ucsc-genomic-api: A Python Wrapper of UCSC Genome Browser RESTful API

Eyad Hamza[1], Mazen El-Nabarawy[1], Sohaila Abd-Elhamied[1], Yasmeen Ahmed[1], Salma Ahmed[1], and Sara El-Metwally[2†],

[†] *Corresponding author:*

[1] Medical Informatics Program, Faculty of Computers and Information, Mansoura University, Mansoura 35516, Egypt.

[2] Computer Science Department, Faculty of Computers and Information, Mansoura University, Mansoura 35516, Egypt.

Abstract

UCSC browser has been updated continuously as more data are generated and new features and technologies are released. Recently, UCSC Genome browser has introduced RESTful API to facilitate access the specific genomic regions or data tracks and allow the flexibility of querying the data through scripting languages. ucsc-genomic-api is a python programmable interface based on RESTful API to facilitate the access of UCSC genome browser databases, make a data query more simple, human-understandable, and requiring no prior knowledge of database schemas. The user has a dynamic flexibility to query different genomic intervals, so the data can be retrieved from specific track hub or native assembly results. ucsc-genomic-api is easy to use, readable and extendable to add more functionalities, can be customized for specific user needs, has no dependencies, and compatible with many operating systems. Installation is available through standard python tools from https://pypi.org/project/ucsc-genomic-api/ or using a public repository at : https://github.com/Eyadhamza/UCSC-Genomic-REST-Api-Wrapper

Keywords:

UCSC genome browser, RESTful API, data retrieval, genomic data analysis, python package.

1. Introduction

UCSC Genome browser is a graphical interface for visualizing and annotating genomic data extracted from multiple sources of biological databases. UCSC browser has been updated continuously as more data are generated and new features and technologies are released [1-5]. The browser allows the genomic data exploration on different scales range from individual bases up to chromosomal level along with the annotated information from different species. Also, the browser allows the users to create their own custom visualization track to visualize their imported data, organize it and share it with a research community [6-9].

To download a specific genomic region or data track stored on the UCSC server, the users need to download the complete data set or navigate a specific region through Table Browser. The Table Browser is available as a tool to use the data directly on the web server while the users can initiate the SQL queries to access data tables on their own local machines [3,10-12]. The extracted data can also be converted into different formats such as BED, wiggle, etc. The user can initiate SQL queries to access the data tables locally or on the download server [3,10,13].

The UCSC genomic browser created resources are not flexible to integrate with the most common bioinformatics scripting languages like R, Perl, and Python. The UCSC genome browser team creates a RESTful API to facilitate access the specific genomic regions or data tracks and allow the flexibility of querying the data through scripting languages. The RESTful API programming interface returns the results of different data queries in a JSON format that is required to be parsed further in order to extract the meaningful attributes and its corresponding values [11].

2. Related Work

The public availability of annotated data sets from large scale projects such as ENCODE [14] and UCSC genome browser has been accelerating the genomic research and encourage the research community to analyze their local data in the light of previously annotated features [2,4,5,8-10,15]. The public research community will access the public databases as a routine activity and integrate the data from different sources in order to start the journey of downstream data analysis [16,17].

UCSC genome database APIs or libraries based on scripting language are still limited. Ruby UCSC API is one of the introduced libraries for accessing UCSC genome database based on Ruby programming language. It utilizes the object-relational mapping component in Ruby called ActiveRecord 3. The UCSC genomic database is configured as a module under a namespace called Bio::Ucsc and each table is represented as a class. To retrieve specific data from a table, each class has a set of defined methods based on a database schema that have a compatible naming convention of the ActiveRecord framework methods and allow the easy access and manipulation through it. The current limitations of this API is its dependency on the table schema to find the table associations and the difficulty of tracking the

Manuscript received March 5, 2022
Manuscript revised March 20, 2022
https://doi.org/10.22937/IJCSNS.2022.22.3.94

databases updates automatically. The API does not support other data type formats such as BigWig, BigBed, and Bam formats [18].

For Perl, Genoman [19] library has many interfaces to biological databases including UCSC genome browser. Cruzdb [16] is a python UCSC genome database API based on SQLAlchemy toolkit. CruzDB utilizes an instance of UCSC's MySQL to function properly on a remote data. The local mirror and parallelization can improve the performance of speeding up the process of data access remotely. CruzDB has a dependency on UCSC's MySQL and python SQLAlchemy. It supports Python 2.6 and 2. 7. Cruzdb can utilize a small subset of tables using local mirror of MySQL or SQLite database. The user can update the local copy of the table by adding or removing data and use it as the original one. CruzDB speed up the interval query operations by reading all the features into a memory and creating an interval tree that maps all the genomic features into memory and there is always a tradeoff between a time to load all the features into memory and time to query the tree on memory. CruzDB utilizes a remote UCSC MySQL instance to access the database through repeated SQL queries. The query time can be reduced through SQLite, or mapping the entire table into memory and representing it as a local interval tree to reduce the network overhead.

The UCSC genome browser RESTful API interface has been introduced in 2020 and has the ability to access different types of browser data such as: the available public hubs, genome assemblies, chromosomes, and genomic DNA sequences. The queries initiated by RESTful API interface can access the available public hubs or genome assemblies or navigate through a specific data track or a region in a genome assembly, specific chromosome or DNA sequences [11]. Transitioning data access through REST APIs make a data query more simple, human-understandable, requiring no prior knowledge of queried database (i.e. in theory database schemas). The returned results are well structured in formats like JSON that is better to handle data complexity. The results can be quicker depend on the request size and the utilization of the concurrency concepts.

The community lacks a user friendly simple scripting programming interface based on a RESTful API for querying genomic databases. Developing such interface will allow downstream data integration and analysis and drive the field of biomedical research in many areas related to integrative genomics and bioinformatics.

3. Method

On UCSC genome browser, there are two types of tracks that contain genomic data assemblies and annotations files: native tracks hosted by the UCSC server and hub tracks located remotely on the user's machine and can be accessed via HTTP request. Track hubs can be displayed on the genomes supported by UCSC or your imported sequence [20,21]. The track hubs can be public hubs registered with UCSC genome browser, customized hubs that is created and setup by users, and unlisted and local hubs that can be also imported and manipulated by the users. When the data is imported from a track hub, only requested region is displayed rather than the entire file content. This on-demand transition property overcomes the network overhead to display the content of large data files, reducing the uploading time and increase access speed.

RESTful API is a quick and easy programmable approach for searching and retrieving data from a database. The RESTful API queries are simple, human readable and understandable, and required no prior knowledge of the queried database. The query results will be in a structured format (i.e. JSON or XML) that can capture the data complexity and optimize its concurrency usage. The returned JSON results are in the form of attribute-value pairs that are required to be parsed and processed step further in order to be more useful, understandable and usable (see Fig. 1). Also, most of downstream data cleaning, processing, and analysis usually accomplished via one of the scripting languages like Python that can also use the RESTful API.

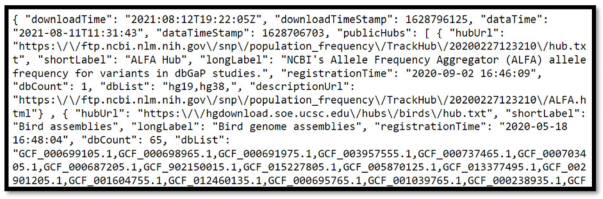
```
{ "downloadTime": "2021:08:12T19:22:05Z", "downloadTimeStamp": 1628796125, "dataTime":
"2021-08-11T11:31:43", "dataTimeStamp": 1628706703, "publicHubs": [ { "hubUrl":
"https:\/\/ftp.ncbi.nlm.nih.gov\/snp\/population_frequency\/TrackHub\/20200227123210\/hub.tx
t", "shortLabel": "ALFA Hub", "longLabel": "NCBI's Allele Frequency Aggregator (ALFA) allele
frequency for variants in dbGaP studies.", "registrationTime": "2020-09-02 16:46:09",
"dbCount": 1, "dbList": "hg19,hg38,", "descriptionUrl":
"https:\/\/ftp.ncbi.nlm.nih.gov\/snp\/population_frequency\/TrackHub\/20200227123210\/ALFA.h
tml"} , { "hubUrl": "https:\/\/hgdownload.soe.ucsc.edu\/hubs\/birds\/hub.txt", "shortLabel":
"Bird assemblies", "longLabel": "Bird genome assemblies", "registrationTime": "2020-05-18
16:48:04", "dbCount": 65, "dbList":
"GCF_000699105.1,GCF_000698965.1,GCF_000691975.1,GCF_003957555.1,GCF_000737465.1,GCF_0007034
05.1,GCF_000687205.1,GCF_902150015.1,GCF_015227805.1,GCF_005870125.1,GCF_013377495.1,GCF_002
901205.1,GCF_001604755.1,GCF_012460135.1,GCF_000695765.1,GCF_001039765.1,GCF_000238935.1,GCF
```

Fig. 1 Example of returned results by UCSC REST API endpoint function /list/publicHubs.

We implemented our Python package (ucsc-genomic-api) to query UCSC genomic databases through RESTful service. ucsc-genomic-api allows the user to retrieve the data from specific track hub or native assembly results. Also, it has a dynamic flexibility for querying specific genomic regions or intervals. ucsc-genomic-api is a simple package that can be easy to use, readable and extendable to add more functionalities, and customized for specific user needs. The API is a plug-in python module that has no dependencies, and is compatible with many operating systems.

There are six primary classes in the ucsc-genomic-api package: Hub, Genome, Track, TrackSchema, Chromosome, and Sequence (Table 1). The defined classes allow retrieving data from different sources such as public hubs, genome assemblies hosted by UCSC server, specific assembly track or track hubs. The retrieved data could be a list of public hubs, data tracks, genome assemblies hosted by UCSC server, genomes, chromosomes, DNA sequences, and a specific data track content (see Fig. 2).

Table 1: The mapping of endpoint functions from UCSC REST API to the created classes of ucsc-genomic-api python package.

UCSC REST API Endpoint functions	***ucsc-genomic-api classes***
/list/publicHubs	Hubs
/list/ucscGenomes /list/hubGenomes	Genomes
/list/tracks /getData/track	Track
/list/chromosomes	Chromosome
/list/schema	TrackSchema
/getData/sequence	Sequence

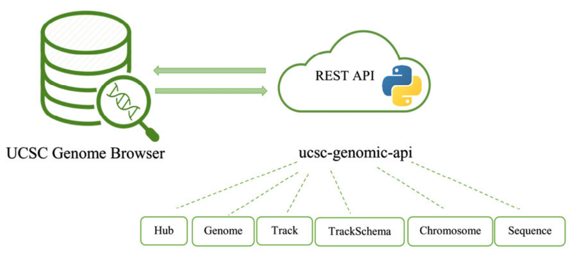

Fig. 2 The six primary classes of ucsc-genomic-api package.

Each class has a set of attributes that can be accessed via a dot (.) notation and has a set of primary methods: get(), find(), findBy(), and exists() that have different parameters according to its corresponding class.

• get(): returns a list of objects from a specific class such as listing the available hubs in the case of using the function with Hub class.

• find(): searches for an object by a specified name such as finding a hub with a name 'ALFA Hub' in the case of using the function with Hub class.

• findBy(): searches for an object by a defined attribute such as finding a hub whose attribute hubName equals ALFA Hub in the case of using the function with Hub class.

• exists(): checks if an object exists such as checking 'ALFA Hub' exists or not in the case of using the function with Hub class.

The previous methods will return their results as python objects or it will throw a not found exception. Also, they are all available with our defined set of classes that target the public hubs, genomes hosted by UCSC server, genomes from specific assembly track or data hub, data tracks, chromosomes, DNA sequences.

Table 2: Parameters that are utilized by the endpoint functions from UCSC REST API and consequently utilized by the methods of ucsc-genomic-api python package.

Parameter	***Meaning***
hubUrl	The URL of assembly/track hub.
genome	The reference genome name located on the genome assemblies hosted by UCSC genome browser or assembly/track hub.
track	The data track name in the genome assemblies hosted by UCSC genome browser or assembly/track hub.
chrom	The chromosome name of a specific DNA sequence or data track.
start	The starting position of data retrieval task from a specific DNA sequence or data track.
end	The ending position of data retrieval task from a specific DNA sequence or data track.
maxItemsOutput	Set a limit on the maximum number of outputs.
trackLeavesOnly	Disable displaying the track container information and only display the tracks information.

4. Use Case

Biological data plays a key role in any bioinformatics analysis pipeline. The downstream analysis steps will be determined according to the data availability and its related features/attributes. There are two basic steps in any bioinformatics data analysis pipeline: retrieving data from genomic sources such as UCSC genome browser and coding/designing your own algorithm/data analysis pipeline in order to apply some computations, infer insights and draw some conclusions about the retrieved data.

ucsc-genomic-api package targets the data retrieval task, making it simpler and faster without initiating a dozen of get requests through the RESTful API. This will help the bioinformaticians to focus on the data processing tasks rather than focusing on the details of querying the data using its related features and defined attributes in JSON format and parsing it accordingly. Also, most of the downstream data analysis tasks and algorithms rely on one of the scripting languages, so the Python programming language is used to implement ucsc-genomic-api package in order to facilitate data manipulation and integration.

One use case example is retrieving the assembly named ‘wuhCor1’ of SARS-CoV-2, its corresponding tracks, download the data or get a portion of it based on some specified coordinates, all of these queries tasks represent a starting step towards many bioinformatics data analysis tasks related to comparative genomics [22]. The following lines of code will display how the data retrieval tasks can be accomplished in a simple, fast, and easy methods using ucsc-genomic-api package.

The ucsc-genomic-api package can be installed using a standard Python pip command:

```
pip install ucsc-genomic-api
```

To retrieve the assembly named ‘*wuhCor1*’ from UCSC genome browser databases, and access its corresponding tracks and sequence, the classes *Genome* and *Sequence* are imported from *ucsc.api* package:

```
from ucsc.api import Genome, Sequence
```

Then the *find* method of Genome class can be used to find the assembly named *'wuhCor1'*:

```
genome = Genome.find('wuhCor1')
```

The returned results will be a python object that can be manipulated further to explore its corresponding attributes such as *name*, *organism*, etc. or list all of the available object attributes via __*dict*__ attribute.

The *tracks* associated with the queried assembly can be accessed by *genome.tracks* and each *track* is an object with a set of defined attributes that can be listed via __*dict*__ attribute. If a specific track is required and its name (i.e. *microdel*) is known, the method *findTrack* can be used to query it and explore its attributes:

```
track = genome.findTrack('microdel')
```

Also, the track can be queried using some of its defined attributes (i.e. *shortLabel*) via a method called *findTrackBy*:

```
track= genome.findTrackBy('shortLabel','Microdeletions')
```

To access the data corresponding to a specific track along with its associated attributes, a method *getTrackData* is used:

```
trackFragments =
                track.getTrackData(genome='wuhCor1')
```

The track data could be retrieved further based on chromosome name (i.e. *'NC_045512v2'*) or other defined parameters listed in Table 2:

```
chrFrag = track.getTrackData (genome='wuhCor1',
                              chrom='NC_045512v2')
```

The data can be downloaded for an offline processing via a method *downloadData*:

```
track.downloadData(genome='wuhCor1',
                   chrom='NC_045512v2')
```

The portion of chromosome sequence can be queried based on a specific coordinates provided to the get method of class *Sequence*:

```
sequence = Sequence.get(genome= 'wuhCor1',chrom=
'NC_045512v2',start=4321,end=5678)
```

The bases of requested region can be printed/accessed using the attribute *dna* of the created sequence object: *sequence.dna*

The user guide of ucsc-genomic-api package contain examples of using the package classes/methods in different scenarios that target the public hubs, genomes hosted by UCSC server, genomes from specific assembly track or data hub, data tracks, chromosomes, DNA sequences. The user guide can be located in the package public repository at: https://github.com/Eyadhamza/UCSC-Genomic-REST-Api-Wrapper

5. Conclusion

UCSC genome browser provides a set of programmable interfaces to facilitate the access of different hub data tracks and native genome assemblies. RESTful API is one of the recently introduced interfaces in order to make a query simpler and easy to handle the data complexity and concurrency usage. The query results will be in a structured JSON format that is required to be parsed and processed step further in order to have a meaningful information and clear insights. ucsc-genomic-api package allows querying the data from public hubs, genomes hosted by UCSC server, genomes from specific assembly track or data hub, data tracks, chromosomes, DNA sequences, making the data retrieval task simpler and faster without initiating a dozen of get requests through the RESTful API. Accordingly, most of downstream data analysis pipelines using our package will focus on data processing and computational approaches rather than data retrieval, parsing and reformatting in order to make it readable and understandable. Python programming language is used to implement the ucsc-genomic-api package in order to facilitate data manipulation and integration in any existing data analysis frameworks. This will open the opportunity to apply powerful computational techniques such as machine and deep learning on the retrieved data and its defined attributes to draw new conclusions and meaningful insights.

References

[1] M. E. Mangan, J. M. Williams, R. M. Kuhn, et al., "The UCSC genome browser: what every molecular biologist should know," Current protocols in molecular biology, vol. Chapter 19, pp. Unit19.9-Unit19.9, 2009.

[2] R. M. Kuhn, D. Haussler, and W. J. Kent, "The UCSC genome browser and associated tools," Briefings in Bioinformatics, vol. 14, pp. 144-161, 2012.

[3] W. J. Kent, C. W. Sugnet, T. S. Furey, et al., "The human genome browser at UCSC," Genome research, vol. 12, pp. 996-1006, 2002.

[4] P. A. Fujita, B. Rhead, A. S. Zweig, et al., "The UCSC genome browser database: update 2011," Nucleic Acids Research, vol. 39, pp. D876-D882, 2010.

[5] B. Rhead, D. Karolchik, R. M. Kuhn, et al., "The UCSC genome browser database: update 2010," Nucleic Acids Research, vol. 38, pp. D613-D619, 2010.

[6] R. Buels, E. Yao, C. M. Diesh, et al., "JBrowse: a dynamic web platform for genome visualization and analysis," Genome Biology, vol. 17, p. 66, 2016/04/12 2016.

[7] J. Stalker, B. Gibbins, P. Meidl, et al., "The Ensembl Web site: mechanics of a genome browser," Genome research, vol. 14, pp. 951-955, 2004.

[8] T. R. Dreszer, D. Karolchik, A. S. Zweig, et al., "The UCSC Genome Browser database: extensions and updates 2011," Nucleic Acids Research, vol. 40, pp. D918-D923, 2012.

[9] R. M. Kuhn, D. Karolchik, A. S. Zweig, et al., "The UCSC genome browser database: update 2009," Nucleic Acids Research, vol. 37, pp. D755-D761, 2009.

[10] J. Navarro Gonzalez, A. S. Zweig, M. L. Speir, et al., "The UCSC Genome Browser database: 2021 update," Nucleic Acids Res, vol. 49, pp. D1046-D1057, Jan 8 2021.

[11] C. M. Lee, G. P. Barber, J. Casper, et al., "UCSC Genome Browser enters 20th year," Nucleic Acids Res, vol. 48, pp. D756-D761, Jan 8 2020.

[12] J. Casper, A. S. Zweig, C. Villarreal, et al., "The UCSC Genome Browser database: 2018 update," Nucleic Acids Research, vol. 46, pp. D762-D769, 2018.

[13] M. Haeussler, A. S. Zweig, C. Tyner, et al., "The UCSC Genome Browser database: 2019 update," Nucleic Acids Research, vol. 47, pp. D853-D858, 2018.

[14] I. Dunham, A. Kundaje, S. F. Aldred, et al., "An integrated encyclopedia of DNA elements in the human genome," Nature, vol. 489, pp. 57-74, 2012/09/01 2012.

[15] C. Tyner, G. P. Barber, J. Casper, et al., "The UCSC Genome Browser database: 2017 update," Nucleic Acids Research, vol. 45, pp. D626-D634, 2017.

[16] B. S. Pedersen, I. V. Yang, and S. De, "CruzDB: software for annotation of genomic intervals with UCSC genome-browser database," Bioinformatics (Oxford, England), vol. 29, pp. 3003-3006, 2013.

[17] D. Karolchik, G. P. Barber, J. Casper, et al., "The UCSC Genome Browser database: 2014 update," Nucleic Acids Research, vol. 42, pp. D764-D770, 2014.

[18] H. Mishima, J. Aerts, T. Katayama, et al., "The Ruby UCSC API: accessing the UCSC genome database using Ruby," BMC Bioinformatics, vol. 13, p. 240, 2012/09/21 2012.

[19] https://mybiosoftware.com/genoman-library-accessing-manipulating-genome-annotation-data.html. Accessed on:June 26, 2021.[Online].Available: https://mybiosoftware.com/genoman-library-accessing-manipulating-genome-annotation-data.html

[20] D. Karolchik, A. S. Hinrichs, T. S. Furey, et al., "The UCSC Table Browser data retrieval tool," Nucleic Acids Research, vol. 32, pp. D493-D496, 2004.

[21] B. J. Raney, T. R. Dreszer, G. P. Barber, et al., "Track data hubs enable visualization of user-defined genome-wide annotations on the UCSC Genome Browser," Bioinformatics, vol. 30, pp. 1003-1005, 2013.

[22] J. D. Fernandes, A. S. Hinrichs, H. Clawson, et al., "The UCSC SARS-CoV-2 genome browser," Nature genetics, vol. 52, pp. 991-998, 2020.