Quick start guide for inveRsion package

Alejandro Cáceres, Suzanne Sindi, Ben Raphael,
Mario Cáceres and Juan Ramon González
acaceres@creal.cat

May 19, 2021

Contents

Abstract

This vignette shows the main functionalities of the `inveRsion` package. A small data set is used to illustrate the main steps for the detection of inversions with haplotype (phased) data. The package also handles genotype data, but as it is more computationally demanding, such analysis is presented in a complementary manual distributed with the software. Please refer to this for more details.

1 Introduction

Inversions of DNA segments are a source of genetic variability within the general population. While their frequency and phenotypic expression is not well understood, SNP-arrays offer a good opportunity to assess the presence of genetic inversions. As the availability of such arrays spread and phenotypic information is collected for a wide range of diseases, the ability to detect

inversions from nucleotide variations allows the assessment of their impact on human genetic variability.

Here, we present `inveRsion`, a bioinformatics tool that enables the detection of genetic inversions from SNP-array data. Previous methodologies have been developed for the detection of inversions from haplotype (phased) data. However, to our knowledge, there is no tool yet available to detect inversions from such data, and no methodology for genotype data. `inveRsion` allows both type of data to be analyzed. It is an optimized implementation of a previously published method for phased data, and generalizes it to genotype data.

The detection of inversions follows a simple work-flow. In this manual, we show the analysis of a small haplotype data set to demonstrate the main features of the package. `inveRsion` accepts text files with the haplotype information for each chromosome (row) and of each SNP (columns) labelled by their spatial coordinates (first row). Phased data from HapMap samples is easily converted to the required format; see Maual.pdf. For analysis of genotype data, the subject genotypes should be given at each row encoded as 0:homozygous, 1:heterozygous and 2:variant heterozygous. Please refer to the manual for more information on how to treat genotypes encoded in a `SNPmatrix` object.

In this work, we refer to a candidate break-point as a pair of contiguous SNPs. All candidate brake-points are flanked by haplotype blocks of n-SNPs, as required by the inversion model. The flanking blocks are then encoded for each candidate break-point. If we treat genotype data, local haplotyping is performed with `haplo.stats` to build the blocks. A window, or trial segment, is a segment of fixed length (window size) delimited by two candidate break-points. Finally, the inversion sequence is the actual segment on the data set that is inverted, for some chromosomes in the population, and that we probe with trial segments.

There are specialized functions to load genotype data onto an R session; `setUpGenoDatSNPmat` and `setUpGenoDatFile`. Both local haplotyping and block flanking is performed with `codeHaplo`. In the case of haplotype data, we can directly pass the file name to `codeHaplo`. The inversion model can be fitted to each possible pair of coded break-points. However, we only scan the whole chromosome with trial segments of fixed widow size (`scanInv`). From this scan a summary function `listInv` determines the full inversion sequence from overlapping window segments.

2 Installation

inveRsion is written on R (www.r-cran.org). The package can be downloaded from www.bioconductor.org. Alternatively, enter in the R prompt the commands

```
if (!requireNamespace("BiocManager", quietly=TRUE))
    install.packages("BiocManager")
biocList(inveRsion)
```

The library is loaded with the command

```
> library(inveRsion)
```

3 Haplotype Data

The data analyzed in this document is a reduced version of the simulation results obtained using invertFregene [1]. The full data set is fully analyzed elsewhere (manual.pdf), and comprises an inversion between 0.75Mb and 1.25Mb with population frequency of 0.4 (final value: 0.4135) in N=1000 individuals. We have selected a region of interest in (0.6Mb, 0.8Mb) for the left-break point and (1.1Mb,1.3Mb) for the right-break point. The haplotype data within these regions is in the file haplotypesROI.dat.

```
> hap <- system.file("extdata", "haplotypesROI.txt", package = "inveRsion")
```

The file contains on the first row the coordinates of the SNPs, and on subsequent rows the presence (1) or absence (0) of the corresponding variant allele for each chromosome in the population. Chromosomes in $2 * i$ and $2 * i - 1$ (i=1...N) rows correspond to the i-th individual.

The first step is to load the data on R and to code the haplotypes for all the candidate break-points. Candidate break-pints are regions limited by to contiguous SNPs. In the haplotype coding step we first flank each break-point by blockSize=5 number of SNPs at each side and then encode the binary sequence as a decimal integer.

```
> hapCode<-codeHaplo(blockSize=5,file=hap,
+                                          minAllele=0.3,saveRes=TRUE)

... labeling frequent allele=0, variant allele =1
.....................................................................
```

Data is filtered such that SNPs for candidate break-points have at least a minor allele frequency of `minAllele` (=0.3). The block flanking and coding is used for fitting the inversion model to any possible segment in the chromosome. The model tells us, for a given pair of break-points (left and right), which subjects are likely to have the defined segment inverted.

Although the potential number of trial segments is very large ($0(n!)$ -n number of SNPs), the inversion model still picks up an inversion signal if the trial segment is contained in the real inversion sequence. As a consequence, we can search the whole chromosome with trial segments of fixed length (window size), as probes for detecting the inversion. After such scan, the inversion sequence has to be reconstructed from all trial segments with high signal. This procedure massively reduces the search to order $0(n)$.

Note that, typically, flanking block sizes are much smaller that the trial segment sizes. However, if SNP density is low, the window size can be lower than the minimum blocking distance allowed ($2 * blockSize$). Trial segments under such conditions are discarded.

4 Inversion search

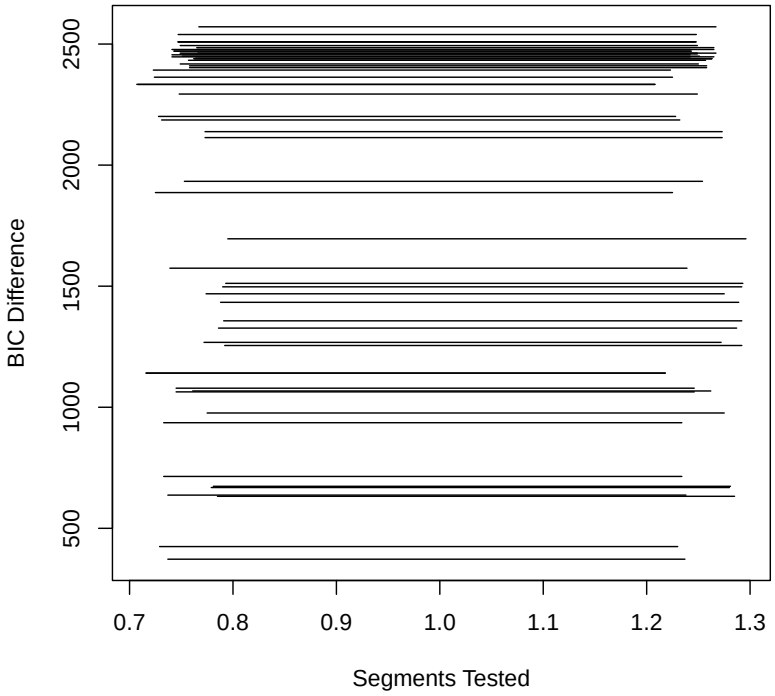
Inversions are scanned with trial segments of fixed window size (`window`=0.5). The inversion model is fitted to each of this segments an their significance measured by a Bayes information criterion (BIC).

```
> window<-0.5
> scanRes<-scanInv(hapCode,window=window,saveRes=TRUE)

.....................................................

> scanRes

-Showing object of class: scan-

Top 10 brake-points with highest Likelihood ratio:

            LeftBP         RightBP LogLike  Prob  BicDiff
1  0.76716-0.76726 1.26722-1.26734 2791.65 0.522 2571.209
2  0.75075-0.75124 1.25132-1.25160 2715.95 0.577 2449.896
3  0.74661-0.74661 1.24666-1.24687 2714.19 0.586 2508.956
4  0.74661-0.74690 1.24666-1.24687 2714.19 0.586 2508.956
5  0.74690-0.74721 1.24760-1.24881 2714.19 0.586 2508.956
6  0.74721-0.74776 1.24760-1.24881 2714.19 0.586 2539.362
```

```
7  0.74852-0.74890 1.24886-1.24911 2714.19 0.586 2493.753
8  0.74944-0.75066 1.24951-1.24951 2714.19 0.586 2417.739
9  0.75718-0.75772 1.25733-1.25803 2714.19 0.586 2432.942
10 0.75772-0.75789 1.25803-1.25821 2714.19 0.586 2410.138

others:
window: length window ( 0.5 ) for searching inversion segments
```

This is a `scan` object that can be readily plotted.

```
> plot(scanRes,which="b",thBic=-Inf)
```



It plots the BIC values for each trial segment, and also offers the option to display the log-likelihood ratio, or frequency of no inversion, setting `which=c("l","p")`, respectively. A threshold for segments with high BIC can be set with `thBic`.

When scanning the whole genome there will be some regions with overlapping significant segments. These regions are automatically identified as regions of interest where the function `listInv` re-evaluetes the model, at each

significant segment, to reconstruct the whole inverted sequence and extract the classification of inversion status of each chromosome.

For this data, there is only one overlapping region ($BIC > 0$) on which the model is re-run

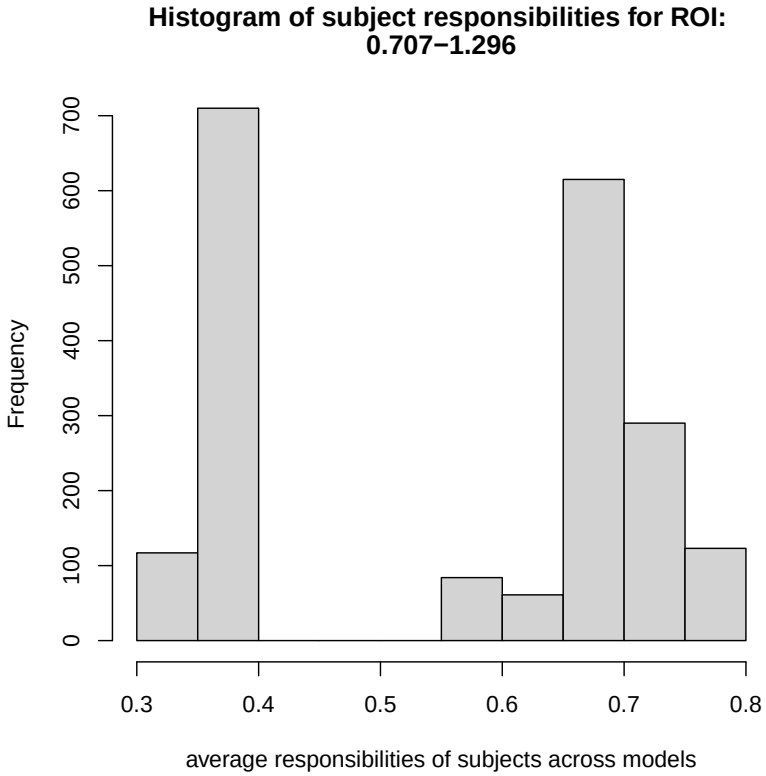
```
> invList<-listInv(scanRes,hapCode=hapCode,geno=FALSE,all=FALSE,thBic=0)

.....................................................

> invList

-Showing object of class: inversionList-

    LBPmin   LBPmax   RBPmin   RBPmax   MaxBic invFreq ModelNum
1 0.707386 0.795269 1.207698 1.295863 3270.649  0.5865       60
```

the output is the reconstructed inverted sequence with basic information for the inversion event, within the ROIs identified using the specified BIC threshold `thBic=0`. The result print contains the intervals for the left and right break-points, the maximum BIC found and the inversion frequency in the population. With the option $all = FALSE$, the sequence is reconstructed from the window segments used in the scan step. Otherwise, `listInv` fits the model for all the possible segments within the region of interest. Each segment used in the reconstruction returns a responsibility for each chromosome, stored but not visible in result print. An average of the responsibilities across segments gives the average responsibility per chromosome, where those > 0.5 are considered as inverted. Average responsibilities are retrieved from `invList` using

```
> r<-getClassif(invList,thBic=0, wROI=1,bin=FALSE,geno=FALSE)
> r[1:10]

     sub1      sub1      sub2      sub2      sub3      sub3      sub4      sub4
0.7441033 0.3690603 0.7422359 0.3756867 0.3579009 0.6733254 0.3576797 0.6932820
     sub5      sub5
0.3690603 0.3246950
```

$bin = TRUE$ returns a binary classification ($r_i > 0.5$) for each chromosome i (`bin=TRUE`), conserving the same ordering as the input data. The plot of `invList` gives the histogram of the average responsibilities per subject r, for a given ROI (`wROI=1`).

```
> plot(invList,wROI=1)
```



The chromosomes in the histogram with $r > 0.5$ are considered to carry the inversion.

Those responsibilities can also be used to compute the probabilities that each subject is homozygote, heterozygote or variant inverted heterozygote

```
> r<-getClassif(invList,thBic=0, wROI=1,geno=TRUE)
> r[1:10,]
```

```
      id hom het homInv
1   sub1   0   1      0
2   sub2   0   1      0
3   sub3   0   1      0
4   sub4   0   1      0
5   sub5   1   0      0
6   sub6   0   1      0
7   sub7   1   0      0
8   sub8   0   1      0
9   sub9   0   1      0
10 sub10   0   1      0
```

In the case of knowing the inversion status of each chromosome, you can assess the accuracy of classification produced with `inveRsion`. For this sample data, we have the true inversion status produced by the `invertFregene` simulation. Different accuracies can be achieved by varying the BIC threshold

```
> mem <- system.file("extdata", "mem.txt", package = "inveRsion")
> ac<-accBic(invList,classFile=mem,nsub=1000,npoints=10)

.........

> ac

An object of class "accuracy"
Slot "out":
         bicInt   prob     ac
 [1,]    0.0000 0.4135 1.0000
 [2,]  363.4055 0.4135 1.0000
 [3,]  726.8109 0.4135 1.0000
 [4,] 1090.2164 0.4135 1.0000
 [5,] 1453.6218 0.4135 1.0000
 [6,] 1817.0273 0.4135 1.0000
 [7,] 2180.4327 0.4135 1.0000
 [8,] 2543.8382 0.5055 0.0820
 [9,] 2907.2437 0.4080 0.2865
```

Note that `getClassif` depends on the BIC threshold, so the classification can be bettered by choosing an optimal $thBic$ resulting from the previous result.

```
> r<-getClassif(invList,thBic=2000, wROI=1,geno=TRUE)
> r[1:10,]

      id hom het homInv
1   sub1   0   1      0
2   sub2   0   1      0
3   sub3   0   1      0
4   sub4   0   1      0
5   sub5   1   0      0
6   sub6   0   1      0
7   sub7   1   0      0
8   sub8   0   1      0
9   sub9   0   1      0
10 sub10   0   1      0
```

5 Final remarks

This guide only covers the analysis of haplotype data. The analysis of genotypes follows the same streamline, although internally they perform different processes. Such analysis is performed with the same functions here presented (`scanInv` and `listInv`), but called with the option (`geno=TRUE`). The reading of different formats and how analyze genotype data is explained in the manual; displayed by typing `manual()`.

References

[1] O'Reilly PF, Coin LJM and Hoggart CJ, *invertFREGENE: software for simulating inversions in population genetic data*, Bioinformatics (2010) 26 (6): 838-840

[2] Sindi SS, Raphael BJ , *Identification and frequency estimation of inversion polymorphisms from haplotype data*, J Comput Biol. 2010 Mar;17(3):517-31

[3] Caceres A, Sindi SS, Raphael BJ, Caceres M, Gonzalez JR, *Identification of inversion polymorphisms from genotypes*, manuscript in preparation.