Politecnico di Torino

Bioinformatics

Technical Manual

Authors:
Guido Pio Mariotti
Giuliano Torrisi

22 July 2016

Contents

1 Introduction

GDC Data Portal gives the opportunity to download large amount of genomic data directly from the web or using the GDC Data Transfer Tool. This program allows users to manage only the files downloaded through GDC Data Transfer Tool.

Full compatibility: Python 2.7.
Compatibility: Python 2.6 if module argparse is installed.
Compatibility: Python 3.x if program 2to3 is used.

2 Configuration

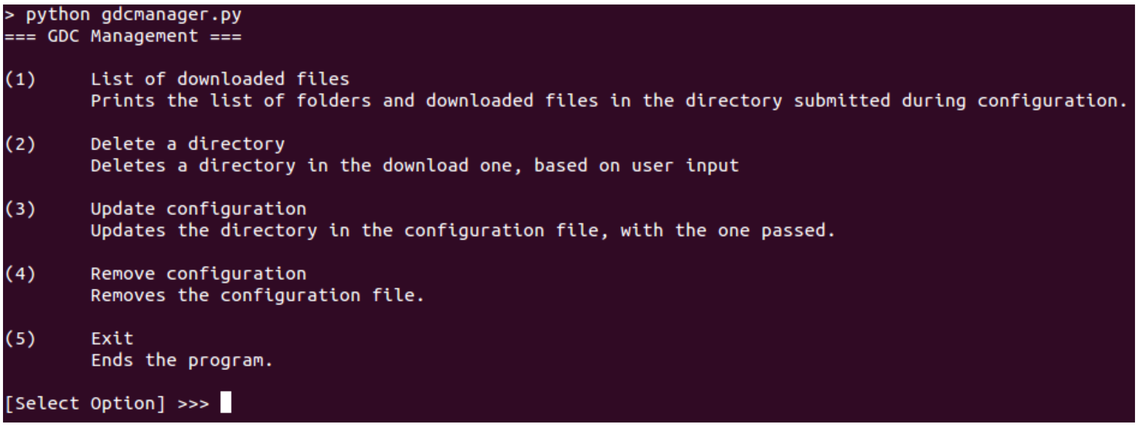
2.1 First Steps

The first time you run the program, it is required to perform a quick configuration. The only information needed is the directory with the downloaded GDC files, that from now on we will refer to as **GDC_download_dir**. This information can be set at first program run, or using the terminal option "-d", as described in *section 4.3*. In order to have more flexibility, is important to insert an absolute path representing the downloads directory, but this is not mandatory, since the program still work even if the path is relative.
After you have created the configuration file, which will be put in the same location where the program is called, there will be two more options to manage it. One is for updating the GDC_download_dir, using the corresponding option during execution or the terminal option **-u** (see *section 4.4* for more information), the other one is the complete remove of the configuration file using the menu or **-rc** terminal option (*section 4.5*).

2.2 How to use the tool

1. Add to GDC Data Portal cart all the files you want to download.
2. Go to cart page and, clicking on “Download” button, choose “Manifest” and “File Metadata”. These files must be copied in the folder chosen during configuration. Manifest are plain text files, while Metadata are json files.
3. From the “GDC_download_dir”, use the GDC Data Transfer Tool to obtain all the files previously selected on the portal. For each file, a new directory will be created using as name the file’s "UUID".

4. Start the tool and something like the image below will be prompted.



```
> python gdcmanager.py
=== GDC Management ===

(1)     List of downloaded files
        Prints the list of folders and downloaded files in the directory submitted during configuration.

(2)     Delete a directory
        Deletes a directory in the download one, based on user input

(3)     Update configuration
        Updates the directory in the configuration file, with the one passed.

(4)     Remove configuration
        Removes the configuration file.

(5)     Exit
        Ends the program.

[Select Option] >>> 
```

3 Functionalities

3.1 Rename Directories

Files downloaded through GDC Data Transfer Tool are saved into a directory (one per file) named as the "UUID" of the data. To each file is associated a name and one or more "entity_submitter_id". These information can be found respectively in Manifest and Metadata files.
The program will rename directories using a concatenation between part of the entity_submitter_id and the file name. For the sake of clarification, we can consider the following example:

Suppose you want to download the following files:

- **File 1**:
  - UUID: “32e84d8b-4132-4aab-8f10-3f78f777e27e”
  - File_name: “00eb7c26-84fd-4ab9-93a1-3d209dfc0f43.FPKM.txt”
  - Entity_submitter_id: “TCGA-M7-A720-01A”
- **File 2**:
  - UUID: “ed77b3cf-a543-439f-8e11-7c1fb9efb157”
  - File_name: “006cb4e4-cb57-4b4f-98ed-475fa7f7f74d_gdc_realn_rehead.bam”
  - Entity_submitter_id: “TCGA-A6-2686-01A”

After the download with GDC Data Transfer Tool, you will find two directories with names:

1. "32e84d8b-4132-4aab-8f10-3f78f777e27e"
2. "ed77b3cf-a543-439f-8e11-7c1fb9efb157"

The tool will rename all of them concatenating the entity submitter id with the file name, with, as separator, an underscore (_).
The result will be:

1. "TCGA-M7-A720-01A_00eb7c26-84fd-4ab9-93a1-3d209dfc0f43.FPKM"
2. "TCGA-A6-2686-01A_006cb4e4-cb57-4b4f-98ed-475fa7f7f74d_gdc_realn_rehead"

Directories will be renamed in three cases:

- Option "List Directories" chosen (explained in *section 3.2*)
- Command line option **-s** or **–scan** chosen (explained in *section 4.1*)
- Command line option **-l** or **–listing** chosen (explained in *section 4.2*)

Manifest and Metadata files The Manifest and Metadata files which have already been used to download data are moved to an hidden directory that works as a cache. This can be found inside "GDC_download_dir" under the name ".cache_gdc". Furthermore for each file a new hidden manifest is created in the same directory, containing all the information related to that file.

Compressed Files Some files downloaded from the GDC Data Portal are compressed. The program will automatically detect and uncompress them.

3.2 List Directories

List all directories present in the main folder in alphabetic order. For each one is also printed the contained file.
The program verifies automatically that all the directories have been renamed and in case run the renaming process, before listing.

3.3 Delete Folder

This option will allow you to remove a directory, and its content, from the GDC_download_dir. First, the list of all the directories, and files inside of them, will be printed in alphabetical order on screen, with a number associated to each directory. Once the list has been printed, you will be prompted to insert the number associated to the directory you want to delete.
Deletion is permanent, and the number associated to each directory is not hardcoded with it, so be careful.

3.4 Update Configuration

This option will allow you to update the configuration file created during the first steps. You will be prompted to insert the new directory in order to update the configuration file. The directory must be a valid absolute or relative path.

3.5 Delete Configuration

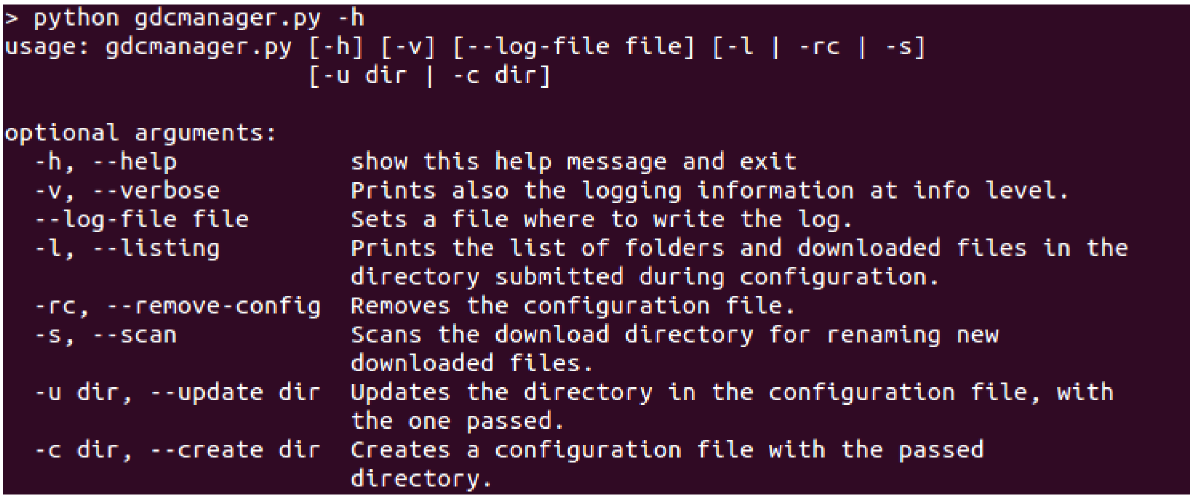
This option will allow you to remove the configuration file created the first time. You will be prompted to insert a new directory in order to create a new configuration and then you will have the possibility to select an option from the menu.
If you are not interested in creating a new configuration, refer to the terminal option **-rc** as explained in *section 4.3*

4 Command Options

You can execute all the functionalities of the tool passing options from command line.

Synopsis:

gdcmanager [options]



```
> python gdcmanager.py -h
usage: gdcmanager.py [-h] [-v] [--log-file file] [-l | -rc | -s]
                     [-u dir | -c dir]

optional arguments:
  -h, --help            show this help message and exit
  -v, --verbose         Prints also the logging information at info level.
  --log-file file       Sets a file where to write the log.
  -l, --listing         Prints the list of folders and downloaded files in the
                        directory submitted during configuration.
  -rc, --remove-config  Removes the configuration file.
  -s, --scan            Scans the download directory for renaming new
                        downloaded files.
  -u dir, --update dir  Updates the directory in the configuration file, with
                        the one passed.
  -c dir, --create dir  Creates a configuration file with the passed
                        directory.
```

4.1 Rename Directories

Command option: “-s” or “--scan”

Can be used with -d and -u
This option allows you to perform just scanning and renaming of the downloaded directories, without the listing, as, instead, happens with the -l option.

4.2 List Directories

Command option: “-l” or “--listing”

Can be used with: -d and -u
List directories and each contained files in alphabetic order. If new files have been downloaded before the last listing, the relative directories will be renamed and the new names printed.

4.3 Create Configuration

Command option: “-d <dir>” or “--directory <dir>”

Can be used with: -l, -rc and -s
This option allows you to create the configuration file using the directory passed as argument.
It will work only if the directory is valid. If the configuration already exist, a message to suggest the use of -u will be print.

4.4 Update Configuration

Command option: “-u <dir>” or “--update <dir>”

Can be used with: -l, -rc and -s
This option allows you to update the GDC_download_dir in the configuration file with the one passed as argument, working exactly as the option in *section 3.4*.
It will work only if the directory is valid. If the configuration doesn’t exist, a message to suggest the use of -d will be print.

4.5 Delete Configuration

Command option: “-rc” or “--remove-config”

Can be used with: -d and -u
The option is the same as the one explained in Delete Folder *section 3.3* but it will prompt you with the request of a new directory for the configuration file.
Use this only if you want to quickly remove the configuration.

4.6 Log Configuration

Command option: “-v” or “--verbose”
Command option: “--log-file” <file>

Can be used with: -s -l -rc -d and -u
The options give the possibility to modify log output. Default configuration will show only errors from WARNING level, without saving into a file.
The -v or --verbose set the log level to INFO, printing all messages from the logger.
The --log-file gives the possibility to save the log on a file.

5 Issues and solutions

5.1 Configuration file management

In order to implement an easy to improve solution, we have decided to use a .ini file. Like most of the file extension, Python offers ConfigParser class in order to perform the parsing and the serialization of the file. This class is present in Python 2.6 and 2.7 but can be easily converted for Python 3.5 using the tool 2to3 offered by the Python Organization. The configuration file has just a section and a <key:value> pair in order to store the directory inserted by the user at creation time. For future versions of the program, new configuration values can be easily inserted in the Configuration class file in order to add new functionalities. Look at “Configuration” section in “Possible Improvements” for future modifications.

5.2 Manage all downloaded files

One of the main problem we had to face with, was how to manage the list of all the files present in our directory.
In order to do that we have decided to create a file, named “MANIFEST.txt”, where we store all the information regarding each file and the name of the folder that contains it.
For each file we save all the info read from the manifest downloaded from the GDC Data Portal and the entity_submitter_id read from the metadata file.

5.3 User deletes a folder manually

In order to handle situations in which a user has deleted manually a folder inside GDC_download_dir, we check if our current version of the “MANIFEST.txt” is compliant with the current state of the folder. If one of the file present in our manifest is no longer available, we delete the entry and update the manifest. This process will be executed all the time that we try to list all the available folders and files.

5.4 Compressed Files

We realized that, in some cases, files downloaded from GDC Data Portal are compressed with “.gz” extension. In order to give the possibility to the user to work with them directly, we decided to uncompress them as soon as we scan the GDC_download_dir to rename all the folders.

5.5 Clear the working folder

Every time a manifest or a metadata is used to download files, it will remain in the same folder until the user will delete it, creating useless garbage.
In order to face with this problem, during the rename process we move all the manifest and metadata files previously downloaded to an hidden folder named “.cache_gdc”, located in GDC_download_dir, and prepend to the manifest or metadata names the string “done_”. This way the user will be able to recollect them if needed.
Furthermore in case one of the downloaded manifest or metadata has not been used yet, it will not be moved to the cache_gdc dir, to allow later use.

5.6 Getting the entity submitter id for each file

During renaming process we retrieve, from the metadata, information on the submitter id of each file, in order to make the name of the folder more readable and to allow each user to link files with related associated entities.
To do that we have used the json python module to read and retrieve information from each metadata file. One of the problem we had to face with was the fact that we could have more than one associated entity and we could not be sure on which one take into account. So we took only the first that we encounter.
Anyway it is very easy to modify the code in order to let the program read another id.

5.7 Menu visualization and Option selection

We have implemented a Menu class to visualize and execute one of the multiple functionalities of the program. To do that while allowing the possibility of future options to be inserted, the Menu object takes as parameter one or more Option objects. Each Option takes as parameter the title of the function, a comment to it and a callable object, that will be called when the option is selected. In order to add new functionalities to the program, is enough to create a class that implements the __call__ method, or simply passing a function pointer to it.

6 Possible improvements

6.1 Configuration

- Multiple directories in order to allow the user to set more than one directory as root for GDC downloaded files.

6.2 Delete Folder

- Terminal option that accept the name of the directory, or a list of directories, to be deleted.

6.3 Uncompress Files

- The program can also handle compressed files with “zip” extension, even if GDC Data Portal does not seem to send files to the user in this format. A possible improvement could be to read the compressed file in chunks to avoid uncontrolled memory consumption (this has been already implemented for “gz” extension).