Vignette: hoardeR

Guided Cross-Species Identification of Novel Gene Candidates

Daniel Fischer and Anu Sironen

July 7, 2016

Contents

1 Installation

The installation of hoardeR is straight forward, it is located on Cran and can be installed with the command

```
install.packages("hoardeR")
```

There are no special system requirements for the installation. However there are a few package dependencies that have to be met. Normally, these dependencies are installed automatically, too. If not, missing Cran packages be installed prior to the installation as above, with

```
install.packages("packageName")
```

Missing Bioconductor packages (e.g. the package *Biostrings*) with

```
source("https://bioconductor.org/biocLite.R")
biocLite("Biostrings")
```

The latest developer version, of hoardeR is located on GitHub, here

https://github.com/fischuu/hoardeR

and the address on Cran with the latest stable release version is

https://cran.r-project.org/web/packages/hoardeR/

Packages from GitHub can be installed directly in R, using the *devtools* package. To install hoardeR from GitHub, the following commands are required

```
install.packages("devtools")
library("devtools")
install_github("fischuu/hoardeR")
```

Once the package is installed, it can be loaded into the namespace

```r
library(hoardeR)
```

2 Using hoardeR for identifying cross-species orthologs of novel candidate genes

2.1 Typical workflow prior the use of hoardeR

The common application of hoardeR is to search cross-species orthologs of unannotated, but active regions in a present organism. For that, typically an RNA-seq experiment has been conducted, the reads have been mapped to a reference genome and gene expressions have been estimated using some annotation.

Further, reads from non-annotated regions have been analysed and a set of novel gene candidate regions has been identified. This can be done either across the whole data set, or individually for each sample.

From this analysis, the user has either a gtf file with novel loci (e.g. Cufflinks provides this), or then a bed file with the new loci.

The first rows of a typical gtf file would like this:

```
1       Cufflinks       exon    242203  242862  .       +       .       gene_id "XLOC_000002"; transcript_id "TCONS_00000002"; exon_number "1"; oId "CUFF.2.1"; class_code "u"; tss_id "TSS2";
1       Cufflinks       exon    242203  242646  .       +       .       gene_id "XLOC_000002"; transcript_id "TCONS_00000003"; exon_number "1"; oId "CUFF.1.1"; class_code "u"; tss_id "TSS2";
1       Cufflinks       exon    254559  256717  .       +       .       gene_id "XLOC_000002"; transcript_id "TCONS_00000003"; exon_number "2"; oId "CUFF.1.1"; class_code "u"; tss_id "TSS2";
1       Cufflinks       exon    254240  256717  .       +       .       gene_id "XLOC_000002"; transcript_id "TCONS_00000004"; exon_number "1"; oId "CUFF.3.1"; class_code "u"; tss_id "TSS3";
1       Cufflinks       exon    341982  343630  .       +       .       gene_id "XLOC_000003"; transcript_id "TCONS_00000005"; exon_number "1"; oId "CUFF.10.1"; class_code "u"; tss_id "TSS4";
1       Cufflinks       exon    342113  342607  .       +       .       gene_id "XLOC_000003"; transcript_id "TCONS_00000006"; exon_number "1"; oId "CUFF.11.1"; class_code "u"; tss_id "TSS5";
1       Cufflinks       exon    342961  343494  .       +       .       gene_id "XLOC_000003"; transcript_id "TCONS_00000006"; exon_number "2"; oId "CUFF.11.1"; class_code "u"; tss_id "TSS5";
1       Cufflinks       exon    3312599 3313720 .       +       .       gene_id "XLOC_000024"; transcript_id "TCONS_00000073"; exon_number "1"; oId "CUFF.75.1"; class_code "u"; tss_id "TSS37";
1       Cufflinks       exon    3446776 3447142 .       +       .       gene_id "XLOC_000024"; transcript_id "TCONS_00000073"; exon_number "2"; oId "CUFF.75.1"; class_code "u"; tss_id "TSS37";
1       Cufflinks       exon    9347375 9347527 .       +       .       gene_id "XLOC_000047"; transcript_id "TCONS_00000118"; exon_number "1"; oId "CUFF.115.1"; class_code "u"; tss_id "TSS63";
```

The first lines of a bed file containing the same information would look like this:

```
1       242203      242862      XLOC_000002.1
1       242203      242646      XLOC_000002.2
1       254559      256717      XLOC_000002.3
1       254240      256717      XLOC_000002.4
1       341982      343630      XLOC_000003.1
1       342113      342607      XLOC_000003.2
1       342961      343494      XLOC_000003.3
1       3312599     3313720     XLOC_000024.1
1       3446776     3447142     XLOC_000024.2
1       9347375     9347527     XLOC_000047.1
```

For the identification of cross-species orthologs, the exon structure is not the primary interest. Rather, the whole genomic region that hosts a novel candidate gene is used here. Typically, not all novel gene candidates are considered and the lists are filtered according to some criteria, e.g. that a certain amount of genes is located within these regions, or a certain amount of samples have to have reads in that region.

The final set of regions of interest is then available in bed format, e.g. like this

```
1       242203      256717      XLOC_000002
1       341982      343630      XLOC_000003
1       3312599     3447142     XLOC_000024
1       9347375     9347527     XLOC_000047
```

For these loci, the nucleotide sequences have to be extracted. There are two way to do that. The first option is to do it outside of R using bedtools. For that, the bed file has to be saved to the HDD first. Assuming the *data.frame* that contains the bed information is called *novelBed* and we defined a system path to the project, called *projFolder*. The command to export the *data.frame* then is

```
projFolder <- "/home/daniel/MyProjects/hoardeR-Example"
write.table(novelBed, file.path(projFolder,"novel.bed"), row.names=FALSE,
            col.names=FALSE, sep="\t", quote=FALSE)
```

In the console this bed file can then be used to extract the fasta files, using the bedtools tool like this

```
bedtools getfasta -fi <input FASTA> -bed novel.bed -fo novel.fa
```

Here, the is a fasta file that contains the genome information of the species under investigation. If the resulting *novel.fa* file is empty or some other errors occur, a common source of error is a mislabeling of the chromosomes between the input fasta file and the corresponding bed file (e.g. leading CHR, Chr, etc.). This approach is especially then adviceable, when the species of interest is rare or the fasta file is not available from ensembl in the latest version.

However, if the species and also the genome assembly version is available at ensembl, the fasta information can be obtained straight with the hoardeR function *getFastaFromBed*. The hoardeR package is able to download the most common species genomes and annotations, a list of available combinations can be found in the *species* dataset, that comes with hoardeR

```
bedtools getfasta -fi <input FASTA> -bed novel.bed -fo novel.fa
```

```
head(species)
```

```
##        Common.name         Scientific.name Taxon.ID       Ensembl.Assembly
## 1   Aardvark (Pre)   Orycteropus afer afer  1230840                      -
## 2           Alpaca          Vicugna pacos    30538                vicPac1
## 3     Amazon molly       Poecilia formosa    48698 Poecilia_formosa-5.1.2
## 4     Anole lizard    Anolis carolinensis    28377              AnoCar2.0
## 5        Armadillo   Dasypus novemcinctus     9361              Dasnov3.0
## 6 Budgerigar (Pre) Melopsittacus undulatus   13146                      -
##          Accession Variation.database Regulation.database Pre.assembly
## 1                -                  -                   -     OryAfe1
## 2                -                  -                   -           -
## 3 GCA_000485575.1                  -                   -           -
## 4 GCA_000090745.1                  -                   -           -
## 5 GCA_000208655.2                  -                   -           -
## 6                -                  -                   -    MelUnd6.3
```

Of particular interest are here the columns *Scientific.name* and *Ensembl.assembly*. If your species of interest matches your used assembly, you can use the automatic hoardeR function *getFastaFromBed* to obtain your fasta object like this. Here, we assume that our species of interest is cow/bos taurus and the fasta files should be downloaded to the folder */home/daniel/fasta/*

```
getFastaFromBed(novelBed, species="Bos taurus", fastaFolder="/home/daniel/fasta/")
```

It is also possible to obtain the genomic information with this command from assemblies that are not provided from Ensembl, or that have a newer/older version number. For that, the full syntax is

```
getFastaFromBed(novelBed, species="Bos taurus", release = "84", fastaFolder=NULL, version=NULL)
```

Here, the Ensembl release version and also the assmebly version can be specified. However, if no release number is given (i.e. *release=NULL*), the function assumes to find a non-ensembl fasta file in the folder *fastaFolder*. These fasta files need then to be of the format, e.g. for the cow assembly version UMD3.1

```
Bos_taurus.UMD3.1.dna.chromosome.1.fa.gz
Bos_taurus.UMD3.1.dna.chromosome.2.fa.gz
Bos_taurus.UMD3.1.dna.chromosome.3.fa.gz
...
```

That means, first the scientific name, with underscores, then a dot, then the assembly identifier followed by another dot, then *dna.chromosome.* and the chromosome identifier, and then the file ending *fa.gz*. That means, the fasta files have to be gzipped.

As a working example, on Ensembl is only the sheep genome assembly 'Oar_v3.1', if one needs to work with the 'Oar_v4.0', one would download the chromosome wise files and then store them in some folder on the HDD as

```
Ovis_aries.Oar_v4.0.dna.chromosome.1.fa.gz
Ovis_aries.Oar_v4.0.dna.chromosome.2.fa.gz
Ovis_aries.Oar_v4.0.dna.chromosome.3.fa.gz
...
```

To obtain then the corresponding fasta objects, the command is

```
getFastaFromBed(novelBed, species="Ovis aries", release = NULL, fastaFolder="/home/daniel/fasta/", vers
```

assuming again that the fasta files are located in */home/daniel/fasta/*