
LOICA

Release 1.0.6

Gonzalo Vidal

Mar 23, 2022

CONTENTS

1	Introduction	1
2	Installation	3
3	LOICA allows you to:	5
4	Tutorials	7
5	API Reference	9
5.1	loica.....	9
6	Introduction	39
7	Installation	41
8	LOICA allows you to:	43
9	Tutorials	45
	Python Module Index	47
	Index	49

INTRODUCTION

Welcome to the LOICA (Logical Operators for Integrated Cell Algorithms) repository, our Python package for designing, modeling and characterizing genetic networks.

As you may have noticed, our logo features a beautiful bird—loica ([Leistes loyca](#)); a bird native to Chile known for its particular red chest and legendary kindness, with which we share name.

INSTALLATION

Installing LOICA is way easier than pronouncing it!

```
pip install loica
```

For more details please refer to our [Wiki](#) for installation instructions and developer guides.

LOICA ALLOWS YOU TO:

- Compile Code into DNA fragments that execute Cell Algorithms
- Easy programation of genetic network models
- Generation of synthetic data
- Communicate with Flapjack
- Use and output SBOL files
- Use all sorts of cellular computation
- Easy, fluid and customisable DNA design

TUTORIALS

Now that you have LOICA installed you can familiarize yourself with the tool using the [Jupyter notebook](#) tutorials designed for this purpose.

API REFERENCE

This page contains auto-generated API reference documentation¹.

5.1 loica

5.1.1 Subpackages

`loica.operators`

Submodules

`loica.operators.hill1`

Module Contents

Classes

<i>Hill1</i>	A class that represents a DNA fragment that encode a genetic operator.
--------------	--

class `Hill1`(*input*, *output*, *alpha*, *K*, *n*, *name=None*, *uri=None*, *sbol_comp=None*, *color='skyblue'*)

Bases: `loica.operators.operator.Operator`

A class that represents a DNA fragment that encode a genetic operator. The Hill1 Operator is an abstraction of a repressible or inducible promoter that maps an input into an output using a Hill function.

...

input [Regulator | Supplement] The input of the operator that regulates the expression of the output

output [Regulator | Reporter] The output of the operator that is regulated by the input

alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]

K [int | float] Half expression input concentration in Molar

n [int | float] Hill coefficient, cooperative degree (unitless)

uri [str, optional] SynBioHub URI

¹ Created with `sphinx-autoapi`

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

characterize(flappjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

__str__(self)
Return str(self).

characterize(self, flappjack, receiver, inverter, media, strain, signal, biomass_signal, gamma)

expression_rate(self, t, dt)

forward_model(self, a_j, b_j, n_i=2, K_i=1, a_A=100.0, b_A=0, K_A=1, n_A=2, Dt=0.05, sim_steps=10, A=0, odval=[1] * 100, gamma=0, p0_1=0, p0_2=0, nt=100)

residuals(self, df, oddf, a_A, b_A, K_A, n_A, gamma)

loica.operators.hill2

Module Contents

Classes

<i>Hill2</i>	A class that represents a DNA fragment that encode a genetic operator.
--------------	--

class Hill2(input, output, alpha, K, n, name=None, uri=None, sbol_comp=None, color='orange')

Bases: *loica.operators.operator.Operator*

A class that represents a DNA fragment that encode a genetic operator. The Hill2 Operator is an abstraction of a set of two repressible or inducible promoters that maps an 2 inputs into an output using a Hill function.

...

input [List [Regulator | Supplement]] The inputs of the operator that regulates the expression of the output

output [Regulator | Reporter | List] The output of the operator that is regulated by the input

alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]

K [int | float] Half expression input concentration in Molar

n [int | float] Hill coefficient, cooperative degree (unitless)

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

characterize(flappjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

```

__str__(self)
    Return str(self).

characterize(self, flapjack, receiver1, receiver2, chemical1, chemical2, nor_inverter, media, strain, signal,
             biomass_signal, gamma, lower_bounds=[0] * 8, upper_bounds=[100000000.0, 8,
             100000000.0, 8, 100000000.0, 100000000.0, 100000000.0, 100000000.0], init_x=[1, 2, 1, 2,
             1, 0, 0, 0])

expression_rate(self, t, dt)

forward_model(self, rep1_K=1, rep1_n=2, rep2_K=1, rep2_n=2, alpha0=1, alpha1=0, alpha2=0,
             alpha3=0, a_A=100.0, b_A=0, K_A=1, n_A=2, a_B=100.0, b_B=0, K_B=1, n_B=2,
             Dt=0.05, sim_steps=10, A=0, B=0, odval=[1] * 100, gamma=0, rep1_0=0, rep2_0=0,
             fp_0=0, nt=100)

residuals(self, df, oddf, a_A, b_A, K_A, n_A, a_B, b_B, K_B, n_B, chem1, chem2, gamma)

```

loica.operators.operator

Module Contents

Classes

<i>Operator</i>	A class that represents a DNA fragment that encode a genetic operator.
-----------------	--

class Operator(output, name=None, uri=None, sbol_comp=None, color='skyblue')

A class that represents a DNA fragment that encode a genetic operator.

...

output [Regulator | Reporter] The output of the operator that is regulated by the input

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

```

__str__(self)
    Return str(self).

```

loica.operators.receiver

Module Contents

Classes

<i>Receiver</i>	A class that represents a DNA fragment that encode a genetic operator.
-----------------	--

```
class Receiver(input, output, alpha, K, n, name=None, uri=None, sbol_comp=None, color='skyblue')
```

Bases: `Operator`

A class that represents a DNA fragment that encode a genetic operator. The Receiver Operator is an abstraction of an inducible promoter that maps an external input into an output using a Hill function.

...

input [Regulator | Supplement] The input of the operator that regulates the expression of the output

output [Regulator | Reporter] The output of the operator that is regulated by the input

alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]

K [int | float] Half expression input concentration in Molar

n [int | float] Hill coefficient, cooperative degree (unitless)

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

unit: str, optional Units of the characterization data

characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

__str__(self)

characterize(self, flapjack, vector, media, strain, signal, biomass_signal)

expression_rate(self, t, dt)

forward_model(self, a=0, b=1, K_A=1, n_A=2, Dt=0.05, sim_steps=10, A=[0], odval=[1] * 100, gamma=0, p0=0, nt=100)

residuals(self, df, oddf)

`loica.operators.source`

Module Contents

Classes

Source

A class that represents a DNA fragment that encode a genetic operator.

```
class Source(output, rate, uri=None, sbol_comp=None, color='blue', name=None)
```

Bases: `Operator`

A class that represents a DNA fragment that encode a genetic operator. The Source Operator is an abstraction of a constitutive promoter that produces output.

...

output [Regulator | Reporter] The output of the operator that is constitutively expressed

rate [float] Output constitutive expression rate in MEFL/second

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

characterize(flappjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

__str__(self)

characterize(self, flappjack, vector, media, strain, signal, biomass_signal)

expression_rate(self, t, dt)

forward_model(self, Dt=0.25, sim_steps=10, odval=[1] * 97, rate=1, gamma=0, p0=0, nt=100)

residuals(self, df, oddf)

Package Contents

Classes

<i>Hill1</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Hill2</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Operator</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Operator</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Operator</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Operator</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Operator</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Receiver</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Receiver</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Receiver</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Source</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Source</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Source</i>	A class that represents a DNA fragment that encode a genetic operator.

continues on next page

Table 6 – continued from previous page

Source	A class that represents a DNA fragment that encode a genetic operator.
<pre>class Hill1(input, output, alpha, K, n, name=None, uri=None, sbol_comp=None, color='skyblue') Bases: <i>loica.operators.operator.Operator</i></pre>	
<p>A class that represents a DNA fragment that encode a genetic operator. The Hill1 Operator is an abstraction of a repressible or inducible promoter that maps an input into an output using a Hill function.</p> <p>...</p>	
<p>input [Regulator Supplement] The input of the operator that regulates the expression of the output</p> <p>output [Regulator Reporter] The output of the operator that is regulated by the input</p> <p>alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]</p> <p>K [int float] Half expression input concentration in Molar</p> <p>n [int float] Hill coefficient, cooperative degree (unitless)</p> <p>uri [str, optional] SynBioHub URI</p> <p>sbol_comp [SBOL Component, optional] SBOL Component</p> <p>name [str, optional] Name of the operator displayed on the network representation</p> <p>color: str, optional Color displayed on the network representation</p>	
<p>characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate</p>	
<p>__str__(self)</p> <p>Return str(self).</p>	
<p>characterize(self, flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma)</p>	
<p>expression_rate(self, t, dt)</p>	
<p>forward_model(self, a_j, b_j, n_i=2, K_i=1, a_A=100.0, b_A=0, K_A=1, n_A=2, Dt=0.05, sim_steps=10, A=0, odval=[1] * 100, gamma=0, p0_1=0, p0_2=0, nt=100)</p>	
<p>residuals(self, df, oddf, a_A, b_A, K_A, n_A, gamma)</p>	
<pre>class Hill2(input, output, alpha, K, n, name=None, uri=None, sbol_comp=None, color='orange') Bases: <i>loica.operators.operator.Operator</i></pre>	
<p>A class that represents a DNA fragment that encode a genetic operator. The Hill2 Operator is an abstraction of a set of two repressible or inducible promoters that maps an 2 inputs into an output using a Hill function.</p> <p>...</p>	
<p>input [List [Regulator Supplement]] The inputs of the operator that regulates the expression of the output</p> <p>output [Regulator Reporter List] The output of the operator that is regulated by the input</p> <p>alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]</p> <p>K [int float] Half expression input concentration in Molar</p> <p>n [int float] Hill coefficient, cooperative degree (unitless)</p> <p>uri [str, optional] SynBioHub URI</p> <p>sbol_comp [SBOL Component, optional] SBOL Component</p>	

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

characterize(flappjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

__str__(self)
Return str(self).

characterize(self, flappjack, receiver1, receiver2, chemical1, chemical2, nor_inverter, media, strain, signal, biomass_signal, gamma, lower_bounds=[0] * 8, upper_bounds=[100000000.0, 8, 100000000.0, 8, 100000000.0, 100000000.0, 100000000.0, 100000000.0], init_x=[1, 2, 1, 2, 1, 0, 0, 0])

expression_rate(self, t, dt)

forward_model(self, rep1_K=1, rep1_n=2, rep2_K=1, rep2_n=2, alpha0=1, alpha1=0, alpha2=0, alpha3=0, a_A=100.0, b_A=0, K_A=1, n_A=2, a_B=100.0, b_B=0, K_B=1, n_B=2, Dt=0.05, sim_steps=10, A=0, B=0, odval=[1] * 100, gamma=0, rep1_0=0, rep2_0=0, fp_0=0, nt=100)

residuals(self, df, oddf, a_A, b_A, K_A, n_A, a_B, b_B, K_B, n_B, chem1, chem2, gamma)

class Operator(output, name=None, uri=None, sbol_comp=None, color='skyblue')

A class that represents a DNA fragment that encode a genetic operator.

...

output [Regulator | Reporter] The output of the operator that is regulated by the input

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

__str__(self)
Return str(self).

class Operator(output, name=None, uri=None, sbol_comp=None, color='skyblue')

A class that represents a DNA fragment that encode a genetic operator.

...

output [Regulator | Reporter] The output of the operator that is regulated by the input

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

__str__(self)
Return str(self).

class Operator(output, name=None, uri=None, sbol_comp=None, color='skyblue')

A class that represents a DNA fragment that encode a genetic operator.

...

output [Regulator | Reporter] The output of the operator that is regulated by the input

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

__str__(self)
Return str(self).

class Operator(output, name=None, uri=None, sbol_comp=None, color='skyblue')
A class that represents a DNA fragment that encode a genetic operator.

...

output [Regulator | Reporter] The output of the operator that is regulated by the input

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

__str__(self)
Return str(self).

class Operator(output, name=None, uri=None, sbol_comp=None, color='skyblue')
A class that represents a DNA fragment that encode a genetic operator.

...

output [Regulator | Reporter] The output of the operator that is regulated by the input

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

__str__(self)
Return str(self).

class Receiver(input, output, alpha, K, n, name=None, uri=None, sbol_comp=None, color='skyblue')
Bases: [Operator](#)

A class that represents a DNA fragment that encode a genetic operator. The Receiver Operator is an abstraction of an inducible promoter that maps an external input into an output using a Hill function.

...

input [Regulator | Supplement] The input of the operator that regulates the expression of the output

output [Regulator | Reporter] The output of the operator that is regulated by the input

alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]

K [int | float] Half expression input concentration in Molar

n [int | float] Hill coefficient, cooperative degree (unitless)

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

unit: str, optional Units of the characterization data

characterize(flappjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

__str__(self)

characterize(self, flappjack, vector, media, strain, signal, biomass_signal)

expression_rate(self, t, dt)

forward_model(self, a=0, b=1, K_A=1, n_A=2, Dt=0.05, sim_steps=10, A=[0], odval=[1] * 100, gamma=0, p0=0, nt=100)

residuals(self, df, oddf)

class Receiver(input, output, alpha, K, n, name=None, uri=None, sbol_comp=None, color='skyblue')

Bases: [Operator](#)

A class that represents a DNA fragment that encode a genetic operator. The Receiver Operator is an abstraction of an inducible promoter that maps an external input into an output using a Hill function.

...

input [Regulator | Supplement] The input of the operator that regulates the expression of the output

output [Regulator | Reporter] The output of the operator that is regulated by the input

alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]

K [int | float] Half expression input concentration in Molar

n [int | float] Hill coefficient, cooperative degree (unitless)

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

unit: str, optional Units of the characterization data

characterize(flappjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

__str__(self)

characterize(self, flappjack, vector, media, strain, signal, biomass_signal)

expression_rate(self, t, dt)

forward_model(self, a=0, b=1, K_A=1, n_A=2, Dt=0.05, sim_steps=10, A=[0], odval=[1] * 100, gamma=0, p0=0, nt=100)

residuals(self, df, oddf)

class Receiver(input, output, alpha, K, n, name=None, uri=None, sbol_comp=None, color='skyblue')

Bases: [Operator](#)

A class that represents a DNA fragment that encode a genetic operator. The Receiver Operator is an abstraction of an inducible promoter that maps an external input into an output using a Hill function.

...

input [Regulator | Supplement] The input of the operator that regulates the expression of the output

output [Regulator | Reporter] The output of the operator that is regulated by the input

alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]

K [int | float] Half expression input concentration in Molar

n [int | float] Hill coefficient, cooperative degree (unitless)

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

unit: str, optional Units of the characterization data

characterize(flappjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

__str__(*self*)

characterize(*self*, *flappjack*, *vector*, *media*, *strain*, *signal*, *biomass_signal*)

expression_rate(*self*, *t*, *dt*)

forward_model(*self*, *a*=0, *b*=1, *K_A*=1, *n_A*=2, *Dt*=0.05, *sim_steps*=10, *A*=[0], *odval*=[1] * 100, *gamma*=0, *p0*=0, *nt*=100)

residuals(*self*, *df*, *oddf*)

class Source(*output*, *rate*, *uri*=None, *sbol_comp*=None, *color*='blue', *name*=None)

Bases: *Operator*

A class that represents a DNA fragment that encode a genetic operator. The Source Operator is an abstraction of a constitutive promoter that produces output.

...

output [Regulator | Reporter] The output of the operator that is constitutively expressed

rate [float] Output constitutive expression rate in MEFL/second

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

characterize(flappjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

__str__(*self*)

characterize(*self*, *flappjack*, *vector*, *media*, *strain*, *signal*, *biomass_signal*)

expression_rate(*self*, *t*, *dt*)

```

forward_model(self, Dt=0.25, sim_steps=10, odval=[1] * 97, rate=1, gamma=0, p0=0, nt=100)
residuals(self, df, oddf)
class Source(output, rate, uri=None, sbol_comp=None, color='blue', name=None)
    Bases: Operator

    A class that represents a DNA fragment that encode a genetic operator. The Source Operator is an abstraction of
    a constitutive promoter that produces output.

    ...

    output [Regulator | Reporter] The output of the operator that is constitutively expressed
    rate [float] Output constitutive expression rate in MEFL/second
    uri [str, optional] SynBioHub URI
    sbol_comp [SBOL Component, optional] SBOL Component
    name [str, optional] Name of the operator displayed on the network representation
    color: str, optional Color displayed on the network representation

    characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the
        Operator model that maps Input concentration into Output expression rate

    __str__(self)

    characterize(self, flapjack, vector, media, strain, signal, biomass_signal)

    expression_rate(self, t, dt)

    forward_model(self, Dt=0.25, sim_steps=10, odval=[1] * 97, rate=1, gamma=0, p0=0, nt=100)
    residuals(self, df, oddf)
class Source(output, rate, uri=None, sbol_comp=None, color='blue', name=None)
    Bases: Operator

    A class that represents a DNA fragment that encode a genetic operator. The Source Operator is an abstraction of
    a constitutive promoter that produces output.

    ...

    output [Regulator | Reporter] The output of the operator that is constitutively expressed
    rate [float] Output constitutive expression rate in MEFL/second
    uri [str, optional] SynBioHub URI
    sbol_comp [SBOL Component, optional] SBOL Component
    name [str, optional] Name of the operator displayed on the network representation
    color: str, optional Color displayed on the network representation

    characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the
        Operator model that maps Input concentration into Output expression rate

    __str__(self)

    characterize(self, flapjack, vector, media, strain, signal, biomass_signal)

    expression_rate(self, t, dt)

    forward_model(self, Dt=0.25, sim_steps=10, odval=[1] * 97, rate=1, gamma=0, p0=0, nt=100)

```

residuals(*self*, *df*, *oddf*)

class Source(*output*, *rate*, *uri=None*, *sbol_comp=None*, *color='blue'*, *name=None*)

Bases: [Operator](#)

A class that represents a DNA fragment that encode a genetic operator. The Source Operator is an abstraction of a constitutive promoter that produces output.

...

output [Regulator | Reporter] The output of the operator that is constitutively expressed

rate [float] Output constitutive expression rate in MEFL/second

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

characterize(*flapjack*, *receiver*, *inverter*, *media*, *strain*, *signal*, *biomass_signal*, *gamma*) Parameterize the Operator model that maps Input concentration into Output expression rate

__str__(*self*)

characterize(*self*, *flapjack*, *vector*, *media*, *strain*, *signal*, *biomass_signal*)

expression_rate(*self*, *t*, *dt*)

forward_model(*self*, *Dt=0.25*, *sim_steps=10*, *odval=[1] * 97*, *rate=1*, *gamma=0*, *p0=0*, *nt=100*)

residuals(*self*, *df*, *oddf*)

5.1.2 Submodules

[loica.assay](#)

Module Contents

Classes

Assay	Assay measures a set of samples in parallel at a set of timepoints.
-----------------------	---

class Assay(*samples*, *n_measurements*, *interval*, *name='Loica assay'*, *description=""*, *biomass_signal_id=None*)

Assay measures a set of samples in parallel at a set of timepoints. Connects to flapjack to generate data, and to fit parameters to data.

...

samples [List[Sample]] List of Samples that belongs to the Assay

n_measurements [int] Number of measurements to take

interval [int] Time in hours between each measurements

name [str] Name of the Assay

description: str Descriptioin of the Assay

biomass_signal_id [int] Flapjack ID of the Assay that is associated with the Assay

run(substeps=10, nsr=0, biomass_bg=0, fluo_bg=0) Runs the Assay time series

upload(flapjack, study) Upload the data produced by running the Assay to Flapjack into the Study

Assay measures a set of samples in parallel at a set of timepoints Connects to flapjack to generate data, and to fit parameters to data

run(self, substeps=10, nsr=0, biomass_bg=0, fluo_bg=0, stochastic=False)

Run the assay measuring at specified time points, with simulation time step dt

upload(self, flapjack, study)

loica.colony

Module Contents

Classes

Colony

class Colony(circuit=None, r0=1, mu0=1)

fun(self, x)

kymograph(self, nx, t0, tmax)

map_kymo(self, kymo)

norm_kymo(self, kymo)

loica.geneproduct

Module Contents

Classes

<i>GeneProduct</i>	A class that represents a gene product, protein or RNA.
<i>Regulator</i>	Representation of a regulatory gene product.
<i>Reporter</i>	Representation of a regulatory gene product.

class GeneProduct(name, init_concentration=0, degradation_rate=0, uri=None, sbol_comp=None, type_='PRO', color='silver')

A class that represents a gene product, protein or RNA.

...

name [str] Name of the gene product

init_concentration [int | float] Initial concentration of the gene product in Molar

degradation_rate [int | float] Degradation rate of the gene product

type_ [str, optional] Molecular type of the gene product, could be 'PRO' or 'RNA'

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

shape = ^

__str__(self)
Return str(self).

express(self, rate)

initialize(self)

step(self, growth_rate, dt)

class Regulator(name, init_concentration=0, degradation_rate=0, sbol_comp=None, color='lightgreen')
Bases: [GeneProduct](#)

Representation of a regulatory gene product. Child of GeneProduct.

class Reporter(name, init_concentration=0, degradation_rate=0, signal_id=None, color='w',
sbol_comp=None)
Bases: [GeneProduct](#)

Representation of a regulatory gene product.

signal_id [str, optional] Flapjack ID of the signal that the reporter is associated with.

color [str, optional] Color of the reporter

loica.genetic_network

Module Contents

Classes

[GeneticNetwork](#)

Representation of a genetic network composed by a set of Operators, Regulators and Reporters.

class GeneticNetwork(vector=None)

Representation of a genetic network composed by a set of Operators, Regulators and Reporters.

...

operators [List[Operator]] List of Operators that are part of the genetic network

regulators [List[Regulator]] List of Regulators that are part of the genetic network

reporters [List[Reporter]] List of Reporters that are part of the genetic network

vector [int] Flapjack ID of the vector that is associated with the genetic network

to_graph() Builds a graph representation of the genetic network

draw() Generates a plot of the graph representation built by to_graph()

to_sbol(sbol_doc=None) Generates a SBOL3 Document representation of the genetic network on sbol_doc

add_operator(self, ops)

```

add_regulator(self, regs)
add_reporter(self, reps)
draw(self, node_shape='o', node_size=500, linewidths=0, alpha=0.5, arrowsize=10, font_size=6,
      font_family='Tahoma', font_weight='bold', pos=nx.kamada_kawai_layout, contracted=False)
initialize(self)
step(self, growth_rate=1, t=0, dt=0.1)
step_stochastic(self, growth_rate=1, t=0, dt=0.1)
substep_stochastic(self, t=0, dt=0.1, growth_rate=1)
to_contracted_graph(self)
to_graph(self)
to_sbol(self, sbol_doc: sbol3.Document = None) → sbol3.Document
    Convert the genetic network to SBOL. :param sbol_doc: The SBOL document to add the genetic network
    to.

```

loica.metabolism

Module Contents

Classes

<i>DataMetabolism</i>	Characterized context for gene expression, incorporates biomass and growth rate.
<i>Metabolism</i>	Context for gene expression, incorporates biomass and growth rate.
<i>SimulatedMetabolism</i>	Simulated context for gene expression, incorporates biomass and growth rate.

Functions

<i>gompertz</i> (t, y0, ymax, um, l)
<i>gompertz_growth_rate</i> (t, y0, ymax, um, l)
<i>ramp_biomass</i> (t, od0, start, slope)
<i>ramp_growth_rate</i> (t, start, slope)
<i>step_biomass</i> (t, od0, start)
<i>step_growth_rate</i> (t, start)

```

class DataMetabolism(name, fj, media, strain, vector, biomass_signal)
    Bases: Metabolism

```

Characterized context for gene expression, incorporates biomass and growth rate. ...

name [str, optional] Name of the metabolism or corresponding strain

fj [Flapjack] Flapjack instance used to fetch data from

media [str] Name of the media to query

strain [str] Name of the strain to query

vector [str] Name of the vector to query

biomass_signal [str] Name of signal to query and use as biomass

biomass(t) Return biomass at a given time from characterization data

growth_rate(t) Return growth rate at a given time from characterization data

biomass(*self*, *t*)

growth_rate(*self*, *t*)

class Metabolism(*name=None*)

Context for gene expression, incorporates biomass and growth rate. ...

name [str, optional] Name of the metabolism or corresponding strain

class SimulatedMetabolism(*name, biomass, growth_rate*)

Bases: [Metabolism](#)

Simulated context for gene expression, incorporates biomass and growth rate. ...

name [str, optional] Name of the metabolism or corresponding strain

biomass A function of time that describes biomass $f(t)=\text{biomass}$

growth_rate A function of time that describes the growth rate $f(t)=\text{growth rate}$

gompertz(*t, y0, ymax, um, l*)

gompertz_growth_rate(*t, y0, ymax, um, l*)

ramp_biomass(*t, od0, start, slope*)

ramp_growth_rate(*t, start, slope*)

step_biomass(*t, od0, start*)

step_growth_rate(*t, start*)

[loica.sample](#)

Module Contents

Classes

[Sample](#)

Representation of a sample that encapsulates Genetic-Network and Metabolism.

class Sample(*genetic_network=None, metabolism=None, assay=None, media=None, strain=None*)

Representation of a sample that encapsulates GeneticNetwork and Metabolism. Incorporate environment information such as Supplements or chemicals, strain and media. Ex: 1 well in a plate, single cell. ...

genetic_network [GeneticNetwork] genetic network that is part of the sample

metabolism [Metabolism] metabolism that drives the genetic network in the sample

assay [Assay] assay to which this sample belongs

media [str] Name of the media in the sample

strain [str]

Name of the strain in the sample

Methods

add_supplement(supplement, concentration) establishes the concentration of Supplement

initialize(self)

set_regulator(self, name, concentration)

set_reporter(self, name, concentration)

set_supplement(self, supplement, concentration)

step(self, t, dt, stochastic=False)

`loica.supplement`

Module Contents

Classes

<i>Supplement</i>	Representation of a chemical
-------------------	------------------------------

class Supplement(name, pubchemid=None, supplier_id=None, sbol_comp=None, color='pink')

Representation of a chemical

...

name [str] Name of the supplement

concentration [int | float] concentration of the supplement in Molar

pubchemid [str] PubChemID URI of the supplement

supplier_id [str] Supplier ID of the supplement. An URL of the product that you acquire. Accepts list of the form [product URL, catalog number, batch].

sbol_comp [str] SBOL component of the supplement.

__str__(self)

Return str(self).

loica.util

Module Contents

Functions

characterize_growth(flapjack, vector, media, strain, biomass_signal, n_gaussians, epsilon)

forward_model_growth(Dt=0.05, sim_steps=10, muval=[0] * 100, od0=0, nt=100)

load_loica(filename)

residuals_growth(data, epsilon, dt, t, n_gaussians)

save_loica(obj, filename)

characterize_growth(flapjack, vector, media, strain, biomass_signal, n_gaussians, epsilon)

forward_model_growth(Dt=0.05, sim_steps=10, muval=[0] * 100, od0=0, nt=100)

load_loica(filename)

residuals_growth(data, epsilon, dt, t, n_gaussians)

save_loica(obj, filename)

5.1.3 Package Contents

Classes

<i>Assay</i>	Assay measures a set of samples in parallel at a set of timepoints.
<i>Colony</i>	
<i>DataMetabolism</i>	Characterized context for gene expression, incorporates biomass and growth rate.
<i>GeneProduct</i>	A class that represents a gene product, protein or RNA.
<i>GeneticNetwork</i>	Representation of a genetic network composed by a set of Operators, Regulators and Reporters.
<i>Hill1</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Hill1</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Hill2</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Hill2</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Metabolism</i>	Context for gene expression, incorporates biomass and growth rate.

continues on next page

Table 16 – continued from previous page

<i>Operator</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Operator</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Receiver</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Receiver</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Regulator</i>	Representation of a regulatory gene product.
<i>Regulator</i>	Representation of a regulatory gene product.
<i>Reporter</i>	Representation of a regulatory gene product.
<i>Reporter</i>	Representation of a regulatory gene product.
<i>Sample</i>	Representation of a sample that encapsulates Genetic-Network and Metabolism.
<i>SimulatedMetabolism</i>	Simulated context for gene expression, incorporates biomass and growth rate.
<i>Source</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Source</i>	A class that represents a DNA fragment that encode a genetic operator.
<i>Supplement</i>	Representation of a chemical
<i>Supplement</i>	Representation of a chemical

Functions

<i>characterize_growth</i> (flapjack, vector, media, strain, biomass_signal, n_gaussians, epsilon)
<i>forward_model_growth</i> (Dt=0.05, sim_steps=10, mu_val=[0] * 100, od0=0, nt=100)
<i>gompertz</i> (t, y0, ymax, um, l)
<i>gompertz_growth_rate</i> (t, y0, ymax, um, l)
<i>load_loica</i> (filename)
<i>ramp_biomass</i> (t, od0, start, slope)
<i>ramp_growth_rate</i> (t, start, slope)
<i>residuals_growth</i> (data, epsilon, dt, t, n_gaussians)
<i>save_loica</i> (obj, filename)
<i>step_biomass</i> (t, od0, start)
<i>step_growth_rate</i> (t, start)

class Assay(samples, n_measurements, interval, name='Loica assay', description="", biomass_signal_id=None)
 Assay measures a set of samples in parallel at a set of timepoints. Connects to flapjack to generate data, and to

fit parameters to data.

...

samples [List[Sample]] List of Samples that belongs to the Assay

n_measurements [int] Number of measurements to take

interval [int] Time in hours between each measurements

name [str] Name of the Assay

description: str Description of the Assay

biomass_signal_id [int] Flapjack ID of the Assay that is associated with the Assay

run(substeps=10, nsr=0, biomass_bg=0, fluo_bg=0) Runs the Assay time series

upload(flapjack, study) Upload the data produced by running the Assay to Flapjack into the Study

Assay measures a set of samples in parallel at a set of timepoints Connects to flapjack to generate data, and to fit parameters to data

run(self, substeps=10, nsr=0, biomass_bg=0, fluo_bg=0, stochastic=False)

Run the assay measuring at specified time points, with simulation time step dt

upload(self, flapjack, study)

class Colony(circuit=None, r0=1, mu0=1)

fun(self, x)

kymograph(self, nx, t0, tmax)

map_kymo(self, kymo)

norm_kymo(self, kymo)

class DataMetabolism(name, fj, media, strain, vector, biomass_signal)

Bases: [Metabolism](#)

Characterized context for gene expression, incorporates biomass and growth rate. ...

name [str, optional] Name of the metabolism or correponding strain

fj [Flapjack] Flapjack instance used to fetch data from

media [str] Name of the media to query

strain [str] Name of the strain to query

vector [str] Name of the vector to query

biomass_signal [str] Name of signal to query and use as biomass

biomass(t) Return biomass at a given time from characterization data

growth:rate(t) Return growth rate at a given time from characterization data

biomass(self, t)

growth_rate(self, t)

class GeneProduct(name, init_concentration=0, degradation_rate=0, uri=None, sbol_comp=None, type_='PRO', color='silver')

A class that represents a gene product, protein or RNA.

...


```

name [str] Name of the gene product
init_concentration [int | float] Initial concentration of the gene product in Molar
degradation_rate [int | float] Degradation rate of the gene product
type_ [str, optional] Molecular type of the gene product, could be 'PRO' or 'RNA'
uri [str, optional] SynBioHub URI
sbol_comp [SBOL Component, optional] SBOL Component
shape = ^
__str__(self)
    Return str(self).
express(self, rate)
initialize(self)
step(self, growth_rate, dt)
class GeneticNetwork(vector=None)
    Representation of a genetic network composed by a set of Operators, Regulators and Reporters.
    ...
operators [List[Operator]] List of Operators that are part of the genetic network
regulators [List[Regulator]] List of Regulators that are part of the genetic network
reporters [List[Reporter]] List of Reporters that are part of the genetic network
vector [int] Flapjack ID of the vector that is associated with the genetic network

to_graph() Builds a graph representation of the genetic network
draw() Generates a plot of the graph representation builded by to_graph()
to_sbol(sbol_doc=None) Generates a SBOL3 Document representation of the genetic network on sbol_doc

add_operator(self, ops)
add_regulator(self, regs)
add_reporter(self, reps)
draw(self, node_shape='o', node_size=500, linewidths=0, alpha=0.5, arrowsize=10, font_size=6,
    font_family='Tahoma', font_weight='bold', pos=nx.kamada_kawai_layout, contracted=False)
initialize(self)
step(self, growth_rate=1, t=0, dt=0.1)
step_stochastic(self, growth_rate=1, t=0, dt=0.1)
substep_stochastic(self, t=0, dt=0.1, growth_rate=1)
to_contracted_graph(self)
to_graph(self)
to_sbol(self, sbol_doc: sbol3.Document = None) → sbol3.Document
    Convert the genetic network to SBOL. :param sbol_doc: The SBOL document to add the genetic network
    to.

```

```
class Hill1(input, output, alpha, K, n, name=None, uri=None, sbol_comp=None, color='skyblue')
```

Bases: [loica.operators.operator.Operator](#)

A class that represents a DNA fragment that encode a genetic operator. The Hill1 Operator is an abstraction of a repressible or inducible promoter that maps an input into an output using a Hill function.

...

input [Regulator | Supplement] The input of the operator that regulates the expression of the output

output [Regulator | Reporter] The output of the operator that is regulated by the input

alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]

K [int | float] Half expression input concentration in Molar

n [int | float] Hill coefficient, cooperative degree (unitless)

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

characterize(flappjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

__str__(self)

Return str(self).

characterize(self, flappjack, receiver, inverter, media, strain, signal, biomass_signal, gamma)

expression_rate(self, t, dt)

forward_model(self, a_j, b_j, n_i=2, K_i=1, a_A=100.0, b_A=0, K_A=1, n_A=2, Dt=0.05, sim_steps=10, A=0, odval=[1] * 100, gamma=0, p0_1=0, p0_2=0, nt=100)

residuals(self, df, oddf, a_A, b_A, K_A, n_A, gamma)

```
class Hill1(input, output, alpha, K, n, name=None, uri=None, sbol_comp=None, color='skyblue')
```

Bases: [loica.operators.operator.Operator](#)

A class that represents a DNA fragment that encode a genetic operator. The Hill1 Operator is an abstraction of a repressible or inducible promoter that maps an input into an output using a Hill function.

...

input [Regulator | Supplement] The input of the operator that regulates the expression of the output

output [Regulator | Reporter] The output of the operator that is regulated by the input

alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]

K [int | float] Half expression input concentration in Molar

n [int | float] Hill coefficient, cooperative degree (unitless)

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

characterize(flappjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

__str__(self)
Return str(self).

characterize(self, flappjack, receiver, inverter, media, strain, signal, biomass_signal, gamma)

expression_rate(self, t, dt)

forward_model(self, a_j, b_j, n_i=2, K_i=1, a_A=100.0, b_A=0, K_A=1, n_A=2, Dt=0.05, sim_steps=10, A=0, odval=[1] * 100, gamma=0, p0_1=0, p0_2=0, nt=100)

residuals(self, df, oddf, a_A, b_A, K_A, n_A, gamma)

class Hill2(input, output, alpha, K, n, name=None, uri=None, sbol_comp=None, color='orange')

Bases: [loica.operators.operator.Operator](#)

A class that represents a DNA fragment that encode a genetic operator. The Hill2 Operator is an abstraction of a set of two repressible or inducible promoters that maps an 2 inputs into an output using a Hill function.

...

input [List [Regulator | Supplement]] The inputs of the operator that regulates the expression of the output

output [Regulator | Reporter | List] The output of the operator that is regulated by the input

alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]

K [int | float] Half expression input concentration in Molar

n [int | float] Hill coefficient, cooperative degree (unitless)

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

characterize(flappjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

__str__(self)
Return str(self).

characterize(self, flappjack, receiver1, receiver2, chemical1, chemical2, nor_inverter, media, strain, signal, biomass_signal, gamma, lower_bounds=[0] * 8, upper_bounds=[100000000.0, 8, 100000000.0, 8, 100000000.0, 100000000.0, 100000000.0, 100000000.0], init_x=[1, 2, 1, 2, 1, 0, 0, 0])

expression_rate(self, t, dt)

forward_model(self, rep1_K=1, rep1_n=2, rep2_K=1, rep2_n=2, alpha0=1, alpha1=0, alpha2=0, alpha3=0, a_A=100.0, b_A=0, K_A=1, n_A=2, a_B=100.0, b_B=0, K_B=1, n_B=2, Dt=0.05, sim_steps=10, A=0, B=0, odval=[1] * 100, gamma=0, rep1_0=0, rep2_0=0, fp_0=0, nt=100)

residuals(self, df, oddf, a_A, b_A, K_A, n_A, a_B, b_B, K_B, n_B, chem1, chem2, gamma)

class Hill2(input, output, alpha, K, n, name=None, uri=None, sbol_comp=None, color='orange')

Bases: [loica.operators.operator.Operator](#)

A class that represents a DNA fragment that encode a genetic operator. The Hill2 Operator is an abstraction of a set of two repressible or inducible promoters that maps an 2 inputs into an output using a Hill function.

...

input [List [Regulator | Supplement]] The inputs of the operator that regulates the expression of the output

output [Regulator | Reporter | List] The output of the operator that is regulated by the input

alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]

K [int | float] Half expression input concentration in Molar

n [int | float] Hill coefficient, cooperative degree (unitless)

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

characterize(flappjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

__str__(self)
Return str(self).

characterize(self, flappjack, receiver1, receiver2, chemical1, chemical2, nor_inverter, media, strain, signal, biomass_signal, gamma, lower_bounds=[0] * 8, upper_bounds=[100000000.0, 8, 100000000.0, 8, 100000000.0, 100000000.0, 100000000.0, 100000000.0], init_x=[1, 2, 1, 2, 1, 0, 0, 0])

expression_rate(self, t, dt)

forward_model(self, rep1_K=1, rep1_n=2, rep2_K=1, rep2_n=2, alpha0=1, alpha1=0, alpha2=0, alpha3=0, a_A=100.0, b_A=0, K_A=1, n_A=2, a_B=100.0, b_B=0, K_B=1, n_B=2, Dt=0.05, sim_steps=10, A=0, B=0, odval=[1] * 100, gamma=0, rep1_0=0, rep2_0=0, fp_0=0, nt=100)

residuals(self, df, oddf, a_A, b_A, K_A, n_A, a_B, b_B, K_B, n_B, chem1, chem2, gamma)

class Metabolism(name=None)

Context for gene expression, incorporates biomass and growth rate. ...

name [str, optional] Name of the metabolism or corresponding strain

class Operator(output, name=None, uri=None, sbol_comp=None, color='skyblue')

A class that represents a DNA fragment that encode a genetic operator.

...

output [Regulator | Reporter] The output of the operator that is regulated by the input

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

__str__(self)
Return str(self).

```

class Operator(output, name=None, uri=None, sbol_comp=None, color='skyblue')
    A class that represents a DNA fragment that encode a genetic operator.

    ...

    output [Regulator | Reporter] The output of the operator that is regulated by the input
    uri [str, optional] SynBioHub URI
    sbol_comp [SBOL Component, optional] SBOL Component
    name [str, optional] Name of the operator displayed on the network representation
    color: str, optional Color displayed on the network representation
    __str__(self)
        Return str(self).

class Receiver(input, output, alpha, K, n, name=None, uri=None, sbol_comp=None, color='skyblue')
    Bases: Operator

    A class that represents a DNA fragment that encode a genetic operator. The Receiver Operator is an abstraction
    of an inducible promoter that maps an external input into an output using a Hill function.

    ...

    input [Regulator | Supplement] The input of the operator that regulates the expression of the output
    output [Regulator | Reporter] The output of the operator that is regulated by the input
    alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]
    K [int | float] Half expression input concentration in Molar
    n [int | float] Hill coefficient, cooperative degree (unitless)
    uri [str, optional] SynBioHub URI
    sbol_comp [SBOL Component, optional] SBOL Component
    name [str, optional] Name of the operator displayed on the network representation
    color: str, optional Color displayed on the network representation
    unit: str, optional Units of the characterization data

    characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the
        Operator model that maps Input concentration into Output expression rate

    __str__(self)

    characterize(self, flapjack, vector, media, strain, signal, biomass_signal)

    expression_rate(self, t, dt)

    forward_model(self, a=0, b=1, K_A=1, n_A=2, Dt=0.05, sim_steps=10, A=[0], odval=[1] * 100,
        gamma=0, p0=0, nt=100)

    residuals(self, df, oddf)

class Receiver(input, output, alpha, K, n, name=None, uri=None, sbol_comp=None, color='skyblue')
    Bases: Operator

    A class that represents a DNA fragment that encode a genetic operator. The Receiver Operator is an abstraction
    of an inducible promoter that maps an external input into an output using a Hill function.

    ...

```

input [Regulator | Supplement] The input of the operator that regulates the expression of the output

output [Regulator | Reporter] The output of the operator that is regulated by the input

alpha [List] [Basal expression rate, Regulated expression rate in MEFL/second]

K [int | float] Half expression input concentration in Molar

n [int | float] Hill coefficient, cooperative degree (unitless)

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

unit: str, optional Units of the characterization data

characterize(flappjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

__str__(self)

characterize(self, flappjack, vector, media, strain, signal, biomass_signal)

expression_rate(self, t, dt)

forward_model(self, a=0, b=1, K_A=1, n_A=2, Dt=0.05, sim_steps=10, A=[0], odval=[1] * 100, gamma=0, p0=0, nt=100)

residuals(self, df, oddf)

class Regulator(name, init_concentration=0, degradation_rate=0, sbol_comp=None, color='lightgreen')
Bases: [GeneProduct](#)

Representation of a regulatory gene product. Child of GeneProduct.

class Regulator(name, init_concentration=0, degradation_rate=0, sbol_comp=None, color='lightgreen')
Bases: [GeneProduct](#)

Representation of a regulatory gene product. Child of GeneProduct.

class Reporter(name, init_concentration=0, degradation_rate=0, signal_id=None, color='w', sbol_comp=None)
Bases: [GeneProduct](#)

Representation of a regulatory gene product.

signal_id [str, optional] Flappjack ID of the signal that the reporter is associated with.

color [str, optional] Color of the reporter

class Reporter(name, init_concentration=0, degradation_rate=0, signal_id=None, color='w', sbol_comp=None)
Bases: [GeneProduct](#)

Representation of a regulatory gene product.

signal_id [str, optional] Flappjack ID of the signal that the reporter is associated with.

color [str, optional] Color of the reporter

class Sample(genetic_network=None, metabolism=None, assay=None, media=None, strain=None)
Representation of a sample that encapsulates GeneticNetwork and Metabolism. Incorporate environment information such as Supplements or chemicals, strain and media. Ex: 1 well in a plate, single cell. ...

genetic_network [GeneticNetwork] genetic network that is part of the sample
metabolism [Metabolism] metabolism that drives the genetic network in the sample
assay [Assay] assay to which this sample belongs
media [str] Name of the media in the sample
strain [str]

Name of the strain in the sample

Methods

add_supplement(supplement, concentration) establishes the concentration of Supplement

initialize(self)

set_regulator(self, name, concentration)

set_reporter(self, name, concentration)

set_supplement(self, supplement, concentration)

step(self, t, dt, stochastic=False)

class SimulatedMetabolism(name, biomass, growth_rate)

Bases: *Metabolism*

Simulated context for gene expression, incorporates biomass and growth rate. ...

name [str, optional] Name of the metabolism or corresponding strain

biomass A function of time that describes biomass $f(t)=\text{biomass}$

growth_rate A function of time that describes the growth rate $f(t)=\text{growth rate}$

class Source(output, rate, uri=None, sbol_comp=None, color='blue', name=None)

Bases: *Operator*

A class that represents a DNA fragment that encode a genetic operator. The Source Operator is an abstraction of a constitutive promoter that produces output.

...

output [Regulator | Reporter] The output of the operator that is constitutively expressed

rate [float] Output constitutive expression rate in MEFL/second

uri [str, optional] SynBioHub URI

sbol_comp [SBOL Component, optional] SBOL Component

name [str, optional] Name of the operator displayed on the network representation

color: str, optional Color displayed on the network representation

characterize(flappjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the Operator model that maps Input concentration into Output expression rate

__str__(self)

characterize(self, flappjack, vector, media, strain, signal, biomass_signal)

expression_rate(self, t, dt)

forward_model(self, Dt=0.25, sim_steps=10, odval=[1] * 97, rate=1, gamma=0, p0=0, nt=100)

```
residuals(self, df, oddf)

class Source(output, rate, uri=None, sbol_comp=None, color='blue', name=None)
    Bases: Operator

    A class that represents a DNA fragment that encode a genetic operator. The Source Operator is an abstraction of
    a constitutive promoter that produces output.

    ...

    output [Regulator | Reporter] The output of the operator that is constitutively expressed
    rate [float] Output constitutive expression rate in MEFL/second
    uri [str, optional] SynBioHub URI
    sbol_comp [SBOL Component, optional] SBOL Component
    name [str, optional] Name of the operator displayed on the network representation
    color: str, optional Color displayed on the network representation

    characterize(flapjack, receiver, inverter, media, strain, signal, biomass_signal, gamma) Parameterize the
        Operator model that maps Input concentration into Output expression rate

    __str__(self)

    characterize(self, flapjack, vector, media, strain, signal, biomass_signal)
    expression_rate(self, t, dt)
    forward_model(self, Dt=0.25, sim_steps=10, odval=[1] * 97, rate=1, gamma=0, p0=0, nt=100)
    residuals(self, df, oddf)

class Supplement(name, pubchemid=None, supplier_id=None, sbol_comp=None, color='pink')
    Representation of a chemical

    ...

    name [str] Name of the supplement
    concentration [int | float] concentration of the supplement in Molar
    pubchemid [str] PubChemID URI of the supplement
    supplier_id [str] Supplier ID of the supplement. An URL of the product that you aquire. Accepts list of the
        form [product URL, catalog number, batch].
    sbol_comp [str] SBOL component of the supplement.

    __str__(self)
        Return str(self).

class Supplement(name, pubchemid=None, supplier_id=None, sbol_comp=None, color='pink')
    Representation of a chemical

    ...

    name [str] Name of the supplement
    concentration [int | float] concentration of the supplement in Molar
    pubchemid [str] PubChemID URI of the supplement
    supplier_id [str] Supplier ID of the supplement. An URL of the product that you aquire. Accepts list of the
        form [product URL, catalog number, batch].
```


sbol_comp [str] SBOL component of the supplement.

__str__(*self*)
Return str(*self*).

characterize_growth(*flapjack*, *vector*, *media*, *strain*, *biomass_signal*, *n_gaussians*, *epsilon*)

forward_model_growth(*Dt*=0.05, *sim_steps*=10, *muval*=[0] * 100, *od0*=0, *nt*=100)

gompertz(*t*, *y0*, *ymax*, *um*, *l*)

gompertz_growth_rate(*t*, *y0*, *ymax*, *um*, *l*)

load_loica(*filename*)

ramp_biomass(*t*, *od0*, *start*, *slope*)

ramp_growth_rate(*t*, *start*, *slope*)

residuals_growth(*data*, *epsilon*, *dt*, *t*, *n_gaussians*)

save_loica(*obj*, *filename*)

step_biomass(*t*, *od0*, *start*)

step_growth_rate(*t*, *start*)

INTRODUCTION

Welcome to the LOICA (Logical Operators for Integrated Cell Algorithms) repository, our Python package for designing, modeling and characterizing genetic networks.

As you may have noticed, our logo features a beautiful bird—loica ([Leistes loyca](#)); a bird native to Chile known for its particular red chest and legendary kindness, with which we share name.

INSTALLATION

Installing LOICA is way easier than pronouncing it!

```
pip install loica
```

For more details please refer to our [Wiki](#) for installation instructions and developer guides.

LOICA ALLOWS YOU TO:

- Compile Code into DNA fragments that execute Cell Algorithms
- Easy programation of genetic network models
- Generation of synthetic data
- Communicate with Flapjack
- Use and output SBOL files
- Use all sorts of cellular computation
- Easy, fluid and customisable DNA design

TUTORIALS

Now that you have LOICA installed you can familiarize yourself with the tool using the [Jupyter notebook](#) tutorials designed for this purpose.

PYTHON MODULE INDEX

|

- loica, 9
- loica.assay, 20
- loica.colony, 21
- loica.geneproduct, 21
- loica.genetic_network, 22
- loica.metabolism, 23
- loica.operators, 9
 - loica.operators.hill1, 9
 - loica.operators.hill2, 10
 - loica.operators.operator, 11
 - loica.operators.receiver, 11
 - loica.operators.source, 12
- loica.sample, 24
- loica.supplement, 25
- loica.util, 26

Symbols

__str__() (*GeneProduct* method), 22, 29
 __str__() (*Hill1* method), 10, 14, 30, 31
 __str__() (*Hill2* method), 10, 15, 31, 32
 __str__() (*Operator* method), 11, 15, 16, 32, 33
 __str__() (*Receiver* method), 12, 17, 18, 33, 34
 __str__() (*Source* method), 13, 18–20, 35, 36
 __str__() (*Supplement* method), 25, 36, 37

A

add_operator() (*GeneticNetwork* method), 22, 29
 add_regulator() (*GeneticNetwork* method), 22, 29
 add_reporter() (*GeneticNetwork* method), 23, 29
 Assay (class in *loica*), 27
 Assay (class in *loica.assay*), 20

B

biomass() (*DataMetabolism* method), 24, 28

C

characterize() (*Hill1* method), 10, 14, 30, 31
 characterize() (*Hill2* method), 11, 15, 31, 32
 characterize() (*Receiver* method), 12, 17, 18, 33, 34
 characterize() (*Source* method), 13, 18–20, 35, 36
 characterize_growth() (in module *loica*), 37
 characterize_growth() (in module *loica.util*), 26
 Colony (class in *loica*), 28
 Colony (class in *loica.colony*), 21

D

DataMetabolism (class in *loica*), 28
 DataMetabolism (class in *loica.metabolism*), 23
 draw() (*GeneticNetwork* method), 23, 29

E

express() (*GeneProduct* method), 22, 29
 expression_rate() (*Hill1* method), 10, 14, 30, 31
 expression_rate() (*Hill2* method), 11, 15, 31, 32
 expression_rate() (*Receiver* method), 12, 17, 18, 33, 34
 expression_rate() (*Source* method), 13, 18–20, 35, 36

F

forward_model() (*Hill1* method), 10, 14, 30, 31
 forward_model() (*Hill2* method), 11, 15, 31, 32
 forward_model() (*Receiver* method), 12, 17, 18, 33, 34
 forward_model() (*Source* method), 13, 18–20, 35, 36
 forward_model_growth() (in module *loica*), 37
 forward_model_growth() (in module *loica.util*), 26
 fun() (*Colony* method), 21, 28

G

GeneProduct (class in *loica*), 28
 GeneProduct (class in *loica.geneproduct*), 21
 GeneticNetwork (class in *loica*), 29
 GeneticNetwork (class in *loica.genetic_network*), 22
 gompertz() (in module *loica*), 37
 gompertz() (in module *loica.metabolism*), 24
 gompertz_growth_rate() (in module *loica*), 37
 gompertz_growth_rate() (in module *loica.metabolism*), 24
 growth_rate() (*DataMetabolism* method), 24, 28

H

Hill1 (class in *loica*), 29, 30
 Hill1 (class in *loica.operators*), 14
 Hill1 (class in *loica.operators.hill1*), 9
 Hill2 (class in *loica*), 31
 Hill2 (class in *loica.operators*), 14
 Hill2 (class in *loica.operators.hill2*), 10

I

initialize() (*GeneProduct* method), 22, 29
 initialize() (*GeneticNetwork* method), 23, 29
 initialize() (*Sample* method), 25, 35

K

kymograph() (*Colony* method), 21, 28

L

load_loica() (in module *loica*), 37
 load_loica() (in module *loica.util*), 26
 loica

- module, 9
- loica.assay
 - module, 20
- loica.colony
 - module, 21
- loica.geneproduct
 - module, 21
- loica.genetic_network
 - module, 22
- loica.metabolism
 - module, 23
- loica.operators
 - module, 9
- loica.operators.hill1
 - module, 9
- loica.operators.hill2
 - module, 10
- loica.operators.operator
 - module, 11
- loica.operators.receiver
 - module, 11
- loica.operators.source
 - module, 12
- loica.sample
 - module, 24
- loica.supplement
 - module, 25
- loica.util
 - module, 26

M

- map_kymo() (*Colony method*), 21, 28
- Metabolism (*class in loica*), 32
- Metabolism (*class in loica.metabolism*), 24
- module
 - loica, 9
 - loica.assay, 20
 - loica.colony, 21
 - loica.geneproduct, 21
 - loica.genetic_network, 22
 - loica.metabolism, 23
 - loica.operators, 9
 - loica.operators.hill1, 9
 - loica.operators.hill2, 10
 - loica.operators.operator, 11
 - loica.operators.receiver, 11
 - loica.operators.source, 12
 - loica.sample, 24
 - loica.supplement, 25
 - loica.util, 26

N

- norm_kymo() (*Colony method*), 21, 28

O

- Operator (*class in loica*), 32
- Operator (*class in loica.operators*), 15, 16
- Operator (*class in loica.operators.operator*), 11

R

- ramp_biomass() (*in module loica*), 37
- ramp_biomass() (*in module loica.metabolism*), 24
- ramp_growth_rate() (*in module loica*), 37
- ramp_growth_rate() (*in module loica.metabolism*), 24
- Receiver (*class in loica*), 33
- Receiver (*class in loica.operators*), 16, 17
- Receiver (*class in loica.operators.receiver*), 11
- Regulator (*class in loica*), 34
- Regulator (*class in loica.geneproduct*), 22
- Reporter (*class in loica*), 34
- Reporter (*class in loica.geneproduct*), 22
- residuals() (*Hill1 method*), 10, 14, 30, 31
- residuals() (*Hill2 method*), 11, 15, 31, 32
- residuals() (*Receiver method*), 12, 17, 18, 33, 34
- residuals() (*Source method*), 13, 19, 20, 35, 36
- residuals_growth() (*in module loica*), 37
- residuals_growth() (*in module loica.util*), 26
- run() (*Assay method*), 21, 28

S

- Sample (*class in loica*), 34
- Sample (*class in loica.sample*), 24
- save_loica() (*in module loica*), 37
- save_loica() (*in module loica.util*), 26
- set_regulator() (*Sample method*), 25, 35
- set_reporter() (*Sample method*), 25, 35
- set_supplement() (*Sample method*), 25, 35
- shape (*GeneProduct attribute*), 22, 29
- SimulatedMetabolism (*class in loica*), 35
- SimulatedMetabolism (*class in loica.metabolism*), 24
- Source (*class in loica*), 35, 36
- Source (*class in loica.operators*), 18–20
- Source (*class in loica.operators.source*), 12
- step() (*GeneProduct method*), 22, 29
- step() (*GeneticNetwork method*), 23, 29
- step() (*Sample method*), 25, 35
- step_biomass() (*in module loica*), 37
- step_biomass() (*in module loica.metabolism*), 24
- step_growth_rate() (*in module loica*), 37
- step_growth_rate() (*in module loica.metabolism*), 24
- step_stochastic() (*GeneticNetwork method*), 23, 29
- substep_stochastic() (*GeneticNetwork method*), 23, 29
- Supplement (*class in loica*), 36
- Supplement (*class in loica.supplement*), 25

T

`to_contracted_graph()` (*GeneticNetwork method*),
[23](#), [29](#)

`to_graph()` (*GeneticNetwork method*), [23](#), [29](#)

`to_sbol()` (*GeneticNetwork method*), [23](#), [29](#)

U

`upload()` (*Assay method*), [21](#), [28](#)