: Internally developed C++ library
- Interpolation on any dimension
- Response surface is created as an interpolation function given a set of data points defined on:
  - Sparse grid
  - Cartesian grid
- Extension of the known 1-D interpolation schemes

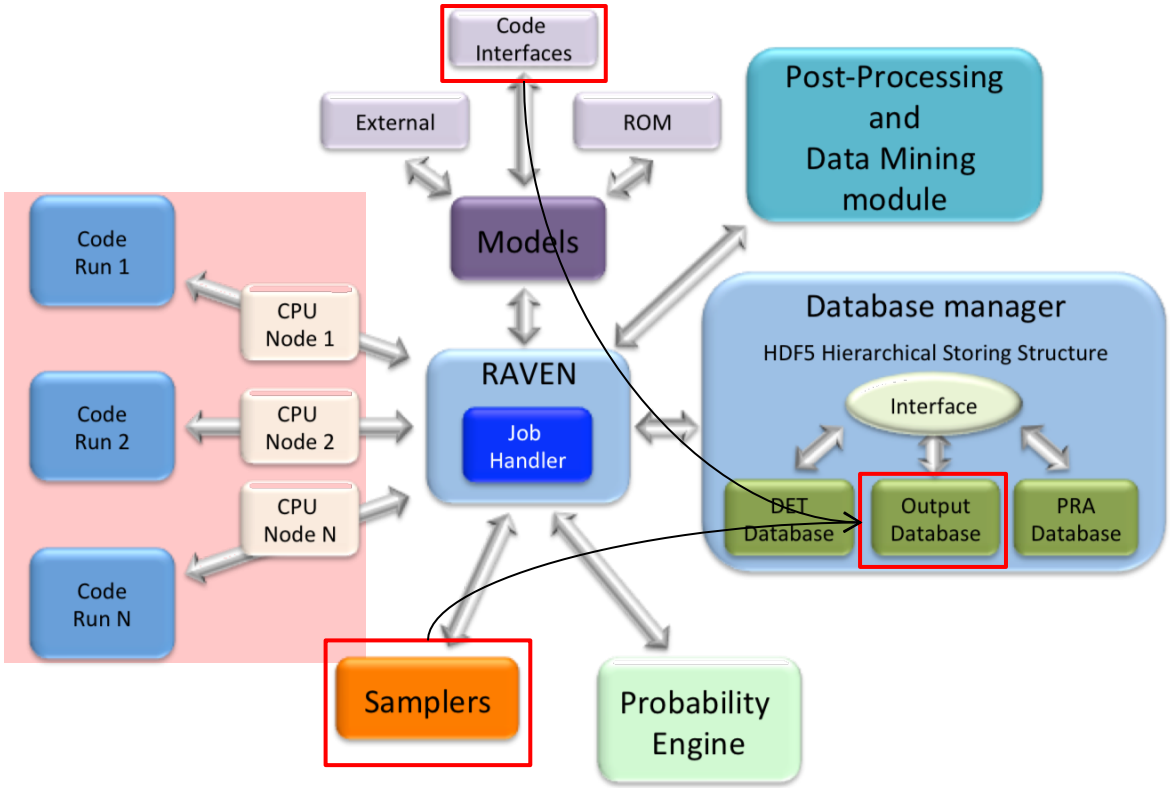
ROM Modeling Within RAVEN

- All modeling steps that involve ROMs are available in RAVEN
  - Create ROMs from a database
  - Perform statistical analysis using ROMs



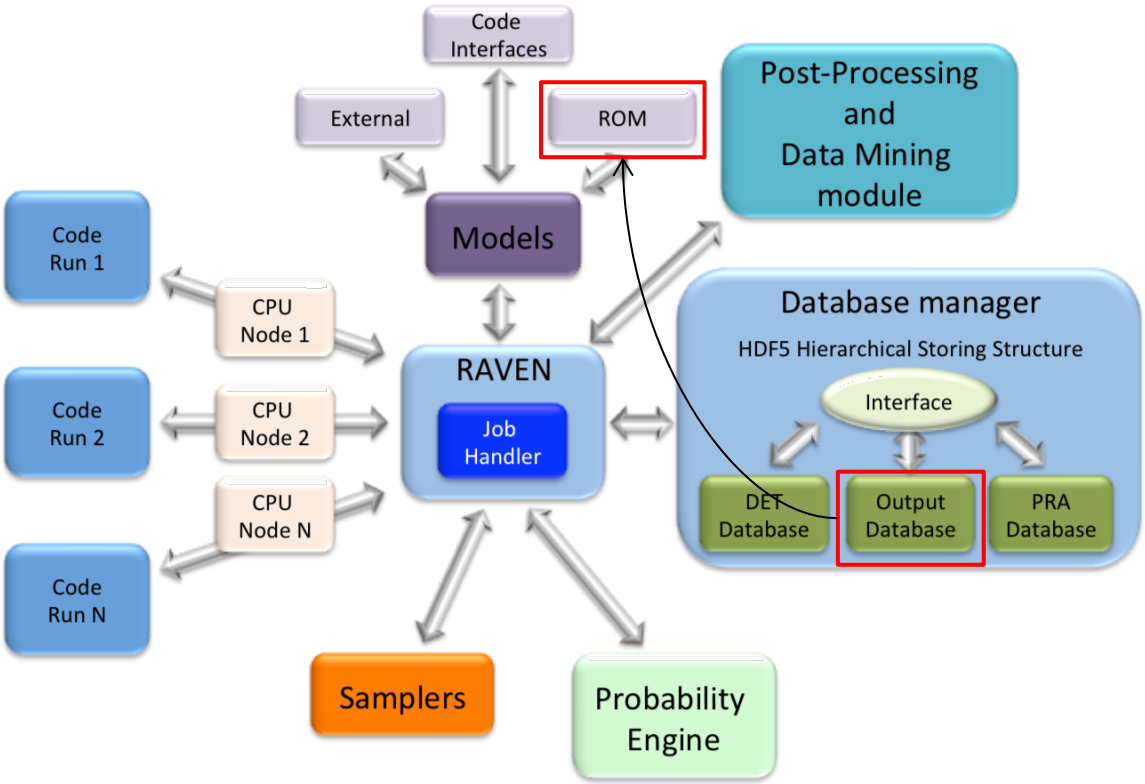
ROM Modeling Within RAVEN

- Create a Database (PointSet)



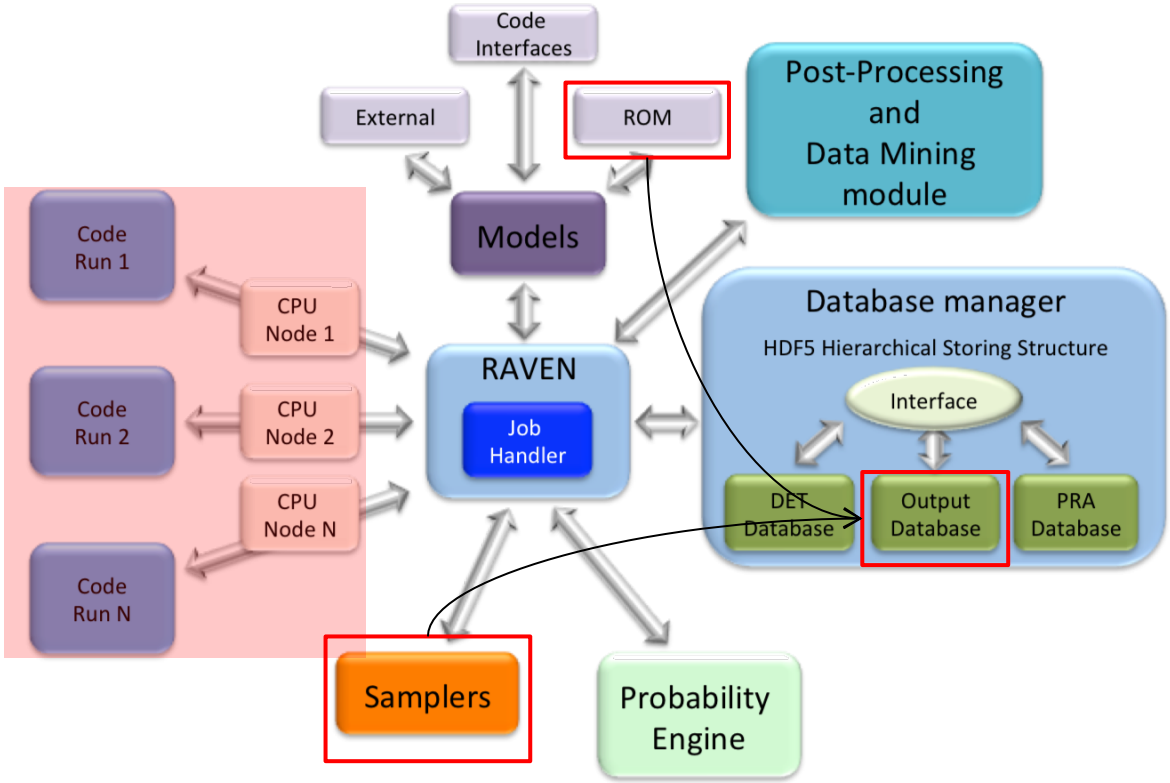
ROM Modeling Within RAVEN

- Create and train a ROM from a Database



ROM Modeling Within RAVEN

- Perform statistical analysis using the ROM



ROM Pickle

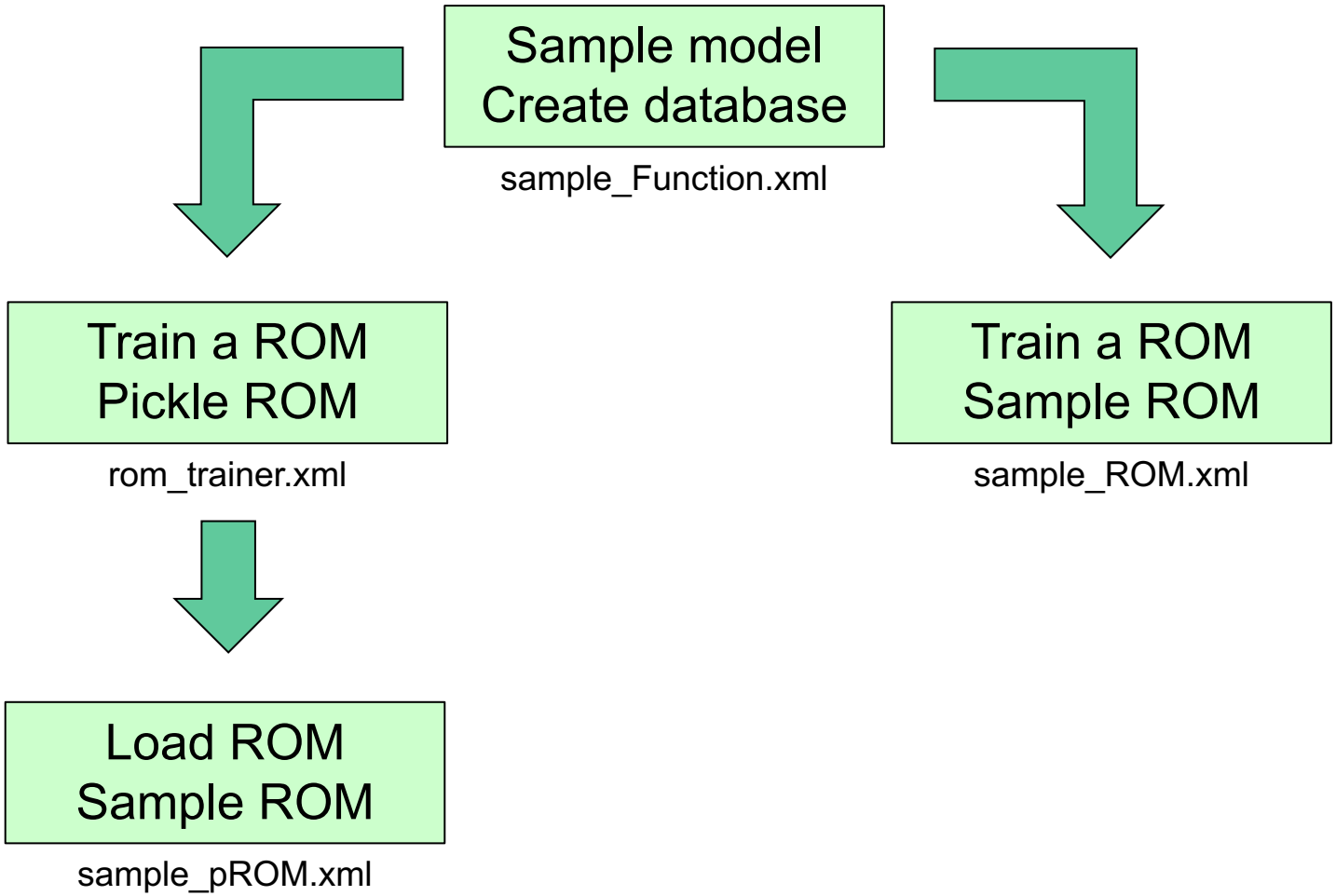
- Compression/serialization scheme
- Pickled object contains all the information necessary to reconstruct the object in another python script
- Pickled object can be saved as a file
- RAVEN ROMs can be pickled

- Applications:
  - Perform statistical analysis on a ROM after they have been generated and/or on a different machine
  - Use pickled ROMs on separate python script (external model for RAVEN)
  - Stochastic analysis for different distributions

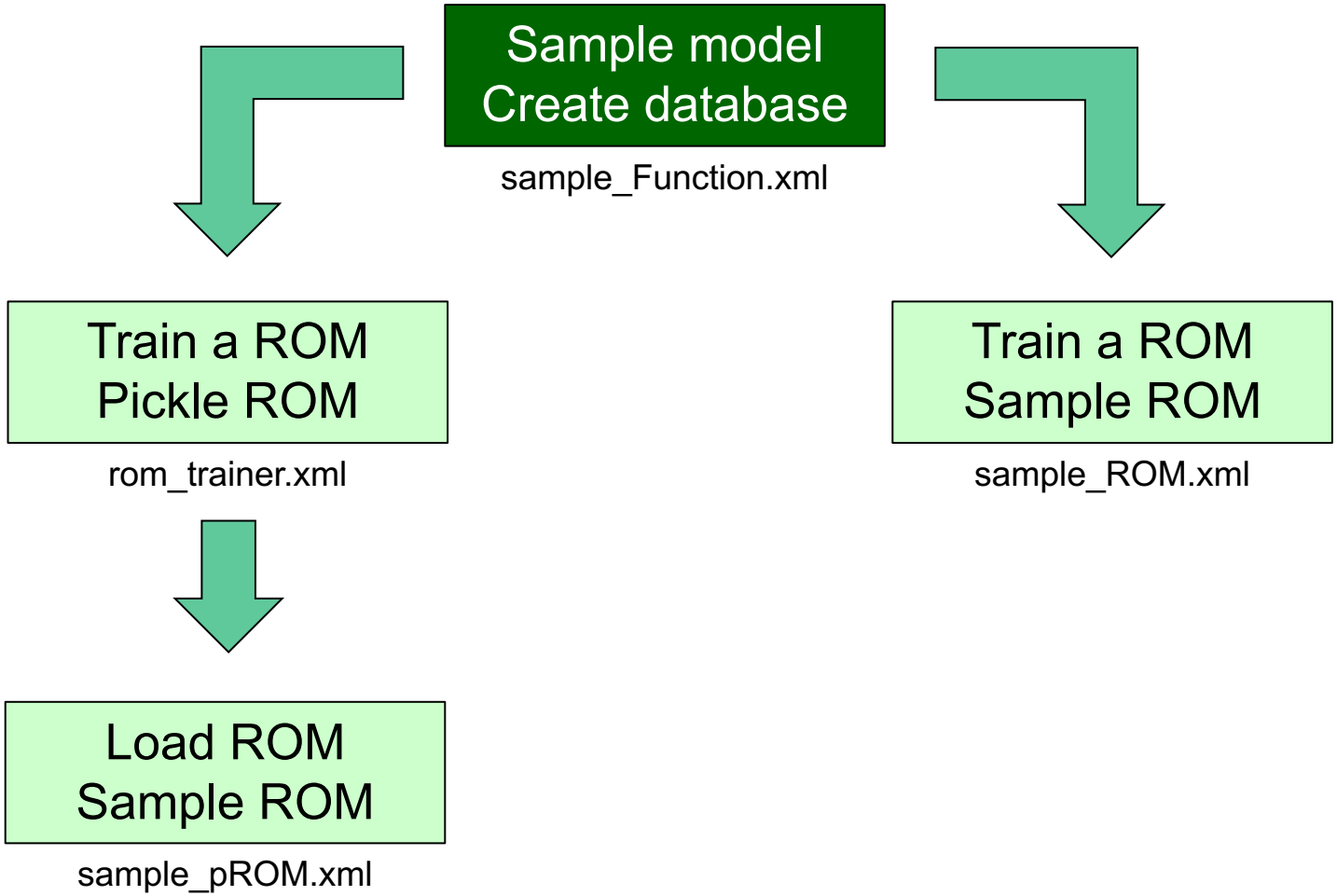
RAVEN Examples

Workflow

External model employed:
workshop_model.py



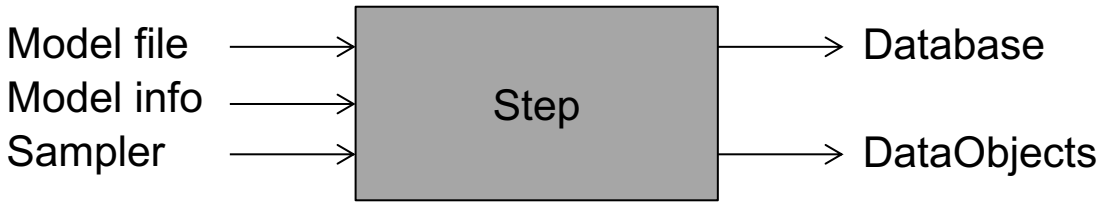
Workflow


Sample model
Create database
sample_Function.xml
Train a ROM
Pickle ROM
rom_trainer.xml
Train a ROM
Sample ROM
sample_ROM.xml
Load ROM
Sample ROM
sample_pROM.xml

Sample a Model and Create a Database

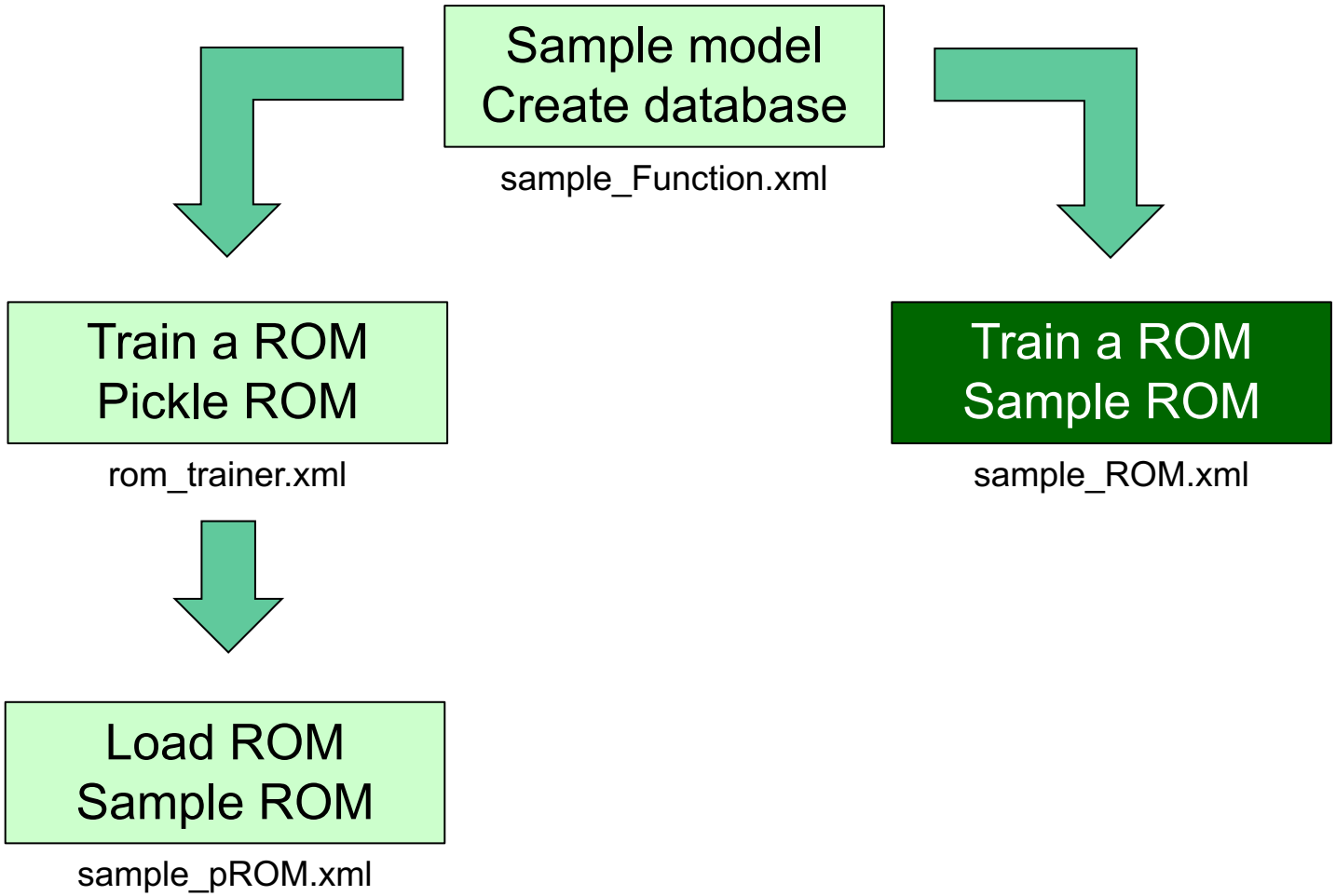
Distributions	Models	Samplers	Databases	DataObjects	Steps

```
<Steps>
  <MultiRun name="FirstMRun">
    <Input   class="DataObjects" type="PointSet"     >inputPlaceHolder</Input>
    <Model   class="Models"      type="ExternalModel">PythonModule</Model>
    <Sampler class="Samplers"    type="Grid"         >Grid_function</Sampler>
    <Output  class="DataObjects" type="PointSet"     >outGRID</Output>
    <Output  class="Databases"   type="HDF5"         >out_db</Output>
    <Output  class="OutStreams"  type="Print"        >out_dump</Output>
    <Output  class="OutStreams"  type="Plot"         >plotResponseFunction</Output>
  </MultiRun>
</Steps>
```



Input file name: sample_Function.xml

Workflow


Sample model
Create database
sample_Function.xml
Train a ROM
Pickle ROM
rom_trainer.xml
Train a ROM
Sample ROM
sample_ROM.xml
Load ROM
Sample ROM
sample_pROM.xml

Train and Sample a ROM

Distributions	Models	Samplers	Databases	DataObjects	Steps

```
<Models>
  <ROM name="ROM4" subType="NDinvDistWeight">
    <Features>x1,x2,x3</Features>
    <Target>y4</Target>
    <p>3</p>
  </ROM>
</Models>
```

Multi-dimensional interpolator (CROW)

Input file name: sample_ROM.xml

Train and Sample a ROM

Distributions	Models	Samplers	Databases	DataObjects	Steps

```
<Samplers>
  <Grid name="Grid_ROM">
    <variable name="x1">
      <distribution>normal_trunc</distribution>
      <grid type="value" construction="equal" steps="10">0.0 1.0</grid>
    </variable>
    <variable name="x2">
      <distribution>normal</distribution>
      <grid type="value" construction="equal" steps="10">1.5 2.5</grid>
    </variable>
    <variable name="x3">
      <distribution>uniform</distribution>
      <grid type="value" construction="equal" steps="10">1.0 4.0</grid>
    </variable>
  </Grid>
</Samplers>
```

Finer grid

Input file name: sample_ROM.xml

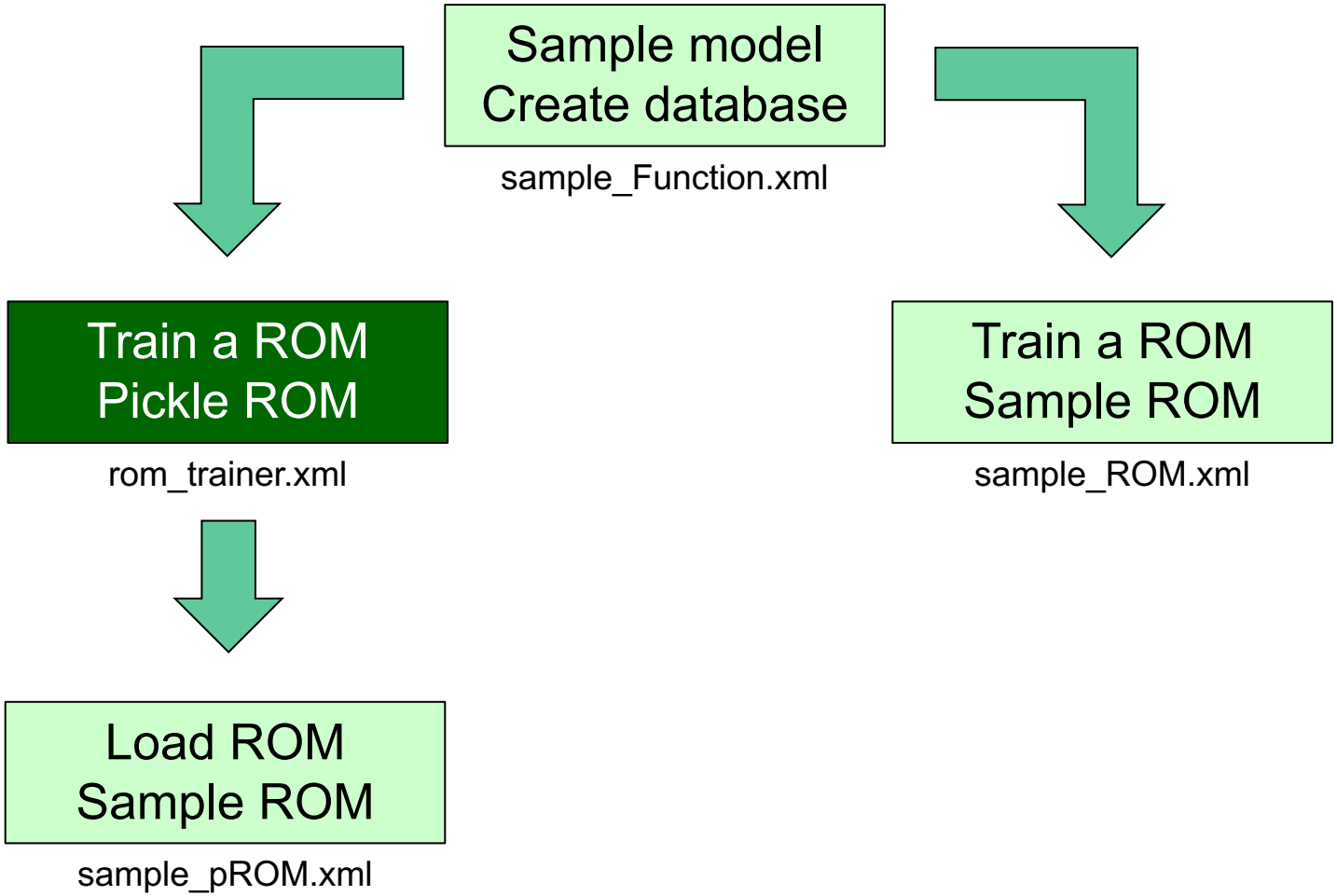
Train and Sample a ROM

Distributions	Models	Samplers	Databases	DataObjects	Steps

```
<Steps>
  <IOStep name="extract_data4">
    <Input   class="Databases"   type="HDF5"       >out_db</Input>
    <Output  class="DataObjects" type="PointSet"   >outGRID_y4</Output>
  </IOStep>
  <RomTrainer name="rom_trainer4">
    <Input   class="DataObjects" type="PointSet"   >outGRID_y4</Input>
    <Output  class="Models"      type="ROM"        >ROM4</Output>
  </RomTrainer>
  <MultiRun name="RunRom4">
    <Input   class="DataObjects" type="PointSet"   >Data1</Input>
    <Model   class="Models"      type="ROM"        >ROM4</Model>
    <Sampler class="Samplers"    type="Grid"       >Grid_ROM</Sampler>
    <Output  class="DataObjects" type="PointSet"   >outROM_y4</Output>
  </MultiRun>
  ...
</Steps>
```

Input file name: sample_ROM.xml

Workflow


Sample model
Create database
sample_Function.xml
Train a ROM
Pickle ROM
rom_trainer.xml
Train a ROM
Sample ROM
sample_ROM.xml
Load ROM
Sample ROM
sample_pROM.xml

Train and Pickle a ROM

Models	Databases	DataObjects	Steps

```
<Models>
    <ROM  name="ROM3" subType="SciKitLearn">
      <Features>x1,x2,x3</Features>
      <Target>y3</Target>
      <SKLtype>linear_model|LinearRegression</SKLtype>
      <fit_intercept>True</fit_intercept>
      <normalize>False</normalize>
    </ROM>
</Models>
```

Inputs / Output

ROM type and parameters

Input file name: rom_trainer.xml

Train and Pickle a ROM

Models	Databases	DataObjects	Steps

```
<Steps>
  <IOStep name="extract_data3">
    <Input   class="Databases"   type="HDF5"        >out_db</Input>
    <Output  class="DataObjects" type="PointSet"    >outGRID_y3</Output>
  </IOStep>
  <RomTrainer name="rom_trainer3">
    <Input   class="DataObjects" type="PointSet"    >outGRID_y3</Input>
    <Output  class="Models"      type="ROM"         >ROM3</Output>
  </RomTrainer>
  <IOStep name="pkDump3">
    <Input   class="Models"   type="ROM"  >ROM3</Input>
    <Output  class="Files"    type=""     >ROM3pk</Output>
  </IOStep>
</Steps>
```

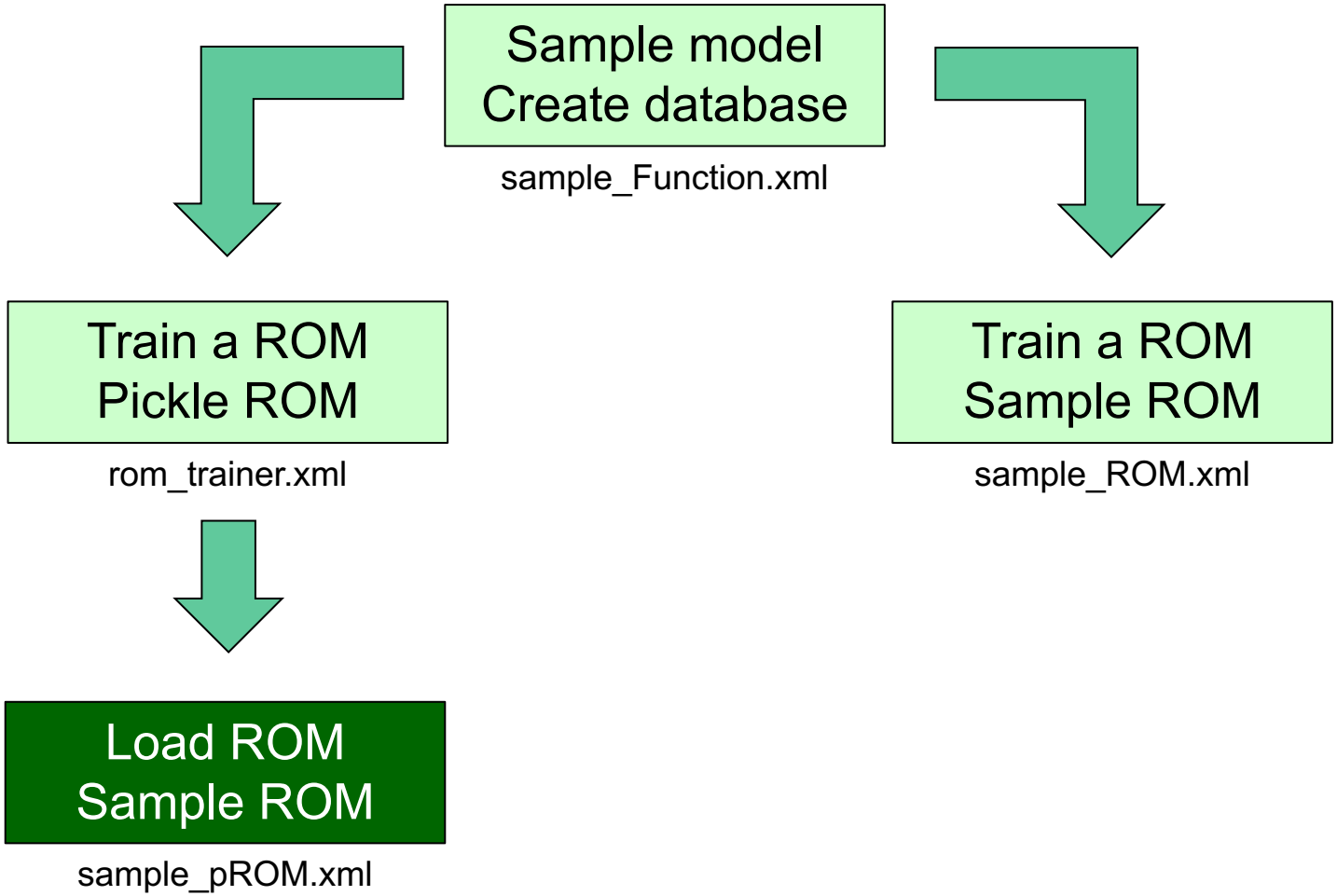
Create a DataObject from the Dababase

Create and train a ROM from the DataObject

Pickle the ROM and save it as a file

Input file name: rom_trainer.xml

Workflow


Sample model
Create database
sample_Function.xml
Train a ROM
Pickle ROM
rom_trainer.xml
Train a ROM
Sample ROM
sample_ROM.xml
Load ROM
Sample ROM
sample_pROM.xml

Load and Sample a Pickled ROM

Distributions	Models	Samplers	DataObjects	Steps

```
<Models>
    <ROM  name="ROM3" subType="SciKitLearn">
      <Features>x1,x2,x3</Features>
      <Target>y3</Target>
      <SKLtype>linear_model|LinearRegression</SKLtype>
      <fit_intercept>True</fit_intercept>
      <normalize>False</normalize>
    </ROM>
</Models>
```

Inputs / Output

ROM type and parameters

Input file name: sample_pROM.xml

Load and Sample a Pickled ROM

Distributions	Models	Samplers	DataObjects	Steps

Load the pickled ROM

```
<Steps>
  ...
  <IOStep name="pk3Load">
    <Input    class="Files"       type=""          >ROM3pk</Input>
    <Output   class="Models"      type="ROM"       >pROM3</Output>
  </IOStep>
  <MultiRun name="RunPROM3">
    <Input    class="DataObjects" type="PointSet"  >Data1</Input>
    <Model    class="Models"      type="ROM"       >pROM3</Model>
    <Sampler  class="Samplers"    type="Grid"      >Grid_ROM</Sampler>
    <Output   class="DataObjects" type="PointSet"  >outPROM_y3</Output>
    <Output   class="Databases"   type="HDF5"      >out_ROM3_db</Output>
  </MultiRun>
  ...
</Steps>
```

Input file name: sample_pROM.xml