HybridCheck in the console

Ben J. Ward

2015-12-01

Contents

This manual is provided as a vignette to the HybridCheck package for R. It can be acessed from an R session with with following:

```
vignette("Scripting_with_HybridCheck")
```

Installing HybridCheck

To install HybridCheck from the GitHub repository executing the following code - snippet into an R session should get HybridCheck installed with minimum difficulty:

```
install.packages("devtools", dep = T)
library(devtools)
install_github("Ward9250/HybridCheck", build_vignettes = TRUE)
```

This installs a package by R community contributor Hadley Wickham which allows you to install R packages from their GitHub Repositories with minimum fuss.

Now you have the latest version of HybridCheck installed in your R library. By default the above command installs the master branch of the HybridCheck repository. This is the "safest" option by which we mean it is the version we are by far most confident in to contain no errors or bugs. Anything new that is written for HybridCheck first is written in it's own branch, before being pulled to the devel branch. When we are then sure this is stable it is merged into the master branch.

Starting HybridCheck and Loading Data

To get started fire up the R console and enter:

```
library(HybridCheck)
```

To get started using HybridCheck to analyse a set of sequences, you first have to create a new HybridCheck object.

The HybridCheck object will contain all data, options, and will carry out the analysis. So everything you do in HybridCheck is done by interacting with this object.

By providing a file-path to a FASTA format sequence file when the new object is created the DNA data will also be loaded and prepared automatically. Base positions of the alignment which are non-informative, or positions with 'N's will be excluded.

```
MyAnalysis <- HC$new("~/Dropbox/MySequences.fas")
```

With the above line a new HybridCheck object would be created and assigned to a variable "MyAnalysis".

The HybridCheck package has been loaded, and data has been read in and is now contained in a HybridCheck object, which also contains all analysis settings and steps.

You can also create a new HybridCheck object from sequences you already have loaded into R as a `DNAbin` object (the DNA type used in the `ape` package). The HybridCheck package comes with an example datasets that is used in this user manual. It can be accessed with the `data` command after loading the HybridCheck package.

```
data(MySequences)
MyAnalysis <- HC$new(MySequences)
```

```
## Reading in sequence file...
## Class of input is DNAbin from the ape package...
## Looking for duplicates...
## Checking length of sequences...
## Subsetting the informative segregating sites...
## Finished DNA input...
## Deciding triplets based on results of Four Taxon Tests.
## Using tests that are globally significant.

## Warning in fttDescision(ftt, "SIGNIFICANT", TripletCombinations, dna): No
## four taxon tests were found to be significant, either none were significant
## or no test has been performed.
```

The warning that is produced by this command may be ignored. It only means that one of the analysis steps has not been performed yet.

Viewing and editing analysis settings

When scripting with a HybridCheck object, you execute several methods that make the HybridCheck object perform several steps of the analysis. Each analysis step is performed according to its settings. Before we walk through executing the analysis steps in detail, it's important to know how to view and alter the settings of the steps.

Viewing HybridCheck current settings

To view the settings and state of your current HybridCheck object, you just need to enter the name of the variable you used to assign the HybridCheck object, in the case of this example, the HybridCheck object created was assigned to the name `MyAnalysis`.

```
MyAnalysis
```

```
## HC object:
##
## DNA Sequence Information:
## ------------------------
## An alignment of 10 sequences.
##
## Full length of alignment: 400000
## Excluding non-informative sites: 33043
##
## Sequence names:
## 1: Seq1
## 2: Seq2
## 3: Seq3
## 4: Seq4
## 5: Seq5
## 6: Seq6
## 7: Seq7
## 8: Seq8
## 9: Seq9
## 10: Seq10
##
## Populations:
## unnamed_1: Seq1
## unnamed_2: Seq2
## unnamed_3: Seq3
## unnamed_4: Seq4
## unnamed_5: Seq5
## unnamed_6: Seq6
## unnamed_7: Seq7
## unnamed_8: Seq8
## unnamed_9: Seq9
## unnamed_10: Seq10
##
##
##
## Settings for Sequence Scan Combinations:
## ---------------------------------------
## Triplet Generation Method (Method): 1
##
## Distance Threshold for excluding triplets with
##  too similar sequences (DistanceThreshold): 0.01
##
## How many sequences from the same partition are
##  allowed in a triplet (PartitionStrictness): 2
##
## A total of 120 triplets will be compared.
##
##
## Settings for sliding window scan of recombination signal:
## --------------------------------------------------------
## Size of the sliding window in base pairs (WindowSize): 100
##
## Number of base pairs to move the sliding window on
##  each iteration of the scan (StepSize): 1
```

```
##
##
## Settings for detecting blocks from recombination signal:
## ---------------------------------------------------------
## Manual sequence similarity thresholds (ManualThresholds): 90
##
## Automatically decide on thresholds (AutoThresholds): TRUE
##
## Fall back to manual thresholds (ManualFallback): TRUE
##
## Standard deviation divisor (SDStringency): 2
##
##
## Settings for testing and dating recombination blocks:
## -----------------------------------------------------
## Assumed substitution rate for dating (MutationRate): 1e-08
##
## Critical alpha value for significance testing (PValue): 0.005
##
## Apply bonferroni correction to critical alpha (BonfCorrection): TRUE
##
## Keep and date blocks that fail the alpha (DateAnyway): FALSE
##
## Assumed mutation model for dating (MutationCorrection): JC69
##
## Plotting Settings are not shown because of the number of settings.
## use showParameters('Plotting') to view them.
```

A text summary printed like that above will be printed which shows the settings for each analysis step.

Alternatively, to view the settings of only a single step, the `showParameters` method can be used:

```
MyAnalysis$showParameters("TripletGeneration")
```

```
## Settings for Sequence Scan Combinations:
## ----------------------------------------
## Triplet Generation Method (Method): 1
##
## Distance Threshold for excluding triplets with
##  too similar sequences (DistanceThreshold): 0.01
##
## How many sequences from the same partition are
##  allowed in a triplet (PartitionStrictness): 2
##
## A total of 120 triplets will be compared.
```

This command outputs a description of the settings as well as the keyword HybridCheck uses to identify them in brackets.

Changing the analysis settings

The settings of each analysis step are altered in a consistent way, using the same method of the HybridCheck object. To change the settings for any given step of the HybridCheck analysis, you use the `setParameters` method. This method accepts first a word that dictates which analysis step it is you want to change:

1. `TripletGeneration` - Settings for Sequence Scan Combinations.
2. `SSAnalysis` - Settings for the sequence similarity scan step.
3. `BlockDetection` - Settings for detecting recombinant blocks from sequence similarity scan data generated from the sequence similarity scan step.
4. `BlockDating` - Settings for calculating the significance values and divergence time estimates, for recombinant blocks detected in the `BlockDetection` step.
5. `Plotting` - Settings for generating figures of the recombination signal in one or more triplets.

In addition the method is then provided a series of name, value pairs where a name is one of the settings, and the value is the value to change it to.

To use an example, we know from the text summary printed for the HybridCheck object above, that the `TripletGeneration` step has three settings, identified by the three keywords: `Method`, `DistanceThreshold`, and `PartitionStrictness`. Let's say we wanted to increace the `DistanceThreshold` parameter from 0.01 to 0.1. We can do that by calling the `setParameters` method:

```
MyAnalysis$setParameters("TripletGeneration", DistanceThreshold = 0.1)
```

```
## Deciding triplets based on results of Four Taxon Tests.
## Using tests that are globally significant.

## Warning in fttDescision(ftt, "SIGNIFICANT", TripletCombinations, dna): No
## four taxon tests were found to be significant, either none were significant
## or no test has been performed.
```

Don't worry about what these settings do right now, they are described more fully in the next manual sections - although the nice text summary gives you a clue e.g. `DistanceThreshold` is a *Distance Threshold for eliminating triplets with too similar sequences.*

You can also edit multiple settings for the same step in one command, for example:

```
MyAnalysis$setParameters("TripletGeneration", DistanceThreshold = 0.1, PartitionStrictness = 1)
```

Summary

Each step of a HybridCheck analysis has settings associated with them, but the settings of each step are all easily changed by calling the `setParameters` method of the HybridCheck object.

In the next sections, each step of the HybridCheck analysis, and how to execute them, will be described.

Loading DNA data, and changing and setting populations.

Load in a sequence into a HybridCheck object:

```
data(MySequences)
MyAnalysis <- HC$new(MySequences)
```

```
## Reading in sequence file...
## Class of input is DNAbin from the ape package...
## Looking for duplicates...
## Checking length of sequences...
## Subsetting the informative segregating sites...
## Finished DNA input...
## Deciding triplets based on results of Four Taxon Tests.
## Using tests that are globally significant.

## Warning in fttDescision(ftt, "SIGNIFICANT", TripletCombinations, dna): No
## four taxon tests were found to be significant, either none were significant
## or no test has been performed.
```

When you print the summary of the HybridCheck object, you will see that, by default, the HybridCheck object has assumed each sequence, was sampled from one population. There are 10 populations listed, and they have been automatically assigned a name.

```
MyAnalysis
```

```
## HC object:
##
## DNA Sequence Information:
## -------------------------
## An alignment of 10 sequences.
##
## Full length of alignment: 400000
## Excluding non-informative sites: 33043
##
## Sequence names:
## 1: Seq1
## 2: Seq2
## 3: Seq3
## 4: Seq4
## 5: Seq5
## 6: Seq6
## 7: Seq7
## 8: Seq8
## 9: Seq9
## 10: Seq10
##
## Populations:
## unnamed_1: Seq1
## unnamed_2: Seq2
## unnamed_3: Seq3
## unnamed_4: Seq4
## unnamed_5: Seq5
## unnamed_6: Seq6
## unnamed_7: Seq7
## unnamed_8: Seq8
## unnamed_9: Seq9
## unnamed_10: Seq10
##
##
```

```
##
## Settings for Sequence Scan Combinations:
## ----------------------------------------
## Triplet Generation Method (Method): 1
##
## Distance Threshold for excluding triplets with
##  too similar sequences (DistanceThreshold): 0.01
##
## How many sequences from the same partition are
##  allowed in a triplet (PartitionStrictness): 2
##
## A total of 120 triplets will be compared.
##
##
## Settings for sliding window scan of recombination signal:
## ---------------------------------------------------------
## Size of the sliding window in base pairs (WindowSize): 100
##
## Number of base pairs to move the sliding window on
##  each iteration of the scan (StepSize): 1
##
##
## Settings for detecting blocks from recombination signal:
## --------------------------------------------------------
## Manual sequence similarity thresholds (ManualThresholds): 90
##
## Automatically decide on thresholds (AutoThresholds): TRUE
##
## Fall back to manual thresholds (ManualFallback): TRUE
##
## Standard deviation divisor (SDStringency): 2
##
##
## Settings for testing and dating recombination blocks:
## -----------------------------------------------------
## Assumed substitution rate for dating (MutationRate): 1e-08
##
## Critical alpha value for significance testing (PValue): 0.005
##
## Apply bonferroni correction to critical alpha (BonfCorrection): TRUE
##
## Keep and date blocks that fail the alpha (DateAnyway): FALSE
##
## Assumed mutation model for dating (MutationCorrection): JC69
##
## Plotting Settings are not shown because of the number of settings.
## use showParameters('Plotting') to view them.
```

If each sequence is from a seperate population, then this is fine, but if not, then the population settings will need to be changed. This is possible by creating a named list that specifies the populations. Then provide that list to the `setPopulations` method of the HybridCheck object:

```
populations <- list(
  Norfolk = c("Seq1", "Seq2", "Seq3"),
  Lincolnshire = c("Seq4", "Seq5", "Seq6"),
  Cambridgeshire = c("Seq7", "Seq8"),
  Suffolk = c("Seq9", "Seq10")
)

MyAnalysis$setPopulations(populations)
```

Check that the populations have indeed been set:

```
MyAnalysis
```

```
## HC object:
##
## DNA Sequence Information:
## -------------------------
## An alignment of 10 sequences.
##
## Full length of alignment: 400000
## Excluding non-informative sites: 33043
##
## Sequence names:
## 1: Seq1
## 2: Seq2
## 3: Seq3
## 4: Seq4
## 5: Seq5
## 6: Seq6
## 7: Seq7
## 8: Seq8
## 9: Seq9
## 10: Seq10
##
## Populations:
## Norfolk: Seq1, Seq2, Seq3
## Lincolnshire: Seq4, Seq5, Seq6
## Cambridgeshire: Seq7, Seq8
## Suffolk: Seq9, Seq10
##
##
##
## Settings for Sequence Scan Combinations:
## ----------------------------------------
## Triplet Generation Method (Method): 1
##
## Distance Threshold for excluding triplets with
##  too similar sequences (DistanceThreshold): 0.01
##
## How many sequences from the same partition are
##  allowed in a triplet (PartitionStrictness): 2
##
## A total of 120 triplets will be compared.
```

```
##
##
## Settings for sliding window scan of recombination signal:
## ----------------------------------------------------------
## Size of the sliding window in base pairs (WindowSize): 100
##
## Number of base pairs to move the sliding window on
##  each iteration of the scan (StepSize): 1
##
##
## Settings for detecting blocks from recombination signal:
## ----------------------------------------------------------
## Manual sequence similarity thresholds (ManualThresholds): 90
##
## Automatically decide on thresholds (AutoThresholds): TRUE
##
## Fall back to manual thresholds (ManualFallback): TRUE
##
## Standard deviation divisor (SDStringency): 2
##
##
## Settings for testing and dating recombination blocks:
## ------------------------------------------------------
## Assumed substitution rate for dating (MutationRate): 1e-08
##
## Critical alpha value for significance testing (PValue): 0.005
##
## Apply bonferroni correction to critical alpha (BonfCorrection): TRUE
##
## Keep and date blocks that fail the alpha (DateAnyway): FALSE
##
## Assumed mutation model for dating (MutationCorrection): JC69
##
## Plotting Settings are not shown because of the number of settings.
## use showParameters('Plotting') to view them.
```

Summary

HybridCheck assumes that each sequence loaded in is from a seperate population. If this is not the case, then you can tell HybridCheck which sequences belong to which population using a named list with the setPopulations method.

Four-Taxon Tests

Introduction

The Four-Taxon test is an optional step of the analysis. It requires four sequences from 4 seperate populations, at the very minimum.

Prepare the tests

To perform a Four taxon test of introgression and gene flow, first use the `prepareFourTaxonTests` method.

```
MyAnalysis$prepareFourTaxonTests()
```

```
## Initializing new FTtest data.
```

If no arguments is provided to the method, then HybridCheck will generate every possible combination of four populations possible for your data. For every combination of four populations, HybridCheck then does a simple distance analysis to decide which population is designated as P1, P2, P3, and A for the purposes of the four taxon test. The test assumes the phylogeny: (((P1, P2), P3), A);

Alternatively you can feed in a list of named vectors. Each named vector is a combination of four populations to test, designated as P1, P2, P3 and A for the purposes of the test.

```
popCombos <- list(
  c(P1 = "Norfolk", P2 = "Lincolnshire", P3 = "Cambridgeshire", A = "Suffolk")
)

MyAnalysis$prepareFourTaxonTests(popCombos)
```

Set the parameters and run tests

Once the tests have been prepared, they can be run with the `runFourTaxonTests` method. This method accepts three parameters:

1. `selections`
2. `numberOfBlocks`
3. `blockLength`

`selections` accepts a list of population combinations to run e.g. `list(c(P1 = "Norfolk", P2 = "Lincolnshire", P3 = "Cambridgeshire", A = "Suffolk"))` to just run that one specific test. Alternatively the keyword `"NOT.TESTED"` (default), tells HybridCheck to run the test for all tests not previously run. The keyword `"ALL"` Tells HybridCheck to run all tests.

The functions requires one of the two parameters `blockLength` or `numberOfBlocks` to be provided. It is used to calculate the number and size of Jack-knife blocks to use in the test:

```
MyAnalysis$runFourTaxonTests(blockLength = 1000L)
MyAnalysis$runFourTaxonTests(numberOfBlocks = 10L)
```

Get the results

To tabulate the results of the four taxon tests, use the tabulateFourTaxonTests method. The method accepts the following parameters:

1. `selections`
2. `neat`
3. `global`

`selections` can be `'ALL'`, `'TESTED'`, or a list of character vectors of length 4, each denoting a four taxon test by the names of the populations involved. If neat is set to `TRUE`, intermediate values used in calculation of the four taxon tests will be excluded. If global is set to `TRUE`. Then global statistics will be included in the tables. The method returns a `data.frame` which can be assigned as a variable and manipulated, explored, and saved, using usual R commands and/or packages.

```
fttResults <- MyAnalysis$tabulateFourTaxonTests(selections = "ALL", neat = TRUE, global = TRUE)
```

Summary

Four-taxon tests are a optional step that can be run if four or more sequences are available, each from a seperate population. Four-taxon tests are computed using the `prepareFourTaxonTests` and `runFourTaxonTests` methods.

Specifying sequence triplets

Introduction

Before performing the sliding window triplet scan, you must inform the HybridCheck object which sequence triplets you would like to scan.

Four methods of triplet generation.

The following methods of generating triplets are provided denoted by the integers 1, 2, 3, and 4. They are selected by changeing the `Method` parameter.

The settings are as follows:

1. Decide which triplets will be generated and scanned based on Four-Taxon Test results.
2. Decide which triplets will be generated and scanned by combinding sequences from seperate populations to form a triplet. The parameter `PartitionStrictness` determines how many sequences from each population is allowed in a triplet. With only 3 populations this should be `2L` or more. For more populations this may be set to `1L`.
3. Decide which triplets will be generated and analysed by considering the distances between sequences in the triplet, and then eliminating triplets with sequences too related using an automatic threshold. This is combinable with methods 1 and 2.
4. Decide which triplets will be generated and analysed by considering the distances between sequences in the triplet, and then eliminating triplets with sequences too related using a manually set threshold (set this using `DistanceThreshold`). This is combinable with methods 1 and 2:

```
# Method 1.
MyAnalysis$setParameters("TripletGeneration", Method = 1L)
```

```
## Deciding triplets based on results of Four Taxon Tests.
## Using tests that are globally significant.
```

```
# Method 2.
MyAnalysis$setParameters("TripletGeneration", Method = 2L)
```

```
## Generating triplets to find recombination in sequences, between partitions.
```

```
# Method 3.
MyAnalysis$setParameters("TripletGeneration", Method = 3L)

# Method 4.
MyAnalysis$setParameters("TripletGeneration", Method = 4L, DistanceThreshold = 0.1)

# Combine Distance methods with method 1 or 2:
MyAnalysis$setParameters("TripletGeneration", Method = c(2L, 4L), DistanceThreshold = 0.1)
```

```
## Generating triplets to find recombination in sequences, between partitions.
```

Summary

Four methods are available by which HybridCheck will generate sequence triplets to scan for recombination signal. It may do this based on the results of four taxon tests, based on population grouping, and/or based on the level of sequence similarity between sequences.

Scanning triplets for recombination signal

Introduction

This step will use a sliding window to measure pairwise sequence similarity across the informative sites of sequence triplets. It is started by using the `analyzeSS()` method of the HybridCheck object.

Settings

The parameters of the step are:

WindowSize The size in bases of the Sliding Window. The default size is 100.
StepSize The number of bases that the sliding window moves across sequences. The default size is 1.

Running the scans

The `analyzeSS` method also accepts an argument which tells it which triplets (of the set you have defined in the `TripletGeneration` step) are to be scanned with the current `SSAnalysis` settings.

This argument can be set as `"NOT.SCANNED"` (default) in which case, every triplet of the set defined by the `TripletGeneration` parameters will be scanned for recombination signal with the current `SSAnalysis` settings, if it has not previously already been scanned. This argument can also be set to `"ALL"`, in which case all the triplets defined by the `TripletGeneration` settings will be scanned for recombination signal.

If the argument is not one of these two arguments, it can be a list of vectors that each contain three sequence names.

For example just two triplets could be scanned:

```
MyAnalysis$setParameters("TripletGeneration", Method = c(2L, 4L), DistanceThreshold = 0.1)
```

```
## Generating triplets to find recombination in sequences, between partitions.
```

```
MyAnalysis$setParameters("SSAnalysis", WindowSize = 500L, StepSize = 1L)

tripletsToScan <- list(c("Seq1", "Seq4", "Seq7"), c("Seq1", "Seq4", "Seq8"))

MyAnalysis$analyzeSS(tripletsToScan)
```

```
## Initializing new triplets data.
## Scanning sequence similarity for triplet Seq1, Seq4, Seq7
## Checking the sliding window parameters...
## Making all the window frames...
## Scanning Now!
## Scanning sequence similarity for triplet Seq1, Seq4, Seq8
## Checking the sliding window parameters...
## Making all the window frames...
## Scanning Now!
## Finished Sequence Similarity Analysis.
```

It is important to note that you can't enter a triplet of names which either contains names which are not names of your sequences, or is not allowed by the `TripletGeneration` settings:

```
# This triplet is not allowed, because earlier the TripletGeneration
# settings were set so as only triplets which try to find recombination regions between
tripletsToScan <- list(c("Seq1", "Seq2", "Seq3"))
MyAnalysis$analyzeSS(tripletsToScan)
```

```
## Finished Sequence Similarity Analysis.
```

Convieniently changing scan settings

If you want to perform scans of various window and step sizes. You could do it like this:

```
tripletsToScan <- list(c("Seq1", "Seq4", "Seq7"), c("Seq1", "Seq4", "Seq8"))
MyAnalysis$setParameters("SSAnalysis", WindowSize = 500L, StepSize = 1L)
MyAnalysis$analyzeSS(tripletsToScan)
## Do stuff here with result...
MyAnalysis$setParameters("SSAnalysis", WindowSize = 1000L, StepSize = 1L)
MyAnalysis$analyzeSS(tripletsToScan)
## Do stuff here after scan...
MyAnalysis$setParameters("SSAnalysis", WindowSize = 100L, StepSize = 1L)
MyAnalysis$analyzeSS(tripletsToScan)
# Do stuff here after scan...
```

But alternatively the analyzeSS method allows you to merge these two repeated statements into one statement:

```
tripletsToScan <- list(c("Seq1", "Seq4", "Seq7"), c("Seq1", "Seq4", "Seq8"))
MyAnalysis$analyzeSS(tripletsToScan, replaceSettings = FALSE, WindowSize = 500L, StepSize = 1L)
# Do stuff after scan...
MyAnalysis$analyzeSS(tripletsToScan, replaceSettings = FALSE, WindowSize = 1000L, StepSize = 1L)
# Do stuff after scan...
MyAnalysis$analyzeSS(tripletsToScan, replaceSettings = FALSE, WindowSize = 100L, StepSize = 1L)
```

The option `replaceSettings` being set to `FALSE` means that you are basically passing in the settings for use with that scan run only, the settings stored in the HybridCheck object will not be overwritten. This is useful if you want to run most scans with the options set, but want to make sure the HybridCheck object will keep these settings for subsequent analyses as it would for the `setParameters` method, you must set `replaceSettings` to `TRUE` e.g.

```
MyAnalysis$analyzeSS(tripletsToScan1, replaceSettings = TRUE, WindowSize = 500L, StepSize = 1L)
# HybridCheck has remembered the settings and will apply them to all future analyses, providing they do
# Do stuff after scan...

# This time HybridCheck will do the scan with the settings provided here instead of the stored settings
MyAnalysis$analyzeSS(tripletsToScan2, replaceSettings = FALSE, WindowSize = 1000L, StepSize = 1L)
# Do stuff after scan...

# Now the analysis will be done with a window size of 500 and a step size of 1. Because it is using the
MyAnalysis$analyzeSS(tripletsToScan3)
```

Summary

HybridCheck can be instructed to scan sequence triplets of a given window and step size, and you can fine tune exactly which triplets are scanned. In addition, scanning triplets introduces a convienient way to change settings for the scan step, without having to use an extra `setParameters` command. You can choose whether HybridCheck should remember those settings for the subsequent scans you perform, or whether it should only apply them to the current scan and then forget them.

Detecting recombination blocks in sequence triplet scan data

Having scanned for recombination signal for the desired sequence triplets, the `BlockDetection` step is executed using the `findBlocks` method.

This step identifies regions of recombination in a given sequence triplet by examining the data from the sliding window scans in the previous `SSAnalysis` step. If there are any regions in which two sequences meet a certain sequence similarity threshold, they are treated as potentially recombinant.

The parameters for this step are:

ManualThresholds A vector of numbers between 0 and 100.
Each is a percenage sequence similarity threshold.
These thresholds are set by the user, and can be used to find recombination,
or used as a fallback if HybridCheck fails to automatically decide on some thresholds.

AutoThresholds A logical argument (`TRUE`, or `FALSE`), if `TRUE` (default) then during execution of the `findBlocks` method,
thresholds will be decided on automatically. If `FALSE` then the `ManualThresholds` will be used.

ManualFallback A logical argument (`TRUE`, or `FALSE`), if `TRUE` (default), then if automatic thresholds cannot be found, the
`ManualThresholds` will be used instead.

SDstringency A numeric value, the higher it is, the closer automatically detected thresholds are allowed to be to the mean
sequence similarity across the sequences. It is usually fine to leave this as 2.

Like the `analyzeSS` method in the previous section, the `findBlocks` must be given an argument that dictates which triplets of the set defined by the `TripletGeneration` parameters will have blocks found from their scan data. By default this is set to `"NOT.SEARCHED"` which will find blocks for triplets which have not already had blocks identified from their scan data. This argument can also be set to `"ALL"` or a list of vectors that each contain three sequence names.

The below example finds blocks in the two triplets that were scanned for recombination signal in the previous section:

```
tripletsToProcess <- list(c("Seq1", "Seq4", "Seq7"), c("Seq1", "Seq4", "Seq8"))
MyAnalysis$findBlocks(tripletsToProcess)
```

```
## Using the autodetect thresholds method...
## Deciding on suitable thresholds...
## Now beginning Block Search...
## Using the autodetect thresholds method...
## Deciding on suitable thresholds...
## Now beginning Block Search...
## Finished finding potential blocks for all triplet selections.
```

Note that, as with the `analyzeSS` method, you cannot specify a triplet which was not specified by the `TripletGeneration` settings:

```
tripletsToSearch <- list(c("Seq1", "Seq2", "Seq3"))
MyAnalysis$findBlocks(tripletsToSearch)
```

```
## Finished finding potential blocks for all triplet selections.
```

You also cannot search for blocks for triplets which have not been analyzed by `analyzeSS` yet:

```
tripletsToSearch <- list(c("Seq1", "Seq4", "Seq9"))
MyAnalysis$findBlocks(tripletsToSearch)
```

```
## Finished finding potential blocks for all triplet selections.
```

```
# A triplet must be scanned for recombination signal first
MyAnalysis$analyzeSS(tripletsToSearch)
```

```
## Finished Sequence Similarity Analysis.
```

```
MyAnalysis$findBlocks(tripletsToSearch)
```

```
## Finished finding potential blocks for all triplet selections.
```

Note that some columns are NA because they are calculated the next step, which is discussed in the next section.

The `tabulateDetectedBlocks` method, just like the `analyzeSS` method, allows you to alter settings conviniently without an extra command. The works exactly as it does for `analyzeSS`.

```
tripletsToProcess <- list(c("Seq1", "Seq4", "Seq7"), c("Seq1", "Seq4", "Seq8"))
MyAnalysis$findBlocks(tripletsToProcess, replaceSettings = FALSE, AutoThresholds = FALSE)
```

If you want to get a Data Frame containing a table of the detected blocks in some triplets, use the `tabulateDetectedBlocks` method.

This method needs to be given a set of sequence names to set the triplets that will be tabulated:

```
tripletsToTabulate <- list(c("Seq1", "Seq4", "Seq7"), c("Seq1", "Seq4", "Seq8"))
MyAnalysis$tabulateDetectedBlocks(tripletsToTabulate)
```

```
## Tabulating blocks for the triplet Seq1:Seq4:Seq7
## Tabulating blocks for the triplet Seq1:Seq4:Seq8

##           Triplet Sequence_Pair Sequence_Similarity_Threshold
## 1  Seq1:Seq4:Seq7     Seq1:Seq4                            65
## 2  Seq1:Seq4:Seq7     Seq1:Seq4                            54
## 3  Seq1:Seq4:Seq7     Seq1:Seq4                            54
## 4  Seq1:Seq4:Seq7     Seq1:Seq7                            82
## 5  Seq1:Seq4:Seq7     Seq1:Seq7                            77
## 6  Seq1:Seq4:Seq7     Seq1:Seq7                            77
## 7  Seq1:Seq4:Seq7     Seq1:Seq7                            55
## 8  Seq1:Seq4:Seq7     Seq1:Seq7                            55
## 9  Seq1:Seq4:Seq7     Seq4:Seq7                            64
## 10 Seq1:Seq4:Seq8     Seq1:Seq4                            69
## 11 Seq1:Seq4:Seq8     Seq1:Seq8                            82
## 12 Seq1:Seq4:Seq8     Seq1:Seq8                            77
## 13 Seq1:Seq4:Seq8     Seq1:Seq8                            77
## 14 Seq1:Seq4:Seq8     Seq1:Seq8                            55
## 15 Seq1:Seq4:Seq8     Seq1:Seq8                            55
## 16 Seq1:Seq4:Seq8     Seq4:Seq8                            56
##    First_BP_Position Last_BP_Position Approximate_Length_BP Number_of_SNPs
## 1             287543          296872                  9330            NA
## 2             287467          287535                    69            NA
## 3             297359          304343                  6985            NA
## 4             284139          287218                  3080            NA
## 5             284107          284132                    26            NA
## 6             287224          287252                    29            NA
## 7             283969          284097                   129            NA
## 8             287259          287397                   139            NA
## 9             265371          280260                 14890            NA
## 10            287568          295873                  8306            NA
## 11            284140          287220                  3081            NA
## 12            284108          284133                    26            NA
## 13            287227          287253                    27            NA
## 14            283969          284099                   131            NA
## 15            287260          287397                   138            NA
## 16            260557          280318                 19762            NA
```

```
##    Corrected_Number_of_SNPs p=0.05_Age p=0.5_Age p=0.95_Age P_Value
## 1                        NA         NA        NA          NA      NA
## 2                        NA         NA        NA          NA      NA
## 3                        NA         NA        NA          NA      NA
## 4                        NA         NA        NA          NA      NA
## 5                        NA         NA        NA          NA      NA
## 6                        NA         NA        NA          NA      NA
## 7                        NA         NA        NA          NA      NA
## 8                        NA         NA        NA          NA      NA
## 9                        NA         NA        NA          NA      NA
## 10                       NA         NA        NA          NA      NA
## 11                       NA         NA        NA          NA      NA
## 12                       NA         NA        NA          NA      NA
## 13                       NA         NA        NA          NA      NA
## 14                       NA         NA        NA          NA      NA
## 15                       NA         NA        NA          NA      NA
## 16                       NA         NA        NA          NA      NA
##    P_Thresh
## 1        NA
## 2        NA
## 3        NA
## 4        NA
## 5        NA
## 6        NA
## 7        NA
## 8        NA
## 9        NA
## 10       NA
## 11       NA
## 12       NA
## 13       NA
## 14       NA
## 15       NA
## 16       NA
```

Testing and dating detected blocks

Having searched for blocks in the desired sequence triplets, the `BlockDating` step is executed using the `dateBlocks` method.

This step assigns a significance value to blocks based on the size of a block, the number of mutations observed, and the binomial probability distribution. Afterwards, 95%CI regions for the divergence time of the blocks are calculated, assuming a mutational model and a strict molecular clock.

The parameters for this step are:

MutationRate The substitution rate assumed when estimating the ages of blocks. By default it is 10e-9.

PValue The critical alpha value when testing the significance values of blocks. By default is is 0.05.

BonfCorrection Logical (`TRUE` / `FALSE`) when `TRUE` (default) the `PValue` will be adjusted with a Bonferroni correction.

DateAnyway Logical (`TRUE` / `FALSE`) when `TRUE` (`FALSE` by default), all detected blocks will be kept and dated.
When `FALSE`, blocks that do not pass the critical alpha when testing for significance are not kept in the results
and are not dated.

MutationCorrection Set to one of "raw", "TS", "TV", "JC69", "K80", "F81", "K81", "F84", "BH87", "T92", "TN93", "GG95".
Base frequencies are calculated from the recombinant block if they are required by the model.

As was the case with the `analyzeSS` and `findBlocks` methods, `dateBlocks` needs a first argument that dictates which triplets of the set defined by the `TripletGeneration` parameters will have their blocks tested and dated. By default this is `"NOT.DATED"`, which means all triplets with blocks detected, that have not been tested or dated, will have their detected blocks tested and dated witht the current settings for the step. This can be set to `"ALL"`, or, as previously, can be set as a list of vectors that each contain three sequence names.

The following example finds blocks in the two triplets that were scanned for recombination signal and had blocks identified in the examples of the previous two sections:

```
tripletsToDate <- list(c("Seq1", "Seq4", "Seq7"), c("Seq1", "Seq4", "Seq8"))
MyAnalysis$dateBlocks(tripletsToDate)
```

```
## Now dating blocks
## Now dating blocks
```

You can't specify triplets which have not been scanned, or analysed to find blocks. You can't date blocks before you've found them!

The blocks can then be tabulated again, the columns of the table that were previously `NA` are now filled with values:

```
MyAnalysis$tabulateDetectedBlocks(tripletsToDate)
```

```
## Tabulating blocks for the triplet Seq1:Seq4:Seq7
## Tabulating blocks for the triplet Seq1:Seq4:Seq8

##          Triplet Sequence_Pair Sequence_Similarity_Threshold
## 1 Seq1:Seq4:Seq7     Seq1:Seq4                            65
## 2 Seq1:Seq4:Seq7     Seq1:Seq4                            54
## 3 Seq1:Seq4:Seq7     Seq1:Seq7                            82
## 4 Seq1:Seq4:Seq7     Seq4:Seq7                            64
## 5 Seq1:Seq4:Seq8     Seq1:Seq4                            69
## 6 Seq1:Seq4:Seq8     Seq1:Seq8                            82
## 7 Seq1:Seq4:Seq8     Seq1:Seq8                            55
## 8 Seq1:Seq4:Seq8     Seq4:Seq8                            56
##   First_BP_Position Last_BP_Position Approximate_Length_BP Number_of_SNPs
## 1            287543           296872                  9330            63
## 2            297359           304343                  6985            59
## 3            284139           287218                  3080            19
## 4            265371           280260                 14890            93
## 5            287568           295873                  8306            53
## 6            284140           287220                  3081            20
## 7            287260           287397                   138             0
## 8            260557           280318                 19762           125
```

```
##   Corrected_Number_of_SNPs p=0.05_Age p=0.5_Age p=0.95_Age       P_Value
## 1                       63     416735    340980     276643  8.516966e-69
## 2                       59     524638    426824     341253  3.869134e-44
## 3                       19     452329    317744     216378  1.666742e-35
## 4                       93     371094    314438     264190  1.441281e-87
## 5                       53     401836    322996     256408  1.174397e-63
## 6                       20     470490    333875     228285  6.529404e-35
## 7                        0    1072689    251330      18349  2.363298e-03
## 8                      126     368042    319854     275722 1.954088e-114
##   P_Thresh
## 1   0.0050
## 2   0.0025
## 3   0.0050
## 4   0.0050
## 5   0.0050
## 6   0.0050
## 7   0.0025
## 8   0.0050
```

Just like `analyzeSS` and `findBlocks`, the `dateBlocks`, settings for the analysis step can be set convieniently without the need for an additional call to `setParameters`. It works the same as for `analyzeSS` and `findBlocks`:

```
tripletsToDate <- list(c("Seq1", "Seq4", "Seq7"), c("Seq1", "Seq4", "Seq8"))
MyAnalysis$dateBlocks(tripletsToDate, replaceSettings = FALSE, DateAnyway = TRUE, MutationCorrection =
```

```
## Now dating blocks
## Now dating blocks
```

Summary

HybridCheck dates and calculates p-values for putative blocks detected by the `findBlocks` method, with the `dateBlocks` method. Like `analyzeSS` and `findBlocks`, it supports conviniently changing the settings for that step. Results can be tabulated with the `tabulateDetectedBlocks` method.

Plotting the recombination signal of a triplet

To generate a graphic of the recombination signal for a given set of triplets, the `plotTriplet` method is used. This method also takes a list of vectors, each containing 3 sequence names that specifies the triplets to plot. The method returns a list containing ggplot objects of each triplet chosen to be visualized.

The parameters for plotting are as follows:

What: This value can either be “Bars”, “Lines” or an R vector of both. This determines which plots HybridCheck will plot. HybridCheck produces two class of plot of a given triplet’s SS similarity data (often on one canvas), a lines plot, and a coloured heatmap-like plot.

PlotTitle: Boolean, set wether to include a title with the plot.

CombinedTitle: Boolean, set whether to only have one overall title for a Lines and Bars plot on the same canvas.

TitleSize: Integer value, sets the font size of the text in the plot title.

TitleFace: A character value "bold" by default, can be changed to a face compatible with ggplot2, see documentation of ggplot2 for more details.

TitleColour: A string denoting a colour "black" by default. Can be changed to any colour notation compatible with ggplot2, see the documentation or examples of ggplot2 for more details.

XLabels: Boolean, set whether to include the value labels of the x-axis in the plots.

YLabels: Boolean, set whether to include the value labels of the y-axis in the plots.

XTitle: Boolean, set whether to include the title of the x-axis in the plots.

XTitleFontSize: Integer value, sets the font size of the x-axis title in plots.

XTitleColour: A character string denoting a colour, "black" by default. Sets the colour of the x-axis title font. Can be changed to any colour notation compatible with ggplot2, see the documentation or examples of ggplot2 for more details.

XLabelSize: Integer value, sets the font size of the x-axis labels.

XLabelColour: A character string denoting a colour, "black" by default. Sets the colour of the x-axis value labels. Can be changed to any colour notation compatible with ggplot2, see the documentation or examples of ggplot2 for more details.

YTitle: Boolean, set whether to include the title of the y-axis in the plots.

YTitleFontSize: Integer value, sets the font size of the y-axis title in plots.

YTitleColour: A character string denoting a colour, "black" by default. Sets the colour of the y-axis title font. Can be changed to any colour notation compatible with ggplot2, see the documentation or examples of ggplot2 for more details.

YLabelSize: Integer value, sets the font size of the y-axis labels.

YLabelColour: A character string denoting a colour, "black" by default. Sets the colour of the x-axis value labels. Can be changed to any colour notation compatible with ggplot2, see the documentation or examples of ggplot2 for more details.

Legends: Boolean value, set's whether to include the legends of plots. TRUE by default.

LegendFontSize: An integer value, sets the legend font size.

MosaicScale: An integer value, 500 by default. This affects how the Bars plot is drawn. With a higher value, a greater resolution is achieved but more and finer sequence similarity data is required. This is achieved with sequences with high numbers of informative sites and by adjusting the WindowSize and StepSize settings for the SSAnalysis step.
HybridCheck will print a message to the console if problems arise during plot drawing due to the MosaicScale settings.

```
plotsInAList <- MyAnalysis$plotTriplets(c("Seq1", "Seq4", "Seq7"))
plotsInAList[[1]]
```

Summary

HybridCheck produces graphics of recombination signal scanned across sequences and supports a wide range of parameter settings, but this range is not 100% exhaustive, therefore HybridCheck returns graphics as `ggplot2` or `grid` objects, which may be manipulated programatically by the user to get the exact look that is desired.

Testing known blocks

It is possible to tell HybridCheck to assume a given region between your two sequences is recombinant, and then test them for significance and estimate their date.

This is done using the `addUserBlock` method, providing the two sequence names, the base position the block starts at, and the base position the block ends at.

User defined blocks can then be tabulated with the `tabulateUserBlocks` method, and they can be tested for significance and dated according to the `BlockDating` step settings, with the `dateUserBlocks` method.

```
MyAnalysis$addUserBlock(c("Seq1", "Seq4"), firstbp = 287463, lastbp = 295890)
MyAnalysis$tabulateUserBlocks()
```

```
##   Sequence_Pair FirstBP LastBP ApproxBpLength SNPs CorrectedSNPs P_Value
## 1     Seq1:Seq4  287463 295890           8428   NA            NA      NA
##   P_Threshold fiveAge fiftyAge ninetyFiveAge
## 1          NA      NA       NA            NA
```

```
MyAnalysis$dateUserBlocks()
MyAnalysis$tabulateUserBlocks()
```

```
##   Sequence_Pair FirstBP LastBP ApproxBpLength SNPs CorrectedSNPs
## 1     Seq1:Seq4  287463 295890           8428   55            55
##        P_Value P_Threshold fiveAge fiftyAge ninetyFiveAge
## 1 1.120252e-63       0.005  410144   330115        263346
```

To clear the user added blocks, use the `clearUserBlocks` method:

```
MyAnalysis$clearUserBlocks(c("Seq1", "Seq4"))
MyAnalysis$tabulateUserBlocks()
```

```
##  [1] Sequence_Pair  FirstBP        LastBP         ApproxBpLength
##  [5] SNPs           CorrectedSNPs  P_Value        P_Threshold
##  [9] fiveAge        fiftyAge       ninetyFiveAge
## <0 rows> (or 0-length row.names)
```

Summary

HybridCheck can be told that regions are putatively recombinant between two sequences, and will calculate a p value and estimated coalesence time for the region dictated.

Problems, Bugs, and assistance

If you have any trouble using the HybridCheck package, or find an error or bug you are unfamiliar with, don't hesitate to contact code author and repo maintainer Ben J. Ward (Ward9250) at ben.ward@tgac.ac.uk, or visit the GitHub Repository Ward9250/HybridCheck and file a bug report or message there.
