Grumpy

GRUMPY Evaluations

Generative Research Utility Model in Python

QC:

Usage:

```
usage: grumpy.py [-h] -p {cutrun,chip,gsea} [-i INPUTDIRECTORY]
[-o OUTPUTDIRECTORY]
[-r {auto,std,gsealist,gseareport,decode}] [-f FORCE]
[-k APIKEY] [--context CONTEXT]
[--species {human,mouse,human_mouse,other}]
[--outfilesPrefix OUTFILESPREFIX] [--hidden HIDDEN]
```

To run Grumpy within current working directory of the standard report:

```
python ~/programs/GIT/sjcab_std_report/commonbin/grumpy.py -p chip
```

Example input QC table:

```
Sample Total(M) Unique Reads(M) Mapping Rate(%) Duplication Rate(%)
Final Reads(M) Fragment Size(bp) Qtag RSC pkCalling PeaksControl
PeaksNoControl
HepG2-POLR2G-GSM3393557 50832538 35321216 92.94% 25.24% 29008747 145 1
1.191042 narrow 152044 51341
HepG2-input-GSM3393574 75322314 67513248 93.59% 4.23% 58689812 100 2
1.590658 unknown n/a n/a
```

Example concise output:

```
The sample `HepG2-POLR2G-GSM3393557` targeting POLR2G, a transcription
factor, shows good quality metrics. The mapping rate is high at 92.94%,
well above the recommended 80%. The duplication rate is slightly below
the threshold of 30% at 25.24%. The final read count is 29M, which is
sufficient for narrow peak calling. The fragment size is 145bp and the
```

Qtag is 1, indicating a good RSC value. The number of peaks called is also satisfactory with 152044 using control and 51341 without control.

The control sample `HepG2-input-GSM3393574` also exhibits good quality. The mapping rate is high at 93.59% and the duplication rate is very low at 4.23%. The final read count is significantly high at 58.6M. The fragment size is 100bp and the Qtag is 2, indicating an excellent RSC value. As expected, no peaks were called in this control sample.

In summary, both samples are of good quality. The POLR2G target seems to be well captured in the ChIP-seq experiment based on the number of peaks called and other QC metrics. The control sample also meets the quality criteria.

GRUMPY working in GSEA/PathwayEnrichment mode

Example command to run the analysis:

```
python ~/programs/GIT/sjcab_std_report/commonbin/grumpy.py \
-i
C2_shared_strict_selected.intervene/setsCounted/541.11111011010_HNd2_vs_
HNd2-_HNd7_vs_HNd7-_HNd9_vs_HNd9-_HNd5_vs_HNd5-
_HNd9_vs_Pd12_HNd3_vs_HNd3-_HNd5_vs_Pd7_HNd12_vs_HNd12-.txt \
--outfilesPrefix 541_pathways \
--context ../context.txt \
-p gsea
```

Example command to decode the result:

```
python ~/programs/GIT/sjcab_std_report/commonbin/grumpy.py -i
541_pathways.evaluation.html -p gsea --reportType decode
```

Run Grumpy on the standard output report from GSEApy:

```
python ~/programs/GIT/sjcab_std_report/commonbin/grumpy.py \
-i
/research_jude/rgs01_jude/groups/cab/projects/automapper/common/wrosikie
/DEV_projects/gptHelper/GSEA/publicData_GSE164073/GSEA/GSE164073_FCPRank
.weighted/individualRunsGSEA/RNK_GSE164073_diff.sclera_CoV2_vs_sclera_mo
ck.regulation.rank_/GMT_h.all.v2023.2.Hs.symbols \
--outfilesPrefix GSE164073_sclera \
```

```
--context context.txt \
-p gsea \
-r gseareport
```

Running Grumpy using separate / private OpenAI instance (importantn e.g. for debugging)

```
python
/research/rgs01/home/clusterHome/wrosikie/programs/GIT/sjcab_std_report/
commonbin/grumpy.py \
-p chip \
-i /research_jude/rgs01_jude/groups/mulligrp/projects/IKZF1-
N159Y/common/CAB/20240610_publicDataSearch/STDrep/report \
-o /scratch_space/wrosikie/STDrep/CHIP/MULLI-GSE145841-CHIPSEQ \
--apikey
/research/rgs01/home/clusterHome/wrosikie/APIKEY/myKey_GrumpyAPI \
--apiType openai \
--gptModel "gpt-4o"
```