



Processing and Analyzing Multiple Genomes Alignments with MafFilter

Julien Y. Dutheil

Abstract

As the number of available genome sequences from both closely related species and individuals within species increased, theoretical and methodological convergences between the fields of phylogenomics and population genomics emerged. Population genomics typically focuses on the analysis of variants, while phylogenomics heavily relies on genome alignments. However, these are playing an increasingly important role in studies at the population level. Multiple genome alignments of individuals are used when structural variation is of primary interest and when genome architecture permits to assemble *de novo* genome sequences. Here I describe MafFilter, a command-line-driven program allowing to process genome alignments in the Multiple Alignment Format (MAF). Using concrete examples based on publicly available datasets, I demonstrate how MafFilter can be used to develop efficient and reproducible pipelines with quality assurance for downstream analyses. I further show how MafFilter can be used to perform both basic and advanced population genomic analyses in order to infer the patterns of nucleotide diversity along genomes.

Key words Multiple genome alignment, Synteny, Alignment post-processing, Quality filtering, Multiple alignment format

1 Introduction: Multiple Genome Alignments

Multiple genome alignments (MGAs) record the homology relationships between related genome sequences. While conventional sequence alignments contain information about nucleotide substitutions, insertions, and deletions, MGAs encode evolutionary events occurring at a larger scale. Such events include chromosome fusion, fission, and rearrangements, which break colinearity between sequences (*akasynteny break*). Furthermore, genome sequences, as opposed to gene sequences, are generally segmented. The underlying cause of this segmentation may be biological (presence of multiple chromosomes) or technical (genome sequence could only be assembled at the contig or scaffold level).

MGAs are typically used to compare genomes in distinct species (see, for instance, the 99 vertebrate genome alignments from the UCSC Genome Browser [1]). Conversely, population genomic analyses typically focus on micro-variation events—single nucleotide polymorphisms (SNP) and short indels—and assume synteny and karyotype conservation between individual genomes. As a result, genetic variation is stored as *variant calls* with respect to a *reference* genome, often as a file in the Variant Call Format (VCF) [2] or MAP format [3]. Variant files, however, do not usually contain information about invariable positions and need to be combined with additional information for most evolutionary applications (e.g., as a list of “callable positions,” that is, positions where enough information was available to detect a SNP if any).

The Multiple Alignment Format (MAF, not to be confounded with the Mutation Annotation Format) describes the homology relationships between several genomes, as flat text files (see <https://genome.ucsc.edu/FAQ/FAQformat.html>, last accessed 29/08/18). A MAF file is a list of several alignment blocks where the constitutive sequences are in synteny (see Fig. 1). While the structure of each block is identical to traditional sequence alignments (as in the Clustal or Phylip formats), where homologous positions in each sequence are on top of each other and form an *alignment column*, sequence names follow a dedicated syntax in order to record genome coordinates. Besides, several annotation lines can be included, including, for instance, sequence quality scores. Genome alignment programs producing MAF files as output include TBA [4], Mugsy [5], ROAST <http://www.bx.psu.edu/~cathy/toast-roast.tmp/README.toast-roast.html> (last accessed 29/08/18), Last [6], and Mauve [7].

MGAs are also used in population genomic studies, either when complete individual genomes can be obtained (e.g., [8, 9]) or when pseudo-genomes can be generated [10] (see **Note 1**). Because they contain information about both variable and invariable positions, MGAs can be directly used for conducting evolutionary analyses, accounting for missing data and structural variation. This, however, comes at the cost of extended computer requirements, in particular in terms of file size. Additional alignment quality checks are also typically required, as full-genome aligners do not include post-processing steps as most variant calling pipelines do.

In this chapter, we will see how to use the `MafFilter` program to conduct population genomic analyses. In the following, we assume that the data is available as a MGA in the MAF format. Conversion to variant call formats will also be discussed.

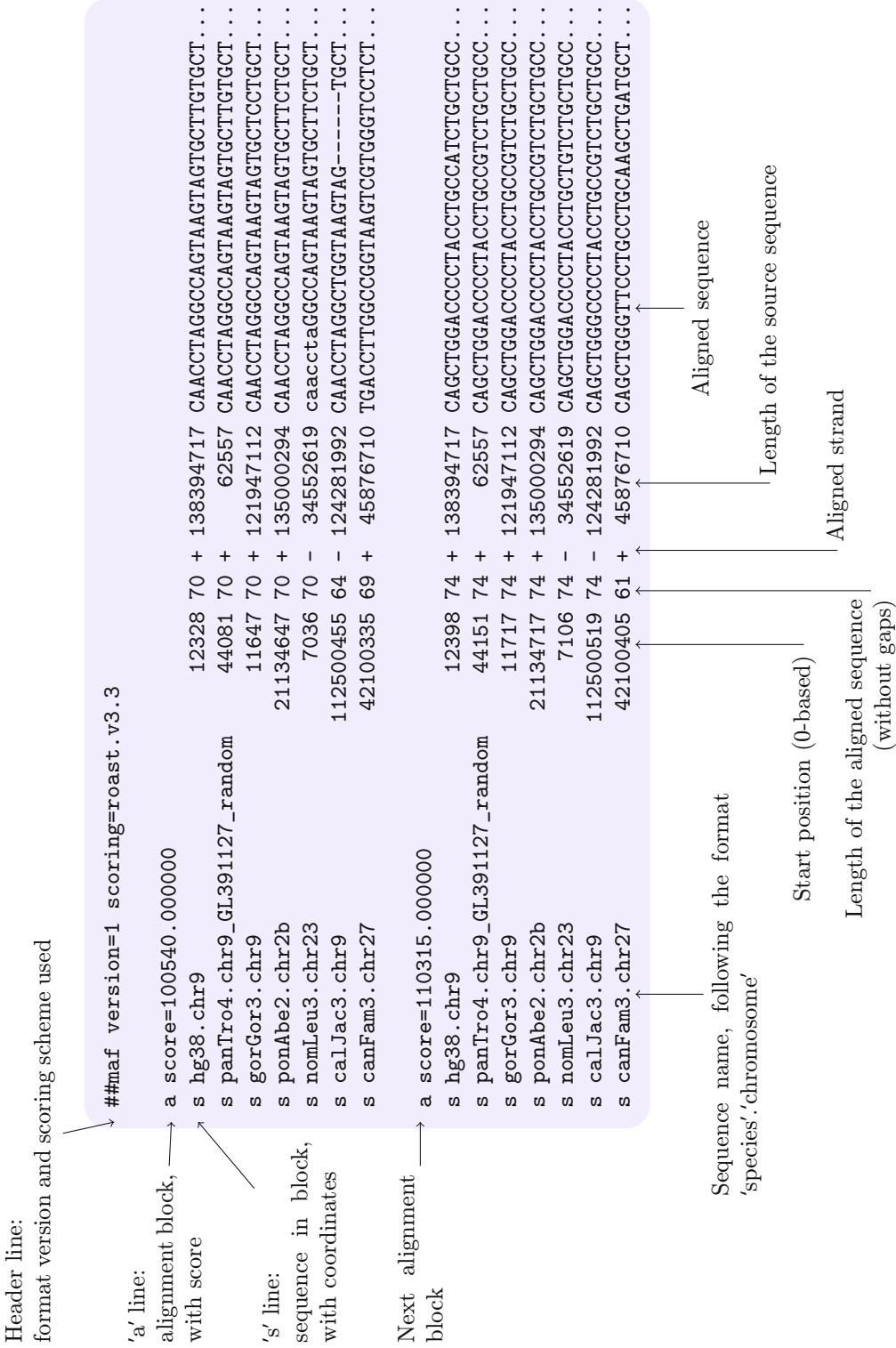


Fig. 1 Structure of a MAF file. Data source: UCSC alignment of human chromosome 9 together with 19 Mammals, among which 16 Primates <http://hgdownload.soe.ucsc.edu/goldenPath/hg38/multiz20way/>

2 General Principles on `MafFilter` Usage

As MAF files were initially used for multi-species alignments, each input genome is referred to as a *species*. In the following, a *species* can, however, also denote a particular strain or individual in a population. Similarly, the term *chromosome* will be used in a broad sense encompassing scaffolds and contigs, in case of unmapped genome assemblies (*see* Fig. 1).

2.1 Serial Processing of Alignment Blocks: Filters

As MAF files are organized into a series of syntenic blocks, `MafFilter` sequentially processes input files one block at a time by applying *filters*. A filter takes a MAF alignment block as input, conducts one or several analyses, and returns a MAF block. Depending on the type of analysis performed, the output block might be identical to the input one or a modified version. In some cases, the filter can compute additional information that can be written to an output file or stored as meta-data (*see* Table 1 for examples). Filters are combined sequentially, the output of one filter serving as input to the next one, allowing to design advanced analysis workflows.

2.2 Option Files and Command Line Arguments

The `MafFilter` program can be controlled by arguments that are passed from the command line or, more conveniently, as a script file. Arguments take the form of ‘parameter’=‘value’ statements, which can potentially be nested. Arguments can also be called within the script, allowing to define global variables. Below is a minimalist example demonstrating the syntax:

Table 1
Example types of filters supported by `MafFilter`

Filter name	Filter function	Output
<code>MafStatistic</code>	Compute statistics on a block	Unmodified input block
<code>MinBlockLength</code>	Filter blocks given alignment length	Unmodified input block if its length is larger than a given threshold, otherwise the block is discarded
<code>Subset</code>	Keep only a subset of species	A block with sequences from the specified set of species
<code>WindowSplit</code>	Split a block into smaller blocks of a given size	Multiple smaller blocks
<code>DistanceEstimation</code>	Compute an evolutionary distance matrix from all sequences in the block	Unmodified input block with a distance matrix attached as meta-data

```

1 # maffilter param=MinimalistExample.bpp DATA=chr9
2 input.file=./Primates/${DATA}.maf.gz
3 input.file.compression=gzip
4 input.format=Maf
5 output.log=${DATA}.maffilter.log
6 maf.filter=\
7   MinBlockLength(min_length=1000),\
8   Output(file=${DATA}.min1kb.maf.gz, compression=gzip)

```

Line 1 is a comment line, which will not be parsed. Bash style comments (starting with #), C style (surrounded by /* and */) and C++ style (starting with //) are recognized. The script uses a global variable named “DATA” that is set via the command line and whose value is called using the Makefile syntax \$(DATA). The script can be run using the command

```
maffilter param=MinimalistExample.bpp DATA=chr9
```

It will parse the input alignment (here human chromosome 9 aligned with 19 other Mammals, downloaded from the UCSC genome browser), keep only blocks that are at least 1 kb in length, and write the result to a new MAF file. Line 2 indicates the path to the input MAF file; line 3 specifies that the file was compressed using gzip; line 4 indicates that the file is in the MAF format. While `MafFilter` is dedicated to the analysis of MAF files, it can also take as input a Fasta file for a single species, with one sequence per chromosome. Line 5 indicates the path to a log file, where information about the analysis will be written. Line 6 shows the main argument, `maf.filter`, which contains a comma-separated list of options, one per filter. Filters will be applied in their order of specification, so that the output of filter 1 will be the input of filter 2, etc. As the line can be rather long, it is split using the “\” character. In this most simple example, there are two filters specified: `MinBlockLength`, which discards blocks below 1 kb, and `Output`, which writes the resulting alignment to a new gzip-compressed MAF file.

In the following, we will see more advanced examples of filters and how they can be combined to conduct genomic analyses.

3 MafFilter as a Data Processor

3.1 Extracting Data of Interest

A MAF alignment contains information about all genomic regions in a set of species, and some analyses can focus on a subset of such species. Besides, certain types of analyses involve only a subset of positions, such as protein-coding sites. `MafFilter` allows to process a MAF alignment and restrict it both to a subset of species and

positions. In the first case, selected species are specified as an argument of a filter, while in the second, a file describing which positions to keep is provided, as a feature file (such as a BED or GFF-like file, see <https://genome.ucsc.edu/FAQ/FAQformat.html>, last accessed 29/08/18).

The following example illustrates these aspects. The pipeline filters blocks to keep only the ones with sequences in Human, Chimpanzee, Bonobo, Gorilla, and Orangutan. Additional sequences for other species, if any, are discarded. In a second step, coding regions are extracted and written as a separate alignment file.

```

1 # maffilter param=ExtractingData.bpp DATA=chr9
2 # Note: need to create subdirectory Alignments
3 # before running this script
4
5 input.file=./Primates/${(DATA) }.maf.gz
6 input.file.compression=gzip
7 input.format=Maf
8 output.log=${(DATA) }.maffilter.log
9 SPECIES=(hg38, panPan1, panTro4, gorGor3, ponAbe2)
10 maf.filter=
11 Subset (
12     species=${(SPECIES)},
13     strict=yes, keep=no,
14     remove_duplicates=yes),
15 Merge (
16     species=${(SPECIES)},
17     dist_max=0),
18 ExtractFeature (
19     ref_species=hg38,
20     feature.file=./Primates/chr9.CDS1kb.gtf,
21     feature.format=GTF,
22     complete=yes,
23     ignore_strand=no),
24 OutputAlignments (
25     format=Clustal,
26     file=Alignments/FivePrimates%i-%c-%b-%e.aln,
27     reference=hg38)

```

The **Subset** filter (line 11) extracts blocks where certain species are aligned (given as a list, here provided as a global variable set line 9). The **strict** and **keep** arguments can be combined to obtain various behaviors: with **strict** set to “yes” and **keep** set to “no”, we only keep blocks where the five selected species are all present and discard sequences from putative additional species. The **remove_duplicates** argument further removes blocks where

any of the selected species might be present more than once (paralogous sequences). The **Merge** filter (line 15) subsequently fuses consecutive blocks in complete synteny, which might have been split apart because of a synteny break in one of the non-selected species.

The position extraction is done by the **ExtractFeature** filter (line 18), which retains regions specified in a file in the Gene Transfer Format (GTF). The GTF file contains only Coding DNA Sequences (CDS) with at least 1 kb in length. We further specify to only extract regions that are fully covered in the alignment (`complete` argument, line 23). The `ignore_strand` argument, line 24, tells whether regions on the negative strand should be reverse-complemented (“no” option) or kept as is (“yes” option).

Finally, the writing of the extracted blocks is done by the **OutputAlignments** filter (line 24). Each block is written in the Clustal alignment format [11] into a file with path `Alignments/FivePrimates%i-%c-%b-%e.aln`, where `%i` will be replaced by the index of the block. As a result, each block will be written in a separate file. If the special `%i` code is omitted, all alignments will be appended to a single output file. The additional special codes `%c`, `%b`, and `%e` can be optionally used in combination with `%i` and correspond to the coordinates of the block (chromosome, begin and end, respectively) according to one “reference” species specified by the `reference` argument. Further note that **MafFilter** cannot create directories, only files. In case the provided output path is not valid, no output will be generated.

3.2 Statistics with **MafFilter**

The effect of each data extraction step can be visualized using statistics filters. The **SequenceStatistics** filter is a powerful and generic way of computing and reporting measures for each block. It takes as input a list of statistics names and generates a table file with computed statistics as columns, and each block as a row. The table also contains the coordinates of the block according to one *reference* species.

The following pipeline is a modification of the one presented in Subheading 3.1. After each step, a **SequenceStatistics** filter is added to report the length (number of alignment columns) and size (number of sequences) of each block. This creates four files, summarized in Fig. 2.

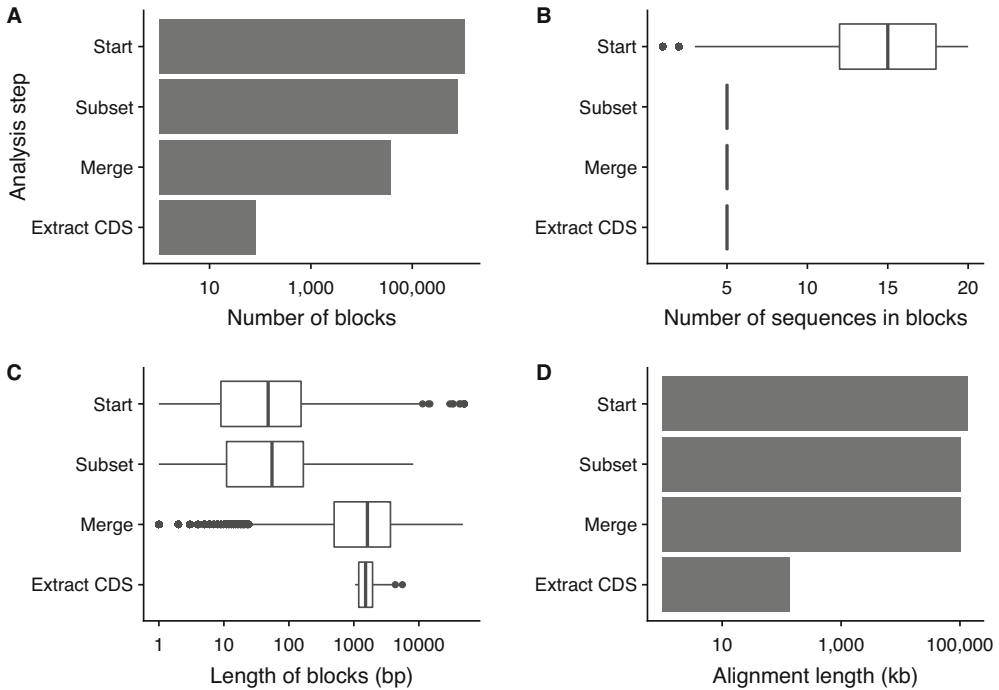


Fig. 2 Effect of data extraction filters, as measured with statistics filters. Four steps are plotted: before filtering (“Start”), after subsetting to five primate species (“Subset”), after merging synteny blocks (“Merge”) and after extracting CDS regions (“Extract CDS”). **(A)** Number of blocks after each step. **(B)** Distribution of block sizes, that is, the number of species represented in each block. **(C)** Distribution of block lengths, that is, number of alignment columns in each block. **(D)** Total alignment length, that is, the sum of all block lengths

```

1 # maffilter param=Statistics.bpp DATA=chr9
2
3 input.file=./Primates/$(DATA).maf.gz
4 input.file.compression=gzip
5 input.format=Maf
6 output.log=$(DATA).maffilter.log
7 SPECIES=(hg38, panPan1, panTro4, gorGor3, ponAbe2)
8 maf.filter=
9   SequenceStatistics(
10     statistics=(BlockLength,BlockSize),
11     ref_species=hg38,
12     file=$(DATA).statistics1.txt),
13   Subset(
14     species=$(SPECIES),
15     strict=yes, keep=no,
16     remove_duplicates=yes),
17   SequenceStatistics(
18     statistics=(BlockLength,BlockSize),
19     ref_species=hg38,
20     file=$(DATA).statistics2.txt),

```



```

21 Merge( \
22     species=$(SPECIES), \
23     dist_max=0), \
24 SequenceStatistics( \
25     statistics=(BlockLength,BlockSize), \
26     ref_species=hg38, \
27     file=$(DATA).statistics3.txt), \
28 ExtractFeature( \
29     ref_species=hg38, \
30     feature.file=./Primates/chr9.CDS1kb.gtf, \
31     feature.format=GTF, \
32     complete=yes, \
33     ignore_strand=no), \
34 SequenceStatistics( \
35     statistics=(BlockLength,BlockSize), \
36     ref_species=hg38, \
37     file=$(DATA).statistics4.txt) \

```

After filtering, 81 alignment blocks are created. This is less than the 146 entries in the GTF file, the difference being due to CDS that are (at least partially) missing or not in synteny in any of the five selected species. When only the human and chimpanzee genomes are considered, for instance, the number of complete CDS present in the alignment becomes 118.

3.3 Pre-Processing the Data for Quality Insurance

Comparative evolutionary analyses of sequences require high-quality input data, as any error at this stage is likely to propagate in the downstream analyses. Such errors may occur both at the individual sequence level (sequencing and assembly errors) and at the alignment level (wrong orthology inference, alignment errors). In some cases, we also want to discard regions (e.g., protein-coding positions) that are likely to violate the prior assumptions of a given analysis (e.g., neutral evolution).

The MAF format allows storing position-specific scores. Using **QualFilter**, it is possible to remove regions with a low score in a given set of species. The filter further allows computing the average score in a sliding window with user-specified size. Windows with an average score below a given threshold are discarded, and the corresponding block split accordingly. Similarly, **MaskFilter** can be used to clean blocks according to the proportion of masked positions in a given set of sequences. Masked regions are coded as lowercase nucleotides and are typically used to annotate low-complexity regions.

The local quality of the alignment can be assessed via the distribution of gaps in sliding windows. **AlnFilter** and **AlnFilter2** both slide windows along the alignment and discard regions with too many gaps. They differ by their scoring criteria: **AlnFilter** computes the global frequency of gap characters, while **AlnFilter2** estimates the number of indel events, independently of the length of the insertion or deletion track. **EntropyFilter** can also be used to remove highly variable regions in the alignment.

Finally, **FeatureFilter** can be used to exclude regions from the alignment. Features to exclude can be specified as an annotation

file, in GFF, GTF, or BedGraph format. When GFF or GTF annotation files are provided, it is further possible to exclude only a given subset of features.

Most filters allow writing the filtered regions in a separate file optionally. This feature enables to finely tune the filtering criteria by visually assessing which regions are kept or removed. Using the **SequenceStatistics** filter is also convenient to monitor the proportion of alignment discarded. In the following sections, concrete example analyses will demonstrate the use of these filters. Before getting there, however, we will introduce a last set of filters enabling inter-operability between analysis tools: format conversion filters.

3.4 Conversion to Other Formats

When MGAs store genomes from individuals of the same species, they can be exported as variants. This requires that a reference genome is specified, usually implying that any synteny break will be further ignored, together with parts of the alignment that do not include the chosen reference species. When exporting to variant formats, it is generally recommended to first project the alignment on the reference species, so that the variants are sorted (see, for instance, program `maf_project` in Subheading 5). `MafFilter` can export in three distinct variant formats: the widely used VCF [2] (**VcfOutput** filter), Plink ped and map files [3] (**PlinkOutput** filter), and MSMC [12] (see Chapter 7, **MsmcOutput** filter).

Synteny block can also be exported into standard alignment format with the **OutputAlignments** filter, as seen in Subheading 3.1. The **OutputAlignments** filter further accepts a `ldhat_header` argument allowing to export alignments readable by the `convert` program from the LDhat package [13].

Meta-data associated with alignment blocks can be exported using dedicated filters. The **OutputDistanceMatrices** filter exports all matrices into a file in the Phylip format. Similarly, the **OutputTrees** filter exports trees in Newick format. Both require the specification of a tag name used to attach the meta-data to each block (e.g., `MLdistance` or `BioNJ`).

4 Examples of Advanced Analyses

4.1 Example Analysis 1: Computing Nucleotide Diversity Along the Genome

This section describes the first complete analysis example. We use the publicly available *Drosophila* Population Genomics Project phase 3 (DPGP3, see Chapter 13) [10], containing 197 genomes from a single African ancestral population. We restrict our analysis to one chromosome arm (2L) and ten individuals. The corresponding dataset has been combined into a single MAF file (see online Supplementary Information). The following script first uses **AlnFilter** to process the data in 10 bp windows slid by one bp in order to remove regions with too many gaps, which discards 10% of the alignment (see Fig. 3A). This leads to many more blocks (see Fig. 3B), of shorter length (see Fig. 3C). The resulting split blocks

are then merged if not further apart than a 100 bp, using the **Merge** filter. When merged, missing regions are filled by unresolved characters (“N”). The resulting blocks are split into non-overlapping windows of 10 kb, and smaller blocks are discarded (**MinBlockLength** and **WindowSplit** filters). As a result, 32% of the original alignment is lost (*see* Fig. 3A). Two statistics are used to compute population genetics quantities: **SiteFrequencySpectrum**, which counts minor allele frequencies (*see* Chapter 1 in this volume) and **DiversityStatistics**, which computes various diversity estimators (*see* Table 2).

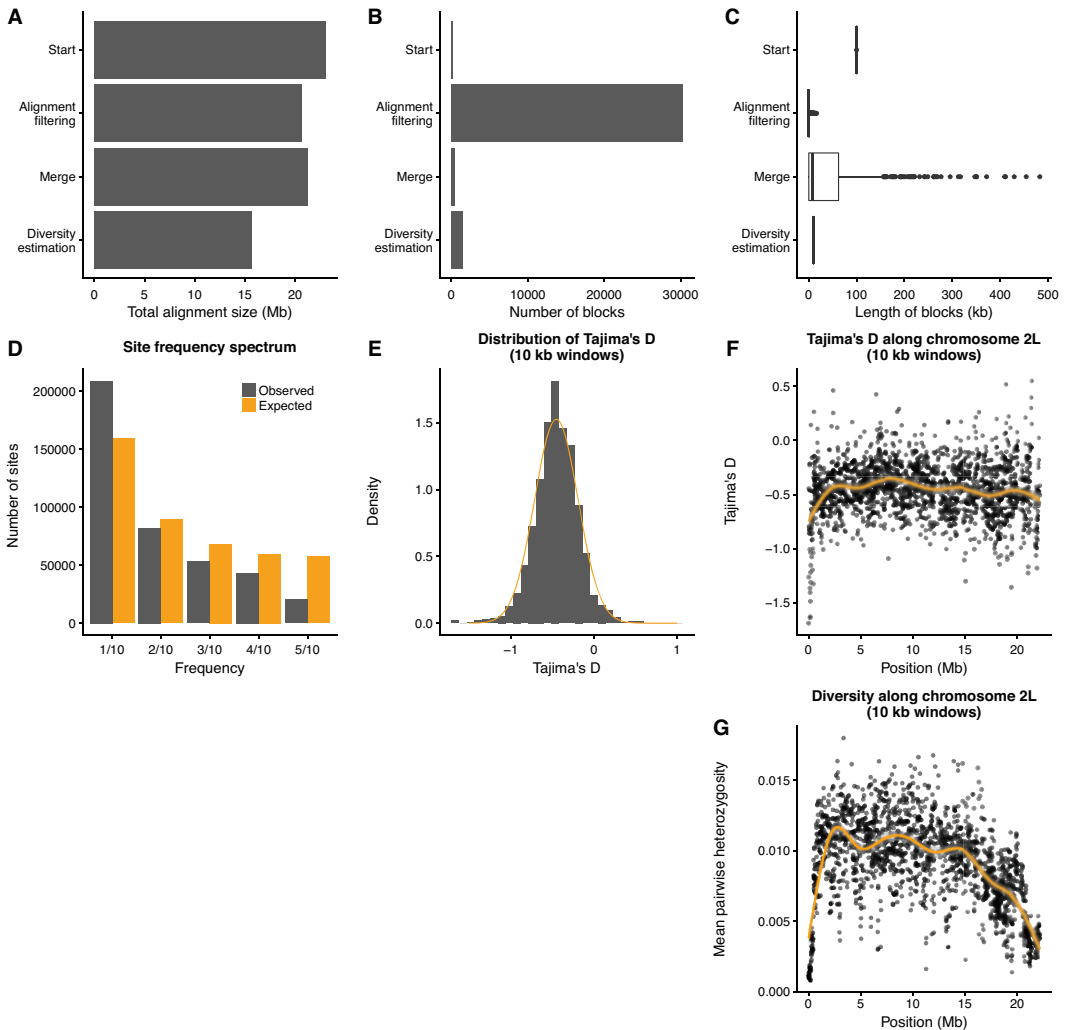


Fig. 3 Patterns of diversity in 10 individuals along *Drosophila melanogaster* chromosome 2L. First row: effect of data filtering. (A) Total alignment length. (B) Number of blocks. Second and third rows: diversity statistics. (C) Distribution of block lengths. (D) Site frequency spectrum, observed and expected under a Wright-Fisher model with identical global diversity. (E) Distribution of Tajima’s D in 10 kb windows (histogram) and normal distribution fit (curve). (F) Tajima’s D in 10 kb windows plotted along the chromosome. The smooth line was computed using a generalized additive model (GAM). (G) Average pairwise heterozygosity in 10 kb windows plotted along the chromosome. The smooth line was computed using a generalized additive model (GAM)

Table 2
Available statistics for the `SequenceStatistics` filter

Statistic name	Statistic function	Output
BlockSize	Report the number of sequences	Numerical value
BlockLength	Report the number of alignment columns	Numerical value
SequenceLength	Report the number of nucleotides for a given sequence	Numerical value
AlnScore	Report the alignment score for the block, as encoded in the input MAF file	Numerical value
BlockCounts	Report the frequencies of each nucleotide	Numerical values, one for each character state
SiteStatistics	Compute the number of sites with missing data/gaps, numbers of mono, di, tri, and quadri-allelic sites, number of parsimony-informative sites	Numerical values, one for each statistic
PairwiseDivergence	Compute the percentage of mismatches between two user-specified species	Numerical values
SiteFrequencySpectrum	Count sites based on their minor allele frequency, according to user-specified bins	Numerical values, one per bin
PolymorphismStatistics	Compare two sets of sequences and compute the number of fixed and polymorphic positions in both sets.	Numerical values, for all combination of fixed and polymorphic sites (e.g., fixed in one set and polymorphic in the other)
DiversityStatistics	Compute the number of segregating sites, Watterson's theta, Tajima's pi, and Tajima's D.	Four numerical values
ModelFit	Given a phylogenetic tree, fit a nucleotide substitution model using maximum likelihood and report the parameter estimates. A large variety of models from Jukes-Cantor to General Time Reversible are available, including rate across sites variation models.	Numerical values, one per estimated parameter

The `MafFilter` script starts by defining a few variables: dataset (line 3), list of individual sequences used (lines 4–5), reference sequence used for the output of coordinates (line 6) and size of the window for which estimators are computed (here 10 kb, line 7).

The script generates a text file with all computed statistic per 10 kb windows, together with their coordinates in the reference genome. Besides, simpler statistics files are generated at each step of the analysis to summarize the data used. The actual alignment is also output as a new MAF file for further assessment.

```

1 # maffilter param=Maffilter-Diversity.bpp
2
3 DATA=dpgp3_Chr2L_10indv
4 INDV=ZI152,ZI173,ZI190,ZI199,ZI211,ZI219,\
5     ZI253,ZI344,ZI374,ZI490
6 REF=ZI152
7 WSIZE=10000
8
9 input.file=./Drosophila/${DATA}.split.maf.gz
10 input.file.compression=gzip
11 input.format=Maf
12 output.log=${DATA}.maffilter-diversity.log
13 maf.filter=
14     SequenceStatistics (
15         statistics=(BlockLength,BlockSize),
16         ref_species=${REF},
17         file=${DATA}.statistics1.txt),
18     AlnFilter (
19         species=( ${INDV} ),
20         window.size=10,
21         window.step=1,
22         missing_as_gap=yes,
23         max.gap=0.3,
24         max.ent=-1,
25         relative=yes,
26         file=${DATA}.trash_aln.maf,
27         compression=none,
28         verbose=yes),
29     SequenceStatistics (
30         statistics=(BlockLength,BlockSize),
31         ref_species=${REF},
32         file=${DATA}.statistics2.txt),
33     Merge(species=( ${INDV} ), dist_max=100),
34     SequenceStatistics (
35         statistics=(BlockLength,BlockSize),
36         ref_species=${REF},
37         file=${DATA}.statistics3.txt),
38     Output (
39         file=${DATA}.filtered.maf.gz,
40         compression=gzip),
41     MinBlockLength(min_length=${WSIZE}),
42     WindowSplit (
43         preferred_size=${WSIZE},
44         align=center,
45         keep_small_blocks=no),
46     SequenceStatistics (
47         statistics=(BlockLength,BlockSize),

```

```

48     DiversityStatistics(, \
49     ingroup=$(INDV)), \
50     SiteFrequencySpectrum( \
51     ingroup=$(INDV), \
52     bounds=(-0.5,0.5,1.5,2.5,3.5,4.5,5.5)) \
53     ), \
54     ref_species=$(REF), \
55     file=$(DATA).diversity_statistics.txt) \

```

This example demonstrates the use of **AlnFilter**: lines 20–21 specify the size of the window and the amount by which it is slid (10 nucleotides slid by 1). Line 22 further tells the filter that missing nucleotides (“N”) should be counted as gaps. The maximal proportion of gaps allowed in the window is set to 0.3 (line 23). Absolute numbers of gaps can also be specified by changing line 25 to “no.” In this example, we do not filter according to the site variability, and the maximal entropy is set to -1 (line 24). Alternatively, windows will be discarded if they both display a number of gaps and entropy higher than the specified thresholds. Discarded regions are output to a separate MAF file (lines 26–27), for further assessment. Finding the optimal alignment filtering criteria requires to compare both the retained and rejected regions. Multiple **AlnFilter** can be combined in order to achieve the desired quality.

Diversity estimators are computed as standard statistics (*see* Subheading 3.2). **DiversityStatistics** takes only one input arguments, the list of individuals to use (line 49, in this case, all of them). **SiteFrequencySpectrum** requires, in addition, specifying boundaries for the frequencies to compute (line 52). As we have ten genomes, the possible SNPs minor frequencies are 0, 1, 2, 3, 4, and 5 out of 10. We therefore specify as boundaries $-0.5, 0.5, 1.5, 2.5, 3.5, 4.5, \text{ and } 5.5$. Note that it is possible to specify fewer boundaries to pull two or more categories. Each category generates one column in the output statistic file. Besides, positions with unresolved characters or more than two alleles are counted separately and excluded from the site frequency spectrum calculation.

The computed site frequency spectrum reveals an excess of low-frequency variants (*see* Fig. 3d), resulting in a globally negative Tajima’s D value (*see* Fig. 3e). The effect is relatively constant along the chromosome, except for the most telomeric region (*see* Fig. 3f), suggesting that this population underwent a demographic expansion. Patterns of heterozygosity, on the other hand, show a substantial reduction at the telomere, and a positive correlation with the distance to the centromere, at the right end of the alignment (Kendall’s tau = 0.28, $p\text{-value} < 2.2 \cdot 10^{-16}$, *see* Fig. 3g).

4.2 Example

Analysis 2: Inferring Phylogenetic Relationships

In this example, we infer the phylogenetic relationships of five great apes. We use the UCSC 20-way genome alignment, containing 16 Primates genomes. For the sake of computational efficiency, we restrict the analysis to chromosome 9 only. We implement the following pipeline:

1. extract the genome alignment for human, chimpanzee, bonobo, gorilla, and orangutan,
2. filter the alignment to remove ambiguously aligned regions,
3. split the resulting filtered alignment into non-overlapping 10 kb windows,
4. compute a pairwise distance matrix using maximum likelihood and estimate a BioNJ tree for each window,
5. root each tree using the orangutan sequence as an outgroup,
6. write the resulting trees to a file,
7. fit a model of sequence evolution on the human, bonobo, chimpanzee, and gorilla ingroup using maximum likelihood and output parameters to a file.

This results in the following `MafFilter` option file:

```

1 # maffilter param=MafFilter-Phylogeny.bpp
2
3 DATA=chr9
4 SPECIES=(hg38, panPan1, panTro4, gorGor3, ponAbe2)
5 WSIZE=10000
6
7 input.file=./Primates/${DATA}.maf.gz
8 input.file.compression=gzip
9 input.format=Maf
10 output.log=${DATA}.maffilter.log
11 maf.filter= \
12   Subset ( \
13     species=${SPECIES}, \
14     strict=yes, keep=no, \
15     remove_duplicates=yes), \
16   XFullGap(species=${SPECIES}, verbose=no), \
17   MinBlockLength(min_length=10), \
18   AlnFilter2(verbose=no, \
19     species=${SPECIES}, \
20     window.size=10, window.step=1, \
21     missing_as_gap=yes, \
22     max.gap=2, max.pos=2, relative=no, \
23     file=None, \
24     compression=none), \
25   Merge ( \
26     species=${SPECIES}, \
27     dist_max=100, \
28     rename_chimeric_chromosomes=yes), \

```

```

29 Output ( \
30     file=$(DATA).filtered.maf.gz, \
31     compression=gzip), \
32 MinBlockLength(min_length=$(WSIZE)), \
33 WindowSplit ( \
34     preferred_size=$(WSIZE), \
35     align=center, \
36     keep_small_blocks=no), \
37 DistanceEstimation(verbose=no, \
38     method=ml, \
39     model=K80(kappa=2), \
40     rate=Gamma(n=4, alpha=0.5), \
41     parameter_estimation=initial, \
42     max_freq_gaps=0.33, \
43     gaps_as_unresolved=yes, \
44     profiler=none, \
45     message_handler=none, \
46     extended_names=yes), \
47 DistanceBasedPhylogeny(verbose=no, \
48     method=bionj, dist_mat=MLDistance), \
49 NewOutgroup( \
50     tree_input=BioNJ, \
51     tree_output=BioNJ_rooted, \
52     outgroup=ponAbe2), \
53 OutputTrees( \
54     tree=BioNJ_rooted, \
55     file=$(DATA).trees.dnd, \
56     compression=none, \
57     strip_names=yes), \
58 DropSpecies( \
59     tree_input=BioNJ_rooted, \
60     tree_output=BioNJ_subtree, \
61     species=ponabe2), \
62 SequenceStatistics( \
63     statistics=( \
64         BlockCounts(suffix=.all), \
65         BlockCounts(species=hg38, suffix=.hs), \
66         BlockCounts(species=panPan1, suffix=.pp), \
67         BlockCounts(species=panTro4, suffix=.pt), \
68         BlockCounts(species=gorGor3, suffix=.gg), \
69         BlockCounts(species=ponAbe2, suffix=.pa), \
70         ModelFit( \
71             model=HKY85(kappa=1, theta=0.5, \
72                 theta1=0.5, theta2=0.5), \
73             rate_distribution=Gamma(n=4, alpha=0.5), \
74             root_freq=Full, \
75             tree=BioNJ_subtree, \
76             parameters_output=(HKY85.theta_1, \
77                 HKY85.theta1_1, HKY85.theta2_1, \
78                 HKY85.kappa_1, Gamma.alpha, \
79                 Full.theta1, Full.theta2, Full.theta), \
80             fixed_parameters=(), \
81             reestimate_brlen=no, \
82             max_freq_gaps=0.3, \
83             gaps_as_unresolved=yes)), \
84     ref_species=hg38, \
85     file=$(DATA).model-statistics.csv), \

```


This rather large option file starts with the selection of the species of interest, which we store as a list in the `SPECIES` variable (line 4). The `Subset` filter (lines 12–15) extracts the corresponding species for each block, excluding blocks where not all five species are present (`strict=yes`), and removing any additional species that might be present (`keep=no`). Besides, we discard any block where a species might be present more than once because of paralogy (`remove_duplicates=yes`). As a result, after this step, all blocks contain exactly five sequences, one for each species.

We then proceed with alignment filtering (starting line 16). We first remove all alignment columns containing a gap in all kept sequences, due to putative indels with more distant species, which have now been discarded. This is achieved via the `XFullGap` filter (line 16). We then slide a 10 bp window in order to exclude regions with a least two indel events, independent of their size. Only indel events involving at least two species are counted (`AlnFilter2`, with arguments `max.pos=2` and `max.gap=2`). The number of gaps is specified as a number of occurrences (`relative=no`); it can also be specified as a proportion of the number of sequences. As we are sliding 10 bp windows, we first discard alignment blocks with less than ten columns (`MinBlockLength` filter, line 17). The resulting alignment is spread into numerous, potentially small blocks. In order not to discard too much data in subsequent steps of the analysis, we perform a merging step (lines 25–28). With the specified configuration, consecutive blocks will only be merged if all input species are syntenic, that is, the sequences in the two blocks are colinear (same chromosome, same strand, same distance between the start of the new block and end of the previous one). By specifying a subset of species only, it is possible to merge according to some focus species, resulting in coordinates being lost for other species. We further consider a maximum distance of 100 bp in order to merge consecutive blocks (line 27). When two blocks are merged, so are the sequence names, which may result in excessively long names. Using the `rename_chimeric_chromosomes` argument, we tell the program to arbitrarily give new names to merged sequences, which will be called `chimgXX`, `XX` being a unique number. When merged, missing positions will be replaced by “N” characters, allowing to preserve coordinates. In effect, the combination of the `AlnFilter2` and `Merge` filters result in a masking of the discarded positions.

We analyze the resulting filtered dataset in windows of 10 kb. The focus window size is specified as a global variable (line 5) and can be changed in order to assess the impact of the window size on the results. The `WindowSplit` filter breaks each block into non-overlapping blocks of a given size (lines 33–36). Input arguments allow specifying how to cut a block when its size exceeds the specified window size: either start from the left, center on the block

while discarding start and end regions or adjust the size in order not to lose any data. Note that in the latter case, the input window size w will be the minimum size. The resulting window size can, therefore, be comprised between w and $2 \times w - 1$. When the `keep_small_blocks` option (line 36) is set to yes, and the window size is not adjusted, out-of-window alignment parts and block smaller than the specified window size will be kept as separate blocks. Otherwise, they will be discarded.

Phylogenetic reconstruction in each window is performed using a distance method (BioNJ, [14]), which requires first to estimate a pairwise distance matrix. We use a maximum likelihood method, with a K80 substitution model (line 39) and a discrete gamma distribution of rates across sites (line 40). For computational efficiency purpose, we only estimate distances and keep other parameters fixed to realistic values (transition/transversion ratio equal to 2 and gamma shape parameter equal to 0.5). We further consider gaps as unknown characters in the modeling and discard positions with more than one-third of unresolved characters (lines 42 and 43). For each windowed block, the resulting matrix is stored as meta-data, with label `MLDistance`. This distance matrix is then given as input to the `DistanceBasedPhylogeny` filter, which reconstructs a tree using the `BioNj` method (line 50) and stores it under the label `BioNJ`. Further processing includes rooting each tree using the Orangutan sequence (`NewOutgroup` filter, lines 49–52) and removal of the outgroup branch (`DropSpecies` filter, lines 58–61). Rooted trees are saved into a text file using the `Output-Trees` filter for further analysis.

The final step of the analysis consists in the estimation of substitution parameters for each window. This is done via a `SequenceStatistics` filter, and two dedicated statistics: `BlockCounts` and `ModelFit`. The `BlockCounts` statistics is rather straightforward, as it computes nucleotide frequencies in a given set of species. We use a combination of six calls to this statistic to compute averaged (line 64) as well as species-specific frequencies (lines 65–69). Input arguments include the set of species to use in the calculation, as well as suffix strings to distinguish the different output results. The `ModelFit` statistic is more complex and requires to specify a substitution model, similar to the `DistanceEstimation` filter. As the model is being fitted to the full tree using Felsenstein's dynamic algorithm [15], a more parameter-rich model can be employed (HKY85, [16]). In particular, we use a non-stationary model, allowing us to estimate the observed and equilibrium frequencies separately. Under such a model, the ancestral nucleotide frequencies are different from the equilibrium ones and are fully parameterized (line 74). In order to reduce computational time, we do not reestimate branch lengths and keep them to the values resulting from the BioNJ algorithm (line 81). Enabling branch length reestimation does not change the results significantly (see

companion material). Further parameters can be fixed to their initial or default value using the `fixed_parameters` argument (line 80). Finally, the `output_parameters` argument allows specifying which estimated parameters should be output to the result file. As the nomenclature of parameter names can be complicated, `MafFilter` outputs the list of available parameters when run. A two-step run might, therefore, be needed in order to fit the desired model.

The results of this analysis are summarized in two files: a spreadsheet file containing numerical values, one statistic per column and one 10 kb window per line (file `chr9.model-statistics.csv`), and one text file containing a list of trees, one line per window (file `chr9.trees.dnd`). R scripts are provided as companion material in order to analyze these output files. The analysis led to 883 trees. A majority rule consensus tree leads to a topology compatible with the well-established phylogeny of the species (*see* Fig. 4A) [17]. This topology is supported by a majority of windows (Fig. 4B), but four other “minor” topologies are also inferred: topology C and D are supported by 55 and 54 windows, and group human with gorilla and chimpanzee + bonobos with gorilla, respectively. Such topologies result from incomplete lineage sorting in the humans-chimpanzees-bonobos ancestral populations

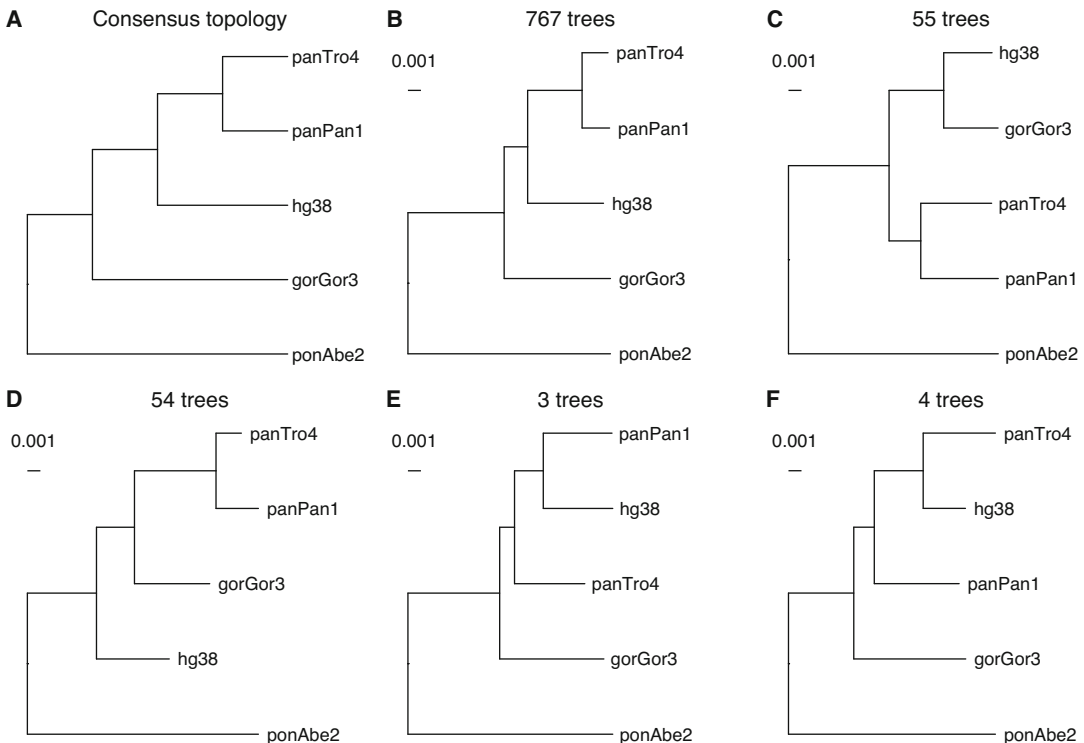


Fig. 4 Incomplete lineage sorting along chromosome 9 of the great apes. A) Consensus topology. B-F) Topologies with mean branch lengths sorted by frequency of occurrence

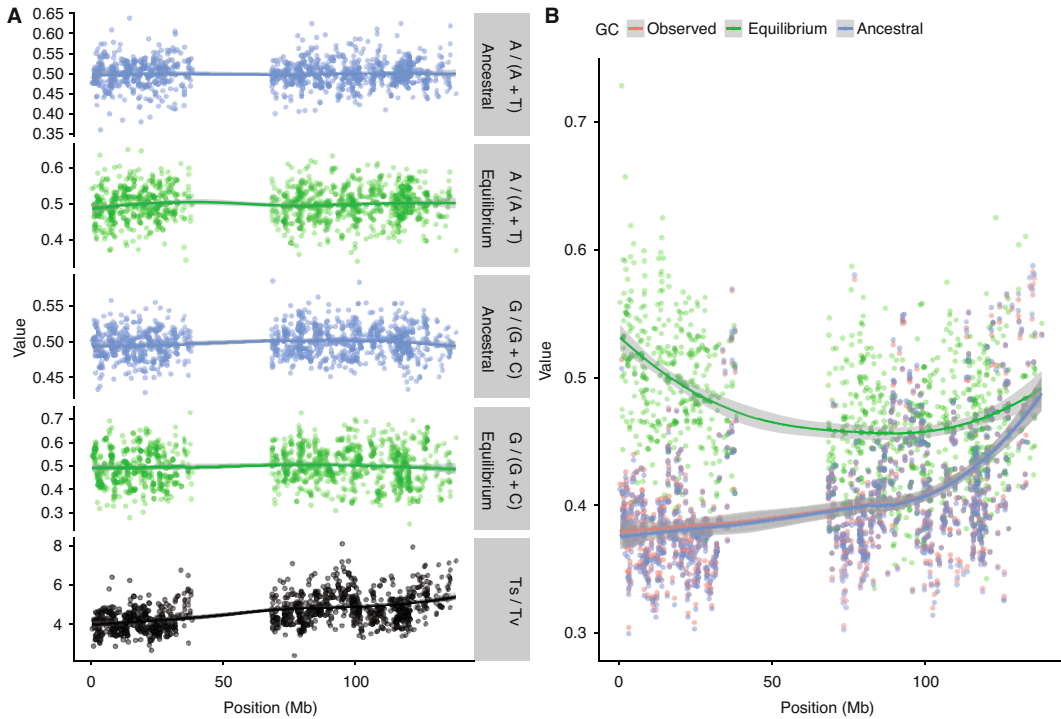


Fig. 5 Characteristics of the substitution process along chromosome 9 of great apes. **(A)** Nucleotide frequencies and substitution rates. Ts: transitions. Tv: transversions. **(B)** GC content. The smooth lines were computed using a generalized additive model (GAM)

[18]. The two last topologies, E and F, are supported by three and four windows and group humans with bonobos and humans with chimpanzees, respectively, revealing incomplete lineage sorting in the common ancestor of bonobos and chimpanzees [19].

Having inferred the underlying genealogy for each window, we could fit a model of sequence evolution and estimate parameters related to the underlying substitution process. We find that the proportions of A vs. T and G vs. C nucleotide are constant over the chromosome and equals $(A/(A + T) \simeq G/(G + C) \simeq 0.5)$. The ratio of transitions over transversions increases along the chromosome, from ~ 4 on the left end to ~ 5 on the right end (see Fig. 5A). Observed GC content is highly conserved between all species (see companion material), and increases at the right end of the chromosome (ancestral GC, see Fig. 5B). Equilibrium GC content, on the other hand, is higher in the two telomeric regions, mirroring the recombination rate. Such relationships between recombination and equilibrium GC content are expected when GC-biased gene conversion is occurring [20]. In the online companion material, an extended version of this script is provided, which removes protein-coding regions in addition to filtering the alignment. This increases the total execution time but does not significantly affect the results.

4.3 Example

Analysis 3: Running External Software

MafFilter can integrate external tools within its analysis pipeline. Programs can be run on each alignment block, and their result parsed and further processed. Two types are currently supported, based on the nature of the output.

The **SystemCall** filter exports each block as a standard alignment file, runs a program generating a new alignment, and subsequently replaces the original alignment block with the new alignment. This procedure allows improving the genome alignment by running standard gene alignment programs on syntenic blocks. The following script demonstrates this ability using the MAFFT aligner [21] on the Primates chromosome 9 alignment:

```

1 DATA=chr9
2 SPECIES=(hg38, panPan1, panTro4, gorGor3, ponAbe2)
3
4 input.file=./Primates/${DATA}.maf.gz
5 input.file.compression=gzip
6 input.format=Maf
7 output.log=${DATA}.maffilter-realign.log
8 maf.filter= \
9   Subset ( \
10     species=${SPECIES}, \
11     strict=yes, keep=no, \
12     remove_duplicates=yes), \
13   XFullGap(species=${SPECIES}, verbose=no), \
14   SequenceStatistics ( \
15     statistics=(BlockSize, BlockLength), \
16     ref_species=hg38, \
17     file=${DATA}_subset.statistics.csv), \
18   WindowSplit ( \
19     preferred_size=10000, \
20     align=adjust, \
21     keep_small_blocks=yes), \
22   SystemCall ( \
23     name=MAFFT, \
24     input.file=blockIn.fasta, \
25     input.format=Fasta, \
26     output.file=blockOut.fasta, \
27     output.format=Fasta, \
28     call=./runMafft.sh), \
29   Merge ( \
30     species=${SPECIES}, \
31     dist_max=0, \
32     rename_chimeric_chromosomes=yes), \
33   SequenceStatistics ( \
34     statistics=(BlockSize, BlockLength), \
35     ref_species=hg38, \
36     file=${DATA}_windows_realigned.statistics.csv), \
37   Output ( \
38     file=${DATA}_realigned.maf.gz, \
39     compression=gzip, \
40     verbose=1) \

```

As in Subheading 4.2, the pipeline starts by extracting sequences for five species and removing full gap positions (lines 9–13). The `systemCall` filter runs a wrapper script named `run-Mafft.sh`, located in the current directory (lines 22–28). The script reads a file named `blockIn.fasta` and writes the realigned sequences into a new file `blockOut.fasta`, which will be parsed by `MafFilter`. It further checks that the input block has at least two sequences:

```

1  #!/bin/bash
2  if [ `grep '>' blockIn.fasta | wc -l` == "1" ];
3  then
4      #Only one sequence in block, we pass...
5      cp blockIn.fasta blockOut.fasta
6  else
7      mafft --fft --nomemsave --maxiterate 2 --thread -1 \
8          blockIn.fasta > blockOut.fasta 2> mafft.log
9  fi

```

For computational efficiency, we ensure that input alignments are no longer than 10,000 sites and split long blocks using the `Window-Split` filter (lines 18–21). The `keep_small_blocks` option is set to yes, so that smaller blocks are kept unsplit and not discarded. Realigned blocks are subsequently re-assembled using the `Merge` filter (lines 29–32). However, note that this script will typically take ca. 1 day to complete on a standard desktop computer. The final alignment is exported to a new Maf file (lines 37–40), and statistics are computed before and after realignment. The total alignment length (number of aligned positions) slightly shrinks from 89.245 to 89.223 Mb after realigning with MAFFT.

The `ExternalTreeBuilding` filter enables running an external phylogeny reconstruction software on each alignment block and import the resulting tree. As done in Subheading 4.2, we filter the alignment of chromosome 9 and reconstruct the phylogenetic tree in 10 kb windows, this time using the PhyML program [22]:

```

1  # maffilter param=MafFilter-Phylogeny.bpp
2
3  DATA=chr9_realigned
4  SPECIES=(hg38, panPan1, panTro4, gorGor3, ponAbe2)
5  WSIZE=10000
6
7  input.file=$(DATA).maf.gz
8  input.file.compression=gzip
9  input.format=Maf
10 output.log=$(DATA).maffilter-phylogeny.log
11 maf.filter= \
12   MinBlockLength(min_length=10), \
13   AlnFilter2(verbose=no, \

```

```

14  species=$(SPECIES), \
15  window.size=10, window.step=1, \
16  missing_as_gap=yes, \
17  max.gap=2, max.pos=2, relative=no, \
18  file=$(DATA).trash_aln.maf, \
19  compression=none), \
20  Merge( \
21  species=$(SPECIES), \
22  dist_max=100, \
23  rename_chimeric_chromosomes=yes), \
24  Output( \
25  file=$(DATA)_filtered.maf.gz, \
26  compression=gzip), \
27  MinBlockLength(min_length=$(WSIZE)), \
28  WindowSplit( \
29  preferred_size=$(WSIZE), \
30  align=center, \
31  keep_small_blocks=no), \
32  ExternalTreeBuilding( \
33  input.file=blockIn.phy , \
34  input.format=Phylip( \
35  order=sequential, type=extended), \
36  output.file=blockIn.phy_phyml_tree, \
37  output.format=Newick, \
38  property_name=PhyML, \
39  call=./runPhyml.sh), \
40  NewOutgroup( \
41  tree_input=PhyML, \
42  tree_output=PhyML_rooted, \
43  outgroup=ponAbe2), \
44  OutputTrees( \
45  tree=PhyML_rooted, \
46  file=$(DATA).trees.dnd, \
47  compression=none, \
48  strip_names=yes) \

```

The **ExternalTreeBuilding** filter exports the current block as an alignment file (lines 32–39) and the `runPhyml.sh` script launches `phyml`:

```

1  #! /bin/bash
2  phyml -i blockIn.phy -d nt -q -m HKY85 -f m -t e -c 4 \
3  -a e -s BEST -o tlr -b 0 > phyml.log 2> phyml.err

```

An HKY85 model of nucleotide substitutions is used with a four-class discrete gamma distribution of rate fitted to the data. The best tree from both nearest neighbor interchange (NNI) and subtree pruning and regrafting (SPR) algorithms for topology search is selected and further read by `MafFilter`. After rerooting (line 40), the trees for every block are collected and output. This pipeline produces exactly 1000 trees, compared to 883 when no realignment was performed, demonstrating that the realignment step substantially increased the quality of the alignment. Results are consistent with the BioNJ analysis, 852 blocks supporting the

well-established phylogeny of the species. 85 and 58 trees cluster human and gorilla or chimpanzee and bonobo with gorilla, respectively, consistent with the occurrence of incomplete lineage sorting. Interestingly, these analyses reveal a dissymmetry in the frequency of the two ILS topologies, the one grouping human and gorilla being more frequent than the one grouping gorilla and chimpanzee. This was previously observed [23] and shown to be due to a higher rate of sequencing errors in the chimpanzee genome [18].

4.4 Example

Analysis 4:

Coordinates

Translation from One Species to Another

Many evolutionary analyses require inter-specific comparisons. When the compared species are closely related enough, it is possible to perform a joint genome alignment in order to work with a single, common reference genome. This may not always be the preferable option, however, in particular when species are divergent and/or have undergone substantial structural variation and the patterns under study are intrinsically dependent on the genome position (e.g., linkage disequilibrium [9]). In such cases, analyses are conducted independently in each species, and coordinates are then converted into a common reference for comparison.

The `LiftOver` utility, available at UCSC, can be used to convert genome coordinates from one genome assembly to another, but should not be used to map genomes of distinct species. `MafFilter`, however, has a function allowing to perform such task, providing a genome alignment of the two species is available. Such an alignment can be obtained with software like `BlastZ` and `LastZ` [24], `TBA` [4], or `Mummer` [25]. The following example shows how to convert human gene coordinates into their gorilla homologs using `MafFilter` and the 20-way genome alignment from UCSC:

```

1 DATA=chr9
2 SPECIES=(hg38, gorGor3)
3
4 input.file=./Primates/${DATA}.maf.gz
5 input.file.compression=gzip
6 input.format=Maf
7 output.log=${DATA}.maffilter-liftover.log
8 maf.filter=
9   Subset (
10     species=${SPECIES},
11     strict=yes, keep=no,
12     remove_duplicates=yes),
13   XFullGap (species=${SPECIES}, verbose=no),
14   Merge (species=${SPECIES}),
15   LiftOver (
16     ref_species=hg38,
17     target_species=gorGor3,
18     target_closest_position=yes,
19     feature.file=./Primates/chr9.CDS1kb.gtf,
20     feature.file.compression=none,
21     feature.format=GTF,
22     file=hg38_to_gorGor3.tln,
23     compression=none)

```


The option-file starts by reading the genome alignment and specifying the path to the log file (lines 4–7). It then selects the two species to compare, as shown in Subheading 3.1. The **LiftOver** filter specifies the path towards the feature file to translate (lines 19–21), here in GTF format (**MafFilter** currently supports translation from GFF, GTF, and BedGraph files, eventually compressed). Lines 16 and 17 allow setting the reference and target species, respectively. Argument `target_closest_position` set the behavior in case the matching target position in the target genome is a gap. If set to `yes`, the closest non-gap position will be returned. Original and translated positions will be returned in a tabular file, specified at lines 22 and 23. Note that for the **LiftOver** filter to work correctly, the alignment should be projected on the reference genome (in this case hg38), for instance using the `maf_project` program from the TBA package. Besides, feature coordinates will only be translated if they are wholly contained in an alignment block, that is if the feature does not overlap with a synteny break. It is therefore essential, for optimal efficiency, that the alignment blocks reflect the synteny structure of the reference and target species only, which will be the case if the two species have been pairwise aligned. When the two species are from a multiple genome alignment, the **Subset** and **Merge** filters should be used to combine syntenic blocks.

The output file recalls the query coordinates and their translation, for the features that could be translated. It is often convenient to merge this translation file with the original query, which can be done in R:

```

1 anno <- read.table("../Primates/chr9.CDS1kb.gtf",
2                   sep = "\t")
3 tln <- read.table("hg38_to_gorGor3.tln", header = TRUE)
4 tln$begin.ref <- tln$begin.ref + 1
5 tln$begin.target <- tln$begin.target + 1
6 anno2 <- merge(anno, tln, by.x = c(1,4,5),
7              by.y = c(1,3,4), all = TRUE)

```

The first 5 lines read the original GTF file as a table. GTF annotations are 1-based, inclusive $[a, b]$, while **MafFilter** uses 0-based, exclusive coordinates $[a, b[$. GTF coordinates are automatically converted when reading the file, but **MafFilter** outputs results in its coordinate system. We convert them back at lines 4 and 5, before merging the two tables, lines 6 and 7. Furthermore, the strand column in the translation table does not match the strand column in the input GTF file. In the feature file, this column indicates on which strand the feature is to be found, information that is not used in the translation step. The “strand” column in the translation file indicates which strand of the sequence was present in the genome alignment. Since the alignment was projected on the

reference genome, the corresponding value is always positive. In the target genome, the value will be positive if the genomes are colinear, and negative in the case of a genomic inversion.

5 Other Useful Tools

`MafFilter` provides tools to analyze a MAF file sequentially. These tools primarily focus on processing data for statistical analyses. It has a limited formatting capacity, in particular when long-range operations are involved, such as reordering alignment blocks. The `TBA` [4] and `Last` [6] packages contain several useful tools for that purpose, which can be used in combination with `MafFilter`.

From the `TBA` package:

- The `maf_order` program permits to select and order sequence according to their species names.
- The `maf_project` program order alignment blocks according to a reference genome. Blocks where the reference genome is on the negative strand will be reversed. All blocks that do not contain the reference species will be discarded.

From the `Last` package:

- The `maf-join` program allows combining several (sorted) multiple alignments.
- The `maf-sort` program permits to sort alignments according to sequence names.

6 Conclusion

The `MafFilter` program allows to efficiently process multiple genome alignment files, by sequentially analyzing syntenic blocks. It features a flexible and extensible syntax permitting the design of reproducible pipelines for the post-processing of genome data. Beyond filtering and quality assessment, `MafFilter` can be used to analyze patterns of diversity along genomes, within and between species.

7 Note

Note 1: pseudo-genomes

1. Pseudo-genomes are obtained by applying a set of inferred variants to the corresponding reference genome. All positions for which a variant could not be called (whether there is one or not) will, therefore, be identical to the reference genome in the resulting pseudo-genome.

References

1. Casper J, Zweig AS, Villarreal C, Tyner C, Speir ML, Rosenbloom KR, Raney BJ, Lee CM, Lee BT, Karolchik D, Hinrichs AS, Haussler M, Guruvadoo L, Navarro Gonzalez J, Gibson D, Fiddes IT, Eisenhart C, Diekhans M, Clawson H, Barber GP, Armstrong J, Haussler D, Kuhn RM, Kent WJ (2018) The UCSC Genome Browser database: 2018 update. *Nucleic Acids Res* 46(D1): D762–D769. <https://doi.org/10.1093/nar/gkx1020>
2. Danecek P, Auton A, Abecasis G, Albers CA, Banks E, DePristo MA, Handsaker RE, Lunter G, Marth GT, Sherry ST, McVean G, Durbin R, 1000 Genomes Project Analysis Group (2011) The variant call format and VCFtools. *Bioinformatics* 27(15):2156–2158. <https://doi.org/10.1093/bioinformatics/btr330>
3. Chang CC, Chow CC, Tellier LC, Vattikuti S, Purcell SM, Lee JJ (2015) Second-generation PLINK: rising to the challenge of larger and richer datasets. *Gigascience* 4:7. <https://doi.org/10.1186/s13742-015-0047-8>
4. Blanchette M, Kent WJ, Riemer C, Elnitski L, Smit AFA, Roskin KM, Baertsch R, Rosenbloom K, Clawson H, Green ED, Haussler D, Miller W (2004) Aligning multiple genomic sequences with the threaded blockset aligner. *Genome Res* 14(4):708–715. <https://doi.org/10.1101/gr.1933104>
5. Angiuoli SV, Salzberg SL (2011) Mugsy: fast multiple alignment of closely related whole genomes. *Bioinformatics* 27(3):334–342. <https://doi.org/10.1093/bioinformatics/btq665>
6. Kielbasa SM, Wan R, Sato K, Horton P, Frith MC (2011) Adaptive seeds tame genomic sequence comparison. *Genome Res* 21(3):487–493. <https://doi.org/10.1101/gr.113985.110>
7. Darling AE, Mau B, Perna NT (2010) progressiveMauve: multiple genome alignment with gene gain, loss and rearrangement. *PLoS ONE* 5(6):e111147. <https://doi.org/10.1371/journal.pone.0011147>
8. Stukenbrock EH, Christiansen FB, Hansen TT, Dutheil JY, Schierup MH (2012) Fusion of two divergent fungal individuals led to the recent emergence of a unique widespread pathogen species. *Proc Natl Acad Sci USA* 109(27):10954–10959. <https://doi.org/10.1073/pnas.1201403109>
9. Stukenbrock EH, Dutheil JY (2018) Fine-scale recombination maps of fungal plant pathogens reveal dynamic recombination landscapes and intragenic hotspots. *Genetics* 208(3):1209–1229. <https://doi.org/10.1534/genetics.117.300502>
10. Lack JB, Cardeno CM, Crepeau MW, Taylor W, Corbett-Detig RB, Stevens KA, Langley CH, Pool JE (2015) The *Drosophila* genome nexus: a population genomic resource of 623 *Drosophila melanogaster* genomes, including 197 from a single ancestral range population. *Genetics* 199(4):1229–1241. <https://doi.org/10.1534/genetics.115.174664>
11. Higgins DG, Sharp PM (1988) CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene* 73(1):237–244
12. Schiffels S, Durbin R (2014) Inferring human population size and separation history from multiple genome sequences. *Nat Genet* 46(8):919–925. <https://doi.org/10.1038/ng.3015>
13. Myers S, Bottolo L, Freeman C, McVean G, Donnelly P (2005) A fine-scale map of recombination rates and hotspots across the human genome. *Science* 310(5746):321–324. <https://doi.org/10.1126/science.1117196>
14. Gascuel O (1997) BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data. *Mol Biol Evol* 14(7):685–695. <https://doi.org/10.1093/oxfordjournals.molbev.a025808>
15. Felsenstein J (1981) Evolutionary trees from DNA sequences: a maximum likelihood approach. *J Mol Evol* 17(6):368–376
16. Hasegawa M, Kishino H, Yano T (1985) Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. *J Mol Evol* 22(2):160–174
17. Hasegawa M, Kishino H, Yano T (1987) Man's place in Hominoidea as inferred from molecular clocks of DNA. *J Mol Evol* 26(1–2):132–147
18. Scally A, Dutheil JY, Hillier LW, Jordan GE, Goodhead I, Herrero J, Hobolth A, Lappalainen T, Mailund T, Marques-Bonet T, McCarthy S, Montgomery SH, Schwalie PC, Tang YA, Ward MC, Xue Y, Yngvadottir B, Alkan C, Andersen LN, Ayub Q, Ball EV, Beal K, Bradley BJ, Chen Y, Clee CM, Fitzgerald S, Graves TA, Gu Y, Heath P, Heger A, Karakoc E, Kolb-Kokocinski A, Laird GK, Lunter G, Meader S, Mort M, Mullikin JC, Munch K, O'Connor TD, Phillips AD, Prado-Martinez J, Rogers AS, Sajjadian S, Schmidt D, Shaw K, Simpson JT,

- Stenson PD, Turner DJ, Vigilant L, Vilella AJ, Whitener W, Zhu B, Cooper DN, de Jong P, Dermitzakis ET, Eichler EE, Flicek P, Goldman N, Mundy NI, Ning Z, Odom DT, Ponting CP, Quail MA, Ryder OA, Searle SM, Warren WC, Wilson RK, Schierup MH, Rogers J, Tyler-Smith C, Durbin R (2012) Insights into hominid evolution from the gorilla genome sequence. *Nature* 483 (7388):169–175. <https://doi.org/10.1038/nature10842>
19. Prüfer K, Munch K, Hellmann I, Akagi K, Miller JR, Walenz B, Koren S, Sutton G, Kodira C, Winer R, Knight JR, Mullikin JC, Meader SJ, Ponting CP, Lunter G, Higashino S, Hobolth A, Dutheil J, Karakoç E, Alkan C, Sajjadian S, Catacchio CR, Ventura M, Marques-Bonet T, Eichler EE, André C, Atencia R, Mugisha L, Junhold J, Patterson N, Siebauer M, Good JM, Fischer A, Ptak SE, Lachmann M, Symer DE, Mailund T, Schierup MH, Andrés AM, Kelso J, Pääbo S (2012) The bonobo genome compared with the chimpanzee and human genomes. *Nature* 486(7404):527–531. <https://doi.org/10.1038/nature11128>
 20. Duret L, Galtier N (2009) Biased gene conversion and the evolution of mammalian genomic landscapes. *Annu Rev Genomics Hum Genet* 10:285–311. <https://doi.org/10.1146/annurev-genom-082908-150001>
 21. Katoh K, Misawa K, Kuma K, Miyata T. (2002), MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Res* 30 (14):3059–3066
 22. Guindon S, Dufayard JF, Lefort V, Anisimova M, Hordijk W, Gascuel O (2010) New algorithms and methods to estimate maximum-likelihood phylogenies: assessing the performance of PhyML 3.0. *Syst Biol* 59 (3):307–321. <https://doi.org/10.1093/sysbio/syq010>
 23. Slatkin M, Pollack JL (2008) Subdivision in an ancestral species creates asymmetry in gene trees. *Mol Biol Evol* 25(10):2241–2246. <https://doi.org/10.1093/molbev/msn172>
 24. Schwartz S, Kent WJ, Smit A, Zhang Z, Baertsch R, Hardison RC, Haussler D, Miller W (2003) Human-mouse alignments with BLASTZ. *Genome Res* 13(1):103–107. <https://doi.org/10.1101/gr.809403>
 25. Kurtz S, Phillippy A, Delcher AL, Smoot M, Shumway M, Antonescu C, Salzberg SL (2004) Versatile and open software for comparing large genomes. *Genome Biol* 5(2):R12. <https://doi.org/10.1186/gb-2004-5-2-r12>

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

