In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import os

from ema_workbench import (Model, RealParameter, CategoricalParameter, BooleanParameter
, IntegerParameter, Constant, TimeSeriesOutcome, perform_experiments, ema_logging)

from ema_workbench.connectors.netlogo import NetLogoModel

from ema_workbench.em_framework.evaluators import LHS, FF

from ema_workbench.util.utilities import save_results, load_results

from ema_workbench.analysis.plotting import lines

import datetime

import os
```

```
C:\Users\stein042\.conda\envs\coasting\lib\site-packages\ema_workbench\em_
framework\optimization.py:48: ImportWarning: platypus based optimization n
ot available
  warnings.warn("platypus based optimization not available", ImportWarnin
g)
C:\Users\stein042\.conda\envs\coasting\lib\site-packages\ema_workbench\con
nectors\__init__.py:17: ImportWarning: vensim connector not available
  warnings.warn("vensim connector not available", ImportWarning)
C:\Users\stein042\.conda\envs\coasting\lib\site-packages\sklearn\externals
\six.py:31: DeprecationWarning: The module is deprecated in version 0.21 a
nd will be removed in version 0.23 since we've dropped support for Python
2.7. Please rely on the official version of six (https://pypi.org/project/
six/).
  "(https://pypi.org/project/six/).", DeprecationWarning)
```

In [2]:

```
ema_logging.log_to_stderr(ema_logging.INFO)

#We can define common uncertainties and outcomes for each model:
uncertainties = [RealParameter('density', 10, 100),
                 RealParameter('wind-speed', 0, 25),
                 RealParameter('probability-of-spread', 10, 100),
                 CategoricalParameter("big-jumps?", ("true", "false")) ,
                 CategoricalParameter("wind-direction",('"WEST"','"WEST-SOUTH-WEST"',
'"SOUTH-WEST"','"SOUTH-SOUTH-WEST"','"SOUTH"'))
                ]

outcomes = [TimeSeriesOutcome('ticks'),
            TimeSeriesOutcome('percent-burned'),
            TimeSeriesOutcome('lone-trees')
           ]

#Define the NetLogo model
wd = os.getcwd()
model = NetLogoModel('firemixedparams',
                         wd=wd,
                         model_file="fire-mixedparams.nlogo")

model.run_length = 200
model.replications = 5
model.uncertainties = uncertainties
model.outcomes = outcomes
```

In [3]:

```
start_time = datetime.datetime.now().isoformat()[:-10].replace(":","-")

nr_experiments = 900
#Using Latin Hypercube sampling
results = perform_experiments(model, nr_experiments, uncertainty_sampling=LHS)
```

```
[MainProcess/INFO] performing 900 scenarios * 1 policies * 1 model(s) = 90
0 experiments
[MainProcess/INFO] performing experiments sequentially
[MainProcess/INFO] 90 cases completed
[MainProcess/INFO] 180 cases completed
[MainProcess/INFO] 270 cases completed
[MainProcess/INFO] 360 cases completed
[MainProcess/INFO] 450 cases completed
[MainProcess/INFO] 540 cases completed
[MainProcess/INFO] 630 cases completed
[MainProcess/INFO] 720 cases completed
[MainProcess/INFO] 810 cases completed
[MainProcess/INFO] 900 cases completed
[MainProcess/INFO] experiments finished
```

In [4]:

```
#rename time output per EMA Wb expectations
experiments, outcomes = results
outcomes['TIME'] = outcomes.pop("ticks")
results = (experiments.copy(), outcomes.copy())
```

In [6]:

```
save_results(results, './' + start_time + '-experiments' + str(nr_experiments) + 'x' +
str(len(model.replications)) + '.tar.gz')
```

```
[MainProcess/INFO] results saved successfully to C:\Local\NetLogo-OutputCo
nversion-Local\2020-08-11T11-43-experiments900x5.tar.gz
```

In [7]:

```
results2 = load_results("2020-08-11T11-43-experiments900x5.tar.gz")
```

```
[MainProcess/INFO] results loaded succesfully from C:\Local\NetLogo-Output
Conversion-Local\2020-08-11T11-43-experiments900x5.tar.gz
```

Experiments

- Pandas dataframe, shape (nr_experiments) x (nr_inputs + 3)
- columns:
  - [input1, input2, ...] <- line 6 of CSV
  - scenario (experiment number) <- sequential numbering from 0
  - policy <- None
  - model_name <- line 1 of CSV
- data from rows of CSV, starting line 7

Outcomes

- dict of numpy arrays, shape:
  - (outcomes) <- line 6 of CSV, last elements
  - x (nr_experiments) <- from rows of CSV, highest unique value in first row divided by replications
  - x (nr_replications) <- from rows of CSV, based on repeated parameter combinations
  - x (timesteps + 1) <- variable, find longest run and fill others by repeating last value