Welcome to Rice recombination predictor's documentation!

- Written by Camila Riccio, Mauricio Peñuela, Camilo Rocha and Jorge Finke

- Last update: 04/04/22

Rice recombination predictor

This is a Python3 implementation to predict local chromosomal recombination in rice using sequence identity, combined with other features derived from genome alignment (including the number of variants, inversions, absent bases, and CentO sequences).

Preliminaries

In order for the user to predict recombination between two parental rice varieties, arbitrarily select one of them as the reference genome and the other as the query genome. As an example we will take the IR64 variety as reference genome and Azucena as query, and we will use the data from chromosome 01 of both varieties to calibrate the prediction model.

We suggest organizing the data as follows:

- **input_data** folder containing:
  - **IR64** folder containig:
    - 12 fasta files with the amino acid sequences of each chromosome of the IR64 variety. Download from NCBI Genome database searching the accession number RWKJ00000000.
  - **Azucena** folder containig:
    - 12 fasta files with the amino acid sequences of each chromosome of the Azucena variety. Download from NCBI Genome database searching the accession number PKQC000000000.
  - **coords** folder (Empty, the corresponding files will be generated later).
  - **snps** folder (Empty, the corresponding files will be generated later).
  - **recombination** folder containig:
    - 12 csv files with windowed experimental recombination values for each chromosome.
  - **CentO** fasta file.
- outpud_data folder (Empty, the corresponding files will be generated later).

The alignment process between the two parental chromosomes must be done independently using the MUMmer software, with the commands shown below executed from the **input_data** folder:

Align reference and query fasta files:

```
nucmer --prefix=IR64_Azucena_chr01 IR64/Osat_IR64_chr01.fasta Azucena/
```

Filter the aligned data:

```
delta-filter -r -q IR64_Azucena_chr01.delta > IR64_Azucena_chr01.filte
```

Extract contig coordinates from the filtered file:

```
show-coords -r IR64_Azucena_chr01.filter > coords/IR64_Azucena_chr01.c
```

Extract variants from the filtered file:

```
show-snps -lr -x 1  -T IR64_Azucena_chr01.filter  > snps/IR64_Azucena_
```

We are now ready to make use of the Rice recombination predictor software.

Setup

Clone the repository:

```
git clone git@github.com/criccio35/Rice-recombination-predictor
```

Requirements

Install the requirements by entering the following commands in the terminal:

Install biopython module:

```
pip install biopython
```

Install Basic Local Alignment Search Tool (BLAST):

```
sudo apt update
sudo apt install ncbi-blast+
```

How to use

For optimal model performance, you also need experimental recombination data from at least one chromosome. Depending on the availability of experimental data, 3 cases can be presented:

- No experimental data at all.
- Experimental data for one chromosome only.
- Experimental data for all chromosomes.

We will show the example of the case in which the experimental data is available for all chromosomes and and along the way we will explain what modifications would be necessary for the other two cases.

Import module:

```
import rice_recombination_predictor as rrp
```

Specify the paths of the input files of the chromosome with which the model is to be calibrated:

```
ref_path = 'input_data/IR64/Osat_IR64_chr01.fasta'
qry_path = 'input_data/Azucena/Osat_Azucena_chr01.fasta'
coords_path = 'input_data/coords/IR64_Azucena_chr01.coords'
variants_path = 'input_data/snps/IR64_Azucena_chr01.snps'
rec_path = 'input_data/recombination/experimental_recombination_chr01.
CentO_path = 'input_data/CentO_AA.fasta'
results_path = 'output_data/'
```

Input window size:

```
size_w = 100_000
```

Instantiate the rice_recombination_predictor class:

```
chr_mod = rrp.rice_recombination_predictor(size_w, ref_path, qry_path,
                                           coords_path, variants_path,
                                           CentO_path, rec_path)
```

Call the method to preprocess the input files:

```
chr_mod.data_preprocessing()
```

Define the initial parameters p1,p2,p3,p4,p5,p5 and p7 (see **Software description** file for model details) and call the method to optimize them:

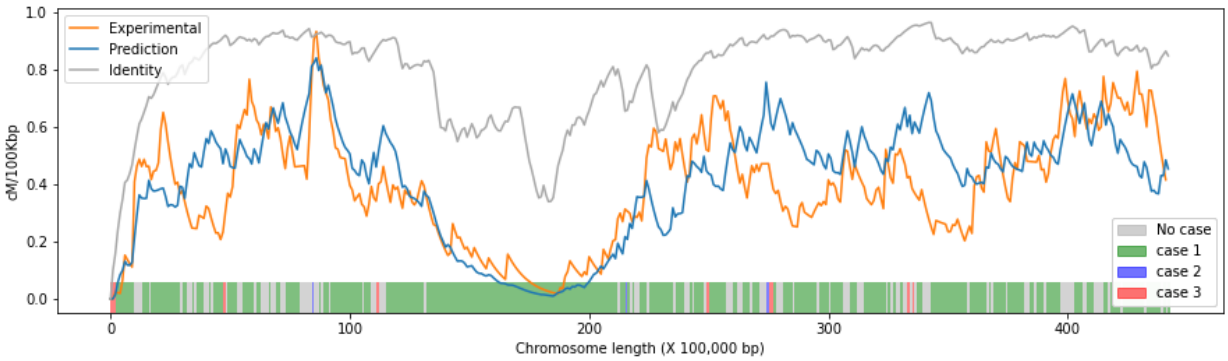
```
initial_parameters = [0.5,0.97,1,0.9,1,0,0.002]
chr_mod.optimize_model_parameters(initial_parameters)
```

You can access the predicted values with the following attribute of the class:

```
chr_mod.prediction
```

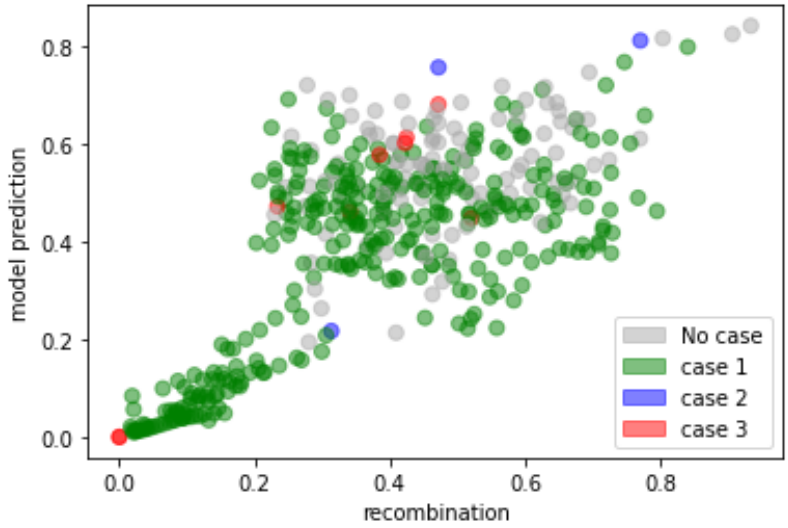
You can visualize the results of the prediction through the method:

```
chr_mod.plot_landscape()
```



You can visualize the performance of the prediction with respect to the experimental data through the method:

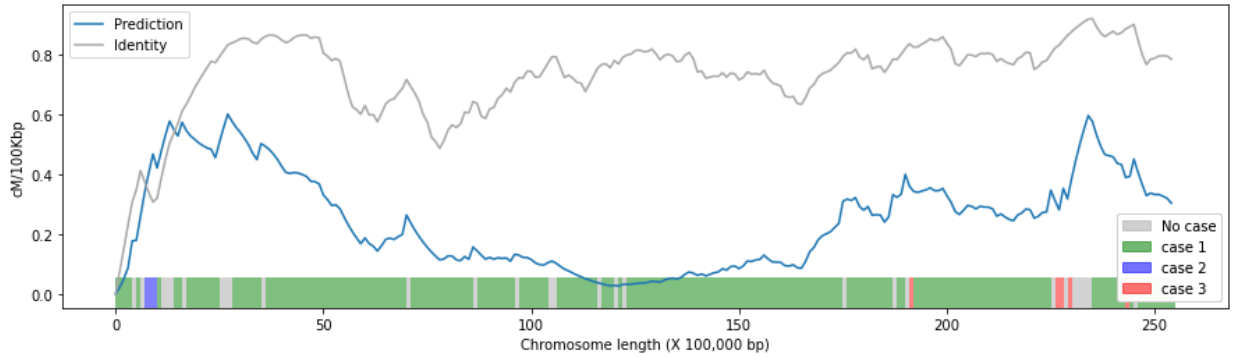
```
chr_mod.plot_correlation()
```



You can now use the optimized parameters to predict recombination on the other chromosomes. If **experimental recombination data is NOT available for the remaining chromosomes** you can proceed as follows:

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

chromosomes = ['01','02','03','04','05','06','07','08','09','10','11',

df_pred = pd.DataFrame()
df_eval = pd.DataFrame(columns=['chromosome','R2_idt','R2_pred','pears

for chr_nbr in chromosomes:
    ref_path = 'input_data/IR64/Osat_IR64_chr{0}.fasta'.format(chr_nbr
    qry_path = 'input_data/Azucena/Osat_Azucena_chr{0}.fasta'.format(c
    coords_path = 'input_data/coords/IR64_Azucena_chr{0}.coords'.forma
    variants_path = 'input_data/snps/IR64_Azucena_chr{0}.snps'.format(

    chr_tmp = rrp.rice_recombination_predictor(size_w, ref_path, qry_p
                                               coords_path, variants_
                                               CentO_path, params=chr

    print('---Chromosome {0}---'.format(chr_nbr))
    chr_tmp.data_preprocessing()
    chr_tmp.predict_recombination()

    df_pred1 = pd.DataFrame({'chr_'+chr_nbr : chr_tmp.prediction})
    df_pred = pd.concat([df_pred,df_pred1], axis=1)

    chr_tmp.plot_landscape()

df_pred.to_csv(results_path+'predictions.csv',header=True,index=True)
```

The above will generate a .csv file where each column corresponds to the prediction of recombination on a chromosome. At the same time, for each chromosome, a plot will be generated as follows:

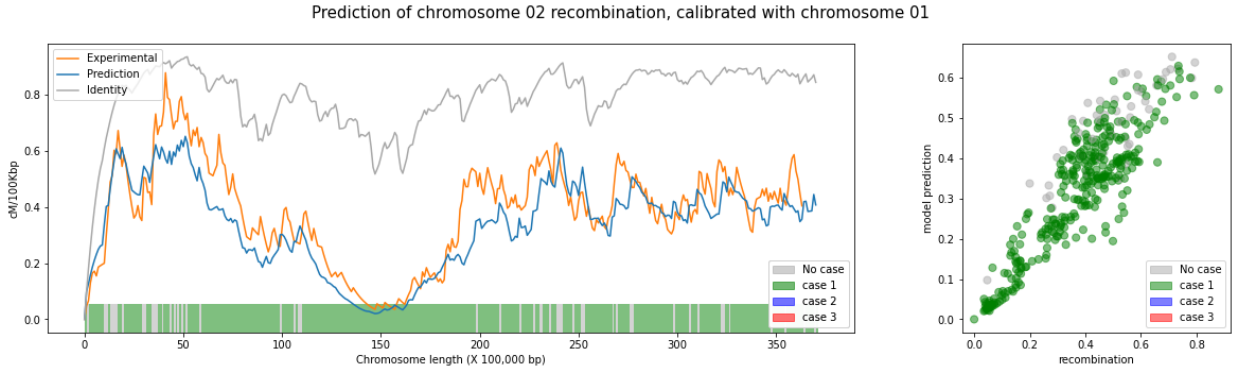


If **experimental recombination data are available for all chromosomes** you can proceed as follows:

```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

model_chr_nbr = '01' # chromosome used in parameter optimization
chromosomes = ['01','02','03','04','05','06','07','08','09','10','11',

df_pred = pd.DataFrame()
df_eval = pd.DataFrame(columns=['chromosome','R2_idt','R2_pred','pears

for chr_nbr in chromosomes:
    ref_path = 'input_data/IR64/Osat_IR64_chr{0}.fasta'.format(chr_nbr
    qry_path = 'input_data/Azucena/Osat_Azucena_chr{0}.fasta'.format(c
    coords_path = 'input_data/coords/IR64_Azucena_chr{0}.coords'.forma
    variants_path = 'input_data/snps/IR64_Azucena_chr{0}.snps'.format(
    rec_path = 'input_data/recombination/experimental_recombination_ch

    chr_tmp = rrp.rice_recombination_predictor(size_w, ref_path, qry_p
                                               coords_path, variants_
                                               CentO_path, rec_path,
                                               params=chr_mod.params)

    print('---Chromosome {0}---'.format(chr_nbr))
    chr_tmp.data_preprocessing()
    chr_tmp.predict_recombination()

    df_pred1 = pd.DataFrame({'chr_'+chr_nbr : chr_tmp.prediction})
    df_pred = pd.concat([df_pred,df_pred1], axis=1)

    # Plot landscape and correlation
    fig, ax = plt.subplots(1,2,figsize=(20,5),gridspec_kw={'width_rati
    chr_tmp.plot_landscape(ax=ax[0])
    chr_tmp.plot_correlation(ax=ax[1])
    fig.suptitle('Prediction of chromosome {0} recombination, calibrat
                 fontsize=15)

    # model evaluation
    corr_mod, r2_mod = chr_tmp.prediction_evaluation('model')
    corr_idt, r2_idt = chr_tmp.prediction_evaluation('identity')
    df_eval = df_eval.append({'chromosome':chr_nbr,
                              'R2_idt':r2_idt,
```

```
                                    'R2_pred':r2_mod,
                                    'pearson_idt':corr_idt,
                                    'pearson_pred':corr_mod},
                                   ignore_index=True)

df_pred.to_csv(results_path+'predictions.csv',header=True,index=True)
```

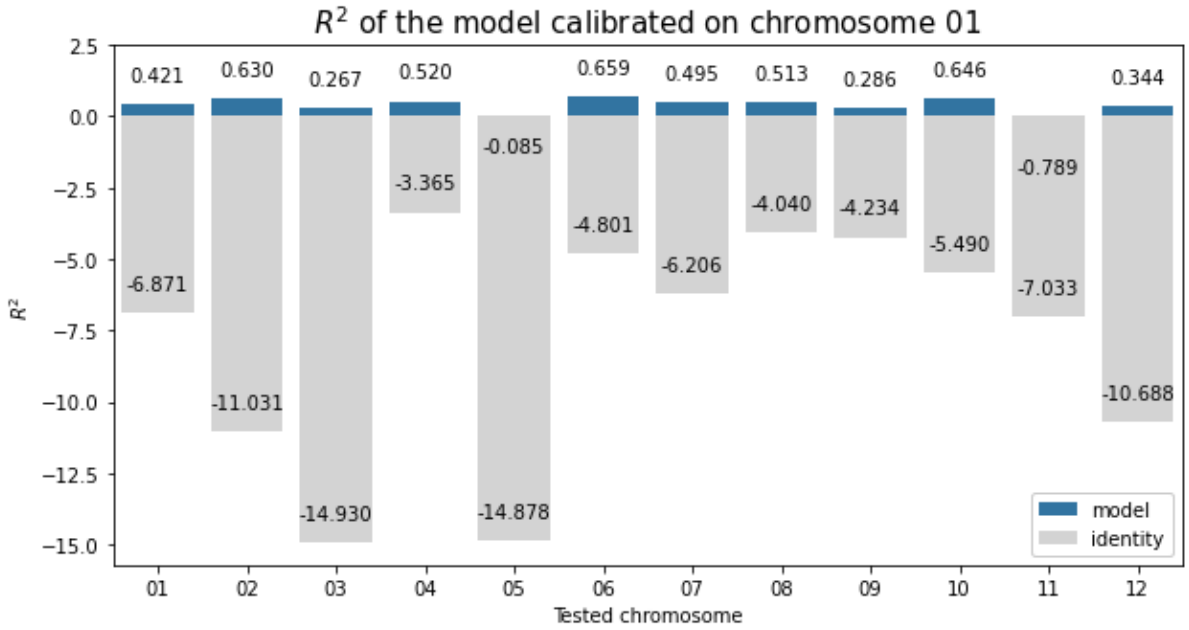
The above will generate a .csv file where each column corresponds to the prediction of recombination on a chromosome. At the same time, for each chromosome, a plot will be generated as follows:



With the performance data stored in the *df_eval* dataframe, the plots shown below can be generated. These plots compare the performance of the two approaches to predict recombination: the identity and the model (modified identity).

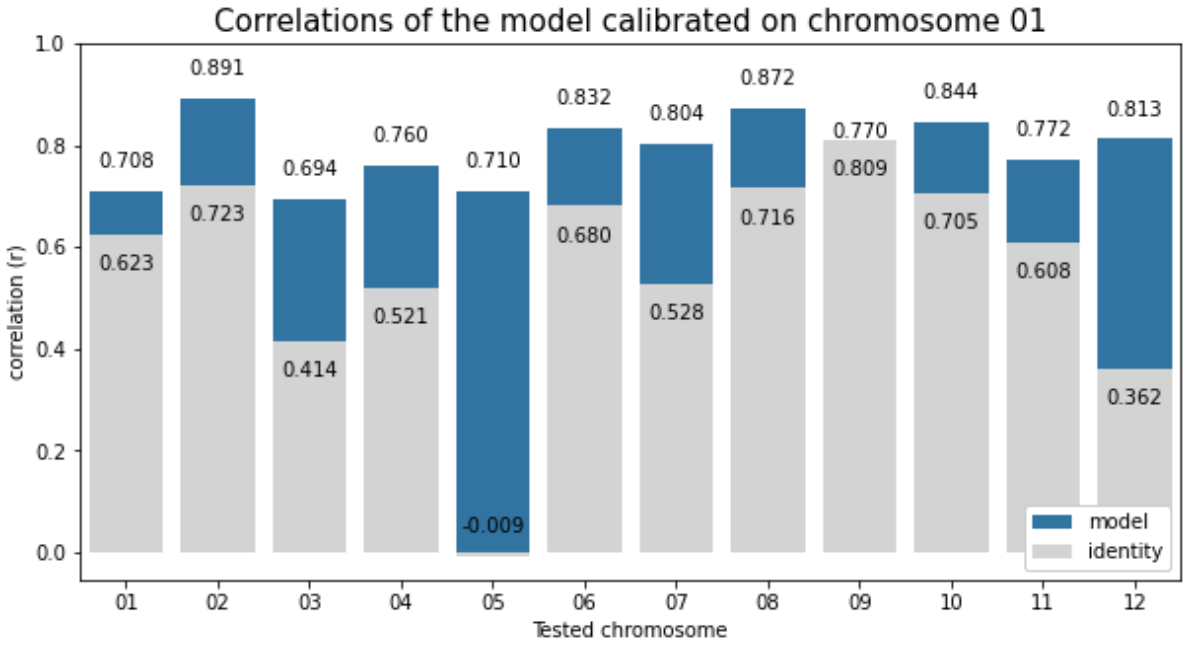
Plot for coefficient of determination R2:

```
plt.figure(figsize=(10,5))
ax0 = sns.barplot(data=df_eval,x='chromosome',y='R2_pred',color='tab:b
ax0.bar_label(ax0.containers[0],label_type='edge',padding=10,fmt='%.3f
ax1 = sns.barplot(data=df_eval,x='chromosome',y='R2_idt',color='lightg
ax1.bar_label(ax1.containers[1],label_type='edge',padding=-20,fmt='%.3
ax1.set_xlabel('Tested chromosome')
ax1.set_ylabel('$R^2$')
ax1.set_title('$R^2$ of the model calibrated on chromosome {0}'.format
              fontsize=15)
ax1.legend(loc='lower right',framealpha=1)
plt.ylim(None,2.5)
```



Plot for pearson correlation coefficient r:

```
plt.figure(figsize=(10,5))
ax0 = sns.barplot(data=df_eval,x='chromosome',y='pearson_pred',color='
ax0.bar_label(ax0.containers[0],label_type='edge',padding=10,fmt='%.3f
ax1 = sns.barplot(data=df_eval,x='chromosome',y='pearson_idt',color='l
ax1.bar_label(ax1.containers[1],label_type='edge',padding=-20,fmt='%.3
ax1.set_xlabel('Tested chromosome')
ax1.set_ylabel('correlation (r)')
ax1.set_title('Correlations of the model calibrated on chromosome {0}'
              fontsize=15)
ax1.legend(loc='lower right',framealpha=1)
plt.ylim(None,1)
```



If there is **no experimental data for any chromosome** you can use your own combination of parameters to make the prediction:

```
parameters = [0.5,0.97,1,0.9,1,0,0.002]
chr_test = rrp.rice_recombination_predictor(size_w, ref_path, qry_path
                                            coords_path, variants_path,
                                            CentO_path, params=parameter
chr_test.data_preprocessing()
chr_test.prediction
chr_test.plot_landscape()
```

The complete example, for the case where experimental recombination data is available for all chromosomes, is in the file **test.py**

Rice recombination predictor package

Rice recombination predictor module

Created on Mon Dec 27 09:42:21 2021

@author: Camila Riccio and Mauricio Peñuela

class rice_recombination_predictor.**rice_recombination_predictor**(*size_w*, *ref_path*, *qry_path*, *coords_path*, *variants_path*, *CentO_path*, *rec_path=None*, *coords=None*, *variants=None*, *experimental_rec=None*, *wbp_r=None*, *wbp_q=None*, *features=None*, *params=None*, *size_wc=50*, *alpha=0.1*, *cases=None*, *prediction=None*)

Bases: **object**

Class for predicting chromosomal recombination in rice (Oryza sativa) from the alignment of the two parental sequences.

Parameters:

- **size_w** (*int*) – Observation window size (base pairs)
- **ref_path** (*str*) – fasta file path with a chromosome sequence of the reference genome.
- **qry_path** (*str*) – fasta file path with a chromosome sequence of the qry genome. The sequence must be of the same chromosome number as ref_path.
- **coords_path** (*str*) – file path with extension .coords that is obtained from MUMmer alignment, containing the contig coordinates.
- **variants_path** (*str*) – file path with extension .snps that is obtained from MUMmer alignment, containing the contig coordinates.
- **CentO_path** (*str*) – fasta file containing the CentO sequence.
- **rec_path** (*str*) – file path with experimental recombination values.
- **coords** (*DataFrame.*) – contig coordinates of the alignment between the reference genome and the query genome.
- **variants** (*DataFrame.*) – variants detected in the alignment between the reference genome and the query genome.

- **experimental_rec** (*list*) – Experimental recombination values per window.
- **wbp_r** (*list of ints*) – reference chromosome bins/windows in base pairs.
- **wbp_q** (*list of ints*) – query chromosome bins/windows in base pairs.
- **features** (*DataFrame*) – relative frequency of the following features by window: absent bases, bases in inversions, variant bases, identical bases.
- **params** (*list of floats*) – recombination prediction model parameters [p1,p2,p3,p4,p5,p6,p7].
- **size_wc** (*int*) – number of windows used to the left and right of the reference and query centromeres to make the linear transition from zero to one of the weight function that corrects for recombination in the centromeric region.
- **alpha** (*float*) – smoothing factor, 0 < alpha <=1. The lower the smoother.
- **cases** (*list of ints*) – listing of the case number (0,1,2 or 3) that was applied in each window for the prediction of recombination, being 0 the non-application of cases.
- **prediction** (*list of floats*) – predicted recombination values per window.

`data_preprocessing()`

Perform pre-processing of the following files: coordinates with extension .coords, variants with extension .snps, sequences (reference and query) with extension .fasta, and experimental recombination with extension .csv.

`optimize_model_parameters(x0, metric='r2')`

Optimization of the 7 parameters of step 1 of the model (p1,p2,p3,p4,p5,p6,p7).

Parameters:
- **x0** (*list of floats*) – Initial values of the parameters to start the search for the optimal ones.
- **metric** (*str*) – Evaluation metric of the model prediction for the optimization process. Choosing between 'pearson' and 'r2', Defoult to 'r2'.

`plot_correlation(ax=None, fontsize_legend=10, fontsize_axtitle=10)`

Plots linear relationship between the experimental recombination and the prediction of the model. The marker color in the scatter plot indicate which case from the first step of the model is applied in each window.

`plot_landscape(ax=None, fontsize_legend=10, fontsize_axtitle=10)`

Landscape of identity, model prediction and experimental recombination (if available). The colored bars at the bottom of the landscapes indicate which case from the first step of the model is applied in each window.

`predict_recombination(p=None)`

Perfom the prediction of the crossover recombination through a 4-step model, using information from the alignment between the reference and query genomes. Calculates the windowed features and then uses them to apply the model and make the prediction.

Parameters: **p** (*list*) – parameters of step 1 of the model [p1,p2,p3,p4,p5,p6,p7]. A value of p1 is subtracted from windows with identity less than p2 and variants greater than p7. A value of p3 is added to windows with identity less than p4

and variants less than p7. A value of p5 is subtracted from windows with absent bases less than p6 and variants less than p7.

prediction_evaluation(*approach='model'*)

Evaluates recombination prediction with Pearson's correlation coefficient and R2. It can compare the experimental recombination with the identity or with the final prediction of the model.

Parameters: **approach** – Recombination prediction approach to compare against experimental recombination. Choosing between 'model' and 'identity'. Default set to 'model'.

Returns: pearson's correlation coefficient, and R2.

Return type: float, float

CentOFinder module

Created on Fri Mar 18 12:54:34 2022

@author: camila Riccio and Mauricio Peñuela

class CentOFinder.**CentOFinder**(*CentO_path*, *chromosome_path*, *size_w*, *wbp=None*, *total_windows=None*, *chromosome_length=None*, *CentO_freq=None*, *c_window_number=None*, *c_window_interval=None*)

Bases: **object**

A class used to detect the location of the centromere on rice chromosomes, based on the frequency of CentO sequences.

Parameters:

- **CentO_path** (*str*) – CentO fasta file path.
- **chromosome_path** (*str*) – chromosome fasta file path.
- **size_w** (*int*) – Observation window size (base pairs)
- **wbp** (*list of ints*) – chromosome bins/windows in base pairs.
- **total_windows** (*int*) – Total windows number
- **chromosome_length** (*int*) – total number of base pairs on the chromosome
- **CentO_freq** (*list of ints.*) – CentO base pair frequency per chromosome.
- **c_window_number** – chromosome window number with the highest frequency of CentO alignments.
- **c_window_interval** (*tuple of ints.*) – chromosome window interval, in base pairs, with the highest frequency of CentO alignments.

detect_centromere(*verbose=True*)

Computes the frequency of CentO alignments per window, the window number with the highest frequency, and the corresponding 2Mbp and 3Mbp centromeric region prediction.

Parameters: **verbose** (*bool*) – If True print information about the results, i.e., chromosome length, window size, Total chromosome windows, window with highest CentO frequency, window midpoint and centromeric region prediction of 2Mbp and 3Mbp. Default to True.

plot_CentO_frequency(*color='dodgerblue', label='CentO frequency'*)

Plot the frequency of base pairs belonging to a CentO alignment for each window of the chromosome. Also displays the predicted centromeric region of 2Mbp and 3Mbp.

Parameters:
- **color** (*string*) – color of the line that represents the frequency of alignments
- **label** (*string*) – label for the frequency line of the alignments

Indices and tables

- Index
- Module Index
- Search Page