A quick guide to RTIGER

9/21/2020

Introduction

Accurate identification of meiotic crossing-over sites (COs) is essential for correct genotyping of recombining samples. RTIGER is a method for predicting genome-wide COs using allele-counts at pre-defined SNP marker positions. RTIGER trains a Hidden Markov Model (HMM) where genomic states (homozygous parent_1, homozygous parent_2 or heterozygous) correspond to the hidden state and the allele-counts as the observed variable. COs are identified as transitions in the HMM state.

To account for variation in the coverage of sequencing data, RTIGER uses Viterbi Path Algorithm and the `rigidity` parameter. This parameter defines the minimum number of SNP markers required to support a state-transition. This filters out low-confidence state-transitions, improving COs identification performance.

Installation

Pre-Requisites:

- R: Version > X_X_X
- Julia-1.0.5 (Which versions of Julia can be supported?): Julia needs to be installed and available in the environment[1]
- REQUIRED R LIBRARIES WOULD BE INSTALLED AUTOMATICALLY... RIGHT?

Preparing input data:

RTIGER uses the allele-count information at the SNP marker positions. The SNP markers correspond to differences between the two genotypes (i.e. parent_1 vs parent_2). RTIGER requires as input one allele-count file for each sample. The allele-count file should be in tab-separated value format, where each row corresponds to a SNP marker. The format of the file is described below:

Table 1: File Format for allele frequency file

Column	Field	Type	Description
1	SeqID	string	Chromosome ID
2	Pos	int (>=0)	Position of the SNP marker
3	RefA	char	Reference allele
4	RefC	int (>=0)	Number of reads with reference allele
5	AltA	char	Alternate allele
6	AltF	int (>=0)	Number of reads with alternate allele

The SNPs can be identified using any generic SNP identification pipeline[2].

SNPs in repetitive regions should be filtered out. Further, as crossing-over usually takes place in syntenic regions between the two genome, for best results, only SNPs in syntenic regions should be selected as markers.

[1] https://www.geeksforgeeks.org/how-to-setup-julia-path-to-environment-variable/?ref=lbp

[2] For example: https://www.ebi.ac.uk/sites/ebi.ac.uk/files/content.ebi.ac.uk/materials/2014/140217_AgriOmics/dan_bolser_snp_calling.pdf

If whole genome assemblies are present for both genomes, then this can be easily achieved using methods like SyRI[3].

NOTE 1: RTIGER assumes that all samples have similar sequencing coverage and, hence, similar distribution of the allele-count values. It does not check or normalise for sequencing coverage variation.

NOTE 2: Crossing-over resolution depends on sequenced marker density. Low sequencing coverage could result in few informative markers, which in turn could decrease resolution CO prediction.

NOTE 3: RTIGER is designed to be robust against individual outliers, however, the user should check for "bad" markers, i.e. marker positions that are prone to mismapping. These markers result in high allele-count at that position.

Using RTIGER

Setting up Julia environment:

RTIGER uses Julia to perform computationally intensive model training. All Julia packages that are used by RTIGER can be installed using using:

```
library(RTIGERJ)
setupJulia()
```

This step is **necessary** when using RTIGER for the first time, but can be skipped for later analysis as all required Julia packages would already be installed.

The Julia functions need to be loaded in the R environment using:

```
sourceJulia()
```

This step is required everytime when using RTIGER.

Creating input objects

The primary input for RTIGER is a data-frame termed `expDesign`. The first column of `expDesign` should have paths to allele-count files for all samples and the second column should have unique samples IDs.

```
# Get paths to example allele count files originating from a
# cross between Col-0 and Ler accession of the A.thaliana
file_paths = list.files(system.file("extdata",  package = "RTIGERJ"), full.names = TRUE)

# Get sample names
sampleIDs <- basename(file_paths)

# Create the expDesign object
expDesign = data.frame(files=file_paths, name=sampleIDs)

print(expDesign)
```

```
##                                                                   files
## 1 /home/goel/R/x86_64-pc-linux-gnu-library/3.5/RTIGERJ/extdata/sampleAA.txt
## 2 /home/goel/R/x86_64-pc-linux-gnu-library/3.5/RTIGERJ/extdata/sampleAC.txt
## 3  /home/goel/R/x86_64-pc-linux-gnu-library/3.5/RTIGERJ/extdata/sampleB.txt
```

[3]https://genomebiology.biomedcentral.com/articles/10.1186/s13059-019-1911-0

```
##           name
## 1 sampleAA.txt
## 2 sampleAC.txt
## 3  sampleB.txt
```

RTIGER also requires chromosome lengths for the parent_1. These need to be provided as a named vector where the values are chromosome lengths and the names are chromosome ids.

```
# Get chromosome lengths for the example data included in the package
chr_len <- RTIGERJ::ATseqlengths
names(chr_len) <- c('Chr1' , 'Chr2', 'Chr3', 'Chr4', 'Chr5')
print(chr_len)
```

```
##     Chr1     Chr2     Chr3     Chr4     Chr5
## 34964571 22037565 25499034 20862711 31270811
```

Finding crossing-over sites using RTIGER

RTIGER does model training, COs identification, per sample and summary plots creation using a single function.

```
myres = RTIGER(expDesign = expDesign,
               outputdir = "PATH/TO/OUTPUT/DIRECTORY",
               seqlengths = chr_len,
               rigidity = 200)
```

The `rigidity` parameter defines the required minimum number of continuous markers that together support a state change of the HMM model. Smaller `rigidity` values increase the sensitivity in detecting COs that are close to each other, but may result in false-positive CO identification because of variation in sequencing coverage. Larger `rigidity` values improve precision but COs that are close to each other might not be identified. **Users are supposed to test and adjust `rigidity` based on their specific experimental setup**.

RTIGER Output:

RTIGER identifies COs for each sample level and provides summary plots and statistics for each sample as well as for the entire population.

Per sample output

RTIGER creates a folder for each sample in the `outputdir`. This folder contains:

- `GenotypePlot.pdf`: Graphical representation of the allele-counts, allele-count ratio, and genotypes
- `GenotypeBreaks.bed`: BED file providing genomic regions corresponding to different genotypes
- `P1/P2/Het.bed`: BED files containing the markers present in genomic regions having genotype: homozygous parent 1, homozygous parent 2, or heterozygous, respectively
- `P1/P2.bw`: BigWig file containing the number of reads per marker position supporting parent 1 and parent 2, respectively
- `CountRatio.bw`: BigWig file containing the ratio of number of reads supporting parent 1 to number of reads supporting number 2 at the marker positions

Summary plots for the population (THE PLOTS AND THE DESCRIPTION HERE NEEDS MORE WORK)

RTIGER creates four summary plots after aggregating results for all samples.

- `COs-per-Chromosome.pdf`: Distribution of number of cross-overs per chromosome
- `CO-count-perSample.pdf`: Number of cross-overs in each sample
- `Goodness-Of-fit.pdf`: ****NEED DESCRIPTION FROM RAFA****
- `GenomicFrequencies.pdf`: Distribution of cross-overs along the length of chromosomes

Analysing backcrossed populations

Backcrossed populations are formed by crossing a hybrid organism with one of its parent. These populations are different from the populations based on outcrossing as only two genomic states are possible (homozygous for the backrossed parent and heterozygous for both parents). To identify COs in such population, set `nstates=2` in the RTIGER command.

```
myres = RTIGER(expDesign = expDesign,
               outputdir = "PATH/TO/OUTPUT/DIR",
               seqlengths = chr_len,
               rigidity = 200,
               nstates=2)
```

Cite:

`Citation`

Appendix:

Effect of varying `rigidity (R)` on crossover identification:

Session info

```
## R version 3.5.1 (2018-07-02)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Debian GNU/Linux 9 (stretch)
##
## Matrix products: default
## BLAS: /opt/share/software/packages/R-3.5.1-debian-9/lib64/R/lib/libRblas.so
## LAPACK: /opt/share/software/packages/R-3.5.1-debian-9/lib64/R/lib/libRlapack.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
```

```
##
## other attached packages:
## [1] knitr_1.23
##
## loaded via a namespace (and not attached):
##   [1] ProtGenerics_1.14.0         bitops_1.0-6
##   [3] matrixStats_0.54.0          bit64_0.9-7
##   [5] STAN_2.10.1                 RColorBrewer_1.1-2
##   [7] progress_1.2.2              httr_1.4.1
##   [9] GenomeInfoDb_1.18.2         tools_3.5.1
##  [11] backports_1.1.4             R6_2.4.0
##  [13] RTIGERJ_0.1.0               rpart_4.1-13
##  [15] Hmisc_4.3-0                 DBI_1.0.0
##  [17] BiocGenerics_0.28.0         Gviz_1.26.5
##  [19] lazyeval_0.2.2              colorspace_1.4-1
##  [21] nnet_7.3-12                 tidyselect_0.2.5
##  [23] gridExtra_2.3               prettyunits_1.0.2
##  [25] curl_4.2                    bit_1.1-14
##  [27] compiler_3.5.1              Biobase_2.42.0
##  [29] htmlTable_1.13.2            DelayedArray_0.8.0
##  [31] rtracklayer_1.42.2          scales_1.0.0
##  [33] checkmate_1.9.4             stringr_1.4.0
##  [35] digest_0.6.19               Rsamtools_1.34.1
##  [37] foreign_0.8-70              rmarkdown_1.14
##  [39] XVector_0.22.0              oompaBase_3.2.9
##  [41] dichromat_2.0-0             base64enc_0.1-3
##  [43] pkgconfig_2.0.2             htmltools_0.3.6
##  [45] oompaData_3.1.1             ensembldb_2.6.8
##  [47] BSgenome_1.50.0             highr_0.8
##  [49] htmlwidgets_1.5.1           rlang_0.4.2
##  [51] rstudioapi_0.10             RSQLite_2.1.2
##  [53] BiocParallel_1.16.6         acepack_1.4.1
##  [55] dplyr_0.8.1                 VariantAnnotation_1.28.13
##  [57] RCurl_1.95-4.12             magrittr_1.5
##  [59] GenomeInfoDbData_1.2.0      Formula_1.2-3
##  [61] Matrix_1.2-14               Rcpp_1.0.1
##  [63] munsell_0.5.0               S4Vectors_0.20.1
##  [65] stringi_1.4.3               yaml_2.2.0
##  [67] SummarizedExperiment_1.12.0 zlibbioc_1.28.0
##  [69] plyr_1.8.4                  grid_3.5.1
##  [71] blob_1.2.0                  parallel_3.5.1
##  [73] crayon_1.3.4                lattice_0.20-35
##  [75] Biostrings_2.50.2           splines_3.5.1
##  [77] GenomicFeatures_1.34.8      hms_0.5.2
##  [79] zeallot_0.1.0               pillar_1.4.1
##  [81] GenomicRanges_1.34.0        JuliaCall_0.17.1
##  [83] TailRank_3.2.1              reshape2_1.4.3
##  [85] biomaRt_2.38.0              stats4_3.5.1
##  [87] XML_3.98-1.20               glue_1.3.1
##  [89] evaluate_0.14               biovizBase_1.30.1
##  [91] latticeExtra_0.6-28         data.table_1.12.2
##  [93] vctrs_0.2.0                 gtable_0.3.0
##  [95] purrr_0.3.3                 assertthat_0.2.1
##  [97] ggplot2_3.2.0               xfun_0.8
```

```
##  [99] AnnotationFilter_1.6.0      e1071_1.7-3
## [101] Rsolnp_1.16                 class_7.3-14
## [103] survival_2.42-3             truncnorm_1.0-8
## [105] tibble_2.1.3                poilog_0.4
## [107] GenomicAlignments_1.18.1    AnnotationDbi_1.44.0
## [109] memoise_1.1.0               IRanges_2.16.0
## [111] cluster_2.0.7-1
```