

This is an Open Access document downloaded from ORCA, Cardiff University's institutional repository: <https://orca.cardiff.ac.uk/id/eprint/110601/>

This is the author's version of a work that was submitted to / accepted for publication.

Citation for final published version:

Biscarini, Filippo , Cozzi, P. and Orozco Ter Wengel, Pablo 2018. Lessons learnt on the analysis of large sequence data in animal genomics. *Animal Blood Groups and Biochemical Genetics* 49 (3) , pp. 147-158. 10.1111/age.12655

Publishers page: <http://dx.doi.org/10.1111/age.12655>

Please note:

Changes made as a result of publishing processes such as copy-editing, formatting and page numbers may not be reflected in this version. For the definitive version of this publication, please refer to the published source. You are advised to consult the publisher's version if you wish to cite this paper.

This version is being made available in accordance with publisher policies. See <http://orca.cf.ac.uk/policies.html> for usage policies. Copyright and moral rights for publications made available in ORCA are retained by the copyright holders.



1
2
3
4 **1 Lessons learnt on the analysis of large sequence data in animal**
5 **2 genomics**

6
7 3 Filippo Biscarini^{1,4¶*}, Paolo Cozzi^{1,3}, and Pablo Orozco-ter Wengel^{2¶}
8
9 4

10 5 ¹ CNR-IBBA, Milan, Italy

11 6 ² School of Biosciences, Cardiff University, Museum Avenue, CF10 3AX Cardiff, UK

12 7 ³ PTP Science Park, Department of Bioinformatics and Biostatistics - Via Einstein, 26900 Lodi, Italy

13 8 ⁴ School of Medicine, Cardiff University, Heath Park, CF14 4XN Cardiff, UK
14
15
16
17 9

18 10 ¶ these authors contributed equally to this work
19
20
21 11

22 **12 Running head:**

23 13 Analysis of large animal-genomics data
24
25 14

26
27 15 *corresponding author: Filippo Biscarini

28 16 Via Bassini 15, 20133 Milan (Italy)

29 17 +39 340 7499754

30 18 filippo.biscarini@ibba.cnr.it
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

Review

19 Summary

20 The 'omics revolution has made a large amount of sequence data available to researchers and the
21 industry. This has had a profound impact in the field of bioinformatics, stimulating
22 unprecedented advancements in this discipline. Mostly, this is usually looked at from the
23 perspective of human 'omics, in particular human genomics. Plant and animal genomics,
24 however, have also been deeply influenced by next-generation sequencing (NGS) technologies,
25 with several genomics applications now popular among researcher and the breeding industry.

26 Genomics tends to generate huge amounts of data: genomic sequence data account for an
27 increasing proportion of Big Data in biological sciences, thanks largely to decreasing sequencing
28 costs and large-scale sequencing and resequencing projects.

29 The analysis of big data poses a challenge to scientists: data gathering currently takes place at a
30 faster pace than data processing and analysis, and the associated computational burden is
31 increasingly taxing, making even simple manipulation, visualization and transferring of data a
32 cumbersome operation. The time taken up by the processing and analysing of huge data sets
33 leaves therefore little time for data quality assessment and critical interpretation. Additionally,
34 when analysing lots of data something is likely to go awry: the software (pipeline, procedure)
35 may crash or stop, and it can be very frustrating to track the error.

36 We hereby review the most relevant issues related to tackling these challenges and problems,
37 from the perspective of animal genomics, and provide researchers with a framework of steps
38 needed when processing large genomic data sets.

39 **KEYWORDS:** big data, genomics, data analysis, next-generation sequencing, animal genetics,
40 'omics, computational biology

42 INTRODUCTION

43 Big data: these two words have become buzzwords in diverse disciplines. They refer -broadly
44 speaking- to the large quantity of data made available through the extraordinary technological
45 improvements in the automated collection of information (Lohr, 2012). Big data have brought
46 about a whole new epistemology, leading to the emergence of a fourth paradigm in science (Hey
47 et al. 2009, Bell, 2009; Kitchin, 2014), that is, after theoretical, experimental and simulation
48 science, it is now the era of data-driven science. This revolution is impacting several fields of

1
2
3
4 49 science, including bioinformatics (Schuster, 2008; Pop and Salzberg, 2008): e.g. the European
5 50 Bioinformatics Institute (EBI) stores over 60 petabytes (60×10^{15} bytes) of data, of which over 2
6
7 51 petabytes are genomic data (Marx, 2013); the Sequence Read Archive (SRA) at the National
8
9 52 Centre for Biotechnology Information (NCBI) contains more than 3.6 petabases of data (4 bases
10
11 53 ≈ 1 byte). Table 1 gives examples of large ‘omics data.

12 54 Genomics is no longer an emerging field but an established one, which is projected to be among
13
14 55 the domains of science and technology that will generate the largest amounts of data by 2025
15
16 56 (Stephens et al. 2015), largely as a consequence of falling sequencing costs (Figure 1). Animal
17
18 57 genomics accounts for an increasing proportion of this amount, thanks also to large-scale
19
20 58 sequencing and resequencing projects such as the 1000 bull genomes project
21
22 59 (<http://www.1000bullgenomes.com/>), or the EU’s FP7 Nextgen project (<http://nextgen.epfl.ch/>)
23
24 60 among others. Genomic selection 2.0 is potentially another source of large amounts of sequence
25
26 61 data in livestock (Hickey, 2013). The challenge represented by the analysis of big data in animal
27
28 62 genetics has been already recognized by the scientific community (e.g. Cole et al., 2011;
29
30 63 Tempelman, 2016; Perez-Enciso, 2017): data gathering has currently a faster pace than data
31
32 64 processing and analysing; the associated computational burden is increasingly taxing, making
33
34 65 even simple manipulation, visualization and transferring of data a cumbersome operation; the
35
36 66 time taken up by the processing and analysing of huge data sets leaves little time for its critical
37
38 67 interpretation; when analysing lots of data, something is likely to go awry, the software, pipeline
39
40 68 or procedure may crash, or stop, and it can be very frustrating to track the error.

41 69 Here we review the most relevant issues related to the analysis of large sequence data in animal
42
43 70 genomics. Additionally, we propose some useful guidelines to tackle these challenges and
44
45 71 problems, and provide researchers with a framework of steps needed to face the processing of
46
47 72 large sequence experiments. These indications were motivated by research work with large
48
49 73 sequence data from livestock genomics experiments; the framework however, applies equally
50
51 74 well to non-livestock animal, plant and human genomics (and, more generally, to the analysis of
52
53 75 big “omics” data). For the sake of illustration, we will refer all-along to a standard mammalian
54
55 76 genome organized in chromosomes, and a setting in which several animals (individuals) are
56
57 77 sampled. Before starting off through this review, we kindly remind the reader of a basic
58
59 78 principle: always conceive effective algorithms and write efficient scripts for your data analysis!
60

79 PRELIMINARY CHECKS AND PLANNING

80 The internet is a very large resource providing links to publications, software download sites,
81 databases and others. However, navigating this forest of options can be difficult and
82 discouraging, resulting in researchers opting for developing tools that enable them answering the
83 questions of immediate pertinence to their work. Usually, the development of such tools requires
84 the knowledge of programming skills (e.g. C++, Java, Python, R), which still today are not part of
85 the standard toolkit of life science researchers (Ditty et al. 2010; Mangui et al., 2017).
86 Developing programming skills is very valuable in terms of i) widening the range of questions that
87 can be tackled by removing the dependency on available software, ii) the applicability of
88 programming skills beyond the immediate area of research, iii) reproducibility of research results,
89 and iv) transferable skills. However, a lack of acquaintance with the available online resources
90 can result in the inevitable re-invention of the wheel.

91 As pointed out already by Osborne et al. (2014), the first question that needs addressing is
92 whether your “question of interest” has already been asked and, especially, answered. Online
93 databases can help solving this issue by providing access to the literature (e.g. Pubmed, Scopus
94 or the Web of Science, Google Scholar), data (e.g. Genbank, Ensembl), and software (e.g.
95 Sourceforge - <http://sourceforge.net/> - and Github - <https://github.com/>). Secondly, what are the
96 resources available to answer the question of interest? A plethora of online resources for
97 genomics already exists, e.g. repositories of gene annotations, SNP (single nucleotide
98 polymorphism) and other variants, as well as cross species comparisons for genomic regions of
99 interest, such as Ensembl (www.ensembl.org), or the UCSC Genome Browser
100 (<https://genome.ucsc.edu/>). Many of these online resources also host up-to-date genome
101 reference sequences and annotations that can be used to compare the data produced by
102 researchers for quality purposes. Third, researchers “are not alone” and are not likely the first to
103 face a particular problem. Beyond these resources, several online portals open the possibility for
104 both experienced and inexperienced researchers to exchange knowledge in the form of question-
105 and-answer forums. SEQanswers (<http://seqanswers.com/>) and Biostars
106 (<https://www.biostars.org/>) are community driven forums of users focused on the discussion of
107 next-generation genomics related issues ranging from technology development to bioinformatics
108 support, and biological data analysis. ResearchGate (www.researchgate.net) hosts a large
109 community of researchers from diverse disciplines to archive, disseminate and discuss scientific

1
2
3 110 publications, ask and answer questions, propose and comment research projects and ideas.
4
5 111 Lastly, but not of least importance, Stack Overflow (<http://stackoverflow.com>) and Stack
6
7 112 Exchange (<http://stackexchange.com/>) are similar users portals, but which exclusively focus on
8
9 113 statistics, programming and computing related issues, with extensive archives on discussions on
10
11 114 both general and specific issues, covering most of the standard computing languages used in life
12
13 115 sciences (e.g. Python, Java, R). Additionally, traditional peer reviewed articles offer further
14
15 116 guidelines on software, data analysis and best practices, e.g. Nicolazzi et al. (2015) provided a
16
17 117 review of currently available software solutions for researchers working in this field, and tools to
18
19 118 streamline the analysis of animal sequence data are constantly being released (e.g. the Zanardi
20
21 119 suite, Marras et al. 2016; Consesa et al 2016). Table 2 summarizes some of the publicly available
22
23 120 resources.
24
25 121 Large sequence data not only comprise the millions of reads (i.e. sequences) from next
26
27 122 generation sequencing platforms, but other data types too, like large scale genotyping data (e.g.
28
29 123 high density SNP arrays with hundreds of thousands of genotypes for thousands of individuals,
30
31 124 such as in genomic selection programmes: e.g. Van Raden et al 2011; Meuwissen et al., 2016).
32
33 125 The data deluge unleashed by “data-driven” biology can easily become overwhelming (Hawkins
34
35 126 et al. 2010; Berger et al. 2013). This problem arises from two main issues related to handling this
36
37 127 type of data. The first one is the sheer size of the data, e.g. the amount of space required to store
38
39 128 the data, work with it (temporary storage) and archiving it to guarantee its availability in the
40
41 129 future. To give an idea, the complete genome of a single bovine is about 20-40 GB in size, in
42
43 130 terms of (compressed) raw sequence data. Researchers need to assert the size of the data that is
44
45 131 expected they will receive from an experiment, and accordingly purchase the hard-disk space
46
47 132 necessary to maintain it, ensuring there is enough working memory (RAM) to handle the data,
48
49 133 plentiful temporary space where intermediate files of multiple analyses can be stored.
50
51 134 Additionally, the data should be backed up regularly, and ideally it should be available to all
52
53 135 users at all time, e.g. via a server with a mirrored system that can be accessed online via secure
54
55 136 shell or other protocol. While many researchers can purchase space/time in a local server
56
57 137 clusters, others have to opt for online alternatives (e.g. cloud-based computing). Whatever the
58
59 138 choice is, researchers need to carefully consider the additional budget necessary for such venture
60
139 as the price per Tb of space is still expensive despite of the continuous fall of the price per byte
140 and personal computers and laptops do not tend to be powerful enough.

1
2
3 141 The second issue deals with a change in paradigm of handling the data. Until not so long ago
4 142 researchers were used to scrupulously look at each piece of data, back up all intermediate steps
5 143 of data analysis, transferring files between storage locations using flash drives or even hard
6 144 drives. However, typical dataset sizes in this era are easily hundreds of Giga bytes (Gb) large, if
7 145 not Tera bytes (Tb) or more (Schadt et al. 2010). Consequently, a new paradigm must be defined
8 146 where data can be i) efficiently summarised in order to identify approaches to trim it (e.g.
9 147 remove data of lower quality and thus less reliability), ii) avoid unnecessary backing up
10 148 intermediate analysis steps that are not crucial, as these can rapidly increase the total data size by
11 149 orders of magnitude, iii) avoiding unnecessary transfer of data between locations, as data can
12 150 take days or hours to transfer using internet protocols, and iv) carefully document the steps taken
13 151 at all stages of data analysis (i.e. write down an analysis pipeline) for reproducibility purposes. In
14 152 other words, be pre-emptive and estimate data size and its associated costs, and be tidy by
15 153 keeping track of all analyses applied with master scripts and copies of the software used to
16 154 handle data. For instance, the National Institutes of Health (NIH) is developing the Big Data to
17 155 Knowledge initiative (BD2K), that aims at managing large dataset in biomedicine, with elements
18 156 such as data handling and standards, informatics training and software sharing (Marx 2013).
19 157 Without these considerations researchers won't have enough space or RAM for analyses, and
20 158 very importantly, researchers won't be able to reproduce results contributing to the endless list of
21 159 unreproducible published data (Nekrutenko & Taylor 2012).
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38

39 161 **COMPUTING INFRASTRUCTURE AND BASIC REQUIREMENTS**

40 162 The advent of large genomics datasets brought about computational challenges which relate to
41 163 the available computing infrastructure. *A de novo* genome assembly requires approximately 1 Gb
42 164 of RAM for every 1 Mbps of genome, which for the bovine genome (~2.7 Gbps) would translate
43 165 to at least 3 TB of available RAM. Traditionally, larger problems were addressed by scaling-up
44 166 i.e. resorting to supercomputers with several processing units and large RAM capabilities (e.g. a
45 167 petaflop supercomputer for protein 3D-folding, Allen et al. 2001). This solution can be very fast
46 168 for medium scale problems, but it requires highly specialized software which tends to be very
47 169 expensive. Additionally, with ever increasing size of the data, this approach would eventually hit
48 170 a wall.
49
50
51
52
53
54
55
56
57
58
59
60

1
2
3 172 Scaling-out to using a network of machines is an appealing alternative. One option are high
4
5 173 performance computer clusters, typically constituted by a number of good quality computing
6
7 174 machines accessible through a local connection like an organization's intranet. An example is the
8
9 175 bioinformatics computing facility at PTP Science Park (www.ptp.it), with over 700 cores and 3.5
10
11 176 TB of memory. Computer clusters are generally high performing and comprise homogeneous
12
13 177 machines, which make it easier to distribute programming over the network. Downsides are the
14
15 178 expensive maintenance and the frequent underutilization: the need for very large computations in
16
17 179 any given organization is typically not continuous, but "bursty" in nature.

18 180 Computer clouds are an alternative option for distributed computing, which may circumvent
19
20 181 some such limitations. Cloud-based infrastructure services build on commodity hardware,
21
22 182 individually cheap, which is assembled into very large networks capable of scaling to massive
23
24 183 computation problems. Commercial services on a pay-per-use basis are attractive since they
25
26 184 permit to avoid investing in infrastructure and maintenance, and limit costs to the actual
27
28 185 calculations that are needed. Examples of such services are Amazon Web Services, HPCloud,
29
30 186 Google Compute Engine, Windows Azure: this market is changing rapidly, and is finding
31
32 187 applications also in genomics (O'Driscoll et al. 2013). Major challenges in cloud computing are
33
34 188 usually represented by network communication and by the additional software complexity
35
36 189 generated by dealing with heterogeneous hardware. This can be handled through frameworks for
37
38 190 distributed computing like Apache Spark (Meng et al., 2015), implemented in platforms such as
39
40 191 DataBricks (<https://databricks.com/>).

41 192 Distributed computing is certainly the way to go for animal genomics, be it private computer
42
43 193 clusters or commercial public cloud services. A pre-requisite is generally to work on a
44
45 194 Unix/Linux environment, although virtualization technology allows access also to Windows
46
47 195 users (Krampis et al. 2012).

48 196

49 197 **DATA STORAGE: DATABASE & CO.**

50 198 The amount of data generated by genomics is huge, and projected to be enormous: Stephens et
51
52 199 al. (2015) determined that over 100 PB of storage are currently used by the 20 largest sequencing
53
54 200 institutions, and estimated that as many as 40 EB (exabytes - 10^{18}) of storage capacity may be
55
56 201 needed by 2025. These requirements may be partially alleviated by data compression (Loh et al.

1
2
3 202 2012) or through techniques like “delta encoding” (Christley et al. 2009), by which only variants
4
5 203 are stored instead of complete genome sequences, at least for some individuals.

6
7 204 High-density genotyping and sequence data are often distributed as ASCII or binary files. Such
8
9 205 files need however to be parsed each time you need to access even a subset of the data, thereby
10
11 206 making the analysis quite cumbersome. While the availability of data files in standard formats is
12
13 207 usually an excellent option (e.g. VCF or BAM files have become a standard in genomics), these
14
15 208 files may be enormous making data handling cumbersome. An alternative are relational
16
17 209 databases, which offer more efficient ways of storing, accessing, extracting and analysing data in
18
19 210 a neater and safer manner. Data in a relational database are represented in tables linked through
20
21 211 unique record IDs, and are processed with SQL (structured query language), a programming
22
23 212 language specifically designed to handle data and their relations. Building a full relational
24
25 213 database (e.g. mySql) is an ideal choice for long-term storage and maintenance of data. However,
26
27 214 such databases may be complex and time- and resources-consuming, as they rely on client/server
28
29 215 applications, and most of the times the server-side component need to reside on a dedicated
30
31 216 infrastructure accessible over a network to guarantee scalability and availability. However, for
32
33 217 smaller projects, simpler solutions like sqLite exist (<https://www.sqlite.org>). SQLite allows
34
35 218 making use of ordinary files to store data and their relations using a transactional model, instead
36
37 219 of building a client/server database. Such files are portable across platforms and besides storing
38
39 220 data, they also encode high-level functionalities (e.g. “Application File Format”, like MS Excel,
40
41 221 Epub or Pdf files). However, this flexibility does not come without a cost: for instance, when
42
43 222 multiple applications or users need to read/write data at the same time (concurrency), or
44
45 223 increasing network operations is desirable (e.g. to generate and record results), or scaling-up has
46
47 224 to be dealt with, SQLite would not be sufficiently performing, and a full server/client approach
48
49 225 has to be considered instead.

50
51 226 Relational databases, both with a database server or in the no-frills sqLite version, are very
52
53 227 powerful tools that need the tables describing the data to be adequately indexed in order to make
54
55 228 efficient use of them. On one hand, without an index, if a specific row is queried the relational
56
57 229 database management (RDBM) system performs a sequential scan row by row in the table to
58
59 230 check whether its name attributes match our query conditions; the speed of such sequential
60
231 search is proportional to the number of rows in the table, i.e. it is $O(N)$ implying that the number
232 of operations required is the number of rows (N) in the table. However if the database is indexed

1
2
3 233 instead, the scanning speed is $O(\log(N))$ (for the default B-tree index type; Owens, 2006),
4
5 234 because only the index needs to be accessed by RDBM. An index is a specialized data structure
6
7 235 that stores the values for one or more columns in the database tables in a highly optimized way.
8
9 236 Additionally, indexing is even more relevant when joining tables, as that enables matching rows
10
11 237 on each table that have the same key, instead of having to sequentially scan each pair of tables
12
13 238 using a total of $O(N*M)$ operations (where N and M are the numbers of rows in each table). On
14
15 239 the other hand, indexes are data structures that take up more space than default attributes (i.e.
16
17 240 table columns), and that need to be maintained by the RDBM when records are modified.
18
19 241 Therefore, indexing too many table columns would *i)* be a waste of resources and *ii)* cause an
20
21 242 overall performance degradation. Consequently, identifying the right descriptors to be used in
22
23 243 indexes is crucial, and requires taking into account the cardinality of the data and anticipating the
24
25 244 most common and suitable queries of the database. For example, when querying sample
26
27 245 genotypes on a chromosomal sequence it would make no sense to index records on the sample
28
29 246 sex attribute (male/female), given its low cardinality; instead, the position of a polymorphism
30
31 247 along the genome would make a good index, allowing accessing a reduced set of rows upon
32
33 248 query.
34
35 249 Recently, innovative database architectures are emerging, such as graph databases, which hold
36
37 250 the promise of better modelling highly interconnected data like for instance computer networks.
38
39 251 Storage and querying such data in graph databases are expected to be faster and, in general, more
40
41 252 efficient (Angles and Gutierrez, 2008). Interconnected data in animal genetics may be illustrated
42
43 253 by genealogies (animals as nodes and relationships as connections), phenotypic records (traits as
44
45 254 nodes and trait-animal connections as trait values) and SNP genotypes (SNP loci as nodes and
46
47 255 SNP genotypes for individual animals as connections; see Biscarini et al., 2013b, for an
48
49 256 example).

257

258 **DATA ANALYSIS**

259 The analysis of genomic data may be very diverse, depending on the objective: this may go from
260
261 260 de novo assembly of a genome, to sequence alignments and variant calling; or may be the
262
263 261 downstream statistical analysis of genomic data, such as phylogenetic studies, genome-wide
264
265 262 association studies or genomic predictions for phenotypes of interest in animal breeding (e.g. de
266
267 263 los Campos et al. 2013). For large problems involving vast sequence data for a large number of

1
2
3 264 individuals (e.g. hundreds of thousands of genotyped animals like the US Holstein cattle
4 population), scalability is certainly an issue, and a distributed computation setting on a computer
5 265 cloud or cluster is needed. Frameworks to run the analysis over a network of machines are used
6 266 to first distribute the computations to where the data reside (Map operation) and then aggregate
7 267 results at the end (Reduce operation). Google MapReduce is one such solution to process big
8 268 data (Taylor 2010), which can be effectively coupled with machine learning algorithms for the
9 269 analysis of large datasets (e.g. Gillick et al. 2006), by resorting for instance to linear algebra
10 270 techniques like inner and outer products between distributed matrix rows and columns, or to
11 271 feature-encoding techniques like one-hot encoding or feature hashing. Machine learning is
12 272 becoming increasingly popular in genomics (e.g. Szymczak et al 2009) and in animal breeding
13 273 (e.g. Gonzalez-Recio & Forni 2011). A popular combination is given by the scripting language
14 274 Python within the Apache Spark framework for distributed computing (Meng et al. 2016).
15 275 Another recent and productive line of research is to develop “streaming” or “online” algorithms
16 276 that can analyze data on the fly without the need of storing it all in memory. Two examples are
17 277 the Sailfish (Patro et al. 2014) and Kallisto (Bray et al. 2016) quantification algorithms for reads
18 278 from RNA sequencing experiments, that are orders of magnitude faster than standard approaches
19 279 while presenting similar or superior accuracy. Such approaches are currently applied to ‘omics
20 280 technologies other than genomics, but it can be envisaged that similar ideas may soon find
21 281 application also for the analysis of large genomic datasets.
22 282 Open-source projects like Galaxy (<https://galaxyproject.org/>) and Jupyter (<http://jupyter.org>)
23 283 offer sophisticated platforms for data analysis which ease entry barriers for comparatively less
24 284 programming-savvy life-science researchers (Grüning et al., 2017).
25 285 Big data are not only large in size but also tend to be heterogeneous in nature: in genomics, one
26 286 may think of different sources (SNP-arrays, RAD-sequencing/Genotyping-by-sequencing,
27 287 whole-genome sequences), different genome assembly or array design and density, gene
28 288 annotations data, and so on (Perez-Enciso, 2017). Heterogeneous data pose challenges for data
29 289 integration and for imputation of missing values, and may harbour a certain amount of noise
30 290 (errors) which should be taken into account when analysing the data (Pompanon et al., 2009;
31 291 Biscarini et al., 2016; Biffani et al., 2017).
32 292
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

293 **WRITING CODE AND RUNNING THE ANALYSIS**

294 The increasing availability of multiple-core computers and computing clusters with several
295 processing units (CPUs), has prompted the use of parallel computing, where large problems can
296 sometimes be divided into smaller ones that are distributed over hundreds of CPUs and solved
297 concurrently ("in parallel") improving execution times. The analysis of sequence data often
298 present embarrassing parallel problems: e.g. genome sequences can be analysed per
299 chromosome, or alignments can be performed on a per sample (and per chromosome) basis (see
300 for instance Sikorska et al., 2013). Embarrassing parallel problems are “embarrassingly” easy to
301 run in parallel, e.g. the user just needs to split the job into sub-jobs and run them independently
302 on different cores/CPUs/machines. In such cases, the computation time is a direct function of the
303 processing resources (n. of machines, processing units such as in Beowulf clusters).
304 Parallelization may though be less straightforward when sub-processes are not thoroughly
305 independent and some degree of communication between them is needed to achieve the final
306 solution. When such communication is minimal, we talk of “coarse-grain” parallelization: an
307 example is algebraic matrix inversion frequently used in genetics and genomics (e.g. Biscarini et
308 al., 2013a). Sometimes though, sub-processes need to communicate extensively by sharing
309 memory, coordinating I/O, or reciprocally update intermediate values. Such fine-grain
310 parallelization problems are more difficult to implement and run in parallel, and require the
311 design of clever algorithms. Examples of fine-grain parallelization with sequence data are the
312 GPU-Blast implementation of the Blast alignment algorithm (Vouzis and Sahinidis, 2011), and
313 the determination of progressive alignments topology in the clustalW algorithm (Li KB 2003).
314 Interpreted scripting languages have many useful features that facilitate the execution of complex
315 tasks. For instance, R (R Core Team, 2013) can implement complex statistical models; or, high-
316 level scripting languages like Python (Van Rossum & Drake, 1995) allow to execute complex
317 tasks with just a few lines of easy-to-read code. Compiled languages like C/C++ or Fortran, on
318 the other side, achieve higher computing performances and a more powerful memory
319 management, because they translate directly to the native code of the specific machine. The
320 latter, however, comes at the expense of easy implementation, since compiled languages
321 typically use low level functions and very simple data structures that force users to write
322 extensive code even for relatively simple tasks. Hybrid solutions between compiled and
323 interpreted languages that improve computational performances with no need of sacrificing the

1
2
3 324 user-friendly syntax of scripting languages exist. Examples include Cython (Behnel et al., 2011),
4 325 SWIG (Beazley et al., 1998), the Rcpp R library (Eddelbuettel & François, 2011), that offer
5 326 frameworks where users can identify and implement in a compiled language only the bottlenecks
6 327 of their algorithms, while keep writing everything else in an interpreted user-friendlier language.
7 328 Such hybrid schemes provide therefore a compromise between performance and complexity.
8 329 Based on our experience, embedding Cython blocks in a script allowed processing 0.5 Gb of
9 330 sequence data in 50.380 seconds compared to 207.266 seconds with the same algorithm solely
10 331 implemented in Python (*ceteris paribus*).
11 332 Modular programming refers to the organization of the code in subunits which act more or less
12 333 independently (Maynard, 1972). Organising the code in modules or functions (or classes, in the
13 334 object-oriented paradigm) is especially useful for complex programmes or pipelines that
14 335 comprise several tasks, entail a considerable running-time, or run extensively in parallel.
15 336 Modularity allows for the code to be recycled -functions, modules or classes are typically used
16 337 repeatedly- and portable across platforms or projects (no need of re-writing everything from
17 338 scratch each time), and is a key component of programming efficiency. Besides, modular code is
18 339 easier to debug, since you can conveniently go through the program/pipeline “piece by piece”,
19 340 and allows to track even problems independent from your code, like machine or cluster
20 341 breakdowns, electric network failures etc ...: you would be able to resume the work from where
21 342 the problem occurred and relaunch only what is really needed, instead of everything from start.
22 343 This makes your pipeline more robust to system crashes, and reduces the risk of losing data. A
23 344 well known example of a modular pipeline of analysis for sequence data is the Ensembl pipeline
24 345 for the annotation of genomic sequence (Potter et al., 2004). To recap, make your code modular
25 346 and you’ll have an array of advantages, at the expense of only little extra planning effort!
26 347 Once you have made your code/pipeline modular, you need to make sure it is reproducible. This
27 348 can be achieved by organizing it into e.g. R packages or Python modules. Or it can be organized
28 349 into a reproducible pipeline making use of a data/analysis serialization format like the XML
29 350 mark-up language, the INI format or YAML. This latter, YAML (recursive acronym: Yaml Ain’t
30 351 Mark-up Language), has the advantage of being human-readable and of having an easy syntax
31 352 suitable for all programming languages (Ben-Kiki et al., 2005). YAML helps dealing with big
32 353 data projects with several parameters and jobs to be launched independently. It is useful to
33 354 handle the serial steps of a pipeline, but is particularly suited for “embarrassing parallel”
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

1
2
3 355 problems, where besides running several consecutive steps, these are to be repeated over a large
4
5 356 number of samples. A modular pipeline plus YAML serialization format is a powerful
6
7 357 combination for the analysis of large sequence data. YAML is usually organised in two files, one
8
9 358 with the serial steps of the analysis, the other with the samples over which the analysis should be
10
11 359 run in parallel (see Box 1 for an illustration). YAML files are written as hash tables/associative
12
13 360 arrays, i.e. in the form of key-value pairs. YAML syntax is overly simple: the most important
14
15 361 rules to remember are indentation, a few keywords (e.g. resources, steps, samples) and
16
17 362 placeholders (i.e. <variable_name>). In order for the analysis to be run, YAML files need to be
18
19 363 interpreted by ad hoc programmes/scripts, like for instance the PipEngine launcher developed in
20
21 364 Ruby (Strozzi & Bonnal, 2017).
22
23 365

Box 2. How YAML works in practice

For bioinformatics tasks, typically the YAML data analysis serialization format comprises two files (.yaml): 1) “configuration file” listing resources (paths to input data and output directories) and samples to run the analysis in parallel; 2) “analysis file” describing the serial steps of the analysis and related resources (programmes, scripts). YAML files are written in the form of hash tables/associative arrays: ‘key’: value. Below an illustration for the SNP calling and missing genotype imputation over 100 samples.

```
37 #-----  
38 # configuration.yaml  
39 #-----  
40 resources:  
41     output: /output/directory/  
42     data: /path/to/data  
43  
44 samples:  
45     'sample1': sample1_name  
46     'sample2': sample2_name  
47     .....  
48     'sample100': sample100_name  
49  
50  
51  
52  
53 #-----  
54 # analysis.yaml  
55 #-----  
56 resources:  
57     snp-calling_program: /path/to/snp-calling_program
```

```

1
2
3
4       imputation_program: /path/to/imputation_program
5
6 steps:
7     snp-calling:
8         desc: call snps from each sample sequence
9         run:  <snp-calling_program> --input <sample> -o called_snp.<sample>
10        cpu: 4
11
12     imputation:
13         desc: impute missing genotypes at SNP loci
14         run:  <imputation_program> --input called_snp.<sample> --output
15        imputed_snp.<sample>
16        cpu: 4
17
18
19
20
21
22
23
24
25
26
27
28

```

In this simple example, the steps of the analysis are organised with a description of the step, the actual code to be run in each step, and the number of CPU to be used. The analysis can then be run through and ad hoc interpreter (see main text) using a command line similar to the following:

```

29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
>> pipengine run --pipeline analysis.yml --samples-file configuration.yml --name
imputation --steps imputation

```

366

367 Processing data loaded onto the (volatile/RAM) memory is much faster compared to the heavy
368 workload of repeated I/O operations involved in reading stored data and writing them back out
369 on the disk (exactly how faster depends on disk and memory architecture: e.g. SSD, HDD,
370 DDR3). When analysing relatively small datasets, this is usually not a problem, even on a
371 laptop/client PC: all the data can be placed in the memory and analysed efficiently from there.
372 With large sequence data this is often not possible, not even if large RAM capacities are
373 available as in computing clusters or high-performance servers. This is especially true when not
374 just a single “large” job has to be executed, but several parallel jobs are to be run simultaneously
375 and have to compete for memory resources: if several “large” jobs are launched in parallel, the
376 memory would soon be full! In such cases, CPU-intensive rather than memory intensive
377 computing strategies should be adopted: the software would thus need to be designed so to resort
378 as much as possible to I/O operations in order to reduce the memory burden. Data can be read in
379 the memory record by record, or in chunks, and then processed by the CPU. In such a setting,
380 there is a trade-off between memory usage and CPU-time: memory efficiency is gained at the
381 expense of increased computation time (repeated I/O operations). An illustration from sequence
382 data is for instance reading FASTA files: these are usually quite big files, and loading them into

1
2
3 383 memory would easily exhaust memory resources. It makes therefore sense to read such files
4
5 384 sequentially, which won't use much system memory. In some circumstances, though, repeated
6
7 385 access to (part of) the file is needed, like in most matrix operations: then the approach of reading
8
9 386 the whole file into memory makes the algorithm much easier to write, at the cost of some system
10
11 387 memory.

12 388

14 389 **PUBLISHING RESULTS, DATA AND CODE**

15 390 In the previous sections we attempted to emphasise that researchers working on large datasets
16
17 391 usually encounter problems that are very similar, and which in many cases others also have
18
19 392 encountered and frequently solved. It is possible to gain access to that communal knowledge by
20
21 393 querying the literature, public databases, open forums and discussion groups. In the same way, it
22
23 394 impends on researchers to make their knowledge publicly available as members of the “scientific
24
25 395 community” (Budd et al. 2015). For that purpose it is important to identify the public databases
26
27 396 where raw data used for research can be stored. Such approach serves two purposes. On one
28
29 397 hand prevents researchers from having to come up with the funds necessary to secure data
30
31 398 archiving and its availability in the future (i.e. public databases are free). On the other hand, by
32
33 399 using public databases researchers make sure that their work contributes to the continuous
34
35 400 growth of the scientific community. Depending on the type of data, several public repositories
36
37 401 are available, e.g. DRYAD (<http://datadryad.org/>), Zenodo (<https://zenodo.org/>), the Short Read
38
39 402 Archive (NCBI, <http://www.ncbi.nlm.nih.gov/sra>), the European Nucleotide Archive (EBI,
40
41 403 <http://www.ebi.ac.uk/ena>).

40 404 While publishing the data used for analyses and the metadata associated to it is a very important
41
42 405 step, publishing the analyses pipelines (i.e. the collections of bioinformatics scripts used) is
43
44 406 crucial, and regrettably, still rarely done (Ince et al. 2012). Several public repositories exist that
45
46 407 enable publishing scripts used for data analysis, e.g. Google Code (<https://code.google.com/>),
47
48 408 Sourceforge (<http://sourceforge.net/>), Github (<https://github.com/>) or GitLab
49
50 409 (<https://about.gitlab.com/>). The users community expects to find in this type of repositories
51
52 410 scripts that can be directly used by others; however, researchers frequently write code that was
53
54 411 intended for their own use or for a specific task (a.k.a. quick and dirty script). While publishing
55
56 412 those scripts is still important, programming skills are no longer a desired skill only for
57
58 413 mathematicians, physicists and engineers: researchers in the biological sciences, too, need to

1
2
3 414 build a basic informatics knowledge (Dudley & Butte 2009, Hawkins et al. 2010) that enables
4
5 415 them writing scripts that are accessible to others (i.e. that can be read and modified).
6
7 416 Finally, although researchers plan their work so to maximize the likelihood of obtaining
8
9 417 significant and relevant results, it is of fundamental importance to also publish lack of or
10
11 418 negative results, so to minimize issues with publication and reporting bias (Dwan et al., 2008):
12
13 419 on-line archives like Bioarxiv (<http://www.biorxiv.org/>) offer a convenient way to make all
14
15 420 research results readily available to the scientific community and the broader public.
16

16 421

17 422 **CONCLUSIONS**

19 423 The advancement in ‘omics technologies has guided the development of a data-driven approach
20
21 424 to biological sciences. This change has marked the need for researchers in the biological sciences
22
23 425 to change their approach to experiment design, data handling and storage, and time allocation for
24
25 426 wet-lab vs. dry-lab (computer based) work, as well as it has resulted in the growing need for
26
27 427 those researchers to at least have a basic understanding of computing language (e.g. to at least be
28
29 428 able to look at files) and information technology (e.g. to understand about file transfer protocols
30
31 429 between servers). Fast computers and vast storage capabilities are giving us plenty of
32
33 430 possibilities to handle large-scale data (besides contributing to produce big-data, in a sort of
34
35 431 virtuous/vicious cycle). However, such resources, though ample, are not infinite, and the design
36
37 432 of good computation strategies is still fundamental to handle today’s large quantities of data. In
38
39 433 this review of common practices we described principles that we feel are very important and that
40
41 434 biological researchers embarking in the field of genomics need to be aware of. Importantly, while
42
43 435 our views derive from our experience working with livestock genomics, our comments are
44
45 436 equally applicable to research on crops, wildlife fauna and flora, humans, and microbial ‘omics
46
47 437 technologies. Lastly, these comments reflect the lessons we learnt during our own experience,
48
49 438 and it is very important to note that no matter how well you plan experiments and how strictly
50
51 439 you follow our guidelines , when analysing large data involving multiple comparisons, methods,
52
53 440 models, samples etc, you must be patient and willing to learn at each step, as you cannot expect
54
55 441 that everything will run smoothly without problems!

53 442

54 443 **Acknowledgements**

1
2
3 444 The authors acknowledge the contribution of the NEXTGEN FP7 EU and ClimGen projects
4
5 445 from which they received financial support and the opportunity of experimenting with large
6
7 446 genomic data. A special thanks/acknowledgement goes to Ian Streeter for his scientific and
8
9 447 technical support in dealing with large sequence data. FB was financed also by the Marie-Curie
10
11 448 European Reintegration Grant NEUTRADAPT.

12 449

14 **Conflicts of interests**

15 451 The authors declare that they have no conflicts of interests.

16 452

19 **REFERENCES**20 453
21 454 Allen F., Almasi G., Andreoni W., Beece D., Berne B.J., Bright A. et al. (2001) Blue Gene: a
22 455 vision for protein science using a petaflop supercomputer. *IBM systems journal* **40**(2): 310-27.23 456 Angles R, Gutierrez C. (2008). Survey of graph database models. *ACM Computing Surveys*
24 457 (*CSUR*), **40**(1): 1.25 458 Beazley DM (1998). Interfacing C/C++ and Python with SWIG. In *7th International Python*
26 459 *Conference, SWIG Tutorial*.27 460 Behnel, S., Bradshaw, R., Citro, C., Dalcin, L., Seljebotn, D. S., & Smith, K. (2011). Cython:
28 461 The best of both worlds. *Computing in Science & Engineering*, *13*(2), 31-39.29 462 Bell G, Hey T, Szalay A (2009) Beyond the data deluge. *Science* **323**: 1297-1298.30 463 Ben-Kiki O, Evans C, Ingerson B (2005). YAML Ain't Markup Language (YAML™) Version
31 464 1.1. *yaml.org, Tech. Rep*.32 465 Berger, B., Peng, J., & Singh, M. (2013). Computational solutions for omics data. *Nature*
33 466 *reviews. Genetics*, *14*(5), 333.34 467 Biffani, S., Pausch, H., Schwarzenbacher, H., & Biscarini, F. (2017). The effect of mislabeled
35 468 phenotypic status on the identification of mutation-carriers from SNP genotypes in dairy cattle.
36 469 *BMC research notes*, *10*(1), 230.37 470 Biscarini F, Picciolini M, Stella A, Iamartino D, Strozzi F (2013a) A graph database to store and
38 471 manage phenotypic, pedigree and genotypic data of livestock. Book of abstracts No. 19 of the
39 472 64th Annual Meeting of the European Federation of Animal Science (Nantes, France).40 473 Biscarini F, Pedretti A, Ober U, Erbe M, Jorjani H, Nicolazzi E, Picciolini M (2013b) Use of
41 474 molecular markers to estimate genomic relationships and marker effects: computation strategies
42 475 in R. In: The R User Conference, user! 2013 July 10-12 2013 University of Castilla-La Mancha,
43 476 Albacete, Spain (Vol. 10, No. 30, p. 13).44 477 Biscarini, F., Nazzicari, N., Broccanello, C., Stevanato, P., & Marini, S. (2016). "Noisy beets":
45 478 impact of phenotyping errors on genomic predictions for binary traits in *Beta vulgaris*. *Plant*
46 479 *methods*, *12*(1), 36.

- 1
2
3 480 Bray NL, Pimentel H, Melsted P, Pachter L (2016) Near-optimal probabilistic RNA-seq
4 481 quantification. *Nature biotechnology* **34**(5): 525-527.
- 6 482 Christley S, Lu Y, Li C, Xie X (2009) Human genomes as email attachments. *Bioinformatics*
7 483 **25**(2): 274-275.
- 9 484 Cole J, Newman S, Foertter F, Aguilar I, Coffey M (2012) Breeding and genetics symposium:
10 485 Really big data: Processing and analysis of very large data sets. *Journal of Animal Science* **90**:
11 486 723–733.
- 13 487 Conesa A, Madrigal P, Tarazona S, Gomez-Cabrero D, Cervera A, McPherson A, Szcześniak
14 488 MW, Gaffney DJ, Elo LL, Zhang X, Mortazavi A (2016) A survey of best practices for RNA-seq
15 489 data analysis. *Genome biology* **17**(1): 1.
- 17 490 de los Campos G, Hickey JM, Pong-Wong R, Daetwyler HD, Calus MP (2013) Whole-genome
18 491 regression and prediction methods applied to plant and animal breeding. *Genetics* **193**(2): 327-
19 492 345.
- 21 493 Ditty J, Kvaal C, Goodner B, et al. (2010) Incorporating genomics and bioinformatics across the
22 494 Life Sciences curriculum. *PLoS Biology* **8**:e1000448
- 24 495 Dwan, K., Altman, D. G., Arnaiz, J. A., Bloom, J., Chan, A. W., Cronin, E., ... & Ghersi, D.
25 496 (2008). Systematic review of the empirical evidence of study publication bias and outcome
26 497 reporting bias. *PloS one*, **3**(8), e3081.
- 28 498 Eddelbuettel, D., François, R., Allaire, J., Chambers, J., Bates, D., & Ushey, K. (2011). Rcpp:
29 499 Seamless R and C++ integration. *Journal of Statistical Software*, **40**(8), 1-18.
- 31 500 Gillick D, Faria A, DeNero J (2006) Map-reduce: Distributed computing for machine learning.
32 501 Berkley 18.
- 34 502 González-Recio O, Forni S (2011) Genome-wide prediction of discrete traits using Bayesian
35 503 regressions and machine learning. *Genetics Selection Evolution* **43**(1):1.
- 37 504 Grüning, B. A., Rasche, E., Rebolledo-Jaramillo, B., Eberhard, C., Houwaart, T., Chilton, J., ... &
38 505 Nekrutenko, A. (2017). Jupyter and Galaxy: Easing entry barriers into complex data analyses for
39 506 biomedical researchers. *PLOS Computational Biology*, **13**(5), e1005425.
- 41 507 Hey T, Tansley S, Tolle K (2009) *The Fourth Paradigm: Data-Intensive Scientific Discovery*,
42 508 Redmond, WA: Microsoft Research.
- 44 509 Hawkins RD, Hon GC, Ren B. (2010) Next-generation genomics: an integrative approach.
45 510 *Nature Reviews Genetics* **11**(7): 476-486.
- 47 511 Hickey JM. (2013) Sequencing millions of animals for genomic selection 2.0. *Journal of Animal*
48 512 *Breeding & Genetics* **130**(5): 331-332.
- 49 513 Kitchin R (2014) Big Data, new epistemologies and paradigm shifts. *Big Data & Society* **1**: 1-12.
- 51 514 Krampis K, Booth T, Chapman B, Tiwari B, Bica M, Field D, Nelson KE. (2012) Cloud
52 515 BioLinux: pre-configured and on-demand bioinformatics computing for the genomics
53 516 community. *BMC Bioinformatics* **13**(1): 1.
- 55 517 Li KB (2003) "ClustalW-MPI: ClustalW analysis using distributed and parallel computing",
56 518 *Bioinformatics* **19**(12): 1585-1586.

- 1
2
3 519 Loh PR, Baym M, Berger B. (2012) Compressive genomics. *Nature biotechnology* **30**(7): 627-
4 520 630.
5
6 521 Lohr, S. (2012). The age of big data. *New York Times*, 11.
7
8 522 Marx, V. (2013). Biology: The big challenges of big data. *Nature* **498**(7453): 255-260.
9
10 523 Mangul, S., Martin, L. S., Hoffmann, A., Pellegrini, M., & Eskin, E. (2017). Addressing the
11 524 digital divide in contemporary biology: Lessons from teaching UNIX. *Trends in Biotechnology*.
12 525 Maynard, J. (1972) "Modular programming." CRC Press
13
14 526 Meng, Xiangrui, et al. (2016) "Mllib: Machine learning in apache spark." *Journal of Machine*
15 527 *Learning Research* **17**(34): 1-7.
16
17 528 Meuwissen, T., Hayes, B., & Goddard, M. (2016). Genomic selection: A paradigm shift in animal
18 529 breeding. *Animal frontiers*, 6(1), 6-14.
19
20 530 "NEXT GENERATION METHODS TO PRESERVE FARM ANIMAL BIODIVERSITY:
21 531 NEXTGEN" FP7-EU Project [<http://nextgen.epfl.ch/>]
22
23 532 Nicolazzi EL, Biffani S, Biscarini F, Orozco ter Wengel P, Caprera A, Nazzicari N, Stella A.
24 533 (2015) Software solutions for the livestock genomics SNP array revolution. *Animal genetics*
25 534 **46**(4): 343-353.
26
27 535 O'Driscoll A, Daugelaite J, Sleator RD. (2013) 'Big data', Hadoop and cloud computing in
28 536 genomics. *Journal of Biomedical Informatics* **46**(5): 774-781.
29
30 537 Osborne JM, Bernabeu MO, Bruna M, Calderhead B, Cooper J, Dalchau N et al. (2014). Ten
31 538 simple rules for effective computational research. *PLoS computational biology* **10**(3): e1003506.
32
33 539 Owens M (2006) "The Definitive Guide to SQLite", Chapter 4, pp 155-158, isbn: 978-1-59059-
34 540 673-9
35
36 541 Patro R, Mount SM, Kingsford C (2014) Sailfish enables alignment-free isoform quantification
37 542 from RNA-seq reads using lightweight algorithms. *Nature biotechnology* **32**(5): 462-464.
38
39 543 Pérez-Enciso, M. (2017). Animal Breeding learning from machine learning. *Journal of Animal*
40 544 *Breeding and Genetics*, 134(2), 85-86.
41
42 545 Pompanon, F., Bonin, A., Bellemain, E., & Taberlet, P. (2005). Genotyping errors: causes,
43 546 consequences and solutions. *Nature reviews. Genetics*, 6(11), 847.
44
45 547 Pop M, Salzberg SL (2008) Bioinformatics challenges of new sequencing technology. *Trends in*
46 548 *Genetics* **24**(3): 142-149.
47
48 549 Potter SC, Clarke L, Curwen V, Keenan S, Mongin E, Searle SM et al (2004). The Ensembl
49 550 analysis pipeline. *Genome research* **14**(5): 934-941.
50
51 551 R Core Team (2013). R: A language and environment for statistical computing. R Foundation for
52 552 Statistical Computing, Vienna, Austria. [<http://www.R-project.org/>]
53
54 553 Schuster SC (2008). Next-generation sequencing transforms today's biology. *Nature methods*
55 554 **5**(1): 16-18.
55
56 555 Stephens ZD, Lee SY, Faghri F, Campbell RH, Zhai C, Efron MJ et al. (2015) Big data:
57 556 astronomical or genetical?. *PLoS Biology* **13**(7): e1002195.
58
59
60

- 1
2
3 557 Strozzi F, Bonnal RJP (2017) Pipengine: an ultra light YAML-based pipeline execution engine.
4 558 *The Journal of Open Source Software* **12**:16.
5
6 559 Sikorska K, Lesaffre E, Groenen PF, Eilers PH (2013). GWAS on your notebook: fast semi-
7 560 parallel linear and logistic regression for genome-wide association studies. *BMC Bioinformatics*
8 561 **14**(1): 166.
9
10 562 Szymczak S, Biernacka JM, Cordell HJ, González-Recio O, König IR, Zhang H, Sun YV (2009)
11 563 Machine learning in genome-wide association studies. *Genetic epidemiology* **33**(S1): S51-57.
12 564 Taylor RC. (2010) An overview of the Hadoop/MapReduce/HBase framework and its current
13 565 applications in bioinformatics. *BMC Bioinformatics* **11**(Suppl 12):S1.
14
15 566 Tempelman, R. J. "The frontier spirit and reproducible research in animal breeding." *Journal of*
16 567 *Animal Breeding and Genetics* **133**, no. 6 (2016): 441-442.
17
18 568 VanRaden PM, O'Connell JR, Wiggans GR, Weigel KA (2011) Genomic evaluations with many
19 569 more genotypes. *Genetics Selection Evolution* **43**(1):1.
20
21 570 Van Rossum G, Drake Jr FL (1995) *Python reference manual*. Amsterdam: Centrum voor
22 571 Wiskunde en Informatica.
23
24 572 Vouzis PD, Sahinidis NV (2011). GPU-BLAST: using graphics processors to accelerate protein
25 573 sequence alignment. *Bioinformatics* **27**(2): 182–188.
26
27 574
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

575 **Tables**

576

577 **Table 1:** [Examples of Big Data from 'omics technologies](#)

Examples of Big Data from 'omics technologies		
Category	Raw data	Size
Whole-genome sequences (WGS)	sequence reads	~ 5 GB for a genome ~ 3 Gbps long at ~ 10x coverage
Transcriptome Sequence Analysis (TSA)	sequence reads	several GB depending on coverage (< WGS)
Bisulphite sequencing	sequence reads	several GB (\leq TSA)
SNP array	genotypes	few kB for sample \rightarrow usually several ples \rightarrow MB/GB
5 GB: giga-bytes; 5 MB: mega-bytes; 5 kB: kilo-bytes; Gbps: giga-base-pairs.		

578

579 **Table 2:** [Publicly available resources](#)

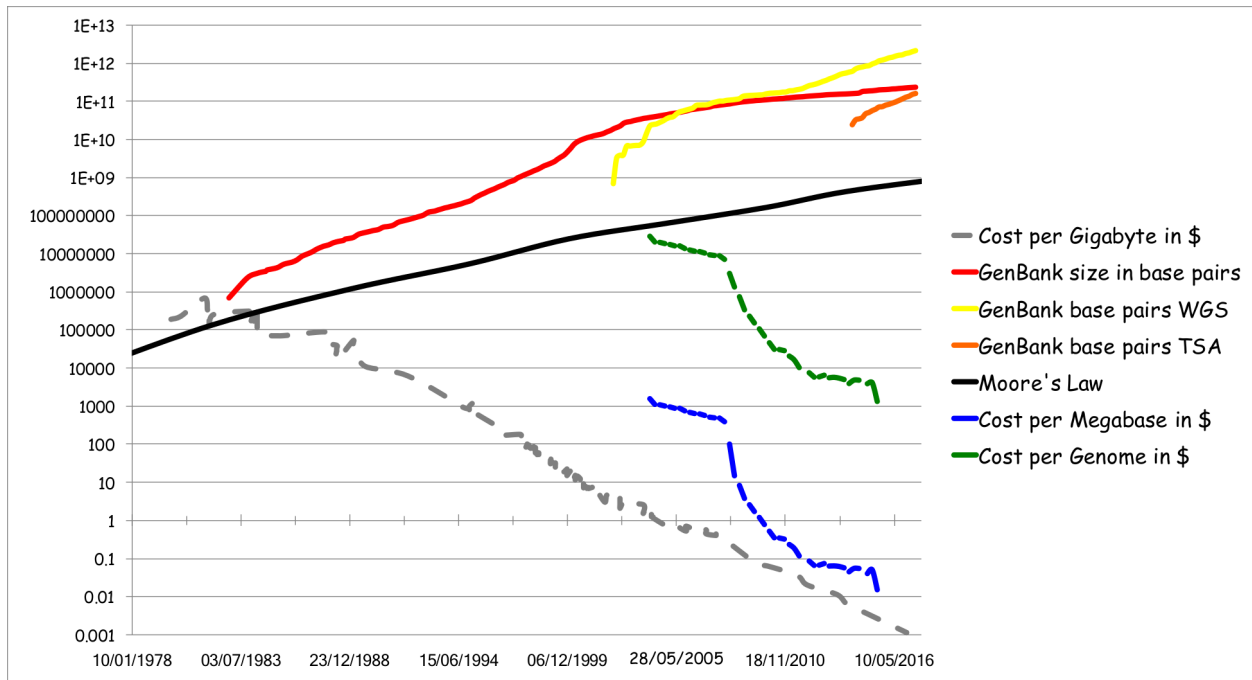
Resource	Name	access	type
Forum	SEQanswers	http://seqanswers.com/	Sequencing, Bioinformatics
	Biostars	https://www.biostars.org/	Bioinformatics, Biological Data Analysis
	Stack Overflow	http://stackoverflow.com/	Informatics
	Stack Exchange	http://stackexchange.com/	Informatics
Software	Sourceforge	http://sourceforge.net/	Repository
	Github	https://github.com/	Repository
	Google Code	https://code.google.com/	Repository
	sqlite	https://www.sqlite.org	Database software
	YAML	http://yaml.org/	Data serialization standard
Database	Pubmed	http://www.ncbi.nlm.nih.gov/pubmed	Literature
	Scopus	http://www.scopus.com/	Literature
	Genbank	http://www.ncbi.nlm.nih.gov/genbank/	Data
	Ensembl	http://www.ensembl.org/index.html	Data

	Short Read Archive	http://www.ncbi.nlm.nih.gov/sra http://www.ebi.ac.uk/ena	Data
	Dryad	http://datadryad.org/	Data
	USGC Genome Browser	https://genome.ucsc.edu/	Data
Large Scale Projects	1000 genomes	http://www.1000genomes.org/	Human genomes
	1000 bull genomes project	http://www.1000bullgenomes.com/	Cattle genomes
	NextGen Consortium	http://nextgen.epfl.ch/	Mouflon, Sheep, Bezoar, Goat, Cattle
	The 3000 rice genomes project	http://gigadb.org/dataset/20000_1	Rice
	1001genomes	http://1001genomes.org/	Arabidopsis

580
581

582 **Figures**

583



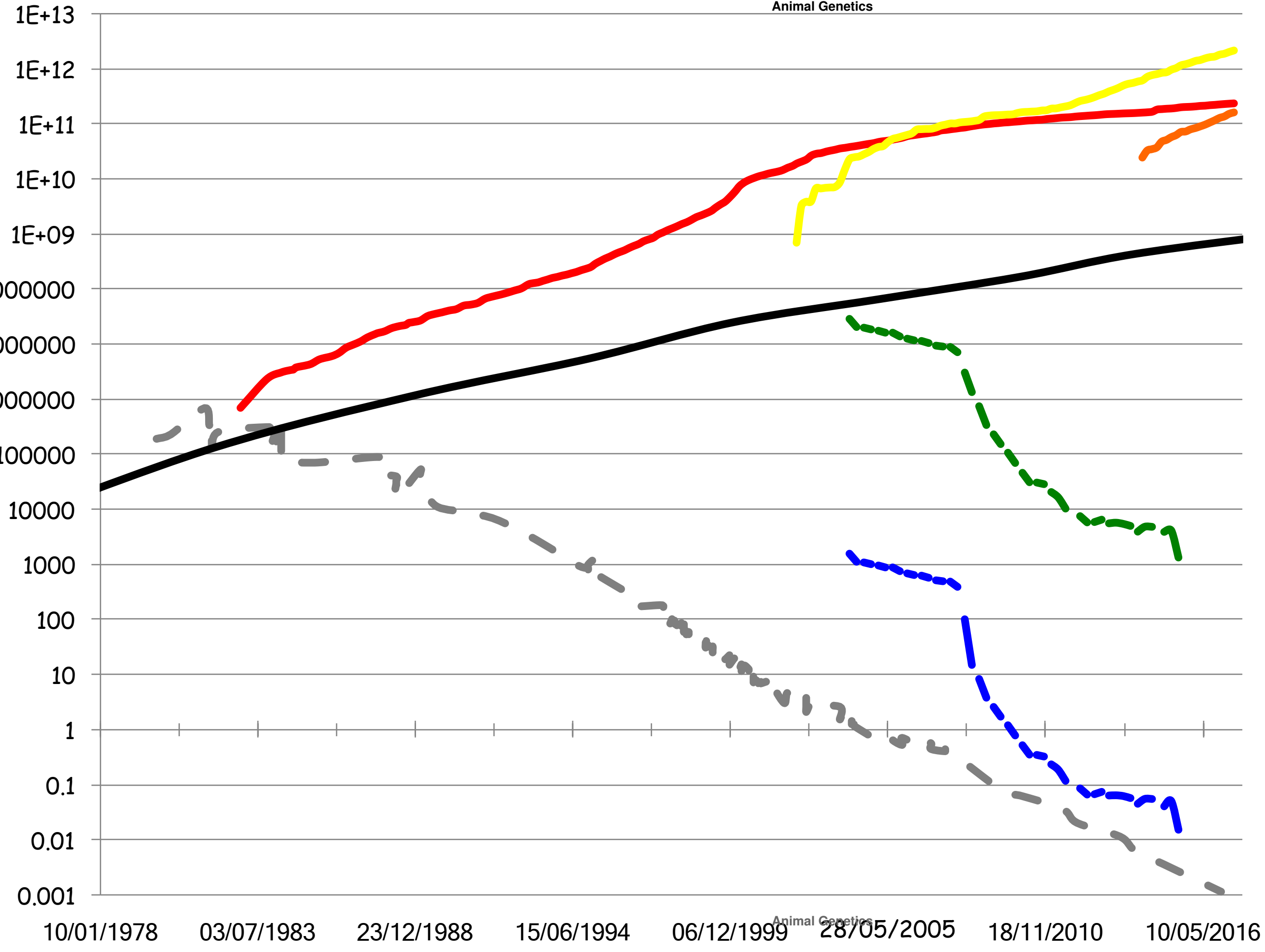
585 **Figure 1:** Trends in costs and data production over time. Cost per giga-byte (gray line), per
 586 genome (green line), per mega-base (blue line). Base-pairs from GenBank (red line), from
 587 whole-genome sequences (WGS, yellow line) and from transcriptome sequence analysis (TSA,

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

588 orange line); Moore’s law (black line). The y-axis holds for all units (dollars, base-pairs, n. of
589 transistors). WGS and TSA data are not distributed in conjunction with GenBank releases. Data
590 from <ftp://ftp.ncbi.nih.gov/genbank/gbrel.txt>, <https://www.genome.gov/sequencingcosts/>
591

For Peer Review

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60



- Cost per Gigabyte in \$
- GenBank size in base pairs
- GenBank base pairs WGS
- GenBank base pairs TSA
- Moore's Law
- Cost per Megabase in \$
- Cost per Genome in \$