README file: RNA-seq pipeline

This is the README file for the RNA-seq differential expression analysis pipeline constructed for the Mercer cluster on the NYU HPC. It contains the following sections:

****

1. INTRO:

1.1: THE PIPELINES: There are two pipelines in this package: The MAIN PIPELINE is a single-button analysis pipeline that takes in a list of paired-end RNAseq-read samples (FastQ files) and applies a series of

analyses and validations resulting in differential expression analysis. The second pipeline is an optional PREPROCESSING PIPELINE, which conducts a format validation and a quality check of the FastQ files to be used in the pipeline.

1.2: HISTORY/CONSTRUCTION: This pipeline was created, initially, for use at NYU in the analysis of genome data from the datepalm (Phoenix dactylifera). It was developed by Michael Dhar (mid224@nyu.edu) as a graduate guided-study project within the NYU-Poly bioinformatics MS program under the supervision of Dr. Jonathan Flowers (NYU, jmf11@nyu.edu) in spring/summer 2015. Test runs were conducted on paried-end (2x50) RNA-seq japonica rice (Azucena) samples from a temperature-stress experiment: two "control" samples (3 hrs at 30 C) and two "treatment" samples (3 hrs at 40 C). The four samples (each containing both forward and reverse reads) all came from chromosome 11 of the rice genome. Testing was done on the Mercer cluster of the NYU HPC system in spring/summer 2015.

1.3: PIPELINE CHARACTERISTICS: This is a "ONE-BUTTON" pipeline, designed to run autonomously via a single command once the appropriate files and packages are in place. It is also designed to be MODULAR, meaning the tools used in the analysis are decoupled as much as possible from the scheduling/backbone of the pipeline. This is done so that alternative analysis tools may be, with minimal adjustments, be slotted in. For example, in place of the default Tophat tool, an alternative alignment program could be used. Modularity is achieved by separating, where possible, the PBS scheduling commands from the BASH scripts that actually execute the analysis tool. Analysis steps are run as DEPENDENCIES on the qsub system, with subsequent steps beginning only when previous steps have returned an OK exit status. When a (time-intensive) analysis tool must loop through a sample list, the tool is called as a PBS ARRAY JOB, with each array subjob calling a new iteration of the BASH script for the analysis tool in question, each of which operates on a single sample in the list.

****

2: PIPELINE STEPS:
This is an outline of the steps that take place in the execution of the Preprocessing and Main pipelines. For details on executing the pipelines, see PIPELINE EXECUTION (3) below.

2.1: MAIN PIPELINE: The main pipeline conducts differential expression analysis on a set of reads stored in FastQ files.

2.1.1: PREP FILES/TOOLS: A collection of preparatory files must be in place for the pipeline to run. (See 3.1 below for tool and package download and version details.) These are:

* Config.txt: A collection of paths to data files needed by the pipeline: Reference genome, Fasta index file, GFF3 genomic feature

file, and Sample Table (see below).
        * Control file: The Sample Table, a list of tab-delimited sample IDs, paths and experimental conditions.
        * Tool Modules: A set of tool modules must be present on the computational environment. (They are present on Mercer):
                * For Main Pipeline: Perl, Bowtie, Tophat, Samtools, R
                * For Preprocessing Pipeline: FastQC
        * Bioconductor packages: Bioconductor, easyRNASeq, DESeq

2.1.2: VALIDATOR: A Perl Validator script checks that the Control File (Sample File) submitted to the program is formatted correctly (see below, 3.1.1, for proper format). On any error found, the Validator script reports the error to the user and dies, forcing the overall Main Pipeline script to fail and stop execution.

2.1.3: DIRECTORY STRUCTURE: The pipeline directory structure is then created by executing a BASH script. Within the working "project" directory, an "alignments.1" directory is created. Within this directory, a folder for each Sample from the Control File (Sample File) is created, which will store all Alignment output (BAM files). Another directory, bam_links, is also created within alignments.1, for storing soft links to all BAM files.

2.1.4: INDEX CREATION: Prior to the ALIGNMENT step, indices must be created. The default alignment tool, Tophat, requires Bowtie indices, created using the Bowtie short-read alignment tool.

2.1.5: ALIGNMENT: The input reads from the samples are aligned to the reference genome. The default alignment tool is the splice-junction mapper Tophat. It first runs the Bowtie short-read aligner on the reads, then performs splice-junction analysis on those results. It produces BAM files, saved as "accepted_hits.bam" for each sample and saved in a folder named for that Sample. Using Samtools, a .bai index file is created in the same directory for every .bam file.

2.1.6: SOFT LINKS: Soft (symbolic) links are created for all BAM files created in the alignment step, and for their accompanying .bai index files. These soft links are named for the sample IDs (e.g., "Sample_A"'s accepted_hits.bam and accepted_hits.bam.bai produce the soft-links Sample_A.bam and Sample_A.bam.bai.) This is done to provide unique BAM file names, which are used as headers in the count table produced by the subsequent COUNT step. The soft-link .bam and .bai files are all stored in the directory "bam_links."

2.1.7: COUNT: This step applies the simpleRNASeq method of the easyRNASeq package within Bioconductor, a set of tools in the R statistical programming language. The simpleRNASeq method calculates the coverage of reads against the reference genome, summarized by a feature (with the default feature being "genes" in this pipeline). Its output is a "summarized experiment" object, which contains various

data, including a count table for the features. The pipeline uses an accessor function within the COUNT step to retrieve this count table, and it is then written to a text file, "count.table.txt", which is accessed by the next step.

2.1.8: DIFFERENTIAL EXPRESSION: This final step of the pipeline applies the R/Bioconductor tool DESeq to calculate differential expression between experimental conditions. First, a Perl script ("ConditionWriter.pl") is called to produce a table of the experimental conditions needed to run DESeq. This Perl script creates a table listing the headers of the count table (BAM filenames that are used to name each sample in the count table) with their appropriate conditions (treatment vs. control). This is done by reading through the count table headers, searching the original Control file for the matching IDs, and then pairing those headers with the appropriate conditions as listed in the Control file. (This is done to ensure that a header is not matched to the wrong condition. That is, if a later BAM file was completed before an earlier one due to parallelization, it would not be accurate to simply list the count table headers in order followed by the conditions in the order found in the control file.)

The pairs of ID-Condition and the count table are then fed to DESeq, which compares expression levels in treatment vs. control samples. This output is dumped into a file,"DESeq.output.txt". The information can be used to plot differential expression within R/DESeq and for other downstream analysis. As part of the DESeq R script, two basic plots are created and saved to a PDF file ("deseq_plots.pdf"): 1) the log2 fold changes against mean normalized counts and 2) a histogram of p values.

This is the final output for the main pipeline.

2.2: PREPROCESS PIPELINE: At the user's discretion, another pipeline may be run before the Main RNASeq pipeline. This Preprocessing pipeline validates the FastQ files listed in the control file.

2.2.1: LENGTH/QC: The FastQ files are checked to ensure that each forward FastQ is the same length as its partner reverse FastQ. Any non-matching pairs are reported to the log file "PreProcLog.txt". In this same step, the FastQ files are checked for the presence of any reads filtered by the Illumina quality check. The number of filtered reads is reported to the same log file.

2.2.2: FASTQC: The FastQC quality-check program is run on each FastQ file, producing reports on base-sequence quality, GC content, sequence overrepresentation, and other measures. The FastQC results are stored in a directory corresponding to each FASTQ file (e.g. "FASTQ_R1_fastqc").

2.2.3: READ_ID: This step checks that each read ID in each forward FastQ file matches the corresponding read ID in the reverse FastQ. If any do not match up, this is reported to the log file. This completes the Preprocessing pipeline.

****

3. MAIN PIPELINE EXECUTION:

3.1: USAGE SUMMARY: To execute the Main pipeline:
	1) Supply a Control file with all sample information and fill out the Configuration file variables
	2) Copy all pipeline scripts to the project directory in /scratch
	3) Execute the following command from a login node: bash RNAseq_master.sh

See below for details.

3.2: PREPARATORY STEPS: Prepare, download or copy the following files as necessary before executing the pipeline.

3.2.1: FASTQ FILES:
RNA-seq reads should be stored in FastQ files. All FastQ files should be saved somewhere on the Mercer system and accessible from a program run in the working directory for the pipeline. Paths to these files will be provided to the pipeline in the Configuration file (below). By default, the pipeline expects paired-end FastQ files, so there should be a forward and reverse FastQ for every sample. FastQ files should be properly formatted. To pass the Preprocessing pipeline, the FastQ files must meet the following criteria: 1) forward and reverse FastQ pairs should be the same length and 2) read IDs in the forward file should match their pairs in the reverse file. (The Preprocessing pipeline will also check for reads filtered by Illumina QC.)

3.2.2: GENOME/INDEX FILES: The following genome and index files should be saved somewhere on the Mercer system, accessible to a program run from the working directory for the pipeline. The Alignment step will copy the reference genome to the working directory, so the user who executes the pipeline should have proper permissions to copy this file. Please be sure that these files are formatted properly.
	* Reference Genome
	* FASTA Index (.fai extension)
	* Genome Feature File: This must be stored in a GFF3 format, and MUST contain the following header as its first line: ##gff-version 3
		(The COUNT step of the pipeline uses simpleRNASeq, part of easyRNASeq, which will not recognize a GFF3 file without that header, halting the pipeline.)

3.2.3: CONTROL FILE

The Control file, or Sample file, contains a list of all paired-end samples to be analyzed in the experiment. Please ensure PROPER FORMATTING. The Control file must be formatted as follows: Each line contains information for one sample. The line contains four and only four tab-delimited columns, listing: 1) the sample ID, 2) the path to the forward FastQ, 3) the path to the reverse FastQ, and 4) the experimental condition for that sample (treatment or control). There should be no blank lines, and lines should end with the proper, Unix end-of-line character (\n). Sample ID names can contain only alphanumeric characters and the underscore. FastQ paths can contain only alphanumeric characters, periods, underscores and forward-slashes. The VALIDATOR step will check that the Control file meets all formatting requirements, and will stop pipeline execution if any errors are found. (The Validator will also check that all FastQs listed exist.)

3.2.4: CONFIGURATION FILE

The Configuration file contains information needed to run the pipeline. Fill out the following slots in the Configuration file with your experiment's information, and save as "Config.txt."

* REF: the path to the reference genome
* FASTA_IND: the path to the .fai FASTA index file
* GFF: the path to the GFF3 genome feature file
* SAMPLE_TABLE: the Control file, or Sample list (default name is "cntrlfile.txt")

3.2.5: MAIN PIPELINE SCRIPTS

Copy all Main pipeline scripts and files to the project directory. These 14 files are needed to run the pipeline:

* Config.txt
* RNASeq_master.sh
* Validator.pl
* DirStruct.sh
* BowtieIndex.pbs
* BowtieIndex.sh
* Tophat.pbs
* Tophat.sh
* SoftBam.pbs
* simpleRNASeq.pbs
* simpleRNASeq.R
* DESeq.pbs
* ConditionWriter.pl
* DESeq.R

3.2.6: TOOLS/PACKAGES

The following program modules must be available on the HPC system. (Note: These are all available on Mercer.)

        * BowTie v 2.2.3 (http://bowtie-bio.sourceforge.net/index.shtml)
        * Tophat v 2.0.12 (https://ccb.jhu.edu/software/tophat/index.shtml)
        * Samtools v 0.1.19 (http://www.htslib.org/doc/samtools.html)
        * [for Preprocessing pipeline]: FastQC v 0.11.2 (http://www.bioinformatics.babraham.ac.uk/projects/fastqc/)

The following R/Bioconductor packages must be downloaded to the working environment before the pipeline can be completed. Download each by opening an R session and typing the commands listed:
        * Bioconductor v 3.1 (https://www.bioconductor.org/install/)
                source("http://bioconductor.org/biocLite.R")
                biocLite()
        * easyRNASeq v 2.4.7 (http://www.bioconductor.org/packages/release/bioc/html/easyRNASeq.html):
                source("http://bioconductor.org/biocLite.R")
                biocLite("easyRNASeq")
        * DESeq v 1.20.0 (http://bioconductor.org/packages/release/bioc/html/DESeq.html)
                source("http://bioconductor.org/biocLite.R")
                biocLite("DESeq")

3.3: USAGE COMMAND: The Main pipeline is executed by entering the following command from a login node: bash RNAseq_master.sh

****

4: PREPROCESSING PIPELINE EXECUTION:

4.1: USAGE SUMMARY: To execute the Preprocessing pipeline:
        1) Have Control and Configuration files present as above
        2) Copy all Preprocessing pipeline scripts to the project directory in /scratch
        3) Execute the following command from a login node: bash PreProc_master.sh

See below for details.

4.2: PREPARATORY STEPS: In order to execute the Preprocessing pipeline, make sure that the necessary FastQ files, Control file and Configuration files are present and properly formatted as in 3.1.1-3 above. The Preprocessing pipeline needs to gunzip gzipped FastQ files, so be sure that the user executing the pipeline has proper permissions to manipulate these FastQ files. Also, be sure that the FastQC module (v 0.11.2, http://www.bioinformatics.babraham.ac.uk/projects/fastqc/)) is available on the system in which the pipeline is being executed. (It is available on the NYU HPC Mercer cluster.) The following Preprocessing pipeline scripts should be copied into the project directory:

* PreProc_master.sh
* LengthFilter.pbs
* FastQC.pbs
* FastQC.sh
* ReadID.pbs

4.2: USAGE COMMAND: The Preprocessing pipeline is executed by entering the following command from a login node: bash PreProc_master.sh