BGA Academy: GENESPACE

JTLovell

2024-10-07

1. Get the run going

We are going to be doing everything here from a terminal instance of R. When doing the run on your own, you might want to open with Rstudio to more easily visualize graphics ect. Here, we'll save all graphics as pdf, then open them with the file explorer on the top left.

Navigate to the GENESPACE working directory, which already contains all the required input data:

```
cd genespace_data
```

If there are not directories named `pep` and `bed` and these don't contain two files each in this working directory, there is a problem with your gitpod and we'll need to start over. We'll go over what the input data looks like once the run starts. Check via:

```
ls -l bed
ls -l peptide
```

GENESPACE is an R package (sorry) to enter R from gitpod keeping the paths, just type `R` in the terminal.

```
R
```

Which will open the R environment and keep the paths … this should print the following information to the console - this is the standard R startup message

```
R version 4.4.1 (2024-06-14) -- "Race for Your Life"
Copyright (C) 2024 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
```

Then, once in R, you load `GENESPACE` as:

```
library(GENESPACE)
```

Now, all is ready to get the run initialized:

```
workHere <- "/workspace/GENESPACE/genespace_data"
pth2mcs <- "/usr/local/bin"
gsInit <- init_genespace(wd = workHere, nCores = 4, path2mcscanx = pth2mcs)
gsOut <- run_genespace(gsParam = gsInit)
```

2. Input file structure (vamping while the run finishes)

The GENESPACE run will take about 10 mins. Most of this time is running OrthoFinder and diamond (fast blast-like search). While this goes, lets take a quick look at the input files ...

The `init_genespace` function looks for two input directories in the working directory (`wd`):

- `/peptide`: input fasta file with amino acid sequences
- `/bed`: bed-formatted annotation input file where the 4th 'id' column exactly matches the headers in the peptide fastas.

There needs to be exactly the same name of files in each directory and they need to be named identically. Here, the structure looks like this:

```
/workingDirectory
-- peptide
     - human.fa
     - chicken.fa
-- bed
     - human.bed
     - chicken.bed
```

3. What GENESPACE does: Run orthofinder

After completing reciprocal pairwise diamond searches, Orthofinder aggregates homologs into 'orthogroups' which is an undirected cyclic graph of all orthologs for each gene. OrthoFinder produces a ton of intermediate files (e.g. trees for most orthogroups etc.), which are not directly needed by GENESPACE. Really, all GENESPACE needs are the blast files, orthogroup memberships and pairwise ortholog edges. GENESPACE copies these files into /results. You can also see a summary of the run printed to the console ...

4. Make dotplots from blast hits

At its heart, GENESPACE does synteny searches on blast hits constrained to hits where query/target genes are in the same orthogroup.

We can load these data into R with utility functions provided by GENESPACE:

syntenic blast hits:

```
humanChickenAllBlastFile <- file.path(
  workHere, "syntenicHits", "human_vs_chicken.synHits.txt.gz")
ab <- read_synHits(humanChickenAllBlastFile)
head(ab)
```

And you can viz it in whatever way you like . . . GENESPACE offers a couple different dotplotting routines. For example (if using Rstudio, you could see this in the plot viewer without save to a pdf):

```
pdfFile <- file.path(workHere, "customSexChrDotplot.pdf")
pdf(pdfFile,  height = 8, width = 8)
sexChrDat <- subset(ab, chr2 == "Z" | chr1 == "X")
gghits(sexChrDat, alpha = 1, useOrder = FALSE)+
  ggplot2::ggtitle("custom sex chromosome dotplot")+
  ggplot2::theme_bw()
dev.off()
```

5. Combine OrthoFinder and synteny information into one text file:

We call this file a "combined/annotated bed", and it is stored in `results/combBed.txt`:

```
combinedBedFile <- file.path(
  workHere, "results", "combBed.txt")
cb <- read_combBed(combinedBedFile)
```

The columns in this file contain most of the main important output for GENESPACE:

- `globOG` now depricated 'orthogroup.txt' OrthoFinder output
- `globHOG` phylogenetically hierarchical orthogroups in the N0.txt OrthoFinder output
- `arrayID` vector of syntenic (often tandem) array IDs - every gene gets a value here, but those in arrays have >1 member.
- `og` the synteny-constrained orthogroup (either OGs or HOGs, depending on if `useHOGs = T/F`)

Many of the main questions you may want to answer can be done by parsing this text file. For example, what are the syntenic orthologs in chicken to the human HLA genes?

```
hlOgs <- subset(cb, grepl("HLA", id))$og
hlBed <- subset(cb, og %in% hlOgs & genome == "chicken")
print(hlBed)
```

And you could then pull the hits associated with these genes:

```
hlHits <- query_hits(gsParam = gsOut, bed = hlBed[1,], synOnly = TRUE)
```

6. Tabulate gene PAV

Gene copy-number- and presence-absence-variation are often cited and obvious statistics from pan-gene annotations. We can pull these in GENESPACE with `query_cnv` which returns copy number and presence absence variation for each of the three orthogroup types and each genome.

If we only want to see what the global HOG gene PAV looks like (subsuming any gene family with more than 1 representative to 1 in the resulting table), we could run:

```
cnvs <- query_cnv(gsParam = gsOut, maxCopyNumber = 1)
print(subset(cnvs, ogType == "globalHOGs"))
```

7. Track syntenic gene families across multiple genomes

Under the hood, this requires first interpolating the syntenic position of all genes in all genomes. Once we have this dataset (written to file as `/pangenomes/[referenceGenome]_integratedSynPos.txt`), we can take a region of interest and ask what genes lie therein.

```
query_pangenes(
    gsParam = gsOut,
    bed = data.frame(genome = "human", chr = "X", start = 0, end = 2e5))

query_pangenes(
    gsParam = gsOut,
    bed = data.frame(genome = "chicken", chr = "Z", start = 0, end = 1e5))
```

8. Make 'riparian' synteny maps

This is probably the main functionality of GENESPACE. Here are some examples of how to customize it:

```
xzBed <- data.frame(genome = c("human", "chicken"), chr = c("X", "Z"), color = c("red", "cyan"))
pdfFile <- file.path(workHere, "customRiparians.pdf")
pdf(pdfFile,  height = 2, width = 8)
p <- plot_riparian(gsParam = gsOut, pdfFile = NULL)
p <- plot_riparian(gsParam = gsOut, pdfFile = NULL, useOrder = FALSE)
p <- plot_riparian(gsParam = gsOut, pdfFile = NULL, highlightBed = xzBed)
dev.off()
```

Try customizing yours ... see the tutorial for ideas.