proto_err Documentation

Release 1

Phelim Bradley

December 09, 2013

CONTENTS

CHAPTER

ONE

PROTO_ERR – MAIN PACKAGE

1.1 proto_err.simulation – Simulating Error Reads

class proto_err.simulation.**complexError**(*record*, *opt*, *id*)
Bases: proto_err.simulation.simulateError

Methods

error()
Function to induce and error

class proto_err.simulation.**simulateError**(*record*, *opt*, *id*)
Class of objects to simulate errors in a SeqRecord

Methods

deletion(*pos*, *dlen*)
function to induce a deletion

indel(*pos*)
function to induce an INDEL

ins(*pos*, *rl*)
function to induce a insertion

qscore None

record None
Return a Bio.SeqRecord

snp(*pos*, *rl*)
function to induce a SNP

class proto_err.simulation.**singleSNP**(*record*, *opt*, *id*)
Bases: proto_err.simulation.simulateError

Methods

error()
Function to induce and error

proto_err.simulation.**subsample**(*ref*, *opt*, *errorSimulator=<class proto_err.simulation.complexError at 0x10544c1f0>*)
Function to take a fasta file subsample reads and generate a list of subsampled reads

1.2 proto_err.errorCount – Error Counting

class proto_err.errorCount.**counter**(*ref*, *errorList=None*, *samfile=None*, *opt=None*)
Takes a list of errors and does some kmer counting

Methods

countAlignedBases(*read*)

countErrorKmer(*maxKmerLength=None*)
Count all kmers in the list of errors of length kmerLen or below before and after and error.

Parameters **maxKmerLength** : int

This is a type. Maximum kmer length

Returns **emmissionDic,beforeAfterCount** :

dict,dict emmissionDic counts the number of times a transition occurs

e.g. {'A':{'T':123,'C':123,...,},'T':{'A':123,...},...}

emmissionDic[trueBase][emmitedBase] = count of trueBase -> emmitedBase transition

beforeAfterCount counts the number of times a kmer appears before or after an error

beforeAfterCount['before'][trueBase][emmitedBase][kmer] = count of kmer occurance before trueBase -> emmitedBase transition

beforeAfterCount['after'][trueBase][emmitedBase][kmer] = count of kmer occurance after trueBase -> emmitedBase transition

These dictonaries are also stored in the counter object in counter.res['errorMode'],counter.res['kmerCounts']

See also:

countRefKmer

Examples

```
>>> from errorCount import counter
>>> errorCounter = counter(ref,samfile)
>>> errorCounter.countErrorKmer(maxKmerLength = 3)
>>> ## counts all possible kmers of length 1,2,3 before and after an error.
>>> ({'A': {'C': 113, 'T': 105, 'G': 84}, ...}, {'after':{'A': {'C': {'CTT': 2, 'GCA': 1,...
```

countRefKmer(*maxKmerLength=None*)
Count all kmers in the reference of length kmerLen or below

i.e. counter.countRefKmer(maxKmerLength = 3) counts all possible kmers of length 1,2,3

returns a dictonary of counts in the form

{'AAT':123,'AA':123,...}

This dictonary is also stored in the counter object in counter.res['RefCounts']

getCount (*truth=None*, *emission=None*, *kmer=''*, *after=False*)
Gets the count for a given {truth,emmision,kmer}

kmerFreq (*seq*, *kmer*)
Calculate the number of times a kmer appears in a sequence seq : SeqRecord Object kmer : kmer string

readDiff (*read*, *ref*)

setup (*opt*)
Setup output

class proto_err.errorCount.**error** (*true*, *emission*, *read*, *readPos*)
Information about the errors in a read

Methods

after (*j*)
Return the following j bases,return N when bases missing e.g. error.after(2) returns the 2 bases after the error

before (*j*)
Return the preceding j bases,return N when bases missing e.g. error.before(2) returns the 2 bases before the error ''

errorType None
The type of read error SNP, Insertion or Deletion

isDeletion None
Is the error a deletion

isIndel None
Is the error an INDEL

isInsertion None
Is the error an insertion

isSnp None
Is the error a SNP

qscore (*i*)
Return the quality score at a base +i i from the error start position error.qscore(0) is equivalent to error.qual

qual None
Quality score of the base where the error occured. equivalent to error.qscore(0)

class proto_err.errorCount.**errorReader** (*samfile*, *ref*)
Iterable over errors in aligned reads

Methods

checkDeletion (*N*)
Checks Deletionread segment for errors. called when cigarstring = D:N

checkHardClipped (*N*)
Checks HardClipped read segment for errors. called when cigarstring = H:N

checkInsertion (*N*)
Checks Insertion read segment for errors. called when cigarstring = I:N

checkPadding(*N*)
Checks Padding read segment for errors. called when cigarstring = P:N

checkRead()
Checks current read for errors

checkSNPs(*N*)
Checks read segment for SNP errors. called when cigarstring = M:N

checkSeqMismatch(*N*)
Checks SeqMismatch read segment for errors. called when cigarstring = =:N

checkSkipped(*N*)
Checks Skipped read segment for errors. called when cigarstring = X:N

checkSoftClipped(*N*)
Checks SoftClipped read segment for errors. called when cigarstring = S:N

next()
Returns the next error in the samfile aligned read

readNext()
Iterates to the next read in samfile

refRead **None**
Get the reference sequence the read is aligned to

CHAPTER

TWO

INDICES AND TABLES

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

p