GENESPACE readme

John T. Lovell

19-September 2019

GENESPACE in a nutshell

The GENESPACE R package, in its most basic form, constructs syntenic blocks from blast results, culls blast results to those in proximity to syntenic blocks, then parses the syntenic blast hits into orthologs, paralogs and un-clustered homologs. GENESPACE does this on whole-genome annotation data (i.e. gff3 and peptide fasta files) and can opperate on anywhere from 2 to 16 diploid genomes simultaneously (depending on memory availability.

GENESPACE includes a primary pipeline, which finds syntenic homologs, orthologs and paralogs (*see the 'pipeline documentation' subheading below for full details on each function call*). The package also includes a method for genome annotation parsing, three main plotting routines, and syntenic block parsing and gene-location predicting functions. The plotting and parsing functions must be run interactively in R. The pipeline can be run interactively as a single function call. In the future, we will also implement running the pipeline as an Rscript call from the command line.

In R, run the pipeline as a single command: `pipe_GENESPACE`. There are three major settings for `pipe_GENESPACE`, which primarily affect the size, density and blast-scores of hits included in syntenic blocks:

The default pipeline:

Blast results calculation and processing:

All pairwise blast hits are calculated with `diamond` either separately, or within the `orthofinder` program. Blast results are then culled to the top 2 hits within each haplotype (so, if a diploid is mapped to a dipliod, four hits would be retained for each gene; if it were mapped to a tetraploid, eight hits would be retained). All hits with a bit score < 50 are dropped.

Initial orthogroup inference:

separate run of `orthofinder` is made for each pair of genomes using the culled blast results. Only those hits within 'orthogroups' are retained.

Pruning blast hits to synteny:

Blocks are formed using `MCScanX` from the culled and orthogroup-constrained blast results, allowing 5x as many gaps in the alignment as 50% of the minimum block size (MBS, default = 10), then pruned with `dbscan` to blocks with MBS hits within a fixed gene-rank radius 5x MBS. All orthogroups are then 'completed', where `igraph` expands the orthogroups to include all possible combinations among genomes. These blast hits are then pruned with `dbscan` with identical parameters as above.

Block cleaning and extension:

To fill potential gaps in blocks left by the stochastic nature of varying orthogroup connectivity, we pull all blast results that passed the score threshold (agnostic to orthogroup identity) that are within a 100-gene radius of any syntenic block and re-form blocks with `MCScanX` with the same parameters as above.

Syntenic orthology inference:

All blast results are culled to those within a 50-gene rank radius of any syntenic block for all genomes. `orthofinder` is run on this entire set, and blast hits are parsed into orthologs, paralogs, or un-clustered homologs. By default, hits that were not in an orthogroup (neither orthologs or paralogs) with a score < 50 or $< 50\%$ of the best bit score for that gene -by- unique genome combination are dropped from this dataset.

The 'relaxed' method:

This follows the default closely, but allows for smaller blocks (MBS = 5) and more gapped alignments (10x MBS). The major difference is that the inital orthofinder run is conducted on all genomes simultaneously. For a set with few, closely related genomes, this will largely result in the same blocks as default; however, as more genomes are included across greater phylogenetic distances, this can retain more diverged regions.

The 'ancientWGD' method:

The goal with this parameterization is to retain as many hits as possible and to capture regions associated with ancient whole-genome duplications (WGDs). To do this, the initial orthofinder step is completely ignored, MBS = 5 and 20x MBS large gaps are allowed. The intial block extension radius is doubled to 200 genes and homologs with a score > 20 and 20% of the best score are retained.

Example 1: Syntenic orthology among Human, Chimpanzee and Mouse genomes

For this and other examples, we will not run the pipeline in one fell swoop, but will work our way though it step by step.

The gff3 and peptide annotation files were downloaded from (ensemble genomes)[https://uswest.ensembl.org/info/data/ftp/index.html] on 19-September 2019.

The first step of GENESPACE is to move the annotation files into a subdirectory of your working directory called '/raw_annotations'. You also can make a subdirectory called '/raw_assemblies', but this can be empty, or contain the assembly fastas, labeled [genomeID].fa. For each genome of interest, make a folder with the name you want to call the genome.

So, the directory structure looks like this:

```
/genespace_directory
  /raw_annotations
    /human
      /Homo_sapiens.GRCh38.97.abinitio.gff3.gz
      /Homo_sapiens.GRCh38.pep.abinitio.fa.gz
    /chimanzee
      /Pan_troglodytes.Pan_tro_3.0.97.abinitio.gff3.gz
      /Pan_troglodytes.Pan_tro_3.0.pep.abinitio.fa.gz
    /mouse
      /Mus_musculus.GRCm38.97.abinitio.gff3.gz
      /Mus_musculus.GRCm38.pep.abinitio.fa.gz
  /raw_assemblies
    /human.fa
    /chimpanzee.fa
    /mouse.fa
```

GENESPACE requires a very specific format for the gff and the fasta files: the attribute column of the gff must have an entry that matches exactly the header name in the fasta. But the files in the `raw_annotation`

subdirectory can be any format, but must be parsed by the `convert_genomes` function. This decompresses the files and parses the fasta header via a user-specified function and regular expression.

For example, the first three raw human sequences:

```
>GENSCAN00000000001 pep chromosome:GRCh38:5:122151991:122153085:1 transcript:GENSCAN00000000001 transcri
MERGKKKRISNKLQQTFHHSKEPTFLINQAGLLSSDSYSSLSPETESVNPGENIKTDTQK
...

>GENSCAN00000000002 pep chromosome:GRCh38:5:122675795:122676286:1 transcript:GENSCAN00000000002 transcri
MMNRMAPENFQPDPFINRNDSNMKYEELEALFSQTMFPDRNLQEKLALKRNLLESTGKGL
...

>GENSCAN00000000003 pep chromosome:GRCh38:5:121876146:121916483:-1 transcript:GENSCAN00000000003 transci
MDDSKGNGKRAKIRGKGPKIFLKSLLATLPNTSYVCASEPQLSPYLCEFFPGVNLLDVEH
...
```

So, we run `convert_genome`, which adds a new subdirectory to the working directory. To do this, we need to specify the character strings that identify the peptide files and the gff. The peptide files are named "[speciesID].[genomeVersion].pep.abinitio.fa.gz". The gff3 files are named similarly "[speciesID].[genomeVersion].abinitio.gff3.gz".

```
convert_genomes(
  directory = wd,
  genomeIDs = genomeIDs,
  verbose = T,
  peptide_str = "pep",  # 'pep' distinguishes the peptide fasta files
  peptide.only = T,     # only worry about the peptide and gff3 files
  gff_str = "gff"       # 'gff' distinguishes the gff3 annotations
)
```

```
/genespace_directory
  /raw_annotations
  /raw_assemblies
  /genome
    /gff
      /human.gff3
      /chimpanzee.gff3
      /mouse.gff3
    /peptide
      /human.fa
      /chimpanzee.gff3
      /mouse.gff3
```

The `human.fa` peptide fasta in the genome/peptide subdirectory now has different headers:

```
>GENSCAN00000000001
MGVKMLLVLQSETGRGRRRRGGRRGRRRRRREEKKEDEEEEAVAVAAAVAVEEEEGEEEGEEEGVEEEEEDGRRRKKKKK

>GENSCAN00000000002
MAGSQPQRLAHPLLSLPLTSASGSPGLEQEDRTTRAINRVTWPCIMIWCRIPLCEAKAASANFSTWKAVTSPCALVAQNP

>GENSCAN00000000003
MENQIIQTAGRTGQESGAAHRRDDATSILQHLGSPAKGWENCQGEGVSDEALAPTMLLLLLVFFLRKPFVSIAVLIFHVT
```

The gff3 annotations are unchanged, except that they have been renamed as [genomeID].fa and decompressed. The first six lines of `human.gff3` looks like this (below). You'll notice that only entries where the 3rd column has 'mRNA' have entries in the 9th 'attribute' column that match the geneIDs in the fasta.

```
1   ensembl mRNA    12190   14149   .   +   .   ID=transcript:17672;Name=GENSCAN00000017672;version=1
1   ensembl exon    12190   12227   .   +   .   Parent=transcript:17672;Name=127823;exon_id=127823;vers
1   ensembl exon    12613   12721   .   +   .   Parent=transcript:17672;Name=127824;exon_id=127824;vers
1   ensembl exon    14051   14149   .   +   .   Parent=transcript:17672;Name=127825;exon_id=127825;vers
1   ensembl mRNA    14696   24886   .   -   .   ID=transcript:17670;Name=GENSCAN00000017670;version=1
1   ensembl exon    14696   14829   .   -   .   Parent=transcript:17670;Name=127818;exon_id=127818;vers
```

So, when we read in the gff3 annotation, we'll want to keep the mRNA lines, and retain the 2nd entry in the attribute column, removing the 'Name=' string. In GENESPACE, we read the gff into memory and parse it use `import_gff`; this also simplifies the output to just the genomeID, geneID, chromosome, start, end, and strand data.

```
gff <- import_gff(
  gff.dir = dir.locs$gff,
  genomeIDs = genomeIDs,
  use = "mRNA")
```

```
                   id chr  start    end strand genome order
1: GENSCAN00000017672   1  12190  14149      +  human     1
2: GENSCAN00000017670   1  14696  24886      -  human     2
3: GENSCAN00000017677   1  51913 106974      +  human     3
```

The last pre-processing step is to build out the remaining subdirectories and get a list of these in memory with: `check_env`.

```
dir.locs <- check_env(
  directory = genespace/working/directory,
  genomeIDs = c("Human", "Chimpanzee", "Mouse"),
  peptide.only = T)
```

Which adds a few new subdirectories to the file tree:

```
/genespace_directory
  /raw_annotations
  /raw_assemblies
  /genome         # parsed and decompressed annotations and assembly files
  /blast          # populated by run_orthofinder
  /tmp            # generic tmp directory
  /cull.blast     # blast files culled to the genomes of interest
  /cull.score     # blast files culled score thresholds
  /syn.blast      # blast files culled to synteny
  /mcscanx        # mcscanx working directory
  /results        # where results are stored
```

The `dir.locs` object that it creates contains paths to each of these subdirctories.

Now that we have all of the files formatted correctly and put in the right place, we can get our blast results. `run_orthofinder` is a wrapper for the `orthofinder` program, but by default, only re-formats the files and runs the pairwise reciprocal blast searches with `diamond`. Blast searches can be slow. If doing more than 3 or 4 genomes and on a personal computer, consider running this overnight, or doing it on a cluster and copying the /blast directory back into your current working directory.

```
with(dir.locs,  run_orthofinder(
  output.dir = blast,
  overwrite.output.dir = T,
  peptide.dir = peptide,
  blast.threads = 9,
  og.threads = 4))
```

So, everything is ready to run the main GENESPACE pipeline.

Step 1: Cull the blast results to the genomes of interest and the top hits for each gene.

```
init <- remake_ofInput(
  dir.list = dir.locs,        # the list of subdirectories from check_env
  genomeIDs = c("Human", "Chimpanzee", "Mouse"),      # the names of the genomes, matches the names of t
  ploidy = c(2,2,2),          # ploidies
  gff = gff,
  min.score = 50,             # no blast hit less than this score are kept
  run.of = F,                 #
  max.dup = 4,                # maximum number of hits to retain / diploid genome
  overwrite.output.dir = F,
  n.cores = 6)
```

Step 2: Run orthofinder algorthm, restricted to each pair of genomes and on the culled blast hits

```
pwblast <- rerun_orthofinder(
  dir.list = dir.locs,
  gff = gff,
  genomeIDs = genomeIDs,
  n.cores = 6,
  method = 'pairwise')      # do the pairwise method
```

Step 3: Form collinear blocks via MCScanX

```
mcsblks <- run_MCScanX(
  blast = mirror_map(pwblast),
  gff = gff,
  mcscan.dir = dir.locs$mcscanx,
  overwrite.output.dir = T,
  genomeIDs = genomeIDs,
  MCScanX.path = MCScanX.path,      # path to MCScanX install
  MCScanX.s.param = 10,             # the minimum number of hits in a block
  MCScanX.m.param = 100,            # the number of gaps allowed
  verbose = T)
```

Now is a good time to check out plots to see what the initial collinear blocks look like

```
chrl <- match_synChrs(
  gff = gff,
  map = mcsblks$map,
  genomeIDs = genomeIDs,
  smallestchrsize2keep = 100,
  genome1.chrs = c(1:23, c("X","Y")))
dp.res <- plot_map(
  chr.list = chrl,
  map = mcsblks$map,
  palette = circos.palette,
  gff = gff,
  genomeIDs = genomeIDs,
  plot.self = T)
```

clnblks <- clean_blocks(map = mcsblks$map, gff = gff, genomeIDs = genomeIDs, complete.graphs = F, n.mappings = 20, radius = 100)

comp.map <- complete_graph(map = clnblks$map, gff = gff, ignore.self = T, expand.all.combs = T)

ext.blast <- extend_blocks(gff = gff, genomeIDs = genomeIDs, map = comp.map, dir.list = dir.locs, rank.buffer = 100, verbose = T)

mcsblks2 <- run_MCScanX(blast = mirror_map(ext.blast), gff = gff, mcscan.dir = dir.locs$mcscanx, overwrite.output.dir = T, genomeIDs = genomeIDs, MCScanX.path = MCScanX.path, MCScanX.s.param = 10, MCScanX.m.param = 50, verbose = T)

assblks <- assign_synHomologs(genomeIDs = genomeIDs, map = mcsblks2$map, min.score4homolog = 50, min.propBestScore4homolog = .5, gff = gff, dir.list = dir.locs, rank.buffer = 50, verbose = T, quiet.orthofinder = T, n.cores = 6)

Introduction

As chromosome-scale whole genome assemblies and annotations become more common across many taxonomic groups, it is crucial to be able to make evolutionary inference regarding sequences that are derived from common ancestors. There are two major lines of evidence to test whether genes are related: 1) sequence similarity (i.e. homology) and 2) similarity of physically proximate genes (i.e. synteny). If genes from two species are homologous, it is possible they arose from duplications (i.e. paralogs), are members of conserved similar gene familes but are unrelated, and arose from a speciation event and share a common ancestor (i.e. orthologs). The same is true for synteny, which can be related to ancient duplications or retained gene order since speciation. However, our confidence in the type of evolutionary relationship between sequences can be increased by testing for the presence of *syntenic* orthologs and paralogs.

To date, a number of programs can infer synteny and homology, and a number of studies have combined the two principals. The R package, 'GENESPACE' does this explicitly by first inferring homologous gene networks ('orthogroups') through the `orthofinder` program, then parses orthogroups into syntenic blocks via `MCScanX`. Finally, blast hits are constrained to syntenic regions and classified as orthologous or paralogous through `orthofinder`. The three output files are 1) a bedmap with syntenic block coordinates, 2) an annotated blast file with all syntenic homologs, and 3) a bedmap with the expected syntenic blocks of all genes.

What can GENESPACE do for me?

GENESPACE is an R package designed to improve the speed, sensitivity, utility and visualization of comparative genomics in the post-genomic world. It is best run interactively in R, but also can be called as a pipeline from the command line.

The primary functionality of GENESPACE is **to develop high-confidence mappings between orthologous genes from multiple related genomes**. Depending on the system and parameters, GENESPACE can also effectively map paralogs among genomes that have undergone both recent and ancient whole genome duplication (WGD) events (e.g. how much synteny is retained, age of WGD, pattern of gene retention, etc.). GENESPACE also can produce highly-customizable publication-quality plots. It outputs datasets that can be used to infer a number of downstream attributes, including tandem arrays, pseudogenization, ancestral state reconstructions and selection / neutrality staatistics.

Data Requirements

GENESPACE is written for default parameters to work directly off the JGI phytozome geneome annotation format (https://phytozome.jgi.doe.gov/pz/portal.html); however, nearly all formatting can be accomodated.

NOTE Depending on the RAM available, GENESPACE can run up to 12 diploid genomes (6 tetraploid genomes, etc.). However, we recommend doing smaller runs (<= 8 genomes) unless synteny is at least marginally conserved across all species.

To run GENESPACE, you need:

1. complete gene annotations from >= 2 genomes.
2. gff3-formatted gene annotations
3. fasta files containing peptide sequence for the primary transcript of each gene.
4. the gff3 and fasta files need to have gene identifiers that can be used to link the two.
5. there must be some synteny between genomes. GENESPACE cannot be used to find related genes between very diverged species (i.e. plants and animals), nor should it be used to look at very ancient whole genome duplications.

NOTE Genomes must have decent contiguity. The required level of contiguity depends on the minimum desired syntenic block size. But, the larger the syntenic blocks, the better the results. So, if using genomes with small or un-ordered contigs, take the results lightly.

System Requirements

1. MacOSX or LINUX
2. R v3.5.2 or later
3. OrthoFinder version 2.3.3 or later
4. MCScanX downloaded from http://chibba.pgml.uga.edu/mcscan2/

Running GENESPACE interactively

With the initial v0.5 release, the GENESPACE pipeline must be run interactively. Later releases will have a commaand line one-liner to call the whole thing.

Data formatting

GENESPACE requires an orthofinder-formatted blast search result, where the gff3 'attribute' columns match the gene IDs in the fasta headers. We can achieve these results via the sata preparatation pipeline:

1. `convert_genomes`: Format / simplify fasta head and match the gene ID in gff3.
2. `run_orthofinder`: Wrapper to run orthofinder and aggregate results
3. `remake_ofInput`: Subset blast results (i.e. to top hits by gene)
4. `rerun_orthofinder`: Re-run orthofinder on subsetted blast results.

Forming and finalizing syntenic blocks

Determining sequence similarity is fairly trival. Programs like orthofinder can parse such blast results.

5. `pipe_mcscanx`: an R wrapper for the MCScanX program. Clusters hits into collinear blocks and drops non-collinear blast hits.
6. `clean_blocks`: uses fixed-radius 2d density clustering to retain hits in high-quality blocks.
7. `complete_graph`: fills out incomplete complete networks (e.g. for genes A,B,C: [A-B,B-C] –> [A-B,B-C,C-A])
8. `extend_blocks`: searches for any blast hits within a fixed radius of previously defined syntenic blocks. Crucial for under-retained duplicate regions, as these may get broken up into small pieces by MCScanX and clean_blocks. Should be followed by a final pipe_mcscanx run.

Classifying hits as orthologous, paralogous or other

9. `assign_synHomologs`: searches for all syntenic blast hits, feeds these to orthofinder, builds gene trees and returns an annotated blast file with a column specifying the hit as ortholog, paralog, or syntenic homolog.

An example with mammals.