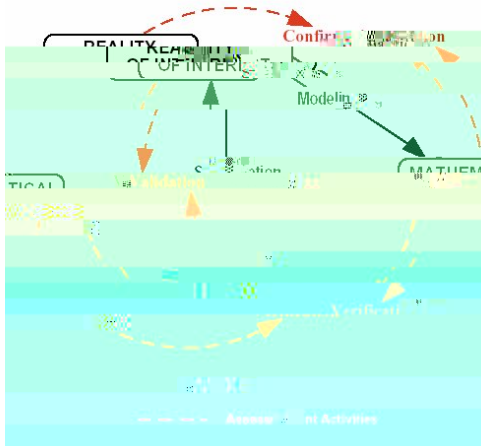
VVUQ

V -
V -



U Q -

. . , *et al.*, . . . / .

VECMA - V E C M A

. .

/

,

/

E VVUQ L

( )

/

- // . / /
- // . . /
- // . / /
- // . / /
- // . / - /
- // . . / /

/

W á á

-

-

( )

/

U ả Q ả ả

et al., ,

. ., // . / . / . .

/

U P :E

U P :S[illegible]

```
import easyvvuq as uq
input_json = "test_cannonsim.json"
output_json = "out_cannonsim.json"
number_of_samples = 15
my_campaign = uq.Campaign(state_filename=input_json)
# Set parameters to vary
my_campaign.vary_param("angle",
                       dist=uq.distributions.uniform(0.0,[illegible] rs
```

U P :Ca a

```
    "runs":
        "Run_0": {
            "angle": 0.7757645082270815,
            "height":
```

U P :A

```
output_filename = 'output.csv'
output_columns = ['Dist', 'lastvx', 'lastvy']
uq.elements.collate.aggregate_samples(my_campaign,
                                      output_filename=output_filename,
                                      output_columns=output_columns,
                                      header=0)
stats = uq.elements.analysis.BasicStats(my_campaign, value_cols=output_columns)
results, output_file = stats.apply()
my_campaign.save_state(output_json)
```

E

- BaseEncoder
- encode
  - params -
  - target_dir -

E :G

(cannonsim.template)

```
CANONSIM_INPUT_FILE:
gravity = #gravity
mass = #mass
velocity = #velocity
angle = #angle
height = #height
air_resistance = #air_resistance
time_step = #time_step
```

C S

// . / - /

V0.1

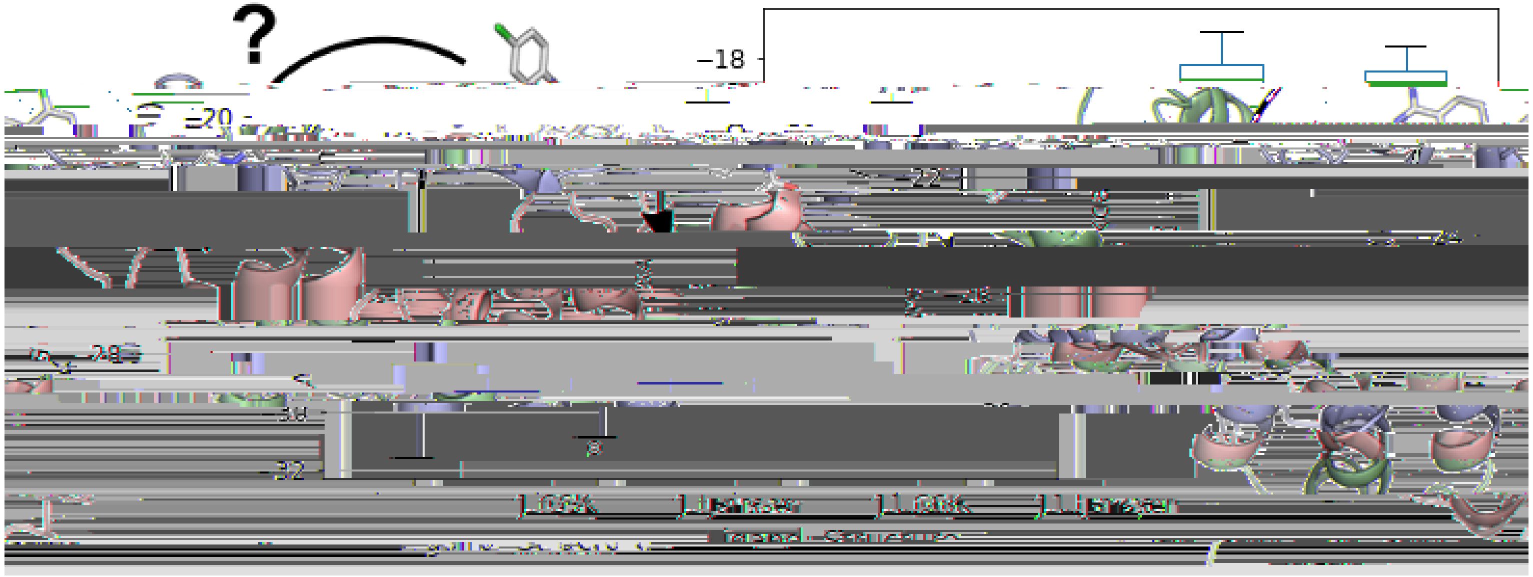
-

-

-

/

E 1: E



. . *et al.* - ,

. / .

E 2: A -

-

-

- -

-

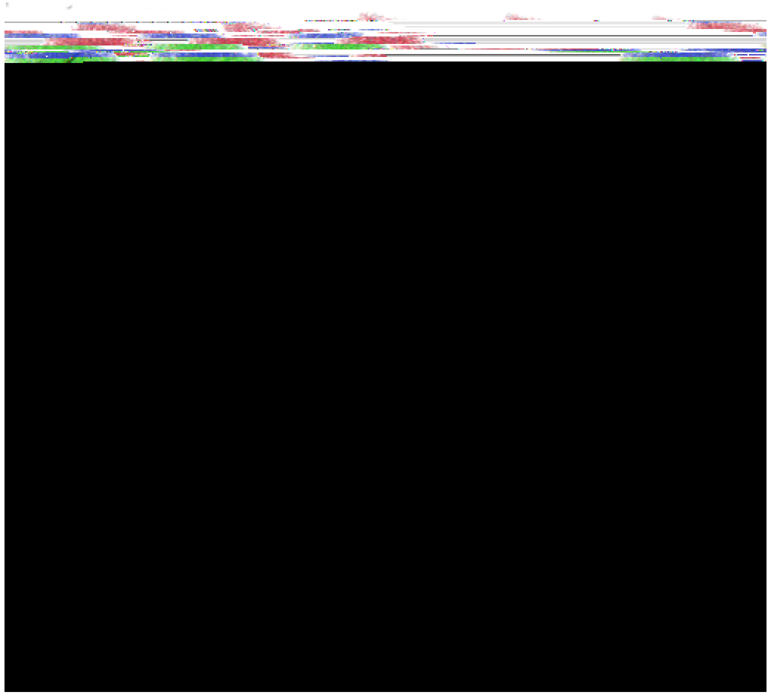
-

-

. , . . & . . ,

,

.



/

F P

-
-

-
- Campaigns
-
-

-
- Encoders/Decoders

/

A