IlluminaHumanMethylation450kprobe

September 24, 2013

`IlluminaHumanMethylation450kprobe`
Probe sequences for microarrays of type IlluminaHumanMethylation450

Description

Reannotation resource for Illumina HumanMethylation450 chips

Usage

```
data(IlluminaHumanMethylation450kprobe)
```

Format

A data frame with 485577 rows and 10 columns, as follows.

Probe_ID	character	Illumina probe ID
chr	factor	Chromosome of probe target in hg19
strand	factor	Best strand match to hg19/GRCh37
start	integer	Start coordinate of target in hg19
end	integer	End coordinate of target in hg19
site	character	Interrogated cytosine in hg19
probe.sequence	character	Probe (allele A) sequence
source.sequence	character	Designed target sequence
forward.genomic.sequence	character	Closest match in hg19
CpGs	integer	CpG sites (CpH/rs probes have 0)

Interrogated di/trinucleotides span (site, site+(SNP=0,CpG=1,CpH=2)). CpH probe coordinates were liftOver()ed from hg18 to hg19 then aligned.

Source

The probe sequence data was obtained from ftp://ftp.illumina.com. Data was extracted from HumanMethylation450_15017482_v.1.2.csv.

Examples

```
##
## UPDATE, April 1st, 2013:
##
## Please consider this package obsolete.
##
## The mapToGenome() function in minfi and methylumi,
## the FDb.InfiniumMethylation.hg18/hg19 packages,
## or the fetchGEOMethylationDataset() function all
## eliminate a great deal of the boilerplate below
## and work well with the VariantAnnotation package to
## locate loci of interest relative to arbitrary features,
## not just genes, transcripts, exons, or promoters.
##
#
# The following is for posterity.
# Known bugs have been fixed and builds updated.
# That does not, however, change the fact that it is deprecated.
#
library(GenomicRanges)
library(IlluminaHumanMethylation450kprobe)
data(IlluminaHumanMethylation450kprobe)
## head(IlluminaHumanMethylation450kprobe, 3)
## summary(IlluminaHumanMethylation450kprobe)
chs = levels(IlluminaHumanMethylation450kprobe$chr)
names(chs) = paste('chr',chs,sep='')
CpGs.450k = sort(keepSeqlevels(with(IlluminaHumanMethylation450kprobe,
                               GRanges(paste('chr',chr,sep=''),
                                 IRanges(start=site, width=2, names=Probe_ID),
                                 strand=strand)
                               ), paste0('chr', c(1:22, 'X', 'Y'))))

# verify the number of CpG sites in each probe:
library(Biostrings)
hm450 = with(IlluminaHumanMethylation450kprobe,
             DNAStringSet(forward.genomic.sequence))

## Verify CpG vs. CpH vs. SNP probes
##
head(dinucleotideFrequency(hm450)[,'CG'])
## [1] 3 2 1 1 3 1
tail(dinucleotideFrequency(hm450)[,'CG'])
## [1] 0 0 0 0 0 0 ...CpH probes
## (todo: add rsID probes; as of 8/2012, done in FDb.InfiniumMethylation.hg19)

## find all SNPs (regardless of frequency) at CpG sites using GenomicRanges
## library(parallel)
```

```
## library(SNPlocs.Hsapiens.dbSNP.20120608) ## dbSNP build 137

## CpG.snps.by.chr = mclapply(chs, function(ch) { # {{{ uses GenomicRanges
##   snps = getSNPlocs(paste('ch', ch, sep=''), as.GRanges=TRUE)
##   seqlevels(snps) <- gsub('ch','chr',seqlevels(snps))
##   names(snps) = paste('rs',elementMetadata(snps)$RefSNP_id,sep='')
##   message(paste('Scanning for CpG SNPs in probes on chromosome', ch))
##   overlapping = findOverlaps(CpGs.450k, snps)@matchMatrix
##   results = data.frame(
##     Probe_ID=as.character(names(CpGs.450k)[overlapping[,1]]),
##     rsID=as.character(names(snps)[overlapping[,2]])
##   )
##   return(results)
## }) # }}}
## SNPs = do.call(rbind, CpG.snps.by.chr)
##
## Obnoxious side effect of do.call(rbind)
## SNPs$rsID = levels(SNPs$rsID)[SNPs$rsID]
## SNPs$Probe_ID = levels(SNPs$Probe_ID)[SNPs$Probe_ID]
##
# For 27k array comparisons you could do...
# SNPs$hm27 = unlist(mget(SNPs$Probe_ID, IlluminaHumanMethylation450kMETHYL27))
#
## However, it is easier to use the FDb.InfiniumMethylation.hg18/hg19 and
## FDb.UCSC.commonSNP... packages, which map probes from both platforms to
## a genome build, and common SNPs from a dbSNP build to an assembly (resp).

# how many probes have SNPs at CpGs?
# message(nrow(SNPs))
# IlluminaHumanMethylation450kprobe$CpG.SNP = FALSE
# probe.SNPs = which(is.element(IlluminaHumanMethylation450kprobe$Probe_ID,
#                               SNPs$Probe_ID))
# IlluminaHumanMethylation450kprobe$CpG.SNP[probe.SNPs] = TRUE
#
# find repeats crossing CpG sites using IRanges
# library(BSgenome.Hsapiens.UCSC.hg19)
# CpG.rpts.by.chr = mclapply(chs, function(ch) { # {{{ uses IRanges
#   chr = Hsapiens[[paste('chr',ch,sep='')]]
#  rpts = union( masks(chr)$RM, masks(chr)$TRF )
#  probes = which(seqnames(CpGs.450k)==paste('chr',ch,sep=''))
#  # note how we have to use RangedData instead!!
#  CpGs.chr = ranges(CpGs.450k[probes])
#  message(paste('Scanning for repeats in CpG sites on chromosome', ch))
#  overlapping = findOverlaps(CpGs.chr, rpts)@matchMatrix
#  results = data.frame(
#    Probe_ID=as.character(names(CpGs.chr)[overlapping@matchMatrix[,1]]),
#    repeatID='RM/TRF'
#  )
#  return(results)
# }) # }}}
# RPTs = do.call(rbind, CpG.rpts.by.chr)

# how many probes have repeats at CpGs?
```

```
# message(nrow(RPTs))
# IlluminaHumanMethylation450kprobe$CpG.repeat = FALSE
# IlluminaHumanMethylation450kprobe$CpG.repeat[RPTs$Probe_ID] = TRUE

# how many have both?
# with(IlluminaHumanMethylation450kprobe,
#      sum(CpG.repeat & CpG.SNP))

# how many have either?
# with(IlluminaHumanMethylation450kprobe,
#     sum(CpG.repeat | CpG.SNP))

# We could change the above to find all SNPs and RPTs within probe targets:
# probes.450k = with(IlluminaHumanMethylation450kprobe,
#                    GRanges(paste('chr',chr,sep=''),
#                            IRanges(start=start, width=50, names=Probe_ID),
#                            strand=strand))
#
# Swap 'probes.450k'for 'CpGs.450k' in the previous lapply() loops to run.
# nb. If we want to look e.g. 10bp from the CpG site, then stranding matters.
#
## Another possibility is to look for only COMMON SNPs, i.e., MAF > 0.01:
##
## library(FDb.UCSC.snp137common.hg19)
## snp137common.hg19 <- features(FDb.UCSC.snp137common.hg19) # may take a while
## probes.with.commonSNPs = subsetByOverlaps(probes.450k, snp137common.hg19)
##

# find the nearest TSS and its corresponding EntrezGene ID
##
CpGs.unstranded = CpGs.450k
strand(CpGs.unstranded) = '*'
library(GenomicFeatures)
# refgene.TxDb = makeTranscriptDbFromUCSC('refGene', genome='hg19')
##
## NOTE!  This is much more easily done with VariantAnnotation and Homo.sapiens:
##
##
##
##
##
require(Homo.sapiens)
require(TxDb.Hsapiens.UCSC.hg19.knownGene) ## just in case
txs = sort(unlist2(transcriptsBy(TxDb.Hsapiens.UCSC.hg19.knownGene, 'gene')))
txs = keepSeqlevels(txs, seqlevels(CpGs.unstranded))
txs = unique(resize(txs, width=1, fix='start'))

## map UCSC KG IDs to Hugo gene names, THE RIGHT WAY (cf. Marc Carlson)
UCSCtoHugo = select(Homo.sapiens,
                    cols=c("SYMBOL","TXNAME"),
                    keys=mcols(txs)$tx_name,
                    keytype="TXNAME")
names(txs) = UCSCtoHugo$SYMBOL[match(mcols(txs)$tx_name, UCSCtoHugo$TXNAME)]
```

```
# nearest (downstream) forward TSS
TSS.forward = txs[which(strand(txs) == '+')]
strand(CpGs.unstranded) = '+'
nearest.fwd = precede(CpGs.unstranded, TSS.forward)
nearest.fwd.eg = nearest.fwd # to keep dimensions right
notfound.fwd = sum(is.na(nearest.fwd)) # number missed
dnf = which(is.na(nearest.fwd)) # track for later

# name of transcript or gene for nearest downstream forward TSS
nearest.fwd.eg[-dnf] = as.character(names(TSS.forward)[nearest.fwd[-dnf]])

# distance to the downstream TSS for that transcript
TSSs.fwd = start(TSS.forward[nearest.fwd[-dnf]])
distToTSS.fwd = nearest.fwd # to keep dimensions right
distToTSS.fwd[-dnf] = start(CpGs.unstranded)[-dnf] - TSSs.fwd
# note that these are NEGATIVE, which is correct


# nearest (downstream) reverse TSS
TSS.reverse = txs[which(strand(txs) == '-')]
strand(CpGs.unstranded) = '-'
nearest.rev = precede(CpGs.unstranded, TSS.reverse)
nearest.rev.eg = nearest.rev # to keep dimensions right
notfound.rev = sum(is.na(nearest.rev)) # number missed
dnf = which(is.na(nearest.rev)) # track for later

# name of transcript or gene for nearest downstream reverse TSS
nearest.rev.eg[-dnf] = as.character(names(TSS.reverse)[nearest.rev[-dnf]])

# distance to the downstream TSS for that transcript
TSSs.rev = start(TSS.reverse[nearest.rev[-dnf]])
distToTSS.rev = nearest.rev # to keep dimensions right
distToTSS.rev[-dnf] = start(CpGs.unstranded)[-dnf] - TSSs.rev
# these are POSITIVE: we are walking up the opposite strand.


## tie up a loose end...
strand(CpGs.unstranded) == '*'

# tabulate and link these together for the annotation package:
IlluminaHumanMethylation450kprobe$fwd.dist <- distToTSS.fwd
IlluminaHumanMethylation450kprobe$fwd.gene_id <- nearest.fwd.eg
IlluminaHumanMethylation450kprobe$rev.dist <- distToTSS.rev
IlluminaHumanMethylation450kprobe$rev.gene_id <- nearest.rev.eg

FWD.CLOSER = with(IlluminaHumanMethylation450kprobe,
                  union( which( abs(fwd.dist) < abs(rev.dist) ),
                         which( is.na(rev.dist) ) ) )
REV.CLOSER = with(IlluminaHumanMethylation450kprobe,
                  union( which( abs(fwd.dist) > abs(rev.dist) ),
                         which( is.na(fwd.dist) ) ) )
```

```
IlluminaHumanMethylation450kprobe$DISTTOTSS = pmin(abs(IlluminaHumanMethylation450kprobe$fwd.dist), abs(Illumin

IlluminaHumanMethylation450kprobe$ENTREZ = NA
IlluminaHumanMethylation450kprobe$ENTREZ[FWD.CLOSER] =
  IlluminaHumanMethylation450kprobe$fwd.gene_id[FWD.CLOSER]
IlluminaHumanMethylation450kprobe$ENTREZ[REV.CLOSER] =
  IlluminaHumanMethylation450kprobe$rev.gene_id[REV.CLOSER]

# note that a few (99) ENTREZ numbers do not appear because
# a probe is equidistant to upstream & downstream TSS
#
# (patch from Kim Siegmund)
#
table(is.na(IlluminaHumanMethylation450kprobe$ENTREZ))
missingNames = which(is.na(IlluminaHumanMethylation450kprobe$ENTREZ))
IlluminaHumanMethylation450kprobe$ENTREZ[ missingNames ] = paste0(
  IlluminaHumanMethylation450kprobe$fwd.gene_id[ missingNames ], ' forward, ',
  IlluminaHumanMethylation450kprobe$rev.gene_id[ missingNames ], ' reverse'
)
table(is.na(IlluminaHumanMethylation450kprobe$ENTREZ))
## NAs eliminated

# Another use case for GRanges that is difficult with the manifest:
# finding the observed/expected CpG density in a window around each locus
#
# library(BSgenome.Hsapiens.UCSC.hg19)
# window.width = 500 # could use larger or smaller, eg. Saxonov (2006) used 3000
# require(parallel)
# oecg.by.chr = mclapply(chs, function(ch) {
#   probes = which(IlluminaHumanMethylation450kprobe$chr == ch)
#   probecpgs = with(IlluminaHumanMethylation450kprobe[probes,],
#                    IRanges(start=site, width=2, names=Probe_ID))
#   cpgwindows = resize(probecpgs, fix="center", width=window.width)
#   chr = Hsapiens[[paste('chr',ch,sep='')]]
#   active(masks(chr)) = FALSE
#   chr.seqs = Views(chr, cpgwindows)
#   ocg = dinucleotideFrequency(chr.seqs, as.prob=T)[,'CG']
#   c.g = alphabetFrequency(chr.seqs, as.prob=T,baseOnly=T)
#   ecg = c.g[,'C'] * c.g[,'G']
#   oecg <- ocg/ecg
#   names(oecg) = rownames(IlluminaHumanMethylation450kprobe)[probes]
#   return(oecg)
# })
# IlluminaHumanMethylation450kprobe$oecg = unlist2(oecg.by.chr)
```

Index