# Transparent Incremental Updates for Genomics Data Analysis Pipelines

Edvard Pedersen[1,3], Nils Peder Willassen[2,3], and Lars Ailo Bongo[1,3]

[1] Department of Computer Science, University of Tromsø, Norway
[2] Department of Chemistry, University of Tromsø, Norway
[3] Centre for Bioinformatics, University of Tromsø, Norway
{edvard.pedersen,nils-peder.willassen}@uit.no,
larsab@cs.uit.no

**Abstract.** A large up-to-date compendium of integrated genomic data is often required for biological data analysis. The compendium can be tens of terabytes in size, and must often be frequently updated with new experimental or meta-data. Manual compendium update is cumbersome, requires a lot of unnecessary computation, and it may result in errors or inconsistencies in the compendium. We propose a transparent file based approach for adding incremental update capabilities to unmodified genomics data analysis tools and pipeline workflow managers. This approach is implemented in the GeStore system. We evaluate GeStore using a real world genomics compendium. Our results show that it is easy to add incremental updates to genomics data processing pipelines, and that incremental updates can reduce the computation time such that it becomes practical to maintain large-scale up-to-date genomics compendia on small clusters.

## 1 Introduction

Recent advances in scientific instruments, such as next-generation sequencing machines, has the potential of producing data that provide views of biological processes at different resolutions and conditions, opening a new era in molecular biology and molecular medicine [1]. Many of the data analysis techniques developed for analyzing such genomic data integrate data from many experiments with metadata from multiple knowledge bases. The information in the knowledge bases [2] is essential for understanding the biological content of the experiment data. For example, the results of DNA sequencing may not become truly useful before the UniProt [3] knowledge base is used to map sequence bases to genes, the per gene results are compared to results from other experiments, and the significant differences have been mapped to biological functions using the Go [4] knowledge base.

Genomic data integration and analysis is typically implemented as a pipeline with several tools, where the output files of one tool acts as the input files for the next tool. The specific set of tools to use depends on the biological problem that is being investigated. Often large amounts of data must be analyzed, since new sequencing machines produce multiple terabytes of data per experiment [5]. The cost of the analysis can therefore be orders of magnitude larger than the cost of creating the data to be analyzed [6]. Cost efficient data analysis is therefore a key challenge for genomics data analysis.

D. an Mey et al. (Eds.): Euro-Par 2013 Workshops, LNCS 8374, pp. 311–320, 2014.

Integrating new experimental data or updating meta-data may provide novel biological insights. It is therefore important to update a biological compendium when new data becomes available. Meta-data updates are especially important since it represents the state of knowledge in the field [7]. However, current biological analysis tools typically require recalculating the entire compendium for meta-data updates. Such full updates increase the computational time and cost; often to the point where reanalysis is not done.

The cost of reanalysis can be greatly reduced by using incremental updates [8] that limit recomputation to new and updated data. We believe such an approach for incremental updates of genomic data must satisfy the following four requirements. First, most existing analysis tools should be supported without any modifications since it is not practical to maintain modified versions of the many analysis tools used in genomic data analysis pipelines. Transparent incremental updates are therefore needed. Second, the incremental updates should be independent of the job and resource management systems used to run the pipeline tools since genomic analysis pipelines are run on many different platforms. Third, it should be easy to add incremental update support to an existing pipeline. The system should therefore handle update detection, processing of incremental updates, and integration of the incremental update with a previous full update. Fourth, it should scale to large-scale compendium.

To our knowledge, no previous incremental update systems for large-scale data [9–15] satisfy all four requirements. These provide the required functionality and scalability, but do not provide the easy to use transparent incremental updates that are necessary to add incremental updates to existing pipelines. Instead they require either porting applications to a specific framework such as Dryad [16] or MapReduce [17], or writing ad-hoc scripts for input generation and output merging.

We present the GeStore system for incremental update management. GeStore uses a transparent file based approach that satisfies all four requirements. Most pipeline tools take as input one or more files with input and meta-data, and produce output consisting of one or more files. Incremental updates can therefore be implemented by modifying the input or meta-data files such that these only contain the data for an incremental update, and then merging the incremental output with the previous results. Tool code is unmodified, and the only modifications to the pipeline are two GeStore calls for generating input files and merging output files. GeStore provides a plugin framework for implementing parsers, tool-specific incremental file generators, and tool-specific output file mergers. GeStore uses the Hadoop software stack for scalable data processing.

Our contributions are threefold: (i) we propose a promising approach for adding incremental updates to unmodified genomic data analysis pipelines, leading to substantial reduction in time and resources needed to update large biological compendium, (ii) we present the design and implementation of our system, including a framework for implementing plugins that enable transparent incremental updates, and (iii) we present the feasibility of our approach and initial experimental evaluation of our system using a real metagenomics analysis pipeline and real data.

## 2 GeStore

GeStore is a system for enabling transparent incremental computations for unmodified file-based data analysis pipelines. GeStore consists of a runtime system that provides a plugin framework for incremental input file generation and output file merging, a toolset for parsing and detecting changes in files, and data storage and management (Fig. 1, left). GeStore exports an interface for data feeders, and an interface for workflow managers. Data feeders are typically scripts that periodically download new input data or updated meta-data from remote repositories or local storage systems. All downloaded files are stored in GeStore. Pipeline managers (or pipeline configurations) are modified to call GeStore before running each tool in the pipeline in order to generate the incremental input files used by the tools and to merge the resulting files (Fig. 1, right). GeStore uses HDFS [18] and Hbase [19] to efficiently store incremental files and GeStore meta-data, and Hadoop MapReduce [17] to run scalable change detection jobs. In addition GeStore comprise library functions and tools to add incremental updates to pipeline tools, and client applications to access data stored within GeStore.

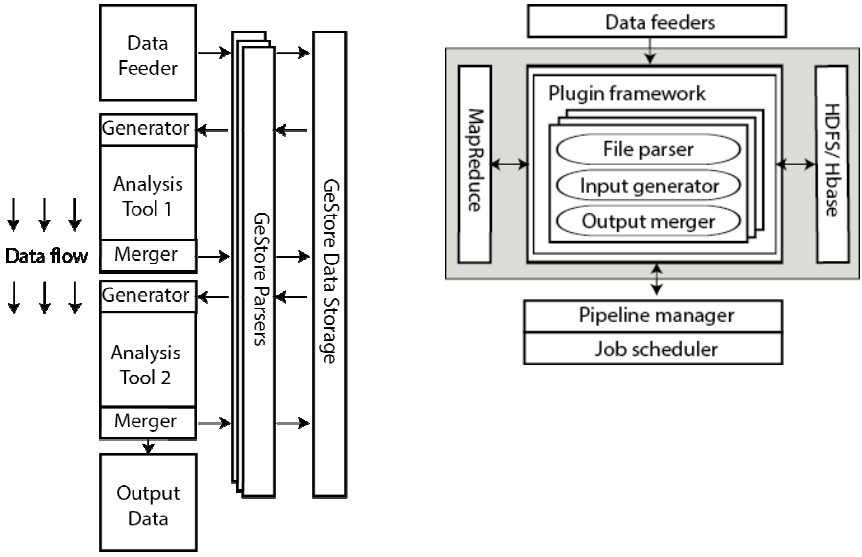


**Fig. 1.** Incremental pipeline execution (left). GeStore architecture (right).

### 2.1 File Based Incremental Updates

GeStore uses a transparent file based approach where incremental updates are implemented by modifying the input and meta-data files read by genomics data analysis tools such that these only contain the data for incremental update computations. The tool will then be run as normal, but it will typically produce a partial result. The partial result is merged with previously produced results and stored in GeStore.

We have chosen a file based approach since there are relatively few file formats that are used by many genomics applications. It is therefore feasible to implement parsers that support most file formats and therefore most genomics pipeline tools. In addition, most file formats are simple and structured which makes it easy to write parsers for each format. However, update file generation and output merging is not trivial to implement. It may be necessary to understand the biological content of the data and how the tools read and write the data. For example, for the widely used BLAST [20] tool most changes to the UniProt [3] input data records are for fields that are not used in the computation, and the output data records contains a field (e-value [15]) that is incorrect for incremental updates. Both of these issues can be fixed by writing relatively simple code for ignoring the insignificant fields during change detection and by fixing the e-values in the output data. In addition, the system must provide, low overhead storage for incremental update data, and efficient generation of incremental update files.

GeStore provides an interface that the pipeline system uses to request one or more incremental update input files, and to merge the partial results with previously produced results. These upcalls can be added by adding stages to the pipeline configuration before and after the execute tool step. Alternatively, the GeStore calls can be added by modifying the code in the pipeline manger that manages the lists of input, meta-data, and output files used to setup a tool for execution and to store the results.

### 2.2 Plugin Framework

GeStore provides a plugin framework to support many different file formats and pipeline tools. To add incremental updates to a pipeline the administrator must first write a plugin for each tool in the pipeline. These are then used by GeStore for each incremental update. A plugin comprise three parts: (i) a parser for each file type used by the pipeline tool, (ii) tool-specific incremental file generator, and (iii) tool-specific incremental output file merger. Each plugin has typically a few tens of lines of code. Many plugins also share parsers and file mergers, and GeStore provides many library functions for parsing, change detection, and merging of files. The plugins are managed by a framework that provides efficient data storage, and low overhead file parsing, generation, and merging.

The file parser must define schemas for the input files and meta files used by a tool, and implement six methods that: (i) provide regular expressions that define the start and end of an entry in the file, (ii) split an entry into columns, (iii) compare two versions of an entry, (iv) check if an entry is well-formed, (v) put the entry into HBase, and (vi) generate output in other formats. The file generator requires implementing one method that specifies the parsers to use for each file format, and the fields to write to the input file. The output merger requires implementing a method to merge the output data with previously produced output data stored in GeStore. This may include minor fixes to output data fields as discussed above.

### 2.3 Data Storage and Management

GeStore uses the Hadoop software stack for scalable data storage and processing. GeStore maintains: (i) HBase tables and HDFS files with pipeline tool input, output and meta-data file data, (ii) a cache of previously generated incremental update files stored in HDFS, and (iii) HBase tables with per plugin instance state that is used to generate file, merge output files, and provide provenance information.

For file types that have a parser implemented, the data is split into entries and entry fields. These are stored as rows and columns in HBase using a file-format specific schema. The only required column in the schema is a unique ID for each row. The HBase schema can be modified by adding new columns to the table if for example the file format changes or the parser is modified. GeStore use the versioning mechanism in HBase to store only updated fields (i.e. delta compression), and to return the data for a given time period. The timestamp for a version corresponds to either the file generation date, release date, or version.

## 3 Incremental METApipe

METApipe is used by our biology collaborators to find novel enzymes by analyzing sequence data from marine microbial communities. METApipe is currently run using the GePan pipeline management system (developed by Tim Kahlke at the University of Tromsø). It includes the following tools:

1. Multiple Genome Aligner (MGA): [21] does alignment of closely related DNA sequences. It does not require meta-data from knowledge bases.
2. MGA-exporter: converts the MGA output to the format used by the next stage.
3. FileScheduler: partitions and distributes the input data to the compute nodes.
4. Protein BLAST (BLASTP) [20]: maps sequences to information from the UniProt Swiss-Prot and TrEMBL [3] knowledge bases.
5. HMMer [22]: maps sequences to information from the Pfam-A and Pfam-B [23] protein family databases.
6. Annotator: gathers the results from the preceding tools, and converts the data to a custom format.
7. Annotator-exporter: converts the annotator output to a format that can be used by data visualization and exploration tools.

To add incremental updates to METApipe we had to write parsers for the six file formats used by the pipeline: FASTA, UniprotKB meta-data, Pfam meta-data, BLAST output, HMMer output, and MGA output. We also had to write plugins for the BLAST and HMMer tools. The BLAST plugin corrects incremental e-values as discussed in [15] during merge. The HMMer plugin only generates input files.

The file format plugins were a total of 844 lines of Java code, and the tool plugins were 283 lines of Java code. The results show that file based incremental could be used for all tools in METApipe, and there are relatively few lines of tool specific code.

To integrate GeStore with METApipe, we modified the code that generates the Grid Engine [24] scripts that run the pipeline tool code. GeStore calls were added by changing the file initialization commands to GeStore calls. In total, about 120 lines of code were changed in METApipe. We expect the changes required to other pipeline management systems to be similarly small.

# 4 Evaluation

Our initial experimental evaluation compares the benefits and overheads of using incremental updates for the METApipe metagenomics analysis pipeline. Our experiments were run on a small cluster with one frontend and eight compute nodes. Each node is equipped with two Intel Xeon E5-1620 CPUs running at 3.6 GHz and 32 gigabytes of RAM. They have a total of 4.5 TB of local HDD. They also have 2.6 TB of NFS storage shared between them. The machines are connected using gigabit Ethernet. We believe such a cluster configuration is realistic for research labs that maintain genomic compendia.

We use a 15 mega base pairs metagenomics dataset from the Yellowstone Park [25] as input data. Processing this small dataset takes 2.5 hours on our small cluster. We incrementally update the dataset on the last day of the month from January 2011 to July 2011. There were 6 updates to Uniprot Swiss-Prot and TrEMBL, and one update to Pfam A and B in that period.

## 4.1 Update Relationships

To analyze relationships between meta-data changes and input file changes, we averaged all changes in UniProt TrEMBL, UniProt Swiss-Prot, and Pfam-A meta-data collections between January and July in 2011(Table 1). In Swiss-Prot and TrEMBL most changes are to annotation that does not require BLAST recomputation, and hence a significant difference in incremental update execution time. Pfam has a naïve plugin that marks all changes as significant, and has therefore a high rate of significant changes (100%). The Pfam plugin could be improved by doing more precise classifications of non-significant updates. These results demonstrate the benefits of tool specific plugins.

**Table 1.** Monthly meta-data collection updates between January-July 2011. Averages reported.

| | Total entries | Total updates | Significant | New entries |
|---|---|---|---|---|
| Swiss-Prot | 527590 | 38.76% | 0.44% | 0.40% |
| TrEMBL | 14738346 | 32.11% | 4.89% | 4.88% |
| Pfam-A | 1076 | 100.00% | 100.00% | 3.25% |

### 4.2 GeStore Improvements and Overhead

We measured METApipe execution time for full updates and incremental updates with 1, 3, and 6 month periods (Table 2). The analysis time is dominated by BLASTp. Since

BLAST execution time scales linearly with the input size, the smaller incremental input data generated by GeStore significantly reduce BLAST analysis time, and hence total execution time.

**Table 2.** METApipe execution time split into analysis time and GeStore overhead (all in seconds)

| | Analysis | Overhead | Total |
|---|---|---|---|
| Full update (Jan 2011) | 9141 | 0 | 9141 |
| with GeStore | 10718 | 2562 | 13280 |
| Incremental (Jan – Feb) | 893 | 755 | 1647 |
| Incremental (Jan – April) | 1736 | 3497 | 5233 |
| Incremental (Jan – Juli) | 2850 | 3736 | 6586 |

GeStore has an overhead for HMMer of 800 seconds when generating a complete database, and 300 seconds when retrieving a cached database. Generating an incremental update database takes 2800 seconds, this is because the PFam plugin marks all updates as important. BLASTp has an overhead of 1700 seconds for generating a full database. The incremental update time is 300 to 800 seconds depending on the size of the update.

Although GeStore overhead is significant for these experiments it will be much smaller for bigger, more realistic, input dataset sizes since the analysis time depends on input data size, while GeStore overhead depends on meta-data size. In addition, we expect to reduce the Pfam change detection overhead by implementing data aware change detection (as discussed above).

The storage overhead increases sub linearly for UniProt since there are relatively few updates per month (as shown above), the January UniProt database file size is 33 Gb. When stored in HBase it requires 48 Gb of space. However, the total size of the UniProt databases is 252 Gb, but only requires 77 Gb of space in GeStore. For Pfam the storage requirements increase linearly, from to 3.3 Gb to 7.1 Gb and 2.9 Gb and 6.3 Gb respectively for GeStore and total file size. The storage requirements can be significantly reduced by improving the plugin for HMMer.

GeStore achieves similar analysis runtime improvements (90%, for 5% meta-data updates) to incremental BLAST as reported in [15]. Execution time improvements ranging from 20% to 99% are reported in [9–14], but for applications from the data center domain. We have not experimentally compared the execution time improvements and overheads to other large scale incremental update tools since these require modifications to the pipeline tools.

## 5 Related Work

Systems and frameworks for incremental updates on large scale datasets include Incoop [10], Percolator [11], Nectar [9], DryadInc [12], CBP [13], and HaLoop [14]. In Percolator and CBP the programmer implements a system specific incremental program using respectively event-driven mini transactions and stateful primitives. Incoop, Nectar, DyradInc, and HaLoop use data dependency graphs of Dryad [16] or MapReduce programs to automatically replace the input data for a computation with previously calculated results. GeStore combines these two main approaches; a programmer implements file generators and mergers for unmodified programs. GeStore is independent of the programming model and job management system, so the applications can be executed using Dryad, MapReduce [11], or the Grid Engine [24].

GeStore extends the work in [15] by providing a framework and libraries to implement the necessary pre and post processing of data moved between a data warehouse and genomic analysis tools. This makes it easier to add additional support for additional genomic analysis tools as we have demonstrated by implementing incremental updates for a complete metagenomics analysis pipeline.

Simple change detection is supported by tools such as Unix diff, delta encoding compression systems [26], and version management systems such as CVS [27]. However, the change detection in these do not take into account the complex inter-file relationships found in genomic datasets.

The file tables maintained by GeStore are similar to declarative views maintained by data warehouses [28]. Incremental updates have also been used for non-distributed computation result caching (memoization) as in [8].

Popular approaches for genomics pipeline management are Galaxy [29] and BioConductor [30]. These do not provide incremental computation.

We evaluated GeStore using the locally developed METApipe pipeline. An alternative is the JCVI metagenomics analysis pipeline [31].

## 6 Conclusions and Future Work

We proposed an approach for adding incremental updates to unmodified genomic data analysis pipelines, leading to substantial reduction in time and resources needed to update large biological compendiums. We presented the design and implementation of the GeStore system, including a framework for implementing plugins that enable transparent incremental updates. We demonstrated the feasibility of our approach and provided an initial experimental evaluation of our system using a real metagenomics analysis pipeline and real data. The cost effective transparent incremental updates provided by GeStore makes it practical to frequently update large genomic compendium with new experimental and meta-data, and thereby enabling novel biological discoveries.

We plan to further evaluate the benefits and overhead of incremental updates for genomics data analysis by applying GeStore to the pipeline producing data for the IMP [32] tool, and to a Galaxy [29] pipeline. Galaxy can also be used to provide a GUI for GeStore configuration and data management.

**Acknowledgements.** Thanks to Espen Robertsen and Tim Kahlke for help with the GePan pipeline, Jon Ivar Kristiansen for maintaining our cluster, and Martin Erntsen for his comments.

## References

[1] Schuster, S.C.: Next-generation sequencing transforms today's biology. Nature Methods 5(1), 16–18 (2008)
[2] Galperin, M.Y., Fernández-Suárez, X.M.: The 2012 Nucleic Acids Research Database Issue and the online Molecular Biology Database Collection. Nucleic Acids Research 40(Database issue), D1–D8 (2012)
[3] Magrane, M., UniProt Consortium: UniProt Knowledgebase: a hub of integrated protein data. Database the Journal of Biological Databases and Curation 2011 (2011)
[4] Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D., Butler, H., Cherry, J.M., Davis, A.P., Dolinski, K., Dwight, S.S., Eppig, J.T., Harris, M.A., Hill, D.P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J.C., Richardson, J.E., Ringwald, M., Rubin, G.M., Sherlock, G.: Gene Ontology: tool for the unification of biology. Nature Genetics 25(1), 25–29 (2000)
[5] Kahn, S.D.: On the Future of Genomic Data. Science 331(6018), 728–729 (2011)
[6] Wilkening, J., Wilke, A., Desai, N., Meyer, F.: Using clouds for metagenomics: A case study. In: 2009 IEEE International Conference on Cluster Computing and Workshops, pp. 1–6 (2009)
[7] Sandberg, R., Larsson, O.: Improved precision and accuracy for microarrays using updated probe set definitions. BMC Bioinformatics 8(1), 48 (2007)
[8] Liu, Y.A., Stoller, S.D., Teitelbaum, T.: Static caching for incremental computation. ACM Transactions on Programming Languages and Systems 20(3), 546–585 (1998)
[9] Gunda, P.K., Ravindranath, L., Thekkath, C.A., Yu, Y., Zhuang, L.: Nectar: automatic management of data and computation in datacenters. In: Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, pp. 1–8 (2010)
[10] Bhatotia, P., Wieder, A., Rodrigues, R., Acar, U.A., Pasquini, R.: Incoop: MapReduce for Incremental Computations. In: Proceedings of the 2nd ACM Symposium on Cloud Computing, p. 7 (2011)
[11] Peng, D., Dabek, F.: Large-scale Incremental Processing Using Distributed Transactions and Notifications. In: Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, vol. 2006, pp. 1–15 (2010)
[12] Popa, L., Budiu, M., Yu, Y., Isard, M.: DryadInc: reusing work in large-scale computations. In: Proceedings of the 2009 Conference on Hot Topics in Cloud Computing, p. 21 (June 2009)
[13] Logothetis, D., Olston, C., Reed, B., Webb, K.C., Yocum, K.: Stateful bulk processing for incremental analytics. In: Proceedings of the 1st ACM Symposium on Cloud Computing, SoCC 2010, p. 51 (2010)
[14] Bu, Y., Howe, B., Balazinska, M., Ernst, M.D.: The HaLoop approach to large-scale iterative data analysis. The VLDB Journal 21(2), 169–190 (2012)
[15] Turcu, G., Nestorov, S., Foster, I.: Efficient Incremental Maintenance of Derived Relations and BLAST Computations in Bioinformatics Data Warehouses. In: Song, I.-Y., Eder, J., Nguyen, T.M. (eds.) DaWaK 2008. LNCS, vol. 5182, pp. 135–145. Springer, Heidelberg (2008)
[16] Isard, M., Budiu, M., Yu, Y., Birrell, A., Fetterly, D.: Dryad: distributed data-parallel programs from sequential building blocks. ACM SIGOPS Operating Systems Review 41(3), 59 (2007)

[17] Dean, J., Ghemawat, S.: MapReduce: a flexible data processing tool. Communications of the ACM 53(1), 72 (2010)
[18] Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The Hadoop Distributed File System. In: 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies, MSST, vol. (5), pp. 1–10 (2010)
[19] Apache, "Apache HBase" (2012), http://hbase.apache.org/ (accessed: April 24, 2012)
[20] Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. Journal of Molecular Biology 215(3), 403–410 (1990)
[21] Höhl, M., Kurtz, S., Ohlebusch, E.: Efficient multiple genome alignment. Bioinformatics 18(Suppl. 1), S312–S320 (2002)
[22] Finn, R.D., Clements, J., Eddy, S.R.: HMMER web server: interactive sequence similarity searching. Nucleic Acids Research 39(Web Server issue), W29–W37 (2011)
[23] Finn, R.D., Mistry, J., Tate, J., Coggill, P., Heger, A., Pollington, J.E., Gavin, O.L., Gunasekaran, P., Ceric, G., Forslund, K., Holm, L., Sonnhammer, E.L.L., Eddy, S.R., Bateman, A.: The Pfam protein families database. Nucleic Acids Research 38(Database issue), D211–D222 (2010)
[24] Oracle Grid Engine, http://www.oracle.com/us/products/tools/oracle-grid-engine-075549.html (accessed: May 02, 2012)
[25] Bhaya, D., Grossman, A.R., Steunou, A.-S., Khuri, N., Cohan, F.M., Hamamura, N., Melendrez, M.C., Bateson, M.M., Ward, D.M., Heidelberg, J.F.: Population level functional diversity in a microbial community revealed by comparative genomic and metagenomic analyses. The ISME Journal 1(8), 703–713 (2007)
[26] Douglis, F., Iyengar, A.: Application-specific Delta-encoding via Resemblance Detection. In: Proceedings of the USENIX Annual Technical Conference, pp. 113–126 (2003)
[27] Grune, D.: Concurrent Versions System, A Method for Independent Cooperation, Working paper. IR 113, Vrije Universiteit (1986)
[28] Ceri, S., Widom, J.: Deriving Production Rules for Incremental View Maintenance. In: Proceedings of the 17th International Conference on Very Large Data Bases, pp. 577–589 (September 1991)
[29] Goecks, J., Nekrutenko, A., Taylor, J.: Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome Biology 11(8), R86 (2010)
[30] Gentleman, R.C., Carey, V.J., Bates, D.M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J., Hornik, K., Hothorn, T., Huber, W., Iacus, S., Irizarry, R., Leisch, F., Li, C., Maechler, M., Rossini, A.J., Sawitzki, G., Smith, C., Smyth, G., Tierney, L., Yang, J.Y.H., Zhang, J.: Bioconductor: open software development for computational biology and bioinformatics. Genome Biology 5(10), R80 (2004)
[31] Tanenbaum, D.M., Goll, J., Murphy, S., Kumar, P., Zafar, N., Thiagarajan, M., Madupu, R., Davidsen, T., Kagan, L., Kravitz, S., Rusch, D.B., Yooseph, S.: The JCVI standard operating procedure for annotating prokaryotic metagenomic shotgun sequencing data. Standards in Genomic Sciences 2(2), 229–237 (2010)
[32] Wong, A.K., Park, C.Y., Greene, C.S., Bongo, L.A., Guan, Y., Troyanskaya, O.G.: IMP: a multi-species functional genomics portal for integration, visualization and prediction of protein functions and networks. Nucleic Acids Research 40(Web Server issue), 1–7 (2012)