MuG - Process TSV Pipelines Documentation

Release 0.1

Mark McDowall

Oct 31, 2018

Table of Contents

1 Requirements and Installation **1**
- 1.1 Requirements . . . 1
- 1.2 Installation . . . 1

2 Pipelines **3**
- 2.1 BED File Indexing . . . 3
- 2.2 WIG File Indexing . . . 4
- 2.3 GFF3 File Indexing . . . 5
- 2.4 3D JSON Indexing . . . 6

3 Tools to index genomic files **9**
- 3.1 BED Indexer . . . 9
- 3.2 WIG Indexer . . . 11
- 3.3 GFF3 Indexer . . . 12
- 3.4 3D JSON Indexer . . . 13

4 License **17**

5 Indices and tables **21**

Python Module Index **23**

CHAPTER 1

Requirements and Installation

1.1 Requirements

1.1.1 Software

- Mongo DB 3.2
- Python 2.7.10+
- SamTools
- bedToBigBed - http://hgdownload.cse.ucsc.edu/admin/exe/linux.x86_64/
- wigToBigWig - http://hgdownload.cse.ucsc.edu/admin/exe/linux.x86_64/
- HDF5

1.1.2 Python Modules

- numpy
- h5py
- pyBigWig
- pysam

1.2 Installation

Directly from GitHub:

```
git clone https://github.com/Multiscale-Genomics/mg-process-files.git
```

Using pip:

```
pip install git+https://github.com/Multiscale-Genomics/mg-process-files.git
```

CHAPTER 2

Pipelines

2.1 BED File Indexing

This pipeline can process bed files into bigbed and HDF5 index files for web use.

2.1.1 Running from the command line

Parameters

assembly [str] Genome assembly ID (e.g. GCA_000001405.22)

chrom [int] Location of chrom.size file

bed_file [str] Location of input bed file

h5_file [str] Location of HDF5 output file

Returns

BigBed [file] BigBed file

HDF5 [file] HDF5 index file

Example

When using a local verion of the [COMPS virtual machine](http://www.bsc.es/computer-sciences/grid-computing/comp-superscalar/downloads-and-documentation):

`chrom.size` file

```
1  123000000
2  50000000
3  25000000
4  10000000
5  5000000
X  75000000
Y  12000000
```

```
runcompss --lang=python /home/compss/mg-process-files/process_bed.py --assembly GCA_
↪000001405.22 --chrom chrom.size --bed_file <data_dir>/expt.bed --h5_file <data_dir>/
↪expt.hdf5
```

2.1.2 Methods

class process_bed.**process_bed**(*configuration=None*)

Workflow to index BED formatted files within the Multiscale Genomics (MuG) Virtural Research Environment (VRE)

run(*input_files*, *metadata*, *output_files*)

Main run function to index the BED files ready for use in the RESTful API. BED files are index in 2 different ways to allow for optimal data retreival. The first is as a bigbed file, this allows the data to get easily extracted as BED documents and served to the user. The second is as an HDF5 file that is used to identify which bed files have information at a given location. This is to help the REST clients make only the required calls to the relevant BED files rather than needing to pole all potential BED files.

Parameters

- **inpout_files** (*list*) – List of file locations
- **metadata** (*list*) –

Returns **outputfiles** – List of locations for the output BED and HDF5 files

Return type list

2.2 WIG File Indexing

This pipeline can process WIG files into bigbed and HDF5 index files for web use.

2.2.1 Running from the command line

Parameters

assembly [str] Genome assembly ID (e.g. GCA_000001405.22)

chrom [int] Location of chrom.size file

wig_file [str] Location of input wig file

h5_file [str] Location of HDF5 output file

Returns

BigWig [file] BigWig file

HDF5 [file] HDF5 index file

Example

When using a local verion of the [COMPS virtual machine](http://www.bsc.es/computer-sciences/grid-computing/comp-superscalar/downloads-and-documentation):

`chrom.size` file:

```
1  123000000
2  50000000
3  25000000
4  10000000
5  5000000
X  75000000
Y  12000000
```

```
runcompss --lang=python /home/compss/mg-process-files/process_wig.py --assembly GCA_
↪000001405.22 --chrom chrom.size --wig_file <data_dir>/expt.wig --h5_file <data_dir>/
↪expt.hdf5
```

2.2.2 Methods

class process_wig.**process_wig**(*configuration=None*)

Workflow to index WIG formatted files within the Multiscale Genomics (MuG) Virtural Research Environment (VRE)

run(*input_files*, *metadata*, *output_files*)

Main run function to index the WIG files ready for use in the RESTful API. WIG files are indexed in 2 different ways to allow for optimal data retreival. The first is as a bigwig file, this allows the data to get easily extracted as WIG documents and served to the user. The second is as an HDF5 file that is used to identify which bed files have information at a given location. This is to help the REST clients make only the required calls to the relevant WIG files rather than needing to pole all potential WIG files.

Parameters

- **inpout_files** (*list*) – List of file locations
- **metadata** (*list*) –

Returns **outputfiles** – List of locations for the output BED and HDF5 files

Return type list

2.3 GFF3 File Indexing

This pipeline can process GFF3 files into Tabix and HDF5 index files for web use.

2.3.1 Running from the command line

Parameters

assembly [str] Genome assembly ID (e.g. GCA_000001405.22)

gff3_file [str] Location of the source gff3 file

h5_file [str] Location of HDF5 index file

Returns

Tabix [file] Tabix index file

HDF5 [file] HDF5 index file

Example

When using a local verion of the [COMPS virtual machine](http://www.bsc.es/computer-sciences/grid-computing/comp-superscalar/downloads-and-documentation):

```
runcompss --lang=python /home/compss/mg-process-files/process_gff3.py --assembly GCA_
↪000001405.22 --gff3_file <data_dir>/expt.gff3 --h5_file <data_dir>/expt.hdf5
```

2.3.2 Methods

class process_wig.**process_wig**(*configuration=None*)

Workflow to index WIG formatted files within the Multiscale Genomics (MuG) Virtural Research Environment (VRE)

run(*input_files*, *metadata*, *output_files*)

Main run function to index the WIG files ready for use in the RESTful API. WIG files are indexed in 2 different ways to allow for optimal data retreival. The first is as a bigwig file, this allows the data to get easily extracted as WIG documents and served to the user. The second is as an HDF5 file that is used to identify which bed files have information at a given location. This is to help the REST clients make only the required calls to the relevant WIG files rather than needing to pole all potential WIG files.

Parameters

- **inpout_files** (*list*) – List of file locations
- **metadata** (*list*) –

Returns **outputfiles** – List of locations for the output BED and HDF5 files

Return type list

2.4 3D JSON Indexing

This pipeline processes the £D JSON models that have been generated via TADbit into a single HDF5 file that can be used as part of a RESTful API for efficient querying and retrieval of the models.

2.4.1 Running from the command line

Parameters

gz_file [str] Location of the input tar.gz file containing all of the output models and data from the TADbit modelling stage.

Returns

HDF5 [file] HDF5 index file

Example

When using a local verion of the [COMPS virtual machine](http://www.bsc.es/computer-sciences/grid-computing/comp-superscalar/downloads-and-documentation):

```
runcompss --lang=python /home/compss/mg-process-files/process_json_3d.py --gz_file
↪<data_dir>/expt.tar.gz
```

2.4.2 Methods

class process_json_3d.**process_json_3d**(*configuration=None*)

Workflow to index JSON formatted files within the Multiscale Genomics (MuG) Virtural Research Environment (VRE) that have been generated as part of the Hi-C analysis pipeline to model the 3D structure of the genome within the nucleus of the cell.

run(*input_files*, *metadata*, *output_files*)

Main run function to index the 3D JSON files that have been generated as part of the Hi-C analysis pipeline to model the 3D structure of the genome within the nucleus of the cellready for use in the RESTful API.

Parameters

- **files_ids** (*list*) –

  file [str] Location of the tar.gz file of JSON files representing the 3D models of the nucleus

- **metadata** (*list*) –

Returns **outputfiles** – List with the location of the HDF5 index file for the given dataset

Return type list

CHAPTER 3

Tools to index genomic files

3.1 BED Indexer

class mg_process_files.tool.bed_indexer.**bedIndexerTool**(*configuration=None*)

Tool for running indexers over a BED file for use in the RESTful API

bed2bigbed(***kwargs*)

BED to BigBed converter

This uses the `bedToBigBed` program binary provided at http://hgdownload.cse.ucsc.edu/admin/exe/linux.x86_64/ to perform the conversion from bed to bigbed.

Parameters

- **file_sorted_bed** (*str*) – Location of the sorted BED file
- **file_chrom** (*str*) – Location of the chrom.size file
- **file_bb** (*str*) – Location of the bigBed file

Example

```
if not self.bed2bigbed(bed_file, chrom_file, bb_file):
    output_metadata.set_exception(
        Exception(
            "bed2bigbed: Could not process files {}, {}.".format(*input_
↪files)))
```

bed2hdf5(***kwargs*)

BED to HDF5 converter

Loads the BED file into the HDF5 index file that gets used by the REST API to determine if there are files that have data in a given region. Overlapping regions are condensed into a single feature block rather than maintaining all of the detail of the original bed file.

Parameters

- **file_id** (*str*) – The file_id as stored by the DM-API so that it can be used for file retrieval later
- **assembly** (*str*) – Assembly of the genome that is getting indexed so that the chromosomes match
- **feature_length** (*int*) – Defines the level of resolution that the features should be recorded at. The 2 options are 1 or 1000. 1 records features at every single base whereas 1000 groups features into 1000bp chunks. The single base pair option should really only be used when features are less than 10bp to
- **file_sorted_bed** (*str*) – Location of the sorted BED file
- **file_hdf5** (*str*) – Location of the HDF5 index file

Example

```
if not self.bed2hdf5(file_id, assembly, bed_file, hdf5_file):
    output_metadata.set_exception(
        Exception(
            "bed2hdf5: Could not process files {}, {}.".format(*input_files)))
```

bed_feature_length(*file_bed*)

BED Feature Length

Function to calcualte the averagte length of a feature in BED file.

Parameters **file_bed** (*str*) – Location of teh BED file

Returns **average_feature_length** – The average length of the features in a BED file.

Return type int

run(*input_files*, *input_metadata*, *output_files*)

Function to run the BED file sorter and indexer so that the files can get searched as part of the REST API

Parameters

- **input_files** (*list*) –

  bed_file [str] Location of the sorted bed file

  chrom_size [str] Location of chrom.size file

  hdf5_file [str] Location of the HDF5 index file

- **metadata** (*list*) –

  file_id [str] file_id used to identify the original bed file

  assembly [str] Genome assembly accession

Returns

bed_file [str] Location of the sorted bed file

bb_file [str] Location of the BigBed file

hdf5_file [str] Location of the HDF5 index file

Return type list

Example

```
import tool

# Bed Indexer
b = tool.bedIndexerTool(self.configuration)
bi, bm = bd.run(
    [bed_file_id, chrom_file_id, hdf5_file_id], [], {'assembly' : assembly}
)
```

3.2 WIG Indexer

class mg_process_files.tool.wig_indexer.**wigIndexerTool**(*configuration=None*)

Tool for running indexers over a WIG file for use in the RESTful API

run(*input_files*, *input_metadata*, *output_files*)

Function to run the WIG file sorter and indexer so that the files can get searched as part of the REST API

Parameters

- **input_files** (*dict*) –

  wig_file [str] Location of the wig file

  chrom_size [str] Location of chrom.size file

  hdf5_file [str] Location of the HDF5 index file

- **meta_data** (*dict*) –

Returns

bw_file [str] Location of the BigWig file

hdf5_file [str] Location of the HDF5 index file

Return type list

wig2bigwig(***kwargs*)

WIG to BigWig converter

This uses the `wigToBigWig` program binary provided at http://hgdownload.cse.ucsc.edu/admin/exe/linux.x86_64/ to perform the conversion from WIG to BigWig.

Parameters

- **file_wig** (*str*) – Location of the wig file
- **file_chrom** (*str*) – Location of the chrom.size file
- **file_bw** (*str*) – Location of the bigWig file

Example

```
if not self.wig2bigwig(wig_file, chrom_file, bw_file):
    output_metadata.set_exception(
        Exception(
            "wig2bigWig: Could not process files {}, {}.".format(*input_
↪files)))
```

wig2hdf5 (***kwargs*)
WIG to HDF5 converter

Loads the WIG file into the HDF5 index file that gets used by the REST API to determine if there are files that have data in a given region. Overlapping regions are condensed into a single feature block rather than maintaining all of the detail of the original WIG file.

Parameters

- **file_id** (*str*) – The file_id as stored by the DMP so that it can be used for file retrieval later
- **assembly** (*str*) – Assembly of the genome that is getting indexed so that the chromosomes match
- **file_wig** (*str*) – Location of the wig file
- **file_hdf5** (*str*) – Location of the HDF5 index file

Example

```
if not self.wig2hdf5(file_id, assembly, wig_file, hdf5_file):
    output_metadata.set_exception(
        Exception(
            "wig2hdf5: Could not process files {}, {}.".format(*input_files)))
```

3.3 GFF3 Indexer

class mg_process_files.tool.gff3_indexer.**gff3IndexerTool**(*configuration=None*)
Tool for running indexers over a WIG file for use in the RESTful API

gff32hdf5 (***kwargs*)
GFF3 to HDF5 converter

Loads the GFF3 file into the HDF5 index file that gets used by the REST API to determine if there are files that have data in a given region. Overlapping regions are condensed into a single feature block rather than maintaining all of the detail of the original bed file.

Parameters

- **file_id** (*str*) – The file_id as stored by the DM-API so that it can be used for file retrieval later
- **assembly** (*str*) – Assembly of the genome that is getting indexed so that the chromosomes match
- **file_sorted_gff3** (*str*) – Location of the sorted GFF3 file
- **file_hdf5** (*str*) – Location of the HDF5 index file

Example

```
if not self.gff32hdf5(file_id, assembly, bed_file, hdf5_file):
    output_metadata.set_exception(
        Exception(
            "gff32hdf5: Could not process files {}, {}.".format(*input_
↪files)))
```

(continues on next page)

(continued from previous page)

gff32tabix(***kwargs*)
GFF3 to Tabix

Compresses the sorted GFF3 file and then uses Tabix to generate an index of the GFF3 file.

Parameters

- **file_sorted_gff3** (*str*) – Location of a sorted GFF3 file
- **file_sorted_gz_gff3** (*str*) – Location of the bgzip compressed GFF3 file
- **file_gff3_tbi** (*str*) – Location of the Tabix index file

Example

```
if not self.gff32tabix(self, file_sorted_gff3, gz_file, tbi_file):
    output_metadata.set_exception(
        Exception(
            "gff32tabix: Could not process files {}, {}.".format(*input_
↪files)))
```

run(*input_files*, *input_metadata*, *output_files*)
Function to run the BED file sorter and indexer so that the files can get searched as part of the REST API

Parameters

- **input_files** (*list*) –

  gff3_file [str] Location of the bed file

  hdf5_file [str] Location of the HDF5 index file

- **meta_data** (*list*) –

  file_id [str] file_id used to identify the original bed file

  assembly [str] Genome assembly accession

Returns

gz_file [str] Location of the sorted gzipped GFF3 file

tbi_file [str] Location of the Tabix index file

hdf5_file [str] Location of the HDF5 index file

Return type list

3.4 3D JSON Indexer

class mg_process_files.tool.json_3d_indexer.**json3dIndexerTool**(*configuration=None*)
Tool for running indexers over 3D JSON files for use in the RESTful API

json2hdf5(***kwargs*)
Genome Model Indexing

Load the JSON files generated by TADbit into a specified HDF5 file. The file includes the x, y and z coordinates of all the models for each region along with the matching stats, clusters, TADs and adjacency values used during the modelling.

Parameters

- **json_files** (*list*) – Locations of all the JSON 3D model files generated by TADbit for a given dataset
- **file_hdf5** (*str*) – Location of the HDF5 index file for this dataset.

Example

```
if not self.json2hdf5(json_files, assembly, wig_file, hdf5_file):
    output_metadata.set_exception(
        Exception(
            "wig2hdf5: Could not process files {}, {}.".format(*input_files)))
```

run (*input_files*, *input_metadata*, *output_files*)

Function to index models of the geome structure generated by TADbit on a per dataset basis so that they can be easily distributed as part of the RESTful API.

Parameters

- **input_files** (*list*) –

  gz_file [str] Location of the archived JSON model files

  hdf5_file [str] Location of the HDF5 index file

- **meta_data** (*list*) –

  file_id [str] file_id used to identify the original wig file

  assembly [str] Genome assembly accession

Returns

hdf5_file [str] Location of the HDF5 index file

Return type list

Example

```
import tool

# WIG Indexer
j3d = tool.json3dIndexerTool(self.configuration)
j3di = j3d.run((gz_file, hdf5_file_id), ())
```

unzipJSON (*file_targz*)

Unzips the zipped folder containing all the models for regions of the genome based on the information within the adjacency matrixes generated by TADbit.

Parameters **archive_location** (*str*) – Location of archived JSON files

Returns **json_file_locations** – List of the locations of the files within an extracted archive

Return type list

Example

```
gz_file = '/home/<user>/test.tar.gz'
json_files = unzip(gz_file)
```

CHAPTER 4

License

Apache License Version 2.0, January 2004 http://www.apache.org/licenses/

1. Definitions.

   "License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

   "Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

   "Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

   "You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

   "Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

   "Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

   "Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

   "Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from

the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "{}" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

mg_process_files.tool, 9

p

process_bed, 3
process_gff3, 5
process_json_3d, 6
process_wig, 4

Index

B

bed2bigbed() (mg_process_files.tool.bed_indexer.bedIndexerTool method), 9
bed2hdf5() (mg_process_files.tool.bed_indexer.bedIndexerTool method), 9
bed_feature_length() (mg_process_files.tool.bed_indexer.bedIndexerTool method), 10
bedIndexerTool (class in mg_process_files.tool.bed_indexer), 9

G

gff32hdf5() (mg_process_files.tool.gff3_indexer.gff3IndexerTool method), 12
gff32tabix() (mg_process_files.tool.gff3_indexer.gff3IndexerTool method), 13
gff3IndexerTool (class in mg_process_files.tool.gff3_indexer), 12

J

json2hdf5() (mg_process_files.tool.json_3d_indexer.json3dIndexerTool method), 13
json3dIndexerTool (class in mg_process_files.tool.json_3d_indexer), 13

M

mg_process_files.tool (module), 9

P

process_bed (class in process_bed), 4
process_bed (module), 3
process_gff3 (module), 5
process_json_3d (class in process_json_3d), 7
process_json_3d (module), 6
process_wig (class in process_wig), 5, 6
process_wig (module), 4

R

run() (mg_process_files.tool.bed_indexer.bedIndexerTool method), 10
run() (mg_process_files.tool.gff3_indexer.gff3IndexerTool method), 13
run() (mg_process_files.tool.json_3d_indexer.json3dIndexerTool method), 14
run() (mg_process_files.tool.wig_indexer.wigIndexerTool method), 11
run() (process_bed.process_bed method), 4
run() (process_json_3d.process_json_3d method), 7
run() (process_wig.process_wig method), 5, 6

U

unzipJSON() (mg_process_files.tool.json_3d_indexer.json3dIndexerTool method), 14

W

wig2bigwig() (mg_process_files.tool.wig_indexer.wigIndexerTool method), 11
wig2hdf5() (mg_process_files.tool.wig_indexer.wigIndexerTool method), 11
wigIndexerTool (class in mg_process_files.tool.wig_indexer), 11