omics$_{p}ipe$Documentation

Release 1.1.3

Kathleen Fisch, Ph.D.

October 10, 2014

Contents

CHAPTER 1

Omics Pipe: An Automated Framework for Next Generation Sequencing Analysis

1.1 Introduction

Welcome to the documentation for Omics Pipe! Omics pipe is an open-source, modular computational platform that automates 'best practice' multi-omics data analysis pipelines published in Nature Protocols and other commonly used pipelines, such as GATK. It currently automates and provides summary reports for two RNA-seq pipelines, variant calling from whole exome sequencing (WES), variant calling and copy number variation analysis from whole genome sequencing (WGS), two ChIP-seq pipelines and a custom RNA-seq pipeline for personalized genomic medicine reporting. It also provides automated support for interacting with the The Cancer Genome Atlas (TCGA) datasets, including automatic download and processing of the samples in this database.

About Omics Pipe

1.2 Installation

Local Cluster

Amazon Web Services (AWS)

1.3 Tutorials

Step-by-Step Tutorial

Creating a Custom Pipeline

Creating Custom Modules

1.4 Online Resources

Homepage:	http://sulab.org/tools/omics-pipe/
Repository:	https://bitbucket.org/sulab/omics_pipe
Online Docs:	http://packages.python.org/omics_pipe
Download & PyPI:	http://pypi.python.org/pypi/omics_pipe

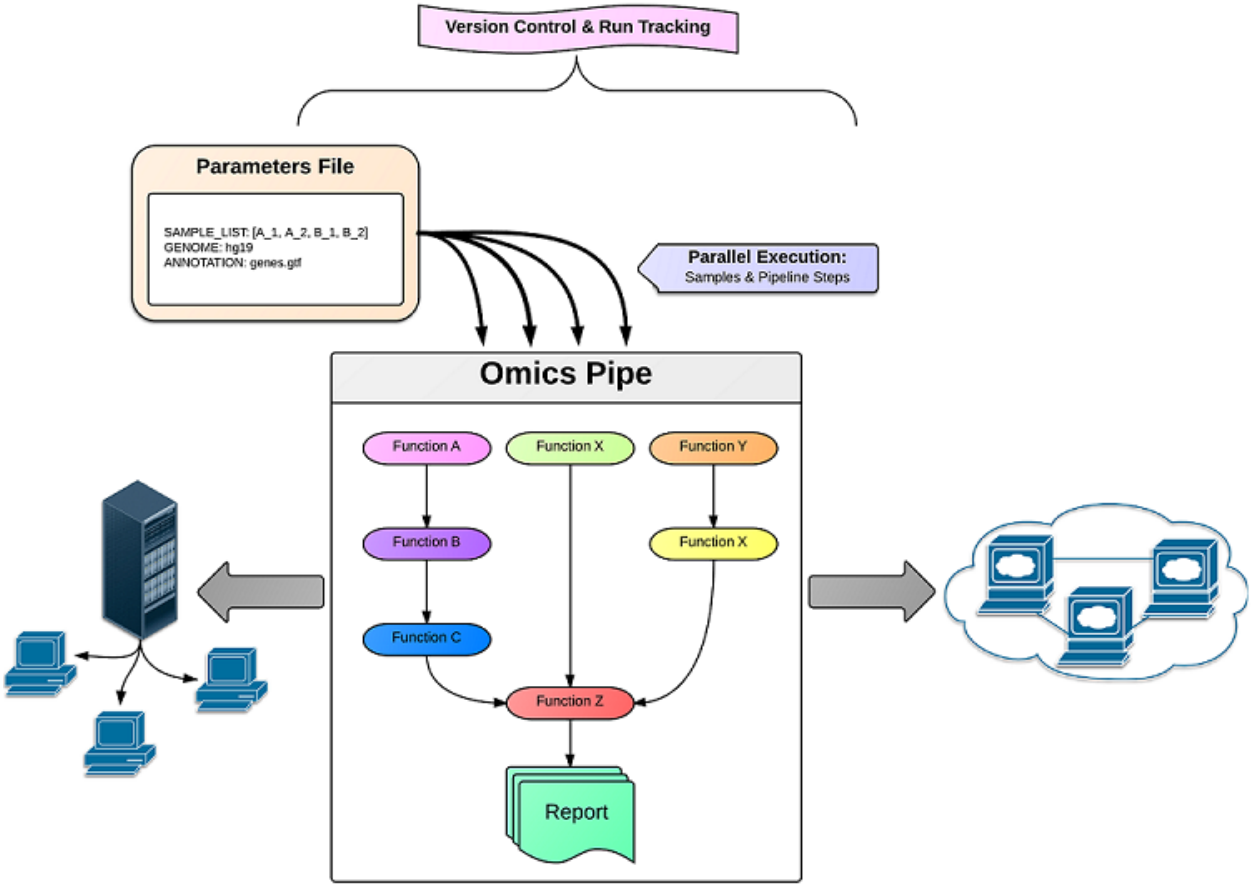
1.5 Site Navigation

- *genindex*
- *modindex*
- *search*

CHAPTER 2

About Omics Pipe

Omics pipe is an open-source, modular computational platform that automates 'best practice' multi-omics data analysis pipelines published in Nature Protocols and other commonly used pipelines, such as GATK. It currently automates and provides summary reports for two RNA-seq pipelines, two miRNA-seq pipelines, variant calling from whole exome sequencing (WES), variant calling and copy number variation analysis from whole genome sequencing (WGS), two ChIP-seq pipelines and a custom RNA-seq pipeline for personalized genomic medicine reporting. It also provides automated support for interacting with the The Cancer Genome Atlas (TCGA) datasets, including automatic download and processing of the samples in this database.

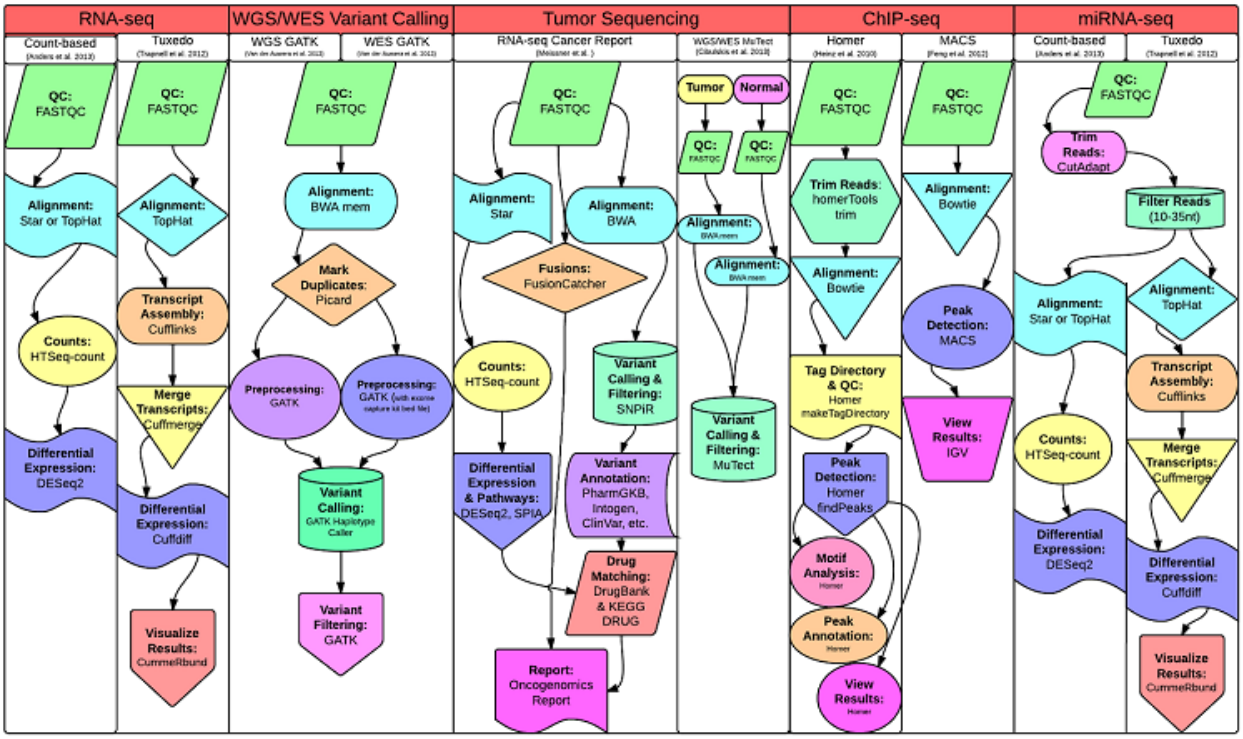


Omics pipe is a Python package that can be installed on a compute cluster, a local installation or in the cloud. It can be downloaded directly from the Omics pipe website for local and cluster installation, or can be used on AWS

in Amazon EC2. The modular nature of Omics pipe allows researchers to easily and efficiently add new analysis tools with Bash scripts in the form of modules that can then be used to assemble a new analysis pipeline. Omics pipe uses Ruffus to pipeline the various analysis modules together into a parallel, automated pipeline. The dependence of Omics pipe on Ruffus also allows for the restarting of only the steps in the pipeline that need updating in the event of an error. In addition, Sumatra is built into Omics pipe, which provides version control for each run of the pipeline, increasing the reproducibility and documentation of your analyses. Omics pipe interacts with the Distributed Resource Management Application API (DRMAA), which automatically submits, controls and monitors jobs to a Distributed Resource Management system, such as a compute cluster or Grid computing infrastructure. This allows you to run samples and steps in the pipeline in parallel in a computationally efficient distributed fashion, without the need to individually schedule and monitor individual jobs. For each supported pipeline in Omics pipe, results files from each step in the pipeline are generated, and an analysis summary report is generated as an HTML report using the R package knitr. The summary report provides quality control metrics and visualizations of the results for each sample to enable researchers to quickly and easily interpret the results of the pipeline.

2.1 Available Pipelines

Omics Pipe Available Pipelines Pipelines supported by this version of omics pipe.



2.2 Users

Projects that have used Omics Pipe for solving biological problems. Please submit your story if you would like to share how you use the pipeline for your own research.

- The Scripps Research Institute, Lotz Lab: The Lotz Lab in the Department of Molecular and Experimental Medicine at TSRI uses Omics Pipe to perform RNA-seq and miRNA-seq analyses on human articular cartilage samples to elucidate molecular pathways dysregulated in Osteoarthritis.

- Avera Health: Researchers working in collaboration with Avera Health use Omics Pipe to analyze sequence data from multiple platforms to provide personalized medicine to breast cancer patients.
- Scripps Laboratories for tRNA Synthetase Research: Researchers working in collaboration with Scripps Laboratories for tRNA Synthetase Research use Omics Pipe to analyze ChIP-seq data to explore transcription factor binding sites under experimental conditions.
- Dorris Neuroscience Center: Researchers working in collaboration with The Maximov Lab in the Department of Molecular and Cellular Neuroscience at The Scripps Research Institute use Omics Pipe to analyze RNA-seq data to determine how extensive activity-dependent alternative mRNA splicing occurs in the transcriptome of a mouse model that is born and develops to adulthood without synaptic transmission in the forebrain.
- Sanford Burnham Medical Research Institute: Researchers working in collaboration with The Peterson Lab in the Bioinformatics and Structural Biology Program at Sanford Burnham Medical Research Institute are using Omics Pipe to perform RNA-seq based global gene expression analysis of dental plaque microbiota derived from twin pairs to identify functional networks of the dental microbiome in relation to dental health and disease.

2.3 Developers

Omics Pipe is developed by Kathleen Fisch, Tobias Meissner and Louis Gioia at The Su Lab in the Department of Molecular and Experimental Medicine at The Scripps Research Institute in beautiful La Jolla, CA.

2.4 Contact

Feedback, questions, bug reports, contributions, collaborations, etc. welcome!

Katie Fisch, Ph.D.

Email: kfisch@scripps.edu

Twitter: @kathleenfisch

CHAPTER 3

Using Omics Pipe

Omics Pipe is a Python framework for automating 'best practice' next generation sequencing pipelines. Omics Pipe can be run from the command-line by providing it with a YAML parameter file specifying your directory structure and software specific parameters. This executes a parallel automated pipeline on a Distributed Resource Management system (local cluster or Amazon Web Services (AWS)) that efficiently handles job resource allocation, monitoring and restarting. The goals of Omics Pipe are to provide researchers with an open-source computational solution to implement 'best practice' pipelines with minimal development overhead and providing visual outputs to aid the researcher in biological interpretation.

To install Omics Pipe, first determine if you are going to be using it on a local compute cluster or on AWS. If you are going to be installing it on your local cluster, follow the directions below (or have your system administrator install it globally). If you are going to create a local installation in your home directory on your cluster but you do not have administrative permissions, you can create a Python Virtual Environment and then follow the instructions below within the virtual environment.

3.1 Requirements

- HPC Cluster or AWS Star Cluster (*Resource Requirements*)
- Python >=2.6
- Modules
- **R >= 2.15**
  - R Packages (*Third Party Software Dependencies*)
- Third Party Software Dependencies (*Third Party Software Dependencies*)
- Reference Databases (*Reference Databases Needed*)

3.2 Installation

- **Option 1**: Install from pypi using pip:

```
pip install omics_pipe
```

- **Option 2**: Install from pypi using easy_install:

```
easy_install omics_pipe
```

- **Option 3**: Install from source: Download/extract the source code and run:

```
python setup.py install
```

- **Option 4**: Install the latest code directly from the repository:

```
pip install -e hg+https://bitbucket.org/sulab/omics_pipe#egg=omics_pipe
```

- **Option 5**: If you do not have administrator privileges on your system:

```
Step 1: Set up a 'Python Virtual Environment'_
Step 2: Use one of the Options (1-4) above to install Omics Pipe within your virtual environment
```

3.3 Usage

Once you have successfully installed Omics Pipe, you can run a pipeline by typing the command:

```
omics_pipe [-h] [--custom_script_path CUSTOM_SCRIPT_PATH]
          [--custom_script_name CUSTOM_SCRIPT_NAME]
                          [--compression {gzip, bzip}]
          {RNAseq_Tuxedo, RNAseq_count_based, RNAseq_cancer_report, RNAseq_TCGA, RNAseq_TCGA_counts,
          parameter_file
```

3.4 Running Omics Pipe on Amazon Web Services (AWS)

AWS Installation Instructions Installation instructions for setting up the AWS Omics Pipe AMI

3.5 Tutorial

Tutorial Step-by-step tutorial for running Omics Pipe

Creating a custom pipeline Tutorial for creating and running a custom pipeline in Omics Pipe using existing modules

Adding new modules/tools Tutorial for adding new modules to Omics Pipe be used in a custom pipeline

3.6 Version history

Version History

3.7 Documentation

The latest copy of this documentation should always be available at: http://packages.python.org/omics_pipe

3.8 Questions

Email: kfisch@scripps.edu

Twitter: @kathleenfisch

CHAPTER 4

OmicsPipe on the Amazon Cloud (AWS EC2) Tutorial

OmicsPipe on AWS uses a custom StarCluster image, created with docker.io (which installs docker.io, environment-modules, and easybuild on an AWS EC2 cluster). All you have to do is get the docker image, upload your data, launch the Amazon cluster and run a single command to analyze all of your data according to published, best-practice methods.

4.1 Step 1: Create an AWS Account

1. Create an AWS account by following the instructions at Amazon-AWS
2. Note your AWS ACCESS KEY ID, AWS SECRET ACCESS KEY and AWS USER ID

4.2 Step 2: Load the the OmicsPipe on AWS docker image on your machine

1. Download docker.io following the instructions for your operating system at Get-Docker
2. From inside the Docker environment, run the command:

```
docker run -i -t omicspipe/aws_readymade /bin/bash
```

Note: If you want to share a file from your local computer with the docker container, follow the instructions for Docker Folder Sharing, put your desired file within the shared folder and run the command below (this is recommended for saving your /.starcluster/config file from the next step:

```
docker run -it --volumes-from NameofSharedDataFolder omicspipe/aws_readymade /bin/bash
```

- If you are on a local Ubuntu installation, skip this step and install the StarCluster client directly.
- If you are using Windows, it might be necessary to update your BIOS to enable virtualization before installing Docker

4.3 Step 3: Configure StarCluster

1. After running the omicspipe/aws_readymade Docker container, run the command below to edit the StarCluster configuration file:

```
    nano ~/.starcluster/config
```

Or if you prefer vim::

```
    vim ~/.starcluster/config
```

2. Enter your "AWS ACCESS KEY ID", "AWS SECRET ACCESS KEY", and "AWS USER ID"
3. Change the AWS REGION NAME and AWS REGION HOST variables if you do not live in the AWS us-west region to the appropriate region AWS Regions.
4. Select your desired pre-configured cluster by editing the "DEFAULT_TEMPLATE" variable or creating a custom cluster. The default is a test cluster with 5 c3.large nodes.
5. Save the edited file (Instructions for Nano and for Vim)
6. Create your starcluster SSH key by running the command:

```
starcluster createkey omicspipe -o ~/.ssh/omicspipe.rsa
```

- To remove a key from the AWS registry, run the command:

```
starcluster removekey omicspipe
```

- For more information on editing the StarCluster configuration file, see the StarCluster website.

4.4 Step 4: Create AWS Volumes

1. Create AWS volumes to store the raw data and results of your analyses. From within the Docker environment, run:

```
starcluster createvolume --name=data -i ami-52112317 -d -s <volume size in GB> us-west-1a

starcluster createvolume --name=results -i ami-52112317 -d -s <volume size in GB> us-west-1a
```

- Specify the <volume size in GB> as a number large enough to accomodate all of your raw data and ~4x that size for your results folder
- Change us-west-1b to your region as described in AWS Regions.

2. Make a volume from the provided snapshot of reference databases (currently only supports H. sapiens)

- Go to the AWS-Console
- Click on the EC2 option
- Click on Volumes
- Click on "Create Volume"
- Set availability zone
- In Snapshot ID search for "omicspipe_db" and click on the resulting Snapshot ID
- Click Create
- From the Volumes tab, note the "VOLUME_ID" of the database snapshot

3. Edit your StarCluster configuration file to add your volume IDs. Run the command below and edit the VOLUME_ID variables for data, results, and database:

```
nano ~/.starcluster/config
```

Edit the fields below:

```
[volume results]
VOLUME_ID =
MOUNT_PATH = /data/results

[volume data]
VOLUME_ID =
MOUNT_PATH = /data/data

[volume database]
VOLUME_ID =
MOUNT_PATH = /data/database
```

4. Save your StarCluster configuration file to ~/.starcluster/config

4.5 Step 5: Launch the Cluster

1. From the Docker container, run the command below to start a new cluster with the name "mypipe":

   ```
   starcluster start mypipe
   ```

2. Optional but Recommended: To load balance the cluster, type the command below:

   ```
   starcluster loadbalance mypipe
   ```

   (see **'_load balance'_** for configuration options. Note: make sure to keep at least one woker node attached to the cluster)

3. SSH into the cluster by running the command below:

   ```
   starcluster sshmaster mypipe
   ```

4.6 Step 6: Upload data to the cluster

Now that you are in your cluster, you can use it like any other cluster. Before running omics pipe on your own data (you can skip this step if you are running the test data, you will want to upload your data. There are two options to upload your data:

1. Upload data from your local machine or cluster using StarCluster put:

   ```
   starcluster put mypipe <myfile> /data/raw
   ```

2. Use Webmin to transfer files from your local system to the cluster (recommended for small files only, like parameter files).
   - In the AWS Management Console go to "Security Groups"
   - Select the "StarCluster-0_95_5" group associated with your cluster's name
   - On the Inbound tab click on "Edit"
   - Click on "Add Rule" and a new "Custom TCP Rule" will apear. On "Port Range" enter "10000" and on "Source" select "My IP"

- Hit "Save"
- Selct Instances in the AWS managemnt console and note the "Public IP" of your instance
- In a Web browser, enter https://the_public_ip:10000. Type in the Login info when prompted: user: root password: sulab
- This will take a few seconds to load, and once it does, you can navigate your cluster's file structure with the tabs on the left
- To upload a file from your local file system, click "upload" and specify the directory /data/data to upload your data.

4.7 Installing extra software

Both the GATK and MuTect software are used by OmicsPipe, but they require licenses from The Broad Institute and cannot be distributed with the OmicsPipe software. GATK and MuTect are free to download after accepting the license agreement.

To install GATK:

1. Download GATK
2. Upload the GenomeAnalysisTK.jar file to the /root/.local/easybuild/software/gatk/3.2-2 using either Webmin or StarCluster put
3. Make the jar file executable by running the command:

```
chmod +x /root/.local/easybuild/software/gatk/3.2-2/GenomeAnalysisTK.jar
```

To install MuTect:

1. Download MuTect
2. Upload the muTect-1.1.4.jar file to the /root/.local/easybuild/software/mutect/1.1.4 using either Webmin or StarCluster put
3. Make the jar file executable by running the command:

```
chmod +x /root/.local/easybuild/software/mutect/1.1.4/muTect-1.1.4.jar
```

Adding software that OmicsPipe was not built with might require a little more configuration, but OmicsPipe is designed as a foundation to which new software can be added. New software can obviously be added in any manner that the user prefers, but to follow the structure that was used to build OmicsPipe, please refer to the "custombuild" scripts.

Important:

- If you configure software that you think extends the functionality of OmicsPipe, please create a pull request on our Bitbucket page.

4.8 To build your own docker image

1. Download docker.io following the instructions at Get-Docker
2. Run the command:

```
docker build -t <Repository Name> https://bitbucket.org/sulab/omics_pipe/downloads/Dockerfile_AW
```

This will store the dockercluster image in the Repository Name of your choice.

There is also an AWS_custombuild Dockerfile, which can be used to build an Amazon Machine Image from scratch

4.9 Add storage > 1TB to the cluster using LVM (for advanced users)

1. Within StarCluster create x new volumes by running:

```
nvolumes=2 #number of volumes
vsize=1000 #in gb
instance=`curl -s http://169.254.169.254/latest/meta-data/instance-id`
akey=<AWS KEY>
skey=<AWS SECRET KEY>
region=us-west-1
zone=us-west-1a

for x in $(seq 1 $nvolumes)
do
  ec2-create-volume \
      --aws-access-key $akey \
      --aws-secret-key $skey \
      --size $vsize \
      --region $region \
      --availability-zone $zone
done > /tmp/vols.txt
```

2. Attach the volumes to the head node:

```
i=0
for vol in $(awk '{print $2}' /tmp/vols.txt)
do
      i=$(( i + 1 ))
      ec2-attach-volume $vol \
      -O $akey \
      -W $skey \
      -i $instance \
      --region $region \
      -d /dev/sdh${i}
done > /tmp/attach.txt
```

3. Mark the EBS volumes as physical volumes:

```
for i in $(find /dev/xvdh*)
do
     pvcreate $i
done
```

4. Create a volume group:

```
vgcreate vg /dev/xvdh*
```

5. Create a logical volume:

```
lvcreate -l100%VG -n lv vg
```

6. Create the file system:

```
mkfs -t xfs /dev/vg/lv
```

8. Mount the file system:

```
mount /dev/vg/lv /data/data_large
```

9. Create mount point and mount the device:

```
mkdir /data/data_large
mount /dev/md0 /data/data_large
```

10. Add new mountpoint to /etc/exports:

```
for x in $(qconf -sh | tail -n +2)
do
      echo '/data/data_large' ${x}'(async,no_root_squash,no_subtree_check,rw)' >> /etc/exports
done
```

11. Reload /etc/exports:

```
exportfs -a
```

12. Mount the new folder on all nodes:

```
for x in $(qconf -sh | tail -n +2)
do
      ssh $x 'mkdir /data/data_large'
      ssh $x 'mount -t nfs master:/data/data_large /data/data_large'
done
```

How to increase volume size?

1. Create and attach EBS volumes as described in steps 1. & 2. and then create the additional physical volumes:

```
for i in $(cat /tmp/attach.txt  | cut -f 4 | sed 's/[^0-9]*//g')
do
       pvcreate /dev/xvdh${i}
done
```

2. Add new volumes to the volume group:

```
for i in $(cat /tmp/attach.txt  | cut -f 4 | sed 's/[^0-9]*//g')
do
       vgextend vg /dev/xvdh${i}
done

lvextend -l100%VG /dev/mapper/vg-lv
```

3. Grow the file system to the new size:

```
xfs_growfs /data/data_large
```

4.10 Add storage > 1TB to the cluster using RAID 0 (for advanced users)

1. Within StarCluster create x new volumes by running:

```
nvolumes=2 #number of volumes
vsize=1000 #in gb
instance=`curl -s http://169.254.169.254/latest/meta-data/instance-id`
akey=<AWS KEY>
skey=<AWS SECRET KEY>
region=us-west-1
zone=us-west-1a

for x in $(seq 1 $nvolumes)
do
  ec2-create-volume \
      --aws-access-key $akey \
      --aws-secret-key $skey \
      --size $vsize \
      --region $region \
      --availability-zone $zone
done > /tmp/vols.txt
```

2. Attach the volumes to the head node:

```
i=0
for vol in $(awk '{print $2}' /tmp/vols.txt)
do
      i=$(( i + 1 ))
      ec2-attach-volume $vol \
      -O $akey \
      -W $skey \
      -i $instance \
      --region $region \
      -d /dev/sdh${i}
done
```

3. Create a raid 0 volume:

```
mdadm --create -l 0 -n $nvolumes /dev/md0 /dev/xvdh*
```

4. Create a file system:

```
mkfs -t ext4 /dev/md0
```

5. Create mount point and mount the device:

```
mkdir /data/data_large
mount /dev/md0 /data/data_large
```

6. Add new mountpoint to /etc/exports:

```
for x in $(qconf -sh | tail -n +2)
do
      echo '/data/data_large' ${x}'(async,no_root_squash,no_subtree_check,rw)' >> /etc/exports
done
```

7. Reload /etc/exports:

```
exportfs -a
```

8. Mount the new folder on all nodes:

```
for x in $(qconf -sh | tail -n +2)
do
```

```
        ssh $x 'mkdir /data/data_large'
        ssh $x 'mount -t nfs master:/data/data_large /data/data_large'
done
```

4.11 Backing up your data to S3

1. Run:

   ```
   s3cmd --configure
   ```

and follow the instructions

2. Create a S3 bucket:

   ```
   s3cmd mb s3://backup
   ```

3. Copy data to the bucket:

   ```
   s3cmd put -r /data/results s3://backup
   ```

More info on s3cmd here: https://github.com/s3tools/s3cmd

CHAPTER 5

Omics Pipe Tutorial – Installation

5.1 Installation

Installation instructions

5.2 Before Running Omics Pipe: Configuring Parameters File

Note: Before running omics_pipe, you must configure the parameters file, which is a YAML document. Follow the instructions here: *Configuring the parameters file*

5.3 Running Omics Pipe

When you are ready to run omics pipe, simply type the command:

```
omics_pipe RNAseq_count_based /path/to/parameter_file.yaml
```

To run the basic RNAseq_count_based pipeline with your parameter file. Additional usage instructions below and are available by typing omics_pipe –h.:

```
omics_pipe [-h] [--custom_script_path CUSTOM_SCRIPT_PATH]
           [--custom_script_name CUSTOM_SCRIPT_NAME]
                          [--compression {gzip, bzip}]
           {RNAseq_Tuxedo, RNAseq_count_based, RNAseq_cancer_report, RNAseq_TCGA, RNAseq_TCGA_counts,
                          Tumorseq_MUTECT, miRNAseq_count_based, miRNAseq_tuxedo, WES_GATK, WGS_GATK,
           parameter_file
```

If your .fastq files are compressed, please use the compression option and indicate the type of compression used for your files. Currently supported compression types are gzip and bzip.

5.4 Running Omics Pipe with the Test Data and Parameters

To run Omics Pipe with the test parameter files and data, type the commands below to run each pipeline.

Note: Replace the ~ with the path to your Omics Pipe installation.

RNA-seq (Tuxedo):

```
omics_pipe RNAseq_Tuxedo ~/tests/test_params_RNAseq_Tuxedo.yaml
```

RNA-seq(Anders 2013):

```
omics_pipe RNAseq_count_based ~/tests/test_params_RNAseq_counts.yaml
```

Whole Exome Sequencing (GATK):

```
omics_pipe WES_GATK ~/tests/test_params_WES_GATK.yaml
```

Whole Genome Sequencing (GATK):

```
omics_pipe WGS_GATK ~/tests/test_params_WGS_GATK.yaml
```

Whole Genome Sequencing (MUTECT):

```
omics_pipe Tumorseq_MUTECT ~/tests/test_params_MUTECT.yaml
```

ChIP-seq (MACS):

```
omics_pipe ChIPseq_MACS ~/tests/test_params_MACS.yaml
```

ChIP-seq (HOMER):

```
omics_pipe ChIPseq_HOMER ~/tests/test_params_HOMER.yaml
```

Breast Cancer Personalized Genomics Report- RNAseq:

```
omics_pipe RNAseq_cancer_report ~/tests/test_params_RNAseq_cancer.yaml
```

TCGA Reanalysis Pipeline - RNAseq:

```
omics_pipe RNAseq_TCGA ~/tests/test_params_RNAseq_TCGA.yaml
```

TCGA Reanalysis Pipeline - RNAseq Counts:

```
omics_pipe RNAseq_TCGA_counts ~/tests/test_params_RNAseq_TCGA_counts.yaml
```

miRNAseq Counts (Anders 2013):

```
omics_pipe miRNAseq_count_based ~/tests/test_params_miRNAseq_counts.yaml
```

miRNAseq (Tuxedo):

```
omics_pipe miRNAseq_tuxedo ~/tests/test_params_miRNAseq_Tuxedo.yaml
```

CHAPTER 6

Omics Pipe Tutorial – Configuring the Parameter File

Before running Omics Pipe, you must configure the parameters file, which is a YAML document. Example parameters files are located within the omics_pipe/test folder for each pipeline. Copy one of these parameters files into your working directory, and edit the parameters to work with your sample names, directory structure, software options and software versions. Make sure to keep the formatting and parameter names exactly the same as in the example parameters files.

Note: Make sure to follow the YAML format exactly. Ensure that there is only one space after each colon.

Note: For parameters in quotes in the test parameters file, please make sure to keep them in quotes in your custom parameter file.

The STEP parameter should be the function name of the last step in the pipeline that you want to run (e.g. run_tophat). To run the pre-installed pipelines all the way through, this should be "last_function."

Warning: Do not change the STEPS or STEPS_DE parameters for a pre-installed pipeline.

Note: Fastq files: paired end: 2 files, "Name_1.fastq" and "Name_2.fastq" representing read 1 and read 2. Have all fastq files in same raw data folder

6.1 Example Omics Pipe Parameter File

test_params.yaml in omics_pipe/tests:

```
SAMPLE_LIST: [test1, test2, test3]

STEP: last_function

STEPS: [fastqc, star, htseq, last_function]

RAW_DATA_DIR: /gpfs/home/kfisch/scripts/omics_pipeline-devel/tests

FLAG_PATH: /gpfs/home/kfisch/scripts/omics_pipeline-devel/tests/test_run/flags

HTSEQ_RESULTS: /gpfs/home/kfisch/scripts/omics_pipeline-devel/tests/test_run/counts

LOG_PATH: /gpfs/home/kfisch/scripts/omics_pipeline-devel/tests/test_run/logs
```

```
QC_PATH: /gpfs/home/kfisch/scripts/omics_pipeline-devel/tests/test_run

RESULTS_PATH: /gpfs/home/kfisch/test

STAR_RESULTS: /gpfs/home/kfisch/scripts/omics_pipeline-devel/tests/test_run/star

WORKING_DIR: /gpfs/home/kfisch/scripts/omics_pipeline-devel/omics_pipe/scripts

REPORT_RESULTS: /gpfs/home/kfisch/scripts/omics_pipeline-devel/tests/test_run

ENDS: SE

FASTQC_VERSION: '0.10.1'

GENOME: /gpfs/group/databases/Homo_sapiens/UCSC/hg19/Sequence/WholeGenomeFasta/genome.fa

HTSEQ_OPTIONS: -m intersection-nonempty -s no -t exon

PIPE_MULTIPROCESS: 100

PIPE_REBUILD: 'True'

PIPE_VERBOSE: 5

REF_GENES: /gpfs/group/databases/Homo_sapiens/UCSC/hg19/Annotation/Genes/genes.gtf

RESULTS_EMAIL: kfisch@scripps.edu

STAR_INDEX: /gpfs/group/databases/Homo_sapiens/UCSC/hg19/star_genome

STAR_OPTIONS: --readFilesCommand cat --runThreadN 8 --outSAMstrandField intronMotif --outFilterIntron

STAR_VERSION: '2.3.0'

TEMP_DIR: /scratch/kfisch

QUEUE: workq

USERNAME: kfisch

DRMAA_PATH: /opt/applications/pbs-drmaa/current/gnu/lib/libdrmaa.so

DPS_VERSION: '1.3.1111'

BAM_FILE_NAME: Aligned.out.bam

PARAMS_FILE: '/gpfs/home/kfisch/scripts/omics_pipeline-devel/tests/test_params_RNAseq_counts.yaml'

DESEQ_META: /gpfs/home/kfisch/scripts/omics_pipeline-devel/tests/counts_meta.csv

DESIGN: '~ condition'

PVAL: '0.05'

DESEQ_RESULTS: /gpfs/home/kfisch/scripts/omics_pipeline-devel/tests/test_run/DESEQ

SUMATRA_DB_PATH: /gpfs/home/kfisch/sumatra
```

```
SUMATRA_RUN_NAME: test_counts_sumatra_project

REPOSITORY: https://kfisch@bitbucket.org/sulab/omics_pipe

HG_USERNAME: Kathleen Fisch <kfisch@scripps.edu>
```

6.2 Explanation of Variables in Omics Pipe Parameter File

These parameters are for the RNAseq Count Based Pipeline. Parameters vary by pipeline. See examples in the tests/ folder.

test_params.yaml in omics_pipe/tests:

```
#sample names ie "Name" for paired and single end reads. So, "Name" for paired-end would expect two f
SAMPLE_LIST: [test1, test2, test3]

#Function to be run within pipeline. If you want to run the whole pipeline, leave this as last_functi
STEP: last_function

#All steps within the pipeline. Do not change this parameter for pre-installed pipelines. If you crea
STEPS: [fastqc, star, htseq, last_function]

#Directory where your raw .fastq files are located.
RAW_DATA_DIR: /gpfs/home/kfisch/scripts/omics_pipeline-devel/tests

#Directory where you would like to have the flag files created. Flag files are empty files that indic
FLAG_PATH: /gpfs/home/kfisch/scripts/omics_pipeline-devel/tests/test_run/flags

#Directory for HTSEQ results
HTSEQ_RESULTS: /gpfs/home/kfisch/scripts/omics_pipeline-devel/tests/test_run/counts

#Directory where log files will be written
LOG_PATH: /gpfs/home/kfisch/scripts/omics_pipeline-devel/tests/test_run/logs

#Directory for QC results
QC_PATH: /gpfs/home/kfisch/scripts/omics_pipeline-devel/tests/test_run

#Upper level results directory. Sumatra will check all subfolders of this directory for new files to
RESULTS_PATH: /gpfs/home/kfisch/test

#Directory where STAR results will be written
STAR_RESULTS: /gpfs/home/kfisch/scripts/omics_pipeline-devel/tests/test_run/star

#Where omics_pipe is installed, this path will be pointing to ~/omics_pipe/scripts.
WORKING_DIR: /gpfs/home/kfisch/scripts/omics_pipeline-devel/omics_pipe/scripts

#Directory for the summary report
REPORT_RESULTS: /gpfs/home/kfisch/scripts/omics_pipeline-devel/tests/test_run

#SE is single end, PE is paired-end sequencing reads
ENDS: SE

#Version number of FASTQC
FASTQC_VERSION: '0.10.1'

#Full path to Genome fasta file
```

```
GENOME: /gpfs/group/databases/Homo_sapiens/UCSC/hg19/Sequence/WholeGenomeFasta/genome.fa

#Options for HTSEQ
HTSEQ_OPTIONS: -m intersection-nonempty -s no -t exon

#Number of multiple processes you want Ruffus to spawn at once
PIPE_MULTIPROCESS: 100

#Ruffus parameter. No need to change.
PIPE_REBUILD: 'True'

#Ruffus parameter. No need to change.
PIPE_VERBOSE: 5

#Full path to reference gene annotations
REF_GENES: /gpfs/group/databases/Homo_sapiens/UCSC/hg19/Annotation/Genes/genes.gtf

#Your email.
RESULTS_EMAIL: kfisch@scripps.edu

#Directory pointing to STAR_INDEX (you may have to create this)
STAR_INDEX: /gpfs/group/databases/Homo_sapiens/UCSC/hg19/star_genome

#Options for STAR
STAR_OPTIONS: --readFilesCommand cat --runThreadN 8 --outSAMstrandField intronMotif --outFilterIntron

#Version number of STAR
STAR_VERSION: '2.3.0'

#Path to temporary directory
TEMP_DIR: /scratch/kfisch

#Name of the queue on your local cluster you wish to use
QUEUE: workq

#Username for local cluster
USERNAME: kfisch

#Path to your local cluster installation of DRMAA (ask your sys admin for this)
DRMAA_PATH: /opt/applications/pbs-drmaa/current/gnu/lib/libdrmaa.so

#Version number of Drug Pair Seeker
DPS_VERSION: '1.3.1111'

#Name of create Bam file. Will be Aligned.out.bam if you are using STAR and accepted_hits.bam if you
BAM_FILE_NAME: Aligned.out.bam

#Full path to your parameter file. Make sure to include the single quotes.
PARAMS_FILE: '/gpfs/home/kfisch/scripts/omics_pipeline-devel/tests/test_params_RNAseq_counts.yaml'

#Location of the meta data csv file for DESEQ. See tests/counts_meta.csv for an example.
DESEQ_META: /gpfs/home/kfisch/scripts/omics_pipeline-devel/tests/counts_meta.csv

#Design for DESEQ differential expression. Leave as is if you use the exact design as in the counts_m
DESIGN: '~ condition'

#P-value threshold
PVAL: '0.05'
```

```
#Directory for DESEQ results
DESEQ_RESULTS: /gpfs/home/kfisch/scripts/omics_pipeline-devel/tests/test_run/DESEQ

#Directory where you want to store your Sumatra database. Once you run this once, you do not have to
SUMATRA_DB_PATH: /gpfs/home/kfisch/sumatra

#Name of your project. You do not need to change this for subsequent runs of the pipeline, but you ca
SUMATRA_RUN_NAME: test_counts_sumatra_project

#Location of omics pipe repository (you can leave this)
REPOSITORY: https://kfisch@bitbucket.org/sulab/omics_pipe

#Your Mercurial username
HG_USERNAME: Kathleen Fisch <kfisch@scripps.edu>
```

CHAPTER 7

Omics Pipe Tutorial – Creating a Custom Pipeline Script

A pipeline script is a .py file that has the steps in the pipeline that you want to run in your analysis. To create a custom pipeline, you will create a new Python script (*.py) file and place it in your working directory (or wherever you want). The available analysis steps built-in to omics_pipe are available in the (List of currently available omics_pipe analysis steps).

You may add new modules directly to the module directory (see *Adding Custom Modules*), and they will become available steps that you can use in your custom pipeline.

There are three steps to creating a custom pipeline:

1. Designing the structure of your pipeline
2. Creating the script
3. Updating your parameters file

The section below details each of these steps.

7.1 Designing the structure of the pipeline

Omics_pipe depends upon the pipelining module Ruffus to handle the automation. Please read the documentation at the Ruffus website http://www.ruffus.org.uk/ for more information. To design your pipeline, you need to decide

- What analysis modules you want to run,
- What order you want the analysis modules to run in,
- Which, if any, of the analysis modules depend upon the results from another analysis module.

For example, we will create a custom pipeline that runs fastqc, star and htseq (depends on output from star).

7.2 Creating the script

To create the script, create a text file name custom_script.py (or whatever name you choose). At the top of the file, cut and copy this text:

```
#!/usr/bin/env python
from ruffus import *
import sys
import os
import time
import datetime
```

```
import drmaa
from omics_pipe.utils import *
from omics_pipe.parameters.default_parameters import default_parameters
p = Bunch(default_parameters)
os.chdir(p.WORKING_DIR)
now = datetime.datetime.now()
date = now.strftime("%Y-%m-%d %H:%M")
print p
for step in p.STEPS:
        vars()['inputList_' + step] = []
        for sample in p.SAMPLE_LIST:
                vars()['inputList_' + step].append([sample, "%s/%s_%s_completed.flag" % (p.FLAG_PATH,
        print vars()['inputList_' + step]
```

After this has been completed, you will need to import each of the analysis modules that you will use in your pipeline. For each analysis module, write the line below (fill in analysis_name with the name of the analysis module):

```
from omics_pipe.modules.analysis_name import analysis_name
```

In our example, you will have three lines (see below):

```
from omics_pipe.modules.fastqc import fastqc
from omics_pipe.modules.star import star
from omics_pipe.modules.htseq import htseq
```

Now you are ready to write the functions to run each of these steps in the analysis. For each step in our analysis pipeline, we will need to write a function. You can cut and copy these from the pre-packaged analysis pipeline scripts (or eventually a function reference) or create them. Each function needs to have two decorators from Ruffus: @parallel(inputList_analysis_name) to specify that the pipeline should run in parallel for more than one sample and @check_if_uptodate(check_file_exists) to call a function to check if that step in the pipeline is up to date. Name each function with the name of the analysis prefixed by "**run_**." The function input should always be (sample, analysis_name_flag). Within the function, you will call the analysis module that you loaded above. If you want an analysis module to run only after a module it depends upon finishes, you must add the @follows() Ruffus decorator before the function, with the name of the step that it depends upon. For example, if htseq needs to run after star, you would put @follows(run_star) above the run_htseq function. If you have steps that do not have functions that are dependent upon them, you can create a more complex pipeline structure by creating a "Last Function" that ties together all steps of your pipeline. The last function below is an example of such a function, and it also produces a PDF diagram of your pipeline when it completes. The functions for our example are below.

```
@parallel(inputList_fastqc)
@check_if_uptodate(check_file_exists)
def run_fastqc(sample, fastqc_flag):
        fastqc(sample, fastqc_flag)
        return

@parallel(inputList_star)
@check_if_uptodate(check_file_exists)
def run_star(sample, star_flag):
        star(sample, star_flag)
        return

@parallel(inputList_htseq)
@check_if_uptodate(check_file_exists)
@follows(run_star)
def run_htseq(sample, htseq_flag):
        htseq(sample, htseq_flag)
        return
```

```
@parallel(inputList_last_function)
@check_if_uptodate(check_file_exists)
@follows(run_fastqc, run_htseq)
def last_function(sample, last_function_flag):
        print "PIPELINE HAS FINISHED SUCCESSFULLY!!! YAY!"
        pipeline_graph_output = p.FLAG_PATH + "/pipeline_" + sample + "_" + str(date) + ".pdf"
        pipeline_printout_graph (pipeline_graph_output,'pdf', step, no_key_legend=False)
        stage = "last_function"
        flag_file = "%s/%s_%s_completed.flag" % (p.FLAG_PATH, stage, sample)
        open(flag_file, 'w').close()
        return
```

Once you have created all of the functions for each step of your pipeline, cut and copy the code below to the bottom of your script:

```
if __name__ == '__main__':

        pipeline_run(p.STEP, multiprocess = p.PIPE_MULTIPROCESS, verbose = p.PIPE_VERBOSE, gnu_make_r
```

At this point, please save your script and move on to step 3.

7.3 Updating your parameters file

In order for your script to run successfully, you need to configure your parameter file so that each analysis module has the necessary parameters to execute successfully. The full list of parameters for all modules in the current version of omics_pipe are located in the omics_pipe/parameters/default_parameters.py file (and eventually organized somewhere). You can view the list of necessary parameters for each analysis module by importing the analysis module into an interactive python session (from omics_pipe.modules.analysis_module import analysis_module) and typing analysis_module.__doc__. The parameters necessary for that analysis module will be listed under "parameters from parameters file." These parameters must be put into your parameters.yaml file and spelled exactly as shown (including all caps). Below is the list of parameters that are necessary to run omics_pipe in addition to the module specific parameters.

```
SAMPLE_LIST: [test, test1]
STEP: run_last_function
STEPS: [fastqc, star, htseq, last_function]
RAW_DATA_DIR: /gpfs/group/sanford/patient/SSKT/test_patient/RNA/RNA_seq/data
FLAG_PATH: /gpfs/group/sanford/patient/SSKT/test_patient/RNA/RNA_seq/logs/flags
LOG_PATH: /gpfs/group/sanford/patient/SSKT/test_patient/RNA/RNA_seq/logs
WORKING_DIR: /gpfs/home/kfisch/virt_env/virt2/lib/python2.6/site-packages/omics_pipe-1.0.7-py2.6.egg/
ENDS: PE
PIPE_MULTIPROCESS: 100
PIPE_REBUILD: 'True'
PIPE_VERBOSE: 5
RESULTS_EMAIL: kfisch@scripps.edu
TEMP_DIR: /scratch/kfisch
DPS_VERSION: '1.3.1111'
QUEUE: bigmem
PARAMS_FILE: /gpfs/home/kfisch/omics_pipe_docs/test_params.yaml
USERNAME: kfisch
GENOME: /gpfs/group/databases/Homo_sapiens/UCSC/hg19/Sequence/WholeGenomeFasta/genome.fa
CHROM: /gpfs/group/databases/Homo_sapiens/UCSC/hg19/Sequence/Chromosomes
REF_GENES: /gpfs/group/databases/Homo_sapiens/UCSC/hg19/Annotation/Genes/genes.gtf
STAR_INDEX: /gpfs/group/databases/Homo_sapiens/UCSC/hg19/star_genome
BOWTIE_INDEX: /gpfs/group/databases/Homo_sapiens/UCSC/hg19/Sequence/Bowtie2Index/genome
```

Once you have all of the necessary parameters in your parameters.yaml file, for your custom script you will need to change the STEP and STEPS parameters. In the STEP parameter, you will write the name of the last function in your pipeline that you want to run, which should be configured so that it captures all steps in the pipeline (as in the example above). Make sure to put **run_** in front of this, since you are calling the function, not the analysis module. In order for omics_pipe to know what steps you have in your pipeline, you need to list each analysis module name in the STEPS parameter separated with commas (without **run_** in the prefix). You are now ready to run your custom script.

Running omics_pipe with a custom pipeline script When you call the omics_pipe function, you will specify the path to your custom script using the command

```
omics_pipe custom  --custom_script_path ~/path/to/the/script -custom_script_name customscript /path/t
```

This will automatically load your custom script and run through the steps in your pipeline using the default modules available in omics_pipe.

CHAPTER 8

Omics Pipe Tutorial – Adding a New Module (Tool)

Users can easily create new analysis modules for use within omics_pipe. The user has two options for creating new analysis modules: - Adding analysis modules directly within the omics_pipe/scripts installation directory - Creating a new working directory where all analysis modules scripts are located (this can be changed in the parameters file by changing the WORKING_DIR parameter to the desired location). If you want to use option 2, in order to use pre-installed analysis modules, for the time being you must copy these analysis modules to your new working directory. If you choose option 1, you can simply add additional analysis modules and they will be accessible along with the pre-installed analysis modules.

To create a new analysis module, you need to perform four steps: 1. Create a Bash script with the command to be sent to the cluster 2. Create a Python module that calls the Bash script 3. Add your module to your custom pipeline 4. Add new module parameters to parameters file

The section below details each of these steps.

8.1 1. Create a Bash script

The first step in creating your custom module is to create the Bash script with the command you would like to run. If you are unsure how to write a Bash script, you can look at the examples in omics_pipe/scripts or work through this tutorial (http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html). In many cases, this will be a simple script with a one line command to call the analysis program. You should name your script something that will be easily identifiable and it should have the suffix .sh (e.g. analysis_script.sh). At the beginning of your analysis script, you should put the following lines:

```
#!/bin/bash
set -x
#Source modules for current shell
source $MODULESHOME/init/bash
#Make output directory if it doesn't exist
mkdir -p ${variable} #RESULTS_DIR
#Move tmp dir to scratch
export TMPDIR=${variable} #TEMP_DIR
#Load specified software version
module load fastqc/${variable} #VERSION
```

The ${variable} will be changed to ${number} (e.g. $1) based on the location of the variable in the input script (more on this below). These settings are assuming you are working on a cluster with a modular structure. If not, "module load" may not be appropriate to load the software, so please ask your system administrator to provide assistance with this if your cluster has a different system. After you specify the software and other configuration variables, you can write the commands for the software you would like to use. When you are finished with the commands, exit the script with 'exit 0.' An example script for running the software program FASTQC is below.

```
#Runs fastqc with $1=SAMPLE, $2=RAW_DATA_DIR, $3=QC_PATH
fastqc -o $3 $2/$1.fastq

exit 0
```

Substitute all variables that you would like to change from the parameter file with a variable notation, in the form of $1, $2, $3, etc for the first, second, third, etc input parameter that will be passed to the script. Once you have appropriately parameterized the script, save the script either in your working directory (along will all the other scripts you will need, possibly copied from omics_pipe/scripts) or in the omics_pipe/scripts directory.

8.2 2. Create a Python module

Now that you have created your custom script, you can create the Python module that will handle that script and schedule a job on the compute cluster using DRMAA (https://code.google.com/p/drmaa-python/wiki/Tutorial). You should name the Python module the same name as your custom analysis module, but with the extension .py. In this example, your Python module would be named analysis_script.py and the function within it would also be called analysis_script. Save your custom Python module within the same directory as your custom pipeline script. At the top of your Python module, cut and copy the text below.

```
#!/usr/bin/env python

import drmaa
from omics_pipe.parameters.default_parameters import default_parameters
from omics_pipe.utils import *
p = Bunch(default_parameters)

You will then write a simple Python function that take the form of the function below. You can direct
this function and then change the necessary names/parameters to fit your custom analysis.  ::

def fastqc(sample, fastqc_flag):
        '''QC check of raw .fastq files using FASTQC
                input: .fastq file
                output: folder and zipped folder containing html, txt and image files
                citation: Babraham Bioinformatics
                link: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
                parameters from parameters file: RAW_DATA_DIR,QC_PATH, FASTQC_VERSION'''
                  spawn_job(jobname = 'fastqc', SAMPLE = sample, LOG_PATH = p.LOG_PATH, RESULTS_EMAI
                job_status(jobname = 'fastqc', resultspath = p.QC_PATH, SAMPLE = sample, outputfilena
        return
```

Name your function the same as the names of both the Bash and Python scripts you just created for consistency. In our example, the first line would look like: "def analysis_script(sample, analysis_script_flag):". As you can see, I changed the name of the function as well as the name of the flag input file. The document string should be change to describe what your analysis module does, what type of input file it takes, a citation and link to the tool that you are calling, as well as the parameters that are needed in the parameters file that will be passed to the Bash script that you created. After you are done documenting your function, you will change a few items within the spawn_job and job_status functions that are called from the omics_pipe.utils module. In the spawn_job function, you should change the job name to match the name of your function, you can customize the resources your job will request from the compute cluster, you will need to change the name of the script to match that of the Bash script that you just created, and then you will change the parameters listed in the variable "args_list." The variable "sample" is lower case because it is passed to this function from omics_pipe, but input parameters coming from the parameters file must be prefixed with "p." List the parameters that you need to feed into your custom analysis script in the order that you numbered them in the Bash script. In the example above, $1 corresponds to 'sample' $2 corresponds to p.RAW_DATA_DIR, etc. Once you have the spawn_job function updated, you will update the job_status function with the job name, results path and a name of an output file that will be produced from your Bash script. This can be any file that is created. This

function will check that this file exists in the specified results directory, check that its size is greater than zero, and then it will create a flag file if it exists. Once you complete this, you are finished creating your custom Python module.

8.3 3. Add custom Python module to your custom pipeline

In order to use your custom analysis module, you will need to create a custom pipeline with your custom analysis module included as a step in the pipeline. For a tutorial on how to create a custom pipeline, see Section "Creating a Custom Pipeline Script." Once you have a custom pipeline script, please make sure your custom analysis module and custom pipeline script are in the same directory.

8.4 4. Add new parameters to parameters file

The final step in custom analysis module creation is to add the parameters necessary for your custom analysis module to run into the parameters file. Simply add the parameters to your parameters script, save it, and then run your custom pipeline.

CHAPTER 9

Omics Pipe Available Pipelines

9.1 RNA-seq (Tuxedo)

RNA-seq Tuxedo Modules Modules included in the Tuxedo RNA-seq pipeline.

- FASTQC
- TopHat
- Cufflinks
- Cuffmerge
- Cuffmergetocompare
- Cuffdiff
- R Summary Report - CummeRbund

9.2 RNA-seq(Anders 2013)

RNA-seq Count Based Modules Modules included in the count-based RNA-seq pipeline.

- FASTQC
- STAR
- HTSEQ
- R Summary Report - DESEQ2

9.3 Whole Exome Sequencing (GATK)

Whole Genome and Whole Exome Sequencing Modules Modules included in the whole exome sequencing pipeline.

- FASTQC
- BWA-MEM
- PICARD Mark Duplicates
- GATK Preprocessing
- GATK Variant Discovery

- GATK Variant Filtering

9.4 Whole Genome Sequencing (GATK)

Whole Genome and Whole Exome Sequencing Modules Modules included in the whole genome sequencing pipeline.

- FASTQC
- BWA-MEM
- PICARD Mark Duplicates
- GATK Preprocessing
- GATK Variant Discovery
- GATK Variant Filtering

9.5 Whole Genome Sequencing (MUTECT)

Whole Genome Sequencing (MUTECT) Modules included in the cancer (paired tumor/normal) whole genome sequencing pipeline.

- FASTQC
- BWA-MEM
- MUTECT

9.6 ChIP-seq (MACS)

ChIP-seq Modules – MACS Modules included in the ChIP-seq MACS pipeline.

- FASTQC
- Homer ChIP Trim
- Bowtie
- MACS

9.7 ChIP-seq (HOMER)

ChIP-seq Modules – HOMER Modules included in the ChIP-seq HOMER pipeline.

- FASTQC
- Homer ChIP Trim
- Bowtie
- Homer Read Density
- Homer Peaks
- Homer Peak Track

- Homer Annotate Peaks
- Homer Find Motifs

9.8 Breast Cancer Personalized Genomics Report- RNAseq

Breast Cancer Personalized Genomics Report- RNAseq Modules included in the RNAseq Cancer pipeline.

- FASTQC
- STAR
- RSEQC
- Fusion Catcher
- BWA/SNPiR
- Filter Variants
- HTseq
- Intogen
- OncoRep Cancer Report

9.9 TCGA Reanalysis Pipeline - RNAseq

TCGA Reanalysis Pipeline - RNAseq Modules included in the RNAseq Cancer pipeline.

- TCGA Download (GeneTorrent)
- FASTQC
- STAR
- RSEQC
- Fusion Catcher
- BWA/SNPiR
- Filter Variants
- HTseq
- Intogen
- OncoRep Cancer Report

9.10 TCGA Reanalysis Pipeline - RNAseq Counts

RNA-seq Count Based Modules- TCGA Modules included in the RNAseq counts pipeline for TCGA reanalysis.

- TCGA Download (GeneTorrent)
- FASTQC
- STAR
- HTSEQ

- Report

9.11 miRNAseq Counts (Anders 2013)

miRNA-seq Count Based Modules Modules included in the miRNAseq counts pipeline.

- Cutadapt
- FASTQ Length Filter
- FASTQC
- STAR
- HTSEQ
- Report

9.12 miRNAseq (Tuxedo)

miRNA-seq Tuxedo Modules Modules included in the miRNAseq Tuxedo pipeline.

- Cutadapt
- FASTQ Length Filter
- TopHat
- Cufflinks
- Cuffmerge
- Cuffmergetocompare
- Cuffdiff
- R Summary Report

9.13 All Available Modules

All Available Modules

CHAPTER 10

Reference Databases Needed

To run the pipelines, you will need to have reference databases installed on your cluster. If you are using the AWS installation, these databases are provided for you. If you need to install your references, please install the ones below. Omics Pipe is compatible with all species genome files. Examples below are for hg19, but you can substitute them for the equivalent files from other species.

10.1 All Pipelines

10.1.1 Genome

- .fa file can be downloaded from: http://cufflinks.cbcb.umd.edu/igenomes.html

10.1.2 Reference Annotation Files

You can use any reference annotations you would like, as long as they are GTF files.

Examples include:

- gencode.v18.annotation.gtf
- UCSC genes.gtf

10.2 Reference Data for Cancer Reporting Scripts (RNAseq cancer, TCGA pipelines)

For the cancer pipelines, please download the file from the link below, extract it and put the files in the respective directories. Reporting_data

10.2.1 In your omics pipe installation directory under omics_pipe/scripts/reporting/ref place the files.

- tcga_brca.R
- brca.txt <http://sulab.scripps.edu/kfisch/omics_pipe/ref/brca.txt>‘_

10.2.2 In your omics pipe installation directory under omics_pipe/scripts/reporting/data place the remaining files.

- brca_mol_class/*
- DoG/*
- geneLists/*
- SPIA
- deseq.tcga_brca.Rdata
- loggeoameansBRCA.Rdata

10.3 References for Variants (RNA-seq cancer, RNA-seq cancer TCGA, WES and WGS pipelines)

For pipelines performing variant calling, please download the references below and put them in the specified directories.

10.3.1 You can put these files in any directory. You will point to their location in the parameters file.

- dbNSFP2.0.txt
- common_no_known_medical_impact_00-latest.vcf

Available within the GATK recource bundle v.2.5:

- dbsnp_137.hg19.vcf
- Mills_and_1000G_gold_standard.indels.hg19.vcf
- 1000G_phase1.indels.hg19.vcf
- hapmap_3.3.hg19.vcf
- 1000G_omni2.5.hg19.vcf

10.3.2 In your omics pipe installation directory under omics_pipe/scripts/reporting/ref place these files.

- cadd.tsv.gz from http://cadd.gs.washington.edu/download
- drugbank.tsv
- cosmic.tsv
- clinvar.txt

from PharmGKB:

- pharmgkbAllele.tsv
- pharmgkbRSID.csv

10.4 WES Pipeline

- truseq_exome_targeted_regions.hg19.bed

10.5 ChIP-seq Pipelines

- hg19.chrom.sizes

10.6 SNPiR Pipelines (RNA-seq cancer and RNA-seq cancer TCGA pipelines)

- BWA Index
- RNA editing sites (Human_AG_all_hg19.bed)
- RepeatMasker.bed
- anno_combined_sorted
- knowngene.bed

CHAPTER 11

Third Party Software Dependencies

Omics Pipe is dependent upon several third-party software packages. Before running Omics Pipe, please install all of the required tools for the pipeline you will be running (see below) as Modules on your local cluster. If you are running the AWS distribution, all third party software is already installed.

11.1 R Packages Needed

In R, you can cut and copy this to install all required packages:

```
install.packages(c("bibtex",    "AnnotationDbi", "cluster", "cummeRbund", "data.table", "DBI", "DESe
"ggplot2", "graphite", "igraph", "KEGGREST","knitr", "knitrBootstrap",  "lattice", "locfit",    "pam
"plyr","RColorBrewer","Rcpp",   "RcppArmadillo", "RCurl", "ReactomePA", "RefManageR","RJSONIO","RSQL
"stringr","survival", "XML", "xtable",  "yaml"))
```

11.2 RNA-seq (Tuxedo)

FASTQC:	http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
TOPHAT:	http://tophat.cbcb.umd.edu/
CUFFLINKS:	http://cufflinks.cbcb.umd.edu/

11.3 RNA-seq (Anders 2013)

FASTQC:	http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
STAR:	http://code.google.com/p/rna-star/
SAMTOOLS:	http://samtools.sourceforge.net/
HTSEQ:	http://www-huber.embl.de/users/anders/HTSeq/doc/index.html

11.4 Whole Exome Sequencing (GATK)

FASTQC:	http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
BWA:	http://bio-bwa.sourceforge.net/
PICARD:	http://picard.sourceforge.net/
GATK:	https://www.broadinstitute.org/gatk/download

11.5 Whole Genome Sequencing (GATK)

FASTQC:	http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
BWA:	http://bio-bwa.sourceforge.net/
PICARD:	http://picard.sourceforge.net/
GATK:	https://www.broadinstitute.org/gatk/download

11.6 Whole Genome Sequencing (MUTECT)

FASTQC:	http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
BWA:	http://bio-bwa.sourceforge.net/
MUTECT:	http://www.broadinstitute.org/cancer/cga/mutect

11.7 ChIP-seq (MACS)

FASTQC:	http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
HOMER:	http://homer.salk.edu/homer/ngs/index.html
BOWTIE:	http://bowtie-bio.sourceforge.net/index.shtml
MACS:	http://liulab.dfci.harvard.edu/MACS/
BEDTOOLS:	https://github.com/arq5x/bedtools2
SAMTOOLS:	http://samtools.sourceforge.net/

11.8 ChIP-seq (HOMER)

FASTQC:	http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
HOMER:	http://homer.salk.edu/homer/ngs/index.html
BOWTIE:	http://bowtie-bio.sourceforge.net/index.shtml
BEDTOOLS:	https://github.com/arq5x/bedtools2
SAMTOOLS:	http://samtools.sourceforge.net/

11.9 Breast Cancer Personalized Genomics Report- RNAseq

FASTQC:	http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
STAR:	http://code.google.com/p/rna-star/
SAMTOOLS:	http://samtools.sourceforge.net/
HTSEQ:	http://www-huber.embl.de/users/anders/HTSeq/doc/index.html
RSEQC:	http://rseqc.sourceforge.net/
PICARD:	http://picard.sourceforge.net/
GATK:	https://www.broadinstitute.org/gatk/download
FusionCatcher:	https://code.google.com/p/fusioncatcher/
Oncofuse:	http://www.unav.es/genetica/oncofuse.html
BWA:	http://bio-bwa.sourceforge.net/
DNANEXUS SAMTOOLS:	https://github.com/dnanexus/samtools
BEDTOOLS:	https://github.com/arq5x/bedtools2
BLAT:	https://genome.ucsc.edu/FAQ/FAQblat.html#blat3
SNPiR:	http://lilab.stanford.edu/SNPiR/
SNPEFF:	http://snpeff.sourceforge.net/
SNPSIFT:	http://snpeff.sourceforge.net/SnpSift.html
VCFTOOLS:	http://vcftools.sourceforge.net/

11.10 TCGA Reanalysis Pipeline - RNAseq

GeneTorrent:	https://cghub.ucsc.edu/docs/user/software.html
FASTQC:	http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
STAR:	http://code.google.com/p/rna-star/
SAMTOOLS:	http://samtools.sourceforge.net/
HTSEQ:	http://www-huber.embl.de/users/anders/HTSeq/doc/index.html
RSEQC:	http://rseqc.sourceforge.net/
PICARD:	http://picard.sourceforge.net/
GATK:	https://www.broadinstitute.org/gatk/download
FusionCatcher:	https://code.google.com/p/fusioncatcher/
Oncofuse:	http://www.unav.es/genetica/oncofuse.html
BWA:	http://bio-bwa.sourceforge.net/
DNANEXUS SAMTOOLS:	https://github.com/dnanexus/samtools
BEDTOOLS:	https://github.com/arq5x/bedtools2
BLAT:	https://genome.ucsc.edu/FAQ/FAQblat.html#blat3
SNPiR:	http://lilab.stanford.edu/SNPiR/
SNPEFF:	http://snpeff.sourceforge.net/
SNPSIFT:	http://snpeff.sourceforge.net/SnpSift.html
VCFTOOLS:	http://vcftools.sourceforge.net/

11.11 TCGA Reanalysis Pipeline - RNAseq Counts

GeneTorrent:	https://cghub.ucsc.edu/docs/user/software.html
FASTQC:	http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
STAR:	http://code.google.com/p/rna-star/
SAMTOOLS:	http://samtools.sourceforge.net/
HTSEQ:	http://www-huber.embl.de/users/anders/HTSeq/doc/index.html

11.12 miRNAseq Counts (Anders 2013)

CutAdapt:	http://code.google.com/p/cutadapt/
FASTQC:	http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
STAR:	http://code.google.com/p/rna-star/
SAMTOOLS:	http://samtools.sourceforge.net/
HTSEQ:	http://www-huber.embl.de/users/anders/HTSeq/doc/index.html

11.13 miRNAseq (Tuxedo)

CutAdapt:	http://code.google.com/p/cutadapt/
FASTQC:	http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
TOPHAT:	http://tophat.cbcb.umd.edu/
CUFFLINKS:	http://cufflinks.cbcb.umd.edu/

CHAPTER 12

System Requirements

If you are running Omics Pipe on a local high performance compute cluster, please ensure that you have the following minimum resource requirements.

- A minimum of 2 processors (nodes) with at least 32GB of memory (the more nodes you have available, the more the pipeline can be parallelized)
- Scratch space that is at least 3x the size of your expected results files.
- Storage space available for ~200 GB of reference-related data
- Storage space available for raw data files
- Storage space for results files (~10x that of raw data)

CHAPTER 13

RNA-seq Tuxedo Modules

Modules available in the RNA-seq Tuxedo Pipeline.

13.1 FASTQC

omics_pipe.modules.fastqc.**fastqc**(*sample*, *fastqc_flag*)

QC check of raw .fastq files using FASTQC.

input: .fastq file

output: folder and zipped folder containing html, txt and image files

citation: Babraham Bioinformatics

link: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

parameters from parameters file: RAW_DATA_DIR:

QC_PATH:

FASTQC_VERSION:

COMPRESSION:

13.2 TopHat

omics_pipe.modules.tophat.**tophat**(*sample*, *tophat_flag*)

Runs TopHat to align .fastq files.

input: .fastq file

output: accepted_hits.bam

citation: Trapnell C, Pachter L, Salzberg SL. TopHat: discovering splice junctions with RNA-Seq. Bioinformatics doi:10.1093/bioinformatics/btp120

link: http://tophat.cbcb.umd.edu/

parameters from parameters file: RAW_DATA_DIR:

REF_GENES:

TOPHAT_RESULTS:

BOWTIE_INDEX:

TOPHAT_VERSION:

TOPHAT_OPTIONS:

BOWTIE_VERSION:

SAMTOOLS_VERSION:

13.3 Cufflinks

omics_pipe.modules.cufflinks.**cufflinks**(*sample*, *cufflinks_flag*)

Runs cufflinks to assemble .bam files from TopHat.

input: accepted_hits.bam

output: transcripts.gtf

citation: Trapnell C, et al. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation Nature Biotechnology doi:10.1038/nbt.1621

link: http://cufflinks.cbcb.umd.edu/

parameters from parameters file: TOPHAT_RESULTS:

CUFFLINKS_RESULTS:

REF_GENES:

GENOME:

CUFFLINKS_OPTIONS:

CUFFLINKS_VERSION:

13.4 Cuffmerge

omics_pipe.modules.cuffmerge.**cuffmerge**(*step*, *cuffmerge_flag*)

Runs cuffmerge to merge .gtf files from Cufflinks.

input: assembly_GTF_list.txt

output: merged.gtf

citation: Trapnell C, et al. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation Nature Biotechnology doi:10.1038/nbt.1621

link: http://cufflinks.cbcb.umd.edu/

parameters from parameters file: CUFFMERGE_RESULTS:

REF_GENES:

GENOME:

CUFFMERGE_OPTIONS:

CUFFLINKS_VERSION:

13.5 Cuffmergetocompare

omics_pipe.modules.cuffmergetocompare.**cuffmergetocompare**(*step*, *cuffmergetocompare_flag*)

Runs cuffcompare to annotate merged .gtf files from Cufflinks.

input: assembly_GTF_list.txt

output: merged.gtf

citation: Trapnell C, et al. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation Nature Biotechnology doi:10.1038/nbt.1621

link: http://cufflinks.cbcb.umd.edu/

parameters from parameters file: CUFFMERGE_RESULTS:

REF_GENES:

GENOME:

CUFFMERGETOCOMPARE_OPTIONS:

CUFFLINKS_VERSION:

13.6 Cuffdiff

omics_pipe.modules.cuffdiff.**cuffdiff**(*step*, *cuffdiff_flag*)

Runs Cuffdiff to perform differential expression. Runs after Cufflinks. Part of Tuxedo Suite.

input: .bam files

output: differential expression results

citation: Trapnell C, et al. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation Nature Biotechnology doi:10.1038/nbt.1621

link: http://cufflinks.cbcb.umd.edu/

parameters from parameters file: CUFFDIFF_RESULTS:

GENOME:

CUFFDIFF_OPTIONS:

CUFFMERGE_RESULTS:

CUFFDIFF_INPUT_LIST_COND1:

CUFFDIFF_INPUT_LIST_COND2:

CUFFLINKS_VERSION:

13.7 R Summary Report

omics_pipe.modules.RNAseq_report_tuxedo.**RNAseq_report_tuxedo**(*sample*, *RNAseq_report_tuxedo_flag*)

Runs R script with knitr to produce report from RNAseq pipeline.

input: results from other steps in RNAseq pipelines

output: html report

citation:

20. Meissner

parameters from parameter file: WORKING_DIR:

R_VERSION:

REPORT_RESULTS:

DPS_VERSION:

PARAMS_FILE:

CHAPTER 14

RNA-seq Count Based Modules

Modules available in the count-based RNA-seq Pipeline.

14.1 FASTQC

omics_pipe.modules.fastqc.**fastqc**(*sample*, *fastqc_flag*)

QC check of raw .fastq files using FASTQC.

input: .fastq file

output: folder and zipped folder containing html, txt and image files

citation: Babraham Bioinformatics

link: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

parameters from parameters file: RAW_DATA_DIR:

QC_PATH:

FASTQC_VERSION:

COMPRESSION:

14.2 STAR Aligner

omics_pipe.modules.star.**star**(*sample*, *star_flag*)

Runs STAR to align .fastq files.

input: .fastq file

output: Aligned.out.bam

citation:

1. Dobin et al, Bioinformatics 2012; doi: 10.1093/bioinformatics/bts635 "STAR: ultrafast universal RNA-seq aligner"

link: https://code.google.com/p/rna-star/

parameters from parameters file: ENDS:

RAW_DATA_DIR:

STAR_INDEX:

STAR_OPTIONS:

STAR_RESULTS:

SAMTOOLS_VERSION:

STAR_VERSION:

14.3 HTSEQ-count

omics_pipe.modules.htseq.**htseq**(*sample*, *htseq_flag*)

Runs htseq-count to get raw count data from alignments.

input: Aligned.out.sort.bam

output: counts.txt

citation: Simon Anders, EMBL

link: http://www-huber.embl.de/users/anders/HTSeq/doc/overview.html

parameters from parameters file: STAR_RESULTS:

HTSEQ_OPTIONS:

REF_GENES:

HTSEQ_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BAM_FILE_NAME:

PYTHON_VERSION:

14.4 R Summary Report - DESEQ2

omics_pipe.modules.RNAseq_report_counts.**RNAseq_report_counts**(*sample*, *RNAseq_report_counts_flag*)

Runs R script with knitr to produce report from RNAseq pipeline.

input: results from other steps in RNAseq pipelines

output: html report

citation:

20. Meissner

parameters from parameter file: WORKING_DIR:

R_VERSION:

REPORT_RESULTS:

PARAMS_FILE:

CHAPTER 15

Breast Cancer Personalized Genomics Report- RNAseq

Modules included in the RNAseq Cancer pipeline.

15.1 FASTQC

omics_pipe.modules.fastqc.**fastqc**(*sample*, *fastqc_flag*)

QC check of raw .fastq files using FASTQC.

input: .fastq file

output: folder and zipped folder containing html, txt and image files

citation: Babraham Bioinformatics

link: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

parameters from parameters file: RAW_DATA_DIR:

QC_PATH:

FASTQC_VERSION:

COMPRESSION:

15.2 STAR Aligner

omics_pipe.modules.star.**star**(*sample*, *star_flag*)

Runs STAR to align .fastq files.

input: .fastq file

output: Aligned.out.bam

citation:

1. Dobin et al, Bioinformatics 2012; doi: 10.1093/bioinformatics/bts635 "STAR: ultrafast universal RNA-seq aligner"

link: https://code.google.com/p/rna-star/

parameters from parameters file: ENDS:

RAW_DATA_DIR:

STAR_INDEX:

STAR_OPTIONS:

STAR_RESULTS:

SAMTOOLS_VERSION:

STAR_VERSION:

15.3 HTSEQ-count

omics_pipe.modules.htseq.**htseq**(*sample*, *htseq_flag*)

Runs htseq-count to get raw count data from alignments.

input: Aligned.out.sort.bam

output: counts.txt

citation: Simon Anders, EMBL

link: http://www-huber.embl.de/users/anders/HTSeq/doc/overview.html

parameters from parameters file: STAR_RESULTS:

HTSEQ_OPTIONS:

REF_GENES:

HTSEQ_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BAM_FILE_NAME:

PYTHON_VERSION:

15.4 RSEQC

omics_pipe.modules.rseqc.**rseqc**(*sample*, *rseqc_flag*)

Runs rseqc to determine insert size as QC for alignment.

input: .bam

output: pdf plot

link: http://rseqc.sourceforge.net/

parameters from parameters file: STAR_RESULTS:

QC_PATH:

BAM_FILE_NAME:

RSEQC_REF:

RSEQC_VERSION:

TEMP_DIR:

15.5 Fusion Catcher

omics_pipe.modules.fusion_catcher.**fusion_catcher**(*sample*, *fusion_catcher_flag*)

Detects fusion genes in paired-end RNAseq data.

input: paired end .fastq files

output: list of candidate fusion genes

citation:

19. Kangaspeska, S. Hultsch, H. Edgren, D. Nicorici, A. Murumgi, O.P. Kallioniemi, Reanalysis of RNA-sequencing data reveals several additional fusion genes with multiple isoforms, PLOS One, Oct. 2012. http://dx.plos.org/10.1371/journal.pone.0048745

link: https://code.google.com/p/fusioncatcher

parameters from parameters file: ENDS:

RAW_DATA_DIR:

FUSION_RESULTS:

FUSIONCATCHERBUILD_DIR:

TEMP_DIR:

SAMTOOLS_VERSION:

FUSIONCATCHER_VERSION:

FUSIONCATCHER_OPTIONS:

TISSUE:

PYTHON_VERSION:

15.6 BWA/SNPiR

15.6.1 BWA

omics_pipe.modules.bwa.**bwa1**(*sample*, *bwa1_flag*)

BWA aligner for read1 of paired_end reads.

input: .fastq

output: .sam

citation: Li H. and Durbin R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics, 25, 1754-1760. [PMID: 19451168]

link: http://bio-bwa.sourceforge.net/bwa.shtml

parameters from parameters file: BWA_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BWA_VERSION:

BWA_INDEX:

RAW_DATA_DIR:

GATK_READ_GROUP_INFO:

COMPRESSION:

omics_pipe.modules.bwa.**bwa2**(*sample*, *bwa2_flag*)
BWA aligner for read2 of paired_end reads.

input: .fastq

output: .sam

citation: Li H. and Durbin R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics, 25, 1754-1760. [PMID: 19451168]

link: http://bio-bwa.sourceforge.net/bwa.shtml

parameters from parameters file: BWA_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BWA_VERSION:

BWA_INDEX:

RAW_DATA_DIR:

GATK_READ_GROUP_INFO:

COMPRESSION:

omics_pipe.modules.bwa.**bwa_RNA**(*sample*, *bwa_flag*)
BWA aligner for single end reads.

input: .fastq

output: .sam

citation: Li H. and Durbin R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics, 25, 1754-1760. [PMID: 19451168]

link: http://bio-bwa.sourceforge.net/bwa.shtml

parameters from parameters file: BWA_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BWA_VERSION:

BWA_INDEX:

RAW_DATA_DIR:

GATK_READ_GROUP_INFO:

COMPRESSION:

15.6.2 SNPiR

omics_pipe.modules.snpir_variants.**snpir_variants**(*sample*, *snpir_variants_flag*)
Calls variants using SNPIR pipeline.

input: Aligned.out.sort.bam or accepted_hits.bam

output: final_variants.vcf file

citation: Piskol, R., et al. (2013). "Reliable Identification of Genomic Variants from RNA-Seq Data." The American Journal of Human Genetics 93(4): 641-651.

link: http://lilab.stanford.edu/SNPiR/

parameters from parameters file: VARIANT_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BWA_VERSION:

PICARD_VERSION:

GATK_VERSION:

BEDTOOLS_VERSION:

UCSC_TOOLS_VERSION:

GENOME:

REPEAT_MASKER:

SNPIR_ANNOTATION:

RNA_EDIT:

DBSNP:

MILLS:

G1000:

WORKING_DIR:

BWA_RESULTS:

SNPIR_VERSION:

SNPIR_CONFIG:

SNPIR_DIR:

ENCODING:

15.6.3 Filter Variants

omics_pipe.modules.filter_variants.**filter_variants**(*sample*, *filter_variants_flag*)

Filters variants to remove common variants.

input: .bam or .sam file

output: .vcf file

citation: Piskol et al. 2013. Reliable identification of genomic variants from RNA-seq data. The American Journal of Human Genetics 93: 641-651.

link: http://lilab.stanford.edu/SNPiR/

parameters from parameters file: VARIANT_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BWA_VERSION:

PICARD_VERSION:

GATK_VERSION:

BEDTOOLS_VERSION:

UCSC_TOOLS_VERSION:

GENOME:

REPEAT_MASKER:

SNPIR_ANNOTATION:

RNA_EDIT:

DBSNP:

MILLS:

G1000:

WORKING_DIR:

BWA_RESULTS:

SNPIR_VERSION:

SNPIR_CONFIG:

SNPIR_DIR:

SNPEFF_VERSION:

dbNSFP:

VCFTOOLS_VERSION:

WORKING_DIR:

SNP_FILTER_OUT_REF:

15.7 Intogen

omics_pipe.modules.intogen.**intogen**(*sample*, *intogen_flag*)

Runs Intogen to rank mutations and implication for cancer phenotype. Follows variant calling.

input: .vcf

output: variant list

citation: Gonzalez-Perez et al. 2013. Intogen mutations identifies cancer drivers across tumor types. Nature Methods 10, 1081-1082.

link: http://www.intogen.org/

parameters from parameter file: VCF_FILE:

INTOGEN_OPTIONS:

INTOGEN_RESULTS:

INTOGEN_VERSION:

USERNAME:

WORKING_DIR:

TEMP_DIR:

SCHEDULER:

VARIANT_RESULTS:

15.8 OncoRep Cancer Report

omics_pipe.modules.BreastCancer_RNA_report.**BreastCancer_RNA_report**(*sample*, *BreastCancer_RNA_report_flag*)

Runs R script with knitr to produce report from RNAseq pipeline.

input: results from other steps in RNAseq pipelines

output: html report

citation:

20. Meissner

parameters from parameter file: WORKING_DIR:

R_VERSION:

REPORT_RESULTS:

PARAMS_FILE:

TABIX_VERSION:

TUMOR_TYPE:

GENELIST:

COSMIC:

CLINVAR:

PHARMGKB_rsID:

PHARMGKB_Allele:

DRUGBANK:

CADD:

CHAPTER 16

TCGA Reanalysis Pipeline - RNAseq

Modules included in the TCGA RNAseq Cancer pipeline.

16.1 TCGA Download

omics_pipe.modules.TCGA_download.**TCGA_download**(*sample*, *TCGA_download_flag*)

Downloads and unzips TCGA data from Manifest.xml downloaded from CGHub. input:

TGCA XML file

output: downloaded files from TCGA

citation: The Cancer Genome Atlas

link: https://cghub.ucsc.edu/software/downloads.html

parameters from parameters file: TCGA_XML_FILE:

TCGA_KEY:

TCGA_OUTPUT_PATH:

CGATOOLS_VERSION:

16.2 FASTQC

omics_pipe.modules.fastqc.**fastqc**(*sample*, *fastqc_flag*)

QC check of raw .fastq files using FASTQC.

input: .fastq file

output: folder and zipped folder containing html, txt and image files

citation: Babraham Bioinformatics

link: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

parameters from parameters file: RAW_DATA_DIR:

QC_PATH:

FASTQC_VERSION:

COMPRESSION:

16.3 STAR Aligner

omics_pipe.modules.star.**star**(*sample*, *star_flag*)

Runs STAR to align .fastq files.

input: .fastq file

output: Aligned.out.bam

citation:

1. Dobin et al, Bioinformatics 2012; doi: 10.1093/bioinformatics/bts635 "STAR: ultrafast universal RNA-seq aligner"

link: https://code.google.com/p/rna-star/

parameters from parameters file: ENDS:

RAW_DATA_DIR:

STAR_INDEX:

STAR_OPTIONS:

STAR_RESULTS:

SAMTOOLS_VERSION:

STAR_VERSION:

16.4 HTSEQ-count

omics_pipe.modules.htseq.**htseq**(*sample*, *htseq_flag*)

Runs htseq-count to get raw count data from alignments.

input: Aligned.out.sort.bam

output: counts.txt

citation: Simon Anders, EMBL

link: http://www-huber.embl.de/users/anders/HTSeq/doc/overview.html

parameters from parameters file: STAR_RESULTS:

HTSEQ_OPTIONS:

REF_GENES:

HTSEQ_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BAM_FILE_NAME:

PYTHON_VERSION:

16.5 RSEQC

omics_pipe.modules.rseqc.**rseqc**(*sample*, *rseqc_flag*)

Runs rseqc to determine insert size as QC for alignment.

input: .bam

output: pdf plot

link: http://rseqc.sourceforge.net/

parameters from parameters file: STAR_RESULTS:

QC_PATH:

BAM_FILE_NAME:

RSEQC_REF:

RSEQC_VERSION:

TEMP_DIR:

16.6 Fusion Catcher

omics_pipe.modules.fusion_catcher.**fusion_catcher**(*sample*, *fusion_catcher_flag*)

Detects fusion genes in paired-end RNAseq data.

input: paired end .fastq files

output: list of candidate fusion genes

citation:

19. Kangaspeska, S. Hultsch, H. Edgren, D. Nicorici, A. Murumgi, O.P. Kallioniemi, Reanalysis of RNA-sequencing data reveals several additional fusion genes with multiple isoforms, PLOS One, Oct. 2012. http://dx.plos.org/10.1371/journal.pone.0048745

link: https://code.google.com/p/fusioncatcher

parameters from parameters file: ENDS:

RAW_DATA_DIR:

FUSION_RESULTS:

FUSIONCATCHERBUILD_DIR:

TEMP_DIR:

SAMTOOLS_VERSION:

FUSIONCATCHER_VERSION:

FUSIONCATCHER_OPTIONS:

TISSUE:

PYTHON_VERSION:

16.7 BWA/SNPiR

16.7.1 BWA

omics_pipe.modules.bwa.**bwa1**(*sample*, *bwa1_flag*)
BWA aligner for read1 of paired_end reads.

input: .fastq

output: .sam

citation: Li H. and Durbin R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics, 25, 1754-1760. [PMID: 19451168]

link: http://bio-bwa.sourceforge.net/bwa.shtml

parameters from parameters file: BWA_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BWA_VERSION:

BWA_INDEX:

RAW_DATA_DIR:

GATK_READ_GROUP_INFO:

COMPRESSION:

omics_pipe.modules.bwa.**bwa2**(*sample*, *bwa2_flag*)
BWA aligner for read2 of paired_end reads.

input: .fastq

output: .sam

citation: Li H. and Durbin R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics, 25, 1754-1760. [PMID: 19451168]

link: http://bio-bwa.sourceforge.net/bwa.shtml

parameters from parameters file: BWA_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BWA_VERSION:

BWA_INDEX:

RAW_DATA_DIR:

GATK_READ_GROUP_INFO:

COMPRESSION:

omics_pipe.modules.bwa.**bwa_RNA**(*sample*, *bwa_flag*)
BWA aligner for single end reads.

input: .fastq

output: .sam

citation: Li H. and Durbin R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics, 25, 1754-1760. [PMID: 19451168]

link: http://bio-bwa.sourceforge.net/bwa.shtml

parameters from parameters file: BWA_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BWA_VERSION:

BWA_INDEX:

RAW_DATA_DIR:

GATK_READ_GROUP_INFO:

COMPRESSION:

16.7.2 SNPiR

omics_pipe.modules.snpir_variants.**snpir_variants**(*sample*, *snpir_variants_flag*)

Calls variants using SNPIR pipeline.

input: Aligned.out.sort.bam or accepted_hits.bam

output: final_variants.vcf file

citation: Piskol, R., et al. (2013). "Reliable Identification of Genomic Variants from RNA-Seq Data." The American Journal of Human Genetics 93(4): 641-651.

link: http://lilab.stanford.edu/SNPiR/

parameters from parameters file: VARIANT_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BWA_VERSION:

PICARD_VERSION:

GATK_VERSION:

BEDTOOLS_VERSION:

UCSC_TOOLS_VERSION:

GENOME:

REPEAT_MASKER:

SNPIR_ANNOTATION:

RNA_EDIT:

DBSNP:

MILLS:

G1000:

WORKING_DIR:

BWA_RESULTS:

SNPIR_VERSION:

SNPIR_CONFIG:

SNPIR_DIR:

ENCODING:

16.7.3 Filter Variants

omics_pipe.modules.filter_variants.**filter_variants**(*sample*, *filter_variants_flag*)

Filters variants to remove common variants.

input: .bam or .sam file

output: .vcf file

citation: Piskol et al. 2013. Reliable identification of genomic variants from RNA-seq data. The American Journal of Human Genetics 93: 641-651.

link: http://lilab.stanford.edu/SNPiR/

parameters from parameters file: VARIANT_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BWA_VERSION:

PICARD_VERSION:

GATK_VERSION:

BEDTOOLS_VERSION:

UCSC_TOOLS_VERSION:

GENOME:

REPEAT_MASKER:

SNPIR_ANNOTATION:

RNA_EDIT:

DBSNP:

MILLS:

G1000:

WORKING_DIR:

BWA_RESULTS:

SNPIR_VERSION:

SNPIR_CONFIG:

SNPIR_DIR:

SNPEFF_VERSION:

dbNSFP:

VCFTOOLS_VERSION:

WORKING_DIR:

SNP_FILTER_OUT_REF:

16.8 Intogen

omics_pipe.modules.intogen.**intogen**(*sample*, *intogen_flag*)

Runs Intogen to rank mutations and implication for cancer phenotype. Follows variant calling.

input: .vcf

output: variant list

citation: Gonzalez-Perez et al. 2013. Intogen mutations identifies cancer drivers across tumor types. Nature Methods 10, 1081-1082.

link: http://www.intogen.org/

parameters from parameter file: VCF_FILE:

INTOGEN_OPTIONS:

INTOGEN_RESULTS:

INTOGEN_VERSION:

USERNAME:

WORKING_DIR:

TEMP_DIR:

SCHEDULER:

VARIANT_RESULTS:

16.9 OncoRep Cancer Report

omics_pipe.modules.BreastCancer_RNA_report.**BreastCancer_RNA_report**(*sample*, *BreastCancer_RNA_report_flag*)

Runs R script with knitr to produce report from RNAseq pipeline.

input: results from other steps in RNAseq pipelines

output: html report

citation:

20. Meissner

parameters from parameter file: WORKING_DIR:

R_VERSION:

REPORT_RESULTS:

PARAMS_FILE:

TABIX_VERSION:

TUMOR_TYPE:

GENELIST:

COSMIC:

CLINVAR:

PHARMGKB_rsID:

PHARMGKB_Allele:

DRUGBANK:

CADD:

CHAPTER 17

RNA-seq Count Based Modules- TCGA

Modules available in the TCGA count-based RNA-seq Pipeline.

17.1 TCGA Download

omics_pipe.modules.TCGA_download.**TCGA_download**(*sample*, *TCGA_download_flag*)

Downloads and unzips TCGA data from Manifest.xml downloaded from CGHub. input:

TGCA XML file

output: downloaded files from TCGA

citation: The Cancer Genome Atlas

link: https://cghub.ucsc.edu/software/downloads.html

parameters from parameters file: TCGA_XML_FILE:

TCGA_KEY:

TCGA_OUTPUT_PATH:

CGATOOLS_VERSION:

17.2 FASTQC

omics_pipe.modules.fastqc.**fastqc**(*sample*, *fastqc_flag*)

QC check of raw .fastq files using FASTQC.

input: .fastq file

output: folder and zipped folder containing html, txt and image files

citation: Babraham Bioinformatics

link: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

parameters from parameters file: RAW_DATA_DIR:

QC_PATH:

FASTQC_VERSION:

COMPRESSION:

17.3 STAR Aligner

omics_pipe.modules.star.**star**(*sample*, *star_flag*)

Runs STAR to align .fastq files.

input: .fastq file

output: Aligned.out.bam

citation:

1. Dobin et al, Bioinformatics 2012; doi: 10.1093/bioinformatics/bts635 "STAR: ultrafast universal RNA-seq aligner"

link: https://code.google.com/p/rna-star/

parameters from parameters file: ENDS:

RAW_DATA_DIR:

STAR_INDEX:

STAR_OPTIONS:

STAR_RESULTS:

SAMTOOLS_VERSION:

STAR_VERSION:

17.4 HTSEQ-count

omics_pipe.modules.htseq.**htseq**(*sample*, *htseq_flag*)

Runs htseq-count to get raw count data from alignments.

input: Aligned.out.sort.bam

output: counts.txt

citation: Simon Anders, EMBL

link: http://www-huber.embl.de/users/anders/HTSeq/doc/overview.html

parameters from parameters file: STAR_RESULTS:

HTSEQ_OPTIONS:

REF_GENES:

HTSEQ_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BAM_FILE_NAME:

PYTHON_VERSION:

17.5 R Summary Report - DESEQ2

omics_pipe.modules.RNAseq_report_counts.**RNAseq_report_counts**(*sample*, *RNAseq_report_counts_flag*)

Runs R script with knitr to produce report from RNAseq pipeline.

input: results from other steps in RNAseq pipelines

output: html report

citation:

20. Meissner

parameters from parameter file: WORKING_DIR:

R_VERSION:

REPORT_RESULTS:

PARAMS_FILE:

CHAPTER 18

miRNA-seq Tuxedo Modules

Modules available in the miRNA-seq Tuxedo Pipeline.

18.1 CutAdapt

omics_pipe.modules.cutadapt_miRNA.**cutadapt_miRNA**(*sample*, *cutadapt_miRNA_flag*)
Runs Cutadapt to trim adapters from reads.

input: .fastq

output: .fastq

citation: Martin 2011. Cutadapt removes adapter sequences from high-throughput sequencing reads. EMBnet.journal 17: 10-12.

link: https://code.google.com/p/cutadapt/

parameters from parameters file: RAW_DATA_DIR:

ADAPTER:

TRIMMED_DATA_PATH:

PYTHON_VERSION

18.2 Fastq Length Filter

omics_pipe.modules.fastq_length_filter_miRNA.**fastq_length_filter_miRNA**(*sample*, *fastq_length_filter_miRNA_fl*
Runs custom Python script to filter miRNA reads by length.

input: .fastq

output: .fastq

parameters from parameter file: TRIMMED_DATA_PATH:

18.3 FASTQC

omics_pipe.modules.fastqc.**fastqc**(*sample*, *fastqc_flag*)
QC check of raw .fastq files using FASTQC.

input: .fastq file

output: folder and zipped folder containing html, txt and image files

citation: Babraham Bioinformatics

link: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

parameters from parameters file: RAW_DATA_DIR:

QC_PATH:

FASTQC_VERSION:

COMPRESSION:

18.4 TopHat

omics_pipe.modules.tophat.**tophat**(*sample*, *tophat_flag*)

Runs TopHat to align .fastq files.

input: .fastq file

output: accepted_hits.bam

citation: Trapnell C, Pachter L, Salzberg SL. TopHat: discovering splice junctions with RNA-Seq. Bioinformatics doi:10.1093/bioinformatics/btp120

link: http://tophat.cbcb.umd.edu/

parameters from parameters file: RAW_DATA_DIR:

REF_GENES:

TOPHAT_RESULTS:

BOWTIE_INDEX:

TOPHAT_VERSION:

TOPHAT_OPTIONS:

BOWTIE_VERSION:

SAMTOOLS_VERSION:

18.5 Cufflinks

omics_pipe.modules.cufflinks.**cufflinks**(*sample*, *cufflinks_flag*)

Runs cufflinks to assemble .bam files from TopHat.

input: accepted_hits.bam

output: transcripts.gtf

citation: Trapnell C, et al. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation Nature Biotechnology doi:10.1038/nbt.1621

link: http://cufflinks.cbcb.umd.edu/

parameters from parameters file: TOPHAT_RESULTS:

CUFFLINKS_RESULTS:

REF_GENES:

GENOME:

CUFFLINKS_OPTIONS:

CUFFLINKS_VERSION:

18.6 Cuffmerge

omics_pipe.modules.cuffmerge.**cuffmerge**(*step*, *cuffmerge_flag*)

Runs cuffmerge to merge .gtf files from Cufflinks.

input: assembly_GTF_list.txt

output: merged.gtf

citation: Trapnell C, et al. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation Nature Biotechnology doi:10.1038/nbt.1621

link: http://cufflinks.cbcb.umd.edu/

parameters from parameters file: CUFFMERGE_RESULTS:

REF_GENES:

GENOME:

CUFFMERGE_OPTIONS:

CUFFLINKS_VERSION:

18.7 Cuffmergetocompare

omics_pipe.modules.cuffmergetocompare.**cuffmergetocompare**(*step*, *cuffmergetocompare_flag*)

Runs cuffcompare to annotate merged .gtf files from Cufflinks.

input: assembly_GTF_list.txt

output: merged.gtf

citation: Trapnell C, et al. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation Nature Biotechnology doi:10.1038/nbt.1621

link: http://cufflinks.cbcb.umd.edu/

parameters from parameters file: CUFFMERGE_RESULTS:

REF_GENES:

GENOME:

CUFFMERGETOCOMPARE_OPTIONS:

CUFFLINKS_VERSION:

18.8 Cuffdiff

omics_pipe.modules.cuffdiff.**cuffdiff**(*step*, *cuffdiff_flag*)

Runs Cuffdiff to perform differential expression. Runs after Cufflinks. Part of Tuxedo Suite.

input: .bam files

output: differential expression results

citation: Trapnell C, et al. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation Nature Biotechnology doi:10.1038/nbt.1621

link: http://cufflinks.cbcb.umd.edu/

parameters from parameters file: CUFFDIFF_RESULTS:

GENOME:

CUFFDIFF_OPTIONS:

CUFFMERGE_RESULTS:

CUFFDIFF_INPUT_LIST_COND1:

CUFFDIFF_INPUT_LIST_COND2:

CUFFLINKS_VERSION:

18.9 R Summary Report

omics_pipe.modules.RNAseq_report_tuxedo.**RNAseq_report_tuxedo**(*sample*, *RNAseq_report_tuxedo_flag*)

Runs R script with knitr to produce report from RNAseq pipeline.

input: results from other steps in RNAseq pipelines

output: html report

citation:

20. Meissner

parameters from parameter file: WORKING_DIR:

R_VERSION:

REPORT_RESULTS:

DPS_VERSION:

PARAMS_FILE:

CHAPTER 19

miRNA-seq Count Based Modules

Modules available in the count-based miRNA-seq Pipeline.

19.1 CutAdapt

omics_pipe.modules.cutadapt_miRNA.**cutadapt_miRNA**(*sample*, *cutadapt_miRNA_flag*)

Runs Cutadapt to trim adapters from reads.

input: .fastq

output: .fastq

citation: Martin 2011. Cutadapt removes adapter sequences from high-throughput sequencing reads. EMBnet.journal 17: 10-12.

link: https://code.google.com/p/cutadapt/

parameters from parameters file: RAW_DATA_DIR:

ADAPTER:

TRIMMED_DATA_PATH:

PYTHON_VERSION

19.2 Fastq Length Filter

omics_pipe.modules.fastq_length_filter_miRNA.**fastq_length_filter_miRNA**(*sample*, *fastq_length_filter_miRNA_fl*

Runs custom Python script to filter miRNA reads by length.

input: .fastq

output: .fastq

parameters from parameter file: TRIMMED_DATA_PATH:

19.3 FASTQC

omics_pipe.modules.fastqc.**fastqc**(*sample*, *fastqc_flag*)

QC check of raw .fastq files using FASTQC.

input: .fastq file

output: folder and zipped folder containing html, txt and image files

citation: Babraham Bioinformatics

link: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

parameters from parameters file: RAW_DATA_DIR:

QC_PATH:

FASTQC_VERSION:

COMPRESSION:

19.4 STAR Aligner

omics_pipe.modules.star.**star**(*sample*, *star_flag*)

Runs STAR to align .fastq files.

input: .fastq file

output: Aligned.out.bam

citation:

1. Dobin et al, Bioinformatics 2012; doi: 10.1093/bioinformatics/bts635 "STAR: ultrafast universal RNA-seq aligner"

link: https://code.google.com/p/rna-star/

parameters from parameters file: ENDS:

RAW_DATA_DIR:

STAR_INDEX:

STAR_OPTIONS:

STAR_RESULTS:

SAMTOOLS_VERSION:

STAR_VERSION:

19.5 HTSEQ

omics_pipe.modules.htseq.**htseq**(*sample*, *htseq_flag*)

Runs htseq-count to get raw count data from alignments.

input: Aligned.out.sort.bam

output: counts.txt

citation: Simon Anders, EMBL

link: http://www-huber.embl.de/users/anders/HTSeq/doc/overview.html

parameters from parameters file: STAR_RESULTS:

HTSEQ_OPTIONS:

REF_GENES:

HTSEQ_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BAM_FILE_NAME:

PYTHON_VERSION:

19.6 R Summary Report - DESEQ2

omics_pipe.modules.RNAseq_report_counts.**RNAseq_report_counts**(*sample*, *RNAseq_report_counts_flag*)

Runs R script with knitr to produce report from RNAseq pipeline.

input: results from other steps in RNAseq pipelines

output: html report

citation:

20. Meissner

parameters from parameter file: WORKING_DIR:

R_VERSION:

REPORT_RESULTS:

PARAMS_FILE:

CHAPTER 20

ChIP-seq Modules – HOMER

20.1 FASTQC

omics_pipe.modules.fastqc.**fastqc**(*sample*, *fastqc_flag*)

QC check of raw .fastq files using FASTQC.

input: .fastq file

output: folder and zipped folder containing html, txt and image files

citation: Babraham Bioinformatics

link: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

parameters from parameters file: RAW_DATA_DIR:

QC_PATH:

FASTQC_VERSION:

COMPRESSION:

20.2 ChIP trim

omics_pipe.modules.ChIP_trim.**ChIP_trim**(*sample*, *ChIP_trim_flag*)

Runs Homer Tools to trim adapters from .fastq files.

input: .fastq file

output: .fastq file

citation: Heinz S, Benner C, Spann N, Bertolino E et al. Simple Combinations of Lineage-Determining Transcription Factors Prime cis-Regulatory Elements Required for Macrophage and B Cell Identities. Mol Cell 2010 May 28;38(4):576-589. PMID: 20513432

link: http://homer.salk.edu/homer/

parameters from parameters file: ENDS:

RAW_DATA_DIR:

HOMER_TRIM_OPTIONS:

TRIMMED_DATA_PATH:

HOMER_VERSION:

20.3 Bowtie

omics_pipe.modules.bowtie.**bowtie**(*sample*, *bowtie_flag*)

Runs Bowtie to align .fastq files.

input: .fastq file

output: sample.bam

citation: Langmead B, Trapnell C, Pop M, Salzberg SL. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biology 10:R25

link: http://bowtie-bio.sourceforge.net/index.shtml

parameters from parameters file: ENDS:

TRIMMED_DATA_PATH:

BOWTIE_OPTIONS:

BOWTIE_INDEX:

BOWTIE_RESULTS:

BOWTIE_VERSION:

SAMTOOLS_VERSION:

BEDTOOLS_VERSION:

20.4 Read Density -HOMER

omics_pipe.modules.read_density.**read_density**(*sample*, *read_density_flag*)

Runs HOMER to visualize read density from ChIPseq data.

input: .bam file

output: .txt file

citation: Heinz S, Benner C, Spann N, Bertolino E et al. Simple Combinations of Lineage-Determining Transcription Factors Prime cis-Regulatory Elements Required for Macrophage and B Cell Identities. Mol Cell 2010 May 28;38(4):576-589. PMID: 20513432

link: http://homer.salk.edu/homer/

parameters from parameters file: BOWTIE_RESULTS:

CHROM_SIZES:

HOMER_RESULTS:

HOMER_VERSION:

TEMP_DIR:

20.5 Peak Detection - HOMER

omics_pipe.modules.homer_peaks.**homer_peaks**(*step*, *homer_peaks_flag*)

Runs HOMER to call peaks from ChIPseq data.

input: .tag input file

output: .txt file

citation: Heinz S, Benner C, Spann N, Bertolino E et al. Simple Combinations of Lineage-Determining Transcription Factors Prime cis-Regulatory Elements Required for Macrophage and B Cell Identities. Mol Cell 2010 May 28;38(4):576-589. PMID: 20513432

link: http://homer.salk.edu/homer/

parameters from parameters file: PAIR_LIST:

HOMER_RESULTS:

HOMER_PEAKS_OPTIONS:

HOMER_VERSION:

TEMP_DIR:

20.6 Peak Annotation & Visualization - HOMER

omics_pipe.modules.peak_track.**peak_track**(*step*, *peak_track_flag*)
Runs HOMER to create peak track from ChIPseq data.

input: .tag input file

output: .txt file

citation: Heinz S, Benner C, Spann N, Bertolino E et al. Simple Combinations of Lineage-Determining Transcription Factors Prime cis-Regulatory Elements Required for Macrophage and B Cell Identities. Mol Cell 2010 May 28;38(4):576-589. PMID: 20513432

link: http://homer.salk.edu/homer/

parameters from parameters file: PAIR_LIST:

HOMER_RESULTS:

HOMER_VERSION:

TEMP_DIR:

omics_pipe.modules.annotate_peaks.**annotate_peaks**(*step*, *annotate_peaks_flag*)
Runs HOMER to annotate peak track from ChIPseq data.

input: .tag input file

output: .txt file

citation: Heinz S, Benner C, Spann N, Bertolino E et al. Simple Combinations of Lineage-Determining Transcription Factors Prime cis-Regulatory Elements Required for Macrophage and B Cell Identities. Mol Cell 2010 May 28;38(4):576-589. PMID: 20513432

link: http://homer.salk.edu/homer/

parameters from parameters file: PAIR_LIST:

HOMER_RESULTS:

HOMER_VERSION:

TEMP_DIR:

HOMER_GENOME:

HOMER_ANNOTATE_OPTIONS:

20.7 Find Motifs - HOMER

omics_pipe.modules.find_motifs.**find_motifs**(*step*, *find_motifs_flag*)

Runs HOMER to find motifs from ChIPseq data.

input: .txt peak file from Homer

output: .txt file

citation: Heinz S, Benner C, Spann N, Bertolino E et al. Simple Combinations of Lineage-Determining Transcription Factors Prime cis-Regulatory Elements Required for Macrophage and B Cell Identities. Mol Cell 2010 May 28;38(4):576-589. PMID: 20513432

link: http://homer.salk.edu/homer/

parameters from parameters file: PAIR_LIST:

HOMER_RESULTS:

HOMER_VERSION:

TEMP_DIR:

HOMER_GENOME:

HOMER_MOTIFS_OPTIONS:

CHAPTER 21

ChIP-seq Modules – MACS

21.1 FASTQC

omics_pipe.modules.fastqc.**fastqc**(*sample*, *fastqc_flag*)

QC check of raw .fastq files using FASTQC.

input: .fastq file

output: folder and zipped folder containing html, txt and image files

citation: Babraham Bioinformatics

link: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

parameters from parameters file: RAW_DATA_DIR:

QC_PATH:

FASTQC_VERSION:

COMPRESSION:

21.2 ChIP trim

omics_pipe.modules.ChIP_trim.**ChIP_trim**(*sample*, *ChIP_trim_flag*)

Runs Homer Tools to trim adapters from .fastq files.

input: .fastq file

output: .fastq file

citation: Heinz S, Benner C, Spann N, Bertolino E et al. Simple Combinations of Lineage-Determining Transcription Factors Prime cis-Regulatory Elements Required for Macrophage and B Cell Identities. Mol Cell 2010 May 28;38(4):576-589. PMID: 20513432

link: http://homer.salk.edu/homer/

parameters from parameters file: ENDS:

RAW_DATA_DIR:

HOMER_TRIM_OPTIONS:

TRIMMED_DATA_PATH:

HOMER_VERSION:

21.3 Bowtie

omics_pipe.modules.bowtie.**bowtie**(*sample*, *bowtie_flag*)

Runs Bowtie to align .fastq files.

input: .fastq file

output: sample.bam

citation: Langmead B, Trapnell C, Pop M, Salzberg SL. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biology 10:R25

link: http://bowtie-bio.sourceforge.net/index.shtml

parameters from parameters file: ENDS:

TRIMMED_DATA_PATH:

BOWTIE_OPTIONS:

BOWTIE_INDEX:

BOWTIE_RESULTS:

BOWTIE_VERSION:

SAMTOOLS_VERSION:

BEDTOOLS_VERSION:

21.4 MACS

omics_pipe.modules.macs.**macs**(*step*, *macs_flag*)

Runs MACS to call peaks from ChIPseq data. input:

.fastq file

output: peaks and .bed file

citation: Zhang et al. Model-based Analysis of ChIP-Seq (MACS). Genome Biol (2008) vol. 9 (9) pp. R137

link: http://liulab.dfci.harvard.edu/MACS/

parameters from parameters file: PAIR_LIST:

BOWTIE_RESULTS:

CHROM_SIZES:

MACS_RESULTS:

MACS_VERSION:

TEMP_DIR:

BEDTOOLS_VERSION:

PYTHON_VERSION:

CHAPTER 22

Whole Genome and Whole Exome Sequencing Modules

22.1 FASTQC

omics_pipe.modules.fastqc.**fastqc**(*sample*, *fastqc_flag*)

QC check of raw .fastq files using FASTQC.

input: .fastq file

output: folder and zipped folder containing html, txt and image files

citation: Babraham Bioinformatics

link: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

parameters from parameters file: RAW_DATA_DIR:

QC_PATH:

FASTQC_VERSION:

COMPRESSION:

22.2 BWA-MEM

omics_pipe.modules.bwa.**bwa_mem**(*sample*, *bwa_mem_flag*)

BWA aligner with BWA-MEM algorithm.

input: .fastq

output: .sam

citation: Li H. and Durbin R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics, 25, 1754-1760. [PMID: 19451168]

link: http://bio-bwa.sourceforge.net/bwa.shtml

parameters from parameters file: BWA_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BWA_VERSION:

GENOME:

RAW_DATA_DIR:

BWA_OPTIONS:

COMPRESSION:

22.3 PICARD Mark Duplicates

omics_pipe.modules.picard_mark_duplicates.**picard_mark_duplicates**(*sample*, *picard_mark_duplicates_flag*)

Picard tools Mark Duplicates.

input: sorted.bam

output: _sorted.rg.md.bam

citation: http://picard.sourceforge.net/

link: http://picard.sourceforge.net/

parameters from parameters file: BWA_RESULTS:

TEMP_DIR:

PICARD_VERSION:

SAMTOOLS_VERSION:

22.4 GATK Preprocessing

22.4.1 WES

omics_pipe.modules.GATK_preprocessing_WES.**GATK_preprocessing_WES**(*sample*, *GATK_preprocessing_WES_flag*)

GATK preprocessing steps for whole exome sequencing.

input: sorted.rg.md.bam

output: .ready.bam

citation: McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, Kernytsky A, Garimella K, Altshuler D, Gabriel S, Daly M, DePristo MA (2010). The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. Genome Res. 20:1297-303.

link: http://www.broadinstitute.org/gatk/

parameters from parameters file: BWA_RESULTS:

TEMP_DIR:

GATK_VERSION:

GENOME:

DBSNP:

MILLS:

G1000:

CAPTURE_KIT_BED:

SAMTOOLS_VERSION:

22.4.2 WGS

omics_pipe.modules.GATK_preprocessing_WGS.**GATK_preprocessing_WGS**(*sample*, *GATK_preprocessing_WGS_flag*)

GATK preprocessing steps for whole genome sequencing.

input: sorted.rg.md.bam

output: .ready.bam

citation: McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, Kernytsky A, Garimella K, Altshuler D, Gabriel S, Daly M, DePristo MA (2010). The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. Genome Res. 20:1297-303.

link: http://www.broadinstitute.org/gatk/

parameters from parameters file: BWA_RESULTS:

TEMP_DIR:

GATK_VERSION:

GENOME:

DBSNP:

MILLS:

G1000:

SAMTOOLS_VERSION:

22.5 GATK Variant Discovery

omics_pipe.modules.GATK_variant_discovery.**GATK_variant_discovery**(*sample*, *GATK_variant_discovery_flag*)

GATK_variant_discovery.

input: sorted.rg.md.bam

output: .ready.bam

citation: McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, Kernytsky A, Garimella K, Altshuler D, Gabriel S, Daly M, DePristo MA (2010). The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. Genome Res. 20:1297-303.

link: GATK_variant_discovery http://www.broadinstitute.org/gatk/

parameters from parameters file: BWA_RESULTS:

TEMP_DIR:

GATK_VERSION:

GENOME:

DBSNP:

VARIANT_RESULTS:

22.6 GATK Variant Filtering

omics_pipe.modules.GATK_variant_filtering.**GATK_variant_filtering**(*sample*, *GATK_variant_filtering_flag*)

GATK_variant_filtering.

input: sorted.rg.md.bam

output: .ready.bam

citation: McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, Kernytsky A, Garimella K, Altshuler D, Gabriel S, Daly M, DePristo MA (2010). The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. Genome Res. 20:1297-303.

link: GATK_variant_filtering http://www.broadinstitute.org/gatk/

parameters from parameters file: VARIANT_RESULTS:

TEMP_DIR:

GATK_VERSION:

GENOME:

DBSNP:

MILLS:

OMNI:

HAPMAP:

R_VERSION:

G1000_SNPs:

G1000_Indels:

omics_pipe.modules.GATK_variant_filtering.**GATK_variant_filtering_group**(*sample*, *GATK_variant_filtering_grou*

GATK_variant_filtering.

input: sorted.rg.md.bam

output: .ready.bam

citation: McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, Kernytsky A, Garimella K, Altshuler D, Gabriel S, Daly M, DePristo MA (2010). The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. Genome Res. 20:1297-303.

link: GATK_variant_filtering http://www.broadinstitute.org/gatk/

parameters from parameters file:

VARIANT_RESULTS:

TEMP_DIR:

GATK_VERSION:

GENOME:

DBSNP:

MILLS_G1000:

OMNI:

HAPMAP:

R_VERSION:

G1000:

CHAPTER 23

Whole Genome Sequencing (MUTECT)

23.1 FASTQC

omics_pipe.modules.fastqc.**fastqc**(*sample*, *fastqc_flag*)
QC check of raw .fastq files using FASTQC.

input: .fastq file

output: folder and zipped folder containing html, txt and image files

citation: Babraham Bioinformatics

link: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

parameters from parameters file: RAW_DATA_DIR:

QC_PATH:

FASTQC_VERSION:

COMPRESSION:

23.2 BWA-MEM

omics_pipe.modules.bwa.**bwa_mem**(*sample*, *bwa_mem_flag*)
BWA aligner with BWA-MEM algorithm.

input: .fastq

output: .sam

citation: Li H. and Durbin R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics, 25, 1754-1760. [PMID: 19451168]

link: http://bio-bwa.sourceforge.net/bwa.shtml

parameters from parameters file: BWA_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BWA_VERSION:

GENOME:

RAW_DATA_DIR:

BWA_OPTIONS:

COMPRESSION:

23.3 MUTECT

omics_pipe.modules.mutect.**mutect**(*sample*, *mutect_flag*)

Runs MuTect on paired tumor/normal samples to detect somatic point mutations in cancer genomes.

input: .bam

output: call_stats.txt

citation: Cibulskis, K. et al. Sensitive detection of somatic point mutations in impure and heterogeneous cancer samples. Nat Biotechnology (2013).doi:10.1038/nbt.2514

link: http://www.broadinstitute.org/cancer/cga/mutect

parameters from parameters file: BWA_RESULTS:

TEMP_DIR:

GATK_VERSION:

GENOME:

DBSNP:

MILLS:

G1000:

CAPTURE_KIT_BED:

CHAPTER 24

All Available Modules

Below are all available modules in the current release of Omics Pipe in alphabetical order. When creating a custom pipeline, you can choose from these modules or create your own.

omics_pipe.modules.annotate_peaks.**annotate_peaks**(*step*, *annotate_peaks_flag*)

Runs HOMER to annotate peak track from ChIPseq data.

input: .tag input file

output: .txt file

citation: Heinz S, Benner C, Spann N, Bertolino E et al. Simple Combinations of Lineage-Determining Transcription Factors Prime cis-Regulatory Elements Required for Macrophage and B Cell Identities. Mol Cell 2010 May 28;38(4):576-589. PMID: 20513432

link: http://homer.salk.edu/homer/

parameters from parameters file: PAIR_LIST:

HOMER_RESULTS:

HOMER_VERSION:

TEMP_DIR:

HOMER_GENOME:

HOMER_ANNOTATE_OPTIONS:

omics_pipe.modules.annotate_variants.**annotate_variants**(*sample*, *annotate_variants_flag*)

Annotates variants with ANNOVAR variant annotator. Follows VarCall. input:

.vcf

output: .vcf

citation: Wang K, Li M, Hakonarson H. ANNOVAR: Functional annotation of genetic variants from next-generation sequencing data Nucleic Acids Research, 38:e164, 2010

link: http://www.openbioinformatics.org/annovar/

parameters from parameters file: VARIANT_RESULTS:

ANNOVARDB:

ANNOVAR_OPTIONS:

ANNOVAR_OPTIONS2:

TEMP_DIR:

ANNOVAR_VERSION:

VCFTOOLS_VERSION:

omics_pipe.modules.bowtie.**bowtie**(*sample*, *bowtie_flag*)

Runs Bowtie to align .fastq files.

input: .fastq file

output: sample.bam

citation: Langmead B, Trapnell C, Pop M, Salzberg SL. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biology 10:R25

link: http://bowtie-bio.sourceforge.net/index.shtml

parameters from parameters file: ENDS:

TRIMMED_DATA_PATH:

BOWTIE_OPTIONS:

BOWTIE_INDEX:

BOWTIE_RESULTS:

BOWTIE_VERSION:

SAMTOOLS_VERSION:

BEDTOOLS_VERSION:

omics_pipe.modules.BreastCancer_RNA_report.**BreastCancer_RNA_report**(*sample*, *BreastCancer_RNA_report_flag*)

Runs R script with knitr to produce report from RNAseq pipeline.

input: results from other steps in RNAseq pipelines

output: html report

citation:

20. Meissner

parameters from parameter file: WORKING_DIR:

R_VERSION:

REPORT_RESULTS:

PARAMS_FILE:

TABIX_VERSION:

TUMOR_TYPE:

GENELIST:

COSMIC:

CLINVAR:

PHARMGKB_rsID:

PHARMGKB_Allele:

DRUGBANK:

CADD:

omics_pipe.modules.bwa.**bwa1**(*sample*, *bwa1_flag*)
BWA aligner for read1 of paired_end reads.

input: .fastq

output: .sam

citation: Li H. and Durbin R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics, 25, 1754-1760. [PMID: 19451168]

link: http://bio-bwa.sourceforge.net/bwa.shtml

parameters from parameters file: BWA_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BWA_VERSION:

BWA_INDEX:

RAW_DATA_DIR:

GATK_READ_GROUP_INFO:

COMPRESSION:

omics_pipe.modules.bwa.**bwa2**(*sample*, *bwa2_flag*)
BWA aligner for read2 of paired_end reads.

input: .fastq

output: .sam

citation: Li H. and Durbin R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics, 25, 1754-1760. [PMID: 19451168]

link: http://bio-bwa.sourceforge.net/bwa.shtml

parameters from parameters file: BWA_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BWA_VERSION:

BWA_INDEX:

RAW_DATA_DIR:

GATK_READ_GROUP_INFO:

COMPRESSION:

omics_pipe.modules.bwa.**bwa_RNA**(*sample*, *bwa_flag*)
BWA aligner for single end reads.

input: .fastq

output: .sam

citation: Li H. and Durbin R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics, 25, 1754-1760. [PMID: 19451168]

link: http://bio-bwa.sourceforge.net/bwa.shtml

parameters from parameters file: BWA_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BWA_VERSION:

BWA_INDEX:

RAW_DATA_DIR:

GATK_READ_GROUP_INFO:

COMPRESSION:

omics_pipe.modules.bwa.**bwa_mem**(*sample*, *bwa_mem_flag*)

BWA aligner with BWA-MEM algorithm.

input: .fastq

output: .sam

citation: Li H. and Durbin R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics, 25, 1754-1760. [PMID: 19451168]

link: http://bio-bwa.sourceforge.net/bwa.shtml

parameters from parameters file: BWA_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BWA_VERSION:

GENOME:

RAW_DATA_DIR:

BWA_OPTIONS:

COMPRESSION:

omics_pipe.modules.bwa.**bwa_mem_pipe**(*sample*, *bwa_mem_pipe_flag*)

BWA aligner with BWA-MEM algorithm.

input: .fastq

output: .sam

citation: Li H. and Durbin R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics, 25, 1754-1760. [PMID: 19451168]

link: http://bio-bwa.sourceforge.net/bwa.shtml

parameters from parameters file: BWA_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BWA_VERSION:

GENOME:

RAW_DATA_DIR:

BWA_OPTIONS:

COMPRESSION:

SAMBAMBA_VERSION:

SAMBLASTER_VERSION:

SAMBAMBA_OPTIONS:

omics_pipe.modules.call_variants.**call_variants**(*sample*, *call_variants_flag*)

Calls variants from alignment .bam files using Varcall.

input: Aligned.out.sort.bam or accepted_hits.bam

output: .vcf file

citation: Erik Aronesty (2011). ea-utils : "Command-line tools for processing biological sequencing data";

link: https://code.google.com/p/ea-utils/wiki/Varcall

parameters from parameters file: STAR_RESULTS:

GENOME:

VARSCAN_PATH:

VARSCAN_OPTIONS:

VARIANT_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

ANNOVAR_VERSION:

VCFTOOLS_VERSION:

VARSCAN_VERSION:

SAMTOOLS_OPTIONS:

omics_pipe.modules.ChIP_trim.**ChIP_trim**(*sample*, *ChIP_trim_flag*)

Runs Homer Tools to trim adapters from .fastq files.

input: .fastq file

output: .fastq file

citation: Heinz S, Benner C, Spann N, Bertolino E et al. Simple Combinations of Lineage-Determining Transcription Factors Prime cis-Regulatory Elements Required for Macrophage and B Cell Identities. Mol Cell 2010 May 28;38(4):576-589. PMID: 20513432

link: http://homer.salk.edu/homer/

parameters from parameters file: ENDS:

RAW_DATA_DIR:

HOMER_TRIM_OPTIONS:

TRIMMED_DATA_PATH:

HOMER_VERSION:

omics_pipe.modules.cuffdiff.**cuffdiff**(*step*, *cuffdiff_flag*)

Runs Cuffdiff to perform differential expression. Runs after Cufflinks. Part of Tuxedo Suite.

input: .bam files

output: differential expression results

citation: Trapnell C, et al. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation Nature Biotechnology doi:10.1038/nbt.1621

link: http://cufflinks.cbcb.umd.edu/

parameters from parameters file: CUFFDIFF_RESULTS:

GENOME:

CUFFDIFF_OPTIONS:

CUFFMERGE_RESULTS:

CUFFDIFF_INPUT_LIST_COND1:

CUFFDIFF_INPUT_LIST_COND2:

CUFFLINKS_VERSION:

omics_pipe.modules.cuffdiff_miRNA.**cuffdiff_miRNA**(*step*, *cuffdiff_miRNA_flag*)
Runs Cuffdiff to perform differential expression. Runs after Cufflinks. Part of Tuxedo Suite.

input: .bam files

output: differential expression results

citation: Trapnell C, et al. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation Nature Biotechnology doi:10.1038/nbt.1621

link: http://cufflinks.cbcb.umd.edu/

parameters from parameters file: CUFFDIFF_RESULTS:

GENOME:

CUFFDIFF_OPTIONS:

CUFFMERGE_RESULTS:

CUFFDIFF_INPUT_LIST_COND1:

CUFFDIFF_INPUT_LIST_COND2:

CUFFLINKS_VERSION:

omics_pipe.modules.cufflinks.**cufflinks**(*sample*, *cufflinks_flag*)
Runs cufflinks to assemble .bam files from TopHat.

input: accepted_hits.bam

output: transcripts.gtf

citation: Trapnell C, et al. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation Nature Biotechnology doi:10.1038/nbt.1621

link: http://cufflinks.cbcb.umd.edu/

parameters from parameters file: TOPHAT_RESULTS:

CUFFLINKS_RESULTS:

REF_GENES:

GENOME:

CUFFLINKS_OPTIONS:

CUFFLINKS_VERSION:

omics_pipe.modules.cufflinks_miRNA.**cufflinks_miRNA**(*sample*, *cufflinks_miRNA_flag*)
Runs cufflinks to assemble .bam files from TopHat. Takes parameter MIRNA_GTF.

input: accepted_hits.bam

output: transcripts.gtf

citation: Trapnell C, et al. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation Nature Biotechnology doi:10.1038/nbt.1621

link: http://cufflinks.cbcb.umd.edu/

parameters from parameters file: TOPHAT_RESULTS:

CUFFLINKS_RESULTS:

miRNA_GTF:

GENOME:

CUFFLINKS_OPTIONS:

CUFFLINKS_VERSION:

omics_pipe.modules.cufflinks_ncRNA.**cufflinks_ncRNA**(*sample*, *cufflinks_ncRNA_flag*)
Runs cufflinks to assemble .bam files from TopHat. Takes parameters LNCRNA_GTF and NONCODE_FASTA.

input: accepted_hits.bam

output: transcripts.gtf

citation: Trapnell C, et al. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation Nature Biotechnology doi:10.1038/nbt.1621

link: http://cufflinks.cbcb.umd.edu/

parameters from parameters file: TOPHAT_RESULTS:

CUFFLINKS_RESULTS:

LNCRNA_GTF:

NONCODE_FASTA:

CUFFLINKS_OPTIONS:

CUFFLINKS_VERSION:

omics_pipe.modules.cuffmerge.**cuffmerge**(*step*, *cuffmerge_flag*)
Runs cuffmerge to merge .gtf files from Cufflinks.

input: assembly_GTF_list.txt

output: merged.gtf

citation: Trapnell C, et al. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation Nature Biotechnology doi:10.1038/nbt.1621

link: http://cufflinks.cbcb.umd.edu/

parameters from parameters file: CUFFMERGE_RESULTS:

REF_GENES:

GENOME:

CUFFMERGE_OPTIONS:

CUFFLINKS_VERSION:

omics_pipe.modules.cuffmerge_miRNA.**cuffmerge_miRNA**(*step*, *cuffmerge_miRNA_flag*)

Runs cuffmerge to merge .gtf files from Cufflinks.

input: assembly_GTF_list.txt

output: merged.gtf

citation: Trapnell C, et al. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation Nature Biotechnology doi:10.1038/nbt.1621

link: http://cufflinks.cbcb.umd.edu/

parameters from parameters file: CUFFMERGE_RESULTS:

miRNA_GTF:

GENOME:

CUFFMERGE_OPTIONS:

CUFFLINKS_VERSION:

omics_pipe.modules.cuffmergetocompare.**cuffmergetocompare**(*step*, *cuffmergetocompare_flag*)

Runs cuffcompare to annotate merged .gtf files from Cufflinks.

input: assembly_GTF_list.txt

output: merged.gtf

citation: Trapnell C, et al. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation Nature Biotechnology doi:10.1038/nbt.1621

link: http://cufflinks.cbcb.umd.edu/

parameters from parameters file: CUFFMERGE_RESULTS:

REF_GENES:

GENOME:

CUFFMERGETOCOMPARE_OPTIONS:

CUFFLINKS_VERSION:

omics_pipe.modules.cuffmergetocompare_miRNA.**cuffmergetocompare_miRNA**(*step*, *cuffmergetocompare_miRNA_flag*)

Runs cuffcompare to annotate merged .gtf files from Cufflinks.

input: assembly_GTF_list.txt

output: merged.gtf

citation: Trapnell C, et al. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation Nature Biotechnology doi:10.1038/nbt.1621

link: http://cufflinks.cbcb.umd.edu/

parameters from parameters file: CUFFMERGE_RESULTS:

miRNA_GTF:

GENOME:

CUFFMERGETOCOMPARE_OPTIONS:

CUFFLINKS_VERSION:

omics_pipe.modules.custom_R_report.**custom_R_report**(*sample*, *custom_R_report_flag*)
Runs R script with knitr to produce report from omics pipeline.

input: results from other steps in RNAseq pipelines

output: html report

citation:

20. Meissner

parameters from parameter file: REPORT_SCRIPT:

R_VERSION:

REPORT_RESULTS:

R_MARKUP_FILE:

DPS_VERSION:

PARAMS_FILE:

omics_pipe.modules.cutadapt_miRNA.**cutadapt_miRNA**(*sample*, *cutadapt_miRNA_flag*)
Runs Cutadapt to trim adapters from reads.

input: .fastq

output: .fastq

citation: Martin 2011. Cutadapt removes adapter sequences from high-throughput sequencing reads. EMBnet.journal 17: 10-12.

link: https://code.google.com/p/cutadapt/

parameters from parameters file: RAW_DATA_DIR:

ADAPTER:

TRIMMED_DATA_PATH:

PYTHON_VERSION

omics_pipe.modules.fastq_length_filter_miRNA.**fastq_length_filter_miRNA**(*sample*, *fastq_length_filter_miRNA_fl*
Runs custom Python script to filter miRNA reads by length.

input: .fastq

output: .fastq

parameters from parameter file: TRIMMED_DATA_PATH:

omics_pipe.modules.fastqc.**fastqc**(*sample*, *fastqc_flag*)
QC check of raw .fastq files using FASTQC.

input: .fastq file

output: folder and zipped folder containing html, txt and image files

citation: Babraham Bioinformatics

link: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

parameters from parameters file: RAW_DATA_DIR:

QC_PATH:

FASTQC_VERSION:

COMPRESSION:

omics_pipe.modules.fastqc_miRNA.**fastqc_miRNA**(*sample*, *fastqc_miRNA_flag*)

QC check of raw .fastq files using FASTQC.

input: .fastq file

output: folder and zipped folder containing html, txt and image files

citation: Babraham Bioinformatics

link: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

parameters from parameters file: RAW_DATA_DIR:

QC_PATH:

FASTQC_VERSION:

omics_pipe.modules.filter_variants.**filter_variants**(*sample*, *filter_variants_flag*)

Filters variants to remove common variants.

input: .bam or .sam file

output: .vcf file

citation: Piskol et al. 2013. Reliable identification of genomic variants from RNA-seq data. The American Journal of Human Genetics 93: 641-651.

link: http://lilab.stanford.edu/SNPiR/

parameters from parameters file: VARIANT_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BWA_VERSION:

PICARD_VERSION:

GATK_VERSION:

BEDTOOLS_VERSION:

UCSC_TOOLS_VERSION:

GENOME:

REPEAT_MASKER:

SNPIR_ANNOTATION:

RNA_EDIT:

DBSNP:

MILLS:

G1000:

WORKING_DIR:

BWA_RESULTS:

SNPIR_VERSION:

SNPIR_CONFIG:

SNPIR_DIR:

SNPEFF_VERSION:

dbNSFP:

VCFTOOLS_VERSION:

WORKING_DIR:

SNP_FILTER_OUT_REF:

omics_pipe.modules.find_motifs.**find_motifs**(*step*, *find_motifs_flag*)

Runs HOMER to find motifs from ChIPseq data.

input: .txt peak file from Homer

output: .txt file

citation: Heinz S, Benner C, Spann N, Bertolino E et al. Simple Combinations of Lineage-Determining Transcription Factors Prime cis-Regulatory Elements Required for Macrophage and B Cell Identities. Mol Cell 2010 May 28;38(4):576-589. PMID: 20513432

link: http://homer.salk.edu/homer/

parameters from parameters file: PAIR_LIST:

HOMER_RESULTS:

HOMER_VERSION:

TEMP_DIR:

HOMER_GENOME:

HOMER_MOTIFS_OPTIONS:

omics_pipe.modules.fusion_catcher.**fusion_catcher**(*sample*, *fusion_catcher_flag*)

Detects fusion genes in paired-end RNAseq data.

input: paired end .fastq files

output: list of candidate fusion genes

citation:

19. Kangaspeska, S. Hultsch, H. Edgren, D. Nicorici, A. Murumgi, O.P. Kallioniemi, Reanalysis of RNA-sequencing data reveals several additional fusion genes with multiple isoforms, PLOS One, Oct. 2012. http://dx.plos.org/10.1371/journal.pone.0048745

link: https://code.google.com/p/fusioncatcher

parameters from parameters file: ENDS:

RAW_DATA_DIR:

FUSION_RESULTS:

FUSIONCATCHERBUILD_DIR:

TEMP_DIR:

SAMTOOLS_VERSION:

FUSIONCATCHER_VERSION:

FUSIONCATCHER_OPTIONS:

TISSUE:

PYTHON_VERSION:

omics_pipe.modules.GATK_preprocessing_WES.**GATK_preprocessing_WES**(*sample*, *GATK_preprocessing_WES_flag*)

GATK preprocessing steps for whole exome sequencing.

input: sorted.rg.md.bam

output: .ready.bam

citation: McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, Kernytsky A, Garimella K, Altshuler D, Gabriel S, Daly M, DePristo MA (2010). The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. Genome Res. 20:1297-303.

link: http://www.broadinstitute.org/gatk/

parameters from parameters file: BWA_RESULTS:

TEMP_DIR:

GATK_VERSION:

GENOME:

DBSNP:

MILLS:

G1000:

CAPTURE_KIT_BED:

SAMTOOLS_VERSION:

omics_pipe.modules.GATK_preprocessing_WGS.**GATK_preprocessing_WGS**(*sample*, *GATK_preprocessing_WGS_flag*)

GATK preprocessing steps for whole genome sequencing.

input: sorted.rg.md.bam

output: .ready.bam

citation: McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, Kernytsky A, Garimella K, Altshuler D, Gabriel S, Daly M, DePristo MA (2010). The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. Genome Res. 20:1297-303.

link: http://www.broadinstitute.org/gatk/

parameters from parameters file: BWA_RESULTS:

TEMP_DIR:

GATK_VERSION:

GENOME:

DBSNP:

MILLS:

G1000:

SAMTOOLS_VERSION:

omics_pipe.modules.GATK_variant_discovery.**GATK_variant_discovery**(*sample*, *GATK_variant_discovery_flag*)

GATK_variant_discovery.

input: sorted.rg.md.bam

output: .ready.bam

citation: McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, Kernytsky A, Garimella K, Altshuler D, Gabriel S, Daly M, DePristo MA (2010). The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. Genome Res. 20:1297-303.

link: GATK_variant_discovery http://www.broadinstitute.org/gatk/

parameters from parameters file: BWA_RESULTS:

TEMP_DIR:

GATK_VERSION:

GENOME:

DBSNP:

VARIANT_RESULTS:

omics_pipe.modules.GATK_variant_filtering.**GATK_variant_filtering**(*sample*, *GATK_variant_filtering_flag*)

GATK_variant_filtering.

input: sorted.rg.md.bam

output: .ready.bam

citation: McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, Kernytsky A, Garimella K, Altshuler D, Gabriel S, Daly M, DePristo MA (2010). The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. Genome Res. 20:1297-303.

link: GATK_variant_filtering http://www.broadinstitute.org/gatk/

parameters from parameters file: VARIANT_RESULTS:

TEMP_DIR:

GATK_VERSION:

GENOME:

DBSNP:

MILLS:

OMNI:

HAPMAP:

R_VERSION:

G1000_SNPs:

G1000_Indels:

omics_pipe.modules.GATK_variant_filtering.**GATK_variant_filtering_group**(*sample*, *GATK_variant_filtering_grou*

GATK_variant_filtering.

input: sorted.rg.md.bam

output: .ready.bam

citation: McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, Kernytsky A, Garimella K, Altshuler D, Gabriel S, Daly M, DePristo MA (2010). The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. Genome Res. 20:1297-303.

link: GATK_variant_filtering http://www.broadinstitute.org/gatk/

parameters from parameters file:

VARIANT_RESULTS:

TEMP_DIR:

GATK_VERSION:

GENOME:

DBSNP:

MILLS_G1000:

OMNI:

HAPMAP:

R_VERSION:

G1000:

omics_pipe.modules.homer_peaks.**homer_peaks**(*step*, *homer_peaks_flag*)

Runs HOMER to call peaks from ChIPseq data.

input: .tag input file

output: .txt file

citation: Heinz S, Benner C, Spann N, Bertolino E et al. Simple Combinations of Lineage-Determining Transcription Factors Prime cis-Regulatory Elements Required for Macrophage and B Cell Identities. Mol Cell 2010 May 28;38(4):576-589. PMID: 20513432

link: http://homer.salk.edu/homer/

parameters from parameters file: PAIR_LIST:

HOMER_RESULTS:

HOMER_PEAKS_OPTIONS:

HOMER_VERSION:

TEMP_DIR:

omics_pipe.modules.htseq.**htseq**(*sample*, *htseq_flag*)

Runs htseq-count to get raw count data from alignments.

input: Aligned.out.sort.bam

output: counts.txt

citation: Simon Anders, EMBL

link: http://www-huber.embl.de/users/anders/HTSeq/doc/overview.html

parameters from parameters file: STAR_RESULTS:

HTSEQ_OPTIONS:

REF_GENES:

HTSEQ_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BAM_FILE_NAME:

PYTHON_VERSION:

omics_pipe.modules.htseq_gencode.**htseq_gencode**(*sample*, *htseq_flag*)
Runs htseq-count to get raw count data from alignments.

input: Aligned.out.sort.bam

output: counts.txt

citation: Simon Anders, EMBL

link: http://www-huber.embl.de/users/anders/HTSeq/doc/overview.html

parameters from parameters file: STAR_RESULTS:

HTSEQ_OPTIONS:

REF_GENES_GENCODE:

HTSEQ_GENCODE_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BAM_FILE_NAME:

omics_pipe.modules.htseq_miRNA.**htseq_miRNA**(*sample*, *htseq_miRNA_flag*)
Runs htseq-count to get raw count data from alignments.

input: Aligned.out.sort.bam

output: counts.txt

citation: Simon Anders, EMBL

link: http://www-huber.embl.de/users/anders/HTSeq/doc/overview.html

parameters from parameters file: TOPHAT_RESULTS:

HTSEQ_OPTIONS:

miRNA_GFF:

HTSEQ_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BAM_FILE_NAME:

omics_pipe.modules.intogen.**intogen**(*sample*, *intogen_flag*)
Runs Intogen to rank mutations and implication for cancer phenotype. Follows variant calling.

input: .vcf

output: variant list

citation: Gonzalez-Perez et al. 2013. Intogen mutations identifies cancer drivers across tumor types. Nature Methods 10, 1081-1082.

link: http://www.intogen.org/

parameters from parameter file: VCF_FILE:

INTOGEN_OPTIONS:

INTOGEN_RESULTS:

INTOGEN_VERSION:

USERNAME:

WORKING_DIR:

TEMP_DIR:

SCHEDULER:

VARIANT_RESULTS:

omics_pipe.modules.macs.**macs**(*step*, *macs_flag*)

Runs MACS to call peaks from ChIPseq data. input:

.fastq file

output: peaks and .bed file

citation: Zhang et al. Model-based Analysis of ChIP-Seq (MACS). Genome Biol (2008) vol. 9 (9) pp. R137

link: http://liulab.dfci.harvard.edu/MACS/

parameters from parameters file: PAIR_LIST:

BOWTIE_RESULTS:

CHROM_SIZES:

MACS_RESULTS:

MACS_VERSION:

TEMP_DIR:

BEDTOOLS_VERSION:

PYTHON_VERSION:

omics_pipe.modules.mutect.**mutect**(*sample*, *mutect_flag*)

Runs MuTect on paired tumor/normal samples to detect somatic point mutations in cancer genomes.

input: .bam

output: call_stats.txt

citation: Cibulskis, K. et al. Sensitive detection of somatic point mutations in impure and heterogeneous cancer samples. Nat Biotechnology (2013).doi:10.1038/nbt.2514

link: http://www.broadinstitute.org/cancer/cga/mutect

parameters from parameters file: BWA_RESULTS:

TEMP_DIR:

GATK_VERSION:

GENOME:

DBSNP:

MILLS:

G1000:

CAPTURE_KIT_BED:

omics_pipe.modules.peak_track.**peak_track**(*step*, *peak_track_flag*)

Runs HOMER to create peak track from ChIPseq data.

input: .tag input file

output: .txt file

citation: Heinz S, Benner C, Spann N, Bertolino E et al. Simple Combinations of Lineage-Determining Transcription Factors Prime cis-Regulatory Elements Required for Macrophage and B Cell Identities. Mol Cell 2010 May 28;38(4):576-589. PMID: 20513432

link: http://homer.salk.edu/homer/

parameters from parameters file: PAIR_LIST:

HOMER_RESULTS:

HOMER_VERSION:

TEMP_DIR:

omics_pipe.modules.picard_mark_duplicates.**picard_mark_duplicates**(*sample*, *picard_mark_duplicates_flag*)

Picard tools Mark Duplicates.

input: sorted.bam

output: _sorted.rg.md.bam

citation: http://picard.sourceforge.net/

link: http://picard.sourceforge.net/

parameters from parameters file: BWA_RESULTS:

TEMP_DIR:

PICARD_VERSION:

SAMTOOLS_VERSION:

omics_pipe.modules.read_density.**read_density**(*sample*, *read_density_flag*)

Runs HOMER to visualize read density from ChIPseq data.

input: .bam file

output: .txt file

citation: Heinz S, Benner C, Spann N, Bertolino E et al. Simple Combinations of Lineage-Determining Transcription Factors Prime cis-Regulatory Elements Required for Macrophage and B Cell Identities. Mol Cell 2010 May 28;38(4):576-589. PMID: 20513432

link: http://homer.salk.edu/homer/

parameters from parameters file: BOWTIE_RESULTS:

CHROM_SIZES:

HOMER_RESULTS:

HOMER_VERSION:

TEMP_DIR:

omics_pipe.modules.RNAseq_QC.**RNAseq_QC**(*sample*, *RNAseq_QC_flag*)

Runs rseqc to determine insert size as QC for alignment.

input: .bam

output: pdf plot

link: http://rseqc.sourceforge.net/

parameters from parameters file: STAR_RESULTS:

QC_PATH:

BAM_FILE_NAME:

RSEQC_REF:

TEMP_DIR:

PICARD_VERSION:

R_VERSION:

omics_pipe.modules.RNAseq_report.**RNAseq_report**(*sample*, *RNAseq_report_flag*)

Runs R script with knitr to produce report from RNAseq pipeline.

input: results from other steps in RNAseq pipelines

output: html report

citation:

20. Meissner

parameters from parameter file: REPORT_SCRIPT:

R_VERSION:

REPORT_RESULTS:

R_MARKUP_FILE:

DPS_VERSION:

PARAMS_FILE:

omics_pipe.modules.RNAseq_report_counts.**RNAseq_report_counts**(*sample*, *RNAseq_report_counts_flag*)

Runs R script with knitr to produce report from RNAseq pipeline.

input: results from other steps in RNAseq pipelines

output: html report

citation:

20. Meissner

parameters from parameter file: WORKING_DIR:

R_VERSION:

REPORT_RESULTS:

PARAMS_FILE:

omics_pipe.modules.RNAseq_report_tuxedo.**RNAseq_report_tuxedo**(*sample*, *RNAseq_report_tuxedo_flag*)

Runs R script with knitr to produce report from RNAseq pipeline.

input: results from other steps in RNAseq pipelines

output: html report

citation:

20. Meissner

parameters from parameter file: WORKING_DIR:

R_VERSION:

REPORT_RESULTS:

DPS_VERSION:

PARAMS_FILE:

omics_pipe.modules.rseqc.**rseqc**(*sample*, *rseqc_flag*)

Runs rseqc to determine insert size as QC for alignment.

input: .bam

output: pdf plot

link: http://rseqc.sourceforge.net/

parameters from parameters file: STAR_RESULTS:

QC_PATH:

BAM_FILE_NAME:

RSEQC_REF:

RSEQC_VERSION:

TEMP_DIR:

omics_pipe.modules.snpir_variants.**snpir_variants**(*sample*, *snpir_variants_flag*)

Calls variants using SNPIR pipeline.

input: Aligned.out.sort.bam or accepted_hits.bam

output: final_variants.vcf file

citation: Piskol, R., et al. (2013). "Reliable Identification of Genomic Variants from RNA-Seq Data." The American Journal of Human Genetics 93(4): 641-651.

link: http://lilab.stanford.edu/SNPiR/

parameters from parameters file: VARIANT_RESULTS:

TEMP_DIR:

SAMTOOLS_VERSION:

BWA_VERSION:

PICARD_VERSION:

GATK_VERSION:

BEDTOOLS_VERSION:

UCSC_TOOLS_VERSION:

GENOME:

REPEAT_MASKER:

SNPIR_ANNOTATION:

RNA_EDIT:

DBSNP:

MILLS:

G1000:

WORKING_DIR:

BWA_RESULTS:

SNPIR_VERSION:

SNPIR_CONFIG:

SNPIR_DIR:

ENCODING:

omics_pipe.modules.star.**star**(*sample*, *star_flag*)

Runs STAR to align .fastq files.

input: .fastq file

output: Aligned.out.bam

citation:

1. Dobin et al, Bioinformatics 2012; doi: 10.1093/bioinformatics/bts635 "STAR: ultrafast universal RNA-seq aligner"

link: https://code.google.com/p/rna-star/

parameters from parameters file: ENDS:

RAW_DATA_DIR:

STAR_INDEX:

STAR_OPTIONS:

STAR_RESULTS:

SAMTOOLS_VERSION:

STAR_VERSION:

omics_pipe.modules.star_piRNA.**star_piRNA**(*sample*, *star_flag*)

Runs STAR to align .fastq files.

input: .fastq file

output: Aligned.out.bam

citation:

1. Dobin et al, Bioinformatics 2012; doi: 10.1093/bioinformatics/bts635 "STAR: ultrafast universal RNA-seq aligner"

link: https://code.google.com/p/rna-star/

parameters from parameters file: ENDS:

RAW_DATA_DIR:

STAR_INDEX:

STAR_OPTIONS:

STAR_RESULTS:

SAMTOOLS_VERSION:

STAR_VERSION:

omics_pipe.modules.TCGA_download.**TCGA_download**(*sample*, *TCGA_download_flag*)
Downloads and unzips TCGA data from Manifest.xml downloaded from CGHub. input:

TGCA XML file

output: downloaded files from TCGA

citation: The Cancer Genome Atlas

link: https://cghub.ucsc.edu/software/downloads.html

parameters from parameters file: TCGA_XML_FILE:

TCGA_KEY:

TCGA_OUTPUT_PATH:

CGATOOLS_VERSION:

omics_pipe.modules.tophat.**tophat**(*sample*, *tophat_flag*)
Runs TopHat to align .fastq files.

input: .fastq file

output: accepted_hits.bam

citation: Trapnell C, Pachter L, Salzberg SL. TopHat: discovering splice junctions with RNA-Seq. Bioinformatics doi:10.1093/bioinformatics/btp120

link: http://tophat.cbcb.umd.edu/

parameters from parameters file: RAW_DATA_DIR:

REF_GENES:

TOPHAT_RESULTS:

BOWTIE_INDEX:

TOPHAT_VERSION:

TOPHAT_OPTIONS:

BOWTIE_VERSION:

SAMTOOLS_VERSION:

omics_pipe.modules.tophat_miRNA.**tophat_miRNA**(*sample*, *tophat_miRNA_flag*)
Runs TopHat to align .fastq files.

input: .fastq file

output: accepted_hits.bam

citation: Trapnell C, Pachter L, Salzberg SL. TopHat: discovering splice junctions with RNA-Seq. Bioinformatics doi:10.1093/bioinformatics/btp120

link: http://tophat.cbcb.umd.edu/

parameters from parameters file: RAW_DATA_DIR:

miRNA_GTF:

TOPHAT_RESULTS:

miRNA_BOWTIE_INDEX:

TOPHAT_VERSION:

TOPHAT_OPTIONS:

BOWTIE_VERSION:

SAMTOOLS_VERSION:

omics_pipe.modules.tophat_ncRNA.**tophat_ncRNA**(*sample*, *tophat_ncRNA_flag*)

Runs TopHat to align .fastq files.

input: .fastq file

output: accepted_hits.bam

citation: Trapnell C, Pachter L, Salzberg SL. TopHat: discovering splice junctions with RNA-Seq. Bioinformatics doi:10.1093/bioinformatics/btp120

link: http://tophat.cbcb.umd.edu/

parameters from parameters file: RAW_DATA_DIR:

REF_GENES:

TOPHAT_RESULTS:

NONCODE_BOWTIE_INDEX:

TOPHAT_VERSION:

TOPHAT_OPTIONS:

BOWTIE_VERSION:

SAMTOOLS_VERSION:

CHAPTER 25

Version History

25.1 1.1.3 (2014/08/22)

*New release for PyPi with bug fixes

25.2 1.1.2b (2014/08/05)

25.2.1 New Features

- Added support for latest GATK version
- Added GATK Group Variant Calling pipeline
- Added noncoding RNA HTseq module

25.2.2 Bug Fixes

- AMI memory handling
- Fixed Sumatra parameters file handling
- RNAseq count based pipeline produces single report for all samples

25.3 1.1.0 (2014/07/09)

First public release!

25.4 1.0.16 (2014/07/01)

25.4.1 New Features

- Added Sumatra for logging of each run to Sumatra database
- Added reading samples from a text file
- Added AWS distribution and docs

25.4.2 Bug Fixes

- Import modules for utils.py sumatra and hgapi

25.5 1.0.15 (2014/06/20)

25.5.1 New Features

- Added TCGA download support

CHAPTER 26

Copyright & License

26.1 Omics Pipe

O

A

B

C

F

G

H

I

M

O

P

R

S

T