IRBIS: a systematic search for conserved complementarity

User Manual

August 23, 2014

Contents

1 Introduction

This document is a minimal description of the IRBIS pipeline and will be updated with the package. Currently it contains a minimal description of pipeline elements and some examples. Specific question can be sent to Dmitri Pervouchine (dpcrg.eu).

2 How to make it work

A copy of IRBIS can be obtained from GitHub. The pipeline uses GNU `make` utility for both compiling the C++ code and data processing.

Enter the folder and run sequentially

```
1. make all

2. make -f database.mk all

3. make -f metacalc.mk all
```

The first step creates binaries in `C/` and `Progs/` directories. The second step does automatic data download from UCSC, including genome sequences, pairwise sequence alignments, and annotations. The download location is specified by a number of variables in the `config.mk` and `config.dat` files. Namely, `$FIXEDPATH` is the path to the database directory. You might want to skip this step (at least the download part) if you already have the corresponding files in your file system. The third step creates metadata in the `$METADATA` directory and some of the output files in `$OUTDIR`.

IRBIS depends on g++, perl, latex + tikz + pgfplots (pgfplots is optional), and muscle (if muscle software is not in the system, you can obtain it by `make muscle`). **Important:** check the `$muscle` variable in `Perl/align.pm`

3 Inside the pipeline

The pipeline is configured by a number of Perl scripts which create makefiles; makefile is that is actually executed. This is controlled by three main scripts.

1. `Perl/make_database.pl` takes config.dat and determines all data preparation steps
2. `Perl/make_download.pl` takes care of the data download and processing
3. `Perl/make_metacalc.pl` takes sequence data, creates hash table metafiles (long), and executes some of the comparisons that are presented in the paper.

3.1 Reference genomes

The annotation of reference genomes is downloaded as stated in `special.mk` file. The input to the pipeline is a gtf file, which is processed sequentially as follows.

1. `Perl/gen_loci.pl` is an optional pre-processing step which determines and merges gene loci that have at least one splice site in common (by taking transitive closure). After this step, each splice site would has unique gene identifier.
2. `Perl/gtf2db.pl` is a step that creates and attributes the database of splice sites (.cps file) and segment relational files (.rel)
3. `Perl/all2all.pl` creates a binary all-to-all relation between splice sites of each gene (all-to-all within gene)
4. `Perl/intergenic_lncrna.pl` removes segments of lncRNA that overlap protein-coding genes
5. `C/transf` transforms genomic sequence data into the compressed format which keeps repeat-masking information.

3.2 Orthologous segments

As soon as exon boundaries in the reference genomes are computed, the next step of the pipeline is to find orthologous segments.

1. `C/map_agnostic` is a routine that maps exon boundaries to the target genome (not uniquely), outputs .aln file
2. The filtering step can be performed by one of the following routines (details can be found in the comments to these programs). Each of this programs outputs a unique mapping file (.map)
   (a) `C/net_filter` selects unique mappings based on NET alignments
   (b) `C/syntenic_filter` selects unique mappings by maximizing syntheny (the longest continuous stretch of exon boundaries)
   (c) `C/best_match` selects unique mappings by maximizing syntheny per gene (the longest continuous stretch of exon boundaries per gene)
3. `Perl/map2bed.pl` takes orthologous exon boundaries (.map) and outputs ortholohous segments (.bed)
4. `C/getsegm` uses the bed output of the previous step to pull out genomic sequences (.sus, single unaligned segment)
5. `C/getwind` can be used to output windows surrounding exon boundaries (.suw, single unaligned window). Currently it is used to create splice site signatures (.sgn)
6. `C/getmuf` is designed to ease the access to specific sets of genes by combining single unaligned segments from different species into one muf file (.mus multiple unaligned segments)
7. `C/indexing` creates index tables for sequence files (for speed-up)

3.3 Metadata

When all the sequence data is prepared, the next step is to compute hash tables, trim and store them as metadata.

1. `Progs/trim` is the routine that takes sequence file (either a collection of .sus defined in the configuration file or a single mus file), creates a hashtable and trims it according to t_1 threshold (.met)
2. `Progs/irbis` takes two metafiles on input, a binary relation file, and outputs conserved complemetary k-mers (.tab)
3. `Perl/tab2maf.pl` takes two configuration files and the tab output of the previous step, reads sequences and aligns them producing .maf (mulptiple alignment file)
4. `Perl/tab2pdf.pl` takes all the above inputs and creates a pdf file with one pair of alignments per page

4 Custom queries

Custom queries can be executed by using two makefiles.

4.1 pipeline.mk

This is a framework to run a search in set A vs. set B given a binary relation $\mathcal{R}$. The inputs are the following.

1. `DOMAIN` is the name of the database directory (no slash at the end)
2. `SPECIES` is the name of the configuration file (which also will be the name of the output directory)
3. `LEFT` a cps file of the set A
4. `RIGHT` a cps file of the set B
5. `PARAMS` is the list of parameters to be passed to `Progs/trim`, `Progs/irbis`, `Perl/tab2maf.pl`, and `Perl/tab2pdf.pl`

Examples:

1. `make -f pipeline.mk DOMAIN=insect SPECIES=insect LEFT=ncpcg RIGHT=ncpcg OUT=ncpcg_ss PARAMS='-t 0.9 -L 12 -g 1 -u 1 -B /db/metadata/insect/dm3.a2a' pdf` — finds conserved complemetary k-mers in non-coding segments of insect protein-coding genes according to all-to-all-within-one-gene relation specified in /db/metadata/insect/dm3.a2a; outputs a pdf
2. `make -f pipeline.mk DOMAIN=insect SPECIES=insect LEFT=ncpcg RIGHT=ncpcg OUT=ncpcg_all PARAMS='-t 0.9 -L 12 -g 1 -u 1' tab` — finds conserved complemetary k-mers in all pairwise combinations of segments of insect protein-coding genes according; outputs a tab file
3. `make -f pipeline.mk DOMAIN=vertebrate SPECIES=mammal LEFT=RP11-439A17.4 RIGHT=HIST OUT=RP11-439A17.4_HIST PARAMS='-t 0.8 -L 12 -g 1' pdf` — finds conserved complemetary k-mers between RP11-439A17.4 lncRNA (the corresponding cps file has to be present under /db/metadata/vertebrate/) and histone genes (same about HIST.cps) with the described parameters, outputs a pdf

4.2 single_gene.mk

This pipeline is designed to work with single genes. The inputs are the following.

1. `DOMAIN` is the name of the database directory (no slash at the end)
2. `SPECIES` is the name of the configuration file (which also will be the name of the output directory)
3. `NAME` is a cps file

4. `PARAMS` is the list of parameters to be passed to `Progs/trim`, `Progs/irbis`, `Perl/tab2maf.pl`, and `Perl/tab2pdf.pl`

Examples:

1. `make -f single_gene.mk DOMAIN=insect SPECIES=insect NAME=Ca-alpha1D PARAMS='-t 0.9 -L 12 -g 1 -u 1' pdf` — finds conserved complemetary k-mers in all segments of the insect Ca-alpha1D gene; outputs a pdf

2. `make -f single_gene.mk DOMAIN=insect SPECIES=insect NAME=Ca-alpha1D PARAMS='-t 0.9 -L 12 -g 1 -u 1 -b /db/metadata/insect/dm3.int -E' pdf` — finds conserved complemetary k-mers in all segments of the insect Ca-alpha1D gene by matching only segments that correspond to annotated introns