Finding Matched Random Controls for DNA Insertion Sites

Charles Berry

March 13, 2015

Contents

1 Introduction

An example is presented of the use of the `restrSiteUtils` package to create a mapping of restriction sites and then use that mapping to find matched random controls.

The data used are from

> Schröder, Astrid RW, et al. "HIV-1 integration in the human genome favors active genes and local hotspots." Cell 110.4 (2002): 521-529.

2 Create Restriction Data Mapping

Schröder et al used a cocktail of three restriction enzymes to recover integration sites. We suppose that the host DNA is broken at all such sites, so the closest one to the intgration site is what matters.

The three recognition sequences are `ACTAGT`, `CCTAGG`, and `GCTAGC`

The 'restrSiteUtils' library must be intalled in a directory whose path is in the `rlib` variable and that directory will be used to install a data package that maps the recognition sites in a way that is useful in finding distances to them. Adjust that path as needed. To use the system library, remove or comment out the line.

```
rlib <- "../Rlib"
.libPaths(rlib)
```

The 'BSgenome.Hsapiens.UCSC.hg18' BioConductor package also must be installed; it holds the genome sequence for the `hg18` freeze.

```
require( restrSiteUtils )

## Loading required package:  restrSiteUtils
## Loading required package:  GenomicRanges
## Loading required package:  BiocGenerics
## Loading required package:  parallel
##
## Attaching package:  'BiocGenerics'
##
## The following objects are masked from 'package:parallel':
##
##    clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##    clusterExport, clusterMap, parApply, parCapply, parLapply,
##    parLapplyLB, parRapply, parSapply, parSapplyLB
##
## The following object is masked from 'package:stats':
##
##    xtabs
##
## The following objects are masked from 'package:base':
##
##    Filter, Find, Map, Position, Reduce, anyDuplicated, append,
##    as.data.frame, as.vector, cbind, colnames, do.call,
##    duplicated, eval, evalq, get, intersect, is.unsorted, lapply,
##    mapply, match, mget, order, paste, pmax, pmax.int, pmin,
```

```
##    pmin.int, rank, rbind, rep.int, rownames, sapply, setdiff,
##    sort, table, tapply, union, unique, unlist, unsplit
##
## Loading required package:  S4Vectors
## Loading required package:  stats4
## Loading required package:  IRanges
## Loading required package:  GenomeInfoDb
```

```
rseq <- c("ACTAGT", "CCTAGG", "GCTAGC")
makeRestrDataPackage( tempdir(), rseq, "hg18",
                      "BSgenome.Hsapiens.UCSC.hg18",
                      "6-CUTTER",
                      installLib=TRUE)
```

```
## Creating DESCRIPTION ...
```

3 Read in Schroder et al Data

The `csv` file has the integration ste positions (actually the position of the base preceeding the point of insertion).

After reading the data, it is rendered as a 'GRanges' object.

```
sites.df <- read.csv("sites.df.csv")

sites.gr <-
    with(sites.df,
         GRanges(seqnames,
                 IRanges( start=start, width=2),
                 strand=strand, Sequence=Sequence,
                 BID=BID))

colnames(sites.df)
```

```
##  [1] "X"                 "seqnames"          "start"
##  [4] "end"               "width"             "strand"
##  [7] "type"              "Sequence"          "setName"
## [10] "BID"               "Restriction.enzyme" "Origin.of.data.set"
```

4 Get Distances From Integration Sites to Restriction Sites

The function `distRL` finds the distances using the `width.GR` object provided by the `restrEnz.Hsapiens.UCSC.hg18.RENZ.6.CUTTER` data package.

```
require( restrEnz.Hsapiens.UCSC.hg18.RENZ.6.CUTTER )

## Loading required package:  restrEnz.Hsapiens.UCSC.hg18.RENZ.6.CUTTER

site.dist <- distRL( sites.gr, width.GR )
```

As a sanity check, the ditribution of distances in the reverse direction (i.e., away from the recognition site that recovered the insertion) should be a lot different.

```
sites.reverse <- sites.gr
strand( sites.reverse ) <-
    ifelse( strand( sites.gr ) == "+", "-", "+")
site.dist.reverse <- distRL( sites.reverse, width.GR )
```

And it is. The mean is a good deal larger. And there is one site (`NA`) that had no recognition site flanking it in *that* direction before a sequence gap or chromosome was found.

```
summary( site.dist )

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    14.0   202.0   385.0   484.7   638.0  4410.0

summary( site.dist.reverse )

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##     1.0   733.8  1739.0  2476.0  3311.0 17480.0       1
```

5 Create Matched Random Controls aka MRCs

The function `sampleDist` creates controls. Here there are 3 such for every integration site. `mrc.gr` is a GRanges object that holds the locations of the MRCs.

```
mrc.gr <- sampleDist(3, site.dist, width.GR,
                     parentNames=sites.gr$Sequence)

mrc.dist <- distRL( mrc.gr, width.GR)

mrc.dist.matrix <-
    do.call(rbind, tapply(mrc.dist, mrc.gr$parentNames, c))
```

As a check the distances for these are found and compared to their parent sites. They all should be (and do) have distances equal to their parents'.

```
## Be sure rows are in same order as sites.gr$Sequence or the
## following check will not work:

all(length(sites.gr$Sequence),
    sites.gr$Sequence==rownames(mrc.dist.matrix))

## [1] TRUE

## Each MRC is found to have the same distance as its parent:

all(site.dist==mrc.dist.matrix)

## [1] TRUE
```

6 Combine MRCs and Sites

To do anything useful, the MRCs and sites must be used together. Here a `GRanges` instance is created that uses `mcols(combo.gr)$Sequence` to identify the parent and its MRCs and `type` to indicate whether the element is a site ("insertion") or MRC ("match").

```
mcols(mrc.gr) <- DataFrame(Sequence=mrc.gr$parentNames)

mcols(sites.gr)$BID <- NULL

combo.gr <- c(sites.gr,mrc.gr)

## use `type' to distinguish MRCs and real sites

mcols(combo.gr)$type <-
    rep(c("insertion","match"),c(length(sites.gr),length(mrc.gr)))
```

7 Check Flanking Sequence of MRCs vs Sites

As an illustration of use, here the favored local sequence is determined.

The sequence motif `GTNAC` favored by HIV is shown here. The table gives the base 2 logarithm of the relative proportion of bases for integration sites minus that for MRCs. Thus, values greater than 0.0 show favored bases. Positions 11–15 are enclosed by the sites of attack.

```
combo.flank.20 <-
    shift( flank( combo.gr, 10L, start=TRUE, both=TRUE),
          ifelse( strand( combo.gr ) == "+", 1L, -1L))

seq.flank.20 <-
      getSeq( Hsapiens, combo.flank.20, as.character=TRUE)

t( sapply( 1:20, function(x) {
    y <- substring( seq.flank.20, x, x)
    tab <- prop.table( table( y, combo.gr$type ), 2)
    round( log( tab[,1] / tab[,2], 2 ), 2 )}))
```

```
##            A     C     G     T
##  [1,]  0.25 -0.20 -0.39  0.09
##  [2,] -0.14 -0.11 -0.23  0.33
##  [3,] -0.22  0.30 -0.14  0.07
##  [4,]  0.18 -0.38 -0.06  0.07
##  [5,] -0.08 -0.10  0.29 -0.08
##  [6,] -0.15  0.11  0.20 -0.07
##  [7,] -0.30  0.31 -0.23  0.19
##  [8,] -0.58 -0.68  0.16  0.59
##  [9,] -0.19 -0.92  0.34  0.33
## [10,]  0.19 -0.53  0.64 -0.55
## [11,] -0.55  0.57  1.02 -1.74
## [12,] -0.15 -0.56 -0.78  0.68
## [13,]  0.10 -0.09 -0.30  0.14
## [14,]  0.58 -0.64 -0.64  0.02
## [15,] -1.74  1.01  0.40 -0.25
## [16,] -0.19  0.72 -0.47 -0.14
## [17,]  0.07  0.31 -0.86  0.12
## [18,]  0.57  0.09 -0.49 -0.64
## [19,]  0.08 -0.13  0.08 -0.05
## [20,]  0.01 -0.11 -0.01  0.07
```

8 sessionInfo

The document required 225 seconds to run. Here is the setup used.

```
toLatex(sessionInfo())
```

- R version 3.1.2 (2014-10-31), `x86_64-apple-darwin10.8.0`
- Locale: `C`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils
- Other packages: BSgenome 1.34.1, BSgenome.Hsapiens.UCSC.hg18 1.3.1000, BiocGenerics 0.12.1, Biostrings 2.34.1, GenomeInfoDb 1.2.4, GenomicRanges 1.18.4, IRanges 2.0.1, S4Vectors 0.4.0, XVector 0.6.0, restrEnz.Hsapiens.UCSC.hg18.RENZ.6.CUTTER 1.0, restrSiteUtils 1.2.5, rtracklayer 1.26.2
- Loaded via a namespace (and not attached): BBmisc 1.9, BatchJobs 1.5, BiocParallel 1.0.0, DBI 0.3.1, GenomicAlignments 1.2.1, RCurl 1.95-4.5, RSQLite 1.0.0, Rsamtools 1.18.2, XML 3.98-1.1, base64enc 0.1-2, bitops 1.0-6, brew 1.0-6, checkmate 1.5.1, codetools 0.2-10, digest 0.6.8, evaluate 0.5.5, fail 1.2, foreach 1.4.2, formatR 1.0, highr 0.4, iterators 1.0.7, knitr 1.9, sendmailR 1.2-1, stringr 0.6.2, tools 3.1.2, zlibbioc 1.12.0

9 License and Acknowledgement

The development of code and this document was supported by the National Institutes of Health (2R01 AI052845 and 5R01 AI082020).