Idaho National Laboratory

RAVEN Statistical Framework

RAVEN Entities and Input Structure introduction

Idaho National Laboratory - Idaho Falls, ID

Objectives

- Learn the “Entities” of a generic statistical analysis
- Learn how these “Entities” are implemented in RAVEN
- Learn the concept of RAVEN “Step”
- Learn how RAVEN Steps and Entities are assembled in the input file

- Basically, you should be able to start playing with RAVEN

- Additional info
  - RAVEN user manual (user guide)
  - Input files shown in this workshop
  - RAVEN regression tests

Statistical Analysis

- Generic term that includes
  - Generating data
  - Collecting data
  - Analyzing data

- Possible directions
  - Describe the nature of the data to be analyzed
  - Explore the relation of the data to the underlying population
  - Create a model to summarize understanding of how the data relates to the underlying population
  - Prove (or disprove) the validity of the model
  - Employ predictive analytics to run scenarios that will help guide future actions

Statistical Analysis: Examples

- Propagation of uncertainties in a code given a set of distributions
- Creation of a surrogate model
- Perform the sampling of a multi-physics code
- Perform probabilistic risk analysis (PRA) of a PWR accident scenario
- Understand input-output correlations of large data sets (Data Mining)
- Reduce the complexity of a model (Dimensionality Reduction)

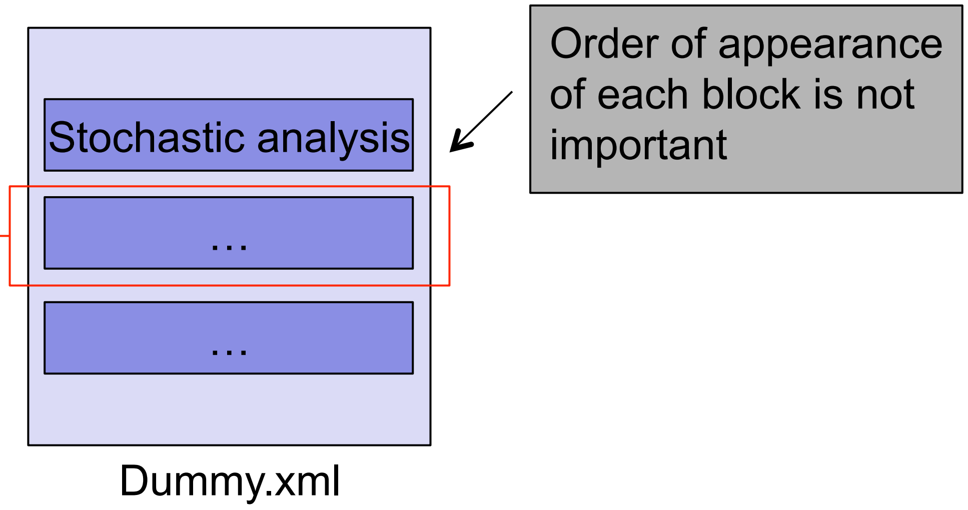
Statistical Analysis: Entities

- From the previous slide I mentioned
  - Code
  - Surrogate Model
  - Data
  - Distribution
  - Sampler

- Note
  - This is not a “one step” process
    - Several steps can be performed in a single stochastic analysis
  - Several Entities can coexist
    - Multiple codes
    - Multiple samplers
    - …

Statistical Analysis: the RAVEN Approach

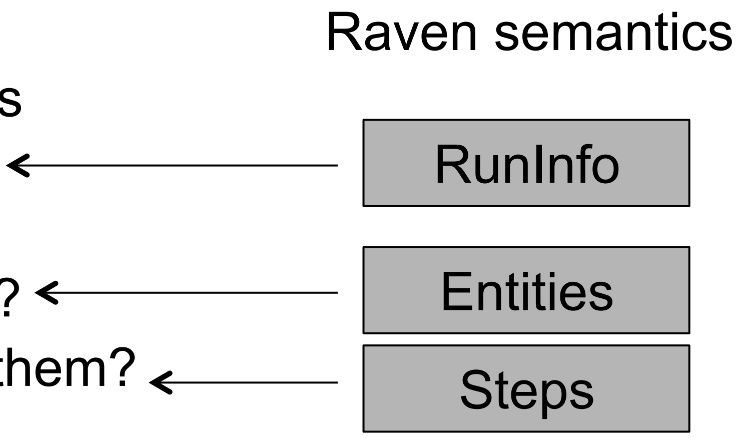
- One single input file
  - High modular input style
  - .xml format



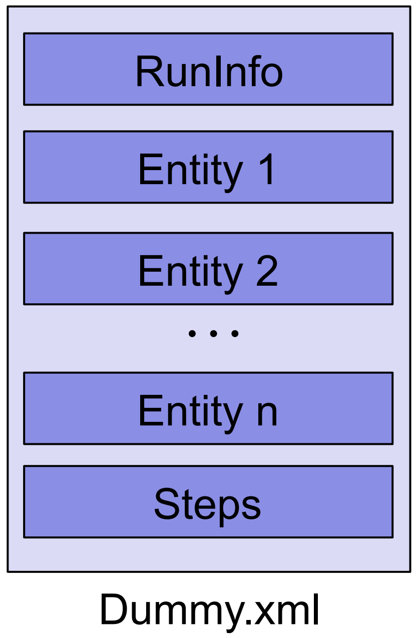
```
<Samplers>
    <Stratified name='test'>
        <samplerInit>
            <seed>1234</seed>
        </samplerInit>
    </Stratified >
</Samplers>
```

Statistical Analysis: the RAVEN Approach

- Type of information
  - Desired stochastic analysis
    - What do I want to do?
  - Entities needed
    - What do I want to use?
    - How do I want to use them?

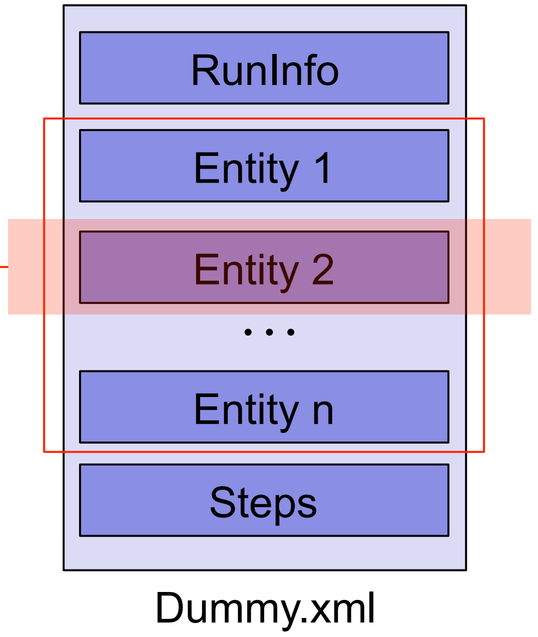


- Template of RAVEN input file



Entities

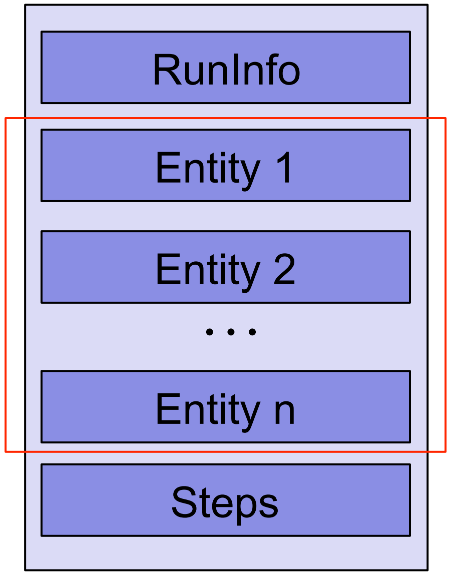
- Available types
  - DataObjects
  - Databases
  - Samplers
  - OutStreamManager
  - Distributions
  - Models
  - Functions



```
<Samplers>
    <SamplerType_1 name='aName1'>
       …
    </SamplerType_1>
    <SamplerType_2 name='aName2'>
       …
    </SamplerType_2>
</Samplers>
```

Entities

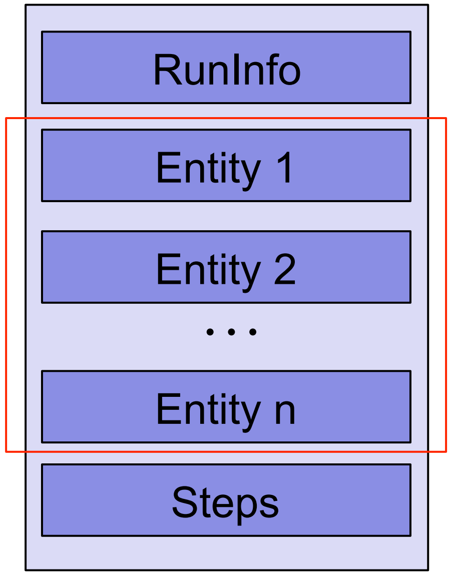
- DataObjects: how data is stored within RAVEN
  - Format: (input params, output params)
  - Static data: TimePoint and TimePointSet
  - Time dependent: History and Histories
- Databases: data storage entities
  - Store data in binary format
  - HDF5 files
  - DataObjects can be saved into Databases
  - Existing Databases can be loaded into the RAVEN framework



Dummy.xml

Entities

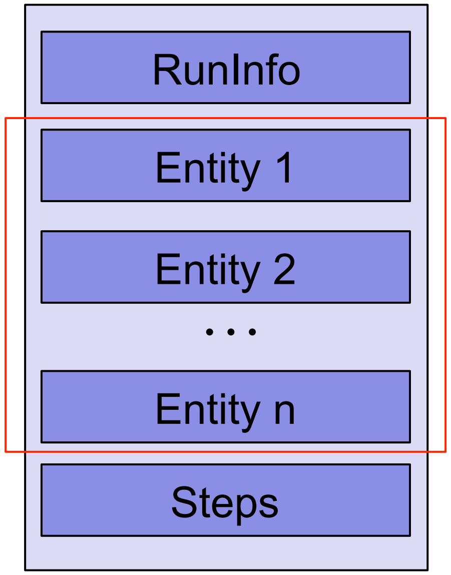
- Samplers: input space sampling entities
  - Forward Samplers: Monte-Carlo, Stratified (LHS), Grid, Response Surface, Factorial Design, etc…
  - Adaptive Samplers (smart sampling)
  - Dynamic Event Tree Samplers
- OutStreamManager: used for data exporting/dumping
  - Printing:
    - DataObjects
    - Reduced Order Models (ROMs)
  - Plotting: both 2D and 3D plotting available
    - 4D by using color mapping
    - 5D by using marker size



Dummy.xml

Entities

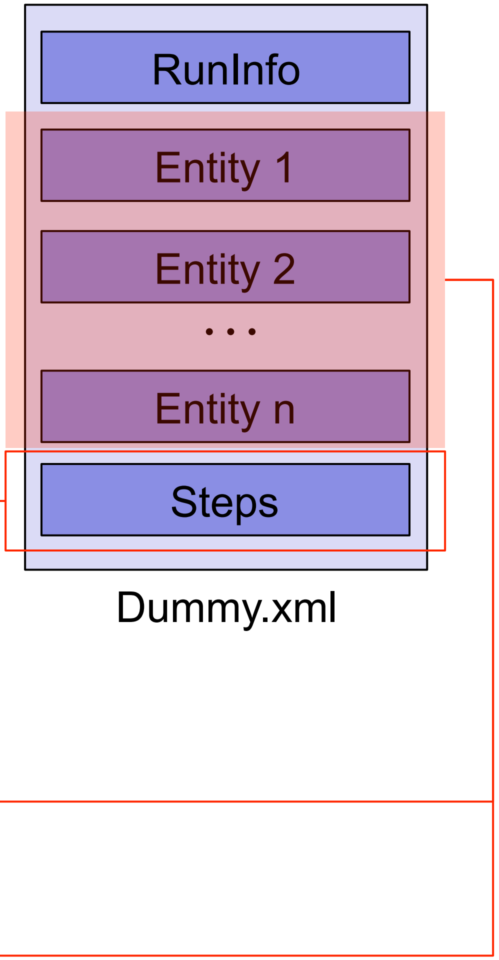
- Distributions: stochastic representation of variable
  - 1D: both continuous and discrete
  - ND: multi-dimensional distributions
- Models: projection from input to output space
  - Codes: through code interfaces
  - External models: python based module
  - Reduced Order Models (ROMs)
  - PostProcessors: used to perform action on data
    - Basic statistic operations
    - Comparison statistic
    - ….
- Functions: user-defined functions



Dummy.xml

Steps

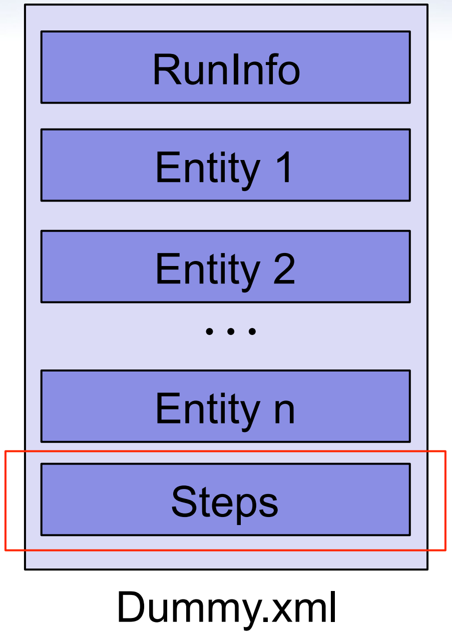
- A Step links Entities together to perform an action
- Multiple heterogeneous Entities are used in a single Step (DataObjects, Samplers, Models, …)
- All these Entities must be defined in their corresponding block
  - They can be defined after the Steps block



```
<Steps>
    <StepType1 name='simple_MultiRun'>
       …
    </StepType1>
    <StepType2 name='simple_PostProcess'>
       …
    </StepType2>
</Steps>
```

Steps

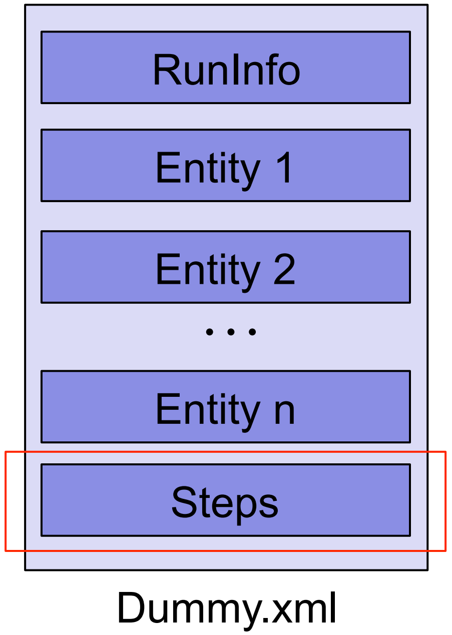
- Each Entity has a role
  - Input
  - Output
  - Model
  - Sampler
  - Function
  - ROM
  - Solution export



```
<Steps>
  ...
  <SingleRun name='StepName'>
    <Input  class='Files'         type=''     >anInputFile.i</Input>
    <Input  class='Files'         type=''     >anAuxiliaryFile</Input>
    <Model  class='Models'        type='Code'>aCode</Model>
    <Output class='Databases'     type='HDF5'>aDatabase</Output>
    <Output class='DataObjects'   type='History'>aData</Output>
  </SingleRun>
</Steps>
```

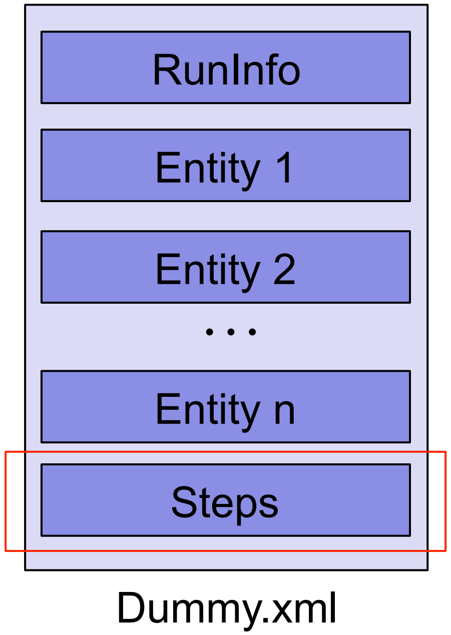
Step Types (1/2)

- SingleRun: perform a single run of a model
- MultiRun: perform multiple runs of a model
- RomTrainer: perform the training of a Reduced Order Model (ROM)
- PostProcess: post-process data or manipulate RAVEN entities



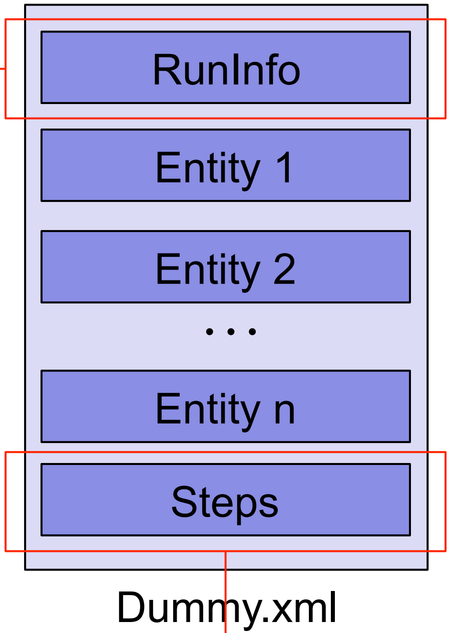
Step Types (2/2)

- IOStep:
  - construct/update a Database from a DataObjects and vice versa
  - construct/update a Database or a DataObjects object from CSV files
  - stream the content of a Database or a DataObjects out through an OutStream
  - store/retrieve a ROM to/from an external File using Pickle module of Python



RunInfo

- Desired stochastic analysis
  - Sequence of Steps
  - Working directory
  - Parallel computation parameters
  - …

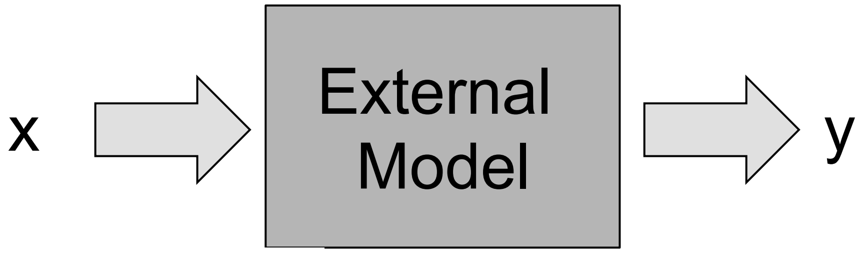


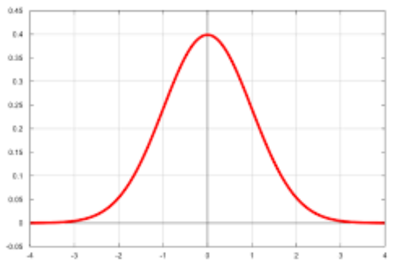
# of simultaneous code evaluations

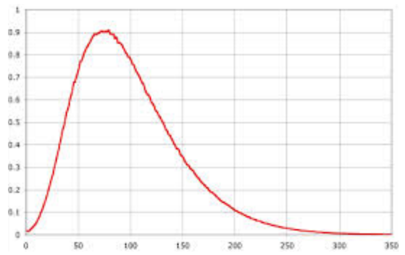
```
<RunInfo>
    <WorkingDir>./myDir</WorkingDir>
    <batchSize>6</batchSize>
    <Sequence>Step2,Step3,Step6</Sequence>
</RunInfo>
```

Example 1: Basic Test

- Sampling of a simple model (External model)
  - One uncertain parameter (input): *x*
  - One output parameter: *y*



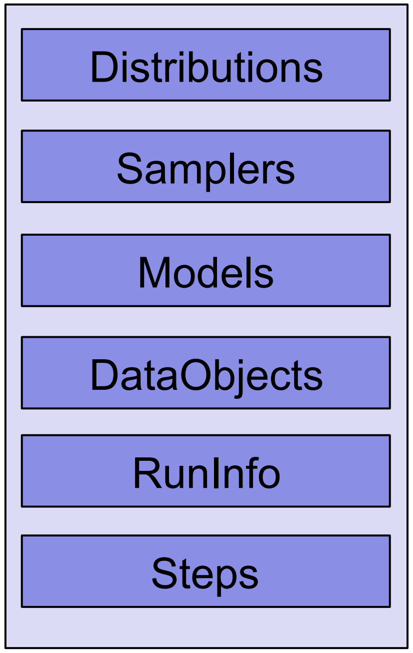




Basic Test: Input Structure

- Sampling of a simple model (External model)
- One uncertain parameter (input): *x*
- One output parameter: *y*

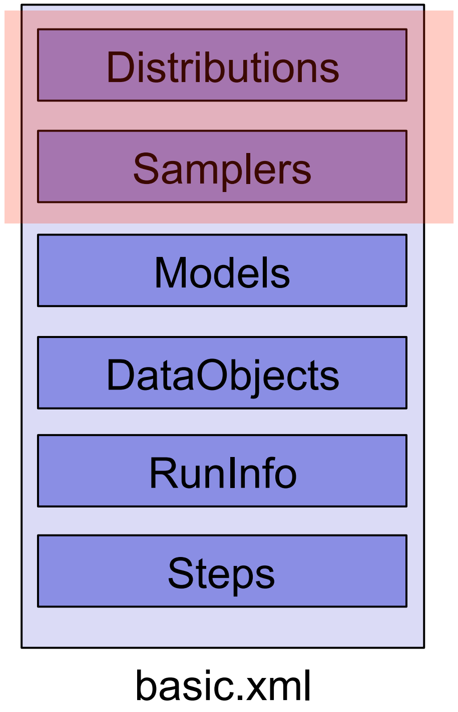
- Input layout



basic.xml

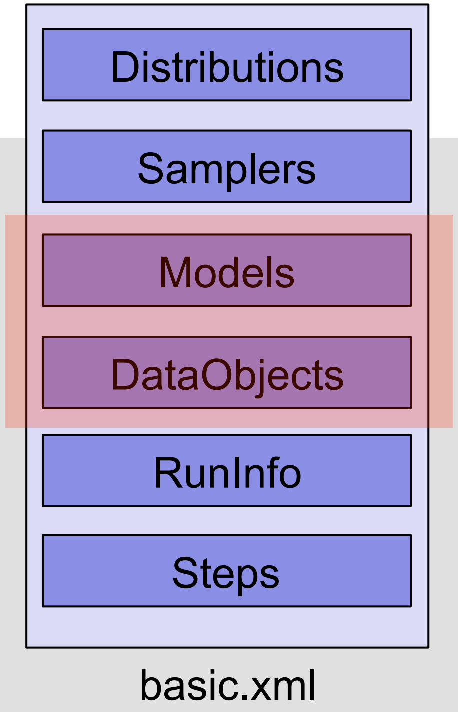
The Input Structure (1/3)

```
<Distributions>
  <Normal name='x_distrib'>
    <mean>2</mean>
    <sigma>0.2</sigma>
  </Normal>
</Distributions>

<Samplers>
  <MonteCarlo name='MCsampler'>
    <samplerInit>
      <limit>5000</limit>
     </samplerInit>
     <variable name='x'>
      <distribution>x_distrib</distribution>
     </variable>
  </MonteCarlo>
</Samplers>
```



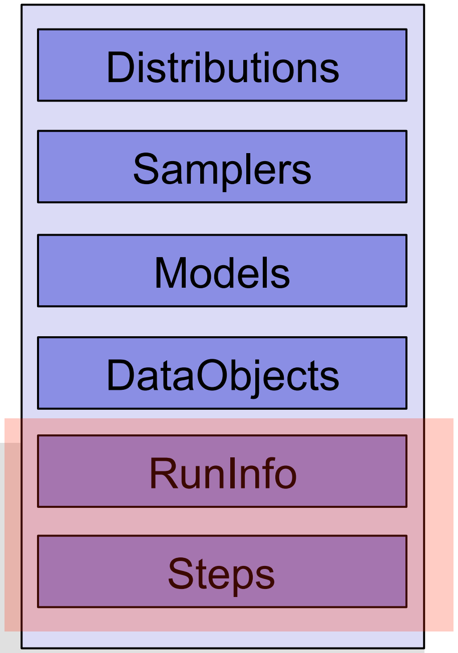
The Input Structure (2/3)

```
<DataObjects>
  <PointSet name='outSampler'>
    <Input>x</Input>
    <Output>y</Output>
  </PointSet >

  <PointSet name='dummy'>
    <Input>x</Input>
    <Output>OutputPlaceHolder</Output>
  </PointSet >
</DataObjects>


<Models>
  <ExternalModel name='aPythonModel' subType='' ModuleToLoad='./externalModel'>
    <variables>x,y</variables>
  </ExternalModel>
</Models>
```



The Input Structure (3/3)

```
<Runinfo>
  <WorkingDir>./myDir</WorkingDir>
  <Sequence>runMC</Sequence>
</Runinfo>

<Steps>
  <MultiRun  name='runMC'>
    <Input   name='DataObjects'   type='PointSet'      >dummy</Input>
    <Model   name='Models'        type='ExternalModel'>aPythonModel</Model>
    <Sampler name='Samplers'      type='MonteCarlo'    >MCsampler</Sampler>
    <Output  name='DataObjects'   type='PointSet'      >outMC</Output>
  </MultiRun>
</Steps>
```



basic.xml

Run RAVEN

- “Executable” file: raven_framework

```
user@ubuntu:~$ cd projects/raven
user@ubuntu:~/projects/raven$ ./raven_framework basic.xml
```

RAVEN Snapshots

- Utility examples that are often used

RAVEN Snapshots: Database Storage in RAVEN

- RAVEN framework provides the capability to store and retrieve data to/from an external database
- Database format: HDF5
- Data can be organized in two ways:
  - Parallel (e.g., if generated from Forward/Adaptive samplers)
  - Hierarchical (e.g., if generated from Dynamic Event Tree samplers)

Database that is going to be created

readMode ='overWrite':
New database. If present in the disk, it is going to be overwritten

```
<Databases>
  <HDF5 name='out_ROM4_db' readMode='overWrite'/>
  <HDF5 name='out_ROM5_db' directory='/pathToDatabase/' readMode='overWrite'/>
  <HDF5 name='out_db' filename='out_db.h5' readMode='read' directory='/
pathToDatabase/'/>
</Databases>
```

readMode ='read':
Existing database that is going to be retrieved

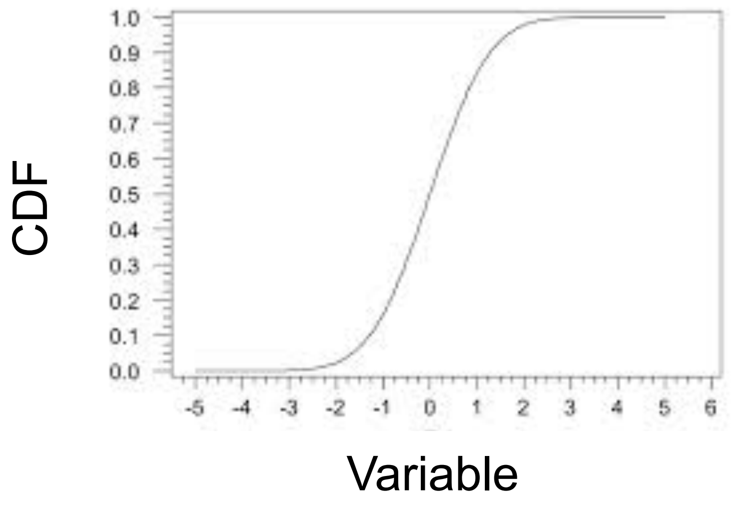
RAVEN Snapshots: Grid Sampling (1/2)

- Used to sample the input space using a cartesian grid scheme

- Sample each variable on
  - Value
  - Cumulative distribution function (CDF)

- Grid types
  - Custom
  - Equally spaced

- Mixing grid types is allowed



RAVEN Snapshots: Grid Sampling (2/2)

- Grid sampling on a 3-dimensional space

Equally spaced Grid

Lower and upper bounds (CDF)

```
<Grid name=’GridSampler’>
  <variable name=‘x1’>
    <distribution>x1_distrib</distribution>
    <grid type=‘CDF’ construction=‘equal’ steps=’10’>0 1</grid>
  </variable>
  <variable name=‘x2’>
    <distribution>x2_distrib</distribution>
    <grid type=‘value’ construction=‘equal’ steps=‘8’>0.9 1.7</grid>
  </variable>
  <variable name=‘x3’>
    <distribution>x3_distrib</distribution>
    <grid type=‘CDF’ construction=‘custom’>0.1 0.3 0.4 0.7 0.9</grid>
  </variable>
</Grid>
```

Number of bins

Lower and upper bounds (value)

Custom Grid

grid values

RAVEN Snapshots: External Models

```python
def run(self,Input):
   a = 1.0
   b = 2.0
   c = 3.0
   l = 1.0
   self.y1 = self.x1*self.x1 + self.x1*self.x2*self.x3
   self.y2 = math.exp(l*self.x1)
```

example.py

```xml
<Models>
  <ExternalModel name=‘aPythonModel’ subType=‘’  ModuleToLoad=‘./example’>
        <variables>x1, x2, x3, y1, y2</variables>
  </ExternalModel>
</Models>
```

Input variables

Output variables

basic.xml

RAVEN Snapshots: Print Data on .csv File

```
<DataObjects>
    <PointSet name='samples'>
        <Input>x1,x2,x3</Input>
        <Output>y1,y2</Output>
    </PointSet>
</DataObjects>

<OutStreamManager>
    <Print name='samples'>
        <type>csv</type>
        <source>samples</source>
    </Print>
</OutStreamManager>

<Steps>
    <MultiRun name="sample">
        <Input   class='DataObjects'  type='PointSet'      >input</Input>
        <Model   class='Models'       type='ExternalModel'>aPythonModel</Model>
        <Sampler class='Samplers'     type='MonteCarlo'    >MCsampler</Sampler>
        <Output  class='DataObjects'  type='PointSet'      >samples</Output>
        <Output  class='OutStreams'   type='Print'         >samples</Output>
    </MultiRun>
</Steps>
```

RAVEN Snapshots: Plotting Data

- Plot engine: Matplotlib

```
<OutStreamManager>
    <Plot name='plot' >
        <plotSettings>
            <plot>
                <type>scatter</type>
                <x>samples|Input|x1</x>
                <y>samples|Input|x2</y>
            </plot>
            <xlabel>x1</xlabel>
            <ylabel>x2</ylabel>
        </plotSettings>
        <actions>
            <how>screen</how>
            <title>
                <text>Sample Points Location</text>
            </title>
        </actions>
    </Plot>
</OutStreamManager>
```

RAVEN Snapshots: Basic Statistics

```
<PointSet name='outMC'>
  <Input>x1,x2,x3</Input>
  <Output>y1,y2,y3,y4,y5</Output>
</PointSet>

<Models>
  <PostProcessor name=‘StatisticsOutput’ subType='BasicStatistics'>
     <all>all
       <targets>y1,y2</targets>
       <features>x1,x2,x3</features>
     </all>
  </PostProcessor>
</Models>

<Steps>
  <PostProcess name='PP'>
     <Input  class='DataObjects’ type=’PointSet’ >outMC</Input>
     <Model  class='Models'      type='PostProcessor’>StatisticsOutput</Model>
     <Output class='Files'       type=''  >output_basicStatistics.xml</Output>
  </PostProcess>
</Steps>
```