Jumpstart your genomics pipelines with genomepy

This manuscript (permalink) was automatically generated from vanheeringen-lab/genomepy_manuscript@b4c6062 on November 27, 2020.

Authors

- **Siebren Frolich**
  0000-0001-6925-8446 · siebrenf
  Department of Molecular Developmental Biology, Radboud University

- **Maarten van der Sande**
  0000-0001-7803-1526 · Maarten-vd-Sande · MaartenvdSande
  Department of Molecular Developmental Biology, Radboud University

- **Simon van Heeringen**
  0000-0002-0411-3219 · simonvh · svheeringen
  Department of Molecular Developmental Biology, Radboud University

Abstract

Summary

Analyzing genomics data, including RNA-, ATAC- and ChIP-sequencing, requires multiple types of support data, such as genome sequence and gene annotations. These resources can generally be retrieved from multiple organizations, where they exist at multiple versions, and may have been generated with varying methods. Which data to use depends on the context of the research, such as collaboration partners, data reuse or data quality. Many of the bioinformatic workflows and pipelines available to date require the user to make this informed decision and supply the support data. Obtaining this data can be a tedious and error-prone process and does not allow for full computational reproducibility.

Here we present genomepy, a quality-of-life enhancement tool, that can navigate the genome databases of Ensembl, UCSC and NCBI. Genomepy can search and install genome sequences and gene annotation data from these providers in a consistent, reproducible and documented manner. The search function retrieves genomes related to the search term, and can do so for one or all providers to allow the user to make an informed decision. The install function downloads a specified genome with sensible defaults, while providing full control to advanced features. Additionally, gene annotations can be downloaded and converted to commonly used formats, with built-in checks for compatibility with the genome. Genomepy can optionally create genome indexes for commonly used aligners, including splice-aware aligners utilizing both genome and gene annotations. Genomes and gene annotations not available on supported databases can be processed by genomepy as well, providing a consistent workflow with any genome.

Genomepy provides this functionality and more via command line interface and Python application programming interface, aimed at easy of use and integration in automated pipelines.

Availability and implementation

Genomepy can be installed using Bioconda and Pip, and the code is available at https://github.com/vanheeringen-lab/genomepy.

Introduction

As high-throughput sequencing matured over the past decade and a half, the size and amount of genomic data hes exploded. Over the past five years, the number of datasets published on the NCBI GEO database increased by an average of 2000 per year, while the number of samples increased by 100.000 per year 1. This explosion of data highlights the need for scalable, robust and automatable methods for data processing.

Much of the genomics analysis preprocessing has already been made automatable, with multiple tools providing standardized output formats. One such step is the alignment of sequence read data to a genome, such as Bowtie2 2, BWA 3, GMAP 4 or Minimap2 5, and splice-aware aligners such as STAR 6 and HISAT2 7. Several steps, mainly including alignment, require additional input to the sequence read data in the form of a reference genome, and in the case of splice-aware aligners, gene annotations. These data can be obtained from a variety of different sources. There are the three major genome providers, which act as a general hub for reference data: Ensembl 8, UCSC 9 and NCBI 10.

Ensembl uses their in-house workflow to add or update genomes and gene annotations in a three-month production cycle 11. As a results, Ensembl provides detailed gene annotations on mature genome assemblies that update infrequently. One downside to Ensembl data is their chromsome naming scheme (e.g. “1” for chromosome 1) which clashes with several common bioinformatical tools.

UCSC hosts and maintains a modest set of reference genomes. Gene annotations for these genomes are generated through a variety of methods, including the Ensembl and NCBI workflow, as well as their own. However, not every version of these gene annotations conforms to the output format.

The NCBI database accepts submissions from the Genome Reference Consortium as well as individuals. In addition to reference assemblies by the reference consortium, uploads by individual groups often provide a trove of different strains per species. For instance, 946 different strains of Homo sapiens and 848 strains of Saccharomyces cerevisiae are available from NCBI. As a result of the open submission system, NCBI updates frequently, and often provides the latest version of an assembly before the other providers do. However, as an upload may contain either genome assembly, gene annotation, or both, the assembly data can be incomplete, and of varying levels of maturity.

In addition to the three major providers, there are many species specific providers, such as flybase 12, wormbase 13 or xenbase 14.

Table 1: Available genome assemblies per provider. Estimated by querying the provider REST API (assembly summaries for NCBI) for all unique assembly names. Ensembl genomes are excluding 43778 bacteria genomes not available programmatically.

Provider	Assemblies
UCSC	213
Ensembl	1741
NCBI	878821

Each provider has a different method of generating genome assemblies and gene annotations, which can affect available output (formats), naming schema, information density, as well as availability (see table 1), accessibility (archive and retrieval methods) and relevance (update frequency). These differences impact the compatibility of the reference data with research tools 15, other reference databases and other research. Therefore, the choice of provider and reference data is significant importance. To make an informed decision requires an overview of available options. To get this overview, one could check each website separately, then download and process the data manually. Such a method creates room for human error in the finding, processing and remembering these steps as well as the reasoning behind them. For the sake of sanity and reproducibility, it would be better if that could be done in a standardized system (see figure 1).

To this end we developed genomepy. Genopmepy is a tool with both command line interface and Python application programming interface, which can be called to search one or all three providers at once. Using the search function one can get an overview of all genomes containing the search term in their name, description or accession identifier, as well as all genomes matching a taxonomy identifier. Once a selection is made the genome and gene annotations can be downloaded and prepared for use with the install function. This includes automatic preparation for aligners (genome indexing with pyfaidx 16, generating support files (chromosome sizes and gaps), matching chromosome names between genome and gene annotation and optional aligner index generation). Which of these features is used is automatic logged for reproducibility. Because of the multiple interfaces, genomepy can be used in workflows to automate these steps.

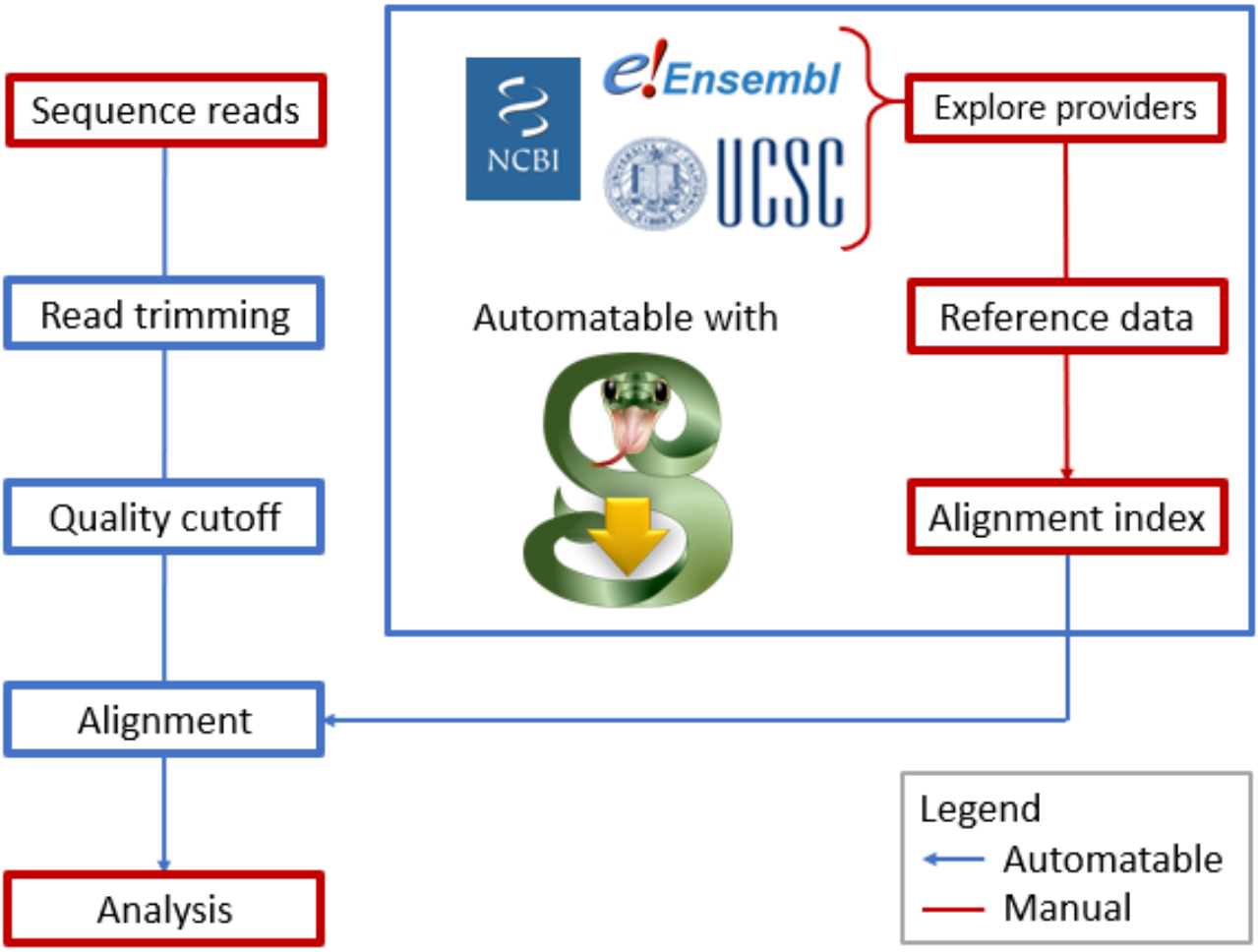

Figure 1: Overview of a sequence alignment workflow reviewing the steps automatable by genomepy

Related Work

Ensembl, UCSC and NCBI all support downloading from their individual databases via accessible FTP archives, web portals, and REST APIs. To access these databases programmatically, there exists several external tools, such as the ncbi-genome-download tool [17] and ucsc-genomes-downloader [18]. However, to our knowledge no tool exists that can search or download from all three major genome providers.

There are several existing tools for reproducibly sharing reference data between projects. These data management tools accept reference data and derived assest such as aligner indexes from any source, such as iGenomes [19], refGenie [20] and Go Get Data [21]. These tools excel in their ability to reproducibly share data, a feature which is not present in genomepy, may therefore be used to obtain previously generated data with ease. However, these tools require the user to supply the reference data to any new assembly (such as non-model organisms), new assembly version (such as the latest path to the human genome) or asset (such as an aligner index not present in the hosted data). For these situations, data management tools would be an excellent extension to genomepy.

In several cases the reference data may not be ready for direct downstream use. For instance, many assemblies do not contain gene annotations in the correct format for splice-aware aligners. Furthermore, many gene annotations have contig (chromosomes and scaffolds) names that do not match the names in the reference genome. Additional steps, including compatibility checks and potentially processing, are required. Many tools exist to perform these actions, most noteably the UCSC gene annotation conversion tools. However, it should not bear mentioning that and automated checklist but would be more efficient that a manual one.

We conclude that there is a need for a tool that can provide an overview of the choices of reference data available, can obtain the specified data, and perform the processing required to utilize the data downstream. Genomepy was created to fit this need, and does so for both automated and human-supervised workflows.

genomepy

The core functionality of genomepy is to search, download and prepare genomes and gene annotations. These functions are split over two CLI functions: `genomepy search` and `genomepy install`.

Search

Search will query all providers for a given search term. If desired only one provider can be searched at a time. Once entered, the search term is normalized for case and whitespace for ease of use. Genomepy can search for text terms, taxonomy identifiers and accession numbers, and will automatically detect which is used in the search term. For taxonomy identifiers, all assemblies with matching IDs are returned. For accession numbers and text terms, all assemblies containing the term in their respective field are returned. Additionally text terms found in any other descriptive fields are also returned.

Install

When an assembly has been selected, the name can be passed to the install function. If the assembly is available from multiple providers, then specific provider can be passed as well. If an assembly from an unsupported provider is preferred, direct download links may be supplied in order to receive the same processing. This may be useful if the external provider contains a novel or more recent assembly of organisms in their specialized field. This results in a consistent output from any desired provider.

The function first downloads the genome assembly with soft masking (repetitive sequences written in lower case). Sequence masking can be turned off, or set to hard (repetitive sequences written as Ns).

Reference assemblies often contain alternate sequences to reflect biological diversity. For the purpose of sequence alignment however, the best results are obtained if there is one reference per nucleotide. Therefore genomepy filters out alternative regions, unless specified otherwise. Additional regex filters may be passed to either include or exclude sequences by name. For instance to filter for chromosomes, or filter out unplaced contigs or the mitochondrial sequence.

Once processing is performed, genomepy generates several commonly used support files. The genome is indexed using pyfaidx [16], and contig sizes and contig gap sizes are stored in separate files.

If specified, genomepy will attempt to download a gene annotation: genomepy will search the database for a GFF, GTF, BED or (for UCSC only) text format gene annotation. The annotation is then processed to the commonly used GTF and BED output formats using the UCSC conversion tools [22]/. If the genome is present after downloading the annotation, the contig names between the genome and gene annotation are checked for compatibility. Should the contig names mismatch, genomepy will attempt to match the names in the annotations to those in the genome.

During the installation process, a README file is created which logs the URLs to the source files, the steps performed, and contigs filtered out. The file is updated should the process is expanded later,

such as by downloading other assets or running a plugin.

Plugins

Genomepy facilitates several optional processing steps via plugins. These include downloading of blacklists (by the Kundaje lab 23) for the genomes that have them, and the generation of aligner indexes. These options can be toggled using the `genomepy plugin` function, which also includes and overview of available plugins. Enabled plugins will run upon (re)running the install function.

Blacklists are available for several UCSC genomes, and GRCh38. If no blacklist is available the program will proceed after giving a warning.

Genomepy support generation of several popular genome aligners: Bowtie2, BWA, GMAP or Minimap2, and splice-aware aligners such as STAR and HISAT2. The splice-aware aligners function both with and without gene annotations. Should no gene annotation be downloaded, the program will issue a warning and proceed to generate a splice-unaware aligner. By default genomepy supports multithreading aligner indexing. Unless specified, up to eight cores are used depending on the number available.

Provider indexing

In order to search, genomepy acquires an overview of available assemblies from each supported providers when first required. For Ensembl and UCSC, this is possible via their REST API. NCBI does not provide a REST API for their genome database, but their FTP archives contain several assembly overviews. Each of these databases is converted to a dictionary with the assembly name as key. The databases are supplied as dictionaries as well, resulting in a nested dictionary. Each provider uses a different filing system, which genomepy parses to group together similar features. For instance, NCBI uses two fields for taxonomy identifiers ("species_taxid" and "taxid"), which are both used by genomepy when searching by taxonomy identifier.

The parsed databases are stored in a local cache for seven days in order to balance quick lookups, while staying up to date.

```
>>> import genomepy
>>> p = genomepy.ProviderBase.create("ucsc")
Downloading assembly summaries from UCSC
>>> g = p.genomes
>>> next(iter(g))
'ailMel1'
>>> g['ailMel1']
{'description': 'Dec. 2009 (BGI-Shenzhen 1.0/ailMel1)', 'nibPath':
        '/gbdb/ailMel1', 'organism': 'Panda', 'defaultPos':
        'GL192818.1:558576-566855', 'active': 1, 'orderKey': 16070,
        'genome': 'Panda', 'scientificName': 'Ailuropoda melanoleuca',
        'htmlPath': '/gbdb/ailMel1/html/description.html', 'hgNearOk': 0,
        'hgPbOk': 0, 'sourceName': 'BGI-Shenzhen AilMel 1.0 Dec. 2009',
        'taxId': 9646}
```

Command line example

Here we demonstrate a typical example using genomepy. Commands are indicated by a dollar sign, and standard output was trimmed for brevity.

```
$ genomepy search "drosophila mel" --provider ucsc
name    provider    accession          species                    tax_id
        other_info
dm1     UCSC        na                 Drosophila melanogaster    7227
        Jan. 2003 (BDGP R3/dm1)
dm2     UCSC        na                 Drosophila melanogaster    7227
        Apr. 2004 (BDGP R4/dm2)
dm3     UCSC        GCA_000001215.2    Drosophila melanogaster    7227
        Apr. 2006 (BDGP R5/dm3)
dm6     UCSC        GCA_000001215.4    Drosophila melanogaster    7227
        Aug. 2014 (BDGP Release 6 + ISO1 MT/dm6)
 ^
 Use name for genomepy install

$ genomepy plugin enable blacklist star
Enabled plugins: blacklist, star

$ genomepy install dm3 --annotation
Downloading genome from UCSC.
Target URL:
        http://hgdownload.soe.ucsc.edu/goldenPath/dm3/bigZips/chromFa.tar.gz..

Genome download successful, starting post processing...

name: dm3
local name: dm3
fasta: /home/siebrenf/.local/share/genomes/dm3/dm3.fa
Downloading annotation from UCSC.
Target URL:
        http://hgdownload.soe.ucsc.edu/goldenPath/dm3/bigZips/genes/dm3.ensGen


Annotation download successful
Creating star index...
Downloading blacklist
        http://mitra.stanford.edu/kundaje/akundaje/release/blacklists/dm3-
        D.melanogaster/dm3-blacklist.bed.gz

$ ls ~/.local/share/genomes/dm3
dm3.annotation.bed.gz  dm3.annotation.gtf.gz  dm3.blacklist.bed  dm3.fa
        dm3.fa.fai  dm3.fa.sizes  dm3.gaps.bed  index  README.txt

$ ls ~/.local/share/genomes/dm3/index
star

$ head ~/.local/share/genomes/dm3/README.txt
name: dm3
```

```
provider: UCSC
original name: dm3
original filename: chromFa.tar.gz
assembly_accession: GCA_000001215.2
tax_id: 7227
mask: soft
genome url:
        http://hgdownload.soe.ucsc.edu/goldenPath/dm3/bigZips/chromFa.tar.gz
annotation url:
        http://hgdownload.soe.ucsc.edu/goldenPath/dm3/bigZips/genes/dm3.ensGen

sanitized annotation: not required
```

Python example

The core genomepy functions, such as `search` and `install` are exposed on import. In order to visualize the install function the individual steps are executed in this example, rather than calling the `genomepy.install()` function directly. Commands are indicated by `>>>`, and prints were trimmed for brevity.

```
>>> import os
>>> import genomepy

>>> for row in genomepy.search("melanogaster", "ucsc"):
>>>     print("\t".join([x.decode('utf-8') for x in row]))

dm1 UCSC    na  Drosophila melanogaster 7227    Jan. 2003 (BDGP R3/dm1)
dm2 UCSC    na  Drosophila melanogaster 7227    Apr. 2004 (BDGP R4/dm2)
dm3 UCSC    GCA_000001215.2 Drosophila melanogaster 7227    Apr. 2006 (BDGP
        R5/dm3)
dm6 UCSC    GCA_000001215.4 Drosophila melanogaster 7227    Aug. 2014 (BDGP
        Release 6 + ISO1 MT/dm6)

>>> p = genomepy.ProviderBase.create("ucsc")
>>> p.download_genome("dm3", annotation=True)
Genome download successful, starting post processing...
name: dm3
local name: dm3
fasta: /home/siebrenf/.local/share/genomes/dm3/dm3.fa

>>> genome = genomepy.Genome("dm3")
>>> plugins = genomepy.plugin.init_plugins()
>>> plugins["star"].after_genome_download(genome)
Creating star index...

>>> plugins["blacklist"].after_genome_download(genome)
Downloading blacklist http://mitra.stanford.edu/kundaje/akundaje/release/black
        blacklist.bed.gz


>>> p.download_annotation("dm3")
Downloading annotation from UCSC.
Target URL:
        http://hgdownload.soe.ucsc.edu/goldenPath/dm3/bigZips/genes/dm3.ensGen

Annotation download successful

>>> genomepy.utils.sanitize_annotation(genome)

>>> path = os.path.expanduser("~/.local/share/genomes/dm3")
>>> os.listdir(path)
['dm3.annotation.gtf.gz', 'dm3.blacklist.bed', 'dm3.gaps.bed', 'dm3.fa',
        'index', 'dm3.annotation.bed.gz', 'README.txt', 'dm3.fa.sizes',
        'dm3.fa.fai']
```

Conclusion

A principal step in all science is making informed decisions. For genomics projects, choosing a genome to serve as reference assembly is no different. Genomepy offers an overview of the three largest genome providers, serving as a catalyst for this step.

After choosing an assembly, data must be downloaded and processed for compatibility with downstream tools. Genomepy provides this functionality, which includes downloading, zipping and unzipping, converting, filtering and altering of the data, while providing logging of the steps undertaken. Even if the required reference data is not available on the three largest genome providers, genomepy can process external data to provide a consistent output.

While genomepy makes choices during the processing, each of these can be tuned to the specific needs of a project using the CLI or Python API. Combined, these features make genomepy ideal for integration in automated sequencing workflows, as demonstrated in seq2science [24].

Future prospects

As science strives to become more open, genomepy assists by making the discussed steps easier, and make the output more FAIR [25]. One method by which this can be further improved is coupling genomepy with a data management tool, such as one discussed in the related works.

Acknowledgements

We thank the Department of Molecular (Developmental) Biology, out github contributors, and issue posters for their patience, feedback and insight. We thank black, pytest, CodeCoverage and TravisCI for enduring our abuse and teaching us patience. And finally, we thank Manubot [26] for assisting with this manuscript.

Code availability

Genomepy can be installed using Bioconda and Pip. The code is available at https://github.com/vanheeringen-lab/genomepy.

Supplementary Information

The full genomepy documentation including examples can be viewed here

References

1. **NCBI GEO: archive for functional genomics data sets–update**
   Tanya Barrett, Stephen E Wilhite, Pierre Ledoux, Carlos Evangelista, Irene F Kim, Maxim Tomashevsky, Kimberly A Marshall, Katherine H Phillippy, Patti M Sherman, Michelle Holko, … Alexandra Soboleva
   Nucleic acids research (2013-01) https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3531084/
   DOI: 10.1093/nar/gks1193 · PMID: 23193258 · PMCID: PMC3531084

2. **Fast gapped-read alignment with Bowtie 2**
   Ben Langmead, Steven L. Salzberg
   Nature Methods (2012-04) https://www.nature.com/articles/nmeth.1923
   DOI: 10.1038/nmeth.1923

3. **Validate User** https://academic.oup.com/crawlprevention/governor?content=%2fbioinformatics%2farticle-lookup%2fdoi%2f10.1093%2fbioinformatics%2fbtp324

4. **Validate User** https://academic.oup.com/crawlprevention/governor?content=%2fbioinformatics%2farticle-lookup%2fdoi%2f10.1093%2fbioinformatics%2fbti310

5. **Validate User** https://academic.oup.com/crawlprevention/governor?content=%2fbioinformatics%2farticle%2f34%2f18%2f3094%2f4994778

6. **Validate User** https://academic.oup.com/crawlprevention/governor?content=%2fbioinformatics%2farticle-lookup%2fdoi%2f10.1093%2fbioinformatics%2fbts635

7. **HISAT: a fast spliced aligner with low memory requirements**
   Daehwan Kim, Ben Langmead, Steven L. Salzberg
   Nature Methods (2015-04) https://www.nature.com/articles/nmeth.3317
   DOI: 10.1038/nmeth.3317

8. **Validate User** https://academic.oup.com/crawlprevention/governor?content=%2fnar%2fadvance-article%2fdoi%2f10.1093%2fnar%2fgkz966%2f5613682

9. **The Human Genome Browser at UCSC**
   W. J. Kent, C. W. Sugnet, T. S. Furey, K. M. Roskin, T. H. Pringle, A. M. Zahler, a. D. Haussler
   Genome Research (2002-05-16) https://doi.org/fpf5rm
   DOI: 10.1101/gr.229102 · PMID: 12045153 · PMCID: PMC186604

10. **The UCSC Table Browser data retrieval tool**
    Donna Karolchik, Angela S Hinrichs, Terrence S Furey, Krishna M Roskin, Charles W Sugnet, David Haussler, W James Kent
    Nucleic acids research (2004-01-01) https://www.ncbi.nlm.nih.gov/pmc/articles/PMC308837/
    DOI: 10.1093/nar/gkh103 · PMID: 14681465 · PMCID: PMC308837

11. **Release Cycle** https://ensembl.org/info/about/release_cycle.html

12. **Validate User** https://academic.oup.com/crawlprevention/governor?content=%2fnar%2farticle%2f47%2fD1%2fD759%2f5144957

13. **Validate User** https://academic.oup.com/crawlprevention/governor?content=%2fnar%2fadvance-article%2fdoi%2f10.1093%2fnar%2fgkz920%2f5603222

14. **Xenbase: a genomic, epigenomic and transcriptomic model organism database**
Kamran Karimi, Joshua D Fortriede, Vaneet S Lotay, Kevin A Burns, Dong Zhou Wang, Malcom E Fisher, Troy J Pells, Christina James-Zorn, Ying Wang, V G Ponferrada, … Peter D Vize
Nucleic Acids Research (2018-01-04) https://doi.org/ghk99d
DOI: 10.1093/nar/gkx936 · PMID: 29059324 · PMCID: PMC5753396

15. **A comprehensive evaluation of ensembl, RefSeq, and UCSC annotations in the context of RNA-seq read mapping and gene quantification**
Shanrong Zhao, Baohong Zhang
BMC Genomics (2015-02-18) https://doi.org/10.1186/s12864-015-1308-8
DOI: 10.1186/s12864-015-1308-8

16. **Efficient "pythonic" access to FASTA files using pyfaidx**
Matthew D. Shirley, Zhaorong Ma, Brent S. Pedersen, Sarah J. Wheelan
PeerJ PrePrints (2015-04-08) https://peerj.com/preprints/970

17. **kblin/ncbi-genome-download**
Kai Blin
(2020-11-25) https://github.com/kblin/ncbi-genome-download

18. **ucsc-genomes-downloader: Python package to quickly download genomes from the UCSC.**
Luca Cappelletti
https://github.com/LucaCappelletti94/ucsc_genomes_downloader

19. **iGenomes** https://support.illumina.com/sequencing/sequencing_software/igenome.html

20. **Validate User** https://academic.oup.com/crawlprevention/governor?content=%2fgigascience%2farticle%2fdoi%2f10.1093%2fgigascience%2fgiz149%2f5717403

21. **Go Get Data (GGD): simple, reproducible access to scientific data**
Michael J. Cormier, Jonathan R. Belyeu, Brent S. Pedersen, Joseph Brown, Johannes Koster, Aaron R. Quinlan
bioRxiv (2020-09-11) https://www.biorxiv.org/content/10.1101/2020.09.10.291377v2
DOI: 10.1101/2020.09.10.291377

22. **Index of /admin/exe** http://hgdownload.cse.ucsc.edu/admin/exe/

23. **The ENCODE Blacklist: Identification of Problematic Regions of the Genome**
Haley M. Amemiya, Anshul Kundaje, Alan P. Boyle
Scientific Reports (2019-06-27) https://www.nature.com/articles/s41598-019-45839-z
DOI: 10.1038/s41598-019-45839-z

24. **seq2science**
Maarten Van Der Sande, Siebren Frölich, Jos Smits, Simon Van Heeringen
Zenodo (2020-11-26) https://doi.org/ghktg7
DOI: 10.5281/zenodo.3921913

25. **The FAIR Guiding Principles for scientific data management and stewardship**
Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E. Bourne, …

Barend Mons
Scientific Data (2016-03-15) https://doi.org/bdd4
DOI: 10.1038/sdata.2016.18 · PMID: 26978244 · PMCID: PMC4792175

26. **Open collaborative writing with Manubot**
Daniel S. Himmelstein, Vincent Rubinetti, David R. Slochower, Dongbo Hu, Venkat S. Malladi, Casey S. Greene, Anthony Gitter
PLOS Computational Biology (2019-06-24) https://doi.org/c7np
DOI: 10.1371/journal.pcbi.1007128 · PMID: 31233491 · PMCID: PMC6611653