Package ‘ReferenceEnhancer’

March 1, 2023

Title: Package for optimizing and assembling genome annotations for 3’ single-cell RNA-sequencing analysis.

Description: ReferenceEnhancer contains a set of tools for optimizing genome annotations for droplet based 3’ single-cell RNA-sequencing (10x Genomics, Dropseq etc.). Regular genome annotations and transcriptomic references generated based on them come with several problems causing discarded sequencing data from final gene expression estimates (outlined in detail in https://www.biorxiv.org/content/10.1101/2022.04.26.489449v1). These include read loss stemming from gene overlaps, sequencing reads mapping to 3’ unannoated exons as well as introns. ReferenceEnhancer enables fixing these issues and assembling optimized genome annotations that circumvent these problems and recover the discarded gene expression data.

URL: https://github.com/PoolLab/ReferenceEnhancer

Version: 1.0

Depends: R(>=4.0.0, GenomicRanges(>=1.50.2), GenomicAlignments (>=1.34.1), stringr(>=1.5.0), GenomicFeatures (>=1.50.4), gdata(>=2.18.01), rtracklayer, BEDOPS(>v 2.0), bedtools(v.2.26.0).

License: Artistic-2.0

Maintainer: Helen Poldsam (helen.poldsam@utsouthwestern.edu), Allan-Hermann Pool (allan-hermann.pool@utsouthwestern.edu)

1. Installation of ReferenceEnhancer

~~~
``` r
install.packages("devtools")
require(devtools)
install_github("PoolLab/ReferenceEnhancer")
```
~~~

2. Basic workflow

~~~
```{r example}
library(ReferenceEnhancer)

genome_annotation <- LoadGtf("unoptimized.gtf")
gene_overlaps <- IdentifyOverlappers(genome_annotation)
OverlapResolutions(genome_annotation, gene_overlaps) # Optional, generates

# Manual curation of overlapping gene list

IsolateIntergenicReads("test_bam.bam", "test_index.bam.bai")
GenerateGeneLocationBed(genome_annotation)
GenerateExtensionCandidates()

# Manual curation 3’ gene ends

OptimizedAnnotationAssembler("unoptimized.gtf", "overlapping_gene_list.csv",
"gene_extension_candidates.csv", "rename_genes.csv")
```
~~~

3. LoadGTF

Description

Use to import the ENSEMBL/10x Genomics default genome annotation file (.gtf). Note: This file can be downloaded from 10x Genomics provided reference transcriptome "gene" folder at "https://support.10xgenomics.com/single-cell-gene-expression/software/downloads/latest" or Ensembl.org if wish to customize more.

<u>Usage</u>

```
LoadGtf(genome_annotation_gtf_path)
```

<u>Arguments</u>

`genome_annotation_gtf_path`	Path to the unoptimized genome annotion (.gtf) file.

<u>Value</u>

Resulting object contains the genome annotation entries from the genome annotation (.gtf) file.

<u>Examples</u>

```
LoadGtf("./genes.gtf")
```

4. IdentifyOverlappers

<u>Description</u>

Identifies all overlapping genes based on the unoptimized genome annotation file (.gtf), rank-order them according to the number of gene overlaps. Prioritize this gene list for manual curation focusing on exonically overlapping genes. Saves the list of overlapping genes in working directory (overlapping_gene_list.csv).

<u>Usage</u>

```
IdentifyOverlappers(genome_annotation)
```

<u>Arguments</u>

`genome_annotation`	Unoptimized genome annotation (.gtf) file. Could be obtained from Ensembl, Refseq, 10x Genomics or elsewhere.

<u>Value</u>

Rank-ordered gene list of same-strand overlapping genes (gene_overlaps).

<u>Examples</u>

```
genome_annotation <- LoadGtf("test_genes.gtf")

IdentifyOverlappers(genome_annotation)
```

5. OverlapResolutions

<u>Description</u>

Based on original genome annotation .gtf file and a list of overlapping genes, generates recommended actions for overlapping genes. This is an optional step that can help with decision making during the manual curation step.

<u>Usage</u>

```
OverlapResolutions(genome_annotation, overlap_data)
```

<u>Arguments</u>

`genome_annotation`	Unoptimized genome annotation (e.g. Ensembl/10x Genomics default genome annotation file (GTF))
`overlap_data`	A list of overlapping genes generated from IdentifyOverlappers

Value

Generates “overlapping_gene_list.csv” with added recommendations for resolving gene overlaps.

Examples

```
genome_annotation <- LoadGtf("test_genes.gtf")

gene_overlaps <- IdentifyOverlappers(genome_annotation)

OverlapResolutions(genome_annotation, gene_overlaps)
```

6. Manual curation of overlapping gene list

Inspect gene overlaps and transcript structure in Ensembl (https://nov2020.archive.ensembl.org/Mus_musculus/Info/Index for mice and https://useast.ensembl.org/Homo_sapiens/Info/Index for humans) to determine which readthrough and premature start transcripts to eliminate (mark under "transcripts_for_deletion" column in "overlapping_gene_list.csv"). Make sure you look up the correct genome build in Ensembl.

For pseudogenes/poorly supported gene models that obscure protein coding genes, examine evidence for its existence (are they reported by other annotation consortia - e.g. Refseq). You can also examine if any reads map to it with regular exonic reference and determine whether to keep or eliminate it from the annotation. To mark a gene for deletion, mark its status as "Delete" in the "final_classification" column in the “overlapping_gene_list.csv”.

For genes that cannot be unentangled due to overlap of terminal exons, delete one of them and rename the other to reflect that reads could orignate from both. To this end, mark one of the genes for deletion (add "Delete" under "overlapping_gene_list.csv" file $final_classification column). Also, create a new csv file "rename_genes.csv" where under column "old_names" you can list gene names that need to be renamed and under column "new_names" list the new hybrid gene name. See https://github.com/PoolLab/generecovery/tree/main/mouse_mm10_input_files for a formatting example.

7. IsolateIntergenicReads

Description

Intergenic reads are extracted from Cell Ranger aligned bam file. Use a scRNA-seq dataset of interest that has been aligned to the unoptimized genome reference with the cellranger count pipeline. Intergenic reads can be identified by two features: their read identity tag RE = "I" (for intergenic) OR their RE=E (for exonic) with AN = <some gene>. The latter reads are in fact intergenic reads since Cell Ranger wrongly classifies reads mapping antisense to an exon as exonic (i.e. RE="E"). The false exonic reads can be recognized and captured as proper intergenic reads by extracting two kinds of reads (RE=I and RE=E & AN=<something else than NA). Also, removing duplicates command in GenomicAlignments package does not work for intergenic (nor for intronic) reads. Duplicate and corrupt read removal has to be done manually (i.e. make sure cellular and molecular barcodes have specified lengths and duplicate barcodes removed).

Usage

```
IsolateIntergenicReads(bam_file_name, index_file_name)
```

Arguments

`bam_file_name`	Path to Cellranger generated bam file (run cellranger count pipeline on sequencing data of interest and aligning it to the unoptimized transcriptomic reference).
`index_file_name`	Path to Cellranger generated bam.bai file (run cellranger count pipeline on sequencing data of interest and aligning it to the unoptimized transcriptomic reference).

Value

Saves extracted intergenic reads as a separate file (intergenic_reads.bed)

Examples

```
IsolateIntergenicReads("./input.bam", "./input.bam.bai")
```

8. GenerateGeneLocationBed

Description

Makes a bed file with gene boundaries, which is required for assigning intergenic reads to a specific gene and discovering genes with large amounts of intergenic reads near its 3’ gene end.

Note: This step is partially run in linux Terminal in Bash and requires bedtools (https://bedtools.readthedocs.io/en/latest/content/installation.html). Put bedtools in PATH after installing.

In linux terminal, navigate to folder with the genome annotation of interest. Assuming that it is named "genes.gtf" per 10x Genomics convention.

Usage

```
GenerateGeneLocationBed(genome_annotation)
```

Arguments

`genome_annotation`	Genome annotation file in .gtf format.
`bedops_loc`	Location of BEDOPS in system.

Value

Saves gene_ranges.bed in working directory.

Examples

```
genome_annotation <- LoadGtf("./genes.gtf", “/usr/bin/bedops”)

GenerateGeneLocationBed(genome_annotation)
```

9. GenerateExtensionCandidates

Description

Identifies candidate genes for extension with excess 3' intergenic reads and create a rank ordered list of genes as a function of 3' intergenic read mapping within 10kb of known gene end. You can use this as a prioritized gene list for gene extension to examine in Integrated Genomics Viewer.

Note: It runs partially in bash/linux terminal. Make sure bedtools is installed and in PATH variable in Linux or MacOS.

Usage

```
GenerateExtensionCandidates(bedtools_loc)
```

Arguments

`Bedtools_loc`	Location of bedtools in system

Value

Rank ordered list of gene extension candidates saved to working directory as gene_extension_candidates.csv

Examples

```
GenerateExtensionCandidates(“/opt/bedtools2/bin”)
```

10. Manual curation 3’ gene ends

Download and install https://software.broadinstitute.org/software/igv/. Load the transcriptome aligned sequencing read data ("xxx.bam" file) to the igv and choose appropriate species/genome. Go through the rank ordered candidate gene list in "gene_extension_candidates.csv" and examine evidence for unannotated 3' exons and UTRs:

a) extensive splicing between annotated and candidate unannotated regions at the genetic locus as visualized by igv.

b) continuous read mapping from known gene end (delayed cutoff of sequencing reads posterior from known gene end)

c) examine external evidence (e.g. Allen in situ brain atlas, Human Protein Atlas etc.)

Based on the latter, provide new gene boundaries under two new columns: "update_start" and "update_end" depending on whether gene is on "-" or "+" strands, respectively. Use excel formulas, if necessary, to scale curation, and save file as "gene_extension_candidates.csv". Resulting manually curated "gene_extension_candidates.csv" will be used as input for the OptimizedAnnotationAssembler function.

11. OptimizedAnnotationAssembler

Description

OptimizedAnnotationAssembler generates the scRNA-seq optimized genome annotation. The resulting optimized genome annotation can be used to generate the transcriptomic reference for mapping single-cell sequencing data (e.g. with cellranger mkref or STAR --runMode genomeGenerate). This function goes through the following steps:

0. Load data and libraries

- genome annotation file (.gtf) to be optimized

- "overlapping_gene_list.csv" file specifying how to resolve gene overlap derived issues. "Delete" entries in $final_classification field mark genes for deletion. Transcript names in $transcripts_for_deletion mark specific transcripts for deletion.

- "gene_extension_candidates.csv" specifying updated gene boundaries for incorporating intergenic reads

- "rename_genes.csv" specifying gene names to be replaced and new names (under $old_names and $new_names fields, respectively)

1. Creates pre-mRNA genome annotation from input genome annotation. This step extracts all transcript entries from the genome annotation and defines them as full length exons with new transcript IDs and corresponding transcripts. This allows to capture many intronically mapped reads that otherwise get discarded.

2. Gene deletion step: Deletes all annotation entries for genes destined for deletion (has "Delete" entry in $final_classification field of "overlapping_gene_list.csv"

3. Transcript deletion step: Deletes all transcripts destined for deletion (transcript names listed in the "transcripts_for_deletion" column in ""overlapping_gene_list.csv"

4. Gene coordinate adjustment step: replace the left most or right most coordinate of the first exon of a gene in genome annotation if there is a coordinate in columns $new_left or $new_right in the "gene_extension_candidates.csv".

5. Add pre-mRNA reads to all genes not in the gene overlap list.

6. Rename genes to avoid discarding expression data with near perfect terminal exon overlap.

7. Save the optimized genome annotation in a new GTF file

Usage

```
OptimizedAnnotationAssembler(

  unoptimized_genome_annotation,

  resolved_gene_overlaps,

  gene_extension_list,

  gene_rename_list

)
```

Arguments

`unoptimized_genome_annotation`	Original genome annotation to be optimized
`resolved_gene_overlaps`	CSV file containing the recommended actions for gene and transcript deletions
`gene_extension_list`	CSV file containing genes with updated coordinates to capture reads from unannotated 3' exons and UTRs.
`gene_rename_list`	CSV file containing the genes to be renamed and new names they should receive.

Value

Single-cell RNA-seq optimized genome annotation that can be used to generate the transcriptomic reference (e.g. with cellranger mkref or STAR --runMode genomeGenerate) for mapping single-cell sequencing data.

Examples

```
OptimizedAnnotationAssembler(“./genes.gtf”, “./overlapping_gene_list.csv”,
“gene_extension_candidates.csv”, “./rename_genes.csv”)
```