DHS_analysis

April 5, 2017

1 DHS Analysis Pipeline

The DHS Analysis pipeline uses a combination of python, bedops and R's DESeq2 package to search for differences in DHS between input samples. To make this easy to use, the pipeline has been loaded on the server as a module.

Because Bill has turned this into a module for you, you will never need to worry about loading any of its requirements/dependencies individually.

1.1 Get the RCC Data Set

To walk you through how to use the pipeline, we will use the RCC dataset that you worked with in previous lessons.

1.1.1 Copy the test data directory to your home directory

From the command line, let's get this information into your home directory and walk through the options.

```
In [ ]: cp -R /home/selasady/public_html/RCC/* ./
```

Change into your new directory:

```
In [ ]: cd ./RCC/
```

1.1.2 Setup your environment variables

First, we'll clean out all the modules you might already have loaded to ensure we don't run into any conflicts:

```
In [ ]: module purge
```

```
In [ ]: module list
```

Then we'll run the "source" command on the .bash_profile file included in the RCC directory. This will setup the environment variables you need to run the script.

```
In [ ]: source ./.bash_profile
```

Now we can safely lost the modules needed to run the pipeline:

```
In [ ]: module load deseq_pipeline
```

```
In [ ]: module list
```

Hint: You can view your paths and environment variables by typing "env" on the command line

```
In [ ]: env
```

1.2 Setup your Experiment Metadata

The pipeline requires two input files: - A text file containing the aggregation ID and grouping information for your experiment - A config file that let's you specify experimental paramaters

1.2.1 Sample Metadata

In your RCC folder, the metadata.txt file contains a tab-separated list of aggregate ID's to analyze, and their associated grouping information. Let's take a look at that file:

```
In [ ]: cat ./metadata.txt
```

```
In [ ]: agg_id    condition
        AG3819    RCC
        AG3898    TUB
        AG3899    TUB
        AG6993    RCC
```

You can add more complexity to your experimental design by adding additional columns, and you can even add a separate column for labels: - If you create a label column, you must specify it in your config file - Any label column you create will appear on the graphs - The metadata file must be tab separated - Your metadata file must contain "agg_id" and "condition" columns

```
In [ ]: agg_id    condition    time    label
        AG3819    RCC          day1    rccDay1
        AG3898    TUB          day2    tubDay2
        AG3899    TUB          day1    tubDay1
        AG6993    RCC          day2    rccDay2
```

1.2.2 Config File

In the RCC directory, config.txt contains the minimum amount of options required to run the pipeline.

You can always find a copy of this file on the Altius GitHub repo

```
In [ ]: cat ./config.txt
```

```
In [ ]: [paths]

        # path to metadata file with agg_id's
        agg_list=./metadata.txt

        [transforms]

        # if a DHS is not found in this many samples, remove it
```

```
# enter 0 for no filter
sample_threshold=1

[deseq_options]

# experimental design for DESeq
# example (treatment, or treatment + time)
experimental_design=condition

# contrast argument for DESeq (column,var1,var2)
# example "condition,wt,dnmt3"
contrasts=condition,RCC,TUB
```

Additional options The config file above leaves a lot of options set at default. The full confing file with all available options can be found on the Altius GitHub repo.

In [ ]:
```
[paths]

# base path for analysis output
working_dir=/home/my_experiment/

# path to master_dhs file, or none to create from input samples
input_master=/path/to/my/master_dhs.bed

# path to aggregation id's, one per line
agg_list=/home/my_experiment/agg_ids.txt

[transforms]

# if a DHS is not found in this many samples, remove it# enter 0 for no fil
sample_threshold=1

# choose custom for John Lazar's normalization script
# choose none to use DESeq filtering
norm_method=custom

# if using John Lazar's normalization method,
# select true to use geomean, false for loess
geomean=true

[deseq_options]

# experimental design for DESeq
# example (treatment, or treatment + time)
experimental_design="condition + time"

# contrast argument for DESeq (column,var1,var2)
# example "condition,wt,dnmt3"
```

```
contrasts="condition,WT,DNMT3+"

# true to save intermediate DESeq and results objects
save_r_objects=true
```

1.3 Run the pipeline

Now that we have our experiment setup, we can run the pipeline by pointing the module to our configuration file:

```
In [ ]: deseq.py -c ./config.txt
```

1.4 View the Results

Results are output into the following directories:

- log files for python and R script: check these if anything went wrong, or to view input parameters

input_files

- symlinks to peak and count files
- metadata.txt with input agg id's, condition, sample ID, library number, taxonomy, assay, view URL and file locations

Counts

- master_dhs.bed
- individual overlap count files
- annotated, combined counts files
- raw counts and normalized counts
- normalization factors

Results

- deseq2_results.bed: all results found for the comparison between your contrasts
- deseq2_significant_results.bed: significant results (adj p<.01) for your contrasts
- Saved R objects
- Pics directory containing PDF and PNG versions of heatmaps, ma plot, counts of significantly different DHS sites, and top 8 loci with greatest fold change

1.5 Get more information

The Altius GitHub repo for the project contains: - Runtime instructions - Example metadata and config files - Additional information on experimental design

The DESeq2 manual provides additional resources for setting up your experimental design and interpreting the results.

If you need help, please contact compbio!