pdProbeRemap

Introduction

The package provides functionality to build new microarray platform annotations, based on the user-specified genomic references. Newly built annotations can be returned either as an annotation package, compatible with `oligo` for further data processing, or as a corresponding annotation object.

This document shows examples on how to build such annotations for several Affymetrix platforms.

Required files

Building new annotation for Affymetrix Gene ST and Affymetrix Expression arrays depends on the following files and software.

- Affymetrix patform annotation files:

CDF, probe sequence file (TAB-delimited), example CEL file for Affymetrix Expression arrays or

PGF, CLF, MPS, TRANSCRIPT.CSV and PROBESET.CSV files for Affymetrix Gene ST arrays

Annotation files can be downloaded from the Affymetrix oficial webpage.

- Reference genome annotation: GTF
- STAR aligner

General strategy

Generation of a new annotation for an Affymetrix microarray involves alignment of the probe sequences from that array, which can not be done on a local machine of a user. For that reason annotation building procedure is split in three steps:

- processing the original annotation and generating FASTA files (probe sequences) to input in the aligner
- alignment of the probe sequences
- processing the alignments and building the new annotation

Set up

Load the package

```
library("pdProbeRemap")
```

Set up an aligner

We used STAR aligner, which can be found at `https://github.com/alexdobin/STAR` Note that, most likely the alignments will need to be run on a remote cluster.

Building new annotation

Step 1: processing original annotation and generating FASTA files

```
#evaluation worked
dir1 = "/Users/milchevs/ownCloud/ProbeRemapSteps14/inst/extdata/mogene2_affyLibFiles_pgfBased/"
outDir = "/Users/milchevs/ownCloud/ProbeRemapSteps14/OutPutDir/"
dir.create(outDir)

annotation2fasta(originalAnnotationDir = dir1,
                 organism = "Mouse", species = "Mus musculus",
                 author = "Vladislava Milchevskaya",
                 email = "milchv@gmail.com",
                 outputDir = outDir,
                 reverse = FALSE)
```

Step 2: alignment of the probe sequences

After `.FASTA` file with probe sequences of the array is generated, it is used to run the alignment to the reference genome with STAR aligner. Depending on the organist and array type, bash command for STAR needs different option set-up, which is provided by `star.command.make` function. Note that the resulting STAR command migh need to be executed not on the user local machine, but on a remote cluster.

We suggest one runs the STAR command in a special output directory, as a number alignment files will be generated.

```
# evaluation worked
path_to_STAR = "/path/to/STAR"
sequence_fasta = "/path/to/sequence.fasta"
genomeDir = "/path/to/reference/genome/"
organism = "Human"
outfile = "output_file."

star.command.make(path_to_STAR, sequence_fasta, genomeDir, organism, outfile)
```

Step 3: processing alignments and building new annotation

When alignments of the probe sequences are obtained, they can be processed, and the new annotation package can be generated.

Build reference genome annotation data.frame

In order to process alignments, one will have to parse the reference genome annotation (GTF, the same one that was used to run alignments). If you are going to build new annotation packages for several different arrays, but same species, note that reference genome `data.frame` needs to be generated only once.

```
# evaluation worked
outputDirAnotationDro <- "/Users/milchevs/Downloads/NewPackages/DroGene1_1"

require("refGenome")
beg <- ensemblGenome()
basedir(beg) <- "/Users/milchevs/databases/sequences/Ensembl/annotation/Dmel/"
ens_gtf <- "dmel-all-r6.09.gtf" # adjusted file, excessive white spaces deleted
read.gtf(beg, ens_gtf)
#tableAttributeTypes(beg)
#moveAttributes(beg,c("gene_name","transcript_id","transcript_name","exon_number"))

ANN_all.list <- (as.list(beg@ev))
ANN_all <- ANN_all.list$gtf
head(ANN_all)
save(ANN_all, file = file.path(outputDirAnotationDro, paste0(ens_gtf, ".AnnotationDataFrame.RData")))
```

Read In alignments, generate the new package

When alignments are generated and the corresponding reference genome annotation is proccessed into a `data.frame` (see above), one can build a new annotation package based of these alignments.

```
alignmentDir = "/Users/milchevs/Documents/Biology/PROJECTS/Paul_Bertone/ReANNot/MoGene2/"
load("/Users/milchevs/Documents/Biology/PROJECTS/Paul_Bertone/ReANNot/Annotations/AnnotationDataFrame.Mu
dir1 = "/Users/milchevs/ownCloud/ProbeRemapSteps14/inst/extdata/mogene2_affyLibFiles_pgfBased/"
outDir = "/Users/milchevs/ownCloud/ProbeRemapSteps14/OutPutDir/"
o1 = makeOriginalPackageObject(originalAnnotationDir = dir1,
                               organism = "Mouse",
                               species = "Mus musculus",
                               outputDir = outDir )

# NEW_ParsedData <- alignments2parsedData <- function(alignment.dir,
#                                    Annotation,
#                                    outputDir,
#                                    level,
#                                    min_probe_number,
#                                    package_seed)


library("pdProbeRemap")
data(Alignments_class_example, package = "pdProbeRemap")
data(Annotation_example)
data(seed_example)
alignmentDir <-
 unlist(strsplit(dir(system.file("extdata",package="pdProbeRemap"),
                  pattern="example_drosophila.Aligned.out.sam",
                  full.names=TRUE), split = "example_drosophila.Aligned.out.sam"))[1]


makeNewAnnotationPackage(alignment.dir = alignmentDir,
                         Annotation = Annotation_example,
                         outputDir = outDir,
                         #outputDir = ".",
                         level = "gene",
                         min_probe_number = 1,
                         pkgNameSUFFIX = ".example")
```

generation of the new annotation data for PGF-based affy annotations

```
library("pdProbeRemap")

alignmentDir = "/Users/milchevs/Documents/Biology/PROJECTS/Paul_Bertone/ReANNot/MoGene2/"
load("/Users/milchevs/Documents/Biology/PROJECTS/Paul_Bertone/ReANNot/Annotations/AnnotationDataFrame.Mu
dir1 = "/Users/milchevs/ownCloud/ProbeRemapSteps14/inst/extdata/mogene2_affyLibFiles_pgfBased/"
outDir = "/Users/milchevs/ownCloud/ProbeRemapSteps14/OutPutDir/Probe_Sequences_Fasta/"
```

```
o1 = pdProbeRemap::makeOriginalPackageObject(originalAnnotationDir = dir1,
                                             organism = "Mouse",
                                             species = "Mus musculus",
                                             outputDir = outDir )
```

```
NEW_PD = makeNewAnnotationPackage(alignment.dir = alignmentDir,
                                  Annotation = ANN_MUS_all,
                                  outputDir = outDir,
                                  level = "gene",
                                  min_probe_number = 1,
                                  #package_seed = o1,
                                  quiet = FALSE,
                                  pkgNameSUFFIX = ".example")
#
```

Annotation Packages Used with Oligo

When analyzed with `oligo`, Affymetrix platforms come with `pd.<chipname>` annotation packages.

```
CelFilesPath = c("/path/to/celfile1.CEL", "/path/to/celfile2.CEL")
celfiles = read.celfiles(filenames=CelFilesPath,
                              pkgname = "pd.mogene.2.0.st")
```

If one does not set the parameter `pkgname` manually, `oligo` will suggest an original annotation package, for instanse `"pd.mogene.2.0.st"`.

In case one wishes to use another annotation, the package must be installed, and the package name must be set as the value of `pkgname` parameter:

```
install.packages("/path/to/new/pd/annotation/package/pd.NEWPACKAGE",
                 repos=NULL, type="source")
library("pd.NEWPACKAGE")

CelFilesPath = c("/path/to/celfile1.CEL", "/path/to/celfile2.CEL")
celfiles = read.celfiles(filenames=CelFilesPath,
                              pkgname = "pd.NEWPACKAGE")
```

One complete example of new annotation package building

```
library("pdProbeRemap")

outDir <- "/Users/milchevs/ownCloud/ProbeRemapStep3/R/testdata/DroGene1_1"
OriginalAnnotDir <-  "/Users/milchevs/Downloads/CD_DrosGenome1\ 2/Full/DrosGenome1/LibFiles/"

FastaFilePath_dro <- annotation2fasta(originalAnnotationDir = OriginalAnnotDir,
                                      organism = "Dmel",
                                      species = "Drosophila melanogaster",
                                      outputDir = outDir,
                                      author = "Name Surname",
                                      email = "e@mail",
                                      reverse = FALSE )
```

```
######### step 2 ###########
#        alignment         #
star_command_plainprobes =
  star.command.make(sequence_fasta = "/var/local/milchevs/ALIGNED/DroGene1/FASTA/DrosGenome1_probe_seque
                    genomeDir = "/var/local/milchevs/databases/sequences/Ensembl/star/Dmel/",
                    species = "Dmel",
                    outfile = "/var/local/milchevs/ALIGNED/DroGene1/DrosGenome1.",
                    runThreadN = 6)

######### step 3 ###########
load("/Users/milchevs/Downloads/NewPackages/DroGene1_1/dmel-all-r6.09.gtf.AnnotationDataFrame.RData") #
alignmentDir = file.path("/Users/milchevs/ownCloud/ProbeRemapStep3/R/testdata/DroGene1_1/", "Alignments

makeNewAnnotationPackage(alignment.dir = alignmentDir,
                         Annotation = ANN_all,
                         outputDir = outDir,
                         level = "gene",
                         min_probe_number = 1,
                         quiet=FALSE,
                         pkgNameSUFFIX = ".NewAnnotation"
)

install.packages("/Users/milchevs/ownCloud/ProbeRemapStep3/R/testdata/DroGene1_1/pd.drosgenome1.NewAnno
```

drafts

removed from introduction

This satisfies the need to have annotations based on the most recent genomic reference, as well as the need to use same references while comparing data from different arrays, or microarray and RNA-seq data. Additionally, the package allowes to build user-specific annotations for a particular version of the reference genome.

`oligo` package from Bioconductor provides an extended functionality for the analysis of the microarray data measured with Affymetrix platforms. To process raw array data (".CEL" files) for each type of Affymetrix array oligo downloads and installs the corresponding annotation package, which is based on the original platform annotation information provided by Affymetrix.

However, in most cases these annotations were generated at the time the particular platform was released, and were produced based on the current information of the reference genome.

Building of the new annotation is split in two steps because usually alignments of the probe sequences can not be executed on users local machines and require acces to clusters with larger available memory.

Thus, in the first step there is written a FASTA file with probe sequences corresponding to the Affymetrix platform subject to re-annotation, then alignments of the probe sequences performed remotely, and in the second step the obtained alignments are processed and a new annotation package built.