GENESPACE MS part 1: Cotton analysis

16 February, 2022

Table 1

Overview

The premise here is that the most precise and sensitive methods will produce the most orthogroups that contain a single copy of each gene (or representative gene for a tandem array) on each of the three genomes and each of the two subgenomes within each genomes. To accomplish this test, we parse the `gffWithOgs.txt.gz` files that `GENESPACE::synteny()` produces. This file is a simplified gff3 annotation with a number of additional data columns added on. See `GENESPACE::annotate_gff()` and `GENESPACE::synteny()` for more information.

Script to build Table 1

```
wd <- "/Users/lovell/Desktop/manuscripts/genespace_2022/GENESPACE_data/results"
library(GENESPACE)
```

Read in the genespace parameter list from the primary script, keeping only cotton

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

```
library(ggplot2)
load(file.path(wd, "gparCotton2x.rda"))
load(file.path(wd, "gparCotton4x.rda"))

gpar <- gparCotton4x
gspl <- gparCotton2x
gparSpl <- gspl
bn2x <- file.path(wd, "cotton_split")
bn4x <- file.path(wd, "cotton4x")
gpar$paths$results <- file.path(bn4x, "results")
gspl$paths$results <- file.path(bn2x, "results")

gpar$paths$mcscanxCall <- "/Users/lovell/Desktop/programs/MCScanX/"
gspl$paths$mcscanxCall <- "/Users/lovell/Desktop/programs/MCScanX/"

gpar$paths$blastDir <- file.path(bn4x, "/orthofinder/Results_Feb13/WorkingDirectory")
gspl$paths$blastDir <- file.path(bn2x, "/orthofinder/Results_Feb13/WorkingDirectory")

gpar$paths$orthogroupsDir <- file.path(bn4x, "/orthofinder/Results_Feb13/Orthogroups")
gspl$paths$orthogroupsDir <- file.path(bn2x, "/orthofinder/Results_Feb13/Orthogroups")
```

```
# -- Read in the gffs
gtet <- fread(file.path(gpar$paths$results, "gffWithOgs.txt.gz"),
              na.strings = c("", "NA"))
gfspl <- fread(file.path(gspl$paths$results, "gffWithOgs.txt.gz"),
              na.strings = c("", "NA"))

# -- combine (long format)
gtet[,run := "tet"]
gfspl[,run := "spl"]
gff <- rbind(gtet, gfspl)

# -- subset to non-scaffold and array rep genes
gff <- subset(gff, !grepl("scaff", chr) & isArrayRep)

# -- reformat chr IDs, splitting subgenome and chr numbers
gff[,`:=`(chrn = as.numeric(substr(chr, 2, 3)), subg = substr(chr,1,1))]

# -- subset to non chr2-5
gff <- subset(gff, !chrn %in% 2:5)

# -- strip off A/D genome classes in split run
gff[,genome := gsub("A|D", "", genome)]
```

Parse the gff-like annotations

```
# -- reformat to long
m <- melt(
  gff,
  id.vars = c("ofID", "arrayID", "run", "chrn", "subg", "genome"),
  measure.vars = c("globOG", "synOG", "og"),
  variable.name = "runType", value.name = "ogID")

# -- count chrs, subgen, genomes, geneIDs for each run, OG type and OGID
cnts <- m[,list(nChrns = uniqueN(chrn),
                nSubg = uniqueN(subg),
                nIDs = uniqueN(ofID),
                nGenome = uniqueN(genome)),
          by = c("ogID", "runType","run")]

# -- classify as single copy or not
cnts[,is1x := nChrns == 1 & nSubg == 2, nIDs == 2]

# -- count unique OGs that are single copy for each runType and OG
cnt <- cnts[,list(uniqueN(ogID[is1x])),
            by = c("runType","run")]
```

Count the 1x copy orthogroups

Make and print the table **Summary of orthogroup ('OG') inference for polyploids**. Orthofinder was run using default settings on three tetraploid inbred cotton genomes (represented as diploid assemblies) and six split subgenomes. Counts of single-copy orthogroups (more = better) are presented for nine cotton chromosomes.

```
# -- reformat wide
out <- dcast(cnt, runType ~ run, value.var = "V1")

# -- calculate present better (spl over tetraploid)
out[,percBetter := round(100*((spl - tet)/spl),1)]

# -- print result
knitr::kable(out[,c("runType","tet", "spl","percBetter")])
```

runType	tet	spl	percBetter
globOG	15394	18320	16.0
synOG	16217	21537	24.7
og	22303	21893	-1.9

Table 2

The premise here is that for the most part these cotton genomes are collinear and that the position on one subgenome corresponds exactly to a single syntenic position on the alternative subgenome. To test the sensitivity of syntenic block breakpoint calculation, we compare the proportion of the Pima cotton subgenomes in single-copy, multi-copy and absent. Higher % of single copy is better.

Parse the syntenic block breakpoints text files.

```
# -- read in the hits and gff
h <- fread(
  file.path(gpar$paths$results, "Gbarbadense_Gbarbadense_synHits.txt.gz"),
  na.strings = c("NA", ""), showProgress = F)
gff <- fread(file.path(gpar$paths$results, "gffWithOgs.txt.gz"),
             na.strings = c("", "NA"))

# -- subset to non-scaffold and get chr coords
gff <- subset(gff, !grepl("scaff", chr) & genome == "Gbarbadense")

# -- reformat chr IDs, splitting subgenome and chr numbers
gff[,`:=`(chrn = as.numeric(substr(chr, 2, 3)), subg = substr(chr,1,1))]

# -- subset to non chr2-5
gff <- subset(gff, !chrn %in% 2:5)

# -- get synteny parameters for just Pima cotton
pimaSyn <- subset(gpar$params$synteny, genome1 == genome2 & genome1 == "Gbarbadense")
idsA <- subset(gff, subg == "A")$ofID
idsD <- subset(gff, subg == "D")$ofID
pimaHits <- subset(h, ofID1 %in% idsA & ofID2 %in% idsD)
```

```
pimaHits <- subset(pimaHits, substr(chr1, 2, 3) == substr(chr2, 2, 3))
pimaHits[,u := paste(ofID1, ofID2)]
```

Calculate the block breakpoints

```
# -- calculate block coordinates from raw hits
tmp <- run_mcscanx(
  gsParam = gpar,
  hits = pimaHits,
  blkSize = pimaSyn$blkSize,
  nGaps = pimaSyn$nGaps,
  path2mcscanx = gpar$paths$mcscanxCall)
rawMcs <- subset(pimaHits, u %in% names(tmp))
rawMcs[,blkID := tmp[u]]
rawMcs <- calc_blkCoords(rawMcs)

# -- calculate block coordinates from collinear array reps
tmp <- run_mcscanx(
  gsParam = gpar,
  hits = subset(pimaHits, isRep1 & isRep2),
  blkSize = pimaSyn$blkSize,
  nGaps = pimaSyn$nGaps,
  path2mcscanx = gpar$paths$mcscanxCall)
repMcs <- subset(pimaHits, u %in% names(tmp))
repMcs[,blkID := tmp[u]]
repMcs <- calc_blkCoords(repMcs)

# -- calculate block coordinates from only hits in the same orthogroup
tmp <- run_mcscanx(
  gsParam = gpar,
  hits = subset(pimaHits, isOg),
  blkSize = pimaSyn$blkSize,
  nGaps = pimaSyn$nGaps,
  path2mcscanx = gpar$paths$mcscanxCall)
ogMcs <- subset(pimaHits, u %in% names(tmp))
ogMcs[,blkID := tmp[u]]
ogMcs <- calc_blkCoords(ogMcs)

# -- pull corresponding syntenic block coords from genespace run
gsBlks <- subset(pimaHits, !is.na(blkID) & isAnchor)
gsBlks <- calc_blkCoords(gsBlks)

# -- reformat blks into single window
gsb <- rbind(
  with(rawMcs, data.table(
    chr = chr1, start = startBp1,
    end = endBp1, blkID = paste0(blkID, "_A"), type = "mcs_raw")),
  with(rawMcs, data.table(
    chr = chr2, start = minBp2,
    end = maxBp2, blkID = paste0(blkID, "_D"), type = "mcs_raw")),
  with(repMcs, data.table(
    chr = chr1, start = startBp1,
```

```
    end = endBp1, blkID = paste0(blkID, "_A"), type = "mcs_rep")),
  with(repMcs, data.table(
    chr = chr2, start = minBp2,
    end = maxBp2, blkID = paste0(blkID, "_D"), type = "mcs_rep")),
  with(ogMcs, data.table(
    chr = chr1, start = startBp1,
    end = endBp1, blkID = paste0(blkID, "_A"), type = "mcs_og")),
  with(ogMcs, data.table(
    chr = chr2, start = minBp2,
    end = maxBp2, blkID = paste0(blkID, "_D"), type = "mcs_og")),
  with(gsBlks, data.table(
    chr = chr1, start = startBp1,
    end = endBp1, blkID = paste0(blkID, "_A"), type = "gs")),
  with(gsBlks, data.table(
    chr = chr2, start = minBp2,
    end = maxBp2, blkID = paste0(blkID, "_D"), type = "gs")))
setkey(gsb, chr, start, end)
```

Make a 10kb-spaced grid on each genome and pull overlaps

```
# -- make 10kb grid for genic region of chrs
tmp <- subset(gff, ofID %in% c(pimaHits$ofID1, pimaHits$ofID2))
chrCoords <- tmp[,list(start = min(start), end = max(end)), by = "chr"]
wind <- 1e4
grd <- chrCoords[,list(
  start = seq(from = start + (wind/2), to = end - (wind/2), by = wind)),
  by = "chr"]
grd[,`:=`(end = start, grdID = 1:.N)]
setkey(grd, chr, start, end)

# -- calculate overlaps
ovlps <- foverlaps(grd, gsb)
```

Count the number of hits for each member of the grid

```
# -- count overlaps by grid
cnts <- ovlps[,list(n = uniqueN(blkID[!is.na(blkID)])), by = c("type", "grdID")]
cnts <- dcast(cnts, grdID ~ type, value.var = "n")
cnts[,`NA` := NULL]
cnts[is.na(cnts)] <- 0
cnts <- melt(cnts, id.vars = "grdID", value.name = "n", variable.name = "runType")
cnts[,cat := ifelse(n > 1, "2+", as.character(n))]
sumCnts <- cnts[,list(n = .N), by = c("runType", "cat")]
sumCnts[,perc := round((n/sum(n)) * 100, 1), by = "runType"]
```

Print the table

```
# -- reformat wide
sumCnts[,runType := factor(runType, levels = c("mcs_raw", "mcs_rep", "mcs_og", "gs"))]
out <- dcast(sumCnts, runType ~ cat, value.var = "perc")

# -- print result
knitr::kable(out)
```

runType	0	1	2+
mcs_raw	6.5	79.4	14.1
mcs_rep	6.1	82.8	11.1
mcs_og	6.1	91.5	2.5
gs	5.6	93.7	0.7

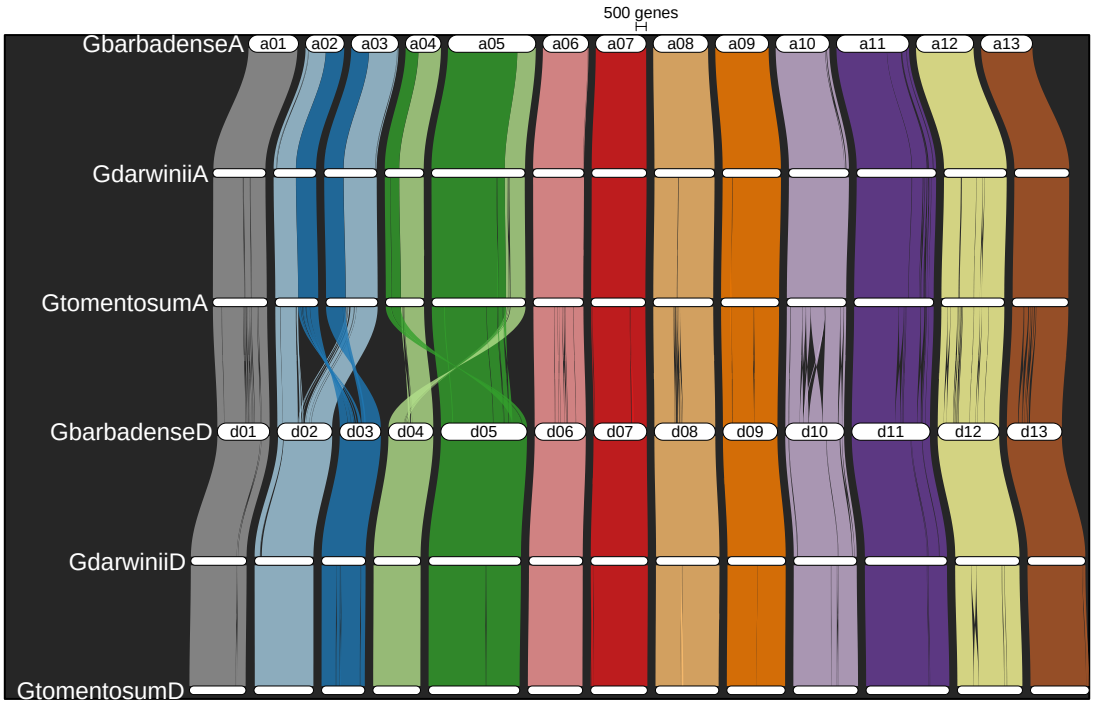
Table 2 | Summary of syntenic block inference between G. barbadense subgenomes. MCScanX_h was run for each subset of blast-like diamond2 (–more-sensitive) hits and the copy number of each non-overlapping 1kb genomic interval was tabulated from the start/end coordinates of the unique blocks from the collinearity file. The percent of 1kb intervals that are never found within a block (absent), found within exactly one block (single-copy) or in more than one block (multi-copy) are reported. Following Table 1, chromosomes 2-5 are excluded due to the presence of within-sub-genome translocations.

Riparian plot of all cottons, split by subgenomes

NOTE Illustrator edits: modified genome ids to species. Captilized subgenome specifier on chrs and removed subgenome ids on genomes. Increased scale bar to 5000 genes. Removed black background.

```
plot_riparianHits(
  gsParam = gspl, useOrder = T, useBlks = T,
  genomeIDs = rev(gspl$genomes$genomeIDs),
  labelTheseGenomes = c("GbarbadenseA", "GbarbadenseD"),
  refChrCols = c("grey60", RColorBrewer::brewer.pal(n = 12, name = "Paired")))
```

```
##  Reading hits ... Done!
##  Generating plot coordinates ... Done!
##  Rendering plot ...
```



Chromosomes scaled by gene rank order

```
## Done!
```

```
#-- make another to write to file
# pdf("/Users/lovell/Desktop/manuscripts/genespace_2022/figures/FigS2_cottonRiparian_sourcePlot.pdf", h
# plot_riparianHits(
#   gsParam = gspl, useOrder = T, useBlks = T,
#   genomeIDs = rev(gspl$genomes$genomeIDs),
#   labelTheseGenomes = c("GbarbadenseA", "GbarbadenseD"),
#   refChrCols = c("grey60", RColorBrewer::brewer.pal(n = 12, name = "Paired")))
# dev.off()
```