Introduction to GENESPACE

JTLovell

10/25/2021

1. Overview

GENESPACE is a comparative genomics framework implemented in the R environment for statistical computing. The premise is that, when analyzing high-quality genome assemblies and annotations, we can improve the confidence of evolutionary inference by combining two sources of evidence for common ancestry: synteny (i.e. collinearity of gene order) and coding sequence similarity (homology). GENESPACE produces pairwise genome comparisons via dotplots and annotated blast8 files as well as multi-genome comparisons via riparian plots and a pan-genome annotation. GENESPACE requires results from `orthofinder` and `MCScanX`. Both of these programs should be installed (see 2. Installation below).

2. Installation

2.1 Install required programs GENESPACE requires third-party software can be installed as follows: `orthofinder` (which includes `diamond`) is most simply installed via conda.

```
conda create -n orthofinder
conda activate orthofinder
conda install -c bioconda orthofinder
```

If conda is not available on your machine, you can install orthofinder from a number of other sources. See orthofinder documentation for details.

`MCScanX` should be installed from github. **NOTE** Due to nested dependencies in the `orthofinder` install, `orthofinder` must be in the path when called, while the file path to `MCScanX` does not need to be in the path and can be specified in R.

2.2 Install GENESPACE If you are planning to run orthofinder from within R, which is needed if using orthofinderInBlk (see below for details) and recommended when using any genomes with ploidy > 1, enter R from the terminal with either `R` (for command line interace), or if Rstudio is installed on your machine `open -na rstudio`.

Once in R, GENESPACE can be installed directly from github via:

```
if (!requireNamespace("devtools", quietly = TRUE))
    install.packages("devtools")
devtools::install_github("jtlovell/GENESPACE", upgrade = F)
```

2.3 Install R dependencies The above command will install the CRAN-sourced dependencies (`data.table`, `dbscan` and `R.utils`). The bioconductor dependencies (`Rtracklayer` and `Biostrings`) need to be installed separately via:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install(c("Biostrings", "Rtracklayer"))
```

3. GENESPACE workflow

3.1 Example data Here, we will use the example data provided with the GENESPACE installation: chromosomes 17-18 in the human, chimpanzee and rhesus macaque genomes. See `?make_exampleData` for more information on data sources.

First off, we need to specify a place to run everything. Here, we'll use a test directory on the desktop:

```
library(GENESPACE)
```

```
## Loading required package: data.table
```

```
runwd <- file.path("~/Desktop/testGenespace")
```

To illustrate all steps of the pipeline, lightly subset NCBI-formatted annotations of human/chimpanzee chrs 3-4 and rhesus chr 2 & 5 are provided in the extData of the GENESPACE R package. These data can be added to the above directory with the correct subdirectory structure for GENESPACE via:

```
make_exampleDataDir(writeDir = runwd)
```

```
## Done! Run GENESPACE with rawGenomeRepo = ~/Desktop/testGenespace/rawGenomes
```

```
list.files(runwd, recursive = T, full.names = F)
```

```
## [1] "rawGenomes/chimp/chimp/annotation/chimp_Clint_PTRv2_gene.gff.gz"
## [2] "rawGenomes/chimp/chimp/annotation/chimp_Clint_PTRv2_pep.fa.gz"
## [3] "rawGenomes/human/human/annotation/human_GRCh38.p13_gene.gff.gz"
## [4] "rawGenomes/human/human/annotation/human_GRCh38.p13_pep.fa.gz"
## [5] "rawGenomes/rhesus/rhesus/annotation/rhesus_Mmul_10_gene.gff.gz"
## [6] "rawGenomes/rhesus/rhesus/annotation/rhesus_Mmul_10_pep.fa.gz"
```

NOTE this creates a subdirectory called `/rawGenomes`. For downstream flexibility (e.g. multiple genome versions for one species, metadata or assembly data, etc), the raw genome directory structure follows: `/rawGenomes/$speciesID/$versionID/annotion`.

When working with your own data, place the raw annotation files in this same directory structure with separate directories for each species, separate subdirectories for each genome version, and the annotation files in a subdirectory called "annotation".

3.2 Initialize the GENESPACE run All elements of GENESPACE require a list of parameters, specified to functions as `gsParam`. This contains paths to files, working directories, program executables the basic parameterization of the run.

```
gpar <- init_genespace(
  genomeIDs = c("human","chimp","rhesus"),
  speciesIDs = c("human","chimp","rhesus"),
  versionIDs = c("human","chimp","rhesus"),
  ploidy = rep(1,3),
  diamondMode = "fast",
  orthofinderMethod = "fast",
  wd = runwd,
  nCores = 4,
  minPepLen = 50,
```

```
  gffString = "gff",
  pepString = "pep",
  path2orthofinder = "orthofinder",
  path2mcscanx = "~/MCScanX",
  rawGenomeDir = file.path(runwd, "rawGenomes"))
```

```
## set working directory to /Users/jlovell/Desktop/testGenespace
##
## found raw gff files:
##   /Users/jlovell/Desktop/testGenespace/rawGenomes/human/human/annotation/human_GRCh38.p13_gene.gff.g
##  /Users/jlovell/Desktop/testGenespace/rawGenomes/chimp/chimp/annotation/chimp_Clint_PTRv2_gene.gff.g
##  /Users/jlovell/Desktop/testGenespace/rawGenomes/rhesus/rhesus/annotation/rhesus_Mmul_10_gene.gff.gz
##
## found raw peptide files:
##   /Users/jlovell/Desktop/testGenespace/rawGenomes/human/human/annotation/human_GRCh38.p13_pep.fa.gz
##  /Users/jlovell/Desktop/testGenespace/rawGenomes/chimp/chimp/annotation/chimp_Clint_PTRv2_pep.fa.gz
##  /Users/jlovell/Desktop/testGenespace/rawGenomes/rhesus/rhesus/annotation/rhesus_Mmul_10_pep.fa.gz
##
##
## Can't find all parsed annotation files ... need to run parse_annotations, parse_ncbi or parse_phytoz
##
## GENESPACE run initialized:
##  Initial orthofinder database generation method: fast
##  Orthology graph method: global
```

3.3 Format the raw annotations for GENESPACE To improve read/write speed, GENESPACE uses a simplified gff3-like text file with a column `id` that exactly matches the peptide fasta header. GENESPACE has built in functions to parse NCBI (`parse_ncbi`) and phytozome (`parse_phytozome`); however, if using non-standard formatted annotations, this can be a tricky step. See 4.1 (using non-standard annotation files). While the example data was originally downloaded from NCBI, much of the NCBI formatting was stripped to make the files smaller. They now can be parsed with the generic `parse_annotations`:

```
parse_annotations(
  gsParam = gpar,
  gffEntryType = "gene",
  gffIdColumn = "locus",
  gffStripText = "locus=",
  headerEntryIndex = 1,
  headerSep = " ",
  headerStripText = "locus=")
```

```
## Parsing annotation files ...
##  human ...
##      Importing gff ... found 1787 gff entires, and 1787 gene entries
##      Importing fasta ... found 1787 fasta entires
##      1786 gff-peptide matches
##  Done!
##  chimp ...
##      Importing gff ... found 1927 gff entires, and 1927 gene entries
##      Importing fasta ... found 1927 fasta entires
##      1926 gff-peptide matches
##  Done!
##  rhesus ...
##      Importing gff ... found 1906 gff entires, and 1906 gene entries
```

```
##      Importing fasta ... found 1906 fasta entires
##      1904 gff-peptide matches
##  Done!
```

3.4 Run orthofinder GENESPACE requires orthofinder to be run. Here, since orthofinder is in the path, we can run it straight from R, using the 'fast' search method

```
gpar <- run_orthofinder(
  gsParam = gpar)
```

```
## Synteny Parameters have not been set! Setting to defaults
##  Running 'draft' a.k.a 'fast' genespace orthofinder method
##  ################################################################
##  ***NOTE***
##  This method should only be used for:
##      (1) closely related diploid species,
##      (2) visualization/genome QC purposes, or
##      (3) inferring orthogroups WITHIN syntenic regions
##  ################################################################
##      Running 1/6 (chimp vs. chimp)
##      Running 2/6 (chimp vs. rhesus)
##      Running 3/6 (rhesus vs. rhesus)
##      Running 4/6 (chimp vs. human)
##      Running 5/6 (rhesus vs. human)
##      Running 6/6 (human vs. human)
##      Done!
##  Inverting intergenomic files ... Done!
##  Running orthofinder -og on pre-computed blast:
```

3.4 Run the GENESPACE synteny search The main engine of GENESPACE is `synteny`. This is a complex function that parses pairwise blast hits into syntenic regions and blocks. See 4.1 (synteny specifications) for details. Here, we will just use the defaults for synteny:

```
gpar <- synteny(gsParam = gpar)
```

```
## Loading annotations ...
##  Indexing location of orthofinder results ... Done!
##  Reading the gffs ... Done!
##  Pulling gene lengths ... Done!
##  Parsing global orthogroups ... Done!
## Defining collinear orthogroup arrays ...
##  Using collinear orthogroups for array identity:
##  chimp: 102 genes in 37 collinear arrays
##  human: 102 genes in 33 collinear arrays
##  rhesus: 97 genes in 37 collinear arrays
##  Choosing array representative genes ... Done!
## Found 5616 genes, 1955 orthogroups and 107 arrays with 301 genes
## Pulling within-genome synteny ...
##  Genome: n raw hits / hits in (regions) / hits in (blks)
##  human   (selfhit): 3834 / 3218 (2) / 3218 (2)
##  chimp   (selfhit): 3834 / 3188 (2) / 3188 (2)
##  rhesus  (selfhit): 3700 / 3056 (2) / 3056 (2)
## Pulling intergenomic synteny ...
```

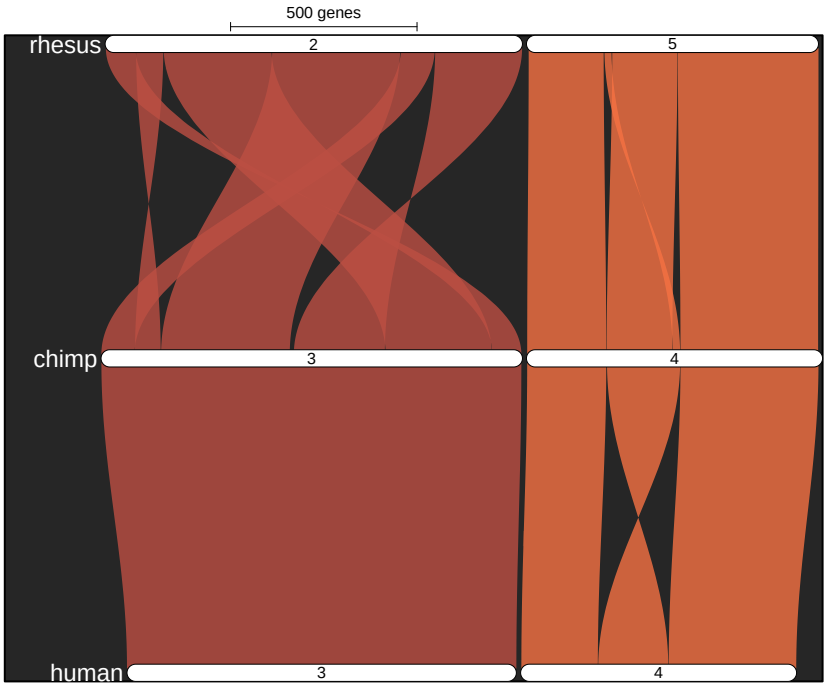
```
##  human  -chimp   (primary): 4282 / 2771 (4) / 2761 (5)
##  human  -rhesus  (primary): 4195 / 2624 (9) / 2613 (12)
##  chimp  -rhesus  (primary): 4402 / 2712 (10) / 2695 (14)
##  Synteny constraints - Done!
##  Syntenic block coordinates written to /results/syntenicBlocks.txt.gz
## Checking synteny-constrained global orthogroups for synOGs
##  n. global OGs = 1955
##  n. syntenic OGs = 2002
## Combining synteny-constrained and inblock orthogroups ...
##  syn OGs: 2002, inblk OGs: 0, combined OGs: 2002
##  Wrote gff to file: /results/gffWithOgs.txt.gz
##  Done!
```

This function populates the results directory (~/Desktop/testGenespace/results) directory with dotplots and annotated blast hits.

3.5 Make multi-species synteny plots GENESPACE visualizes multi-species synteny with a 'riparian' plot. The default specification orders chromosomes by maximum synteny to a reference genome and colors the synteny by their reference chromosome of origins.

```
ripdat <- plot_riparian(
  gpar,
  colByChrs = c("#BC4F43", "#F67243"))
```

```
##  Loading the gff ... Done!
##  Mapping genes against human chromosomes ... Done!
##  Projecting linear coordinate system ... Done!
##  Generating block coordinates ... Done!
##  Rendering plot ...
```



Chromosomes scaled by gene rank order

```
## Done!
```

3.6 Build a pangenome annotation The main output of GENESPACE is a synteny-anchored pangenome annotation, where every unique synteny-constrained orthogroup is placed in a position on the reference genome gene order. This is constructed by `pangenome`. See 4.2 (pan-genome specification and querying) for more details.

```
pg <- pangenome(gpar)
```

```
## Building pangenome source data ...
##  Reading gff ...
##  ... found 5616 genes 5422 array reps in 1760 merged OGs
##  No orthologues found ... will not be added to the pangenome
##  Pulling block coordinates against human ...
##  ... found 5047 anchors for 31 blocks
##  Linear interpolation of gene order in reference ... Done!
##  Orthogroup clustering and median position determination ... Done!
##  Pos. count: 0x = 10, 1x = 1750, 2x = 3, 3x = 0 4x = 0, 4+x = 0
## Building the pan-genome annotation ...
##  Initial build with ...
##  ... 5422 genes, 1760 OGs and 1763 placements and 7 unplaced OGs
##  Adding in array members ...
##  ... found 194 genes, 61 OGs and 61/4 placed/unplaced OGs    Formating and writing the pangenome ...
## Pangenome written to results/human_pangenomeDB.txt.gz
```

3.7 Inspect the pangenome The pangenome long-format is saved to file and provides fast access to large-scale parsing etc. However, for small scale viewing and querying, the `pangenome` function returns a data table, here `pg`, that condenses the hits within each genome into a list. You can explore this manually with starndard R/data.table or R/tidyverse subsetting routines. However, genespace also offers a query function to look at specific regions. Here is what the first 20 entries on human chr3 look like.

```
query_pangenome(
  pg = pg,
  refChrom = "3",
  startOrder = 1,
  endOrder = 5)
```

```
## $raw
##    pgID og chr ord       chimp       human            rhesus
## 1:    1  1   3   1        CHL1        CHL1              CHL1
## 2:    2  2   3   2 CNTN6,CNTN4 CNTN6,CNTN4 CNTN6,CNTN3,CNTN4
## 3:    3  3   3   3       IL5RA       IL5RA             IL5RA
## 4:    4  4   3   4       TRNT1       TRNT1             TRNT1
## 5:    5  5   3   5        CRBN        CRBN              CRBN
##
## $pav
##    pgID og chr ord chimp human rhesus
## 1:    1  1   3   1  TRUE  TRUE   TRUE
## 2:    2  2   3   2  TRUE  TRUE   TRUE
## 3:    3  3   3   3  TRUE  TRUE   TRUE
## 4:    4  4   3   4  TRUE  TRUE   TRUE
## 5:    5  5   3   5  TRUE  TRUE   TRUE
##
## $cnts
```

```
##    pgID og chr ord chimp human rhesus
## 1:    1  1   3   1     1     1      1
## 2:    2  2   3   2     2     2      3
## 3:    3  3   3   3     1     1      1
## 4:    4  4   3   4     1     1      1
## 5:    5  5   3   5     1     1      1
```

Note the array present in all three genomes at position 2. We can pull a separate regions to look at PAV:

```
query_pangenome(
  pg = pg,
  refChrom = "3",
  startOrder = 38,
  endOrder = 42)
```

```
## $raw
##    pgID og chr ord           chimp    human              rhesus
## 1:   38 38   3  38            EMC3     EMC3               EMC3
## 2:   39 39   3  39    LOC107970560                LOC106996836
## 3:   40 40   3  40 FANCD2,FANCD2P2   FANCD2 LOC716070,LOC723158
## 4:   41 41   3  41        FANCD2OS FANCD2OS            FANCD2OS
## 5:   42 42   3  42            BRK1     BRK1               BRK1
##
## $pav
##    pgID og chr ord chimp human rhesus
## 1:   38 38   3  38  TRUE  TRUE   TRUE
## 2:   39 39   3  39  TRUE FALSE   TRUE
## 3:   40 40   3  40  TRUE  TRUE   TRUE
## 4:   41 41   3  41  TRUE  TRUE   TRUE
## 5:   42 42   3  42  TRUE  TRUE   TRUE
##
## $cnts
##    pgID og chr ord chimp human rhesus
## 1:   38 38   3  38     1     1      1
## 2:   39 39   3  39     1     0      1
## 3:   40 40   3  40     2     1      2
## 4:   41 41   3  41     1     1      1
## 5:   42 42   3  42     1     1      1
```

4. Parameter / argument details

4.1 Parameterizing your run There are four major considerations when parameterizing your GENESPACE run.

1. Do you **just want to show a synteny plot** and don't care about the membership of orthogroups in the pangenome?
   - If so, you can set `orthofinderMethod = "fast"` and `diamondMode = "fast"` in `init_genespace()`.
2. Are any of your genomes represented at **higher ploidy than haploid**?
   - If so, you should definitely specify `orthofinderInBlk = TRUE` in `init_genespace()`. This takes longer, but can recover >50% more orthologs than the global orthofinder run.
   - If your polyploid (or have both haplotypes in an outbred diploid genome) genomes have subgenomes that are very closely related, the benefits of the within-block orthofinder runs are tempered, so this isn't as necessary.
3. Are all of your genomes haploid, **very closely related** (e.g. the same species) and you are only interested in orthologs?

- If so, you can get away with `orthofinderMethod = "fast"` and `diamondMode = "fast"` in 'init_genespace().
- This is especially useful if you have a large number of genomes, as this specification can result in 10X speed improvement over orthofinder defaults.

4. What is the expected **scale of synteny** among your genomes?
   - If you think that some of your genomes may have very little retained synteny, you should consider NOT using GENESPACE at all.
   - If there is still complete genome-wide synteny, but it is divided into very small blocks, you should decrease the block size in the synteny search (see 4.3 synteny parameters below).
5. Are you only interested in orthologs, or **are paralogs of interest too**?
   - If you are interested in paralogs (e.g. regions that retain > ploidy syntenic hits), you should look closely at your dotplots.
   - If the dotplots do not capture the paralogous regions, it is likely that they are highly diverged and may not be captured by genome-wide synteny rules. You can adjust this by increasing the number of secondary hits to retain to the number of paralogs you expect to fine.

Below we go into detail about these cases can be handled, but here is some basic advice. For each of these and other GENESPACE functions, you can see the help files in R via `?`. For example, to see the parameteris in synteny, run `?synteny` from R.

4.2 `init_genespace` parameter details See `?init_genespace` for additional information. Here is the important stuff.

- `genomeIDs`: A character vector of > 1 unique entries. The genomeIDs are the labels for all plots, files, columns etc. They cannot be duplicated and can't start with a number of special character. If you wish to compare a genome to itself, you need to specify two separate unique genomeIDs, with identical speciesIDs and versionIDs (see below).
- `wd`: the working directory for the GENESPACE run. This should be an existing directory, or a path within an existing directory to initiate the run.
- `rawGenomeDir`: file path point to the parent directory where are the raw genomes are stored. The directory structure here must follow `/rawGenomes/$speciesID/$versionID/annotion`.
- `speciesIDs`: exact match of a subdirectory in the rawGenomeDir that corresponds to the genomeIDs. So, if there are three genomeIDs, there need to be three speciesIDs that match up to the order of the genomeIDs. for example, the human genome might be stored here: /mammalGenespace/rawRepo/Homo_sapiens/GRCh38.p13/annotation. In this case the speciesID for the human genomeID is "Homo_sapiens". This is case sensitive.
- `versionIDs`: Like speciesIDs, but the next level up in the rawGenomeDir directory structure. So, in the above example, the versionID for the human genome would be "GRCh38.p13".
- `ploidy`: a ploidy for each genome - this is an integer vector that matches the length of genomeIDs. This is not necessarily the ploidy of the organism, but the ploidy of the genome. In most cases this is 1. If not specified, the default is to set all genomes to haploid ploidy.
- `orthofinderMethod`: this can be either 'default' (the default specification or orthofinder, including running diamond with mode = –more-sensitive) or 'fast'. The fast specification can only be run if GENESPACE is initiated in an environment with orthofinder in the path. The 'fast' method accomplishes all unique pairwise blasts, treating the genome with more proteins as the query and the smaller genome as the target. The resulting blast files are inverted so that the reciprocal blasts are also present for orthofinder. The 'fast' method cannot accurately produce orthologs, so is only run to the orthogroup inference step. This means that non-syntenic orthologs are not flagged during pangenome construction.
- `diamondMode`: only respected if orthofinderMethod = "fast". It can be any of the specifications in diamond. Namely, "fast", "sensitive", "more-sensitive", "ultra-sensitive". See diamond help files for more information.
- `nCores`: the number of parallel processes to run throughout the pipeline. By default, this is set to 1, but can be as many as your system can handle. If running on MacOSX and you see a warning during

installation of the data.table library some steps may not be parallelized.

- `gffString`: the unique character string ("word") that identifies your gff files from all others in the rawGenomeDir. Usually "gff", but if you have other files in there (like a repeat annotation gff), you may need to be more specific.
- `pepString`: the unique character string ("word") that identifies your peptide fasta files from all others in the rawGenomeDir. Usually "pep" or "fa", but if you have other files in there (like a cds fa), you may need to be more specific.
- `path2orthofinder`: this should not be specified, but is here for future compatibility with orthofinder installed outside of the path.
- `path2diamond`: this should not be specified, but is here for future compatibility with diamond installed outside of the path.
- `path2mcscanx`: the path to your MCScanX installation. This directory should include an executable called `MCScanX_h`.

4.3 `synteny` parameter details Synteny has a lot of parameters that allow you to fine-tune the synteny search. For most cases the defaults should be fine, but in some cases (e.g. very diverged genomes or studying ancient whole-genome duplicates) it will be crucial to manipulate these. If you do not run `set_syntenyParams` prior to `synteny`, the defaults are used unless you specify them in the call. For example, if you want to only find larger blocks, you could run `synteny(gsParam, blkSize = 20)`. However, if you have previously run `synteny` you will need to separately re-specify your parameters with `set_syntenyParams`, since `synteny` will respect the previously defined parameters in the gsParam list.

Below are a list of the synteny parameters. For many there is a "second" option. This is the parameter specification for homeologs (duplicates within polyploid genomes) or paralogs after masking the orthologous regions.

- `nSecondHits`: the number of secondary hits, after accounting for ploidy. This is usually set to 0, unless you wish to test for paralogous regions. Any parameters below ending in "Second" are passed to these secondary hits.
- `onlyOgAnchors`; `onlyOgAnchorsSecond`: logical, should only orthogroup-constrained hits be considered as candidates for syntenic anchors? This should almost always be TRUE, unless the genomes you are studying are very diverged.
- `synBuff`; `synBuffSecond`: the gene-rank order radius surround an anchor hit to include nearby hits that are in the buffer region.
- `blkSize`; `blkSizeSecond`: the minimum size of a colinear run of genes for that region to be called a syntenic block. This parameter is used for both `dbscan(min.pts)` and `MCScanX_h -s` specification.
- `nGaps`; `nGapsSecond`: the `MCScanX_h -m` parameter; the number of gaps allowed for a collinear region to be called a block.
- `synBuff`; `synBuffSecond`: the gene-rank order position around syntenic anchors within which hits are considered within a syntenic region and retained in the pan-genome db.
- `selfRegionMask`: this is an additional masking parameter for secondary blast hits. This can be increased in cases where tandem arrays are very large and might extend beyond the synBuff region. This is only used when ploidy > 1 or nSecondHits > 0.
- `dropInterleavesSmallerThan`: when two blocks overlap, they are checked for duplicate genes in both block anchors. If there are duplicates, GENESPACE assumes the overlap to be a true duplicate. However, if there are no duplicated block anchors, then its assumed that the block coordinates are inaccurate extensions of interleaved blocks. This is a common problem with MCScanX and is corrected here by splitting out the interleaved blocks. The smallest interleaved blocks to retain are set here. Set to 0 if you wish all anchors to be retained regardless of whether they contain multiple collinear anchors or not.
- `minRbhScore`: to improve block coordinate determination in diverged regions, the reciprocal best hits are added to the orthogroup graphs during block determination. All RBHs are retained with a score greater than this.

4.4 `plot_riparian` parameter details See `?plot_riparian` - you'll notice there are a lot of customization options. So many that it is not useful to describe here. Instead, we have a separate vignette dedicated to plot_riparian. Check that out for more info.

5. Pipeline details

Below, we provide a detailed explanation of the source code pipeline for GENESPACE.

5.1 Syntenic region inference

The `synteny` pipeline acts on pairs of genomes, choosing the genome with the most gene annotation entries as the query sequence. The inputs are (1) the reciprocal blastp hits from diamond, (2) the merged gff with gene rank order position along chromosomes pre-calculated, and (3) orthofinder orthogroups.tsv genome-split orthogroup text file. Below, we use the following terms for parameters / statistics:

- `synBuff`: the gene-rank order radius surround an anchor hit to include nearby hits that are in the buffer region
- `blkSize`: the minimum size of a colinear run of genes for that region to be called a syntenic block. This parameter is used for both `dbscan(min.pts)` and `MCScanX_h -s` specification.
- `nGaps`: the `MCScanX_h -m` parameter; the number of gaps allowed for a collinear region to be called a block.
- `radius`: the `dbscan(eps)` parameter, either taken as the blkSize for final block calculation or synBuff for syntenic region calculation.
- "anchors": blast hits that are in colinear orthogroups that form the backbone for syntenic regions and block regions
- "inBuffer": hits that are within the `synBuff` of anchors.
- "non-syntenic": hits outside of the buffer region
- "blast": here, using this to mean diamond blast-like hits.

5.1.1 Defining self-synteny regions

For each genome, genespace finds the self hits and uses these as anchors, extracting these and inBuffer hits for downstream analysis. These are the "self-syntenic hits".

5.1.2 Defining synteny between two genomes.

This routine is applied to paralogous regions in polyploids (see 4.1.3) and intergenomic hits. The steps for synteny determination are as follows:

1. read in the blast file with `parse_blast4synteny`:
   - reads in and mirrors the blast hits, retaining the unique hit with the highest score
   - adds orthofinder orthogroup information
   - adds genome position (chromosome, bp, gene rank order)
   - adds score-rank, where, for each gene, the blast hit is ranked by highest to lowest scoring
2. subsets out potential anchor hits
   - both query and target genes are array representatives (see 4.3)
   - both query and target genes are in the same orthogroup
   - the top n scoring hits, where n is informed by the ploidy of the query and target genomes.

3. finds initial anchor hits
    - runs MCScanX_h on potential anchors
    - runs dbscan on MCScanX_h-defined colinear hits. These retained genes are the seed anchor hits.
    - pulls all hits within the synBuff of the seed anchors. These are the seed anchor regions
4. refines anchor hits
    - adds within-seed anchor region reciprocal best hits as orthogroups. This can give better block breakpoints in regions where paralogs are not adequately captured in the orthogroups.
    - re-runs MCScanX_h on og or RBH-constrained hits
    - re-calculates gene-rank order
    - re-runs dbscan to get block identities
    - splits overlapping dbscan blocks that are not duplicated.
    - extracts all hits within the block coordinates
    - subsets within-block coordinate hits to those within synBuff of the anchors.
    - These are the syntenic region hits.
5. finalize block coordinates
    - within each syntenic region (see above), MCScanX_h is run and colinear hits are retained
    - gene rank order is re-calculated on colinear hits within regions
    - dbscan is run on colinear hits, using `eps = sqrt((blkSize^2)*2))` and `min.hits = blkSize`. This means that any inversion > blkSize will cause a block breakpoint
    - gene rank order is re-calculated on these smaller blocks and dbscan is re-run with `eps = sqrt(2)+.1)`, so that all hits need to be exactly colinear.
    - as above, hits within the block coordinates and within synBuff are retained.
    - These are the "syntenic block hits".

5.1.3 Defining self-synteny paralogous regions

For genomes with ploidy > 1, the self-syntenic hits are masked from the intragenomic blast hits (4.1.1). The gene rank order is re-calculated and the basic synteny pipeline (4.1.2) is applied to the data. The user can specify that paralagous hits should have different synteny parameters than the self/primary hits. See `set_synParams` for more info.

5.2 Producing syntenic orthogroups

... coming soon

5.3 Building pangenomes

... coming soon

6. Legal

reproduce, distribute copies to the public, prepare derivative works, and perform publicly and display publicly, and to permit others to do so.

*** License Agreement ***