
shmir Documentation

Release 2.0

Sylewster Brzęczkowski, Mateusz Flieger, Piotr Rogulski, Michał R

September 03, 2015

1	shmir package	3
1.1	Subpackages	3
1.2	Submodules	8
1.3	shmir.async module	8
1.4	shmir.contextmanagers module	8
1.5	shmir.decorators module	8
1.6	shmir.mfold module	8
1.7	shmir.result_handlers module	8
1.8	shmir.settings module	9
1.9	shmir.testing module	9
1.10	shmir.utils module	10
1.11	shmir.views module	10
1.12	Module contents	11
2	Indices and tables	13
	Python Module Index	15

Contents:

shmir package

1.1 Subpackages

1.1.1 shmir.data package

Submodules

shmir.data.models module

```
class shmir.data.models.Backbone (**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Base
    Backbone class with information about miRNA scaffolds
    active_strand
    flanks3_a
    flanks3_s
    flanks5_a
    flanks5_s
    generate_regexp()
        Function creates regexps based on active_strand and saves it to the database.
        active_strand: if is equal to 3, function use miRNA_a; if is equal to 1 or 5, function use miRNA_s if is
            equal to 0, function use both
    classmethod generate_regexp_all()
        Function takes all objects from the database and creates regular expressions for each.
    homogeneity
    id
    loop_a
    loop_s
    mirBase_link
    miRNA_a
    miRNA_end_3
```

`miRNA_end_5`
`miRNA_length`
`miRNA_max`
`miRNA_min`
`miRNA_s`

`name`
`regex`
`siRNA1 = None`
`siRNA2 = None`

`structure`

`template()`

Returns the template of DNA (sh-miR)

siRNA1 and siRNA2 are siRNA strands and they must be initialized before using this method

Returns: Sequence of sh-miR molecule on the base of chosen miRNA scaffold

class `shmir.data.models.Immuno(**kwargs)`

Bases: `sqlalchemy.ext.declarative.api.Base`

Immuno motives class

classmethod `check_is_in_sequence(input_sequence)`

Checks if input sequence contains sequences from immuno database

Args: `input_sequence`: RNA sequence of about 20nt length

Returns: Bool if the `input_sequence` contains immunostimulatory motifs

`id`

`link`

`receptor`

`sequence`

class `shmir.data.models.InputData(**kwargs)`

Bases: `sqlalchemy.ext.declarative.api.Base`

Table storing input data to sh-miR algorithm

`id`

`immunostimulatory`

`maximum_CG`

`maximum_offtarget`

`minimum_CG`

`results`

`scaffold`

`transcript_name`

```
class shmir.data.models.Result (**kwargs)
    Bases: sqlalchemy.ext.declarative.api.Base
    sh-miR results table
    as_json()
    backbone
    get_task_id()
    id
    input_data
    input_id
    pdf
    score
    sequence
    shmir
```

shmir.data.ncbi_api module

```
shmir.data.ncbi_api.get_data (transcript,                                database='nucleotide',
                               email='amupoznan@gmail.com')
    Function responsible to get data from given ncbi database.
```

Args: email(str): Email to authorization with ncbi. transcript(str): Transcript to search. database(str): Name of database from ncbi. ids(list of str): List of ids to search.

```
shmir.data.ncbi_api.get_mRNA (transcript,                                database='nucleotide',
                               email='amupoznan@gmail.com')
    Function to connect with NCBI database and get transcript by name
```

Args: transcript(str): name of transcript (from NCBI) database: name of database in which we look for (default “nucleotide”) email(str): email to which NCBI needs to validate

Returns: mRNA(str)

Raises: NoResultError

Module contents

1.1.2 shmir.designer package

Submodules

shmir.designer.design module

shmir.designer.errors module

```
exception shmir.designer.errors.BaseException (message=None)
    Bases: exceptions.Exception
    Base exception class.
```

exception `shmir.designer.errors.IncorrectDataError` (*message=None*)

Bases: `shmir.designer.errors.BaseException`

Exception error class for incorrect data input

exception `shmir.designer.errors.NoResultError` (*message=None*)

Bases: `shmir.designer.errors.BaseException`

Exception error class for no results

exception `shmir.designer.errors.ValidationError` (*message=None*)

Bases: `shmir.designer.errors.BaseException`

Exception error class for incorrect input

shmir.designer.offtarget module

shmir.designer.score module

shmir.designer.search module

shmir.designer.ss module

shmir.designer.utils module

`shmir.designer.utils.adjusted_frames` (*seq1, seq2, shift_left, shift_right, all_frames*)

Take output of `parse_input` function and insert into flanking sequences. take from database all miRNA results and check if ends of input is suitable for flanking sequences. If first value == and `miRNA_end_5` second value == `miRNA_end_3` then simply concatenate sequences `flanks5_s` + `first_sequence` + `loop_s` + `second_sequence` + `flanks3_s`. If any end is different function has to modify end of the insert: Right end: if `miRNA_end_5` < `first_end` add to right site of second sequence additional nucleotides (as many as `miRNA_end_5 - first_end`) like (dots are nucleotides to add, big letter are flanking sequences, small are input):

AAAGGGGCTTTTagtcttaga TTTCCCCGAA....agaatct

if `miRNA_end_5` > `first_end` cut nucleotides from right site of `flanks3_s` and/or from right site of second sequence

before cut: AAAGGGGCTTTTagtcttaga TTTCCCCGAAAATTcctcagaatct (-2, +2)

After AAAGGGGCTTTTagtcttaga TTTCCCCGAAAAtcagaatct

Nucleotides are always added to the right side of sequences. We cut off nucleotides only from flanking sequences or loop.

Args: `seq1(str)` first sequence. `seq2(str)` second sequence. `shift_left(int)` - shift on sequence from left side. `shift_right(int)` - shift on sequence from right side. `all_frames(pri-miRNA objects)` frames from which we create sh-miRs.

Returns: list of tuples (changed frame, first sequence, second sequence).

`shmir.designer.utils.reverse_complement` (*sequence*)

Generates reverse complement sequence to given

Args: `sequence(str)`.

Returns: reverse complement sequence(str) to given.

shmir.designer.validators module

Module contents

1.1.3 shmir.tests package

Submodules

shmir.tests.test_db module

```
class shmir.tests.test_db.TestBackboneModel (methodName='runTest')
    Bases: shmir.testing.TestModelBase

    test_adjusted_frames ()

    test_reverse_complement ()

    test_template ()

class shmir.tests.test_db.TestModelBaseCase (methodName='runTest')
    Bases: shmir.testing.TestModelBase

    class TestModel (**kwargs)
        Bases: sqlalchemy.ext.declarative.api.Base

        id

        TestModelBaseCase.test_get_and_put_to_db ()
```

shmir.tests.test_ncbi_api module

```
class shmir.tests.test_ncbi_api.TestGetDataFromNcbi (methodName='runTest')
    Bases: unittest.case.TestCase

    BAD_MNRA = 'NC_000071.6'

    DATA_BASE = 'nucleotide'

    GOOD_MRNA = 'NM_001128164.1'

    test_get_data (*args, **kwargs)

    test_get_mRNA (*args, **kwargs)

    test_incorrect_get_mRNA (*args, **kwargs)

    test_no_results_get_mRNA (*args, **kwargs)
```

shmir.tests.test_validators module

```
class shmir.tests.test_validators.TestTranscriptValidators (methodName='runTest')
    Bases: unittest.case.TestCase

    test_calculate_gc_content ()

    test_validate_gc_content ()

    test_validate_gc_content_not_in_range ()
```

Module contents

1.2 Submodules

1.3 shmir.async module

`shmir.async.get_async_result(task, task_id, timeout=1.0, only_status=False)`

Gets AsyncResult of task, excepting TimeoutError and handling failures

Args: task: the task from which want extract result from task_id: id of task timeout(float): timeout of task only_status(bool): boolean if function should return only status (default: False)

Returns: Task response

1.4 shmir.contextmanagers module

`shmir.contextmanagers.blast_path(*args, **kws)`

Context manager which changes path for blast executions

`shmir.contextmanagers.generic_path(path)`

Function which changes current path to given path

Args: path(str): path where interpreter should be

`shmir.contextmanagers.mfold_path(*args, **kws)`

Context manager which changes path for mfold tasks

Args: task_id: id of mfold task

1.5 shmir.decorators module

`shmir.decorators.catch_errors(*errors)`

Decorator to catch specific errors given

Args: *errors: All error which should be caught

Returns: dict with error status and its message or function result

`shmir.decorators.send_email(file_handler=None)`

Decorator to send email after task

Args: file_handler: function to handle files

1.6 shmir.mfold module

1.7 shmir.result_handlers module

`shmir.result_handlers.zip_file_mfold(path)`

Function which creates zip_msg to send via email

Args: path: path of zip file

Returns: tuple of zip message

`shmir.result_handlers.zip_files_from_siRNA` (*struct*)
Function which generates email msg from siRNA

Args: struct: sh-miR struct

Returns: zip message

`shmir.result_handlers.zip_files_from_transcript` (*struct*)
Function which generates email msg from transcript

Args: struct: sh-miR struct

Returns: zip message

1.8 shmir.settings module

`shmir.settings.get_bool` (*section, option, default=None*)
Function which gets settings

Args: section(str): section of settings option(str): option of settings

Returns: config(bool) or default value

`shmir.settings.get_config` (*section, option, default=None*)
Function which gets settings

Args: section(str): section of settings option(str): option of settings

Returns: config or default value

`shmir.settings.get_db_config` (*option, default=None*)

`shmir.settings.get_int` (*section, option, default=None*)
Function which gets settings

Args: section(str): section of settings option(str): option of settings

Returns: config(int) or default value

1.9 shmir.testing module

`class shmir.testing.TestModelBase` (*methodName='runTest'*)
Bases: `unittest.case.TestCase`

`get_all` (*model*)

`put_to_db` (*obj*)

`setUp` ()

`tearDown` ()

`shmir.testing.create_backbone` ()
Function to create default backbone object

1.10 shmir.utils module

1.11 shmir.views module

`shmir.views.mfold_task_creator(*args, **kwargs)`

Handler to create folded structure via mfold for specific sequence.

Args: data: sequence

Returns: Task id of this task

`shmir.views.mfold_task_result(task_id)`

Handler for getting mfold result from just a task id

Args: task_id: Id of task generated via RESTful API

Returns: Json object of status or sends zipped file

`shmir.views.mfold_task_status(task_id)`

Mfold status getter

Args: task_id: Id of task generated via RESTful API

Returns: Json object with status of specific id

`shmir.views.scaffolds(*args, **kwargs)`

Handler to list all possible backbones

Returns: Json object with names of backbones list

`shmir.views.sirna_task_creator(*args, **kwargs)`

Handler to initialize task which creates sh-miR(s) from siRNA

Args:

data: one siRNA strand (active) or two siRNA strands separated by space. First strand is active, both are in 5-3 orientation.

Returns: Task id

`shmir.views.sirna_task_result(task_id)`

Handler to give result of sh-miR(s) created from siRNA

Args: task_id: Id of task generated via RESTful API

Returns: Json object with sh-miRs created from siRNA: [(score, sh-miR, bacbone name, pdf(id to download pdf via mfold)...]

`shmir.views.sirna_task_status(task_id)`

Handler to check status of task which creates sh-miR from siRNA

Args: task_id: Id of task generated via RESTful API

Returns: Json object with status of given task_id

`shmir.views.transcript_task_creator(*args, **kwargs)`

Handler to create sh-miR from transcript

Args: transcript_name: name of transcript

Returns: Id of task

`shmir.views.transcript_task_result(task_id)`

Handler to get results of task which creates sh-miR(s) from transcript

Args: task_id: Id of task generated via RESTful API

Returns: Json object with list of sh-miRs in structure: [sh-miR, score, pdf(id to download pdf via mfold), sequence and backbone name...]

`shmir.views.transcript_task_status(task_id)`

Handler to check status of task which creates sh-miR from transcript

Args: task_id: Id of task generated via RESTful API

Returns: Json object with status of given task_id

1.12 Module contents

Indices and tables

- *genindex*
- *modindex*
- *search*

S

- shmir, 11
- shmir.async, 8
- shmir.context_managers, 8
- shmir.contextmanagers, 8
- shmir.data, 5
- shmir.data.models, 3
- shmir.data.ncbi_api, 5
- shmir.decorators, 8
- shmir.designer, 7
- shmir.designer.error, 5
- shmir.designer.errors, 5
- shmir.designer.models, 3
- shmir.designer.utils, 6
- shmir.result_handlers, 8
- shmir.settings, 9
- shmir.testing, 9
- shmir.tests, 8
- shmir.tests.test_db, 7
- shmir.tests.test_ncbi_api, 7
- shmir.tests.test_validators, 7
- shmir.views, 10

A

active_strand (shmir.data.models.Backbone attribute), 3
adjusted_frames() (in module shmir.designer.utils), 6
as_json() (shmir.data.models.Result method), 5

B

Backbone (class in shmir.data.models), 3
backbone (shmir.data.models.Result attribute), 5
BAD_MNRA (shmir.tests.test_ncbi_api.TestGetDataFromNcbi attribute), 7
BaseException, 5
blast_path() (in module shmir.contextmanagers), 8

C

catch_errors() (in module shmir.decorators), 8
check_is_in_sequence() (shmir.data.models.Immuno class method), 4
create_backbone() (in module shmir.testing), 9

D

DATA_BASE (shmir.tests.test_ncbi_api.TestGetDataFromNcbi attribute), 7

F

flanks3_a (shmir.data.models.Backbone attribute), 3
flanks3_s (shmir.data.models.Backbone attribute), 3
flanks5_a (shmir.data.models.Backbone attribute), 3
flanks5_s (shmir.data.models.Backbone attribute), 3

G

generate_regexp() (shmir.data.models.Backbone method), 3
generate_regexp_all() (shmir.data.models.Backbone class method), 3
generic_path() (in module shmir.contextmanagers), 8
get_all() (shmir.testing.TestModelBase method), 9
get_async_result() (in module shmir.async), 8
get_bool() (in module shmir.settings), 9
get_config() (in module shmir.settings), 9
get_data() (in module shmir.data.ncbi_api), 5

get_db_config() (in module shmir.settings), 9
get_int() (in module shmir.settings), 9
get_mRNA() (in module shmir.data.ncbi_api), 5
get_task_id() (shmir.data.models.Result method), 5
GOOD_MRNA (shmir.tests.test_ncbi_api.TestGetDataFromNcbi attribute), 7

H

homogeneity (shmir.data.models.Backbone attribute), 3

I

id (shmir.data.models.Backbone attribute), 3
id (shmir.data.models.Immuno attribute), 4
id (shmir.data.models.InputData attribute), 4
id (shmir.data.models.Result attribute), 5
id (shmir.tests.test_db.TestModelBaseCase.TestModel attribute), 7
Immuno (class in shmir.data.models), 4
immunostimulatory (shmir.data.models.InputData attribute), 4
IncorrectDataError, 5
input_data (shmir.data.models.Result attribute), 5
input_id (shmir.data.models.Result attribute), 5
InputData (class in shmir.data.models), 4

L

link (shmir.data.models.Immuno attribute), 4
loop_a (shmir.data.models.Backbone attribute), 3
loop_s (shmir.data.models.Backbone attribute), 3

M

maximum_CG (shmir.data.models.InputData attribute), 4
maximum_offtarget (shmir.data.models.InputData attribute), 4
mfold_path() (in module shmir.contextmanagers), 8
mfold_task_creator() (in module shmir.views), 10
mfold_task_result() (in module shmir.views), 10
mfold_task_status() (in module shmir.views), 10
minimum_CG (shmir.data.models.InputData attribute), 4
miRBase_link (shmir.data.models.Backbone attribute), 3

miRNA_a (shmir.data.models.Backbone attribute), 3
miRNA_end_3 (shmir.data.models.Backbone attribute), 3
miRNA_end_5 (shmir.data.models.Backbone attribute), 3
miRNA_length (shmir.data.models.Backbone attribute), 4
miRNA_max (shmir.data.models.Backbone attribute), 4
miRNA_min (shmir.data.models.Backbone attribute), 4
miRNA_s (shmir.data.models.Backbone attribute), 4

N

name (shmir.data.models.Backbone attribute), 4
NoResultError, 6

P

pdf (shmir.data.models.Result attribute), 5
put_to_db() (shmir.testing.TestModelBase method), 9

R

receptor (shmir.data.models.Immuno attribute), 4
regexp (shmir.data.models.Backbone attribute), 4
Result (class in shmir.data.models), 4
results (shmir.data.models.InputData attribute), 4
reverse_complement() (in module shmir.designer.utils), 6

S

scaffold (shmir.data.models.InputData attribute), 4
scaffolds() (in module shmir.views), 10
score (shmir.data.models.Result attribute), 5
send_email() (in module shmir.decorators), 8
sequence (shmir.data.models.Immuno attribute), 4
sequence (shmir.data.models.Result attribute), 5
setUp() (shmir.testing.TestModelBase method), 9
shmir (module), 11
shmir (shmir.data.models.Result attribute), 5
shmir.async (module), 8
shmir.context_managers (module), 8
shmir.contextmanagers (module), 8
shmir.data (module), 5
shmir.data (module), 3
shmir.data.ncbi_api (module), 5
shmir.decorators (module), 8
shmir.designer (module), 7
shmir.designer.error (module), 5
shmir.designer.errors (module), 5
shmir.designer.models (module), 3
shmir.designer.utils (module), 6
shmir.result_handlers (module), 8
shmir.settings (module), 9
shmir.testing (module), 9
shmir.tests (module), 8
shmir.tests.test_db (module), 7
shmir.tests.test_ncbi_api (module), 7
shmir.tests.test_validators (module), 7
shmir.views (module), 10
siRNA1 (shmir.data.models.Backbone attribute), 4

siRNA2 (shmir.data.models.Backbone attribute), 4
sirna_task_creator() (in module shmir.views), 10
sirna_task_result() (in module shmir.views), 10
sirna_task_status() (in module shmir.views), 10
structure (shmir.data.models.Backbone attribute), 4

T

tearDown() (shmir.testing.TestModelBase method), 9
template() (shmir.data.models.Backbone method), 4
test_adjusted_frames() (shmir.tests.test_db.TestBackboneModel method), 7
test_calculate_gc_content()
(shmir.tests.test_validators.TestTranscriptValidators method), 7
test_get_and_put_to_db()
(shmir.tests.test_db.TestModelBaseCase method), 7
test_get_data() (shmir.tests.test_ncbi_api.TestGetDataFromNcbi method), 7
test_get_mRNA() (shmir.tests.test_ncbi_api.TestGetDataFromNcbi method), 7
test_incorrect_get_mRNA()
(shmir.tests.test_ncbi_api.TestGetDataFromNcbi method), 7
test_no_results_get_mRNA()
(shmir.tests.test_ncbi_api.TestGetDataFromNcbi method), 7
test_reverse_complement()
(shmir.tests.test_db.TestBackboneModel method), 7
test_template() (shmir.tests.test_db.TestBackboneModel method), 7
test_validate_gc_content()
(shmir.tests.test_validators.TestTranscriptValidators method), 7
test_validate_gc_content_not_in_range()
(shmir.tests.test_validators.TestTranscriptValidators method), 7
TestBackboneModel (class in shmir.tests.test_db), 7
TestGetDataFromNcbi (class in shmir.tests.test_ncbi_api), 7
TestModelBase (class in shmir.testing), 9
TestModelBaseCase (class in shmir.tests.test_db), 7
TestModelBaseCase.TestModel (class in shmir.tests.test_db), 7
TestTranscriptValidators (class in shmir.tests.test_validators), 7
transcript_name (shmir.data.models.InputData attribute), 4
transcript_task_creator() (in module shmir.views), 10
transcript_task_result() (in module shmir.views), 10
transcript_task_status() (in module shmir.views), 11

V

`ValidationError`, [6](#)

Z

`zip_file_mfold()` (in module `shmir.result_handlers`), [8](#)

`zip_files_from_sirna()` (in module `shmir.result_handlers`),
[9](#)

`zip_files_from_transcript()` (in module
`shmir.result_handlers`), [9](#)